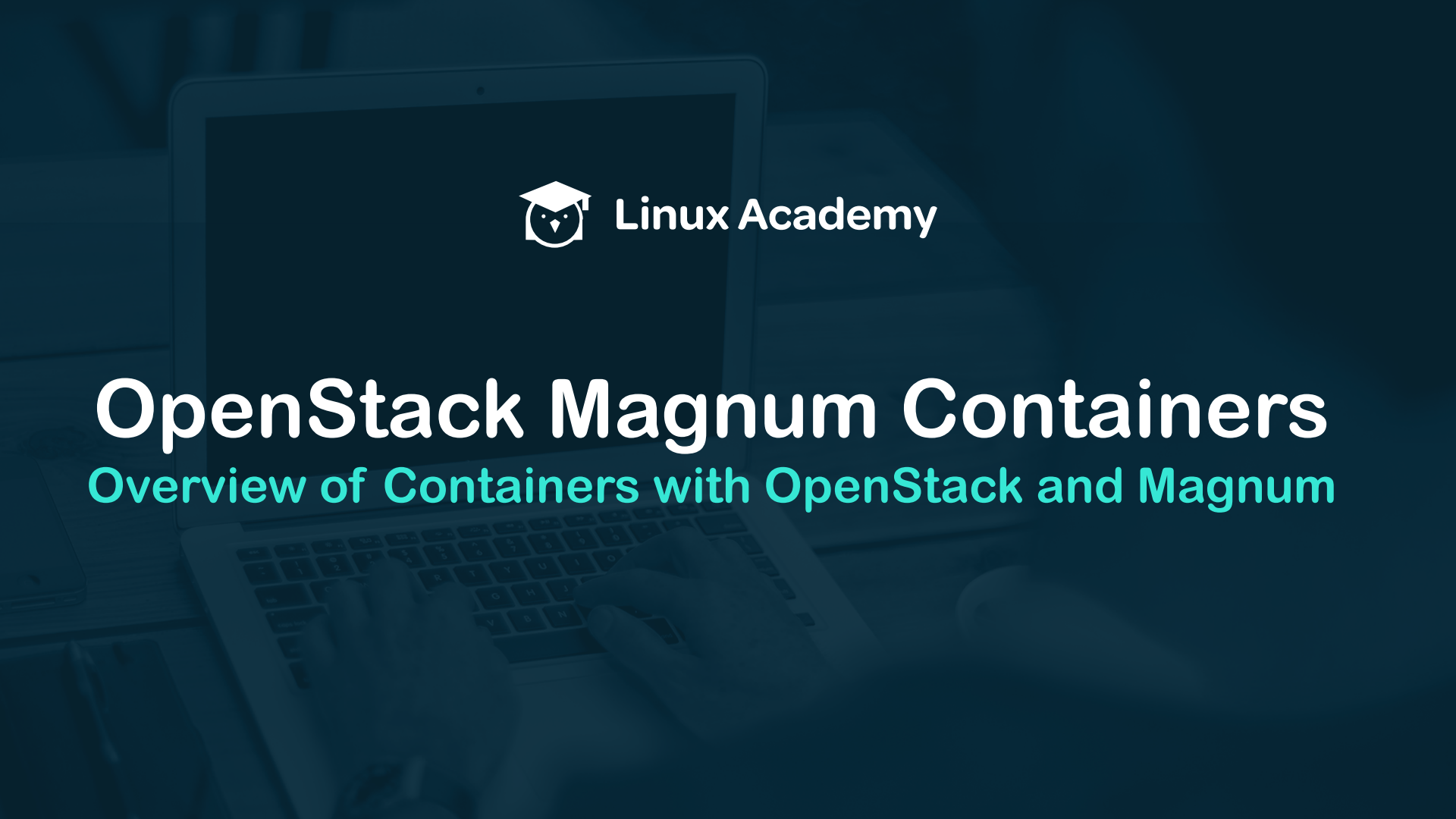


OpenStack Magnum Containers

Overview of Containers with OpenStack and Magnum

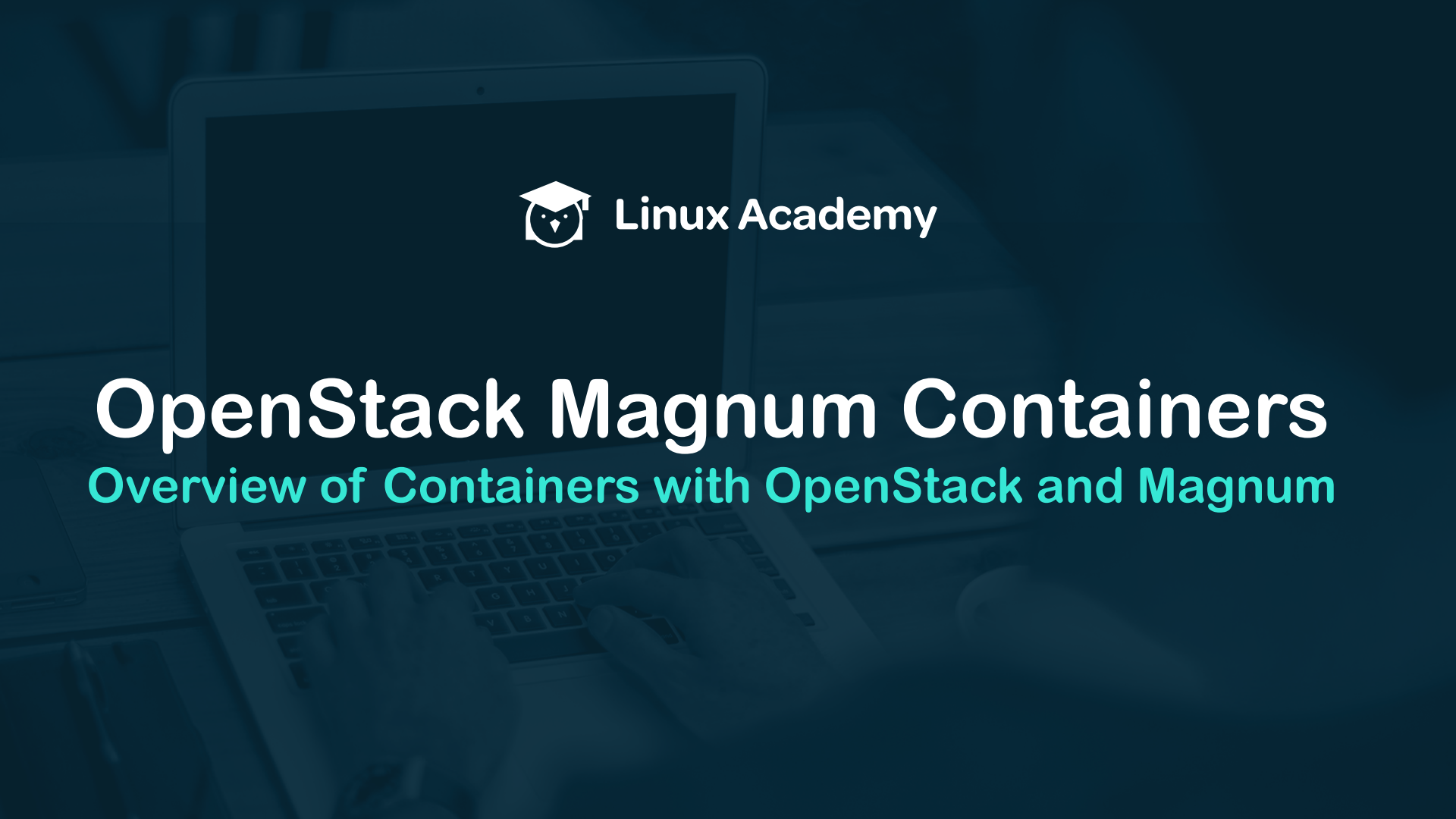
About This Course

- 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.

- Linux Essentials
- Cloud Essentials
- Docker Quickstart
- OpenStack CLI Commands
- Kubernetes



