Linux Academy

Study Guide

OpenStack Magnum Containers

Contents

About This Course

- This course is designed to give you an overview of containers within OpenStack, with a focus on the Magnum project.
- We will explain what containers are and why you want to use them, as well as use cases for containers and where containers fit in within OpenStack.
- Once you have this background information, we will explain what Magnum is and the underlying infrastructure including architecture, networking, and security. You'll also have an opportunity to build out your own cluster.

Course Prerequisites

The course assumes the student has an understanding of Linux, Docker, OpenStack, and Kubernetes. We recommend the following pre-requisites:

- Linux Essentials
- Cloud Essentials
- Docker Quick Start
- OpenStack CLI commands
- Kubernetes

What are Containers?

- Containers are isolated, portable environments where you can run applications. They include all the libraries and dependencies they need.
- Containers share system resources for access to compute, networking, and storage, along with the same OS kernel.
- Containers keep applications, runtimes, and various other services separated from each other using kernel features known as namespaces and cgroups.
- The container image allows containers to be used on any host with a modern Linux kernel. They allow for a more rapid deployment of applications than if they were packaged in a virtual machine image.

VM vs Container

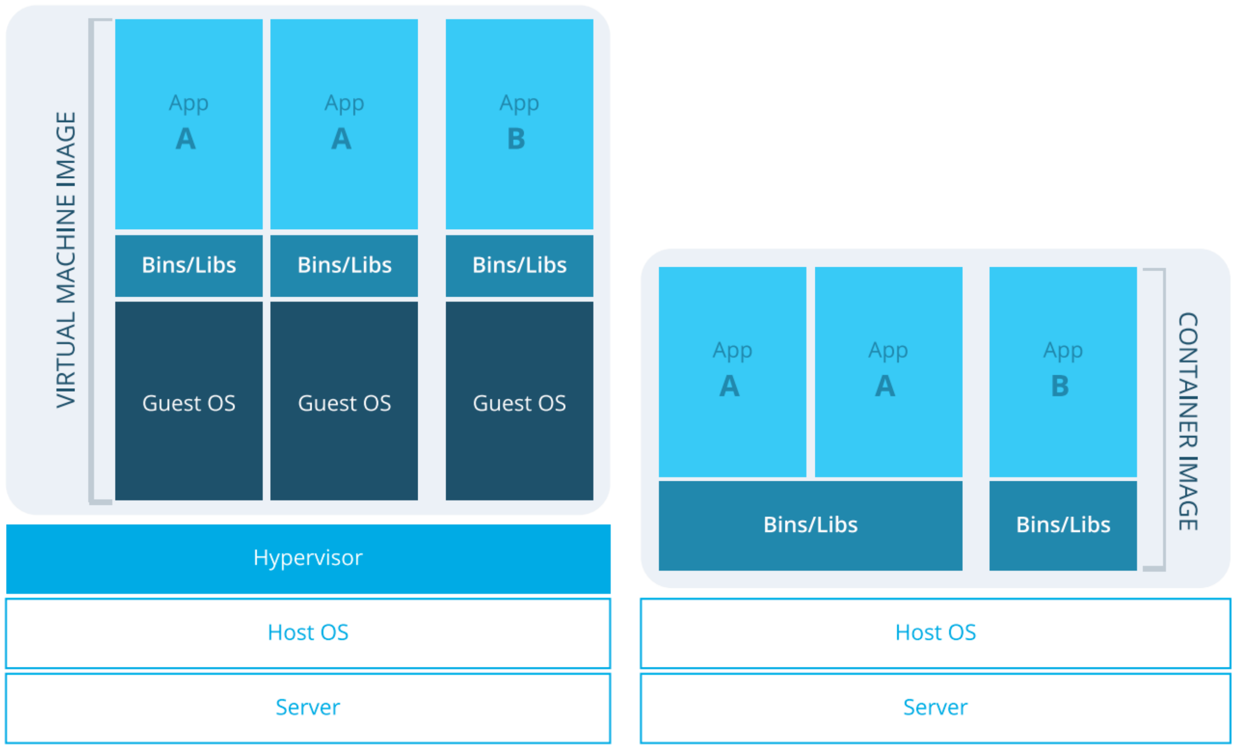


Figure 1: Containers vs. VMs

Containers provide isolation for processes while sharing compute, network, and storage resources residing on the same host. They are similar to VMs in that they share the host kernel but avoid hardware emulation.

Applications can be packaged with all needed dependencies that are not provided by the host. This is what makes containers efficient to run and easy to move from host to host all while enabling more granular control of the applications.

