

# FlexPod Infrastructure as Code (IaC) for Red Hat OpenShift Container Platform 4.7 Bare Metal

Bare Metal Installer Provisioned Infrastructure for Red Hat OpenShift Container Platform 4.7 with Cisco UCS, Cisco Nexus, NetApp AFF A-Series and Red Hat Ansible

Published: August 2021



In partnership with:



## About the Cisco Validated Design Program

The Cisco Validated Design (CVD) program consists of systems and solutions designed, tested, and documented to facilitate faster, more reliable, and more predictable customer deployments. For more information, go to:

<http://www.cisco.com/go/designzone>.

ALL DESIGNS, SPECIFICATIONS, STATEMENTS, INFORMATION, AND RECOMMENDATIONS (COLLECTIVELY, "DESIGNS") IN THIS MANUAL ARE PRESENTED "AS IS," WITH ALL FAULTS. CISCO AND ITS SUPPLIERS DISCLAIM ALL WARRANTIES, INCLUDING, WITHOUT LIMITATION, THE WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE. IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THE DESIGNS, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

THE DESIGNS ARE SUBJECT TO CHANGE WITHOUT NOTICE. USERS ARE SOLELY RESPONSIBLE FOR THEIR APPLICATION OF THE DESIGNS. THE DESIGNS DO NOT CONSTITUTE THE TECHNICAL OR OTHER PROFESSIONAL ADVICE OF CISCO, ITS SUPPLIERS OR PARTNERS. USERS SHOULD CONSULT THEIR OWN TECHNICAL ADVISORS BEFORE IMPLEMENTING THE DESIGNS. RESULTS MAY VARY DEPENDING ON FACTORS NOT TESTED BY CISCO.

CCDE, CCENT, Cisco Eos, Cisco Lumin, Cisco Nexus, Cisco StadiumVision, Cisco TelePresence, Cisco WebEx, the Cisco logo, DCE, and Welcome to the Human Network are trademarks; Changing the Way We Work, Live, Play, and Learn and Cisco Store are service marks; and Access Registrar, Aironet, AsyncOS, Bringing the Meeting To You, Catalyst, CCDA, CCDP, CCIE, CCIP, CCNA, CCNP, CCSP, CCVP, Cisco, the Cisco Certified Inter-network Expert logo, Cisco IOS, Cisco Press, Cisco Systems, Cisco Systems Capital, the Cisco Systems logo, Cisco Unified Computing System (Cisco UCS), Cisco UCS B-Series Blade Servers, Cisco UCS C-Series Rack Servers, Cisco UCS S-Series Storage Servers, Cisco UCS Manager, Cisco UCS Management Software, Cisco Unified Fabric, Cisco Application Centric Infrastructure, Cisco Nexus 9000 Series, Cisco Nexus 7000 Series, Cisco Prime Data Center Network Manager, Cisco NX-OS Software, Cisco MDS Series, Cisco Unity, Collaboration Without Limitation, EtherFast, EtherSwitch, Event Center, Fast Step, Follow Me Browsing, FormShare, Giga-Drive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, Power-Panels, ProConnect, ScriptShare, SenderBase, SMARTnet, Spectrum Expert, StackWise, The Fastest Way to Increase Your Internet Quotient, TransPath, WebEx, and the WebEx logo are registered trademarks of Cisco Systems, Inc. and/or its affiliates in the United States and certain other countries. (LDW\_U1).

All other trademarks mentioned in this document or website are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (0809R)

© 2021 Cisco Systems, Inc. All rights reserved.

---

## Contents

Executive Summary .....	4
Solution Overview.....	5
Technology Overview .....	8
Solution Design .....	17
Solution Deployment.....	32
References .....	109
Summary .....	110
About the Authors .....	111
Feedback.....	112

---

## Executive Summary

To help organizations with their digital transformation and to enhance their cloud-native and application modernization practices, Cisco and NetApp have partnered to produce this Cisco Validated Design (CVD) for the FlexPod™ for Red Hat OpenShift Container Platform bare metal solution delivered as Infrastructure as Code (IaC).

FlexPod delivers an integrated architecture that incorporates compute, storage, and network design best practices thereby minimizing IT risks by validating the integrated architecture to ensure compatibility between various components. The solution also addresses IT pain points by providing documented design guidance, deployment guidance and support that can be used in various stages (planning, designing and implementation) of a deployment. FlexPod delivered as IaC further eliminates error-prone manual tasks, allowing quicker and more consistent solution deployments.

Red Hat® OpenShift® is an enterprise-ready Kubernetes container platform with full-stack automated operations to manage hybrid cloud and multi-cloud deployments. Red Hat OpenShift is optimized to improve developer productivity and promote innovation. The Red Hat OpenShift Container Platform gives developers a self-service platform on which to build and run containerized applications. With Red Hat OpenShift you can quickly start creating new cloud-native applications or cloud-enabling existing applications and spawning an environment for a new microservice in minutes.

Combining Red Hat OpenShift with the FlexPod solution can simplify the deployment and the management of the container infrastructure. The Red Hat Ansible integration with FlexPod solution automates deployment of FlexPod infrastructure along with the OpenShift Container platform installation enabling customers to take advantage of programming and automating the infrastructure at scale with agility, extending the benefits of automation to the entire stack.

This combined solution helps organizations achieve the speed, flexibility, security, and scale required for all their application modernization and digital transformation initiatives.



---

## Solution Overview

### Introduction

The featured FlexPod for OpenShift Container platform solution delivered as IaC is a pre-designed, integrated, and validated architecture for the data center that combines Cisco UCS servers, the Cisco Nexus family of switches, and NetApp AFF A-series storage into a single, flexible architecture. FlexPod is designed for high availability (HA), with no single point of failure, while maintaining cost-effectiveness and flexibility in the design to support a wide variety of workloads. The FlexPod solution covered in this document is for bare metal implementation of Red Hat OpenShift Container Platform (OCP) installer provisioned infrastructure (IPI), built on Enterprise Kubernetes for an on-premises deployment.

Integration between OpenShift Container Platform and the storage and data management services occur at several levels, all of which are captured in the design aspects of this document. The main storage integration is based on Container Storage Interface (CSI) Astra Trident for Kubernetes Driver for NetApp storage systems, which enables container orchestrators such as Kubernetes to manage the life cycle of persistent storage.

The OCP platform is installed as a bare metal cluster with the OCP nodes running Red Hat Enterprise Linux CoreOS (RHCOS) on Cisco UCS servers.

The following design and deployment aspects of the FlexPod for OCP solution are explained in this document:

- Red Hat OpenShift Container Platform 4.7
- Red Hat OpenShift Virtualization 2.6
- FlexPod converged infrastructure
- CSI Astra Trident for Kubernetes – Dynamic storage provisioner for OpenShift
- Cisco UCS Manager 4.1(3)

The document also covers key configurations based on the validated environment and best practices.

### Audience

The intended audience of this document includes but is not limited to data scientists, IT architects, sales engineers, field consultants, professional services, IT managers, partner engineering, DevOps, and Site Reliability Engineers (SREs) and customers who want to take advantage of an infrastructure built to deliver IT efficiency and enable IT innovation.

### Purpose of This Document

The purpose of this design and deployment guide is to provide a reference architecture with specific examples indicating how the solution was designed, deployed, and tested. In addition, the document provides several best practices and recommendations that simplify your implementation of this solution.

### What's New in this Release?

The following elements distinguish this version of FlexPod from previously published solutions:

- Support for Red Hat OpenShift Container Platform 4.7.
- Support for Red Hat OpenShift Virtualization 2.6

- 
- Support for Red Hat OCP Bare Metal Installer provisioned infrastructure implementation.
  - Fully automated solution deployment covering FlexPod infrastructure and OCP installation.
  - Support for the Cisco UCS release 4.1(3)
  - NetApp Astra Trident CSI for Kubernetes v1.20

## Solution Summary

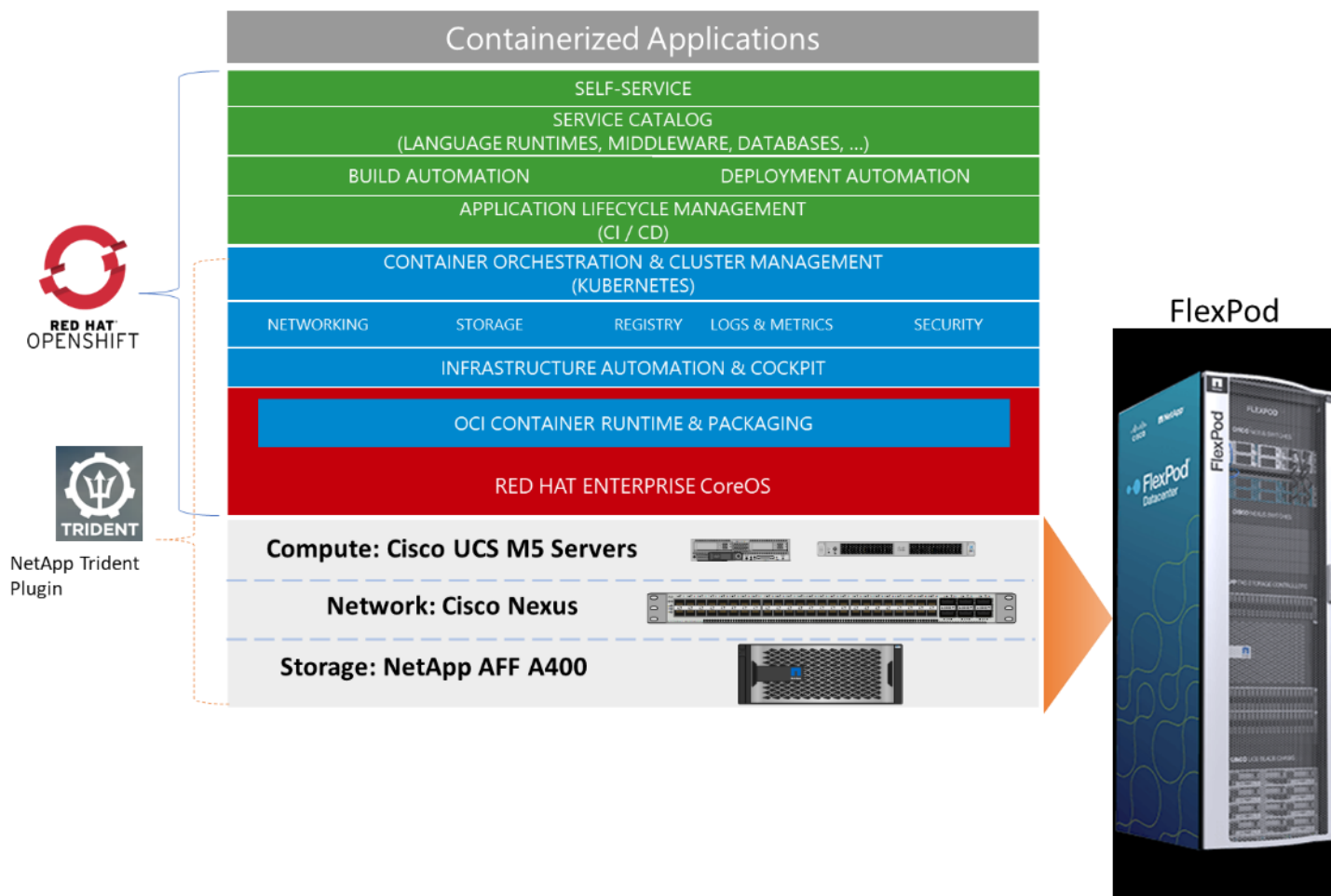
This solution includes a hardware stack from Cisco and NetApp, an OCP software platform and Ansible from Red Hat, a set of domain managers and tools for integration and management.

The FlexPod solution for OpenShift Container Platform 4.7 bare metal comprises of the following core components:

- Compute and networking components from Cisco
- Storage Systems and Astra Trident CSI plugin from NetApp
- OpenShift Container Platform software and Ansible from Red Hat

These components are integrated and validated, and the entire stack is automated so that customers can deploy the solution quickly and economically while eliminating many of the risks associated with researching, designing, building, and deploying similar solutions from the ground up.

Figure 1. FlexPod IaC for OpenShift Container Platform 4 Bare Metal



Like all other FlexPod solution designs, FlexPod for OCP 4 bare metal is configurable according to demand and usage. Customers can purchase exactly the infrastructure they need for their current application requirements and can then scale-up by adding more resources to the FlexPod system or scale-out by adding more FlexPod instances.

## Technology Overview

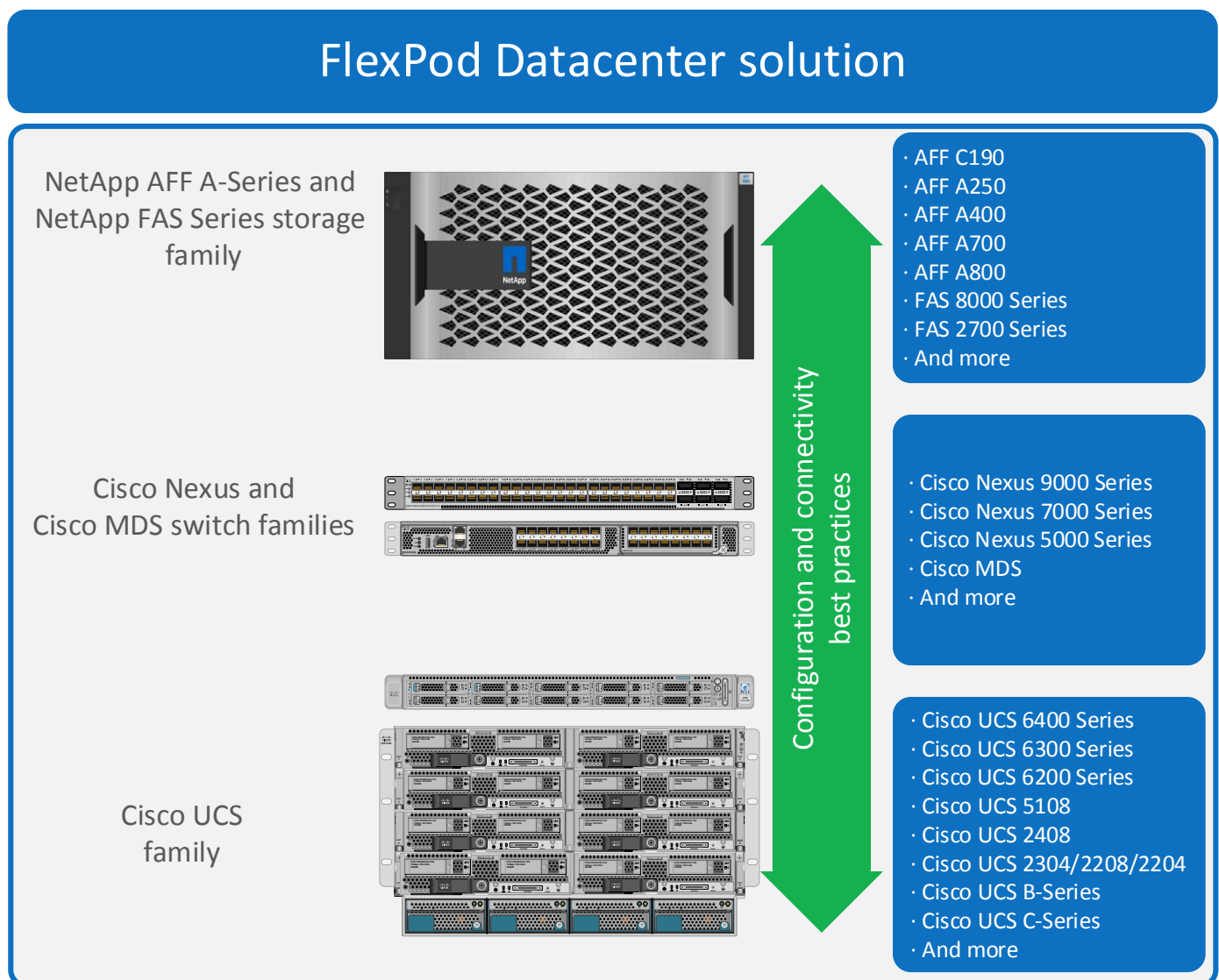
FlexPod is a best practice datacenter architecture that includes the following components:

- Cisco Unified Computing System (Cisco UCS)
- Cisco Nexus and Cisco MDS\* Switches
- NetApp AFF Systems



\* This CVD does not explain Fiber Channel storage connectivity; therefore, Cisco MDS is not part of the design.

Figure 2. FlexPod Component Families



These components are connected and configured according to the best practices of both Cisco and NetApp to provide an ideal platform for running a variety of enterprise workloads with confidence. FlexPod can scale up for greater performance and capacity (adding compute, network, or storage resources individually as needed), or it can scale out for environments that require multiple consistent deployments (such as rolling out of additional FlexPod stacks). The reference architecture covered in this document leverages Cisco Nexus 9000 for the network switching element.

One of the key benefits of FlexPod is its ability to maintain consistency during scale. Each of the component families shown (Cisco UCS, Cisco Nexus, and NetApp AFF) offers platform and resource options to scale the infrastructure up or down, while supporting the same features and functionality that are required under the configuration and connectivity best practices of FlexPod.

The FlexPod reference architecture explained in this document leverages:

- Cisco UCS Manager on Cisco 4<sup>th</sup> Generation 6454 Fabric Interconnects to support 10GbE, 25GbE, 40GbE, and 100GbE connectivity from various components.
- Cisco UCS 5108 Chassis with Cisco UCS B200 M5 blade servers and Cisco UCS C220 M5 rack servers to support Red Hat OCP 4 bare metal deployment.
- High-Speed Cisco NXOS based Nexus 93180YC-FX switching designed to support up to 100GbE connectivity.
- NetApp AFF A400 with NVMe disk shelves and 25GbE connectivity to the Cisco Nexus switching fabric.
- Red Hat OpenShift Container Platform (version 4.7)
- NetApp Astra Trident CSI Plugin version 21.04

The key features and highlights for these FlexPod components are explained below.

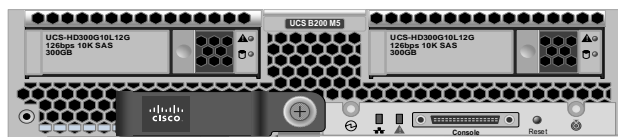
## Cisco Unified Computing System

Cisco Unified Computing System™ (Cisco UCS®) is an integrated computing infrastructure with intent-based management to automate and accelerate deployment of all your applications, including virtualization and cloud computing, scale-out and bare metal workloads, and in-memory analytics, as well as edge computing that supports remote and branch locations and massive amounts of data from the Internet of Things (IoT). The system is flexible, agile, and adaptable, and the portfolio of products supported by Cisco UCS includes blade, rack, multi-node, and storage-intensive servers; converged infrastructure; hyperconverged infrastructure; and solutions for the network edge such as Cisco UCS Mini. Cisco UCS supports blade, rack, multinode, and storage servers in a single domain of up to 160 servers.

### Cisco UCS B200 M5 Blade Servers

The Cisco UCS B200 M5 server shown in [Figure 3](#), is a half-width blade upgrade from the Cisco UCS B200 M4.

**Figure 3. Cisco UCS B200 M5 Blade Server**



It features the following:

- 2<sup>nd</sup> Gen Intel® Xeon® Scalable and Intel® Xeon® Scalable processors with up to 28 cores per socket
- Up to 24 DDR4 DIMMs for improved performance with up to 12 DIMM slots ready for Intel Optane™ DC Persistent Memory
- Up to two GPUs
- Two Small-Form-Factor (SFF) drive slots
- Up to two Secure Digital (SD) cards or M.2 SATA drives
- Up to 80 Gbps of I/O throughput with Cisco UCS 6454 FI

For more information about the Cisco UCS B200 M5 Blade Servers, see: <http://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/ucs-b-series-blade-servers/datasheet-c78-739296.html>.

### Cisco UCS C220 M5 Rack Servers

The Cisco UCS C220 M5 rack server shown in [Figure 4](#), is a high-density 2-socket rack server that is an upgrade from the Cisco UCS C220 M4.

**Figure 4. Cisco UCS C220 M5 Rack Server**



It features the following:

- 2<sup>nd</sup> Gen Intel® Xeon® Scalable and Intel® Xeon® Scalable processors, 2-socket
- Up to 24 DDR4 DIMMs for improved performance with up to 12 DIMM slots ready for Intel Optane™ DC Persistent Memory
- Up to 10 Small-Form-Factor (SFF) 2.5-inch drives or 4 Large-Form-Factor (LFF) 3.5-inch drives (77 TB storage capacity with all NVMe PCIe SSDs)
- Support for 12-Gbps SAS modular RAID controller in a dedicated slot, leaving the remaining PCIe Generation 3.0 slots available for other expansion cards
- Modular LAN-On-Motherboard (mLOM) slot that can be used to install a Cisco UCS Virtual Interface Card (VIC) without consuming a PCIe slot
- Dual embedded Intel x550 10GBASE-T LAN-On-Motherboard (LOM) ports
- Up to 100 Gbps of I/O throughput with Cisco UCS 6454 FI

For more information about the Cisco UCS B200 M5 Blade Servers, see: <https://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/ucs-c-series-rack-servers/datasheet-c78-739281.html>.

### Cisco UCS 6400 series Fabric Interconnects

The Cisco UCS Fabric Interconnects (FIs) provide a single point for connectivity and management for the entire Cisco UCS system. Typically deployed as an active-active pair, the system's FIs integrate all components into a single, highly available management domain controlled by the Cisco UCS Manager. Cisco UCS FIs provide a sin-



gle unified fabric for the system, with low-latency, lossless, cut-through switching that supports LAN, SAN and management traffic using a single set of cables.

The Cisco UCS 6454 ([Figure 5](#)) deployed for this validation, provides the management and communication backbone for the Cisco UCS B-Series Blade Servers, Cisco UCS 5108 B-Series Server Chassis and Cisco UCS Managed C-Series Rack Servers. All servers attached to the Cisco UCS 6454 Fabric Interconnect become part of a single, highly available management domain. In addition, by supporting a unified fabric, the Cisco UCS 6454 provides both the LAN and SAN connectivity for all servers within its domain. The Cisco UCS 6454 supports deterministic, low-latency, line-rate 10/25/40/100 Gigabit Ethernet ports, a switching capacity of 3.82 Tbps, and 200 Gbps bandwidth between FI 6454 and IOM 2408 per 5108 blade chassis, independent of packet size and enabled services.

**Figure 5. Cisco UCS 6400 series Fabric Interconnect**



### Cisco UCS 2408 Fabric Extender

The Cisco UCS 2408 connects the I/O fabric between the Cisco UCS 6454 Fabric Interconnect and the Cisco UCS 5100 Series Blade Server Chassis, enabling a lossless and deterministic converged fabric to connect all blades and chassis together.

The Cisco UCS 2408 Fabric Extender has eight 25-Gigabit Ethernet, FCoE-capable, Small Form-Factor Pluggable (SFP28) ports that connect the blade chassis to the fabric interconnect. Each Cisco UCS 2408 provides 10-Gigabit Ethernet ports connected through the midplane to each half-width slot in the chassis, giving it a total of 32 10G interfaces to Cisco UCS blades. Typically configured in pairs for redundancy, two fabric extenders provide up to 400 Gbps of I/O from FI 6400's to 5108 chassis.

### Cisco UCS 1400 Series Virtual Interface Cards (VICs)

Cisco VICs support Cisco SingleConnect technology, which provides an easy, intelligent, and efficient way to connect and manage computing in your data center. Cisco SingleConnect unifies LAN, SAN, and systems management into one simplified link for rack servers and blade servers. This technology reduces the number of network adapters, cables, and switches needed and radically simplifies the network, reducing complexity. Cisco VICs can support 256 Express (PCIe) virtual devices, either virtual Network Interface Cards (vNICs) or virtual Host Bus Adapters (vHBAs), with a high rate of I/O Operations Per Second (IOPS), support for lossless Ethernet, and 10/25/40/100-Gbps connection to servers. The PCIe Generation 3 x16 interface helps ensure optimal bandwidth to the host for network-intensive applications, with a redundant path to the fabric interconnect. Cisco VICs support NIC teaming with fabric failover for increased reliability and availability. In addition, it provides a policy-based, stateless, agile server infrastructure for your data center.

The Cisco VIC 1400 series is designed exclusively for the M5 generation of Cisco UCS B-Series Blade Servers and Cisco UCS C-Series Rack Servers. The adapters can support 10/25/40/100-Gigabit Ethernet and Fibre Channel over Ethernet (FCoE). It incorporates Cisco's next-generation Converged Network Adapter (CNA) technology and offers a comprehensive feature set, providing investment protection for future feature software releases.

---

## Cisco UCS Management

While Cisco UCS is stateless, programmable infrastructure, the Cisco UCS unified API is how management tools program it. This enables the tools to help guarantee consistent, error-free, policy-based alignment of server personalities with workloads. Through automation, transforming the server and networking components of your infrastructure into a complete solution is fast and error-free because programmability eliminates the error-prone manual configuration of servers and integration into solutions. Server, network, and storage administrators are now free to focus on strategic initiatives rather than spending their time performing tedious tasks.

### Cisco UCS Manager

Cisco UCS® Manager (UCSM) provides unified, integrated management for all software and hardware components in Cisco UCS manages a single domain through an intuitive HTML 5-based GUI. is embedded in each fabric interconnect. Running in a redundant, high-availability configuration, it creates a single, self-aware, self-integrating unified system that recognizes and integrates components as they are added to the system. It quickly and accurately configures computing, network, storage, and storage-access resources to reduce the chance of errors that can cause downtime. Its role and policy-based approach helps organizations more easily align policies and configurations with workloads. While Cisco UCS Manager requires an “always on” connection, our other tools are evolving to manage systems to which they are not continuously connected.

### Cisco Intersight Software-as-a-Service Management

This platform has the broadest scope of the Cisco UCS management tools. It enables programming the infrastructure by automating configuration and management, but it goes the farthest in integrating with outside services and tools.

Accessed from the cloud or through an optional local management appliance, Intersight provides a single interface from which you can undertake lifecycle management of your servers whether they are in a core data center or at the network edge. New features are continually integrated, and you can keep up to date on the most current enhancements by visiting [cisco.com/go/intersight](https://cisco.com/go/intersight).

The Intersight platform enables you to configure the identity, personality, and connectivity of blade and rack servers. Intersight provides the following additional capabilities that are complementary to the basic deployment and configuration features:

- **Global dashboard and inventory** - When you manage your infrastructure with Cisco Intersight, you can view a global dashboard that gives you overall server status and enables you to drill down to view individual components (such as disk drives) With a global inventory of your devices, it's easy to track the location of each of your assets.
- **Cisco TAC** - With Intersight's integration with Cisco TAC, you can quickly remediate problems because expertise and information can flow seamlessly between Intersight and your Cisco support center. The system can open cases and upload supporting documentation for fast resolution. It maintains the status of your contracts and licenses so that you can administer them from the same interface.
- **Recommendation engine** - This feature gives you recommendations on configurations and help you implement best practices. Intersight has insight into your operating system and driver versions. It can use these to validate that your implementations are supported by Cisco's Hardware Configuration List (HCL).

## DevOps and Tool Support

The Cisco UCS unified API is of great benefit to developers and administrators who want to treat physical infrastructure the way they treat other application services, using processes that automatically provision or change IT resources. Similarly, your IT staff needs to provision, configure, and monitor physical and virtual resources; automate routine activities; and rapidly isolate and resolve problems. The Cisco UCS unified API integrates with DevOps management tools and processes and enables you to easily adopt DevOps methodologies.

## Cisco Nexus

Cisco Nexus series switches provide an Ethernet switching fabric for communications between the Cisco UCS, NetApp storage controllers, and the rest of a customer's network. There are many factors to consider when choosing the main data switch in this type of architecture to support both the scale and the protocols required for the resulting applications. All Nexus switch models including the Cisco Nexus 5000 and Cisco Nexus 7000 are supported in this design and may provide additional features such as FCoE or OTV. However, be aware that there may be slight differences in setup and configuration based on the switch used. The validation for this deployment leverages the Cisco Nexus 9000 series switches, which deliver high performance 10/25/40/50/100GbE ports, density, low latency, and exceptional power efficiency in a broad range of compact form factors.

Many of the most recent single-site FlexPod designs also use this switch due to the advanced feature set and the ability to support Application Centric Infrastructure (ACI) mode. When leveraging ACI fabric mode, the Nexus 9000 series switches are deployed in a spine-leaf architecture. Although the reference architecture covered in this design does not leverage ACI, it lays the foundation for customer migration to ACI in the future, and fully supports ACI today if required.

For more information, go to: <http://www.cisco.com/c/en/us/products/switches/nexus-9000-series-switches/index.html>.

This FlexPod design deploys a single pair of Cisco Nexus 93180YC-FX top-of-rack switches ([Figure 6](#)) within each placement, using the traditional standalone mode running NX-OS.

**Figure 6. Cisco Nexus 93180YC-FX**



## NetApp AFF A400

The NetApp AFF A400 offers full end-to-end NVMe support. The front-end NVMe/FC connectivity makes it possible to achieve optimal performance from an all-flash array for workloads that include artificial intelligence, machine learning, real-time analytics as well as business-critical databases. On the back end, the A400 supports both serial-attached SCSI (SAS) and NVMe-attached SSDs, offering the versatility for current customers to move up from their legacy A-Series systems and satisfying the increasing interest that all customers have in NVMe-based storage. Furthermore, this system was built to provide expandability options, so you won't have to make a costly leap from a midrange to a high-end system to increase scalability. Consider this a way to future-proof your NetApp investment.