OpenStack Magnum Containers

Overview of Containers with OpenStack and Magnum

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 more rapid deployment of applications than if they were packaged in a virtual machine image.

What are Containers?

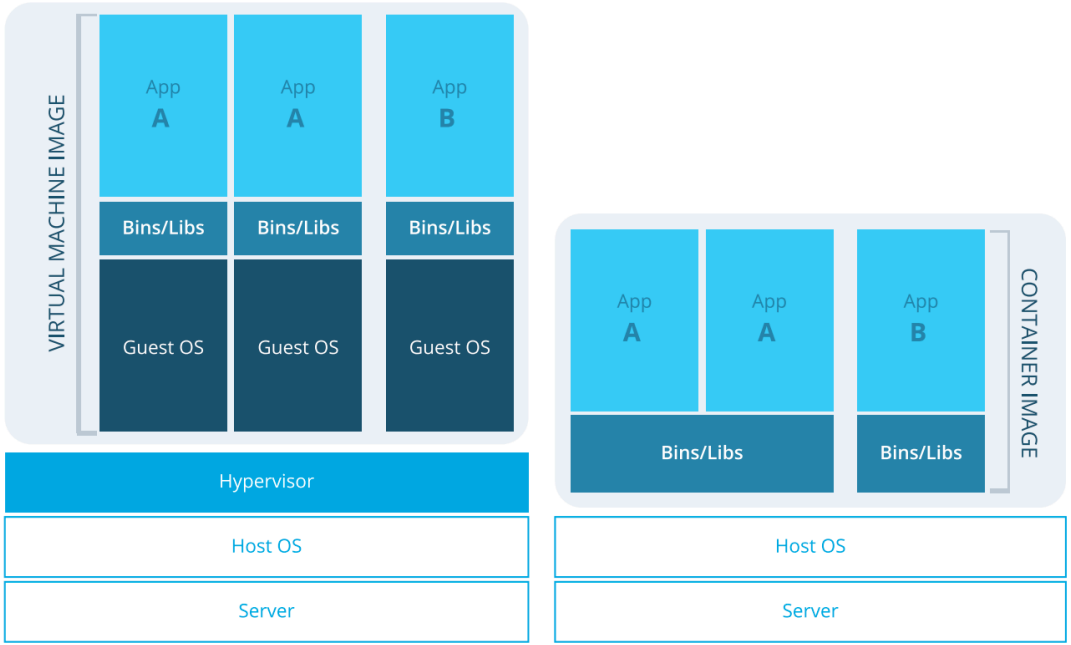
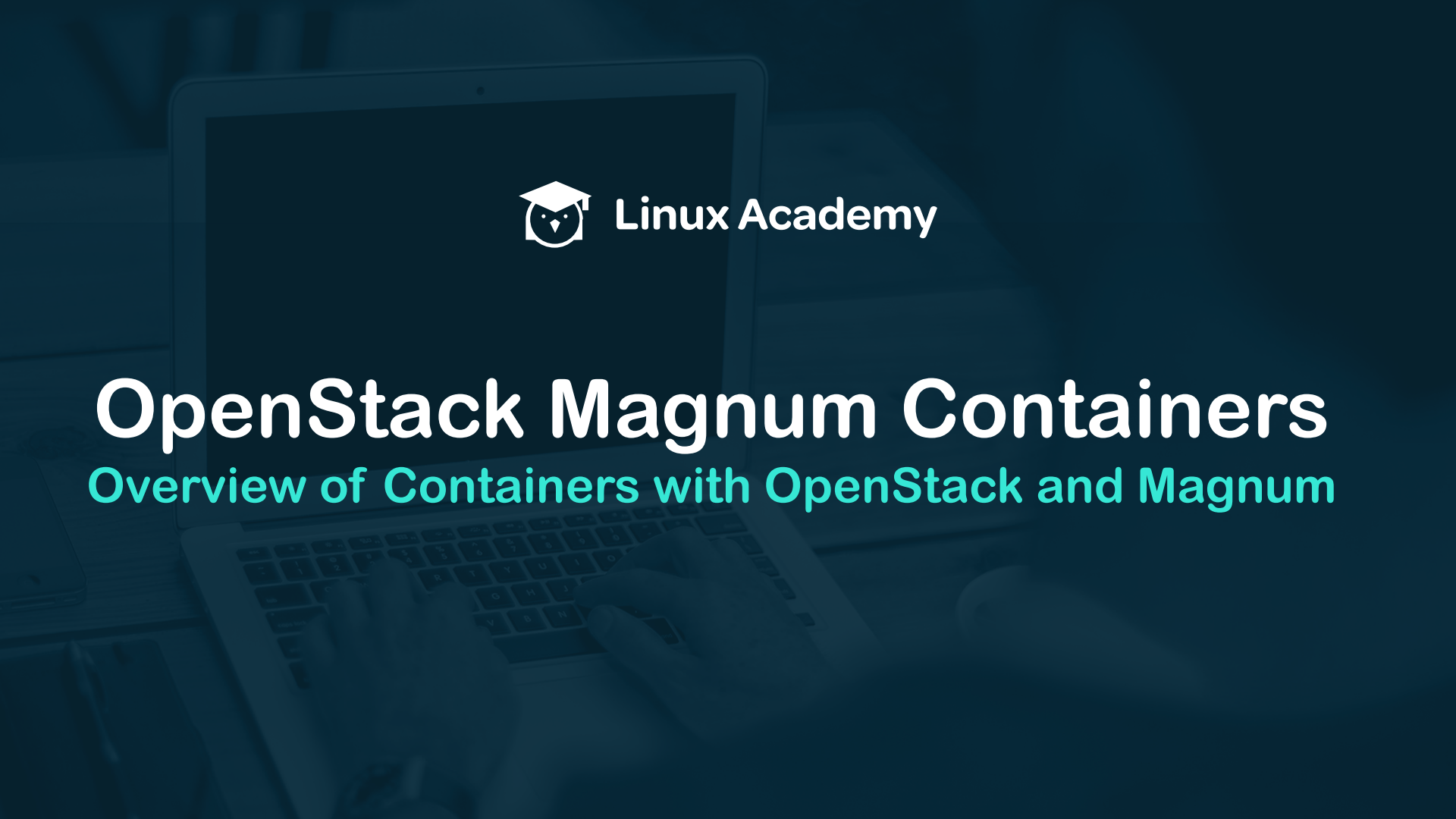