Why Use Containers?

Containers can be used in one of two ways:

- System or OS containers
- Application containers

As a result of having these two types of containers, administrators and developers are interested in containers for the following reasons:

- Application containers, compared with virtual machines, are very lightweight – minimizing compute, storage, and bandwidth requirements.

- Containers are portable, effectively running on any hardware that runs the relevant operating system.

More on Application Containers

- Application containers can be used to break down and isolate parts of isolated applications called microservices. This breakdown allows you to have a smaller code base and reduces the likelihood of both regression issues and accidentally pushing a feature. It also allows for increased stability as microservices can be spread across multiple machines.
- Microservices allow for more granular scaling, simplified management, and improved security configurations. You can scale the parts of your application that are used more while leaving less heavily used sections alone.
- An application or service can be put inside a container, along with the runtime requisites and services the application requires, without having to include a full operating system as shown previously.

Most Common Third-Party Systems for Managing Containers

- Docker Swarm
    - Enables native clustering for Docker.
- Kubernetes
    - Developed originally by Google but is now an open source orchestration system for Docker containers. It handles scheduling on to nodes in a compute cluster and manages workloads and desired states.
- Apache Mesos
    - Developed by Apache, it can be used to deploy and manage application containers in large-scale clustered environments.

OpenStack refers to these three options as Container Orchestration Engines (COE). All three of these COE systems are supported in OpenStack Magnum, the containers service for OpenStack.

Where Containers Fit in OpenStack

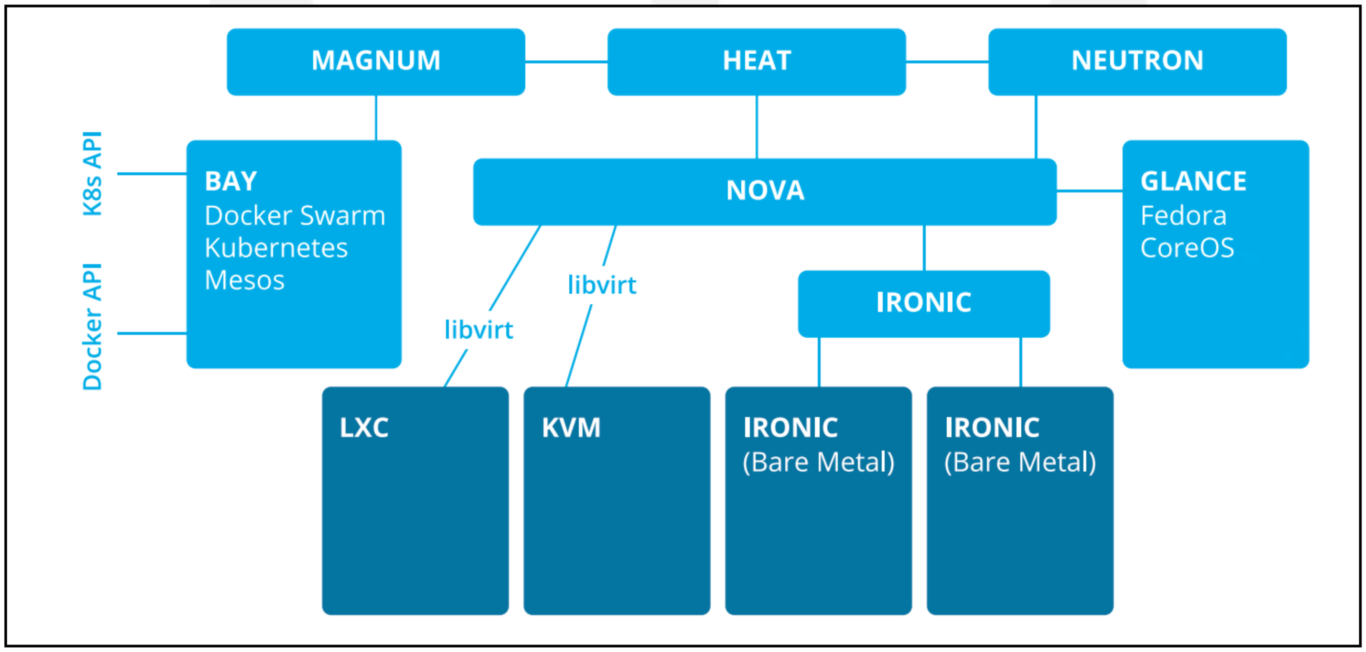
Cloud infrastructure provides a complete data center management solution in which containers, or hypervisors for that matter, are only part of a much bigger system.