The NetApp AFF A400 offers greater port availability, network connectivity, and expandability. The NetApp AFF A400 has 10 PCIe Gen3 slots per high availability pair. The NetApp AFF A400 offers 25GbE or 100GbE, as well as 32Gb/FC and NVMe/FC network connectivity. This model was created to keep up with changing business

needs and performance and workload requirements by merging the latest technology for data acceleration and ultra-low latency in an end-to-end NVMe storage system.

**Figure 7. NetApp AFF A400**



The NetApp AFF A400 has a 4U enclosure with two possible onboard connectivity configurations (25GbE or 32Gb/FC). In addition, the A400 is the only A-Series system that has the new smart I/O card with offload engine. The offload engine is computational and independent from the CPU, which allows better allocation of processing power. This system also offers an improved level of serviceability over the previous NetApp 4U chassis: The fan cooling modules have been moved from inside the controller to the front of the chassis, so cabling does not have to be disconnected and reconnected when replacing an internal fan.

The NetApp AFF A400 is well suited for enterprise applications that require the best balance of performance and cost, as well as very demanding workloads that require ultra-low latency. The smart I/O card serves as the default cluster interconnect, making the system an ideal solution for highly compressible workloads.

For more information, go to: <https://docs.netapp.com/platstor/topic/com.netapp.nav.a400/home.html?cp=3>. This FlexPod design deploys a HA pair of A400 controller running ONTAP 9.8.

## NetApp Astra Trident CSI Plugin

Astra Trident is an open-source, fully supported storage orchestrator for containers created by NetApp. It has been designed from the ground up to help you meet your containerized applications' persistence demands using industry-standard interfaces, such as the [Container Storage Interface \(CSI\)](#). With Astra Trident, microservices and containerized applications can take advantage of enterprise-class storage services provided by the full NetApp portfolio of storage systems. In a FlexPod environment, Astra Trident is utilized to allow end users to dynamically provision and manage persistent volumes for containers backed by FlexVols and LUNs hosted on ONTAP-based products such as NetApp AFF and FAS systems.

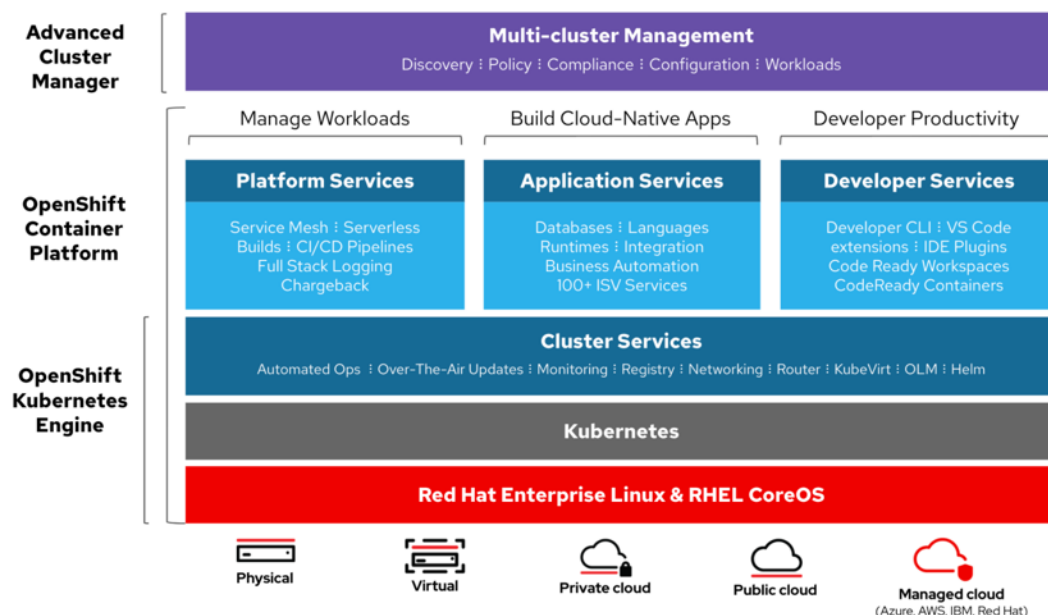
Astra Trident has a rapid development cycle, and just like Kubernetes, is released four times a year. Starting with the v21.04.0 release, the setup of Astra Trident is performed by the Trident operator using a Helm chart which makes large scale deployments easier, and provides additional support including self-healing for the pods that are deployed as a part of the Astra Trident install.

## Red Hat OpenShift Container Platform

The Red Hat OpenShift Container Platform (OCP) is a container application platform that brings together CRI-O and Kubernetes and provides an API and web interface to manage these services. CRI-O is an implementation of the Kubernetes CRI (Container Runtime Interface) to enable using Open Container Initiative (OCI) compatible runtimes. It is a lightweight alternative to using Docker as the runtime for Kubernetes.

OCP allows customers to create and manage containers. Containers are standalone processes that run within their own environment, independent of operating system and the underlying infrastructure. OCP helps developing, deploying, and managing container-based applications. It provides a self-service platform to create, modify, and deploy applications on demand, thus enabling faster development and release life cycles. OCP has a micro-services-based architecture of smaller, decoupled units that work together. It runs on top of a Kubernetes cluster, with data about the objects stored in etcd, a reliable clustered key-value store.

**Figure 8. OpenShift Container Platform Overview**



## Kubernetes Infrastructure

Within OpenShift Container Platform, Kubernetes manages containerized applications across a set of CRI-O runtime hosts and provides mechanisms for deployment, maintenance, and application-scaling. The CRI-O service packages, instantiates, and runs containerized applications.

A Kubernetes cluster consists of one or more masters and a set of worker nodes. This solution design includes HA functionality at the hardware as well as the software stack. A Kubernetes cluster is designed to run in HA mode with 3 master nodes and a minimum of 2 worker nodes to help ensure that the cluster has no single point of failure.

## Red Hat Core OS

OpenShift Container Platform uses Red Hat Enterprise Linux CoreOS (RHCOS), a container-oriented operating system that combines some of the best features and functions of the CoreOS and Red Hat Atomic Host operating systems. RHCOS is specifically designed for running containerized applications from OpenShift Container Platform and works with new tools to provide fast installation, Operator-based management, and simplified upgrades.

RHCOS includes the following:

- Ignition, which OpenShift Container Platform uses as a first boot system configuration for initially bringing up and configuring machines.

- CRI-O, a Kubernetes native container runtime implementation that integrates closely with the operating system to deliver an efficient and optimized Kubernetes experience. CRI-O provides facilities for running, stopping, and restarting containers. It fully replaces the Docker Container Engine, which was used in OpenShift Container Platform 3.
- Kubelet, the primary node agent for Kubernetes that is responsible for launching and monitoring containers.



RHCOS was used on all control plane and worker nodes to support the automated OCP 4 deployment.

---

## Red Hat Ansible

Red Hat Ansible Automation helps Red Hat OpenShift Container Platform users create and run reusable infrastructure code and automate provisioning tasks for infrastructure components.

Ansible is simple and powerful, allowing users to easily manage various physical devices within FlexPod- including the provisioning of Cisco UCS bare metal servers, Cisco Nexus switches and NetApp AFF storage. Using Ansible's Playbook-based automation is easy and integrates into your current provisioning infrastructure.

Finally, Ansible also provides robust container and native Kubernetes management, expanding to Red Hat OpenShift Container Platform and other container technologies.



---

## Solution Design

The FlexPod for Red Hat OpenShift Container Platform provides an end-to-end architecture with Cisco and NetApp technologies that demonstrate support for OCP workloads with high availability and server redundancy. The architecture consists of an OCP bare metal cluster deployed on Cisco UCS M5 servers within FlexPod infrastructure, with the Cisco UCS servers and NetApp storage attached to the Cisco Nexus 93180YC-FX switches in NXOS mode.

[Figure 9](#) illustrates a base design. Each of the components can be scaled easily to support specific business requirements. For example, additional OCP nodes can be deployed to scale the OCP environment to increase compute capacity, additional storage controllers or disk shelves can be deployed to improve I/O capability and throughput.

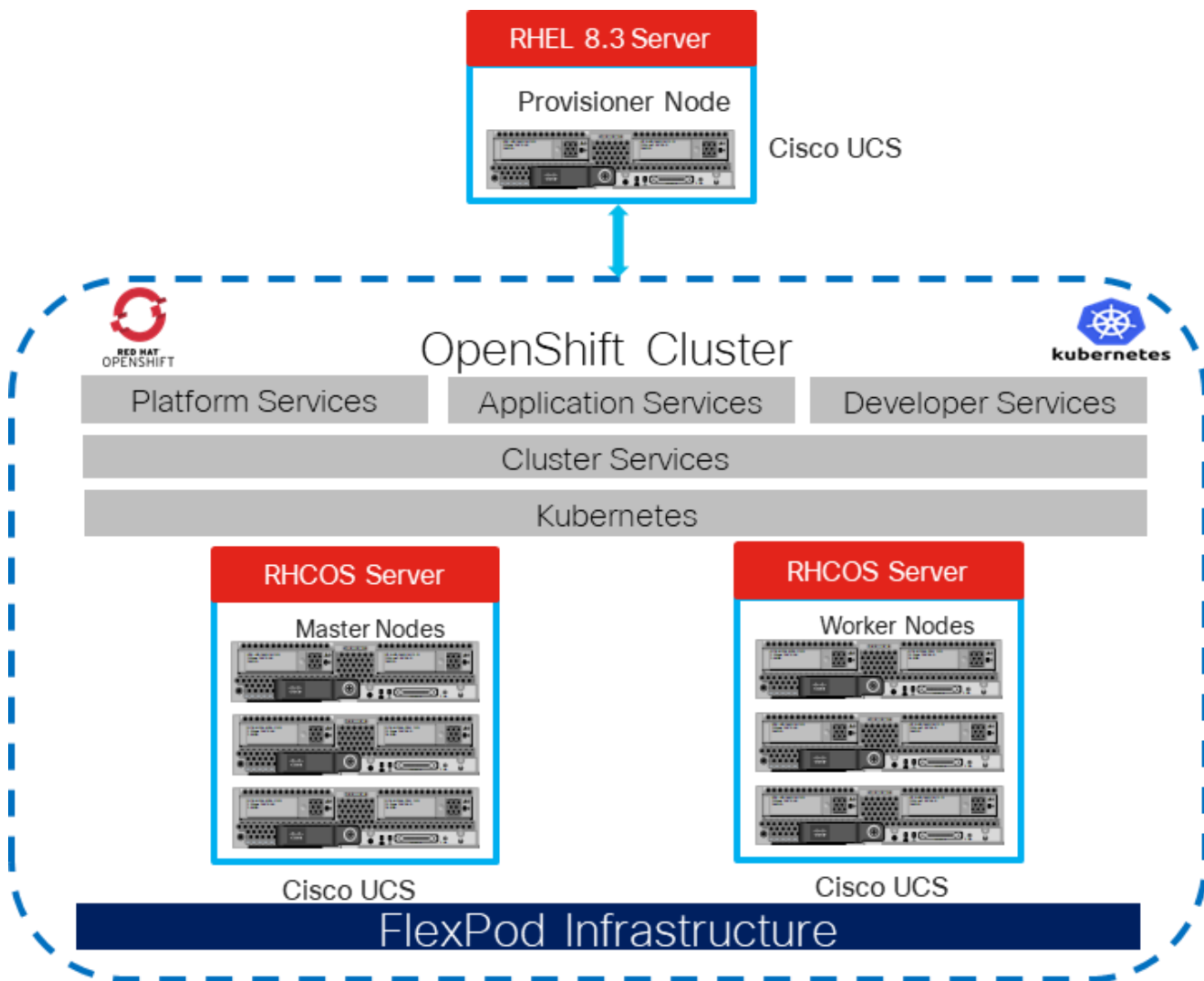


The solution was validated using Cisco UCS B200 M5 and Cisco UCS C220 M5 servers to show the versatility of the Cisco UCS platform. Customers can choose to deploy OCP on just the Cisco UCS B-Series servers, the Cisco UCS C-Series or Cisco UCS C125 servers depending on their requirements.

---

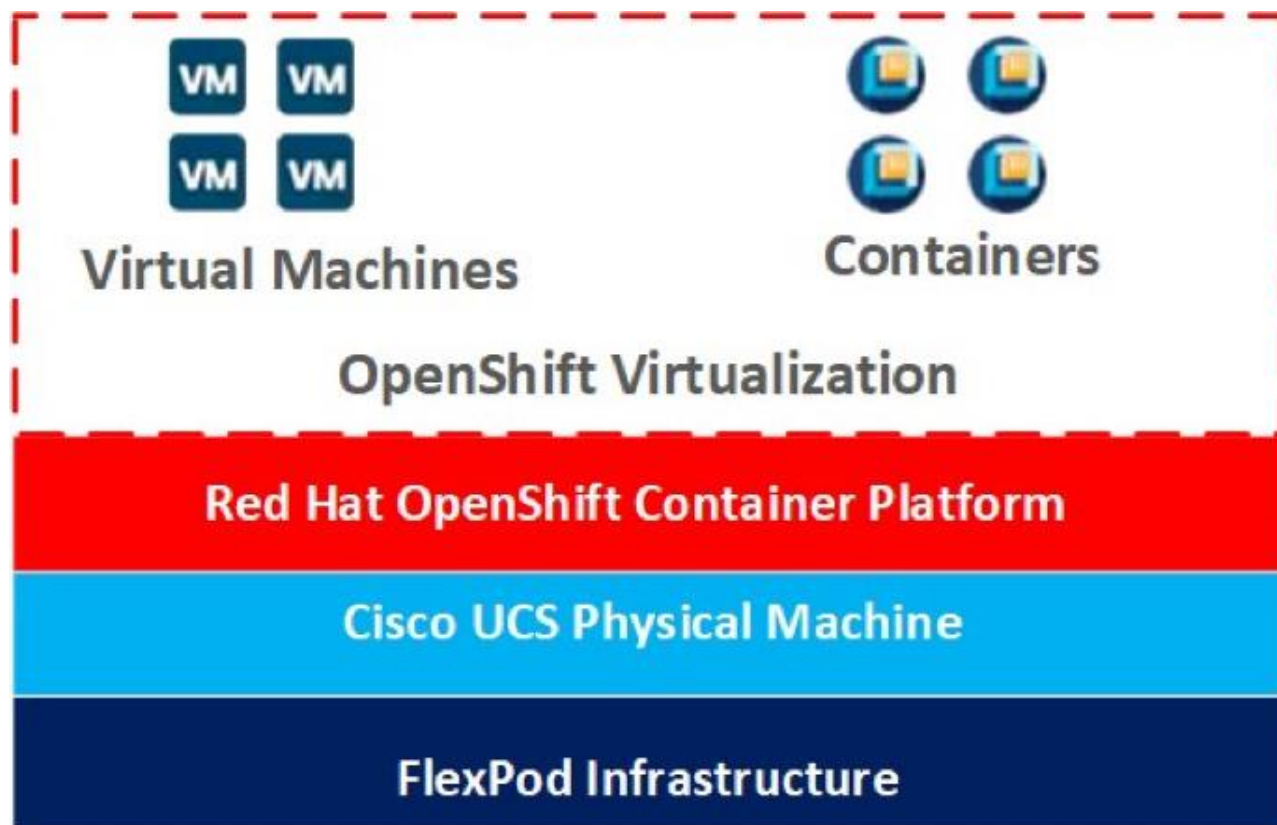
[Figure 9](#) provides a high-level overview of the FlexPod for OCP cluster architecture.

Figure 9. OpenShift Container Platform 4 Bare Metal on FlexPod



OpenShift Virtualization is an add-on to OpenShift Container Platform that allows you to run and manage virtual machine workloads alongside container workloads. The OpenShift Virtualization feature has been validated within this solution to deploy traditional VMs into OpenShift where they run side by side with containers on the same OCP cluster deployed on the FlexPod infrastructure.

Figure 10. OpenShift Virtualization



## Design Requirements

This section explains the key design requirement and various prerequisites for delivering this new solution.

The FlexPod solution for OCP bare metal closely aligns with NXOS based FlexPod CVDs and meets the following general design requirements:

- Resilient design across all layers of the infrastructure with no single point of failure.
- Scalable design with the flexibility to add compute capacity, storage, or network bandwidth as needed.
- Modular design that can be replicated to expand and grow as the needs of the business grow.
- Flexible design that can support components beyond what is validated and documented in this guide.
- Simplified design with ability to automate and integrate with external automation and orchestration tools.

For Red Hat OCP 4 integration into a traditional FlexPod solution, the following specific design considerations are also observed:

- High Availability of master nodes with a minimum of 3 master nodes deployed.
- A minimum of 2 worker nodes with ability to increase the nodes as the load requirements increase.
- Automating the FlexPod infrastructure deployment and OCP installation by utilizing Ansible Playbooks to simplify the installation and reduce the deployment time.

- Present persistent storage (volumes) to the containerized applications by utilizing the NetApp Astra Trident CSI framework.
- Dedicated Cisco UCS vNICs for different traffic needs with UCS Fabric Failover for high availability.

## FlexPod Physical Topology

This FlexPod design utilizes Cisco UCS servers connected and managed through Cisco UCS 6454 Fabric Interconnects and the integrated Cisco UCS Manager (UCSM). These high-performance servers are configured as compute nodes where Red Hat Core OS (RHCOS) is loaded using local boot leveraging the onboard M.2 drives. The persistent storage volumes for containers are provisioned on the NetApp AFF A400 using NFS NAS storage and iSCSI block storage.

This design has the following physical connectivity between the components of FlexPod:

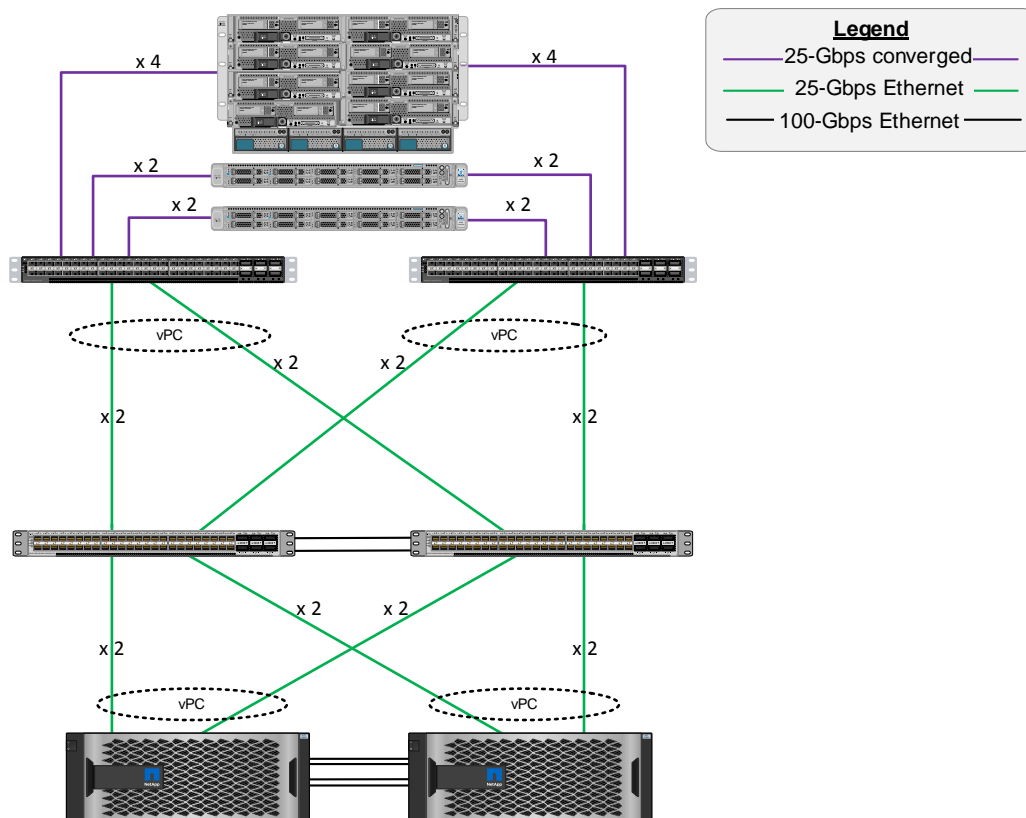
- 4 X 25 Gb Ethernet connections port-channelled between the Cisco UCS 5108 Blade Chassis and the Cisco UCS Fabric Interconnects
- 2 or 4 25 Gb Ethernet connections between the Cisco UCS C-Series rackmounts and the Cisco UCS Fabric Interconnects
- 4 X 25 Gb Ethernet connections port-channelled between the Cisco UCS Fabric Interconnect and Cisco Nexus 9000 switches
- 2 X 100 Gb Ethernet connections port-channelled between the Cisco Nexus Switches for the vPC peer-link
- 4 X 25 Gb Ethernet connections port-channelled between the Cisco Nexus Switches and each NetApp AFF A400 storage controller

**Figure 11. FlexPod Physical Topology**

**Cisco Unified Computing System**  
Cisco UCS 6454 Fabric Interconnects, UCS 2408 Fabric Extenders, UCS B-Series Blade Servers with UCS VIC 1440, UCS C-Series Rack Servers with UCS VIC 1457, UCS C4200 Chassis, and UCS C125 Servers with UCS VIC 1455

**Cisco Nexus 93180YC-FX**

**NetApp storage controllers AFF-A400**



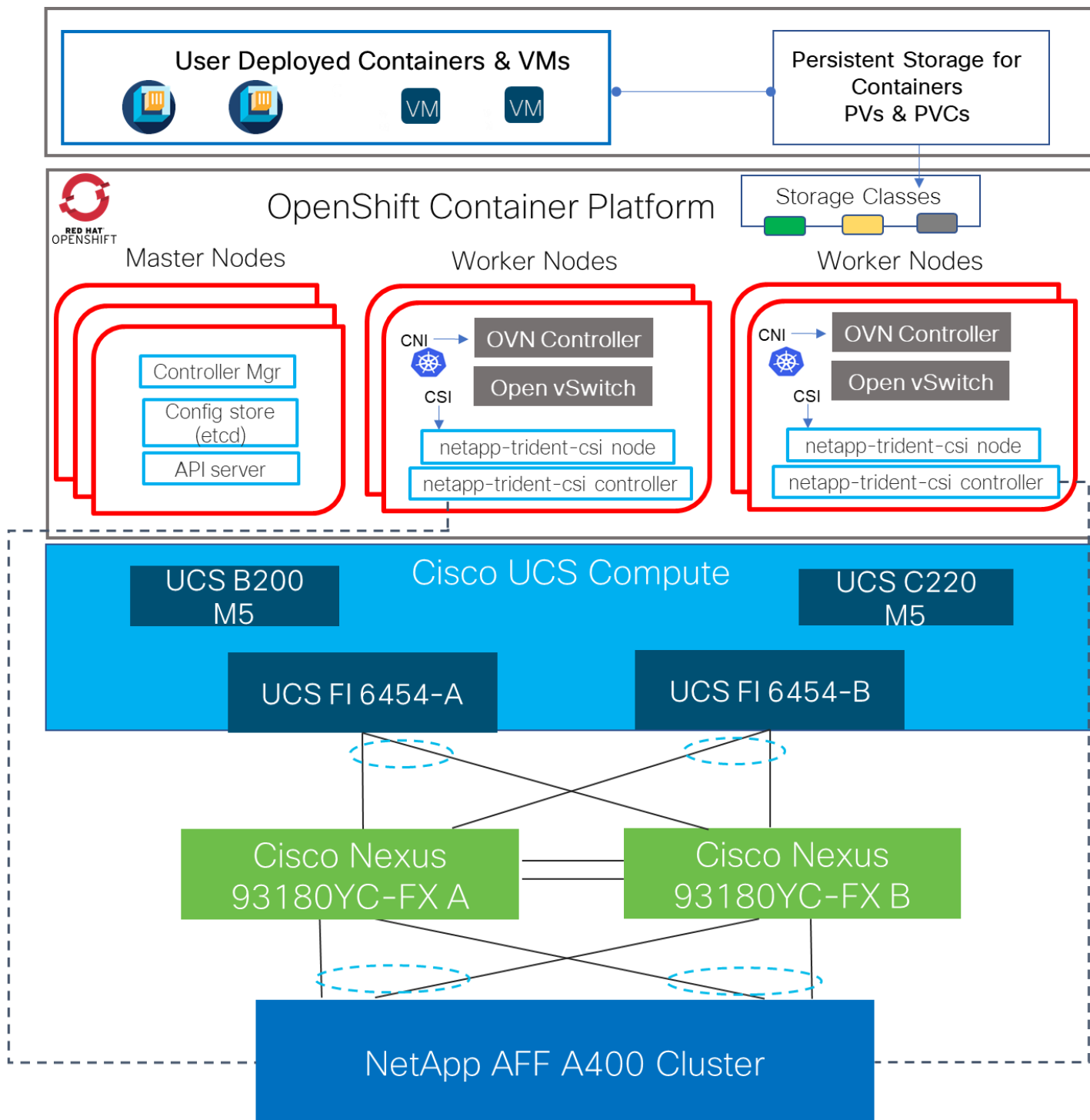
## FlexPod with OCP Logical Topology

[Figure 12](#) illustrates the FlexPod with Red Hat OpenShift Container Platform logical topology with OCP components utilizing compute, network, and storage resources on FlexPod. The storage and network connectivity to Red Hat OpenShift Container Platform Cluster nodes running on Cisco UCS M5 servers is enabled by the Cisco Nexus 9k switches within FlexPod.

Persistent storage is a critical part of running stateful containers, and Red Hat OpenShift Container Platform with Kubernetes simplifies storage management by abstracting details of how storage is provisioned and how it is consumed. Persistent volumes for containers can be static or dynamically provisioned, in this case it is dynamic with FlexPod and is enabled by the NetApp Astra Trident CSI Driver. Dynamic volume provisioning allows storage volumes to be created on-demand, NetApp Astra Trident CSI eliminates the need to pre-provision storage for containers and allows persistent storage provisioning during the container deployment. This solution used NFS and iSCSI storage for dynamic storage provisioning.

OpenShift Container Platform uses a software-defined networking (SDN) approach to provide a unified cluster network that enables communication between pods across the OpenShift Container Platform cluster. This pod network is established and maintained by the OpenShift SDN, which configures an overlay network using Open vSwitch (OVS). The default OpenShift SDN solution is built on top of Open vSwitch (OVS). With OpenShift, the cluster admin can choose to deploy with one of the OpenShift native SDN plug-ins or they can opt to deploy the cluster using a third-party SDN from the supported ecosystem such as Cisco ACI. For this solution, we have used the OpenShift native SDN plug-in (OVN-Kubernetes).

Figure 12. FlexPod for OCP 4 Bare Metal Logical Topology



## FlexPod Network Connectivity and Design

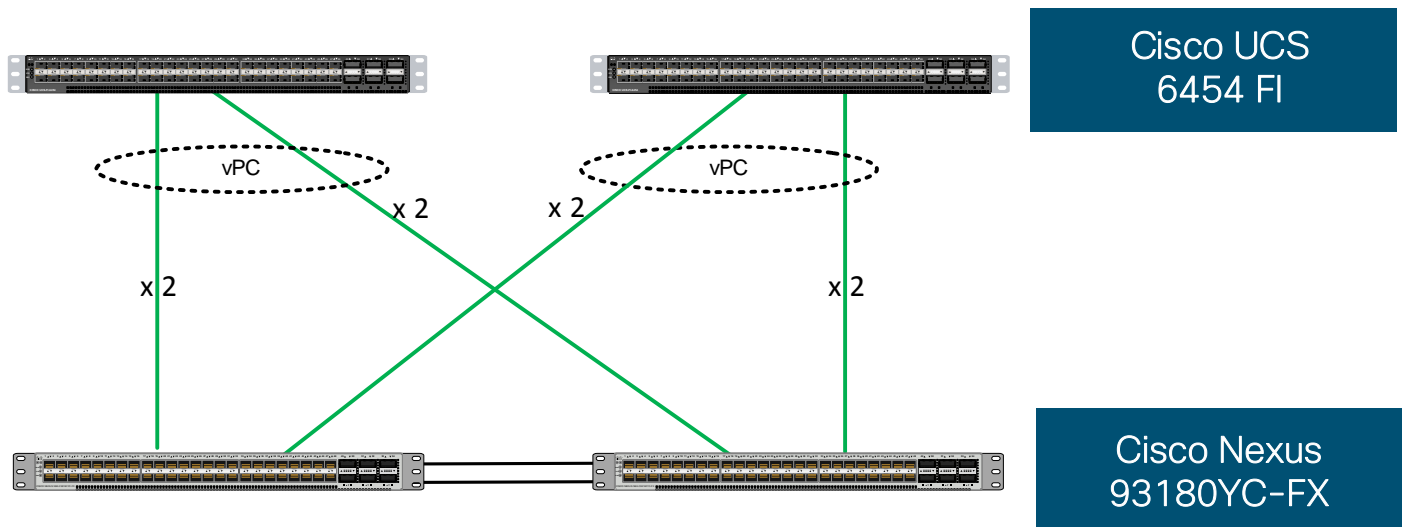
The Layer 2 network connection to each Fabric Interconnect is implemented as Virtual Port Channels (vPC) from the upstream Nexus Switches. In the switching environment, the vPC provides the following benefits:



- Allows a single device to use a Port Channel across two upstream devices
- Eliminates Spanning Tree Protocol blocked ports and use all available uplink bandwidth
- Provides a loop-free topology
- Provides fast convergence if either one of the physical links or a device fails
- Helps ensure high availability of the network

The upstream network switches can connect to the UCS 6454 Fabric Interconnects using 10G, 25G, 40G, or 100G port speeds. In this design, the 25GB ports were tested for the virtual port channels. Virtual port channels were also configured between the Nexus switches and NetApp AFF A400s to also transport the storage traffic between the Cisco UCS servers and the NetApp AFF A400s.