Figure 1: Containers vs. VMs



OpenStack Magnum Containers

Overview of Containers with OpenStack and Magnum

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:
  - 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.

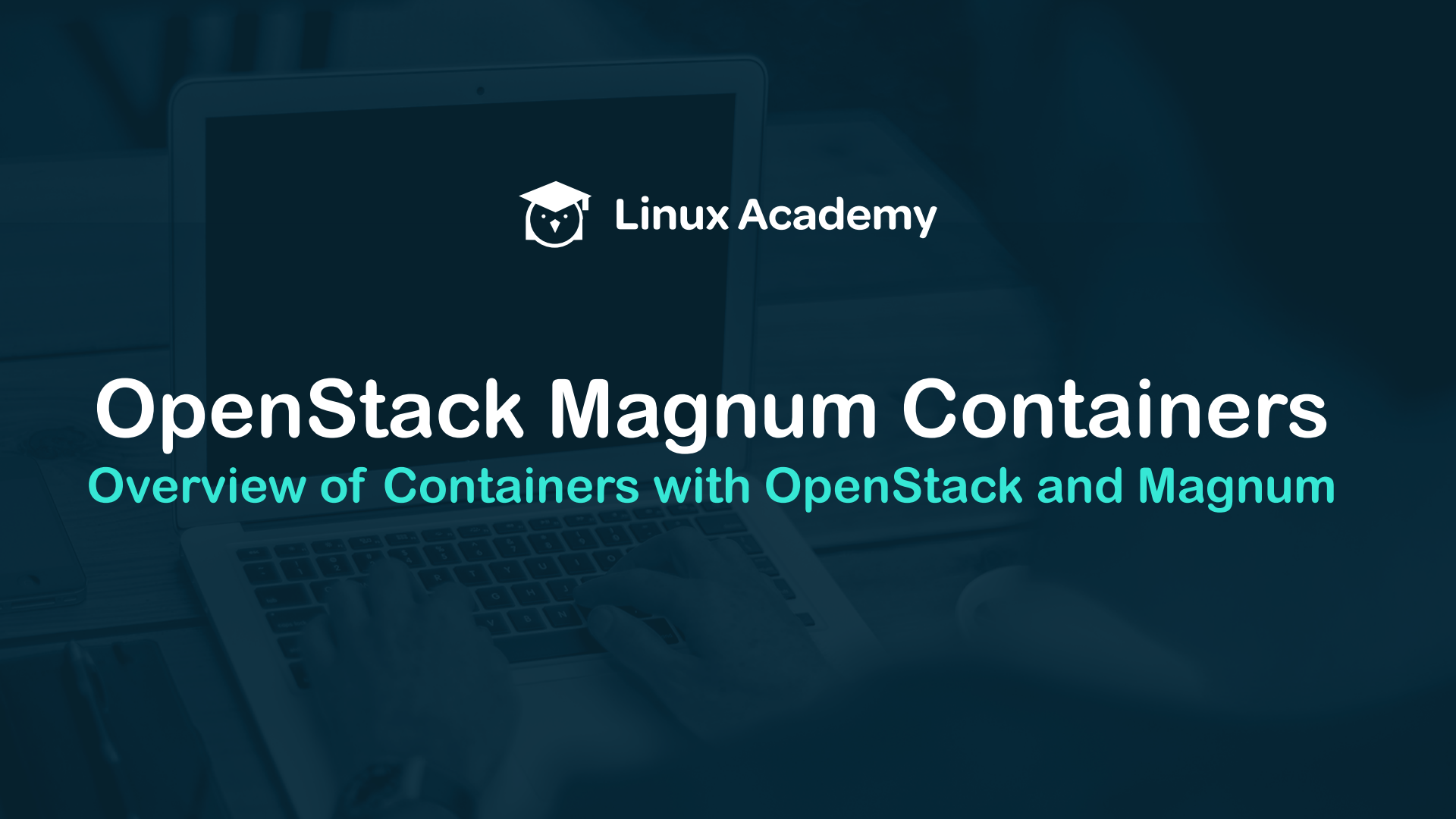
More on Application Containers

- 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 accidently 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. So 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.

