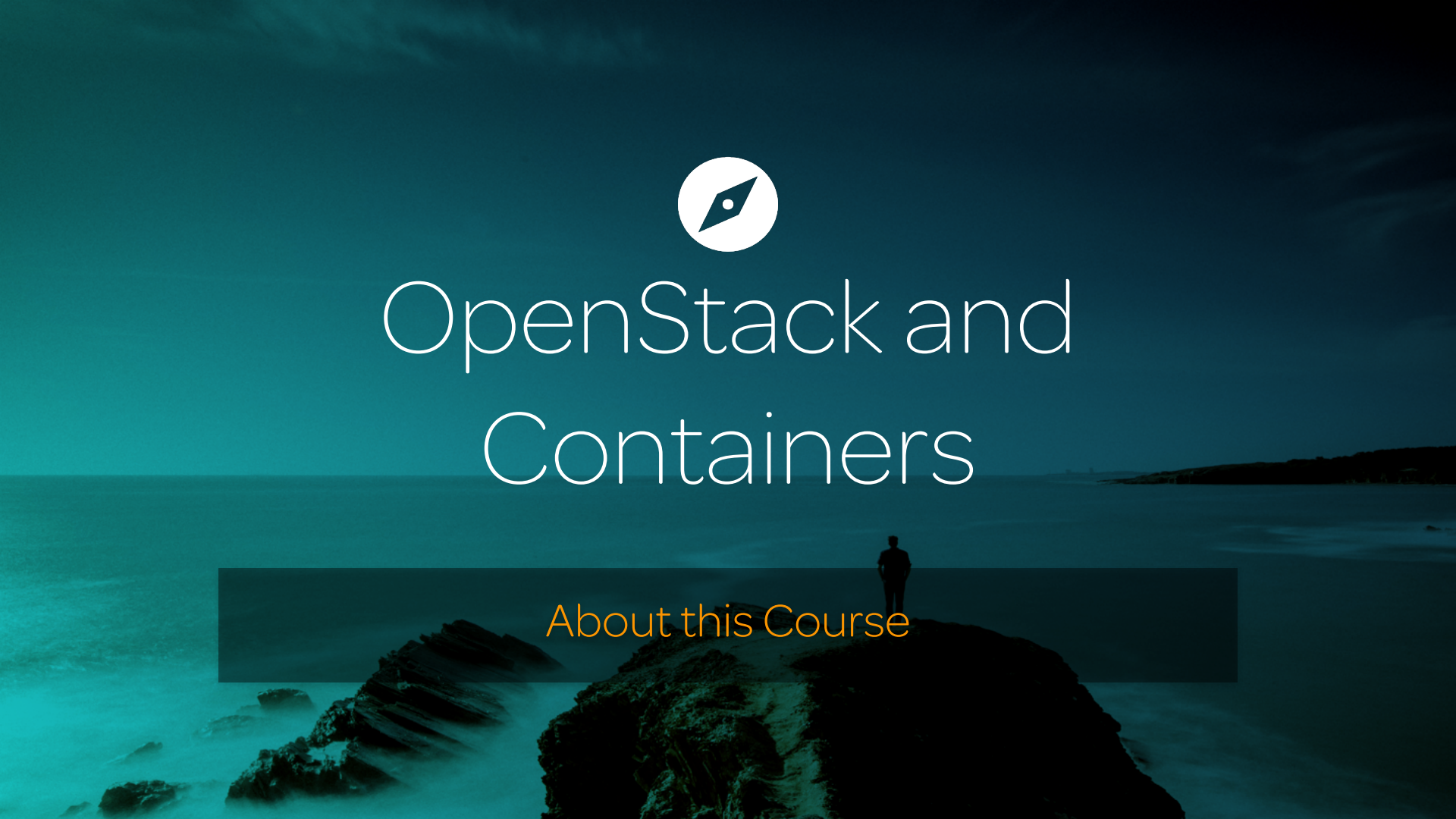


OpenStack and Containers

About this Course

About the Course

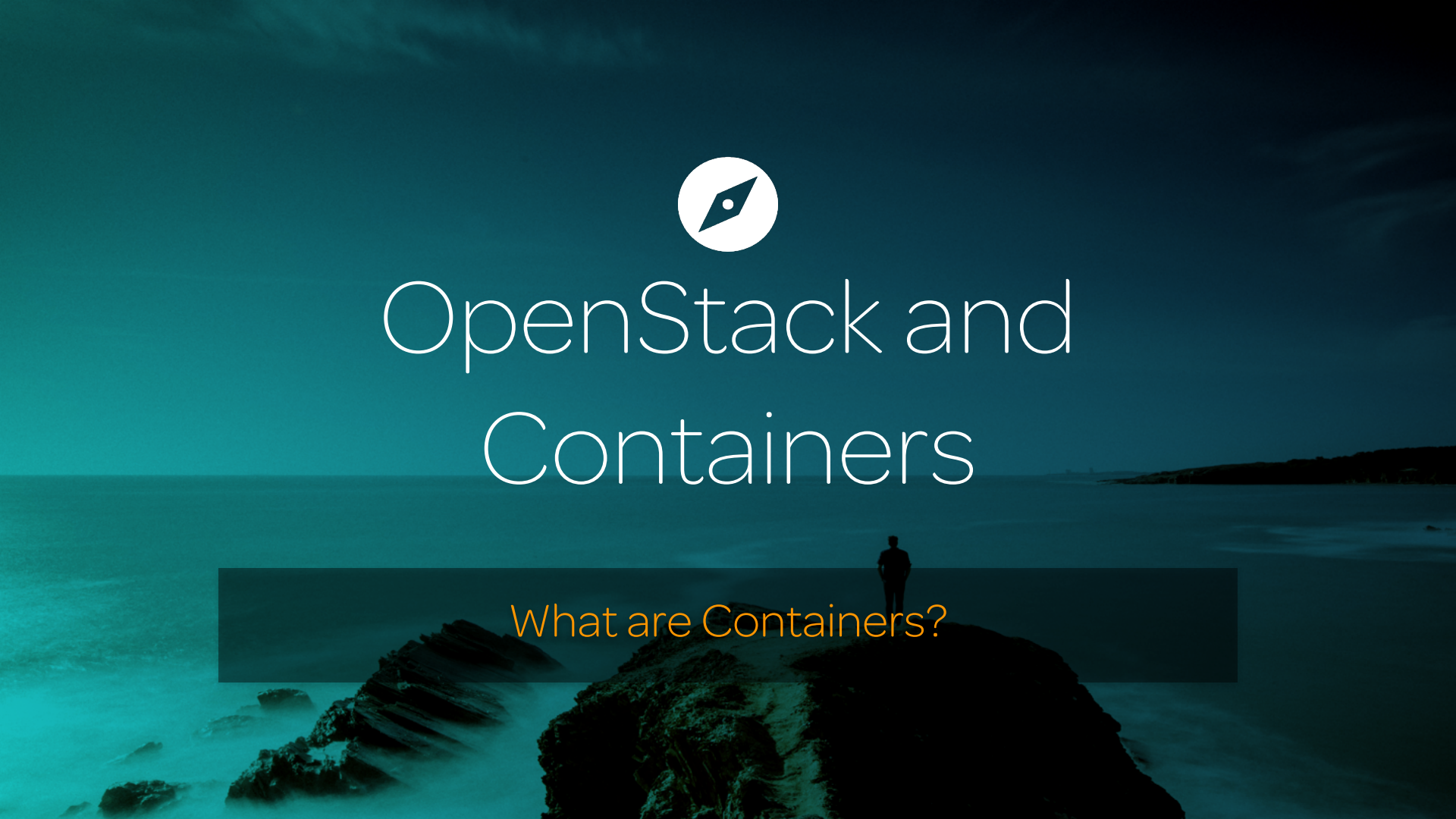
This course is designed to give you an overview of containers within the OpenStack ecosystem with a focus on:

- Deployment on Containers:
  - OpenStack-Ansible
  - Kolla
- OpenStack Zun:
  - Provides the Container Service
- OpenStack Magnum:
  - Provides the Container Infrastructure Management Service

Course Pre-Requisites

The course assumes that the student has an understanding of Linux, Docker, OpenStack, and Kubernetes:

- Mastering the Linux Command Line
- Docker Quick Start
- LXC/LXD Deep Dive
- Ansible Quick Start
- OpenStack Essentials



OpenStack and Containers

What are Containers?

What Are Containers?

Containers

- Are isolated, portable environments where you can run applications.
- Include all the libraries and dependencies they need.
- Share system resources for access to compute, networking, and storage, along with the same OS kernel.