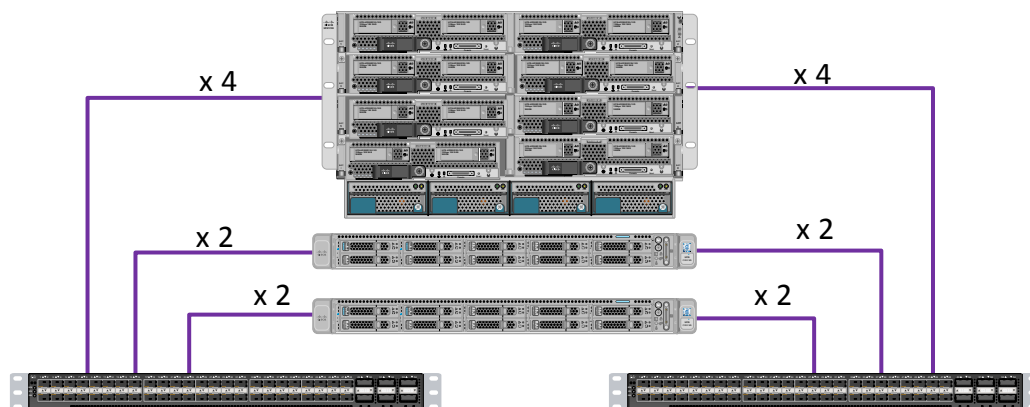
**Figure 13. Network Connectivity - vPC Enabled Connections**



## FlexPod Compute Connectivity

The FlexPod compute design supports both Cisco UCS B-Series and C-Series. Cisco UCS supports the OpenShift environment by providing robust, highly available, and integrated compute resources centrally managed from Cisco UCS Manager in the Enterprise or from Cisco Intersight Software as a Service (SaaS) in the cloud. In this validation effort, multiple Cisco UCS B-Series and C-Series servers are booted from local M.2 SATA SSDs, these drives are configured in Raid 1 using the Cisco Boot Optimized M.2 Raid Controller. The servers have access to NFS and iSCSI storage for persistent storage volumes presented from the NetApp AFF storage cluster.

**Figure 14. Compute Connectivity**



The Cisco UCS chassis in the design are populated with Cisco UCS B200 M5 blade servers and each of these blade servers contain one physical network adapter (Cisco VIC 1440) and a port expander that passes converged fibre channel over Ethernet (FCoE) and Ethernet traffic through the chassis mid-plane to the 2408 FEXs. The FEXs are redundantly connected to the fabric interconnects using 4X25Gbps ports per FEX to deliver an aggregate bandwidth of 200Gbps to the chassis. Full population of each 2408 FEX can support 8x25Gbps ports, providing an aggregate bandwidth of 400Gbps to the chassis. The connections from the Cisco UCS Fabric Interconnects to the FEXs are automatically configured as port channels by specifying a Chassis/FEX Discovery Policy within UCSM.

The Cisco UCS C220 M5 nodes are equipped with Cisco UCS VIC 1457 or Cisco UCS PCIe VIC 1455. Cisco UCS VIC 1455/1457 has four 25GbE ports which are connected to the Cisco UCS 6454 FI in pairs such that ports 1 and 2 are connected to the Cisco UCS 6454 FI-A and the ports 3 and 4 are connected to the FI-B as shown in [Figure 13](#). Optionally, only ports 1 and 3 from each VIC 1455/57 and be connected with network bandwidth reduced from 50 Gbps to 25 Gbps to each FI.

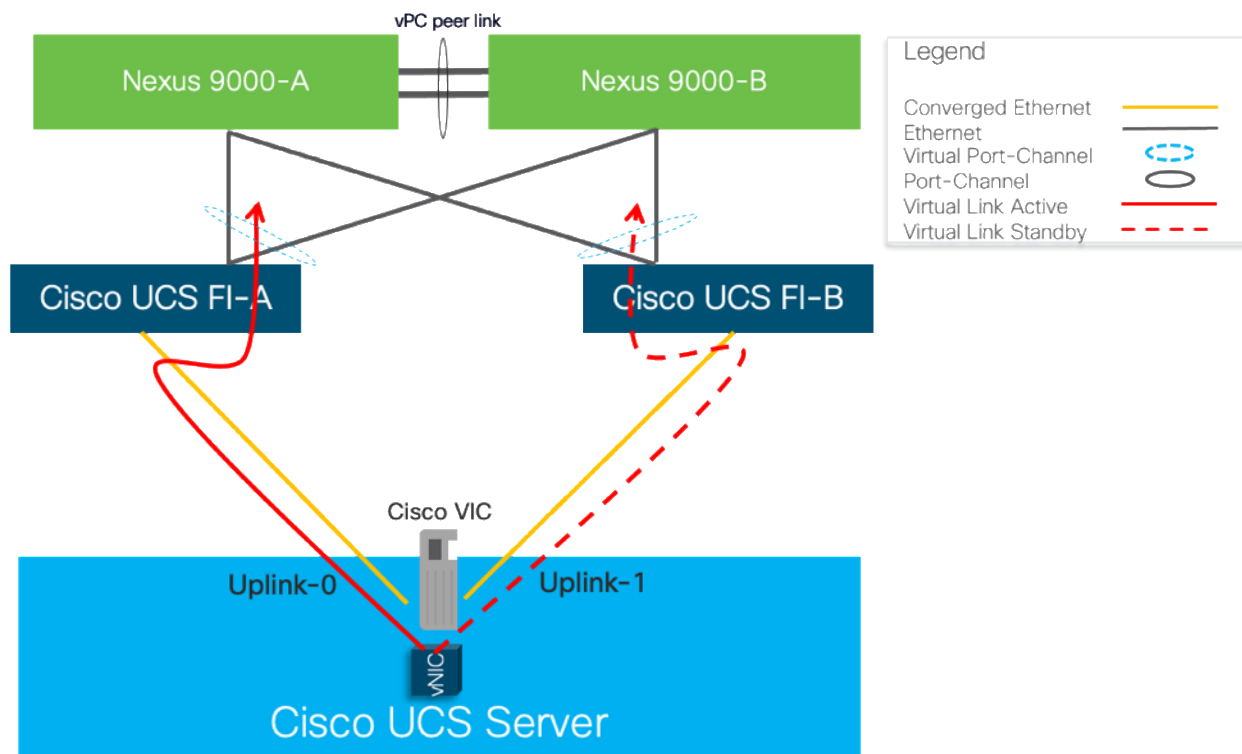
## Fabric Failover for Ethernet: Architecture for High Availability

Cisco UCS is designed for high availability, with no single point of failure in its network infrastructure. Each adapter in Cisco UCS connects to both fabrics (A and B) and the fabric interconnects are designed to work in an active-active model, with automated failover of network in the event of a failure. The system is designed so that if either fabric A or fabric B fails, the remaining fabric can take on the traffic from the failed fabric. Cisco VICs support fabric failover by moving traffic from one fabric to the other according to failover policies established on a per-vNIC basis. This eliminates complicated operating system NIC teaming configurations. Fabric failover makes the failure of a fabric transparent to the operating system.

[Figure 15](#) illustrates the UCS Fabric failover mechanism, in this example one vNIC connects to fabric A but fails over to fabric B in the event of a fabric failover.

Cisco UCS fabric failover is an important feature because it reduces the complexity of defining NIC teaming software for failover on the host. It does this transparently in the fabric based on the network property that is defined in the service profile. With Cisco UCS fabric failover, NIC teaming is not necessary on the OCP nodes, and the high availability is managed at the UCS level more efficiently.

**Figure 15. Cisco UCS Fabric Failover**



### NIC Bonding versus Cisco UCS Fabric Failover

OpenShift Container Platform network requirements in this design are standard Ethernet only, while OCP 4 deployment can work with two network interfaces in bonded mode for each traffic type (bare metal public VLAN and VM network VLAN), it is recommended to use a single network interface for each traffic type and enable Cisco UCS Fabric Failover for resiliency versus NIC bonding in the operating system. With Cisco UCS Fabric Failover the management and operation of failover and link aggregation is handled in the networking fabric. The Fabric Failover is enabled in the vNIC templates within the Cisco UCS service profiles which makes it easy to implement NIC resiliency across any number of servers managed by Cisco UCS, this eliminates the need to configure every server individually.

NIC teaming is often implemented to aggregate lower-speed NICs in order to gain throughput. Since OCP design with Cisco UCS leverages 25/40GbE connections, aggregation is generally not required.

### Service Profile Configuration

The Cisco UCS servers are deployed using Cisco UCS Service Profiles (SP) that consists of server identity information pulled from pools (WWPN, MAC, UUID, and so on) as well as policies covering connectivity, firmware, and power control options, and so on. The service profiles are provisioned from the Cisco UCS Service Profile Templates that allow rapid creation, as well as guaranteed consistency of the hosts at the Cisco UCS hardware layer.

### Service Profile for OCP Hosts

In FlexPod deployments, each Cisco UCS server (B-Series or C-Series), equipped with a Cisco Virtual Interface Card (VIC), is configured for multiple virtual interfaces (vNICs) which appear as standards-compliant PCIe device-

es to the OS. The service profile configuration for an OCP host is shown in [Table 1](#), [Figure 16](#), and [Figure 17](#) for OCP Worker and Master nodes respectively.

Each OCP host service profile supports:

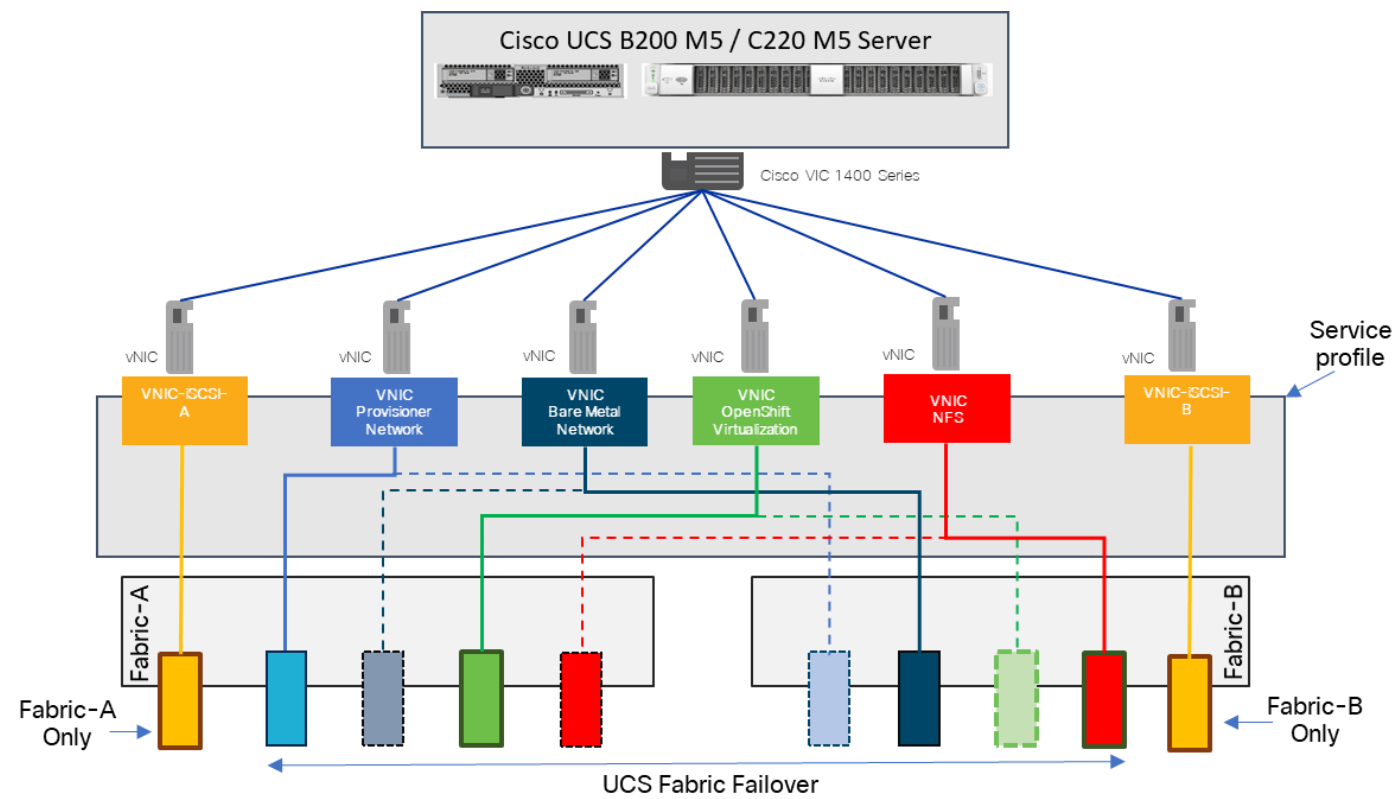
- Managing the OCP hosts using a common management segment
- OS local boot using the mirrored M.2 onboard drives on the Cisco UCS servers (boot from SAN was not supported with OCP nodes at the time of this validation).
- Six vNICs (NFS and iSCSI vNICs are only required on the Worker Nodes) used as follows in the same order specified below:
- One vNIC for provisioning traffic to support OCP installer provisioned infrastructure. The MTU value for this interface is set as a Jumbo MTU (9000). This should be the first vNIC on the UCS servers and this supports provisioning network which is non-routable network used for provisioning the underlying operating system (RHCOS) on each node that is part of the OpenShift Container Platform Cluster.
- One vNIC for OCP Public Bare Metal Network traffic. The baremetal network is a routable network. The second vNIC on the UCS servers is used to support the baremetal network.
- One vNIC for OpenShift Virtualization VM Network traffic.
- One NFS vNIC for NFS storage traffic. The MTU value for this interface is set as a Jumbo MTU (9000).
- One iSCSI-A vNIC utilizes iSCSI-A VLAN (defined only on Fabric A) to provide access to iSCSI-A path. The MTU value for this interface is set as a Jumbo MTU (9000).
- One iSCSI-B vNIC utilizes iSCSI-B VLAN (defined only on Fabric B) to provide access to iSCSI-B path. The MTU value for this interface is set as a Jumbo MTU (9000).

**Table 1. OCP Host Service Profile**

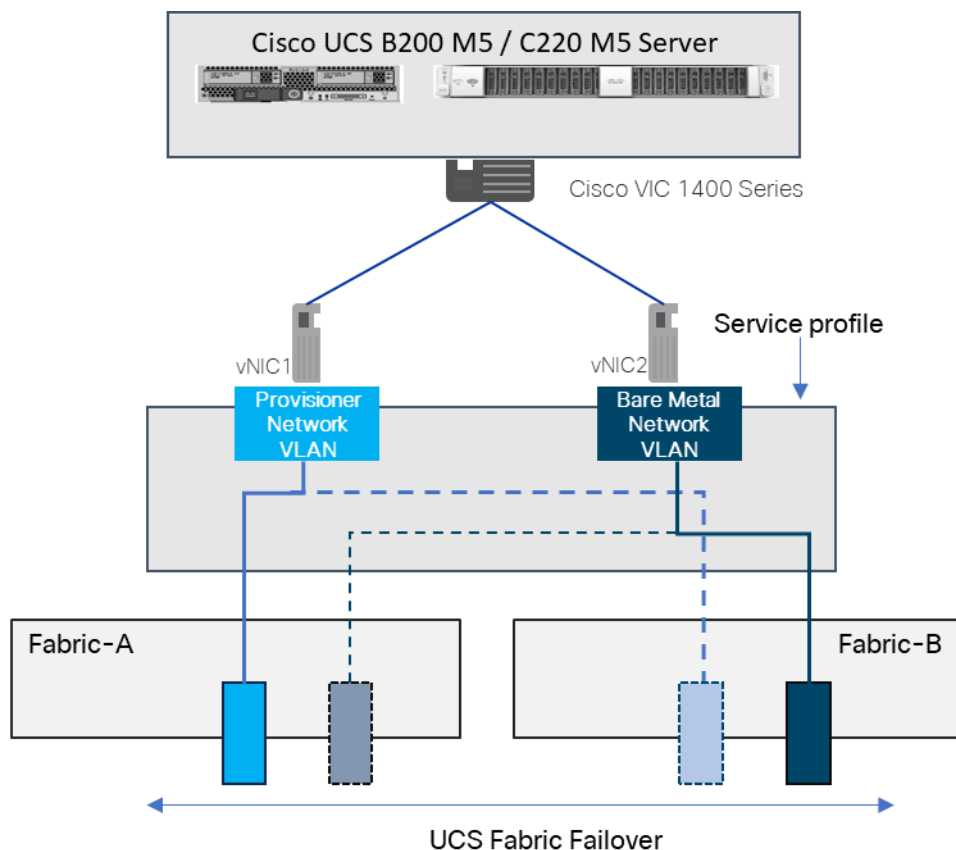
Machine	Provisioning Traffic	BareMetal Public Traffic	OpenShift Virtualization	NFS	iSCSI-A	iSCSI-B
<b>Provisioner Node</b>	vNIC1  Fabric-A failover to Fabric-B	vNIC2  Fabric-B failover to Fabric-A	vNIC3  Fabric-A failover to Fabric-B	vNIC4  Fabric-B failover to Fabric-A	vNIC5  Fabric-A Only	vNIC6  Fabric-B Only node
<b>Master Node</b>	vNIC1  Fabric-A failover to Fabric-B	vNIC2  Fabric-B failover to Fabric-A	NA	NA	NA	NA
<b>Worker Node</b>	vNIC1  Fabric-A failover to Fabric-B	vNIC2  Fabric-B failover to Fabric-A	vNIC3  Fabric-B failover to Fabric-A	vNIC4  Fabric-B failover to	vNIC5  Fabric-A Only	vNIC6  Fabric-B Only node

Machine	Provisioning Traffic	BareMetal Public Traffic	OpenShift Virtualization	NFS	iSCSI-A	iSCSI-B
				Fabric-A		

Figure 16. Cisco UCS - Server Interface Design for OCP Worker Nodes



**Figure 17. Cisco UCS - Server Interface Design for OCP Master Nodes**



## FlexPod Storage Design for OCP

The FlexPod for OCP uses the NetApp Astra Trident CSI driver that is an add-on component that needs to be installed on the OpenShift Container Platform cluster. Astra Trident enables the integration between the storage and OCP cluster.

The NetApp AFF A400 storage array supports both NFS and iSCSI protocols. For the purpose of this validated design, both NFS and iSCSI were used for dynamic persistent storage for Containers and VMs.

### Dynamic Storage Provisioning

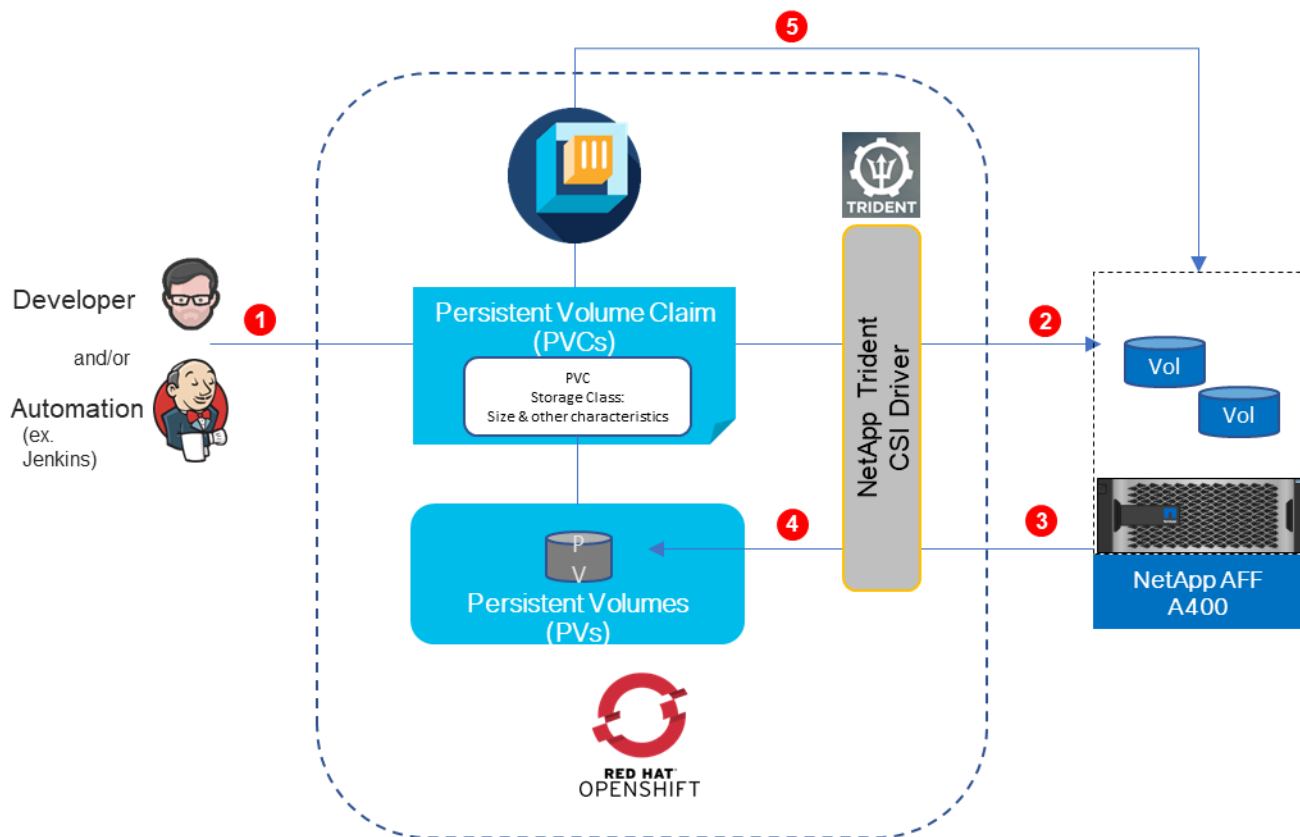
OpenShift provides dynamic provisioning of storage for applications by utilizing the StorageClass resource. Using dynamic storage, you can select different types of back-end storage. The back-end storage is segregated into different tiers depending on the needs of your application. When requesting storage, a user can specify a PersistentVolumeClaim with an annotation that specifies the value of the StorageClass they prefer.

To order the storage, you must create a PVC. The PVC determines the specification for the storage that you want to provision. After the PVC is created, the storage device and the PV are automatically created for you.

[Figure 18](#) shows how block storage is dynamically provisioned in a cluster. This sample flow works similarly with other storage types, such as file storage.



Figure 18. Dynamic Storage Provisioning Workflow



Developer/Automation submits storage requirements in the form of standard Persistent Volume Claims that specifies the storage type, storage class, size, and so on.

NetApp Astra Trident CSI Plugin listens to and intercepts Persistent Volume Claims based on Storage Class. Creating a PVC in a cluster automatically triggers the storage plug-in for the requested type of storage to provision storage with the given specification.

Storage provisioning API call sent to NetApp AFF A400, and storage is provisioned.

The storage plug-in automatically creates a persistent volume (PV) in the cluster, a virtual storage device that points to the actual storage device on your NetApp AFF A400.

The PVC and PV are automatically connected to each other. The status of the PVC and the PV changes to Bound and the PVC is used to mount persistent storage to your app. If you delete the PVC, the PV and related storage instance are also deleted.

## OCP Virtual Switching Architecture

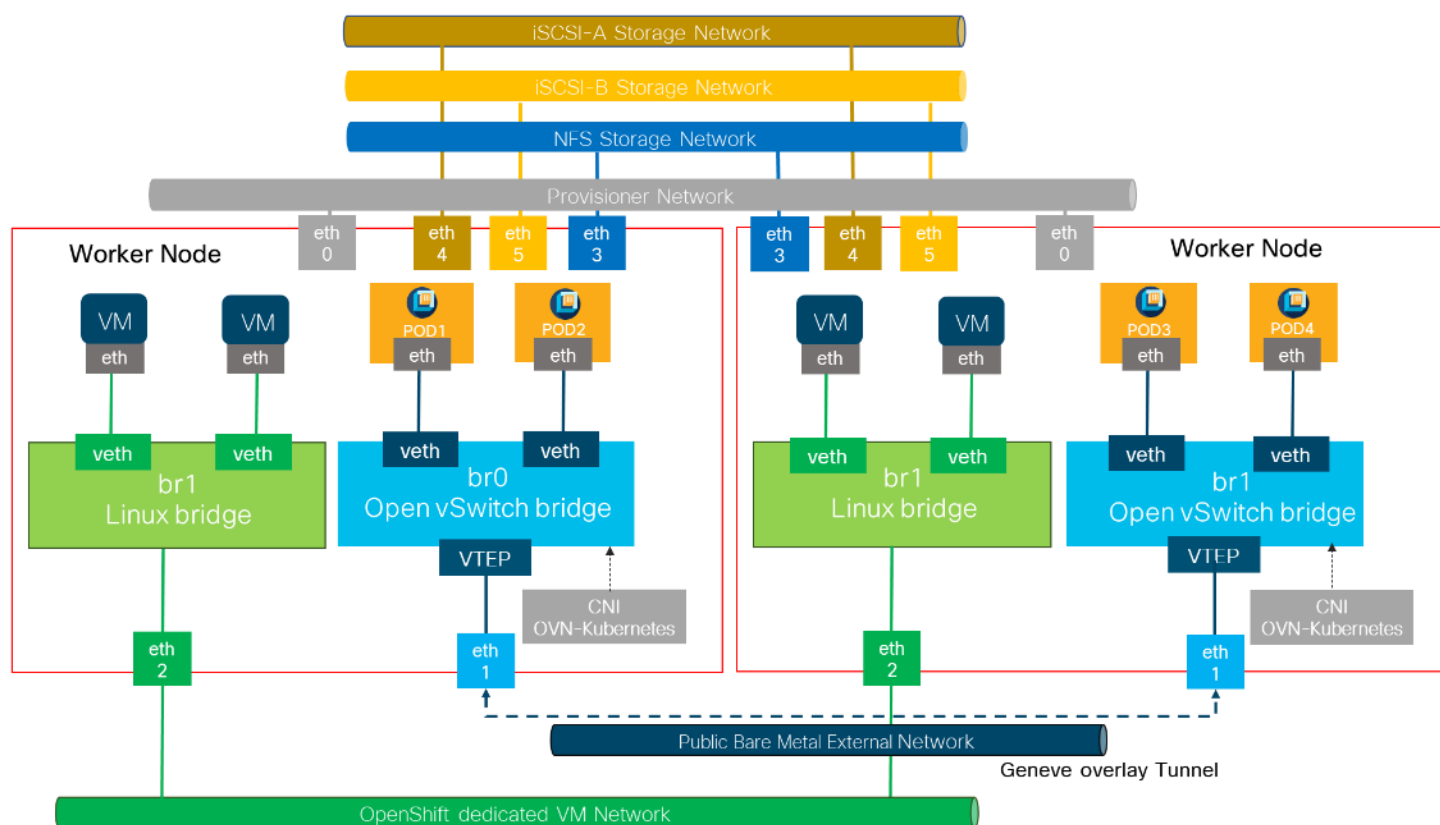
The OpenShift Container Platform cluster uses a virtualized network for pod and service networks. The OVN-Kubernetes Container Network Interface (CNI) plug-in is a network provider for the default cluster network. A cluster that uses the OVN-Kubernetes network provider also runs Open vSwitch (OVS) on each node. OVN configures OVS on each node to implement the declared network configuration.

The OVN-Kubernetes default Container Network Interface (CNI) network provider implements the following features:

- Uses OVN (Open Virtual Network) to manage network traffic flows. OVN is a community developed, vendor agnostic network virtualization solution.
- Implements Kubernetes network policy support, including ingress and egress rules.
- Uses the Geneve (Generic Network Virtualization Encapsulation) protocol rather than VXLAN to create an overlay network between nodes.

[Figure 19](#) shows the distribution of network interfaces on each OCP worker node with one Open vSwitch bridge, for Pod-to-Pod communication and the other dedicated Linux bridge created for VM external network when VMs are deployed leveraging the OpenShift Virtualization feature. Each bridge has one NIC within the OS providing access to external networks. The other four network interfaces on the OCP nodes are used for communication via the provisioning network and access to NFS, iSCSI-A, and iSCSI-B traffic via dedicated interfaces. Appropriate VLANs are enabled at the UCS level to support different traffic types.

**Figure 19. Virtual Switching and Connectivity Diagram for a Cisco UCS M5 OCP Host**



With OpenShift Virtualization, each VM deployed is controlled via a virt-launcher pod that is created with each VM. The default networking type for OpenShift Virtualization VMs is Masquerade. The VM will be assigned a non-routable IP and you can access the VM using the IP of the virt-launcher pod that was deployed alongside it. This makes the VM accessible in the same way that containers are accessed.