Most Common Third-Party Systems for Managing Containers

- Docker Swarm
- Kubernetes
- Apache Mesos

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.



OpenStack Magnum Containers

Overview of Containers with OpenStack and Magnum

Where Containers Fit in OpenStack

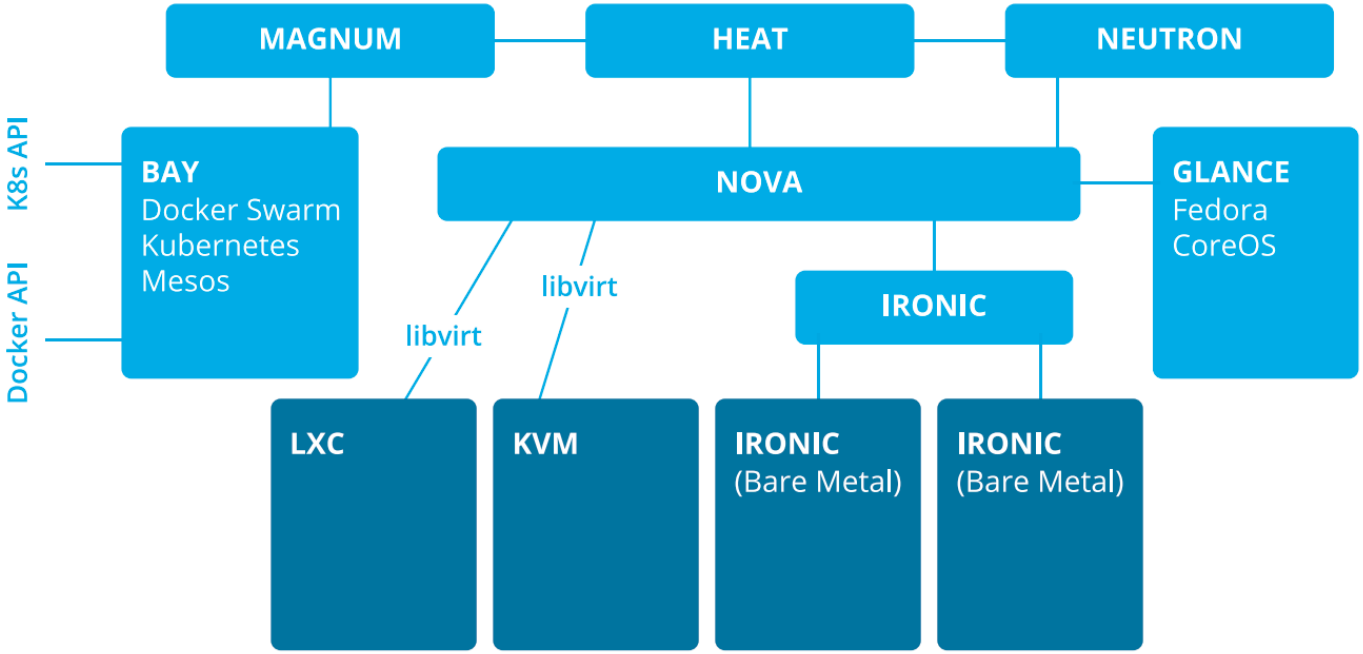
Cloud infrastructure provides a complete data center management solution in which containers, or hypervisors for that matter, are only part of a much bigger system.

- 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.

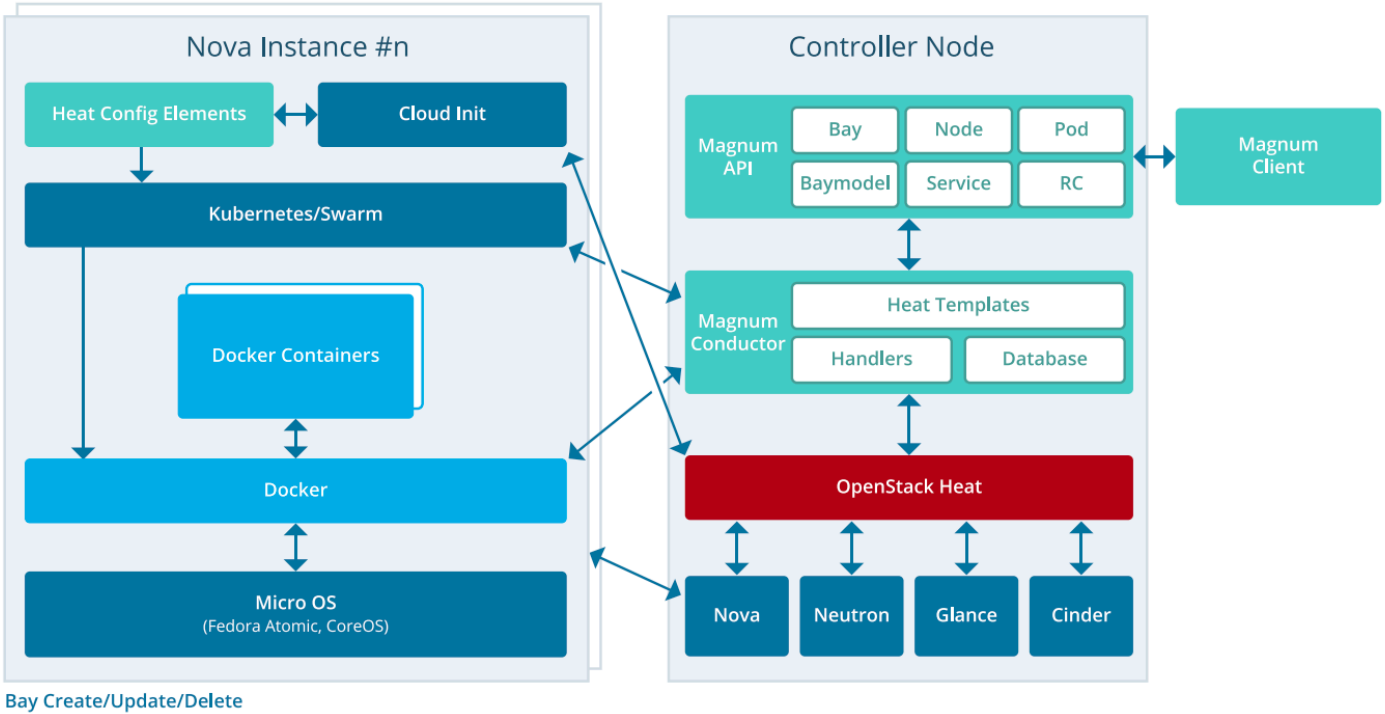
OpenStack Projects that Work with Containers