What Are Containers?

Containers

Container Images

- Allow containers to be used on any host with a modern Linux kernel.
- Allows for more rapid deployment of applications than if they were packaged in a virtual machine image.

VMs vs. Containers

Container Orchestration Engines

Kubernetes

- The most popular container orchestration system for automating deployment, scaling, and management of containerized applications.
- Originally designed by Google, it now falls under the Cloud Native Computing Foundation.

Container Orchestration Engines

Kubernetes

Docker Swarm

- Swarm is a clustering and scheduling tool for Docker containers.
- It allows for the establishment and management of a cluster as a single virtual system.
- It is overseen by Docker.

Container Orchestration Engines

Kubernetes

Docker Swarm

Mesos

- Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively.
- Mesos is part of the Apache Foundation.

Container Types

LXC/LXD

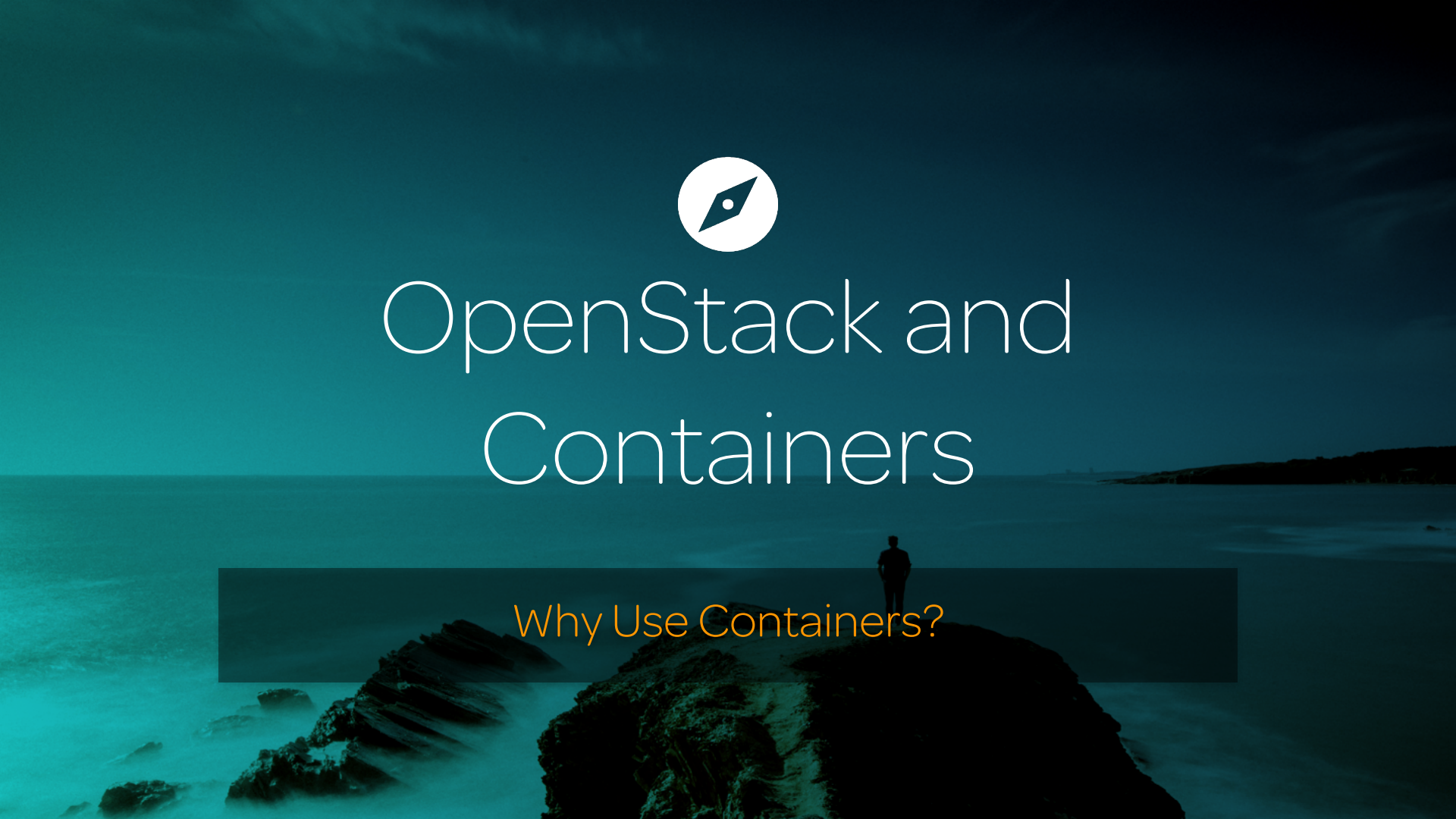
- Is a Linux operating system level virtualization method which can run multiple isolated Linux systems on a single host.
- Namespaces and cgroups make LXC possible.

Container Types

LXC/LXD

Docker

- Originally based off of LXC.
- It abstracts away more of the networking, storage and Operating System details from the application.



OpenStack and Containers

Why Use Containers?

Ways to Use Containers?

System Containers

- Similar to VMs in that they contain an Operating System, but more lightweight and still portable.
- Can support multiple applications within a single container.
- Utilize an init system for process management.

Ways to Use Containers?

System Containers

Application Containers

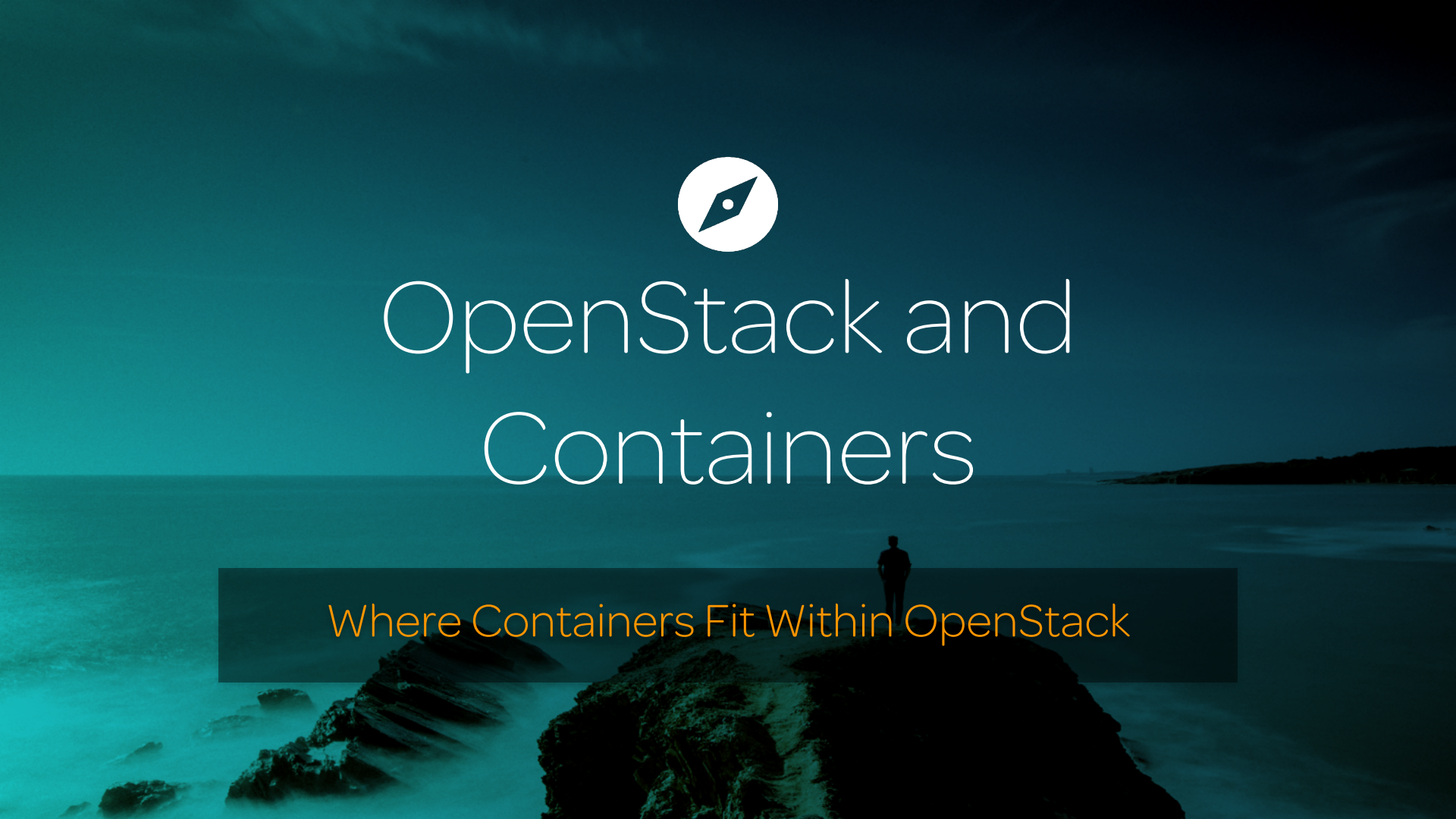
- Extremely lightweight and portable as they contain binaries, libraries, and the applications they are running.
- Less overhead as they share resources with the host machine.
- Expect a single application to execute within the container, though there may be one or more processes.
- Ideal for microservice architecture, with different application containers running different aspects of the application.