---

Alternatively, you can connect the VM to the host network by creating a bridge interface on the OCP nodes using Nmstate. The Nmstate operator is installed with OpenShift Virtualization and provides you with the Node Network Configuration Policy (NNCP) object to update the host network settings. [Figure 19](#) has a sample config bridge called br1 created from an interface called eth2 (the interface name differs based on how the host views it) on the OCP nodes.

## Solution Deployment

### Deployment Hardware and Software

The deployment of hardware and software for FlexPod with OpenShift Container Platform is detailed in the following sections.

#### Software Revision

[Table 2](#) lists the software versions for hardware and virtual components used in this solution. Each of these versions been used have been certified within interoperability matrixes supported by Cisco and NetApp. For more current supported version information, consult the following sources:

- [NetApp IMT](#) (NetApp Support Login Required)
- [Cisco UCS Hardware and Software Interoperability Tool](#) (Cisco ID Required)

Additionally, it is also strongly suggested to align FlexPod deployments with the recommended release for the Cisco Nexus 9000 switches used in the architecture:

- [Recommended Cisco NX-OS Releases for Cisco Nexus 9000 Series Switches](#)

**Table 2. Hardware and Software Revisions**

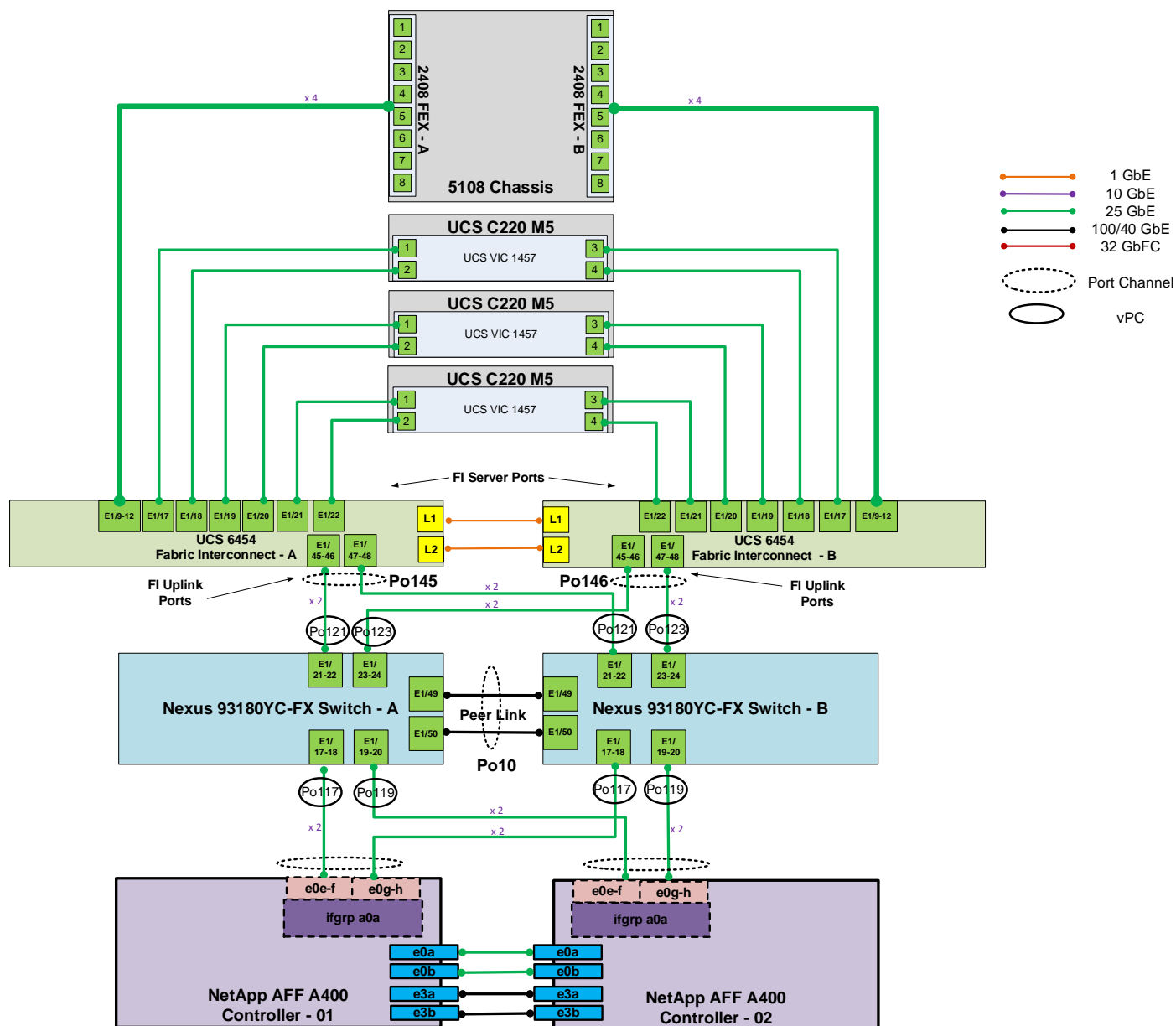
Layer	Device	Image	Comments
Compute	Cisco UCS Fabric Interconnects 6400 Series, Cisco UCS B200 M5, Cisco UCS C220 M5	4.1(3b)	Includes the Cisco UCS-IOM 2408, Cisco UCS Manager, Cisco UCS VIC 1440, and Cisco UCS VIC 1457/1455
Network	Cisco Nexus Switches	9.3(7)	Nexus Switches
Storage	NetApp AFF A400 ONTAP	9.8	Software version
Software	OpenShift Container Platform	4.7	Software version
	OpenShift Virtualization	2.5	Software version
	OCP Master Node	RHCOS 4.7	OS version
	OCP Worker Node	RHCOS 4.7	OS version
	Provisioner Node	RHEL 8.3	OS version
	NetApp Astra Trident CSI Plugin	21.04	Software version

#### Physical Cabling for FlexPod with OCP Bare Metal

The information in this section is provided as a reference for cabling the physical equipment in a FlexPod environment. Customers can adjust the ports according to their individual setup but when done so the variables must be changed in the Ansible automation Playbooks. This document assumes that out-of-band management ports are plugged into an existing management infrastructure at the deployment sites. The interfaces shown in [Figure](#)

[20](#) will be used in various configuration steps. Additional 1Gb management connections will be needed for an out-of-band network that sits apart from the FlexPod infrastructure. Each Cisco UCS fabric interconnect, Cisco Nexus switch and NetApp storage controller is connected to the out-of-band network. Layer 3 network connectivity is required between the Out-of-Band (OOB) and In-Band (IB) Management Subnets.

**Figure 20. Physical Cabling for FlexPod with OCP 4 Bare Metal**



## VLANs Configuration

To enable connectivity between various layers of the FlexPod system and to provide external connectivity to the OCP cluster and workloads deployed in the form of Containers and VMs, several VLANs are configured and enabled on various paths.

[Table 3](#) list VLANs configured for setting up the FlexPod environment along with their usage.

**Table 3. VLANs**

VLAN ID	Name	Usage
2	Native-VLAN	Use VLAN 2 as Native VLAN instead of default VLAN (1)
13	OOB-MGMT-VLAN	Out of Band Management VLAN to connect the management ports for various devices
914	OpenShift Virtualization	Dedicated VLAN for OpenShift Virtualization Public Traffic
904	OCP Provisioning	VLAN for OpenShift Provisioning Network
113	OCP Bare Metal	VLAN for OpenShift Bare Metal Public Network
3154	NFS	NFS VLAN for provisioning NFS persistent volumes
3014	iSCSI-A	iSCSI-A VLAN for provisioning iSCSI persistent volumes
3024	iSCSI-B	iSCSI-B VLAN for provisioning iSCSI persistent volumes

Some of the key highlights of VLAN usage are as follows:

- VLAN 13 allows customers to manage and access out of band management interfaces of various devices.
- VLAN 113 is used for all the OCP infrastructure (DNS, DHCP, and so on) as well as OCP cluster nodes.
- VLAN 904 is used by the VMs deployed on the OpenShift Cluster for public access
- VLAN 3154 is configured to provide access to NFS storage on the NetApp storage cluster.
- A pair of iSCSI VLANs (3014 and 3024) are configured to provide access to iSCSI storage on the NetApp storage cluster.

## Bare Metal Compute Options and Sizing

[Table 4](#) lists the Cisco UCS compute options that are supported for OCP installation.

**Table 4. Cisco UCS Server Node Configuration Options**

Server Node	Cisco UCS B200 M5	Cisco UCS C220 M5
CPU	2x 2 <sup>nd</sup> Gen Intel® Xeon® Scalable	2x 2 <sup>nd</sup> Gen Intel® Xeon® Scalable
Memory	DDR4-2933-MHz	DDR4-2933-MHz

Server Node	Cisco UCS B200 M5	Cisco UCS C220 M5
Storage (Boot Drives)	(2) M.2 SATA SSD - RAID1	(2) M.2 SATA SSD - RAID1
Storage Controller	Cisco Boot optimized M.2 RAID controller (UCS-M2-HWRAID)	Cisco Boot optimized M.2 RAID controller (UCS-M2-HWRAID)
Network	VIC 1440 + Port Expander	VIC 1457/1455

The deployment procedure in this document covers OCP cluster installation on Cisco UCS B200M5 and Cisco UCS C220M5 servers. In the lab validation, 3 Cisco UCS B200 M5 servers were used for the master nodes and 3 Cisco UCS C220 M5 servers were used for the worker nodes.

## Sizing

This is a general recommendation and not specific to a customer environment. It is important to properly size the solution with all of its components by a qualified Engineer or Architect per the specific requirements of the customer. There is no one size fits all approach, hence specific sizing and performance testing were excluded from the CVD.

For example, at the Cisco UCS level, customers have the option to include servers with different processors and core counts, and with the combination of the right amount of memory the servers can be optimized for the right cost-performance configuration. The same strategy is applicable across all the layers of FlexPod including network and storage.

It is important to size the servers to meet the minimal requirements of the OCP platform, to account for failures of servers and by that to make sure that OCP HA related rules can be followed upon server failure with enough resources available for OCP to redistribute the workloads from the failing host or when performing upgrades and other maintenance tasks.

## Example Sizing Guidelines (Worker Nodes)

Determine how many nodes and pods are required for your OpenShift Container Platform cluster. Cluster scalability correlates to the number of pods in a cluster environment. That number influences the other numbers in your setup. See [Cluster Limits](#) for the latest limits for objects in OpenShift Container Platform.

Environment sizing can be done according to tested cluster maximums or according to your application requirements. While planning your environment, determine how many pods are expected to fit per node:

Required Pods per Cluster / Pods per Node = Total Number of Nodes Needed

If you want to scope your cluster at 2500 pods, assuming the 250 maximum pods per node, you will need at least ten nodes:

$2500 / 250 = 10$

If you increase the number of nodes to 15, the pods per node distribution changes to 167 pods per node.



The current maximum number of pods per node is 250. However, the number of pods that fit on a node is dependent on the application itself. Consider the application's memory, CPU, and storage requirements.

[Table 5](#) lists components you might consider for a sample application environment.

**Table 5. Environment Components**

Pod type	Pod quantity	Max memory	CPU cores	Persistent storage
apache	100	500 MB	0.5	1 GB
node.js	200	1 GB	1	1 GB
postgresql	100	1 GB	2	10 GB

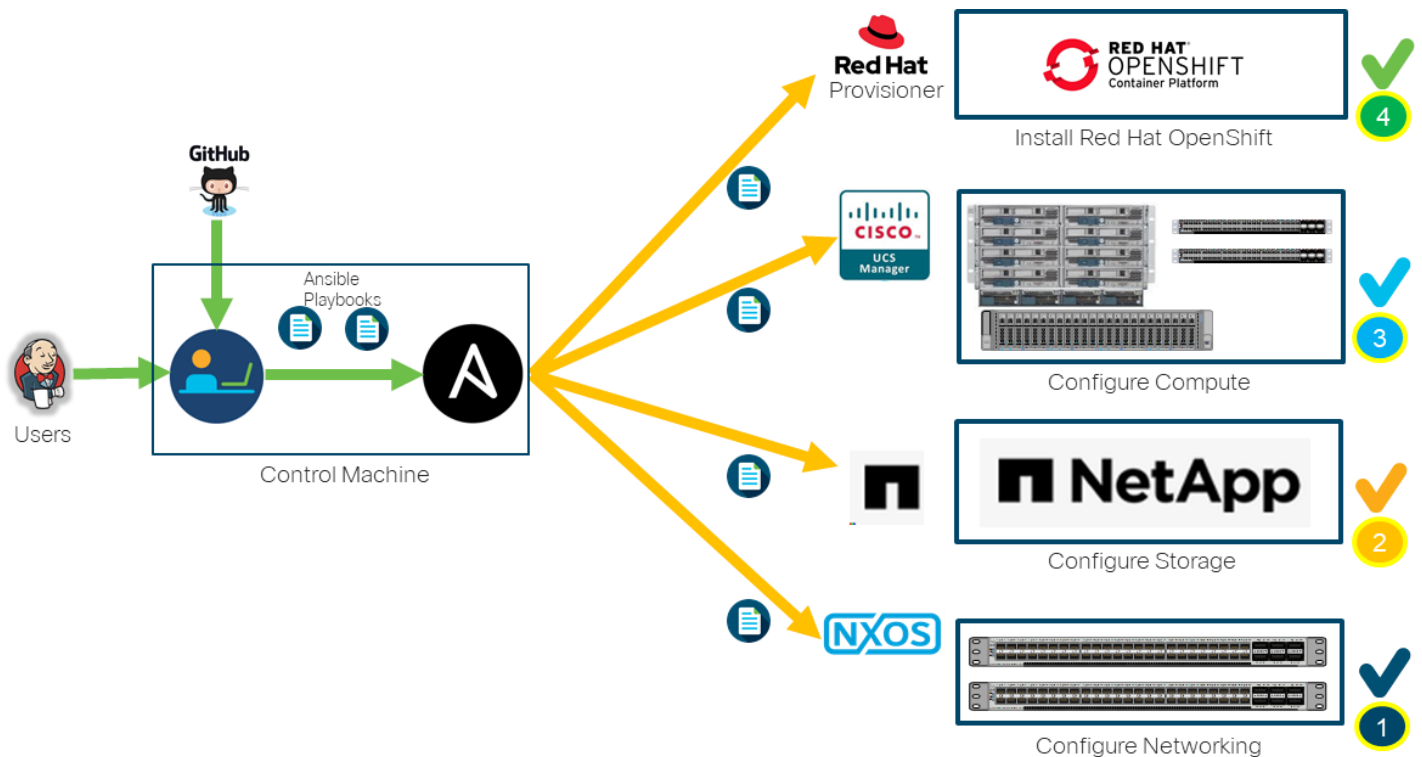
The overall resource requirements for this application are: 450 CPU cores, 350GB RAM, and 1.3TB storage plus overhead required for OCP operations.

## Ansible Automation Workflow and Solution Deployment

The FlexPod with OCP 4 bare metal solution uses a management workstation (control machine) to run Ansible playbooks to configure Cisco Nexus, Cisco UCS, NetApp ONTAP Storage and Install the OCP Cluster.

[Figure 21](#) illustrates the FlexPod with OCP 4 bare metal solution implementation workflow which is explained in the following sections. The FlexPod infrastructure layers are first configured in the order illustrated in the following figure before installing the OCP on the UCS bare metal cluster.

Figure 21. Ansible Automation Workflow



## Prerequisites

Setup of the solution begins with a management workstation that has access to internet and with a working installation of Ansible. The management workstation commonly runs a variant of Linux or MacOS for ease of use with these command-line-based tools. Instructions for installing the workstation are not included in this document, but basic installation and configuration of Ansible is covered. A guide for getting started with Ansible can be found at the following link:

- Getting Started with Red Hat Ansible: <https://www.ansible.com/resources/get-started>

To use the Ansible playbooks demonstrated in this document, the management workstation must also have a working installation of Git and access to the Cisco DevNet public GitHub repository. The Ansible playbooks used in this document are cloned from the public repositories, located at the following links:

- Cisco DevNet: <https://developer.cisco.com/codeexchange/github/repo/ucs-compute-solutions/FlexPod-IaC-OCP4>
- GitHub repository: <https://github.com/ucs-compute-solutions/FlexPod-IaC-OCP4>
- Red Hat OpenShift repository for IPI automated install: <https://github.com/openshift-kni/baremetal-deploy.git>

The Cisco Nexus Switches, NetApp Storage and Cisco UCS must be physically racked, cabled, powered, and configured with the management IP addresses before the Ansible-based installation procedure can begin as shown in the cabling diagram (Figure 20). If necessary, upgrade the Cisco Nexus Switches to release 9.3(7) and the Cisco UCS System to 4.1(3b) with the default firmware packages for both blades and rack servers set to 4.1(3b).

---

Before running each Ansible Playbook to setup the Network, Storage, Cisco UCS, and install OCP various variables must be updated based on the customers environment and specific implementation with values such as the VLANs, pools and ports on Cisco UCS, IP addresses for NFS and iSCSI interfaces and values needed for the OCP installation.



Day 2 Configuration tasks such as OpenShift Virtualization configuration and Astra Trident installation after the OCP cluster is installed have been performed manually and the information has been provided in the later sections of this document.

---

## Prepare Management Workstation (Control Machine)

In this section, the installation steps are performed on the CentOS management host to prepare the host for solution deployment to support the automation of Cisco UCS, Cisco Nexus, NetApp Storage and OCP installation using Ansible Playbooks.



The following steps were performed on a RHEL 8.3 Virtual Machine.

---

To prepare the management workstation, follow these steps:

1. Install EPEL repository on the management host.

```
[root@fp-ansible ~]# dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
```

2. Install Ansible engine.

```
[root@fp-ansible ~]# dnf install ansible
```

3. Verify Ansible version to make sure it is at a minimum release 2.9.

```
[root@fp-ansible ~]# ansible --version
ansible 2.9.21
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Mar 18 2021, 08:58:41) [GCC 8.4.1 20200928 (Red Hat 8.4.1-1)]
```

4. Install UCS SDK.

```
[root@fp-ansible ~]# pip3 install ucsm-sdk
```

5. You should be able to SSH into each of the Cisco Nexus switches that will be configured using Ansible so that the SSH keys are cached.

```
[root@fp-ansible ~]# ssh admin@192.168.156.21
```

The authenticity of host '192.168.156.21 (192.168.156.21)' can't be established.

---

```
RSA key fingerprint is SHA256:YWSl7OaDF7VbOqg9ImRTY2bwFXIrajHAKd/xoOwBCgk.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.156.21' (RSA) to the list of known hosts.
User Access Verification
Password:
```

6. Install NetApp specific python modules.

```
[root@fp-ansible ~]# pip3 install netapp-lib
```

7. Install ansible-galaxy collections for Cisco UCS, Cisco Nexus and NetApp as follows:

```
[root@fp-ansible ~]# ansible-galaxy collection install cisco.nxos
```

```
[root@fp-ansible ~]# ansible-galaxy collection install cisco.ucs
```

```
[root@fp-ansible ~]# ansible-galaxy collection install netapp.ontap
```

8. Install the following python modules required for OCP installation using Ansible:

- python3-dns
- uri
- python3-netaddr

```
[root@fp-ansible ~]# dnf install python3-dns
[root@fp-ansible ~]# pip3 install uri
[root@fp-ansible ~]# dnf install python3-netaddr
```



We validated the Ansible automation with both python 2.7.5 and python 3.6 as the python interpreter for Ansible.

---

## Clone GitHub Collection

You will use GitHub repos from two public locations, the first step in the process is to clone the GitHub collection named flexpod-IaC-OCP4 (<https://github.com/ucs-compute-solutions/FlexPod-IaC-OCP4>) and baremetal-deploy (<https://github.com/openshift-kni/baremetal-deploy>) to new empty folders on the management workstation. Cloning the collections creates a local copy, which is then used to run the playbooks that have been created for this solution. To clone the GitHub collection, follow these steps:

1. From the management workstation, create a new folder for the project. The GitHub collection will be cloned in a new folder inside this one, named ocp4.
2. Open a command-line or console interface on the management workstation and change directories to the new folder just created.
3. Clone the GitHub collection using the following commands:

```
git clone https://github.com/ucs-compute-solutions/FlexPod-IaC-OCP4.git
```

```
git clone https://github.com/openshift-kni/baremetal-deploy.git
```

4. Change directories to the new folder named ocp4.
5. Change directories to the folder named FlexPod-IaC-OCF4.

## FlexPod Deployment using Playbooks

This section explains the installation and configuration of all the infrastructure layers with in FlexPod. The Ansible Playbook tree structure is shown below with the directory structure and various roles and tasks:

```
├── group_vars
│   ├── all.yml
│   ├── nexus.yml
│   ├── ontap
│   └── ucs.yml
├── host_vars
│   ├── n9kA.yml
│   └── n9kB.yml
├── inventory
├── LICENSE
├── licenses.yaml
├── README.md
├── roles
│   ├── NEXUSconfig
│   │   ├── defaults
│   │   │   └── main.yml
│   │   └── tasks
│   │       ├── configure_default_gw.yml
│   │       ├── configure_nxos_features.yml
│   │       ├── configure_nxos_global_settings.yml
│   │       ├── configure_nxos_ntp.yml
│   │       ├── configure_nxos_vlans.yml
│   │       ├── configure_nxos_vpc.yml
│   │       ├── initiate_nxos_config_backup.yml
│   │       ├── main.yml
│   │       ├── save_nxos_config.yml
│   │       └── set_nxos_interfaces.yml
│   └── ONTAP
│       ├── ontap_lifs
│       │   └── tasks
│       │       └── main.yml
│       └── ontap_network
```



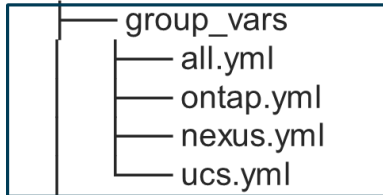
```
| | | ├── configure_ucs_NCP.yml
| | | ├── configure_ucs_OCP_LCP.yml
| | | ├── configure_ucs_system_qos.yml
| | | ├── configure_ucs_vlans.yml
| | | ├── configure_ucs_vnic_templates.yml
| | | └── main.yml
| ├── UCSserver
| | ├── defaults
| | | └── main.yml
| | └── tasks
| | | ├── add_ucs_pmem_policy_to_SPT.yml
| | | ├── configure_ucs_app_direct_pmem_policy.yml
| | | ├── configure_ucs_bios_policy.yml
| | | ├── configure_ucs_boot_policy.yml
| | | ├── configure_ucs_ipmi_profile.yml
| | | ├── configure_ucs_PCP.yml
| | | ├── configure_ucs_server_pool.yml
| | | ├── configure_ucs_SPT.yml
| | | ├── configure_ucs_uuid_pool.yml
| | | ├── main.yml
| | | └── update_ucs_maintenance_policy.yml
└── UCSstorage
    ├── defaults
    |   └── main.yml
    └── tasks
        ├── configure_ucs_disk_group_policy.yml
        ├── configure_ucs_storage_profile.yml
        └── main.yml
Setup_LAN_Connectivity.yml
Setup_Lic.yaml
Setup_Nexus.yml
Setup_ONTAP.yml
Setup_UCS.yml
vars
└── ontap main.yml
```

The following information must be modified based on your environment; more information needs to be modified specific to each device automation which is explained later in this document in the device automation sections.

- inventory – contains the variables such as device IP addresses and authentication details:



- group\_vars/all.yml – contains the VLAN ids required for the solution deployment, update this file based on your environment
- vars/ontap\_main.yml – contains the variables that are required for the configuration of NetApp ONTAP



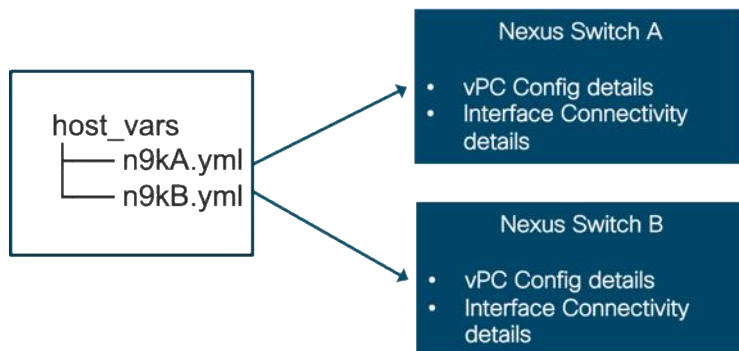
## FlexPod Network Configuration



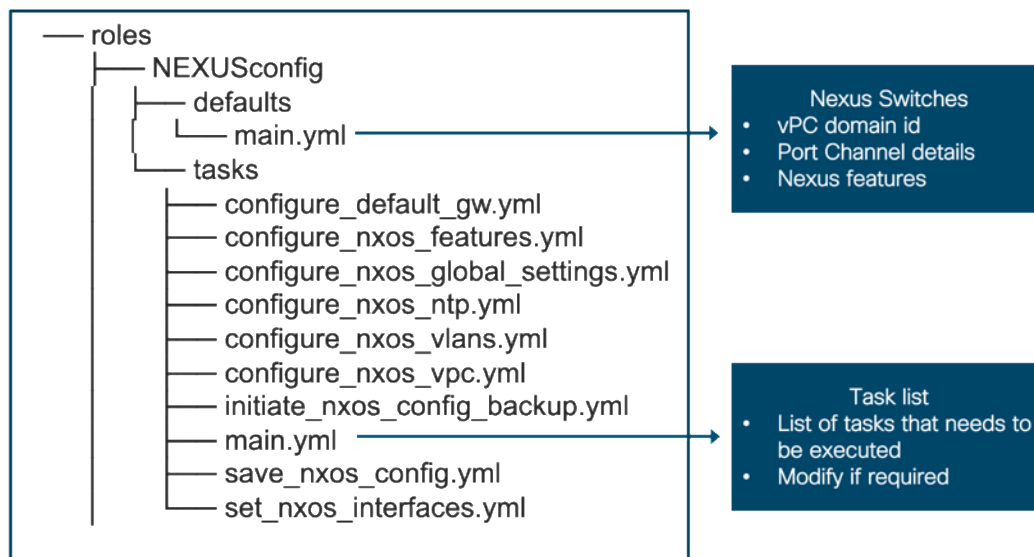
Make sure the FlexPod cabling and initial configuration has been completed on the Cisco Nexus switches.

The following information must be modified based on your specific environment, before running the Nexus Automation Playbook:

- Switch Interface details in the following files if using different ports.



- vPC domain id, Port Channel details and Nexus features in the following files if using different port channel ids or features.



Once the information has been updated in the respective files, run the Nexus switch Ansible playbook:

```
[root@fp-ansible FlexPod-IaC-OCF4]# ansible-playbook ./Setup_Nexus.yml -i inventory
```

Login into the Nexus switches and verify the configuration has been completed as desired before proceeding with the next section to configure NetApp Storage and Cisco UCS.

## FlexPod Storage Configuration

Prior to executing the Setup\_ONTAP.yml playbook to configure the ONTAP storage system as per the requirements of OCP 4, the following operations must be completed:

- Physical rack and stack of hardware
- Cabling and Power-on
- Initialization of ONTAP OS - disk assignment
- Cluster Create - Node Management IPs and ONTAP Cluster with IP and password



Aggregate creation is part of the automation.

## ONTAP Initial Setup

From a console port program attached to the storage controller A (node 01) console port, run the node setup script. This script appears when ONTAP 9.8 boots on the node for the first time.

To setup ONTAP, follow these steps:

1. Follow the prompts to set up node 01.

Welcome to node setup.

You can enter the following commands at any time:

---

"help" or "?" - if you want to have a question clarified,  
"back" - if you want to change previously answered questions, and  
"exit" or "quit" - if you want to quit the setup wizard.  
Any changes you made before quitting will be saved.

You can return to cluster setup at any time by typing "cluster setup".  
To accept a default or omit a question, do not enter a value.

This system will send event messages and weekly reports to NetApp Technical Support.  
To disable this feature, enter "autosupport modify -support disable" within 24 hours.  
Enabling AutoSupport can significantly speed problem determination and resolution  
should a problem occur on your system.  
For further information on AutoSupport, see:  
<http://support.netapp.com/autosupport/>

Type yes to confirm and continue {yes}: yes  
Enter the node management interface port [e0M]: Enter  
Enter the node management interface IP address: <node01-mgmt-ip>  
Enter the node management interface netmask: <node01-mgmt-mask>  
Enter the node management interface default gateway: <node01-mgmt-gateway>  
A node management interface on port e0M with IP address <node01-mgmt-ip> has been created

Use your web browser to complete cluster setup by accessing <https://<node01-mgmt-ip>>

Otherwise press Enter to complete cluster setup using the command line interface:

2. To complete the cluster setup, open a web browser and navigate to <https://<node01-mgmt-ip>>. The setup wizard will step you through Cluster configuration setup.
3. Enter the storage cluster name and administrative password.



← → ↻ Not secure | 192.168.156.61/sysmgr/v4/

## ONTAP System Manager

### Health

✓ 2 healthy nodes were found.

AFF-A400

### Initialize Storage System

STORAGE SYSTEM NAME

aa11-a400

You will see this name when managing the storage system.


ADMINISTRATIVE PASSWORD

.....