- 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



More In-depth View of Magnum in OpenStack



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.

Openstack Magnum Containers

Overview of Related Technologies

What is Virtualization?

- OS virtualization is the process of creating a software-based (or virtual) representation of something instead of a physical one.

- Virtualization describes a technology in which an application, guest operating system or data storage is abstracted away from the underlying hardware or software.
  - Application virtualization
  - Server virtualization
  - Storage virtualization
  - Network virtualization
  - Desktop virtualization
  - Data virtualization

Virtualization Types

- Hardware virtualization, also known as platform virtualization, is the creation of a virtual machine that behaves like a real computer with an operating system.

- Software executed on these VMs is separated from the underlying hardware resources.

    - **Full virtualization** – An almost complete simulation of the actual hardware to allow software, which typically consists of a guest OS, to run unmodified.
    - **Partial virtualization** – Some but not all of the target environment attributes are simulated. As result, some guest programs may need modifications to run in such a virtual environments.
    - **Paravirtualization** – A hardware environment that is not simulated, but the guest programs are executed in their own isolated domains as if they are running on a separate system.

Virtualization Types

- OS-level virtualization is a server virtualization method in which the kernel of an operating system allows the existence of multiple isolated user-space instances.
- These instances, sometimes referred to as containers, software containers, virtualization engines, or chroot jails, run on top of an existing host operating system and provide a set of libraries that applications can interact with, giving the illusion of running on a dedicated machine.
- The host OS constrains the container's access to physical resources, such as CPU and memory, so a single container cannot consume all of a host's physical resources.

LXC

- LXC (Linux containers) is an OS-level virtualization method for running multiple isolated Linux systems on a control host using a single Linux kernel.

- The Linux kernel cgroups functionality, released in version 2.6.24, allows limitation and prioritization of resources like CPU, RAM, block I/O, and networking, without the need for starting virtual machines, along with namespace isolation functionality that allows complete isolation of an application's view of the operating environment.
  - Includes process trees, networking, user IDs, and mounted file systems.

- Combines the kernel cgroups and support for isolated namespaces to provide an isolated environment for applications.

- Docker can also use LXC as one of it's execution drivers, enabling image management and providing deployment services.

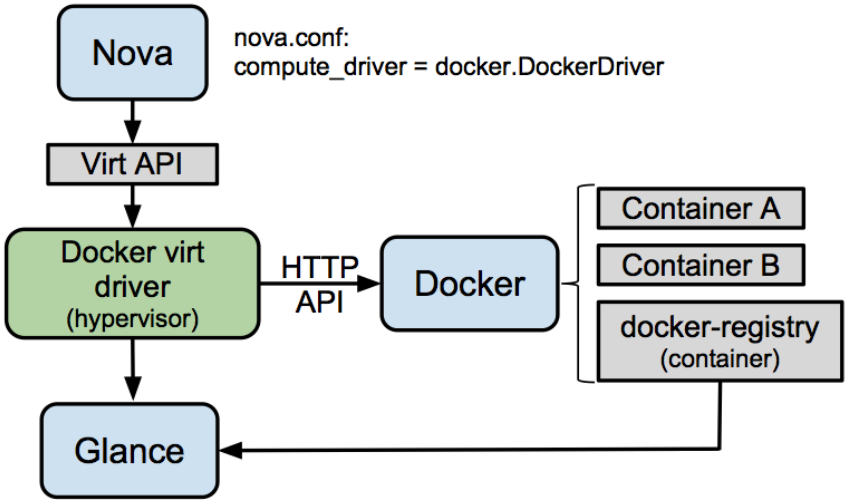
OpenStack Containers Project

- **Nova Docker** – A Docker hypervisor driver for Nova Compute to treat containers and images as the same type of resource as a virtual machine.
- **Heat Docker** – A plugin template orchestrating Docker resources on top of OpenStack resources. Allows access to full Docker API.
- **Murano** – Provides an application catalog of containerized applications that can be deployed to an OpenStack cloud.
- **Solum** – An OpenStack project designed to make cloud services easier to consume and integrate into the application development process.
- **Zun** – Container service for OpenStack. Provides APIs for launching and managing containers backed by different container technologies.

Nova Docker

- Nova Docker is a Docker hypervisor driver for Nova compute to treat containers and images as the same type of resource as a virtual machine.
- The Compute driver is pointed to the Docker driver
- The Nova Docker virtualization driver talks to the Docker agent using HTTP API calls
- Docker images are stored in the Docker registry and images are exported to Glance from the Docker registry, which Nova uses to create containers.
- The nova-docker project has reached EOL and is no longer maintained. It has been replaced with OpenStack Zun.