General Use Cases for Application Containers

- 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 and 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.

Microservices

- Application containers can be used to break down and isolate parts of isolated applications called microservices:
    - This 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.



OpenStack and Containers

Where Containers Fit Within OpenStack

Projects in OpenStack Utilizing Containers

OpenStack on Containers

- OpenStack-Ansible (OSA)
- Kolla
- OpenStack-Helm

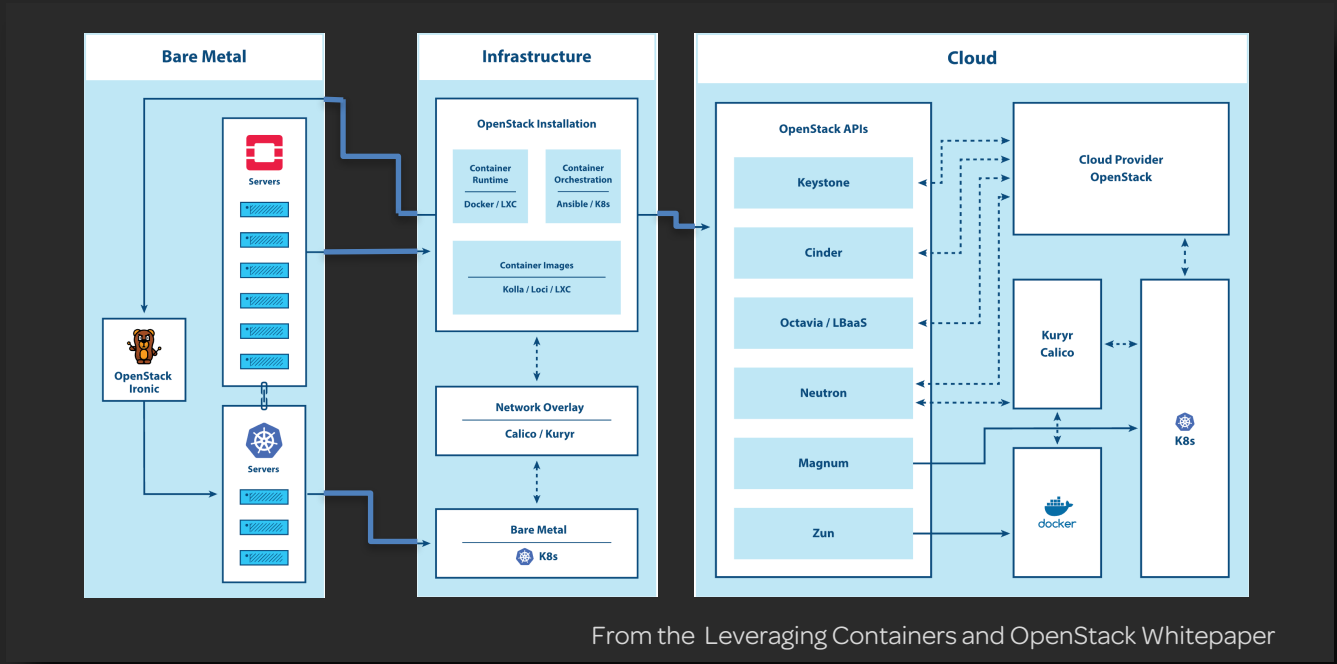
Projects in OpenStack Utilizing Containers

OpenStack on Containers

Containers Running on OpenStack

- Magnum
- Zun
- Murano
- Kuryr

Containers in OpenStack



From the Leveraging Containers and OpenStack Whitepaper

Real World Use Cases

OpenStack in Containers

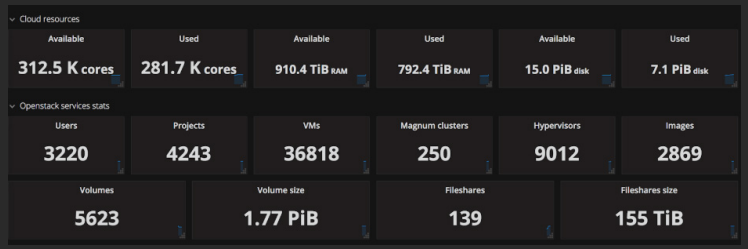
- There are several private and public cloud offerings utilizing OpenStack deployments within containers.
- Deployment projects such as OpenStack-Ansible and Kolla make deployments of OpenStack easier at scale, or for smaller internally operated clouds.

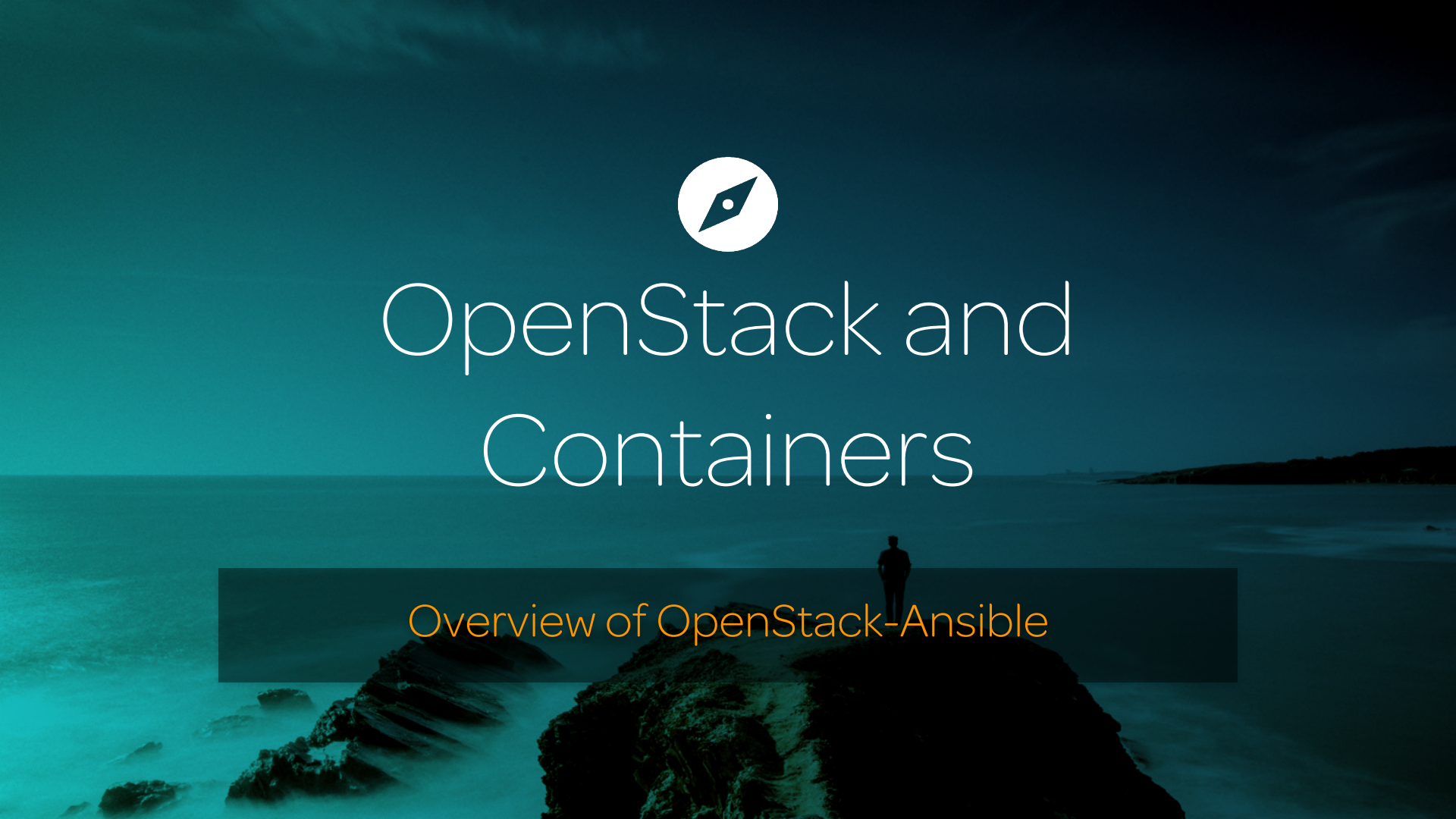
Real World Use Cases

OpenStack in Containers

Containers on OpenStack

- CERN deploys 250 Container Clusters through Magnum





OpenStack and Containers

Overview of OpenStack-Ansible

History of the Project

- Created in 2013 as an internal tool for Rackspace Private Cloud
- Moved to Stackforge in 2014
- Became an official project in 2015 and was migrated into the main repositories

OpenStack-Ansible Manifesto

Project Scope

- Batteries Included
- Solely focused on the deployment of OpenStack and its requirements

OpenStack-Ansible Manifesto

Project Scope

Ansible Usage

- Ansible provides the automation platform.
- All roles are independent of each other and testable separately.
- All roles are built as Galaxy compatible.