- Containers provide deterministic software packaging and fit nicely with an immutable infrastructure model.
- Containers are excellent for encapsulation of microservices.
- Containers add additional portability within OpenStack on virtual machines, as well as bare metal servers (Ironic), using a single, lightweight image.
- One of the benefits of using an orchestration framework with containers is that it can allow switching between OpenStack or bare metal environments at any given point in time, abstracting the application away from the infrastructure.

OpenStack Projects That Work with Containers

- **Nova** • Nova contains a Docker hypervisor driver for Nova Compute to treat containers and images as the same type of resource as virtual machines.
- **Heat** • Heat contains a plugin template for orchestrating Docker resources on top of OpenStack resources. Allows access to full Docker API.
- **Magnum** • Magnum provides an API to manage multi-tenant Containers-as-a-Service, leveraging Heat, Nova, and Neutron.
- **Kolla** • Kolla containerizes the OpenStack control services themselves as microservices to simplify the operational experience
- **Murano** • Murano provides an application catalog of containerized applications that can be deployed to an OpenStack cloud.

Where Magnum Fits In

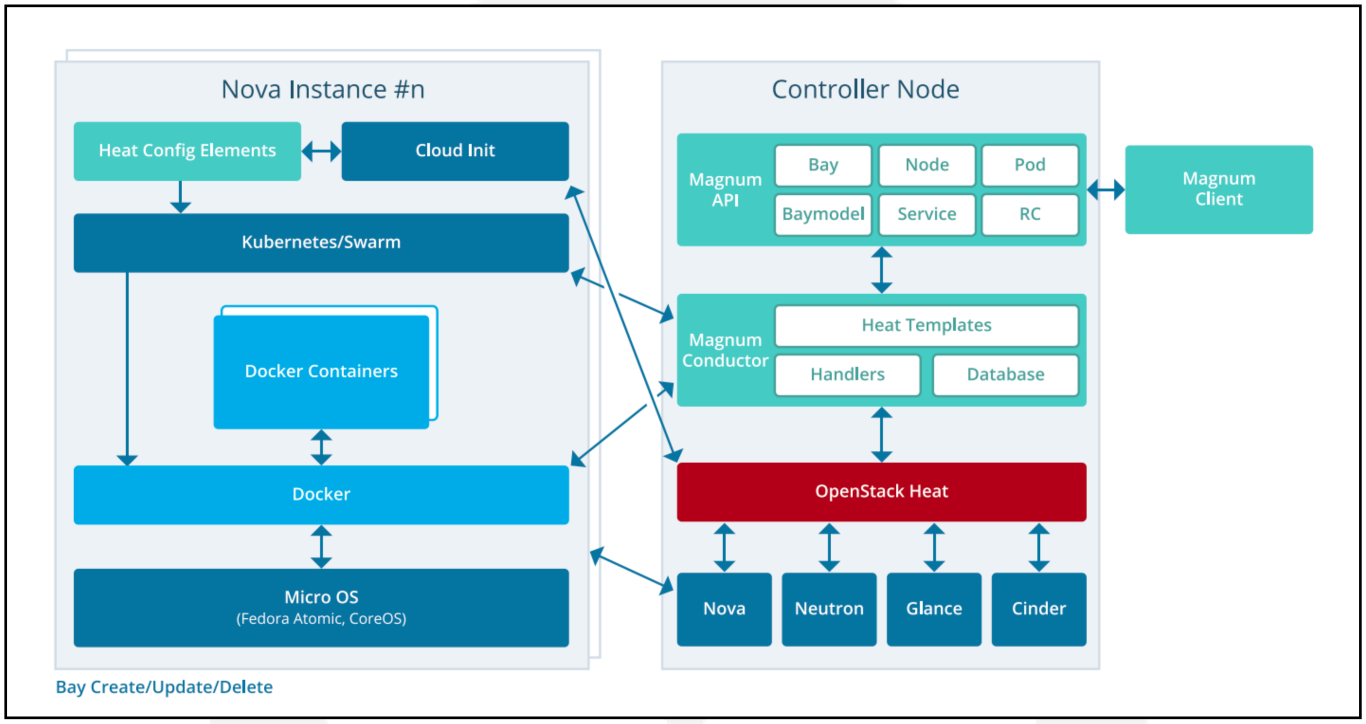


- In November 2013, 101 Engineers with 28 different affiliations came together to develop OpenStack Magnum, a multi-tenant Containers-as-a-Service platform, combining the best of Infrastructure software with the best of container software.