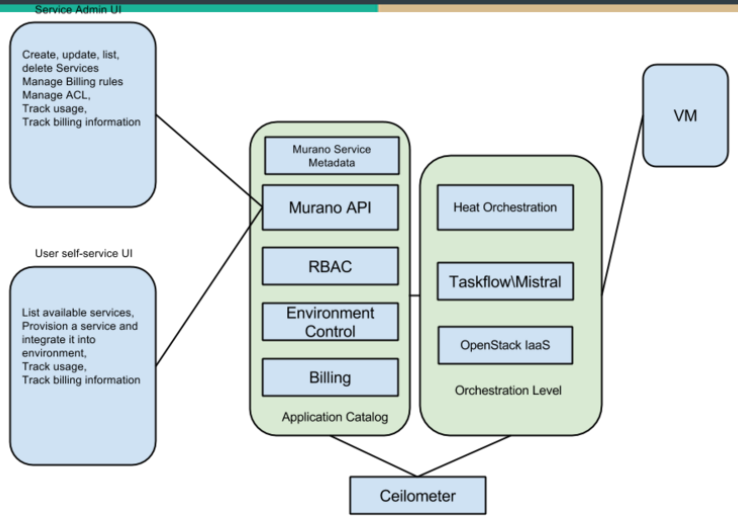
Heat Docker



- A plugin template orchestrating Docker resources on top of OpenStack resources, making it compatible with existing OpenStack clouds. Heat Docker allows access to the full Docker API.

- The Docker plugin allows Heat to “talk” directly to Docker.

Murano



- Provides an application catalog of containerized applications that can be deployed to an OpenStack cloud, enabling application developers and cloud administrators to publish various cloud-ready applications in a browsable and categorized catalog.

- Murano provides a user interface and API that allows users to compose and deploy composite environments on the application abstraction level, then manage their lifecycle.

Solum

- An OpenStack project designed to make cloud services easier to consume and integrate into your application development process.
- Provides an easy to use platform for building, testing, and deploying applications on OpenStack for developers.
- Provides a declarative model for application developers to deploy and run their applications on OpenStack starting from the app's source code.
- Multiple language run-time environments are supported with the modular "languagepack" solution

Zun

- A container service for OpenStack previously known as Higgins, provides APIs for launching and managing containers backed by different container technologies.
- Created for users who want to create and manage containers as OpenStack-managed resource. Containers managed by Zun should be well integrated with other OpenStack resources, such as Neutron network and Cinder volume.
- While Magnum provides APIs to provision container orchestration engines, Zun aims to provide APIs to manage containers themselves.
- Users are provided with simplified APIs to manage containers without the need to explore the complexities of different container technologies.

OpenStack Magnum Containers

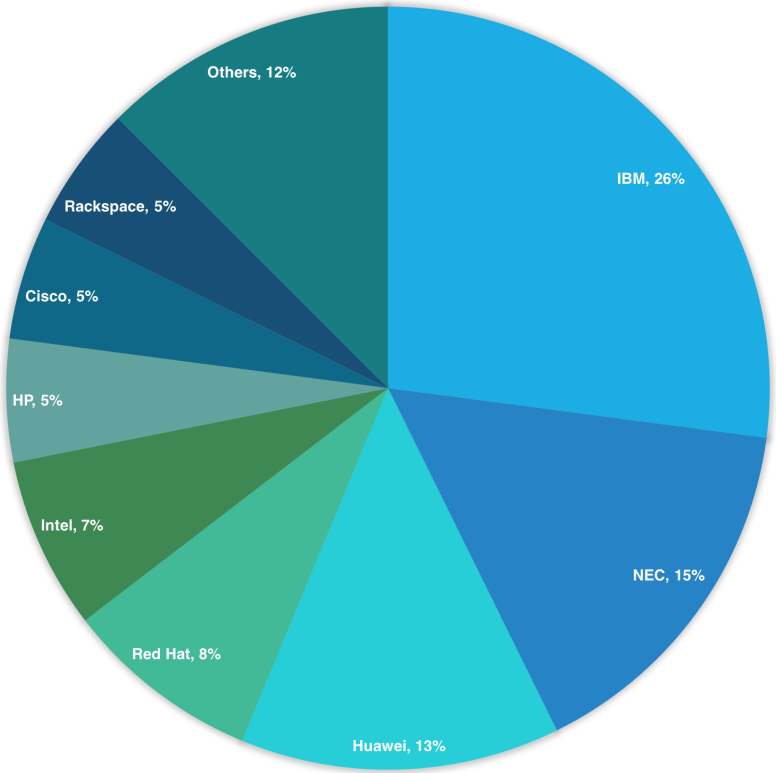
What is Magnum?

What is Magnum?

- OpenStack Magnum is a multi-tenant Containers-as-a-Service 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.