OpenStack-Ansible Manifesto

Project Scope

Ansible Usage

Source Based Deployments

- OpenStack services and their Python dependencies are built and installed from source code as found within the OpenStack Git repositories by default.
- Deployers can point to their own repositories.

OpenStack-Ansible Manifesto

Project Scope

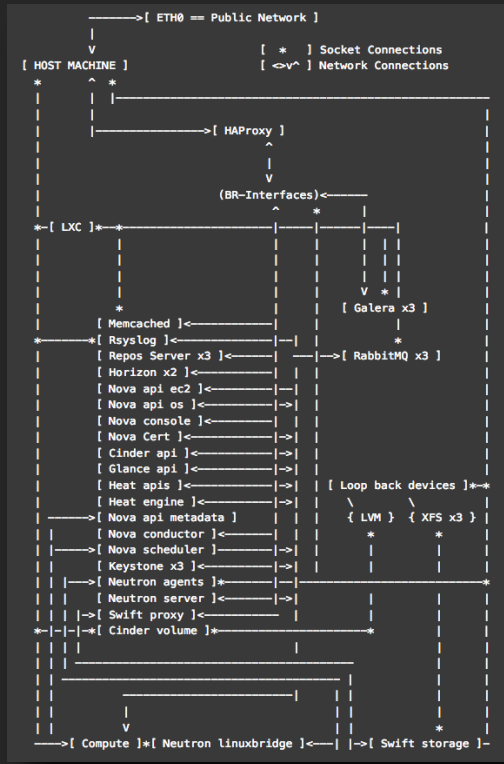
Ansible Usage

Source Based Deployments

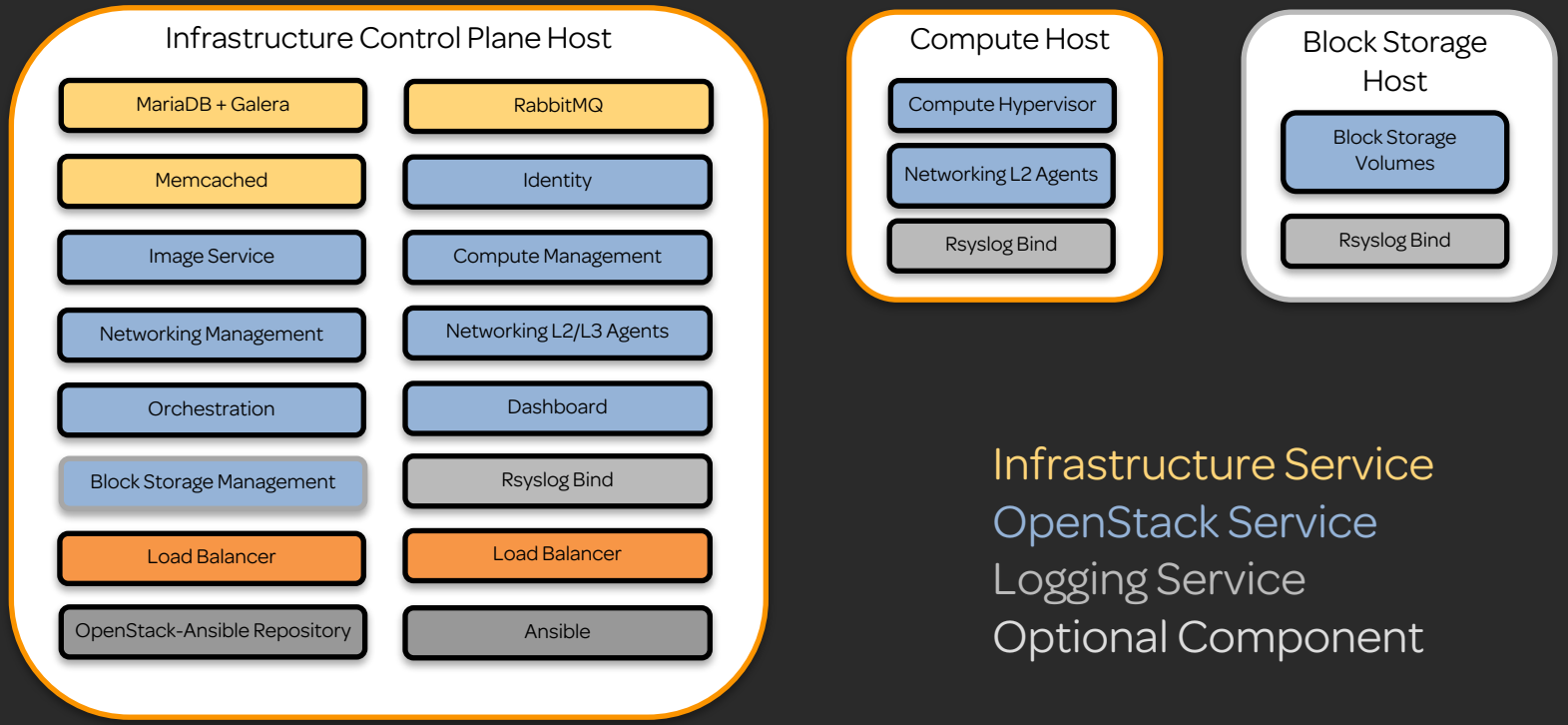
Containerized Deployments

- Containers as a means to abstract services from one another.
- Use of containers allows for additional abstractions of entire application stacks to be run all within the same physical host machines.
- The default container architecture has been built in such a way to allow for scalability and highly available deployments.
- The System Containers can be treated like physical machines.