.....

4. Cluster configuration will take few minutes and after the setup the URL will be redirected to the cluster IP.

↻ Not secure | 192.168.156.60/sysmgr/v4/

 **NetApp**  
ONTAP System Manager

[Sign In](#)

[NetApp Support](#) | [NetApp](#)

5. In the System manager, click Overview under Cluster to verify the storage nodes.

The screenshot shows the ONTAP System Manager web interface. The browser address bar indicates the URL is 192.168.156.60/sysmgr/v4/cluster/overview. The interface has a dark blue sidebar with a menu icon and the title 'ONTAP System Manager'. The main content area is titled 'Overview' and displays details for a cluster named 'aa11-a400'. The details include the Name, Version (NetApp Release 9.8P2), and Management Interfaces (192.168.156.60). Below this, there is a 'Nodes' section with a table listing the nodes in the cluster. The table has columns for Nodes, Name, S..., U..., Utilization, and Management IP. Two nodes are listed: aa11-a400-01 and aa11-a400-02, both showing 1% utilization and management IP 192.168.156.61 and 192.168.156.62 respectively.

## ONTAP Automated Configuration

To complete the automated setup/ configuration of the ONTAP storage system, follow these steps:

1. Navigate to the '/vars/ontap\_main.yml' file and fill it out with your environment specific values.



The format of the variable file needs to be maintained as it is, any changes to the structure of the file may lead to failure in execution of the playbook.



Sample values are pre-populated against some variables to provide the user additional clarity on how the variable needs to be filled out. Please replace the sample values with your environment specific information.

2. Navigate to the '/group\_vars/ontap' file and update it with the admin credentials for the ONTAP cluster and Node Management Ips.
3. Update the 'inventory' file with a record of the ONTAP Cluster Management IP.
4. Run the playbook to setup ONTAP storage as per the requirements of OCP 4 using the below command:

```
ansible-playbook Setup_ONTAP.yml -i inventory
```



NetApp Astra Trident will be installed and configured after the OpenShift Container Platform has been setup.

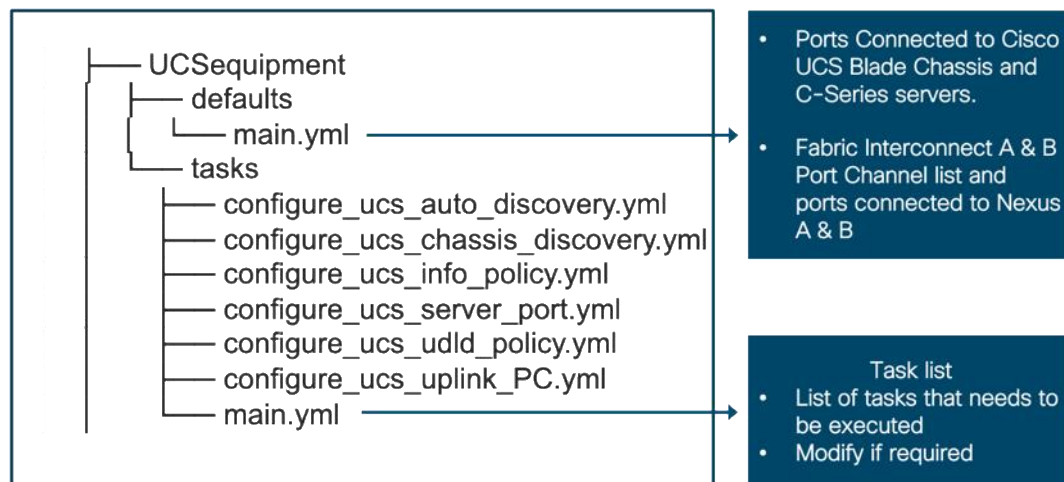
## FlexPod UCS Compute Configuration

To configure FlexPod UCS Compute, follow these steps:

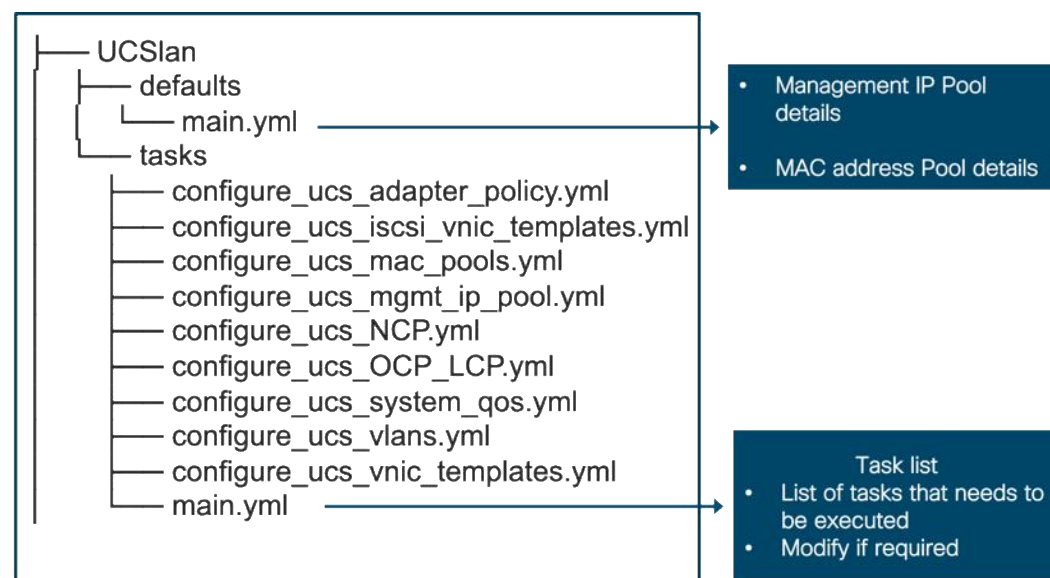


Update the following information as required based on your environment before running the UCS Automation Playbook.

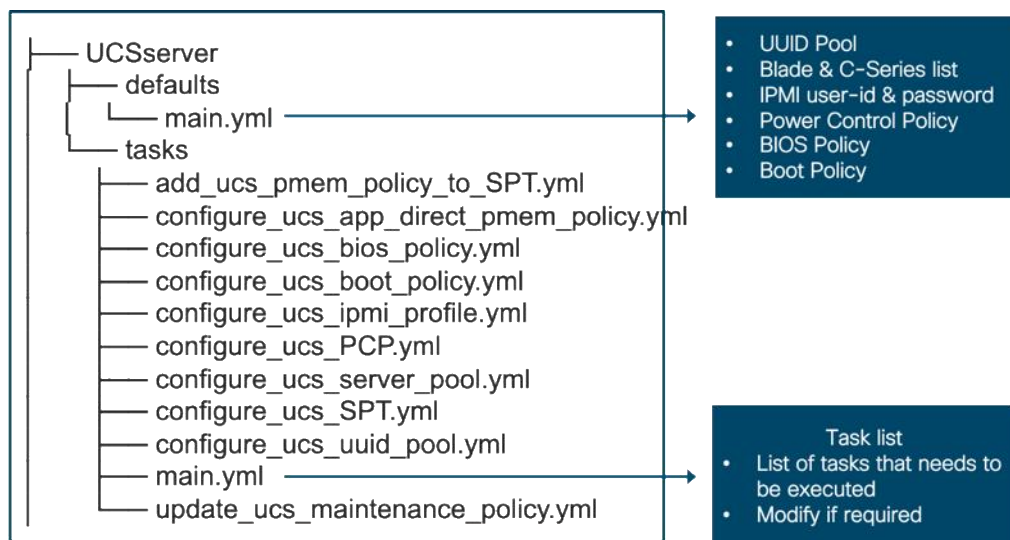
1. Port details and tasks to be included for UCSEquipment configuration role if different from the defaults.



2. Management and MAC address pool details for UCSlan configuration role.



3. UUID pool, UCS servers list and IPMI details for UCSServer configuration role.



Change the values in the mentioned files with caution, only change the information that is required. All the other files can be left to defaults, modify them only if you want to go with a different naming convention or if you do not have the identical hardware discussed in this design.

4. Once the above information has been updated in the respective files, run the UCS Ansible playbook:

```
[root@fp-ansible FlexPod-IaC-OCP4]# ansible-playbook ./Setup_UCS.yml -i inventory
```

5. Login into the Cisco UCS Manager and verify the configuration has been completed as desired.

6. As part of the UCS Playbook, there are two Service Profile templates created, one for Master nodes and the other for the worker nodes, we will use these service profile templates to create and assign Service Profiles to the UCS servers. Either the master node or worker node service profile can be used for the provisioner node.

## Create UCS Service Profiles

To create service profiles from the service profile template, follow these steps:

1. Connect to the Cisco UCS 6454 Fabric Interconnect UCS Manager, click the Servers tab in the navigation pane.
2. Click Service Profile Templates > Sub-Organizations > OCP > Service Template OCP-Master.
3. Right-click OCP-Master and choose Create Service Profiles from Template.
4. Enter OCP-Master- for the service profile prefix.
5. Enter 0 for the "Name Suffix Starting Number."
6. Enter 3 for the "Number of Instances."
7. Click OK to create the service profiles.



- 
8. Click OK in the confirmation message to provision three FlexPod Service Profiles.
  9. Click Service Profile Templates > Sub-Organizations > OCP > Service Template OCP-Worker.
  10. Right-click OCP-Worker and choose Create Service Profiles from Template.
  11. Enter OCP-Worker- for the service profile prefix.
  12. Enter 0 for the “Name Suffix Starting Number.”
  13. Enter 3 for the “Number of Instances.”
  14. Click OK to create the service profiles.
  15. Click OK in the confirmation message to provision three FlexPod Service Profiles.
  16. Wait for the Service Profile association to complete and proceed to the next steps details in the next section.



You will use one of the worker nodes service profile and use it as a provisioner node, the same machine can be added as a worker node into the cluster later if desired without any changes, though a dedicated provisioner node is recommended to simplify the expansion process that can initiated from the provisioner node. The validation team used a dedicated provisioner node and installed a cluster with two worker nodes initially and expanded the cluster with one additional worker node.

---

## OpenShift Container Platform Installation and Configuration

OCP 4.7 is deployed on the Cisco UCS infrastructure as bare metal servers. Three master nodes and two worker nodes are deployed in the validation environment and additional worker nodes can easily be added to increase the scalability of the solution. This document will guide you through the process of using the Ansible playbooks to deploy a Baremetal Installer Provisioned Infrastructure (IPI) of Red Hat OpenShift 4.7. For the manual deployment details refer to Red Hat documentation at “[https://docs.openshift.com/container-platform/4.7/installing/installing\\_bare\\_metal\\_ipi/ipi-install-overview.html](https://docs.openshift.com/container-platform/4.7/installing/installing_bare_metal_ipi/ipi-install-overview.html)”.

### OpenShift Container Platform - Installation Requirements

Installer-provisioned installation provides support for installing OpenShift Container Platform on bare metal nodes. This guide provides a methodology to achieving a successful installation using Ansible.

During installer-provisioned installation on bare metal, the installer on the bare metal node labeled as provisioner creates a bootstrap VM. The role of the bootstrap VM is to assist in the process of deploying an OpenShift Container Platform cluster. The bootstrap VM connects to the baremetal network and to the provisioning network, if present, via the network bridges. When the installation of OpenShift Container Platform control plane nodes is complete and fully operational, the installer destroys the bootstrap VM automatically and moves the virtual IP addresses (VIPs) to the appropriate nodes accordingly. The API VIP moves to the control plane nodes and the Ingress VIP moves to the worker nodes.



The installation of the OCP platform is a multi-step process that must be carefully planned for each customer environment. There are several prerequisites for preparing the infrastructure and the nodes for the

---

installation, and it is also important to recognize that the installation procedure may vary between the different versions of OCP, the validation team tested installation of OCP 4.6 and 4.7 versions.

---

## Prerequisites

The FlexPod for OCP utilizes installer-provisioned infrastructure (IPI) cluster configuration for OCP installation therefore when provisioning and managing the FlexPod infrastructure, you must provide all the supporting cluster infrastructure and resources, including the provisioning node, networking, storage, and individual cluster machines.

The following supporting cluster resources are required for the IPI installation:

- The control plane and compute machines that make up the cluster
- Cluster networking
- Storage for the cluster infrastructure and applications



Before starting the installation of OpenShift platform, make sure Cisco UCS, NetApp Storage, and the Cisco Nexus switches are configured as per the design discussed in the design section of this document.

---

## Network Requirements

The following infrastructure services need to be deployed to support the OCP cluster, during the validation of this solution we have provided VMs to run the required services. Customers can use existing DNS and DHCP services available in the data center:

There are various infrastructure services prerequisites for deploying OCP 4. These prerequisites are as follows:

- DNS and DHCP services – these services were configured on Microsoft Windows Server VMs
- NTP Distribution was done with Nexus switches
- Specific DNS entries for deploying OCP – added to the DNS server
- An HTTP server to host VM image files – OpenShift Virtualization
- A Linux VM for initial automated installation and cluster management – a CentOS / RHEL VM with appropriate packages
- A RHEL server used as provisioner host.

Customers can choose to combine some of these services on a single VM e.g., DNS and DHCP on a single VM and can choose to deploy these services on a platform and version of their choice, for example a windows-based DNS or DHCP server.

## NTP

Each OpenShift Container Platform node in the cluster must have access to an NTP server.

## NICs

NICs configured on the Cisco UCS servers based on the design previously discussed.

---

## DNS

Clients access the OpenShift Container Platform cluster nodes over the baremetal network. Configure a subdomain or subzone where the canonical name extension is the cluster name.

The following domain and OCP cluster names are used in this deployment guide:

- Base Domain: flexpod.cisco.com
- OCP Cluster Name: ocp

[Table 6](#) lists the information for fully qualified domain names used during validation. The API and Nameserver addresses begin with canonical name extensions. The hostnames of the control plane and worker nodes are exemplary, so you can use any host naming convention you prefer.

**Table 6. DNS FQDN Names Used**

Usage	Hostname	IP Address
API	api.ocp.flexpod.cisco.com	10.1.156.226
Ingress LB (apps)	*.apps.ocp.flexpod.cisco.com	10.1.156.227
Worker-2	provisioner.ocp.flexpod.cisco.com	10.1.156.216
Master-0	master-0.ocp.flexpod.cisco.com	10.1.156.211
Master-1	master-1.ocp.flexpod.cisco.com	10.1.156.212
Master-2	master-2.ocp.flexpod.cisco.com	10.1.156.213
Worker-0	worker-0.ocp.flexpod.cisco.com	10.1.156.214
Worker-1	worker-1.ocp.flexpod.cisco.com	10.1.156.215

## DHCP

For the bare metal network, a network administrator must reserve several IP addresses, including:

- One IP address for the API endpoint
- One IP address for the wildcard Ingress endpoint
- One IP address for the provisioner node (DHCP server assigns to the node)
- One IP address for each master node (DHCP server assigns to the node)
- One IP address for each worker node (DHCP server assigns to the node)



Get the MAC addresses of the bare metal Interfaces from the UCS service profiles for each node to be used in the DHCP configuration to assign reserved IP addresses to the nodes. KVM IP address also needs to be gathered for the master and worker nodes from the service profiles.

---

To gather the MAC addresses of the bare metal interfaces on the nodes, follow these steps:

1. Launch the Cisco UCS Manager GUI. In the navigation pane, click the Servers tab. Expand Servers > Service Profiles > Sub-Organization > OCP.
2. Click each node service profile and then click the Network tab on the right. Note “MAC Address” displayed for each bare metal vNIC.
3. Click the General tab and expand the Management IP address on the right to note IP address assigned to the server from the KVM management pool.

[Table 7](#) lists the IP address used for the OCP cluster including bare metal network IPs and UCS KVM Management IPs for IPMI access.

**Table 7. IP Addresses**

Hostname	IP Address	UCS KVM Mgmt. IP Address	MAC Address
provisioner.ocp.flexpod.cisco.com	10.1.156.216		00:25:B5:A1:3B:09
master-0.ocp.flexpod.cisco.com	10.1.156.211	192.168.156.245	00:25:B5:A1:3B:00
master-1.ocp.flexpod.cisco.com	10.1.156.212	192.168.156.246	00:25:B5:A1:3B:01
master-2.ocp.flexpod.cisco.com	10.1.156.213	192.168.156.247	00:25:B5:A1:3B:02
worker-0.ocp.flexpod.cisco.com	10.1.156.214	192.168.156.248	00:25:B5:A1:3B:03
worker-1.ocp.flexpod.cisco.com	10.1.156.215	192.168.156.249	00:25:B5:A1:3B:06

## Setup the Provisioner Node

When the prerequisites are completed, install Red Hat Enterprise Linux (RHEL) 8.3 on your UCS server designated as provisioner host and follow the steps below to prepare the host.

### Download RHEL 8.3 DVD ISO

If the RHEL DVD image has not been downloaded, follow these steps to download the ISO:

1. Click the following link [RHEL 8.3 Binary DVD](#).



A user\_id and password are required on the website (redhat.com) to download this software.

---

2. Download the .iso (rhel-server-8.3-x86\_64-dvd.iso) file.
3. Follow the prompts to launch the KVM console.

## Operate System Installation

To install RHEL 8.3 on the UCS server, follow these steps:

1. Copy the disk image downloaded into your workstation.
2. If Cisco UCS Manager GUI is not open, log in.

- 
3. In the Navigation pane, click the Servers tab
  4. On the Servers tab, expand Servers > Service Profiles.
  5. Click the Provisioner node service profile
  6. In the Work pane, click the General tab.
  7. In the Actions area, click KVM Console. The KVM Console opens in a separate window.
  8. From the KVM console, choose Virtual Media > Activate Virtual Devices to open the Virtual Media Session dialog box. Then choose Virtual Media > CD/DVD.
  9. In the Virtual Disk Management dialog box, map the virtual media selecting Choose File. Navigate to the location of the RHEL 8.3 DVD ISO, select the ISO, and click Open. Back in the Virtual Disk Management dialog box, select Map Drive.
  10. Boot the server. During the boot process, when you see the Cisco splash screen, press F6 to go into the Boot Menu. (You can also use a User Defined Macro to send F6).
  11. Once in the Boot Menu, select UEFI: Cisco vKVM-Mapped vDVD1.24 and press Enter.
  12. On boot, RHEL installation media is detected. From the Installation menu, use arrow keys to select Install Red Hat Enterprise Linux 8.3. This should stop automatic boot countdown.
  13. Continue with the prompts and complete OS installation, installing Server with GUI from Local Media not registered with Red Hat Network.



The server will reboot after the OS installation.



The server gets an IP address assigned dynamically on the second interface from the DHCP server since you reserved an IP address for the second vNIC MAC address.

---

14. Accept the RHEL license and Quit the INITIAL SETUP. Close the KVM console.

## Create the User kni

To create the user kni, follow these steps:

1. Login into the provisioner node via ssh as root.
2. Create a user (kni) to deploy as non-root and provide that user sudo privileges:

```
[root@provisioner ~]# useradd kni
[root@provisioner ~]# passwd kni
[root@provisioner ~]# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
[root@provisioner ~]# chmod 0440 /etc/sudoers.d/kni
```

## Enable Local yum Repository on Provisioner Host

To enable the local yum repository on the provisioner host, follow these steps:

1. Using a secure copy (scp) tool, copy rhel-8.3-x86\_64-dvd.iso to /root on the provisioner host.
2. SSH to the provisioner host as root.
3. Using an editor, add “/root/rhel-8.3-x86\_64-dvd.iso /media/rhel8dvd iso9660 loop 0 0” as the last line in /etc/fstab.
4. Permanently mount the RHEL 8 installation ISO to a /media/rhel8dvd:

```
[root@provisioner ~]# mkdir /media/rhel8dvd
[root@provisioner ~]# mount -a
mount: /media/rhel8dvd: WARNING: device write-protected, mounted read-only.
```

5. Create new repo file as shown below. There are two repositories in RHEL 8, named 'BaseOS' and 'AppStream'.

```
[root@provisioner ~]# vi /etc/yum.repos.d/my.repo
[root@provisioner ~]# cat /etc/yum.repos.d/my.repo
[dvd-BaseOS]
name=DVD for RHEL8 - BaseOS
baseurl=file:///media/rhel8dvd/BaseOS
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

```
[dvd-AppStream]
name=DVD for RHEL8 - AppStream
baseurl=file:///media/rhel8dvd/AppStream
enabled=1
gpgcheck=1
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-redhat-release
```

6. As a final step, it is a good idea to run the command yum clean all once:

```
[root@provisioner ~]# yum clean all
```

7. Check whether you can get the packages list from the DVD repositories.

```
[root@provisioner ~]# yum --noplugins list
```

8. Manually install python3-crypto and python3-pyghmi packages on the provision host:

```
[root@provisioner ~]# curl -o python3-crypto-2.6.1-18.el8ost.x86_64.rpm
https://trunk.rdoproject.org/rhel8-master/deps/latest/Packages/python3-crypto-2.6.1-18.el8ost.x86_64.rpm
```

```
[root@provisioner ~]# dnf -y install ./python3-crypto-2.6.1-18.el8ost.x86_64.rpm
```

```
[root@provisioner ~]# curl -o python3-pyghmi-1.0.22-2.el8ost.noarch.rpm
https://trunk.rdoproject.org/rhel8-master/deps/latest/Packages/python3-pyghmi-1.0.22-2.el8ost.noarch.rpm
```

```
[root@provisioner ~]# dnf -y install ./python3-pyghmi-1.0.22-2.el8ost.noarch.rpm
```

## Deploy IPI on Bare Metal using Ansible Playbook

This section provides detailed instructions to install OpenShift Container Platform 4.7 on Cisco UCS nodes using Ansible.

### Run the Ansible Playbook to Install OCP Cluster

1. Change directory to ocp4/baremetal-deploy/ansible-ipi-install. The Ansible files tree structure is shown below:

```
├─ ansible.cfg
├─ inventory
│   └─ hosts.sample
├─ playbook.yml
└─ roles
    └─ installer
        ├── defaults
        │   └─ main.yml
        ├── files
        ├── handlers
        │   └─ main.yml
        ├── meta
        │   └─ main.yml
        ├── tasks
        │   ├── 10_get_oc.yml
        │   ├── 15_disconnected_registry_create.yml
        │   ├── 15_disconnected_registry_existing.yml
        │   ├── 20_extract_installer.yml
        │   ├── 23_rhcos_image_paths.yml
        │   ├── 24_rhcos_image_cache.yml
        │   ├── 25_create-install-config.yml
        │   ├── 30_create_metal3.yml
        │   ├── 40_create_manifest.yml
        │   ├── 50_extramanifests.yml
        │   ├── 55_customize_filesystem.yml
        │   ├── 59_cleanup_bootstrap.yml
        │   └─ 60_deploy_ocp.yml
```



```
| | | 70_cleanup_sub_man_registration.yml
| | |   └─ main.yml
| | └─ templates
| | | 99-etc-chrony.conf.j2
| | | chrony.conf.j2
| | | install-config-appends.j2
| | | install-config.j2
| | |   └─ metal3-config.j2
| | └─ tests
| | | inventory
| | |   └─ test.yml
| | └─ vars
| | |   └─ main.yml
└─ node-prep
  └─ defaults
    └─ main.yml
  └─ handlers
    └─ main.yml
  └─ library
    └─ nmcli.py
  └─ meta
    └─ main.yml
  └─ tasks
  └─ 100_power_off_cluster_servers.yml
  └─ 10_validation.yml
    └─ 15_validation_disconnected_registry.yml
    └─ 20_sub_man_register.yml
    └─ 30_req_packages.yml
    └─ 40_bridge.yml
    └─ 45_networking_facts.yml
    └─ 50_modify_sudo_user.yml
    └─ 60_enabled_services.yml
    └─ 70_enabled_fw_services.yml
    └─ 80_libvirt_pool.yml
    └─ 90_create_config_install_dirs.yml
    └─ main.yml
  └─ templates
  └─ dir.xml.j2
  └─ tests
```

```
|   |— inventory
|   |— test.yml
|— vars
|   |— main.yml
```

The following information must be modified based on your specific environment:

- inventory - contains the file hosts.sample that:
- contains all the modifiable variables, their default values, and their definition. Some variables are empty ensuring users give an explicit value.
- the setting up of your provision node, master nodes, and worker nodes. Each section will require additional details (i.e., Management credentials).
- roles - contains two roles: node-prep and installer. node-prep handles all the prerequisites that the provisioner node requires prior to running the installer. The installer role handles extracting the installer, setting up the manifests, and running the Red Hat OpenShift installation.

2. Ensure that your environment is using Ansible 2.9 or greater. The following command can be used to verify.

```
[root@fp-ansible ansible-ipi-install]# ansible --version
ansible 2.9.21
  config file = /root/ocp4/baremetal-deploy/ansible-ipi-install/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Mar 18 2021, 08:58:41) [GCC 8.4.1 20200928 (Red Hat
8.4.1-1)]
```

3. Copy your public SSH key to your provisioner node using ssh-copy-id.

4. From the automation host, execute the following commands to copy the ssh key to provisioner host:

```
[root@fp-ansible ansible-ipi-install]# ssh-keygen
[root@fp-ansible ansible-ipi-install]# ssh-copy-id
kni@provisioner.ocp.flexpod.cisco.com
[root@fp-ansible ansible-ipi-install]# ssh kni@provisioner.ocp.flexpod.cisco.com
[kni@provisioner ~]$ exit
```

5. Modify the inventory/hosts file:

- a. The hosts file provides all the definable variables and provides a description of each variable. Some of the variables are explicitly left empty and require user input for the playbook to run.
- b. The hosts file ensures all your nodes that will be used to deploy IPI on baremetal are setup. There are 4 groups: masters, workers, provisioner, and registry\_host (optional). The masters and workers group collects information about the host such as its name, role, user management (such as BMC) user, user management (such as BMC) password, ipmi\_address, ipmi\_port to access the server and the provision mac address (NIC1) that resides on the provisioning network.

- c. Below is a sample of the inventory/hosts file that has been used to install OCP cluster in the validation lab:

```
[root@fp-ansible ansible-ipi-install]# cat inventory/hosts
[all:vars]

#####
# Required configuration variables for IPI on Baremetal Installations      #
#####

# ansible_python_interpreter=/usr/bin/python3

# The provisioning NIC (NIC1) used on all baremetal nodes
prov_nic=eno5

# The public NIC (NIC2) used on all baremetal nodes
pub_nic=eno6

# (Optional) Set the provisioning bridge name. Default value is 'provisioning'.
#provisioning_bridge=provisioning

# (Optional) Set the baremetal bridge name. Default value is 'baremetal'.
#baremetal_bridge=baremetal

# (Optional) Activation-key for proper setup of subscription-manager, empty value skips
registration
#activation_key=""

# (Optional) Activation-key org_id for proper setup of subscription-manager, empty val-
ue skips registration
#org_id=""

# The directory used to store the cluster configuration files (install-config.yaml,
pull-secret.txt, metal3-config.yaml)
dir="{{ ansible_user_dir }}/clusterconfigs"

# The version of the openshift-installer, undefined or empty results in the playbook
failing with error message.
# Values accepted: 'latest-4.3', 'latest-4.4', explicit version i.e. 4.3.0-0.nightly-
2019-12-09-035405
version=latest-4.7
```

```
# (Optional) Fully disconnected installs require manually downloading the release.txt
file and hosting the file
# on a webserver accessible to the provision host. The release.txt file can be down-
loaded at
# https://mirror.openshift.com/pub/openshift-v4/clients/ocp-dev-preview/{{ version
}}/release.txt (for DEV version)
# https://mirror.openshift.com/pub/openshift-v4/clients/ocp/{{ version }}/release.txt
(for GA version)
# Example of hosting the release.txt on your example.com webserver under the 'latest-
4.3' version directory.
# http://example.com:<port>/latest-4.3/release.txt
# Provide the webserver URL as shown below if using fully disconnected
#webserver_url=http://example.com:<port>

# Enter whether the build should use 'dev' (nightly builds) or 'ga' for Generally
Available version of OpenShift
# Empty value results in playbook failing with error message.
build=ga

# Provisioning IP address (default value)
prov_ip=172.22.0.3

# (Optional) Enable playbook to pre-download RHCOS images prior to cluster deployment
and use them as a local
# cache. Default is false.
#cache_enabled=True

# (Optional) The port exposed on the caching webserver. Default is port 8080.
#webserver_caching_port=8080

# (Optional) Enable IPv6 addressing instead of IPv4 addressing on both provisioning and
baremetal network
#ipv6_enabled=True

# (Optional) When ipv6_enabled is set to True, but want IPv4 addressing on provisioning
network
# Default is false.
#ipv4_provisioning=True

# (Optional) When ipv6_enabled is set to True, but want IPv4 addressing on baremetal
network
#ipv4_baremetal=True
```

```
# (Optional) A list of clock servers to be used in chrony by the masters and workers
#clock_servers=["pool.ntp.org","clock.redhat.com"]

# (Optional) Provide HTTP proxy settings
#http_proxy=http://USERNAME:PASSWORD@proxy.example.com:8080

# (Optional) Provide HTTPS proxy settings
#https_proxy=https://USERNAME:PASSWORD@proxy.example.com:8080

# (Optional) comma-separated list of hosts, IP Addresses, or IP ranges in CIDR format
# excluded from proxying
# NOTE: OpenShift does not accept '*' as a wildcard attached to a domain suffix
# i.e. *.example.com
# Use '.' as the wildcard for a domain suffix as shown in the example below.
# i.e. .example.com
#no_proxy_list="172.22.0.0/24,.example.com"

# The default installer timeouts for the bootstrap and install processes may be too
short for some baremetal
# deployments. The variables below can be used to extend those timeouts.