Magnum Development

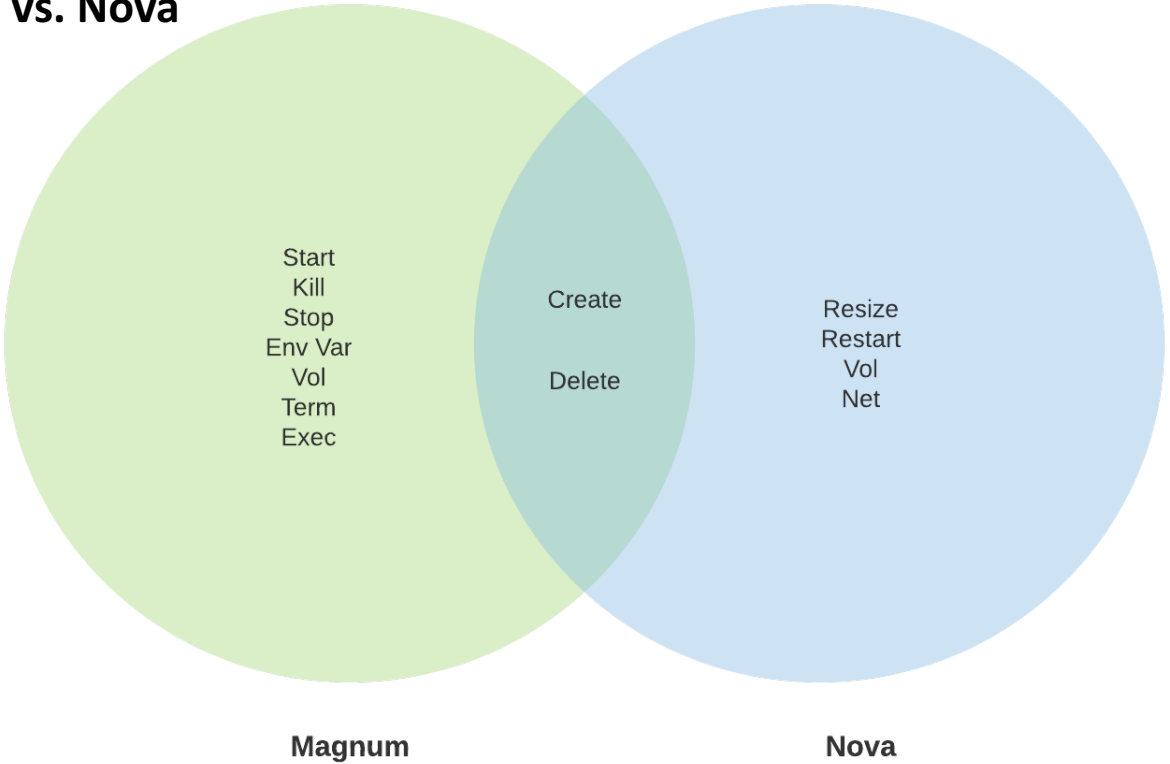
- OpenStack Magnum development started in November 2013
- A collaboration between 101 Engineers with 28 different affiliations
- The first iteration was released January 20th, 2015, three months into the Liberty release of OpenStack
- During the OpenStack Summit Austin 2016, it was decided that Magnum would primarily focus on container infrastructure, while other products would be used for container management.


Others, 12%
IBM, 26%
Rackspace, 5%
Cisco, 5%
HP, 5%
Intel, 7%
NEC, 15%
Red Hat, 8%
Huawei, 13%

Why Use Magnum?

- Provisioning
  - Provides a service to provision different COEs per pod.
- Configuration
  - Supports several container, storage, and networking options
- Scale
  - Can easily be scaled up or down via Magnum API
- Minimal hard dependencies
  - Hard dependency on Barbican has been removed after feedback from users

Magnum vs. Nova



How Magnum Works

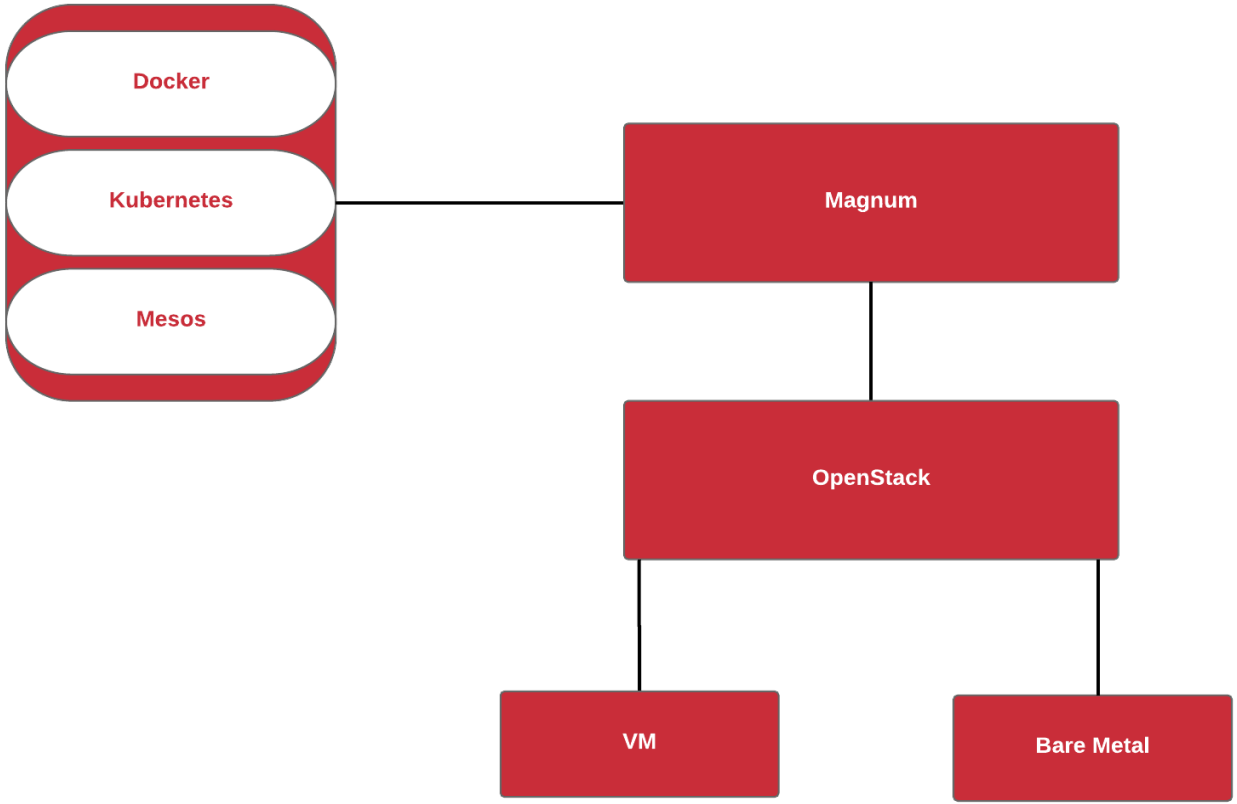
- Magnum containers are run on top of an OpenStack resource referred to as a cluster. Clusters are collections of Nova instances that are created using Heat.