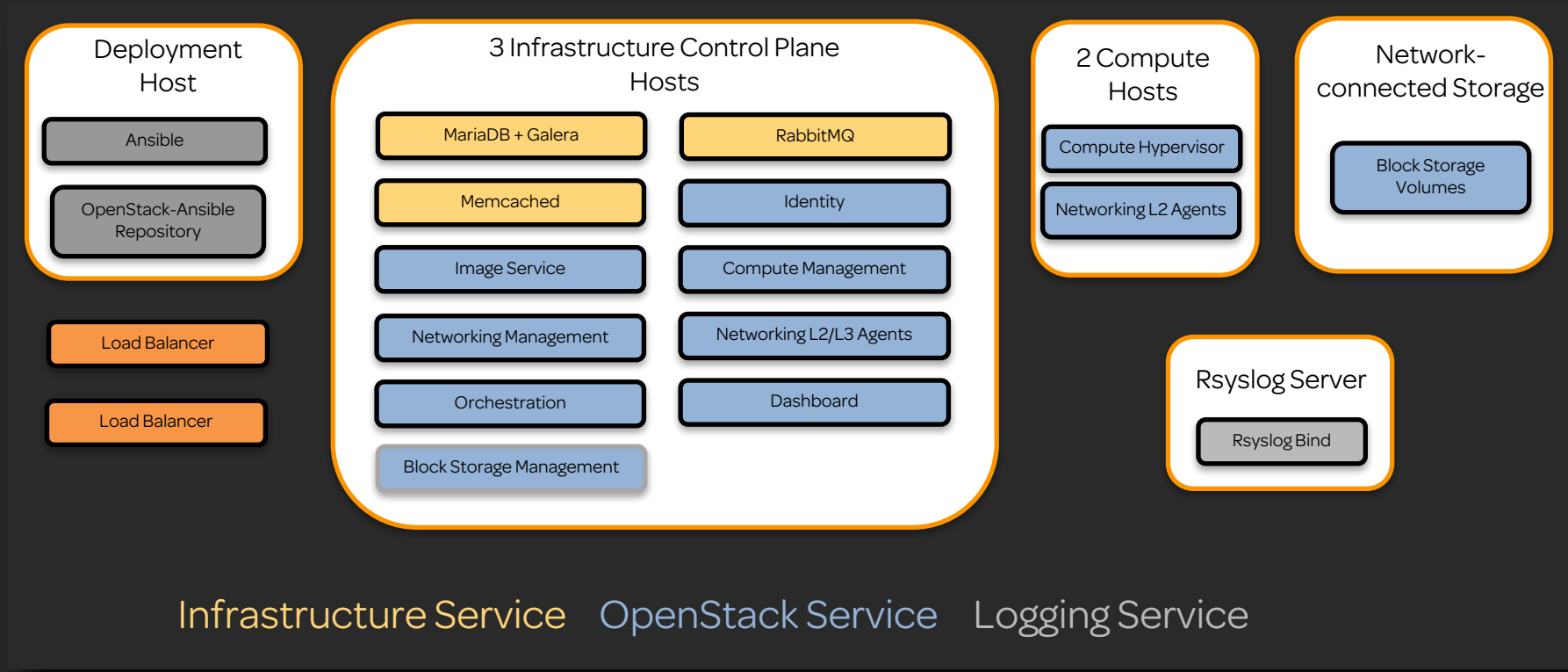
All-In-One

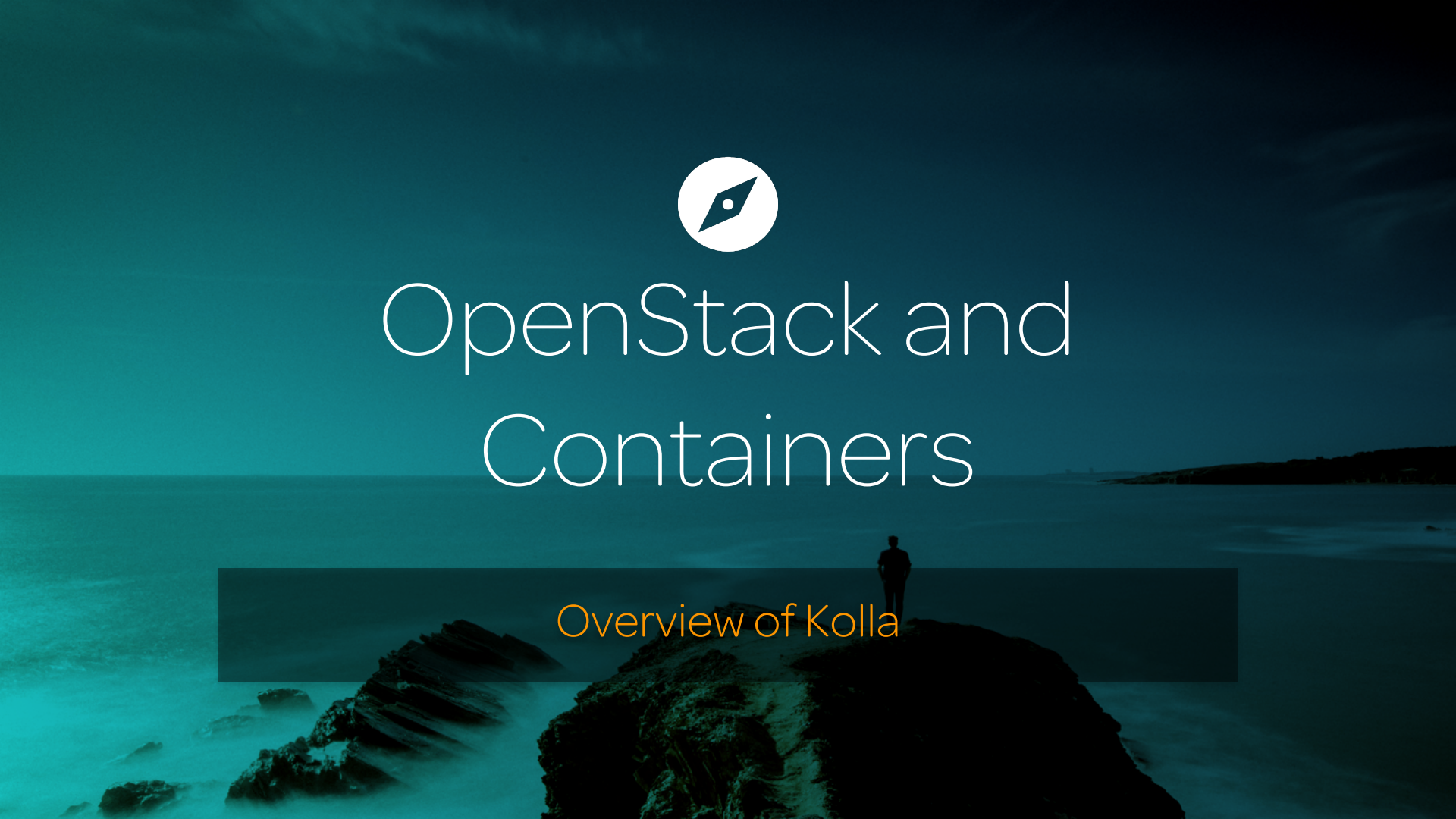

[ETH0 == Public Network]
[HOST MACHINE]
[*] Socket Connections
[<>v^] Network Connections
[HAProxy]
(BR-Interfaces)
[LXC]
[Memcached]
[Rsyslog]
[Repos Server x3]
[Horizon x2]
[Nova api ec2]
[Nova api os]
[Nova console]
[Nova Cert]
[Cinder api]
[Glance api]
[Heat apis]
[Heat engine]
[Nova api metadata]
[Nova conductor]
[Nova scheduler]
[Keystone x3]
[Neutron agents]
[Neutron server]
[Swift proxy]
[Cinder volume]
[Galera x3]
[RabbitMQ x3]
[Loop back devices]
{ LVM } { XFS x3 }
[Compute]
[Neutron linuxbridge]
[Swift storage]

Test Environment



Production Environment





OpenStack and Containers

Overview of Kolla

History of the Project

Began as a way to containerize OpenStack with Docker and Kubernetes.

Became an official project in 2015 and was migrated into the main repositories.

Kolla's Mission Statement

Kolla's mission is to provide production-ready containers and deployment tools for operating OpenStack Clouds.

Key Features

Image Building

- kolla-build builds the container images on multiple distros.
- Multiple dependent components built at the same time.

Key Features

Image Building

Support for Docker Hub

- Kolla has images stored on Docker Hub.

Key Features

Image Building

Support for Docker Hub

Local Registry Support

- Images can be pushed to a local registry.

Key Features

Image Building

Support for Docker Hub

Local Registry Support

Build Sources

- Supports building from multiple sources and source binaries.