# (Optional) Increase bootstrap process timeout by N iterations.
#increase_bootstrap_timeout=2

# (Optional) Increase install process timeout by N iterations.
#increase_install_timeout=2

# (Optional) Disable RedFish inspection to intelligently choose between IPMI or RedFish
protocol.
# By default this feature is enabled and set to true. Uncomment below to disable and
use IPMI.
redfish_inspection=false

# (Optional) Modify files on the node filesystems, you can augment the "fake" roots for
the
# control plane and worker nodes.
# If defined, playbook will look for files in control plane and worker subdirectories.
# Otherwise, it will look in {{ role_path }}/files/customize_filesystem (default)
# For more information on modifying node filesystems visit: https://bit.ly/36tD30f
```

```
#customize_node_filesystems="/path/to/customized/filesystems"

# (Optional) Modify the path to add external manifests to the deployed nodes.
# If defined, the playbook will copy manifests from the user provided directory.
# Otherwise, files will be copied from the default location
'roles/installer/files/manifests/*'
#customize_extramanifests_path="/path/to/extra/manifests

#####
# Vars regarding install-config.yaml #
#####

# Base domain, i.e. example.com
domain=flexpod.cisco.com
# Name of the cluster, i.e. openshift
cluster=ocp
# The public CIDR address, i.e. 10.1.1.0/21
extcidrnet=10.1.156.0/24
# An IP reserved on the baremetal network.
# Deprecated in OpenShift 4.5 (https://github.com/openshift/installer/pull/3304)
dnsvip=10.1.156.250
# An IP reserved on the baremetal network for the API endpoint.
# (Optional) If not set, a DNS lookup verifies that api.<clustername>.<domain> provides
an IP
apivip=10.1.156.226
# An IP reserved on the baremetal network for the Ingress endpoint.
# (Optional) If not set, a DNS lookup verifies that *.apps.<clustername>.<domain> pro-
vides an IP
ingressvip=10.1.156.227
# The master hosts provisioning nic
# (Optional) If not set, the prov_nic will be used
#masters_prov_nic=""
# Network Type (OpenShiftSDN or OVNKubernetes). Playbook defaults to OVNKubernetes.
# Uncomment below for OpenShiftSDN
#network_type="OpenShiftSDN"
# (Optional) A URL to override the default operating system image for the bootstrap
node.
# The URL must contain a sha256 hash of the image.
# See
https://github.com/openshift/installer/blob/master/docs/user/metal/customization_ipi.md
# Example https://mirror.example.com/images/qemu.qcow2.gz?sha256=a07bd...
```

```
#bootstraposimage=""

# (Optional) A URL to override the default operating system image for the cluster
nodes.

# The URL must contain a sha256 hash of the image.

# See
https://github.com/openshift/installer/blob/master/docs/user/metal/customization_ipi.md
# Example https://mirror.example.com/images/metal.qcow2.gz?sha256=3b5a8...

#clusterosimage=""

# A copy of your pullsecret from https://cloud.redhat.com/openshift/install/metal/user-
provisioned

pullse-
cret='{"auths":{"cloud.openshift.com":{"auth":"b3BlbnNoaWZ0LXJlbGVhc2UtZGV2K3NyZWZlZGEEx
eG9kbzV2N2llZmVjdHFGN2ZrcGVmMWkwdG06STEzWDUwSEo2SDdGVDZQVFNWVFDWEZFS0dET0VKN1pRRUtIQkc
2VFRMNExMV0IyVktHT0ZUZE9CNVhXVlhXVA==","email":"email@cisco.com"},"quay.io":{"auth":"b3
BlbnNoaWZ0LXJlbGVhc2UtZGV2K3NyZWZlZGEExeG9kbzV2N2llZmVjdHFGN2ZrcGVmMWkwdG06STEzWDUwSEo2S
DdGVDZQVFNWVFDWEZFS0dET0VKN1pRRUtIQkc2VFRMNExMV0IyVktHT0ZUZE9CNVhXVlhXVA==","email":"e
mail@cisco.com"},"registry.connect.redhat.com":{"auth":"NTMxOTQxOTV8dWhjLTFYb0RvNVY3SUV
mRWNUUUo3RmtQZUYxaTB0bTpleUpoYkdjaU9pSlNVelV4TWlKOS5leUp6ZFdJaU9pSmtaakZsTlRGaU0yVTJZVG
cwTTRmM1lUQXl0ek5pTlRBM1lqQmhOalEwWWlKOS5BeU9CbzBDQU13dW9paEZ5M1lXTVlaS2JsdVVFOWJ4NXFye
mJEbWk-
tYm5FUU1aa1k3WGJsaW9qX1BZb2RLd1AtRU5LZGdWYkpEakhqMDJjQXYyc3J6TnVfOURhWjFVQWJGbmhIMl9GLU
s0dnc4V3hoU0pPdk9raU95XzZjeERHVHhoQVQ4ZlZlZmVjdHFGN2ZrcGVmMWkwdG06STEzWDUwSEo2SDdGVDZQVFNWVFDWEZFS0dET0VKN1pRRUtIQkc2VFRMNExMV0IyVktHT0ZUZE9CNVhXVlhXVA==","email":"email@cisco.com"},"registry.redhat.io":{"auth":"NTMxOTQxOTV8dWhjLTFYb0RvNVY3SUVmRWNUUUo3RmtQZUYxaTB0bTpleUpoYkdjaU9pSlNVelV4TWlKOS5leUp6ZFdJaU9pSmtaakZsTlRGaU0yVTJZVGcwTTRmM1lUQXl0ek5pTlRBM1lqQmhOalEwWWlKOS5BeU9CbzBDQU13dW9paEZ5M1lXTVlaS2JsdVVFOWJ4NXFyemJEbWk-
tYm5FUU1aa1k3WGJsaW9qX1BZb2RLd1AtRU5LZGdWYkpEakhqMDJjQXYyc3J6TnVfOURhWjFVQWJGbmhIMl9GLU
s0dnc4V3hoU0pPdk9raU95XzZjeERHVHhoQVQ4ZlZlZmVjdHFGN2ZrcGVmMWkwdG06STEzWDUwSEo2SDdGVDZQVFNWVFDWEZFS0dET0VKN1pRRUtIQkc2VFRMNExMV0IyVktHT0ZUZE9CNVhXVlhXVA==","email":"email@cisco.com"},"registry.redhat.io":{"auth":"NTMxOTQxOTV8dWhjLTFYb0RvNVY3SUVmRWNUUUo3RmtQZUYxaTB0bTpleUpoYkdjaU9pSlNVelV4TWlKOS5leUp6ZFdJaU9pSmtaakZsTlRGaU0yVTJZVGcwTTRmM1lUQXl0ek5pTlRBM1lqQmhOalEwWWlKOS5BeU9CbzBDQU13dW9paEZ5M1lXTVlaS2JsdVVFOWJ4NXFyemJEbWk-
tYm5FUU1aa1k3WGJsaW9qX1BZb2RLd1AtRU5LZGdWYkpEakhqMDJjQXYyc3J6TnVfOURhWjFVQWJGbmhIMl9GLU
s0dnc4V3hoU0pPdk9raU95XzZjeERHVHhoQVQ4ZlZlZmVjdHFGN2ZrcGVmMWkwdG06STEzWDUwSEo2SDdGVDZQVFNWVFDWEZFS0dET0VKN1pRRUtIQkc2VFRMNExMV0IyVktHT0ZUZE9CNVhXVlhXVA==","email":"email@cisco.com"}}}'

# (Optional) Disable BMC Certification Validation. When using self-signed certificates
for your BMC, ensure to set to True.

# Default value is False.

#disable_bmc_certificate_verification=True
```

```
# (Optional) Enable RedFish VirtualMedia/iDRAC VirtualMedia
#enable_virtualmedia=True

# (Required when enable_virtualmedia is set to True) Set an available IP address from
the baremetal net for these two variables
#provisioningHostIP=<baremetal_net_IP1>
#bootstrapProvisioningIP=<baremetal_net_IP2>

# (Optional) Change the boot mode of the OpenShift cluster nodes to legacy mode (BIOS).
Default is UEFI.
#bootmode=legacy

# Master nodes
# The hardware_profile is used by the baremetal operator to match the hardware discov-
ered on the host
# See https://github.com/metal3-io/baremetal-
operator/blob/master/docs/api.md#baremetalhost-status
# ipmi_port is optional for each host. 623 is the common default used if omitted
# poweroff is optional. True or omitted (by default) indicates the playbook will power
off the node before deploying OCP
# otherwise set it to false
# (Optional) OpenShift 4.6+, Set Root Device Hints to choose the proper device to in-
stall operating system on OpenShift nodes.
# root device hint options include: ['device-
Name', 'hctl', 'model', 'vendor', 'serialNumber', 'minSizeGigabytes', 'wnn', 'rotational']
# Root Device Hint values are case sensitive. If incorrect case given, entry omitted
from install-config.yaml
# root_device_hint="deviceName"
# root_device_hint_value="/dev/sda"

[masters]
master-0 name=master-0 role=master ipmi_user=ipmiadmin ipmi_password=password ip-
mi_address=192.168.156.245 ipmi_port=623 provision_mac=00:25:B5:A1:3A:00 hard-
ware_profile=default poweroff=true
master-1 name=master-1 role=master ipmi_user=ipmiadmin ipmi_password=password ip-
mi_address=192.168.156.246 ipmi_port=623 provision_mac=00:25:B5:A1:3A:01 hard-
ware_profile=default poweroff=true
master-2 name=master-2 role=master ipmi_user=ipmiadmin ipmi_password=password ip-
mi_address=192.168.156.247 ipmi_port=623 provision_mac=00:25:B5:A1:3A:02 hard-
ware_profile=default poweroff=true

# Worker nodes
[workers]
```



```
worker-0 name=worker-0 role=worker ipmi_user=ipmiadmin ipmi_password=password ip-  
mi_address=192.168.156.248 ipmi_port=623 provision_mac=00:25:B5:A1:3A:03 hard-  
ware_profile=unknown poweroff=true
```

```
worker-1 name=worker-1 role=worker ipmi_user=ipmiadmin ipmi_password=password ip-  
mi_address=192.168.156.249 ipmi_port=623 provision_mac=00:25:B5:A1:3A:06 hard-  
ware_profile=unknown poweroff=true
```

```
# Provision Host
```

```
[provisioner]
```

```
provisioner.ocp.flexpod.cisco.com
```

```
# Registry Host
```

```
# Define a host here to create or use a local copy of the installation registry
```

```
# Used for disconnected installation
```

```
# [registry_host]
```

```
# registry.example.com
```

```
# [registry_host:vars]
```

```
# The following cert_* variables are needed to create the certificates
```

```
# when creating a disconnected registry. They are not needed to use
```

```
# an existing disconnected registry.
```

```
# cert_country=US # two letters country
```

```
# cert_state=MyState
```

```
# cert_locality=MyCity
```

```
# cert_organization=MyCompany
```

```
# cert_organizational_unit=MyDepartment
```

```
# The port exposed on the disconnected registry host can be changed from
```

```
# the default 5000 to something else by changing the following variable.
```

```
# registry_port=5000
```

```
# The directory the mirrored registry files are written to can be modified from the de-  
fault /opt/registry by changing the following variable.
```

```
# registry_dir="/opt/registry"
```

```
# The following two variables must be set to use an existing disconnected registry.
```

```
#
```

```
# Specify a file that contains extra auth tokens to include in the
```

```
# pull-secret if they are not already there.
```

```
# disconnected_registry_auths_file=/path/to/registry-auths.json
```

```
# Specify a file that contains the addition trust bundle and image
# content sources for the local registry. The contents of this file
# will be appended to the install-config.yml file.
# disconnected_registry_mirrors_file=/path/to/install-config-append.json
masters_prov_nic="" variable needs to be adjusted if the interface name on masters is
different than the interface name from the provisioning node. This can be the case if
there are different Cisco adapters used on the UCS servers.
Default IMPI access userid and password have been used that are hardcoded in the Ansible
scripts, change them as needed.
```

- d. Prior to the installation of Red Hat OpenShift, include the machine config file to be included during the installation. This file has been included before starting the installation to assign IP addresses to the NFS and iSCSI interfaces on Worker nodes. Any other extra configuration can be performed during installation by placing the machineconfig files in the “baremetal-deploy/ansible-ipi-install/roles/installer/files/manifests” directory.
- e. To create the machine-config file, MAC addresses and iSCSI IP addresses need to be gathered.
- f. To gather the MAC addresses of the iSCSI interfaces on the worker nodes, launch the Cisco UCS Manager GUI. In the navigation pane, click the Servers tab. Expand Servers > Service Profiles > Sub-Organization > OCP.
- g. Click each worker node service profile and then click the Network tab on the right. Note “MAC Address” displayed for each iSCSI vNIC.
- h. Create a new ifcfg text file which defines a HWADDR which corresponds to the MAC address of the adapter to be configured. Create one file for each adapter on all worker nodes.

```
HWADDR=00:25:B5:A1:3B:04
```

```
TYPE=Ethernet
```

```
BOOTPROTO=none
```

```
IPADDR=192.168.154.214
```

```
PREFIX=24
```

```
ONBOOT=yes
```

- i. Run the following command to base64 encode the ifcfg file(s):

```
# cat ifcfg-file | base64 -w 0
```

```
SFd-
```

```
BRERSPTAwOjI1OkI1OkExOjNCOjA0ClRZUEU9RXRoZXJuZXQKQk9PVFBST1RPPW5vbmUKSVBBERERSPTE5Mi4xNjguMTU0LjIxNApQUkVGSVg9MjQKT05CT09UPXl1cwo=
```

- j. Create a new machineconfig yaml file which contains the base64 encoded ifcfg files. Append the base64 content after data:text/plain;charset=utf-8;base64,

---

Below is an example to configure three network adapters (NFS, iSCSI-A and iSCSI-B) on each worker node in a cluster consisting of two worker nodes. A file was created with the name the-machine-config and the contents were updated as follows:

```
[root@fp-ansible manifests]# cat machine.yaml
{
  "apiVersion": "machineconfiguration.openshift.io/v1",
  "kind": "MachineConfig",
  "metadata": {
    "labels": {
      "machineconfiguration.openshift.io/role": "worker"
    },
    "name": "99-storage-network"
  },
  "spec": {
    "config": {
      "ignition": {
        "config": {},
        "timeouts": {},
        "version": "2.1.0"
      },
      "networkd": {},
    }
  }
}
```

```
"passwd": {},

"storage": {

  "files": [

    {

      "filesystem": "root",

      "path": "/etc/sysconfig/network-scripts/ifcfg-worker-0-eno8",

      "contents": {

        "source": "data:text/plain;charset=utf-8;base64,SFdBREERSPTAwOjI1OkI1OkExOjNCOjA0ClRZUEU9RXRoZXJuZXQKQk9PVFBST1RPPW5vbmUKSVBBRE  
RSPTe5Mi4xNjguMTU0LjIxNApQUkVGSVg9MjQKT05CT09UPXl1cwo=",

        "verification": {}

      },

      "mode": 420

    },

    {

      "filesystem": "root",

      "path": "/etc/sysconfig/network-scripts/ifcfg-worker-0-eno9",

      "contents": {

        "source": "data:text/plain;charset=utf-8;base64,SFdBREERSPTAwOjI1OkI1OkExOjNBOjA1ClRZUEU9RXRoZXJuZXQKQk9PVFBST1RPPW5vbmUKSVBBRE  
RSPTe5Mi4xNjguMTQumjE0ClBSRUZJWD0yNApPTkJPt1Q9eWVzCg==",

        "verification": {}

      }

    }

  ]

}
```

```
    },

    "mode": 420

  },

  {

    "filesystem": "root",

    "path": "/etc/sysconfig/network-scripts/ifcfg-worker-0-eno10",

    "contents": {

      "source": "data:text/plain;charset=utf-8;base64,SFdBREERSPTAwOjI1OkI1OkExOjNCb2JlR2ZUEU9RXRoZXJuZXQKQk9PVFBST1RPPW5vbmUKSVBBRE  
RSPTE5Mi4xNjguMjE0ClBSRUZJWD0yNApPTkJPPT1Q9eWVzCgo=",

      "verification": {}

    },

    "mode": 420

  },

  {

    "filesystem": "root",

    "path": "/etc/sysconfig/network-scripts/ifcfg-worker-1-eno8",

    "contents": {

      "source": "data:text/plain;charset=utf-8;base64,SFdBREERSPTAwOjI1OkI1OkExOjNCb2JlR2ZUEU9RXRoZXJuZXQKQk9PVFBST1RPPW5vbmUKSVBBRE  
RSPTE5Mi4xNjguMTU0LjIxNQpQUkVGSVg9MjQKT05CT09UPXl1cwo=",

      "verification": {}

    }

  }

}
```

```
    },

    "mode": 420

  },

  {

    "filesystem": "root",

    "path": "/etc/sysconfig/network-scripts/ifcfg-worker-1-eno9",

    "contents": {

      "source": "data:text/plain;charset=utf-8;base64,SFdBREERSPTAwOjI1OkI1OkExOjNBOjA4ClRZUEU9RXRoZXJuZXQkQk9PVFBST1RPPW5vbmUKSVBBRE  
RSPTE5Mi4xNjguMTQuMjE1ClBSRUZJWD0yNApPTkJPt1Q9eWVzCg==",

      "verification": {}

    },

    "mode": 420

  },

  {

    "filesystem": "root",

    "path": "/etc/sysconfig/network-scripts/ifcfg-worker-1-eno10",

    "contents": {

      "source": "data:text/plain;charset=utf-8;base64,SFdBREERSPTAwOjI1OkI1OkExOjNCOjA4ClRZUEU9RXRoZXJuZXQkQk9PVFBST1RPPW5vbmUKSVBBRE  
RSPTE5Mi4xNjguMjE1ClBSRUZJWD0yNApPTkJPt1Q9eWVzCgo=",

      "verification": {}

    }

  }

}
```

```

    },
    "mode": 420
  },
]
},
"systemd": {}
},
"osImageURL": ""
}
}

```

- k. The Ansible playbook connects to your provision host and runs through the node-prep role and the installer role. No modification is necessary. All modifications of variables may be done within the inventory/hosts file.

- l. Following is a sample playbook.yml file:

```

---
- name: IPI on Baremetal Installation Playbook
  hosts: provisioner
  roles:
    - node-prep
    - installer

```

- m. Run the Ansible playbook to install OCP Cluster, the output below has been trimmed down with only the end portion shown for a successful installation:

```

[root@fp-ansible ansible-ipi-install]# export ANSIBLE_CONFIG=./ansible.cfg
[root@fp-ansible ansible-ipi-install]# ansible-playbook -i inventory/hosts playbook.yml

```

```

TASK [installer : Deploy OpenShift Cluster]
*****
Friday 21 May 2021  14:48:08 -0400 (0:00:00.132)          0:01:14.635 *****
changed: [provisioner.ocp.flexpod.cisco.com]

```

```

TASK [installer : Run OpenShift Cluster install as async task]
*****
Friday 21 May 2021  15:56:29 -0400 (1:08:20.237)          1:09:34.872 *****
skipping: [provisioner.ocp.flexpod.cisco.com]

```

```

TASK [installer : Wait for kubeconfig file]
*****
Friday 21 May 2021  15:56:29 -0400 (0:00:00.254)          1:09:35.126 *****

```

skipping: [provisioner.ocp.flexpod.cisco.com]

TASK [installer : Wait for Bootstrap Complete]

\*\*\*\*\*

Friday 21 May 2021 15:56:29 -0400 (0:00:00.129) 1:09:35.256 \*\*\*\*\*

skipping: [provisioner.ocp.flexpod.cisco.com]

TASK [installer : Wait for Install Complete]

\*\*\*\*\*

Friday 21 May 2021 15:56:29 -0400 (0:00:00.125) 1:09:35.381 \*\*\*\*\*

skipping: [provisioner.ocp.flexpod.cisco.com]

PLAY RECAP

\*\*\*\*\*  
\*\*\*\*\*

provisioner.ocp.flexpod.cisco.com : ok=88 changed=32 unreachable=0 failed=0  
skipped=97 rescued=2 ignored=1

Friday 21 May 2021 15:56:29 -0400 (0:00:00.020) 1:09:35.402 \*\*\*\*\*

=====

installer : Deploy OpenShift Cluster -----  
----- 4100.24s

installer : Power off hosts -----  
----- 5.11s

Extracting the installer -----  
----- 4.00s

node-prep : Install required packages -----  
----- 3.93s

installer : Untar the openshift-client-linux-4.7.11.tar.gz -----  
----- 2.32s

Gathering Facts -----  
----- 2.31s

node-prep : Gather the rpm package facts -----  
----- 1.77s

node-prep : Enable Services (firewalld) -----  
----- 1.69s

node-prep : Create Bridge labeled baremetal for ipv4 -----  
----- 1.63s

installer : Copy openshift-baremetal-install binary to /usr/local/bin -----  
----- 1.62s

installer : Copy oc binary to /usr/local/bin -----  
----- 1.51s

node-prep : Add "{{ ansible\_user }}" user to libvirt group and get ssh key setup -----  
----- 1.42s



```

installer : Get the ocp client tar gunzip file -----
----- 1.24s
installer : Create OpenShift Manifest -----
----- 1.12s
node-prep : Reload baremetal bridge and slave interfaces -----
----- 1.08s
node-prep : Enable and restart Services -----
----- 1.04s
node-prep : Reload provisioning bridge and slave interfaces -----
----- 1.04s
installer : Generate install-config.yaml -----
----- 0.99s
node-prep : Confirm whether or not internet connectivity on provisioner host -----
----- 0.91s
node-prep : Define Storage Pool for default -----
----- 0.88s

```

n. Once the playbook has successfully completed, verify that your environment is up and running.

o. Log into the provisioner node:

```
[root@fp-ansible ansible-ipi-install]# ssh kni@provisioner.ocp.flexpod.cisco.com
```

p. Export the kubeconfig file located in the ~/clusterconfigs/auth directory:

```
[kni@provisioner ~]$ export KUBECONFIG=/home/kni/clusterconfigs/auth/kubeconfig
```

q. Verify the nodes in the OpenShift Cluster:

```
[kni@provisioner ~]$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master-0.ocp.flexpod.cisco.com	Ready	master	132m	v1.20.0+2817867
master-1.ocp.flexpod.cisco.com	Ready	master	132m	v1.20.0+2817867
master-2.ocp.flexpod.cisco.com	Ready	master	132m	v1.20.0+2817867
worker-0.ocp.flexpod.cisco.com	Ready	worker	104m	v1.20.0+2817867
worker-1.ocp.flexpod.cisco.com	Ready	worker	101m	v1.20.0+2817867

r. Check the overall status by issuing the tail command to the .openshift\_install.log log file in the install directory folder:

```
[kni@provisioner ~]$ tail -f /home/kni/clusterconfigs/.openshift_install.log
```

```
time="2021-05-21T15:56:28-04:00" level=info msg="Install complete!"
```

```
time="2021-05-21T15:56:28-04:00" level=info msg="To access the cluster as the sys-
tem:admin user when using 'oc', run 'export KUBECON-
FIG=/home/kni/clusterconfigs/auth/kubeconfig'"
```

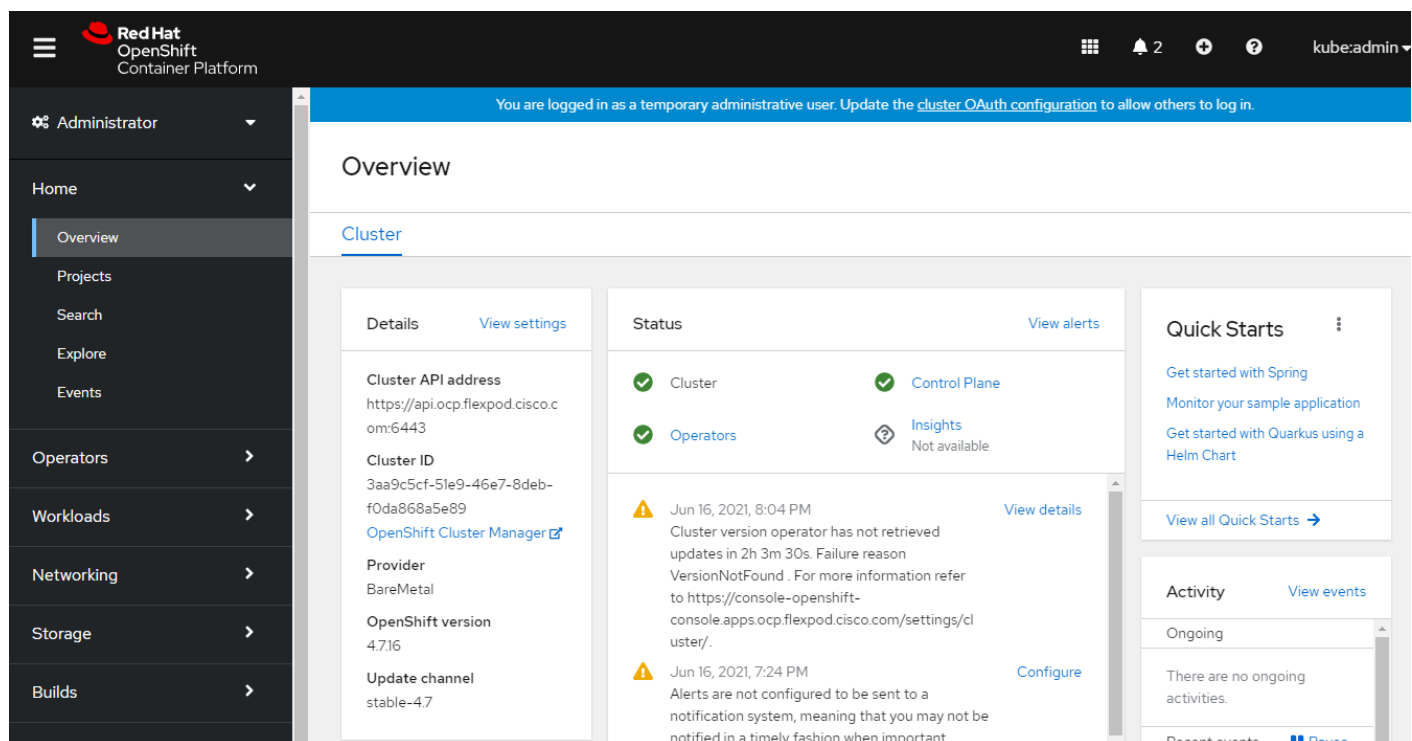
```
time="2021-05-21T15:56:28-04:00" level=info msg="Access the OpenShift web-console here:
https://console-openshift-console.apps.ocp.flexpod.cisco.com"
```

```
time="2021-05-21T15:56:28-04:00" level=info msg="Login to the console with user:
\"kubeadmin\", and password: \"AU7fG-cbGzA-UrsXB-De4bT\""
```

```
time="2021-05-21T15:56:28-04:00" level=debug msg="Time elapsed per stage:"
```

```
time="2021-05-21T15:56:28-04:00" level=debug msg="    Infrastructure: 18m2s"
time="2021-05-21T15:56:28-04:00" level=debug msg="Bootstrap Complete: 14m22s"
time="2021-05-21T15:56:28-04:00" level=debug msg=" Bootstrap Destroy: 10s"
time="2021-05-21T15:56:28-04:00" level=debug msg=" Cluster Operators: 35m19s"
time="2021-05-21T15:56:28-04:00" level=info msg="Time elapsed: 1h8m20s"
```

- s. On successful completion of the installation, a message with access information will be displayed as shown above, the default kube-admin password is also stored in the file `"/home/kni/clusterconfigs/auth/kubeadmin-password."`
- t. Point your browser to your web-console URL to login to OCP. Use kubeadmin for the username and the generated password to log into the system.
- u. The OCP Dashboard is loaded upon successful login:



- v. To ssh into one of the nodes, from the provisioner node logged in as kni:

```
[kni@provisioner ~]$ ssh core@worker-0
```

- w. To be able to log into the nodes from the Ansible VM, create the kni user on the Ansible VM, then copy the kni keys and the clusterconfigs directory from the provisioner to the Ansible VM:

```
[root@fp-ansible ansible-ipi-install]# useradd kni
[root@fp-ansible ansible-ipi-install]# passwd kni
[root@fp-ansible ~]# echo "kni ALL=(root) NOPASSWD:ALL" | tee -a /etc/sudoers.d/kni
[root@fp-ansible ~]# chmod 0440 /etc/sudoers.d/kni
[root@fp-ansible ansible-ipi-install]# su -l kni
[kni@fp-ansible ~]$ cd .ssh
[kni@fp-ansible .ssh]$ scp kni@provisioner.ocp.flexpod.cisco.com:/home/kni/.ssh/id_* ./
```

```
[kni@fp-ansible .ssh]$ cd
[kni@fp-ansible ~]$ scp -r
kni@provisioner.ocp.flexpod.cisco.com:/home/kni/clusterconfigs ./
```

- x. To be able to run the oc command from the Ansible VM, download the Linux command-line tools from <https://cloud.redhat.com/openshift/install/metal/installer-provisioned>, and as root, copy the openshift-client-linux.tar.gz file to /tmp on the Ansible VM:

```
[kni@fp-ansible ~]$ cd /tmp
[kni@fp-ansible tmp]$ sudo tar -xvzf openshift-client-linux.tar.gz
[kni@fp-ansible tmp]$ sudo cp oc /usr/local/sbin/
[kni@fp-ansible tmp]$ sudo cp kubectl /usr/local/sbin/
[kni@fp-ansible ~]$ export KUBECONFIG=/home/kni/clusterconfigs/auth/kubeconfig
[kni@fp-ansible ~]$ oc get nodes
```

- y. Begin the process of converting the provisioner node to the worker-2 node by first Installing ipmitool on the Ansible VM:

```
[kni@fp-ansible ~]$ sudo dnf install ipmitool
```

- z. Create ifcfg files for the worker-2 storage interfaces and convert them to base64 using the “cat ifcfg-file | base64 -w 0” command. In the OpenShift Container Platform console, select Compute > Machine-Configs > 99-storage-network. In the dialog for 99-storage-network, select YAML. Select and copy the three entries for the worker-1 interfaces and paste just below worker-1. Edit the entries for worker-2 and replace the base64 info with the generated base64 entries as shown below. Click Save, then Reload.