- Magnum was created to make current container technology and tools work with OpenStack and allows cloud operators to offer Containers-as-a-Service as a managed hosting service.

- In the same way that Nova provides plugability for hypervisors, Magnum does the same for containers.

More In-Depth View of Magnum in OpenStack



- Magnum containers are run on top of an OpenStack resource referred to as a cluster. Clusters are collections of Nova instances that are created using Heat.

- Magnum uses Heat to orchestrate an OS image which uses Docker Swarm, Kubernetes, or Apache Mesos, and runs that image in either virtual machines or bare metal in a cluster configuration.

- The cluster, previously known as a bay, acts as a security barrier between instances.

- Magnum simplifies the required integration with OpenStack and allows cloud users who can already launch cloud resources, such as Nova instances, Cinder volumes, or Trove databases, to create clusters where they can start application containers.

Magnum Terminology

- **Cluster** • A cluster (previously known as a bay) is an instance of the ClusterTemplate of a

Container Orchestration Engine. The construct in which Magnum launches container orchestration engines. After a cluster has been created, the user is able to add containers to it either directly, or in the case of the Kubernetes container orchestration engine, within pods - a logical construct specific to that implementation. A cluster is created based on a ClusterTemplate, along with a few additional parameters for the cluster.

- Magnum deploys the orchestration templates provided by the cluster driver to create and configure all the necessary infrastructure.

- **ClusterTemplate** • A ClusterTemplate, previously known as a BayModel, is roughly the equivalent to a flavor in Nova. It acts as a template that defines options, such as the container orchestration engine and the keypair and image used when Magnum is creating clusters using the given ClusterTemplate.

  - A cloud provider can also define a number of ClusterTemplates and provide them to users.
  - A ClusterTemplate cannot be updated or deleted if a cluster using this ClusterTemplate exists.
  - Access to the ClusterTemplate is by default limited to the administrator, owner, or users within the same tenant as the owners unless the `--public` flag is used.

- **Container Orchestration Engine** • A container orchestration engine manages the lifecycle of one or more containers, logically represented in Magnum as a cluster. Magnum supports a variety of container orchestration engines and allows more to be added over time. Given several choices, users can run one or more clusters concurrently, and each can use a different container orchestration engine.

  - Magnum has been tested with the fedora-atomic micro-OS and CoreOS. It will likely work with other Micro-OS images, but each image requires individual support in the Heat template.

Building a Cluster with Kubernetes

Create a Keypair for Use With the ClusterTemplate

```
ssh-keygen
openstack keypair create --pub-key ~/.ssh/id_rsa.pub [NAME]
```

Verify That Your DNS Server Can Resolve Hostnames

Example:

```
dig www.openstack.org @8.8.8.8 +short
```

Create a ClusterTemplate

- Identify the image, flavor, and public network you want to use in your ClusterTemplate
- Identify the image you want to use for your ClusterTemplate

  - `openstack image list`
- Identify server flavor to use with your ClusterTemplate
  - `openstack flavor list`
- Identify Public network
  - `openstack network list`
- Create a ClusterTemplate

  -
    ```
    magnum cluster-template-create --name [TEMPLATE_NAME] --image
    [IMAGE] --keypair [KEYPAIR_NAME] --external-network [NETWORK] \
    dns-nameserver [NAMESERVER] --flavor [FLAVOR] \
    --docker-volume-size [SIZE] --network-driver [DRIVER] --coe [COE]
    ```

Create a Cluster

```
magnum cluster-create --name [CLUSTER_NAME] --cluster-template \
[TEMPLATE_NAME] --node-count [CLUSTER_NODE_COUNT]
```

View Cluster Status

- Print a list of all clusters
  - `magnum cluster list`
- View details on specific cluster
  - `magnum cluster-show [name | UUID]`

Delete a Magnum cluster

```
magnum cluster-delete [Name | UUID]
```

Use Cases for Containers in OpenStack

- Developers can create an application container, containing the app, runtimes, libraries, etc., and move it to any machine - physical or virtual.
- In CI/CD environments, containers enable organizations to rapidly test more system permutations.
- For quality assurance, containers enable better black box testing as well as help organizations shift from governance to compliance.
- Containers can be created using a single consistent container image, and changes can immediately be layered upon all the container instances.

- Containers can be run on different underlying hardware, so if one host goes down, administrators can route traffic to live application containers running elsewhere.