Key Features

Image Building

Support for Docker Hub

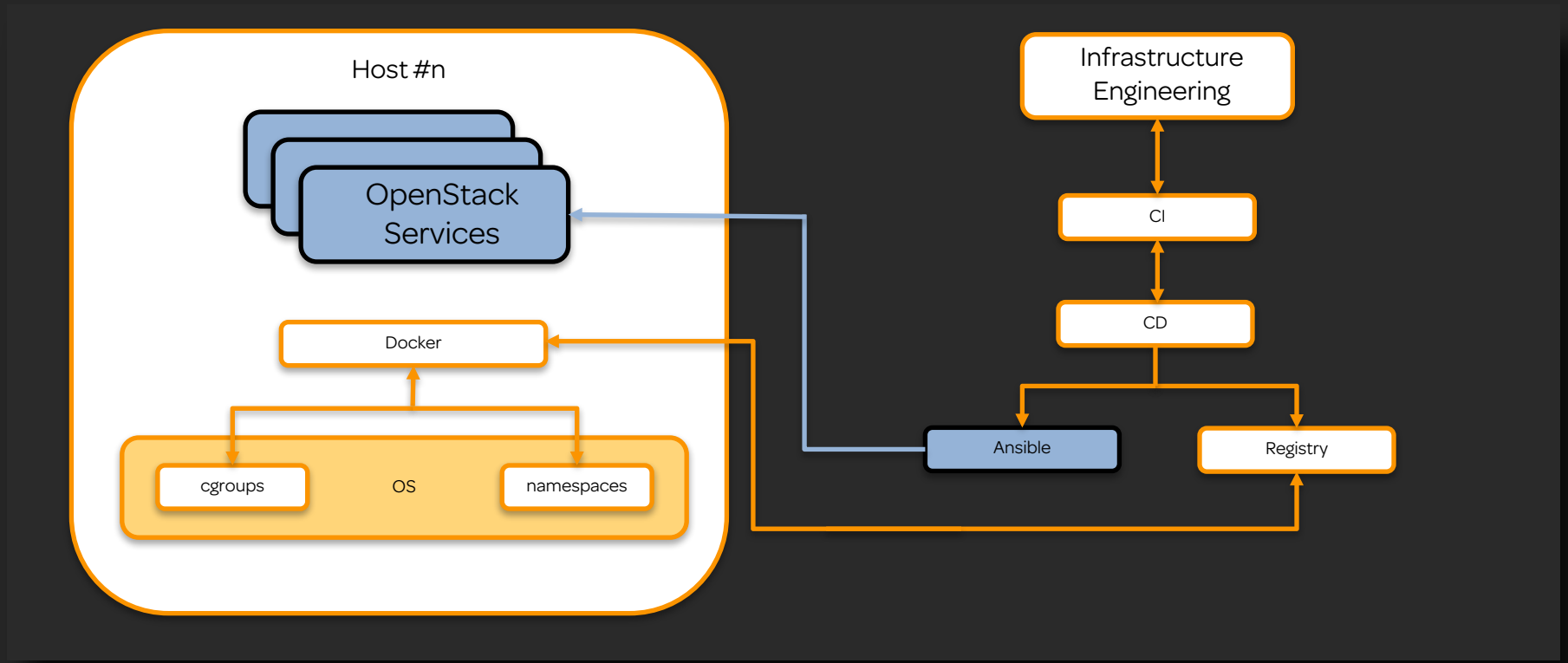
Local Registry Support

Build Sources

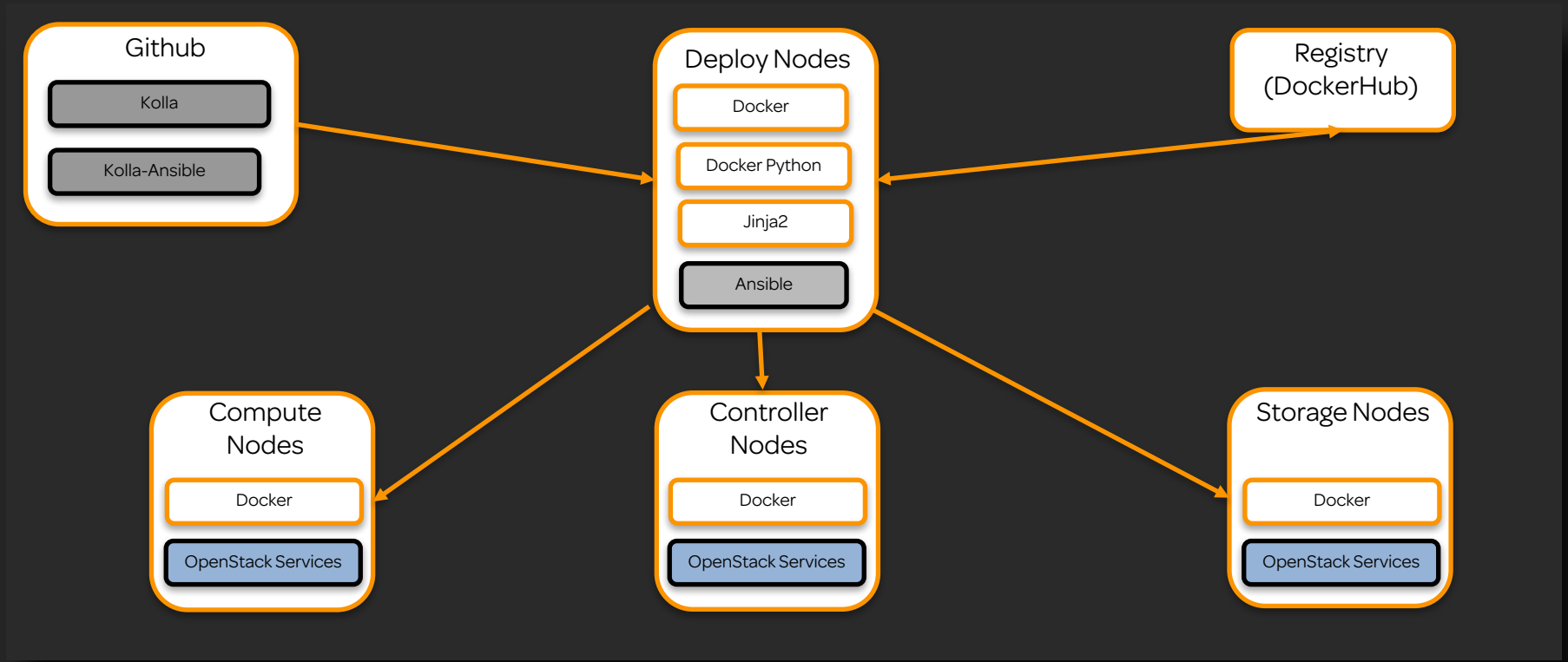
Dockerfile Customization

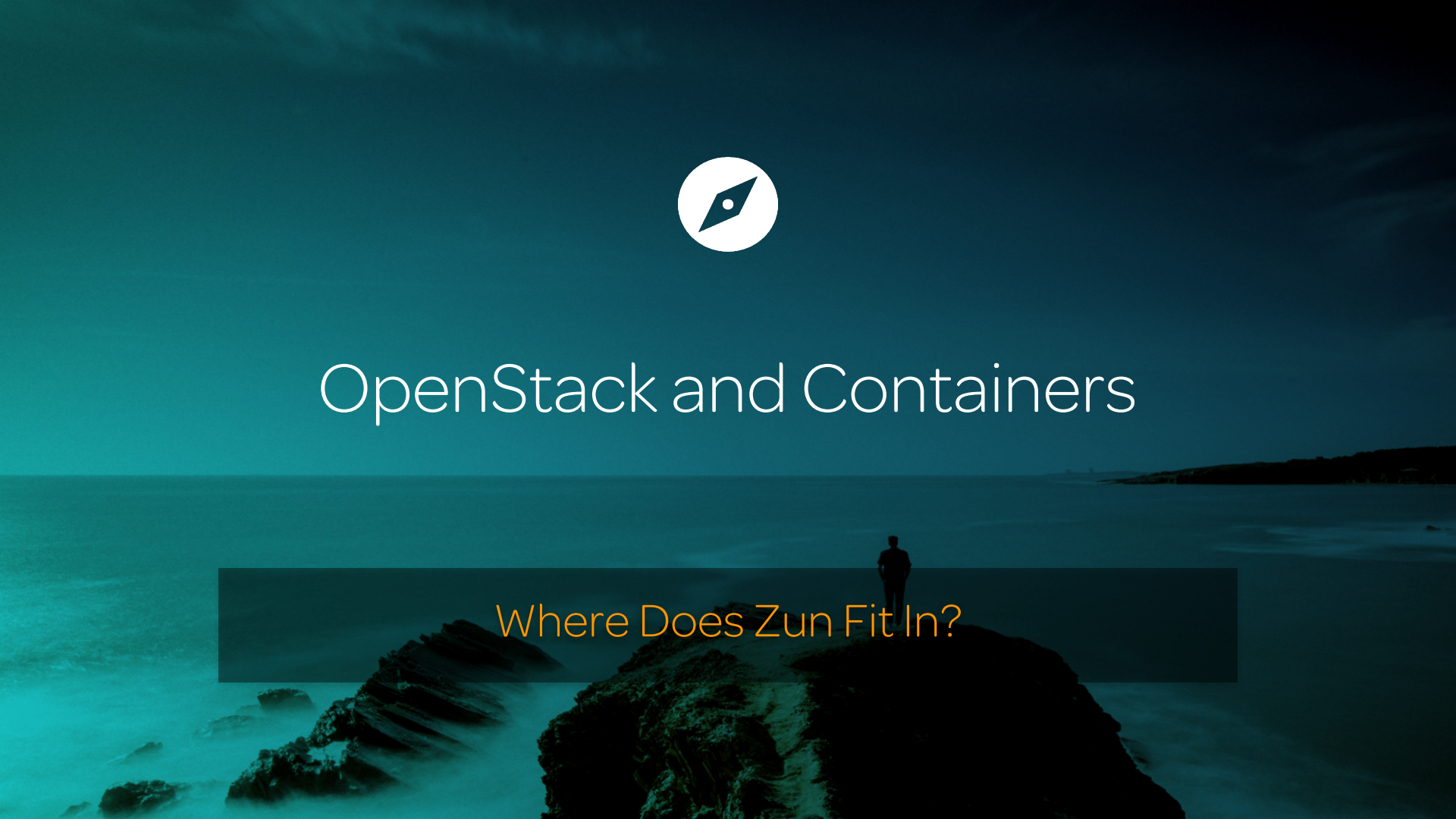
- Images can be configured to include additional packages, install plugins, change configuration settings, etc.