Red Hat OpenShift Container Platform

MachineConfigs > MachineConfig details

MC 99-storage-network

Details YAML Events

View shortcuts | View sidebar

```

94     data:text/plain;charset=utf-8;base64,SFdBREVSPTAwOjI1OkI1OkExOjNC0jBCC1RZUEU9RXRoZXJ1ZXN
95     verification: {}
96     filesystem: root
97     mode: 420
98     path: /etc/sysconfig/network-scripts/ifcfg-worker-1-iscsi-b
99     contents:
100     source: >-
101     data:text/plain;charset=utf-8;base64,SFdBREVSPTAwOjI1OkI1OkExOjNC0jBCC1RZUEU9RXRoZXJ1ZXN
102     verification: {}
103     filesystem: root
104     mode: 420
105     path: /etc/sysconfig/network-scripts/ifcfg-worker-2-nfs
106     contents:
107     source: >-
108     data:text/plain;charset=utf-8;base64,SFdBREVSPTAwOjI1OkI1OkExOjNC0jBCC1RZUEU9RXRoZXJ1ZXN
109     verification: {}
110     filesystem: root
111     mode: 420
112     path: /etc/sysconfig/network-scripts/ifcfg-worker-2-iscsi-a
113     contents:
114     source: >-
115     data:text/plain;charset=utf-8;base64,SFdBREVSPTAwOjI1OkI1OkExOjNC0jBCC1RZUEU9RXRoZXJ1ZXN
116     verification: {}
117     filesystem: root
118     mode: 420
119     path: /etc/sysconfig/network-scripts/ifcfg-worker-2-iscsi-b
120     systemd: {}
121     osImageURL: ''
122

```

99-storage-network has been updated to version 941701

Save Reload Cancel Download

aa. In the DNS and DHCP servers, remove the provisioner FQDN DNS and DHCP reservation entry and create corresponding worker-2 entries. Also, remove the provisioner entry from the reverse lookup DNS zone if that zone exists and confirm the worker-2 entry.

bb. From the Ansible VM, logged in as the kni user, use the ipmitool to power down the provisioner node. The server's CIMC IP address can be obtained by selecting the Service Profile in Cisco UCS Manager and checking Management IP Address under the General tab. Verify that the server shuts down in UCS Manager.

```
[kni@fp-ansible ~]# ipmitool -I lanplus -H 192.168.156.250 -U ipmiadmin -P password chassis power off
```

cc. Using two ssh logins to the Ansible VM, create the bmh.yaml file to be used to add worker-2 to the OCP cluster.

```
[kni@fp-ansible ~]$ echo -ne "ipmiadmin" | base64
aXBtaWFkZWlu
[kni@fp-ansible ~]$ echo -ne "password" | base64
```

```

cGFzc3dvcmQ=
[kni@fp-ansible ~]$ vi bmh.yaml
[kni@fp-ansible ~]$ cat bmh.yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: worker-2-bmc-secret
type: Opaque
data:
  username: aXBtaWFkbWlu
  password: cGFzc3dvcmQ=
---
apiVersion: metal3.io/v1alpha1
kind: BareMetalHost
metadata:
  name: worker-2
spec:
  online: true
  bootMACAddress: 00:25:B5:A1:3A:09
  bmc:
    address: ipmi://192.168.156.250
    credentialsName: worker-2-bmc-secret

```

**dd. Create the bare metal node:**

```

[kni@fp-ansible ~]$ oc -n openshift-machine-api create -f bmh.yaml
secret/worker-2-bmc-secret created
baremetalhost.metal3.io/worker-2 created

```

**ee. OCP will power up and inspect the bare metal node:**

```

[kni@fp-ansible ~]$ oc -n openshift-machine-api get bmh worker-2

```

NAME	STATUS	PROVISIONING STATUS	CONSUMER	BMC	HARDWARE
PROFILE	ONLINE	ERROR			
worker-2	OK	inspecting		ipmi://192.168.156.250	
true					

**ff. Ensure the PROVISIONING STATUS is ready before provisioning the bare metal node:**

```

[kni@fp-ansible ~]$ oc -n openshift-machine-api get bmh worker-2

```

NAME	STATUS	PROVISIONING STATUS	CONSUMER	BMC	HARDWARE
PROFILE	ONLINE	ERROR			
worker-2	OK	ready		ipmi://192.168.156.250	unknown
true					

**gg. Get a count of the number of worker nodes:**

```
[kni@fp-ansible ~]$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master-0.ocp.flexpod.cisco.com	Ready	master	3h2m	v1.20.0+2817867
master-1.ocp.flexpod.cisco.com	Ready	master	3h2m	v1.20.0+2817867
master-2.ocp.flexpod.cisco.com	Ready	master	3h2m	v1.20.0+2817867
worker-0.ocp.flexpod.cisco.com	Ready	worker	154m	v1.20.0+2817867
worker-1.ocp.flexpod.cisco.com	Ready	worker	151m	v1.20.0+2817867

#### hh. Get the machine set:

```
[kni@fp-ansible ~]$ oc get machinesets -n openshift-machine-api
```

NAME	DESIRED	CURRENT	READY	AVAILABLE	AGE
ocp-snsd-worker-0	2	2	2	2	3h21m

#### ii. Increase the number of worker nodes by one:

```
[kni@fp-ansible ~]$ oc scale --replicas=3 machineset ocp-snsd-worker-0 -n openshift-machine-api
```

```
machineset.machine.openshift.io/ocp-snsd-worker-0 scaled
```

#### jj. Check the status of the bare metal node until the PROVISIONING STATUS is provisioned and the server has rebooted. This can take 30 minutes or more:

```
[kni@fp-ansible ~]$ oc -n openshift-machine-api get bmh worker-2
```

NAME	STATUS	PROVISIONING STATUS	CONSUMER	BMC
HARDWARE PROFILE	ONLINE	ERROR		
worker-2	OK	provisioned	ocp-snsd-worker-0-ptwzd	ip-mi://192.168.156.250
		unknown	true	

#### kk. Once provisioned, ensure the bare metal node is Ready:

```
[kni@fp-ansible ~]$ oc get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
master-0.ocp.flexpod.cisco.com	Ready	master	3h20m	v1.20.0+2817867
master-1.ocp.flexpod.cisco.com	Ready	master	3h20m	v1.20.0+2817867
master-2.ocp.flexpod.cisco.com	Ready	master	3h20m	v1.20.0+2817867
worker-0.ocp.flexpod.cisco.com	Ready	worker	172m	v1.20.0+2817867
worker-1.ocp.flexpod.cisco.com	Ready	worker	169m	v1.20.0+2817867
worker-2.ocp.flexpod.cisco.com	Ready	worker	80s	v1.20.0+2817867

## Install OpenShift Virtualization

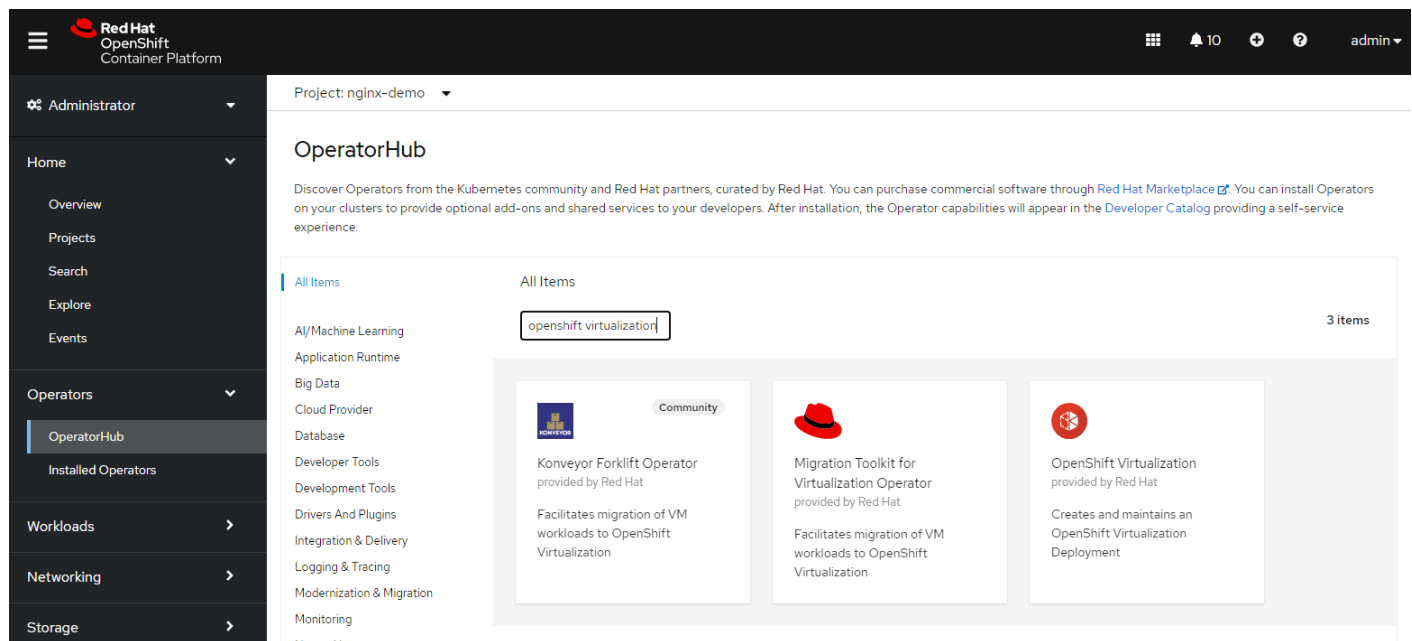
Install OpenShift Virtualization to add virtualization functionality to your OpenShift Container Platform cluster. You can use the OpenShift Container Platform 4.7 web console to subscribe to and deploy the OpenShift Virtualization Operators.

### Subscribe to the OpenShift Virtualization Catalog

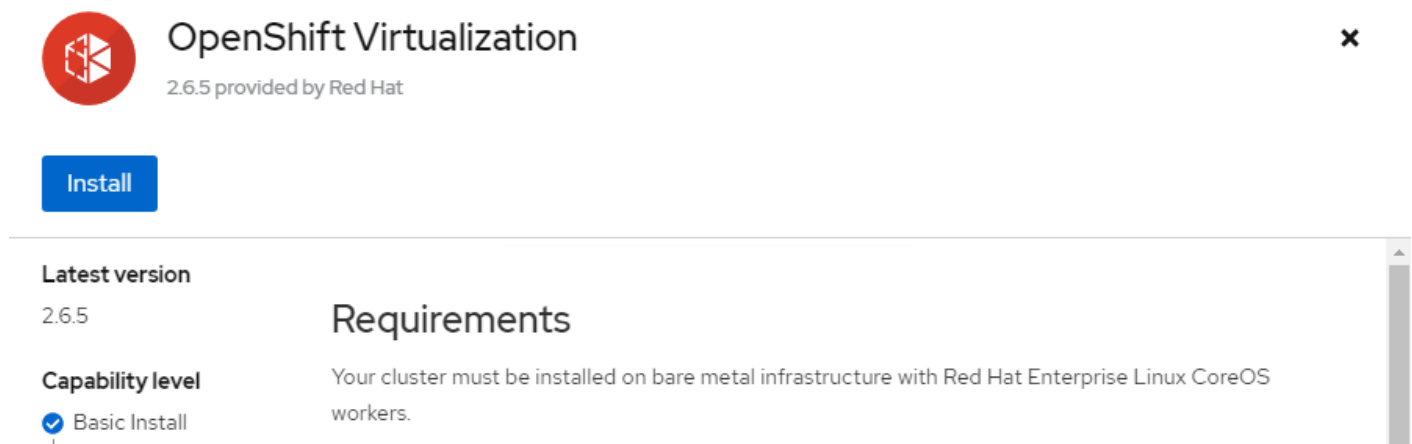
Before you install OpenShift Virtualization, subscribe to the OpenShift Virtualization catalog from the OpenShift Container Platform web console. Subscribing gives the openshift-cnv namespace access to the OpenShift Virtualization Operators.

To subscribe to the OpenShift Virtualization catalog, follow these steps:

1. Open a browser window and log into the OpenShift Container Platform web console.
2. Navigate to the Operators > OperatorHub page.
3. Search for OpenShift Virtualization and then select it.



4. Read the information about the Operator and click Install.



5. On the Install Operator page: for Installed Namespace, ensure that the Operator recommended namespace option is selected. This installs the Operator in the mandatory openshift-cnv namespace, which is automatically created if it does not exist.

6. Select stable from the list of available Update Channel options. This ensures that you install the version of OpenShift Virtualization that is compatible with your OpenShift Container Platform version.
7. For Approval Strategy, ensure that Automatic, which is the default value, is selected. OpenShift Virtualization automatically updates when a new z-stream release is available.
8. Click Install.

[OperatorHub](#) > Operator Installation

## Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

### Update channel \*

- ☐ 2.1
- ☐ 2.2
- ☐ 2.3
- ☐ 2.4
- ☒ stable

### Installation mode \*

- ☐ All namespaces on the cluster (default)  
This mode is not supported by this Operator
- ☒ A specific namespace on the cluster  
Operator will be available in a single Namespace only.

### Installed Namespace \*

- ☒ Operator recommended Namespace: **PR** openshift-cnv



#### Namespace creation

Namespace **openshift-cnv** does not exist and will be created.

- ☐ Select a Namespace

### Approval strategy \*

- ☒ Automatic
- ☐ Manual

Install

Cancel



OpenShift Virtualization  
provided by Red Hat

#### Provided APIs



OpenShift  
Virtualization  
Deployment

**Required**

Represents the deployment of  
OpenShift Virtualization



HostPathProvisioner  
deployment

Represents a HostPathProvisioner  
deployment

9. OpenShift Virtualization will take a few minutes to install. Click View installed Operators in Namespace openshift-cnv.





OpenShift Virtualization  
2.6.5 provided by Red Hat



## Installing Operator

The Operator is being installed. This may take a few minutes.



[View installed Operators in Namespace openshift-cnv](#)

10. On the Installed Operators screen, the Status displays Succeeded when OpenShift Virtualization finishes installation.

Project: openshift-cnv ▼

## Installed Operators

Installed Operators are represented by ClusterServiceVersions within this Namespace. For more information, see the [Understanding Operators documentation](#). Or create an Operator and ClusterServiceVersion using the [Operator SDK](#).

Name ▼	Search by name...	
Name ↑	Managed Namespaces ⓘ	Status
 OpenShift Virtualization 2.6.5 provided by Red Hat	 openshift-cnv	✓ Succeeded Up to date
		Provided APIs <a href="#">OpenShift Virtualization Deployment</a> <a href="#">HostPathProvisioner deployment</a>

## Install NetApp Astra Trident on OpenShift Container Platform

In a FlexPod environment, Trident is utilized to allow end users to dynamically provision and manage persistent volumes for containers backed by FlexVols and LUNs hosted on NetApp A400. Beginning with version v21.04.0 release, the setup of Astra Trident is performed by the Trident operator using Helm chart. The Astra Trident Operator controls the installation of Astra Trident, taking care to self-heal the install and manage changes as well as upgrades to the Astra Trident installation. The following procedure details the steps required to install and configure Astra Trident to manage persistent storage for containers in the OpenShift on FlexPod solution.

## Prerequisites to Deploy Trident Operator by using Helm

To deploy Trident Operator using Helm, follow these steps:

1. Full support/ access to a Kubernetes cluster min. version 1.16
2. Install Helm.
3. Download the Trident installer bundle using the below commands and extract it to a directory:

```
[kni@fp-ansible-2 ~]$ wget
https://github.com/netApp/trident/releases/download/v21.04.0/trident-installer-21.04.0.tar.gz
```

```
[kni@fp-ansible ~]$ tar -xvf trident-installer-21.04.0.tar.gz
```

```
[kni@fp-ansible ~]$ cd trident-installer/helm
```

```
[kni@fp-ansible helm]$ oc create namespace trident
```

```
[kni@fp-ansible helm]$ helm install ocp-trident ocp-trident-operator-21.04.0.tgz -n trident
```

4. Verify that the Trident operator and the pods related to Trident are deployed and running:

```
kni@fp-ansible-2 helm]$ oc get pod -n trident
```

NAME	READY	STATUS	RESTARTS	AGE
trident-csi-5bqxq	2/2	Running	0	
trident-csi-696b685cf8-bh7dj	6/6	Running	0	21s
trident-csi-chwg5	2/2	Running	0	21s
trident-csi-jsrj8	2/2	Running	0	21s
trident-csi-mmxcx	2/2	Running	0	21s
trident-csi-rxpsx	2/2	Running	0	21s
trident-csi-w29sm	2/2	Running	0	21s
trident-operator-7c748d957-rmgj9	1/1	Running	0	21s



If any of the pods are not in Running state, it could be because of the pull rate limit enforced by Docker.

5. To confirm, run the oc describe command:

```
kni@fp-ansible-2 helm]$ oc describe pod <<POD Name>> -n trident
```

6. If you see the following response, proceed to step 4 to fix the issue:

```
You have reached your pull rate limit. You may increase the limit by authenticating and upgrading: https://www.docker.com/increase-rate-limit
```

```
Docker image pulls for anonymous users is restricted to 100 pulls in 24 hours.
```



A docker user login is required to pull the Trident images.

---

To authenticate using a Docker Account to pull the images, follow these steps:

1. SSH to the worker node where the pod is scheduled and login to Docker hub using your user credentials.
2. After login, it will create an authentication config file at `${XDG_RUNTIME_DIR}/containers/auth.json`:

```
[core@worker-0 ~]$podman login docker.io
[core@worker-0 ~]$cat ${XDG_RUNTIME_DIR}/containers/auth.json
```

3. Copy the contents of the authentication config file from the worker node and paste it in a file (json) on the management host.
4. On the management host, set the docker authentication to be used by OC for pulling the image from Docker Hub:

```
kni@fp-ansible-2 helm]$ oc set data secret/pull-secret -n openshift-config --from-file=.dockerconfigjson=docker-auth.json
```

5. Move the trident-installer directory to the desired destination:

```
[kni@fp-ansible-2 ~]$ echo $PATH

[kni@fp-ansible-2 ~]$ mv trident-installer /usr/local/bin

[kni@fp-ansible-2 ~]$ tridentctl -n trident version
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 21.04.0        | 21.04.0        |
+-----+-----+
```

## Prepare the Worker Node

All the worker nodes in the Kubernetes cluster need to be able to mount the volumes that users have provisioned for their pods. For the ontap-nas driver (NAS backend), workers need the NFS tools and for the ontap-san driver (SAN backend) iSCSI tools are required. Also, multipath needs to be enable for iSCSI.

Recent versions of RedHat CoreOS have both NFS and iSCSI utilities installed by default. Make sure that the utilities are installed on all worker nodes.

```
rpm -qa | grep nfs-utils
systemctl status nfs-client.target
rpm -q iscsi-initiator-utils
```

To enable iSCSI and multipath service on the worker nodes a machine config file needs to be created which will be handled by Machine Config Operator (MCO).






1. Log into the OCP web console and navigate to Compute > Machine Configs. Click Create Machine Config. Copy and paste the YAML file and click Create

```






apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  name: 99-worker-ontap-iscsi
  labels:
    machineconfiguration.openshift.io/role: worker
spec:
  config:
    ignition:
      version: 3.2.0
    storage:
      files:
      - contents:
          source: data:text/plain;charset=utf-8;base64,ZGVmYXVsdHMgewogICAgICAgIHVzZXJfZnJpZW5kbHlfbmFtZXMgeWVzCiAgICAgICAgZmluZF9tdWx0aXBhdGhzIHllcwp9CgpibGFja2xpc3RfZXhjZXB0aW9ucyB7CiAgICAgICAgcHJvcGVydHkgIihTQ1NjX01ERU5UX3xJRF9XV04pIgp9CgpibGFja2xpc3Qgewp9Cgo=
          verification: {}
      filesystem: root
      mode: 400
      path: /etc/multipath.conf
    systemd:
      units:
      - name: iscsid.service
        enabled: true
        state: started
      - name: multipathd.service
        enabled: true
        state: started
  osImageURL: ""

```

2. Once the configuration is created it will take approximately 20 to 30 minutes to apply the configuration on the worker nodes. During that time the Updating section will be “True” under Compute > MachineConfigPool

MachineConfigPools						Create MachineConfigPool
Name ▾	Search by name... 					
Name ↑	Configuration ↓	Updated	Updating	Paused	Degraded	
 master	 rendered-master-d5579ee21a6eba71ba735b962ca35c83	True	False	False	False	⋮
 worker	 rendered-worker-0886ee17a5ecfd33459b82ee5c54370a	False	True	False	False	⋮

- Once the configuration is applied followed by a reboot the Updated field will be set to “True” for the worker node.

MachineConfigPools						
Name ▾ Search by name... 		<a href="#">Create MachineConfigPool</a>				
Name ↑	Configuration ↓	Updated	Updating	Paused	Degraded	
 master	 rendered-master-d5579ee21a6eba71ba735b962ca35c83	True	False	False	False	⋮
 worker	 rendered-worker-60732d7b7485e556065befcc6fb1ff95	True	False	False	False	⋮

- Verify that both iSCSI and multipath services are up and running by login into the worker nodes.

```
systemctl status iscsid
```

```
[core@worker-0 ~]$ systemctl status iscsid
● iscsid.service - Open-iSCSI
   Loaded: loaded (/usr/lib/systemd/system/iscsid.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2021-07-27 08:56:45 UTC; 30min ago
     Docs: man:iscsid(8)
           man:iscsiuio(8)
           man:iscsiadm(8)
```

```
systemctl status multipathd
```

```
[core@worker-0 ~]$ systemctl status multipathd
● multipathd.service - Device-Mapper Multipath Device Controller
   Loaded: loaded (/usr/lib/systemd/system/multipathd.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2021-07-27 08:56:42 UTC; 31min ago
 Main PID: 2589 (multipathd)
    Status: "up"
     Tasks: 7
  Memory: 14.3M
     CPU: 178ms
   CGroup: /system.slice/multipathd.service
           └─2589 /sbin/multipathd -d -s
```

## Provision NAS Storage for the Applications Running in OCP

To provision NAS storage, follow these steps:

- Create a NAS Backend - paste the below content into a yaml file. Example - ocp\_nas\_backend.yaml:

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ocp-nas-backend",
  "managementLIF": "Management_LIF_IP",
  "dataLIF": "NFS_LIF_IP",
```

```

"svm": "OCP-SVM",
"username": "admin",
"password": "#####",
"limitAggregateUsage": "80%",
"nfsMountOptions": "nfsvers=3",
"defaults": {
    "spaceReserve": "volume",
    "exportPolicy": "default",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
}
}

```

```
[root@fp-ansible-2 OCP_Trident]# cat ocp_nas_backend.yaml
```

```

{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ocp-nas-backend",
  "managementLIF": " ",
  "dataLIF": " ",
  "svm": "OCP-SVM",
  "username": "admin",
  "password": " ",
  "limitAggregateUsage": "80%",
  "nfsMountOptions": "nfsvers=3",
  "defaults": {
    "spaceReserve": "volume",
    "exportPolicy": "default",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
  }
}

```

2. Run the tridentctl command with the yaml file to create the backend.

```
tridentctl create backend -f ocp_nas_backend.yaml -n trident
```

```
[root@fp-ansible-2 OCP_Trident]# tridentctl create backend -f ocp_nas_backend.yaml -n trident
```

NAME	STORAGE DRIVER	UUID	STATE	VOLUMES
ocp-nas-backend	ontap-nas	31970319-f43f-4a37-9116-612bc859d037	online	0

3. Create a NAS Storage Class - paste the following content into a yaml file, for example ocp\_nas\_sc.yaml:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ocp-nas-sc-gold
provisioner: netapp.io/trident

```

```
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
```

```
[root@fp-ansible-2 OCP_Trident]# cat ocp_nas_sc.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ocp-nas-sc-gold
provisioner: netapp.io/trident
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
allowVolumeExpansion: true
[root@fp-ansible-2 OCP_Trident]# oc create -f ocp_nas_sc.yaml
storageclass.storage.k8s.io/ocp-nas-sc-gold created
[root@fp-ansible-2 OCP_Trident]#
```

#### 4. Create the NAS storage class using the yaml file:

```
oc create -f ocp_nas_sc.yaml
```

```
[root@fp-ansible-2 OCP_Trident]# oc create -f ocp_nas_sc.yaml
storageclass.storage.k8s.io/ocp-nas-sc-gold created
[root@fp-ansible-2 OCP_Trident]#
[root@fp-ansible-2 OCP_Trident]# oc get sc
NAME                                PROVISIONER             RECLAIMPOLICY    VOLUMEBINDINGMODE    ALLOWVOLUMEEXPANSION    AGE
ocp-nas-sc-gold                    csi.trident.netapp.io   Delete           Immediate             true                    11s
[root@fp-ansible-2 OCP_Trident]#
```

#### 5. Create a NAS persistent volume claim (PVC) that uses the storage class that was created in the previous step:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ocp-nas-pvc
  namespace: trident
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: ocp-nas-sc-gold
```

```
[root@fp-ansible-2 OCP_Trident]# cat ocp_nas_pvc.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ocp-nas-pvc
  namespace: trident
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 10Gi
  storageClassName: ocp-nas-sc-gold
```

6. Create the NAS PVC from the yaml file:

```
oc create -f ocp_nas_pvc.yaml -n trident
```

```
[root@fp-ansible-2 OCP_Trident]# oc create -f ocp_nas_pvc.yaml -n trident
persistentvolumeclaim/ocp-nas-pvc created
```

7. From the ONTAP CLI verify the newly created volume with a size of 10 GB.

```
aail-a400:> volume show -vserver OCP-SVM
```

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
OCP-SVM	ocp_root	aggr01_node01	online	RW	1GB	969.5MB	0%
OCP-SVM	ocp_rootvol_m01	aggr01_node01	online	LS	1GB	969.5MB	0%
OCP-SVM	ocp_rootvol_m02	aggr01_node02	online	LS	1GB	969.5MB	0%
OCP-SVM	trident_pvc_bfe1361f_2ad3_4360_9589_4260c2dc445a	aggr01_node01	online	RW	10GB	9.00GB	0%

4 entries were displayed.

## Provision SAN Storage for the Applications running in OCP

To provision the SAN storage, follow these steps:

1. Create an iSCSI Backend; paste the following content into a yaml file. For example, ocp\_san\_backend.yaml.

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
```



```
"backendName": "ocp-san-backend",
"managementLIF": "Management_LIF_IP",
"dataLIF": "iSCSI_LIF_IP",
"svm": "OCP-SVM",
"username": "admin",
"password": "#####",
"defaults": {
    "spaceReserve": "volume",
    "spaceAllocation": "false",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
}
}
```

```
[root@fp-ansible-2 OCP_Trident]# cat ocp_san_backend.yaml
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "ocp-san-backend",
  "managementLIF": " ",
  "dataLIF": " ",
  "svm": "OCP-SVM",
  "username": "admin",
  "password": " ",
  "defaults": {
    "spaceReserve": "volume",
    "spaceAllocation": "false",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
  }
}
[root@fp-ansible-2 OCP_Trident]#
```

2. Create a SAN storage class; paste the following content into a yaml file, for example ocp\_san\_storageclass.yaml:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ocp-san-gold
```

```
provisioner: csi.trident.netapp.io
mountOptions:
- discard
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
```

```
[root@fp-ansible-2 OCP_Trident]# cat ocp_san_storageclass.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ocp-san-gold
provisioner: csi.trident.netapp.io
mountOptions:
- discard
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
```

### 3. Create the SAN storage class using the yaml file:

```
[root@fp-ansible-2 OCP_Trident]# oc create -f ocp_san_storageclass.yaml
[root@fp-ansible-2 OCP_Trident]# oc create -f ocp_san_storageclass.yaml
storageclass.storage.k8s.io/ocp-san-gold created
[root@fp-ansible-2 OCP_Trident]#
```

### 4. Create a SAN persistent volume claim (PVC) using the storage class created in the previous step:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ocp-san-pvc
  namespace: trident
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: ocp-san-gold
```



The accessModes for iSCSI Volumes cannot be set as RWX (Read-Write-Many) since iSCSI disks cannot be shared between hosts with write access.