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

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

Magnum API

- Two binaries work together to compose the Magnum system. The first binary is the Magnum-api REST server. The REST server may run as one process or as multiple processes.
- When a REST request is sent to the client API, the request is sent via AMQP to the magnum-conductor process. The REST server is horizontally scalable.
- At this time, the conductor is limited to one process, but horizontal scalability will be added to the conductor in the future.


Docker
Kubernetes
Mesos
Magnum
OpenStack
VM
Bare Metal

Terminology

- **Cluster** - 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.

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

- **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 number of container orchestration engines, each with their own pros and cons, including Docker Swarm, Kubernetes, and Mesos.

Cluster

- A Cluster (previously known as a bay) is an instance of the ClusterTemplate of a COE. Magnum deploys a Cluster by referring to the attributes defined in the particular 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.

- When ready, the cluster is a fully operational COE that can host containers.

ClusterTemplate

- A ClusterTemplate, previously known as a BayModel, is a collection of parameters to describe how a cluster can be constructed. Some parameters are relevant to the infrastructure of a cluster, while others are for the particular COE.

- 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 still exists.

- In typical workflow, a user creates a ClusterTemplate, then creates one or more clusters using the ClusterTemplate.

Container Orchestration Engine

- Magnum supports a variety of COE options 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 COE.
- Kubernetes
  - Image: Fedora-atomic, CoreOS
  - Volume Driver: Cinder
  - Network Driver: flannel
- Swarm
  - Image: Fedora-atomic
  - Volume Driver: Rexray
  - Network Driver: Docker, flannel
- Mesos
  - Image: Ubuntu
  - Volume Driver: Rexray
  - Network Driver: Docker

Building a Kubernetes Cluster

- Create a keypair for use with the ClusterTemplate
  - Can be done via CLI using ssh-keygen
- Verify that your DNS server can resolve hostnames properly
  - Ex: dig www.openstack.org @8.8.8.8 +short
- Create a ClusterTemplate
  - Can be done using the Magnum CLI client, or through Horizon if Magnum-UI is installed.
- Create a Cluster
  - Can be done with Magnum CLI or through Horizon if Magnum-UI is installed

Required Parameters

- The following parameters are required for creating a functional ClusterTemplate:
  - --**name** sets the name of the ClusterTemplate to create.
  - **--image** specifies the name or UUID of the base image in Glance to boot the servers for the cluster.
  - **--keypair** sets the name or UUID of the SSH keypair to configure in the cluster servers for SSH access.
  - **--external-network** sets the name or network ID of a Neutron network to provide connectivity to the external internet for the cluster.
  - **--flavor** is the Nova flavor ID for booting the node servers.
  - **--network-driver** specifies the name of the driver that will provide networks for the containers.
  - --**coe** specifies the Container Orchestration Engine to use. Supported COEs include Kubernetes, Swarm, and Mesos.

Required Parameters

- Once the ClusterTemplate has been created, you can use the command magnum cluster-create to spin up a new cluster.
  - --**name** sets the name of the Cluster.
  - --**cluster-template** specifies the ClusterTemplate to use
  - **--node-count** specifies the number of nodes to be created.
- Add additional nodes to an existing cluster with the command
  - magnum cluster-update $NAME replace node_count=$VALUE

Magnum Networking

- As of the Mitaka release, Magnum Engineers developed a Neutron plugin to integrate Magnum with OpenStack Networking.

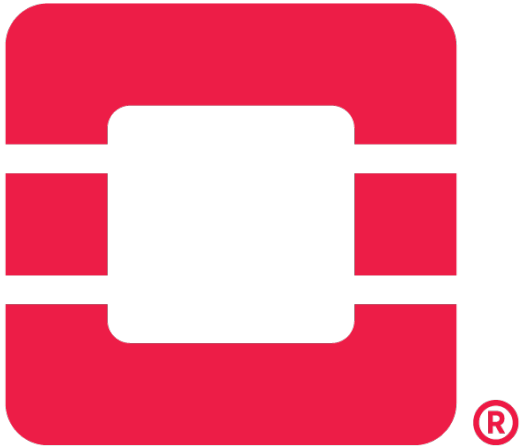
- Magnum leverages OpenStack Networking capability when creating clusters. This allows each node in a cluster to communicate with the other nodes.

- If the Kubernetes cluster type is used, a Flannel overlay network is used, which allows Kubernetes to assign IP addresses to containers in the cluster, while allowing multi-host communication between containers.

Openstack Magnum Containers

What's Next?

Course Summary



- OpenStack, above all, is an integration engine, bringing various technologies together through common APIs. With its ever-increasing popularity, containers have been plugged into several existing projects and will find their way into others.

What to Do After Taking the Course

- Start another course!
  - Running Container Clusters with Kubernetes - https://linuxacademy.com/cp/modules/view/id/80
- Deploy it yourself!
  - Ubuntu Devstack - https://linuxacademy.com/cp/modules/view/id/31
  - RDO Packstack – https://linuxacademy.com/cp/modules/view/id/28
- Learning Paths!
  - https://linuxacademy.com/cp/learningpaths
  - OpenStack Junior, Associate, and Senior level Learning Paths
- Quick Training!
  - Courses under two hours on subjects ranging from AWS to Serverless concepts.

What to Do After Taking the Course

- Get Certified!
  - OpenStack Foundation Certified OpenStack Administrator
  - Red Hat Certified OpenStack Administrator
- Join the Linux Academy Community!
  - Interact with instructors and other students
  - Opportunity to win prizes!
- Study Groups!
  - Join an existing group, or create your own
- Join the OpenStack Community!
  - https://www.openstack.org
  - #openstack on freenode