Kolla Architecture



Production Environment





OpenStack and Containers

Where Does Zun Fit In?

History of the Project

- At the Austin Summit in 2016, it was decided to make Containers and Container Orchestration Engines(COE) into 2 projects.
- It became an official project in November 2016.

Zun Overview

Zun

- It is an OpenStack Container service.
- Provides an API service for running application containers without the need to manage servers or clusters.

Zun Overview

Zun

Features

- OpenStack-native APIs.
- Common infrastructure for VMs, Bare metal, as well as containers in addition to SR-IOV support.
- Docker support including runc and clear container for container management.
- Utilizes a container composition known as capsules which lets users run multiple containers with related resources as a single unit.
- Ability to run heavy workloads by exposing CPU sets.

Zun Overview

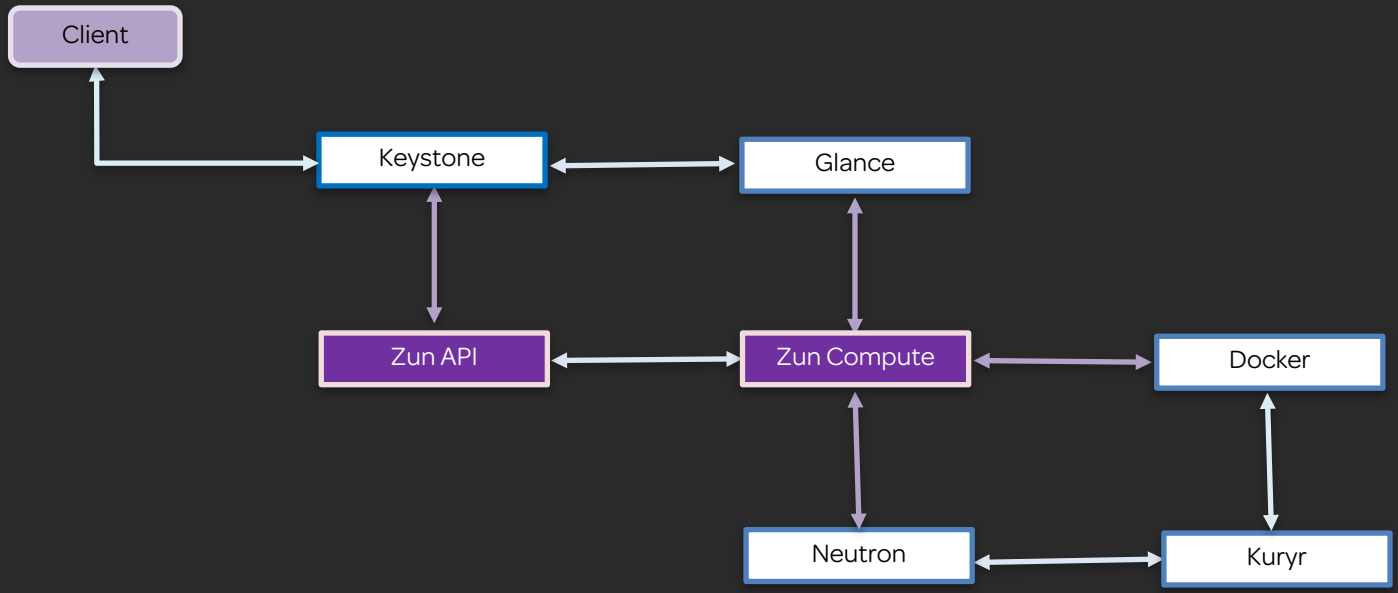
Zun

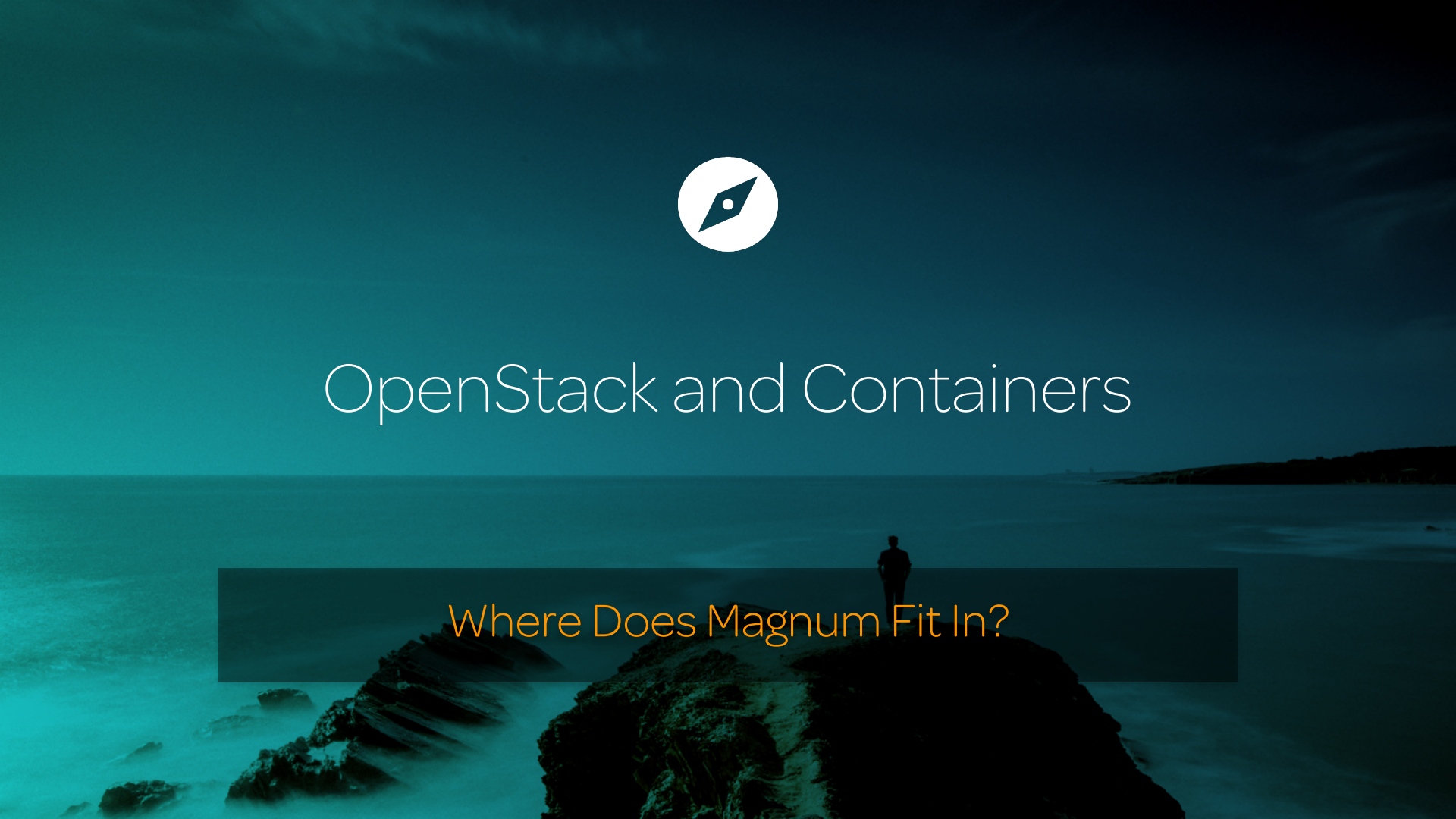
Features

OpenStack Service Integration

- Requires Keystone, Neutron, and Kuryr-libnetwork to function
- Can be integrated with Cinder, Heat, and Glance

Zun Architecture





OpenStack and Containers

Where Does Magnum Fit In?

History of the Project

- Created by the OpenStack Containers team which was formed in May 2014.
- Became an official project in March 2015.
- Passed CNCF Conformance testing in August 2018 to be certified.

Magnum Overview

Magnum

- Is an OpenStack API service that makes Container Orchestration Engines (COE) available in OpenStack.
- Uses Heat to orchestrate an OS image which contains Docker and the COE.
- Offers complete life-cycle management of COEs in an OpenStack environment.