---

```
[root@fp-ansible-2 OCP_Trident]# cat ocp_san_pvc.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ocp-san-pvc
  namespace: trident
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi
  storageClassName: ocp-san-gold
[root@fp-ansible-2 OCP_Trident]#
```

5. Create the SAN storage PVC:

```
[root@fp-ansible-2 OCP_Trident]# oc create -f ocp_san_pvc.yaml -n trident
[root@fp-ansible-2 OCP_Trident]# oc create -f ocp_san_pvc.yaml -n trident
persistentvolumeclaim/ocp-san-pvc created
```

## Use Persistent Volumes to Deploy Applications in OpenShift Container Platform

The Persistent Volume Claims that were created earlier using NAS and SAN backends can be declared in deployment.yaml files to mount the associated Persistent Volumes to Pods/ Containers in the OpenShift Container Platform Kubernetes environment.

Below is a sample deployment.yaml file that uses the 'ocp-nas-pvc' Persistent Volume Claim to provide persistent storage to the nginx web application's html pages:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx
  namespace: nginx
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
```

```
    app: nginx
spec:
  containers:
  - name: nginx
    imagePullPolicy: Always
    image: nginx
    ports:
    - containerPort: 80
    volumeMounts:
    - mountPath: /usr/share/nginx/html
      name: nginx-data
  volumes:
  - name: nginx-data
    persistentVolumeClaim:
      claimName: ocp-nas-pvc
```

A demonstration video that details the NetApp Astra Trident's Data Management features for OpenShift Kubernetes operations is available here [Data Management with Containers Featuring Persistence with Trident](#).

## Deploy OpenShift Virtualization

To deploy OpenShift virtualization, after subscribing to the OpenShift Virtualization catalog, create the OpenShift Virtualization Operator Deployment custom resource to deploy OpenShift Virtualization, and follow these steps:

1. Navigate to the Operators > Installed Operators page.
2. Click OpenShift Virtualization.
3. Click the OpenShift Virtualization Operator Deployment tab and click Create HyperConverged.

[Installed Operators](#) > [Operator details](#)



**OpenShift Virtualization**  
2.6.5 provided by Red Hat

[Details](#)

[YAML](#)

[Subscription](#)

[Events](#)

[All instances](#)

[OpenShift Virtualization Deploy](#)



**HyperConverged required**

Create a HyperConverged instance to use this Operator.

[Create HyperConverged](#)

4. Click Create to launch OpenShift Virtualization. Enter ocp-nas-sc-gold as the Storage Class Name and enable Bare Metal Platform. Click Create.

Project: openshift-cnv ▼

OpenShift Virtualization > Create HyperConverged

## Create HyperConverged

Create by completing the form. Default values may be provided by the Operator authors.

Configure via: ☒ Form view ☐ YAML view

**Note:** Some fields may not be represented in this form view. Please select "YAML view" for full control.



OpenShift Virtualization Deployment  
provided by Red Hat

Represents the deployment of OpenShift Virtualization

Name \*

kubevirt-hyperconverged

Labels

app=frontend

Infra



infra HyperConvergedConfig influences the pod configuration (currently only placement) for all the infra components needed on the virtualization enabled cluster but not necessarily directly on each node running VMs/VMLs.

Workloads



workloads HyperConvergedConfig influences the pod configuration (currently only placement) of components which need to be running on a node where virtualization workloads should be able to run. Changes to Workloads HyperConvergedConfig can be applied only without existing workload.

Bare Metal Platform



true

BareMetalPlatform indicates whether the infrastructure is baremetal.

Feature Gates



featureGates is a map of feature gate flags. Setting a flag to `true` will enable the feature. Setting `false` or removing the feature gate, disables the feature.

Local Storage Class Name

ocp-nas-sc-gold

LocalStorageClassName the name of the local storage class.

Create

Cancel

5. Navigate to the Workloads > Pods page and monitor the OpenShift Virtualization pods under the openshift-cnv project until they are all Running. After all the pods display the Running state, you can access OpenShift Virtualization.

Red Hat

OpenShift

Container Platform

☰

Administrator

Home

Operators

OperatorHub

Installed Operators

Workloads

Pods

Virtualization

Deployments

DeploymentConfigs

StatefulSets

Secrets

ConfigMaps

CronJobs

Jobs

DaemonSets

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Project: openshift-cnv

nmstate-webhook-b4b9747f-shvtn	Running	1/1	0	nmstate-webhook-b4b9747f	38.2 MB	0.001 cores	Mar 19, 4:35 pm
node-maintenance-operator-79f56bf5dc-f7m4	Running	1/1	0	node-maintenance-operator-79f56bf5dc	375 MB	0.002 cores	Mar 19, 4:32 pm
hup-operator-5659c478c4-dsv8	Running	1/1	0	hup-operator-5659c478c4	91.3 MB	0.004 cores	Mar 19, 4:32 pm
virt-api-75b69cccf-p8d8n	Running	1/1	0	virt-api-75b69cccf	1377 MB	0.003 cores	Mar 19, 4:35 pm
virt-api-75b69cccf-nfjzd	Running	1/1	0	virt-api-75b69cccf	149.0 MB	0.003 cores	Mar 19, 4:35 pm
virt-controller-5cbc57b8bb-v76c	Running	1/1	0	virt-controller-5cbc57b8bb	135.4 MB	0.005 cores	Mar 19, 4:35 pm
virt-controller-5cbc57b8bb-wx8hs	Running	1/1	0	virt-controller-5cbc57b8bb	147.6 MB	0.005 cores	Mar 19, 4:35 pm
virt-handler-8rnfd	Running	1/1	0	virt-handler	154.9 MB	0.004 cores	Mar 19, 4:35 pm
virt-handler-54546	Running	1/1	0	virt-handler	163.5 MB	0.004 cores	Mar 19, 4:35 pm
virt-operator-6c648875bd-m27zc	Running	1/1	0	virt-operator-6c648875bd	135.5 MB	0.003 cores	Mar 19, 4:32 pm
virt-operator-6c648875bd-cvxfz	Running	1/1	0	virt-operator-6c648875bd	132.2 MB	0.003 cores	Mar 19, 4:32 pm
virt-template-validator-7899b4845c-56hgz	Running	1/1	0	virt-template-validator-7899b4845c	41.9 MB	0.002 cores	Mar 19, 4:35 pm

## Connect Virtual Machines to an External Network

Each VM deployed is controlled via a virt-launcher pod that is created with each VM. The default networking type for OpenShift Virtualization VMs is Masquerade. The VM will be assigned a non-routable IP and you can access the VM using the IP of the virt-launcher pod that was deployed alongside it. This makes the VM accessible in the same way that containers are accessed.

Alternatively, you can connect the VM to the host network by creating a bridge interface on the OCP nodes using Nmstate. The Nmstate operator is installed with OpenShift Virtualization and provides you with the Node Network Configuration Policy (NNCP) object to update the host network settings. Container-native virtualization uses nmstate to report on and configure the state of the node network. This makes it possible to modify network policy configuration, such as by creating a Linux bridge on all nodes, by applying a single configuration manifest to the cluster.

The following is a sample configuration to create a bridge called br1 from an interface called eth1 on the OCP nodes.

## Network Configuration using Nmstate

OpenShift Virtualization uses nmstate to report on and configure the state of the node network. This makes it possible to modify network policy configuration, such as by creating a Linux bridge on all nodes, by applying a single configuration manifest to the cluster.

To configure the network using Nmstate, follow these steps:

1. Verify the state of the network on the OCP nodes.

```
[kni@fp-ansible ~]$ oc get nns
NAME                                     AGE
master-0.ocp.flexpod.cisco.com         2m56s
master-1.ocp.flexpod.cisco.com         2m54s
master-2.ocp.flexpod.cisco.com         2m54s
worker-0.ocp.flexpod.cisco.com         2m52s
worker-1.ocp.flexpod.cisco.com         2m51s
```

```
worker-2.ocp.flexpod.cisco.com 2m59s
```

2. Inspect a NodeNetworkState object to view the network on a worker node within the cluster .

```
[kni@fp-ansible ~]$ oc get nns worker-0.ocp.flexpod.cisco.com -o yaml
```

3. Create an interface on nodes in the cluster by applying a NodeNetworkConfigurationPolicy manifest to the cluster. The manifest details the requested configuration for the interface. You will use the dedicated interface you created on Cisco UCS for OpenShift virtualization.



By default, the manifest applies to all nodes in the cluster. To add the interface to specific nodes, add the spec: nodeSelector parameter and the appropriate <key>:<value> for your node selector.

---

4. Create the NodeNetworkConfigurationPolicy manifest. The following example configures a new Linux bridge on all worker nodes with one interface eno7 with no IP address assigned and name it br1:

```
[kni@fp-ansible ~]$ cat linux-bridge.yaml
apiVersion: nmstate.io/v1beta1
kind: NodeNetworkConfigurationPolicy
metadata:
  name: worker-br1
spec:
  nodeSelector:
    node-role.kubernetes.io/worker: ''
  desiredState:
    interfaces:
      - name: br1
        description: Linux bridge with eno7 as a port
        type: linux-bridge
        state: up
        ipv4:
          dhcp: true
          enabled: true
        bridge:
          options:
            stp:
              enabled: false
          port:
            - name: eno7
```

5. Create Linux bridge using nm state operator using the file created with contents shown in the previous step.

```
[kni@fp-ansible ~]$ oc apply -f linux-bridge.yaml
nodenetworkconfigurationpolicy.nmstate.io/worker-br1 created
```



## 6. Verify created node network policy using nnce:

```
[kni@fp-ansible ~]$ oc get nnce
```

NAME	STATUS
master-0.ocp.flexpod.cisco.com.worker-br1	NodeSelectorNotMatching
master-1.ocp.flexpod.cisco.com.worker-br1	NodeSelectorNotMatching
master-2.ocp.flexpod.cisco.com.worker-br1	NodeSelectorNotMatching
worker-0.ocp.flexpod.cisco.com.worker-br1	SuccessfullyConfigured
worker-1.ocp.flexpod.cisco.com.worker-br1	SuccessfullyConfigured
worker-2.ocp.flexpod.cisco.com.worker-br1	SuccessfullyConfigured

## Create Network Attachment Definition

To create the network attachment definition to allow VMs to be able to connect to external network using the dedicated vNIC on Cisco UCS, follow these steps:

1. In the OpenShift GUI console, click the Networking drop-down list and select Network Attachment Definitions. The Network Attachment Definition screen lists your network attachment definitions.
2. Click Create Network Attachment Definition. The Create Network Attachment Definition page displays. Complete the following fields based on the bridge configuration you created to connect to an external network.
3. Enter a Name for the network attachment definition.
4. Optionally, enter a Description of the network attachment definition.
5. From Network Type, select CNV Linux bridge.
6. Enter the Bridge name.
7. Optionally, enter the VLAN Tag Number.

Project: openshift-cnv ▼

---

## Create Network Attachment Definition

[Edit YAML](#)

Name \*

ucs-ext-network

Description

Network Type \*

CNV Linux bridge ▼

Bridge Name \*

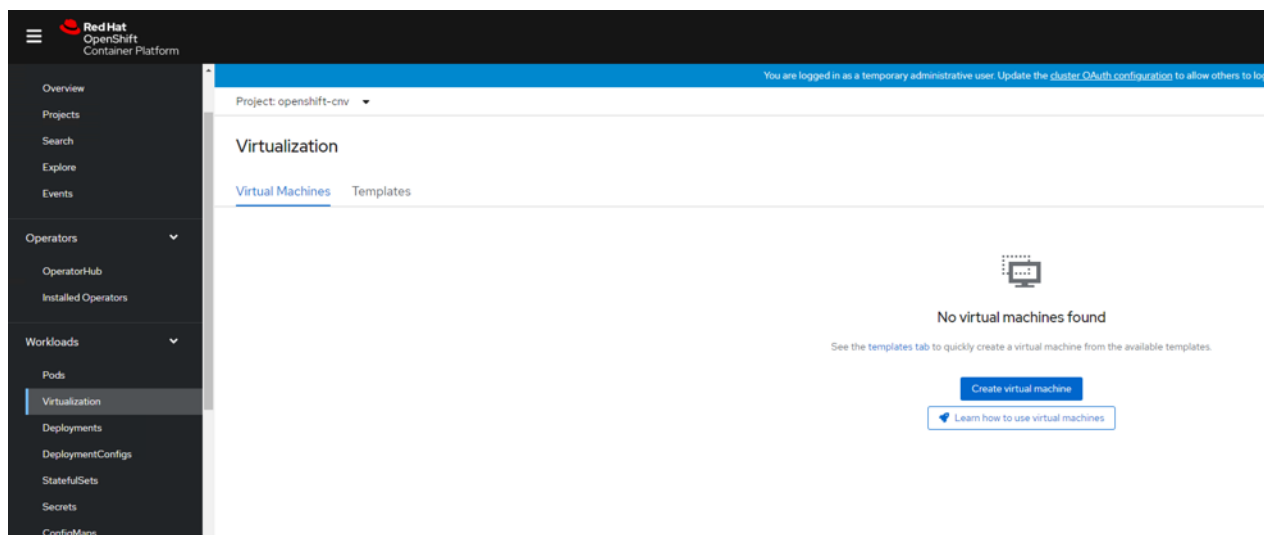
br1

VLAN Tag Number

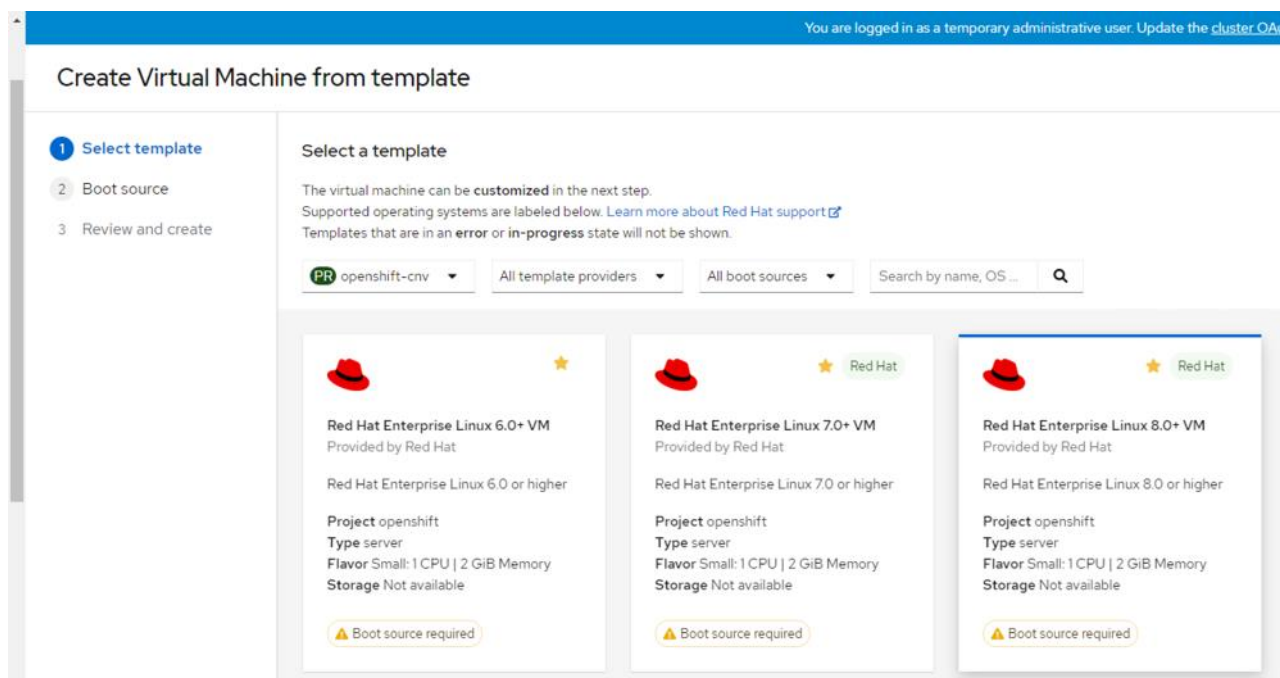
Create

Cancel

- Click Create.
- While logged into OpenShift Console, click Workloads > Virtualization.
- Click Create virtual machine.



11. Select the Operating System type and version. In this case we have selected RHEL 8.0+ VM to install a RHEL 8.0 VM using the image previously downloaded and hosted on a web server.



12. Click Next.
13. In the Boot source type, select Import via URL.
14. In the Import URL, add the webserver information hosting your image, [http://10.1.156.150/OCBPM/RHEL/rhel-8.3-update-2-x86\\_64-kvm.qcow2](http://10.1.156.150/OCBPM/RHEL/rhel-8.3-update-2-x86_64-kvm.qcow2), or follow the prompts to download the KVM guest image to your web server.
15. Leave Persistent Volume Claim size as “ 20 GB”, change it if needed.

16. Click Customize virtual machine.

## Create Virtual Machine from template

- 1 Select template
- 2 **Boot source**
- 3 Review and create

### Boot source

This template does not have a boot source. Provide a custom boot source for this **Red Hat Enterprise Linux 8.0+ VM** virtual machine.

#### Boot source type \*

Import via URL (creates PVC) ▼

#### Import URL \*

http://10.156.150/OCPBIM/RHEL/rhel-8.3-update-2-x86\_64-kvm.qcow2

Example: For RHEL, visit the [RHEL download page](#) (requires login) and copy the download link URL of the KVM guest image

☐ Mount this as a CD-ROM boot source ⓘ

#### Persistent Volume Claim size \*

20 GiB ▼

Ensure your PVC size covers the requirements of the uncompressed image and any other space requirements. More storage can be added later.

#### ▼ Advanced

#### Storage class \*

ocp-nas-sc-gold ▼

#### Access mode \*

Single User (RWO) ▼

#### Volume mode \*

Filesystem ▼



**Access and Volume modes should follow storage feature matrix**

[Learn more](#) ⓘ

Next

Customize virtual machine

Back

Cancel



---

Optionally, change the size of the VM and Workload Type based on what application will be installed on this VM.

---

17. Change the name of this VM and click Next.

## Create Virtual Machine from Red Hat Enterprise Linux 8.0+ VM

1 General

2 Networking

3 Storage

4 Advanced

5 Review

6 Result

Name \*

ocp-test-vm

Description

Operating System \*

Red Hat Enterprise Linux 8.0 or higher ▼

Operating system image not available. You can either [upload a new disk image](#) or define a boot source manually in the boot source dropdown

Boot Source \* ⓘ

Import via URL (creates PVC) ▼

Enter URL below or edit the rootdisk in the [Storage](#) step.

To boot this source from a CD-ROM, edit the disk in the storage step and change to type: CD-ROM

URL

[http://10.1156.150/OCPBm/RHEL/rhel-8.3-update-2-x86\\_64-kvm.qcow2](http://10.1156.150/OCPBm/RHEL/rhel-8.3-update-2-x86_64-kvm.qcow2)

Example: For RHEL, visit the [RHEL download page](#) (requires login) and copy the download link URL of the KVM guest image

Flavor \* ⓘ

Small (default): 1 CPU | 2 GiB Memory ▼

Workload Type \* ⓘ

Server (default) ▼

Next

Review and confirm

Back

Cancel

18. Click Add Network Interface on the right size of the page:



A new Network interface will be created along with the Pod Networking interface to provide VM the ability to have external connectivity as well as connectivity to POD network to communicate with the PODs deployed if there is an application consisting of VMs and container PODs. Optionally, click the three virtual dots on the default network interface displayed to change the network from Pod Networking to the external network created in previous section to have external connectivity for the VM or leave it to defaults to have connectivity to POD network only.

### Add Network Interface

Name \*

Model \*

Network \*

Type \*

MAC Address

CancelAdd

19. Change the Network Type to ucs-ext-network.

20. Click Add.

You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to login.

Project: openshift-cnv

### Create Virtual Machine from Red Hat Enterprise Linux 8.0+ VM

1 General  
2 **Networking**  
3 Storage  
4 Advanced  
5 Review  
6 Result

Network Interfaces

Name	Model	Network	Type	MAC Address
default	virtio	Pod Networking	macvlan	-
nic-0	virtio	vcc-wed-network	bridge	-

[Add Network Interface](#)

21. Click Next and click Next again to review the information Click Create Virtual Machine to create the VM.



## Successfully created virtual machine

You can either go to the details of this virtual machine or see it in the list of available virtual machines.

[See virtual machine details](#)

[Go to list](#)

22. Click See virtual machine details to verify the VM details and IP addresses assigned after a few minutes. For external access to the VM, either a DHCP server can be setup in the OpenShift Virtualization VLAN and L3 access configured, or a static IP address can be assigned in the VM once it has been brought up.




## Details

---


### Name

ocp-test-vm

### Namespace

 openshift-cnv

### Created

 3 minutes ago

### Hostname

localhost.localdomain

### Node

 worker-1.ocp.flexpod.cisco.com

### IP Address

10.128.2.131, 10.4.156.110, fe80::34b5:6e8a:80bf:355d

### Operating System

Red Hat Enterprise Linux 8.3 (Ootpa)

### Time Zone

EDT

### Active Users

No users logged in

The following image shows a 20G volume created for the VM dynamically on the NetApp cluster.

trident\_pvc\_f4427e04\_21e4\_4957\_9723\_68f16c1d11a2 [All Volumes](#)

[Edit](#) [More](#)

Overview

[Snapshot Copies](#)

[SnapMirror \(Local or Remote\)](#)

[Explorer](#)

STATUS

✓ Online

STYLE

FlexVol

MOUNT PATH

/trident\_pvc\_f4427e04\_21e4\_4957\_9723\_68f16c1d11a2

STORAGE VM

[OCP47-SVM](#)

LOCAL TIER

[aggr1\\_node01](#)

SNAPSHOT POLICY

[default](#)

QUOTA

Off

TYPE

Read/Write

SPACE RESERVATION

Thin Provisioned

STORAGE EFFICIENCY

Enabled

EXPORT POLICY

▼ [default](#)

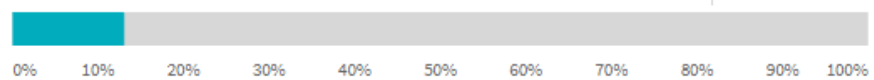
## Capacity

2.35 GB

USED

15.9 GB

AVAILABLE



SNAPSHOT CAPACITY

2 GB Available | 0 Bytes Used

INACTIVE DATA STORED LOCALLY

700 KB  
INACTIVE

## Performance



Hour

Day

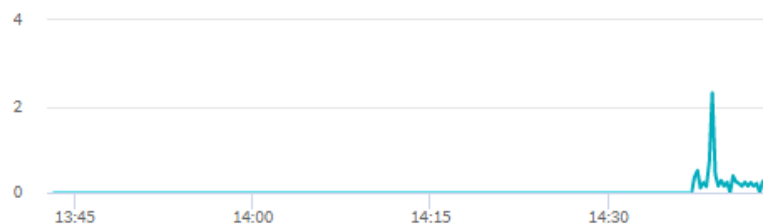
Week

Month

Year

Latency

0 ms



23. Click the Console tab to verify the VM installed successfully and to verify the VM access password.

Project: openshift-cnv ▾

Virtualization > Virtual Machines > ocp-test-vm Details

**VM** ocp-test-vm

Actions ▾

Overview Details YAML Environment Events Console Network Interfaces Disks Snapshots

Open Console in new Window

**i Guest login credentials**

The following credentials for this operating system were created via **Cloud-init**. If unsuccessful cloud-init could be improperly configured. Please contact the image provider for more information.

**User name:** cloud-user **Password:** [Show password](#)

VNC Console

☐ Disconnect before switching

Send Key

```
Red Hat Enterprise Linux 8.3 (Ootpa)
Kernel 4.18.0-240.15.1.el8_3.x86_64 on an x86_64

Activate the web console with: systemctl enable --now cockpit.socket

ocp-test-vm login: _
```

24. Log into the VM using External IP address or via console and access credentials to verify connectivity to the External network as well as the POD Network.

```
[cloud-user@ocp-test-vm ~]# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1400
    inet 10.0.2.2 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::5054:ff:fe75:3987 prefixlen 64 scopeid 0x20<link>
    ether 52:54:00:75:39:87 txqueuelen 1000 (Ethernet)
    RX packets 46 bytes 6180 (6.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 71 bytes 6480 (6.3 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.4.156.110 netmask 255.255.255.0 broadcast 10.4.156.255
    inet6 fe80::34b5:6e8a:80bf:355d prefixlen 64 scopeid 0x20<link>
    ether 4e:9a:4e:31:a3:27 txqueuelen 1000 (Ethernet)
    RX packets 360 bytes 26472 (25.8 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 18 bytes 1854 (1.8 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 22 bytes 1776 (1.7 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 22 bytes 1776 (1.7 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

---

## References

### Products and Solutions

Cisco Unified Computing System:

<http://www.cisco.com/en/US/products/ps10265/index.html>

Cisco UCS 6400 Series Fabric Interconnects:

<https://www.cisco.com/c/en/us/support/servers-unified-computing/ucs-6400-series-fabric-interconnects/tsdproducts-support-series-home.html>

Cisco UCS 5100 Series Blade Server Chassis:

<http://www.isco.com/en/US/products/ps10279/index.html>

Cisco UCS B-Series Blade Servers:

<http://www.cisco.com/c/en/us/products/servers-unified-computing/ucs-b-series-blade-servers/index.html>

Cisco UCS C-Series Rack Servers:

<http://www.cisco.com/c/en/us/products/servers-unified-computing/ucs-c-series-rack-servers/index.html>

Cisco UCS Adapters:

[http://www.cisco.com/en/US/products/ps10277/prod\\_module\\_series\\_home.html](http://www.cisco.com/en/US/products/ps10277/prod_module_series_home.html)

Cisco UCS Manager:

<http://www.cisco.com/en/US/products/ps10281/index.html>

Cisco Intersight:

<https://www.cisco.com/c/en/us/products/servers-unified-computing/intersight/index.html>

Cisco Nexus 9000 Series Switches:

<http://www.cisco.com/c/en/us/support/switches/nexus-9000-series-switches/tsd-products-support-serieshome.html>

Red Hat OpenShift:

<https://www.openshift.com/>

---

## Summary

This solution design based on Cisco FlexPod User Provisioned Infrastructure for Red Hat OpenShift Container Platform 4.7 is optimal for production-ready deployment processes with latest best practices, a stable, highly available environment to run enterprise-grade application containers offered as infrastructure as code with fully automated deployment of FlexPod infrastructure and installation of OCP cluster.

FlexPod is the optimal shared infrastructure foundation to deploy a variety of IT workloads. It is built on leading computing, networking, storage, and infrastructure software components. The integration of FlexPod converged infrastructure with OpenShift Container Platform provides a very good starting point for enterprise IT to make inroads into DevOps and CI/CD model for application development to address immediate business needs and reducing time to market. Enterprises can accelerate on the path to an enterprise-grade Kubernetes solution with Red Hat OpenShift Container Platform running on Cisco FlexPod infrastructure installer provisioned infrastructure.

---

## About the Authors

John George, Technical Marketing Engineer, Cisco UCS Data Center Solutions Engineering, Cisco Systems, Inc.

John has been involved in designing, developing, validating, and supporting the FlexPod Converged Infrastructure since it was developed over nine years ago. Before his roles with FlexPod, he supported and administered a large worldwide training network and VPN infrastructure. John holds a Master's degree in Computer Engineering from Clemson University.

Arvind Ramakrishnan, Sr. Solutions Architect, Hybrid Cloud Infrastructures, NetApp Inc.

Arvind focusses on development, validation and implementation of Hybrid Cloud Infrastructure solutions that include NetApp products. He has more than 10 years of experience in the IT industry specializing in Data Management, Security, Cloud and Data Center technologies. Arvind holds a bachelor's degree in Electronics and Communication and a master's degree in Compute Systems and Infrastructure.

Abhinav Singh, Technical Marketing Engineer, Hybrid Cloud Infrastructures, NetApp Inc.

Abhinav has more than 11 years of experience in Data Center infrastructure solutions which includes On-prem and Hybrid cloud space. He focuses on the validating, supporting, implementing cloud infrastructure solutions that include NetApp products. Prior joining to NetApp, he was with Cisco Systems India as a Technical Consulting Engineer working on Cisco Application Centric Infrastructure (ACI). Abhinav holds multiple certifications like CCNP Enterprise, Cisco Specialist - Data Center Core, VMware Double VCP (DCV, NSX-T), NSX-V Implementation Expert. Abhinav holds a bachelor's degree in Electrical & Electronics.

## Acknowledgements

For their support and contribution to the design, validation, and creation of this Cisco Validated Design, the authors would like to thank:

- Sreeni Edula, Technical Marketing Engineer, Cisco Systems, Inc.
- Haseeb Niazi, Technical Marketing Engineer, Cisco Systems, Inc.

---

## Feedback

For comments and suggestions about this guide and related guides, join the discussion on [Cisco Community](https://cs.co/en-cvds) at <https://cs.co/en-cvds>.



---

**Americas Headquarters**

Cisco Systems, Inc.  
San Jose, CA

**Asia Pacific Headquarters**

Cisco Systems (USA) Pte. Ltd.  
Singapore

**Europe Headquarters**

Cisco Systems International BV Amsterdam,  
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at <https://www.cisco.com/go/offices>.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: <https://www.cisco.com/go/trademarks>. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1110R)