Magnum Overview

Magnum

Features

- Standard API based complete life-cycle management for Container Clusters.
- Choice of COE.
- Choice of the container cluster deployment model.
- Multi-tenant security and auth management through Keystone.
- Multi-tenant network control and isolation through Neutron.
- Volume service for containers with Cinder.
- Integrated with OpenStack: SSO experience for cloud users.
- Secure container cluster access .

Magnum Overview

Magnum

Features

ClusterTemplate

- It is a collection of parameters to describe how to construct a cluster.
- Parameters can be for the cluster infrastructure or for the COE.

Magnum Overview

Magnum

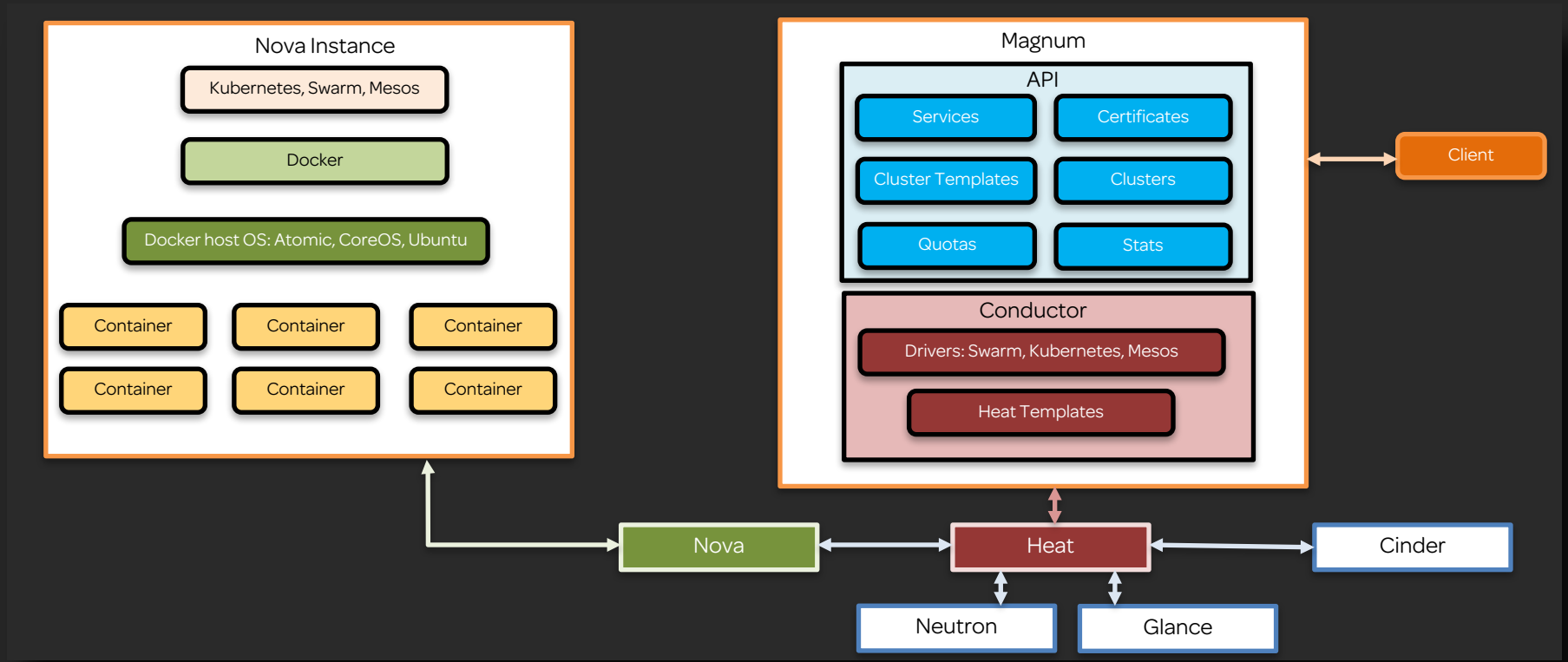
Features

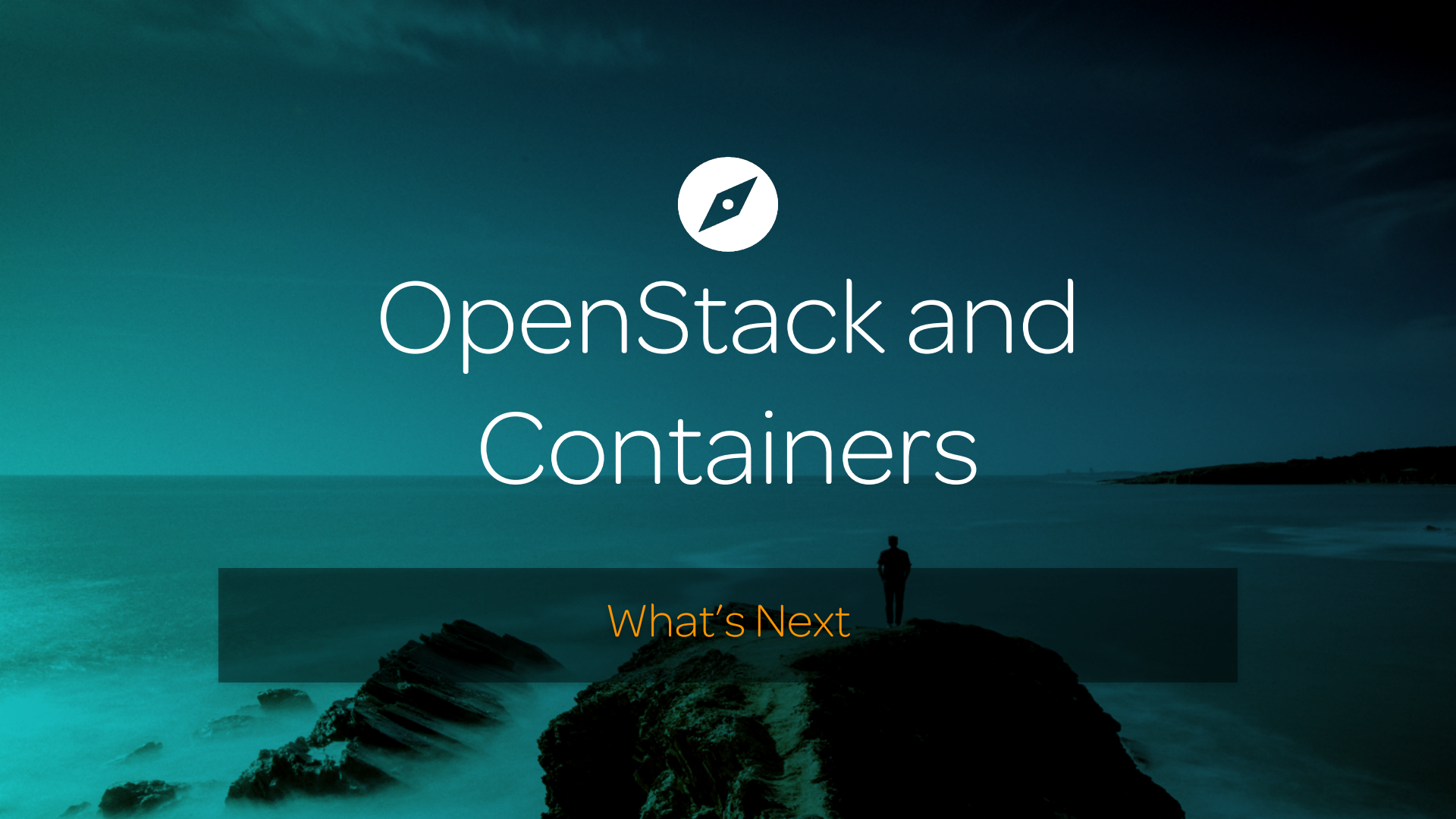
ClusterTemplate

Cluster

- Previously known as a Bay model.
- It is an instance of the ClusterTemplate of a COE.
- Once successfully deployed, the cluster is a fully operational COE that can host containers within it.

Magnum Architecture




OpenStack and
Containers
What's Next

What's Next

OpenStack Certification

- Certified OpenStack Administrator (COA)
- OpenStack MCA100 – Associates Certification

Container Related Certification

- Certified Kubernetes Administrator (CKA)
- Docker Certified Administrator Prep Course

What's Next (cont.)

Other Areas to Explore

- AWS Essentials
- Linux Academy Red Hat Certified Systems Administrator Prep Course
- DevOps Essentials
- Docker Deep Dive