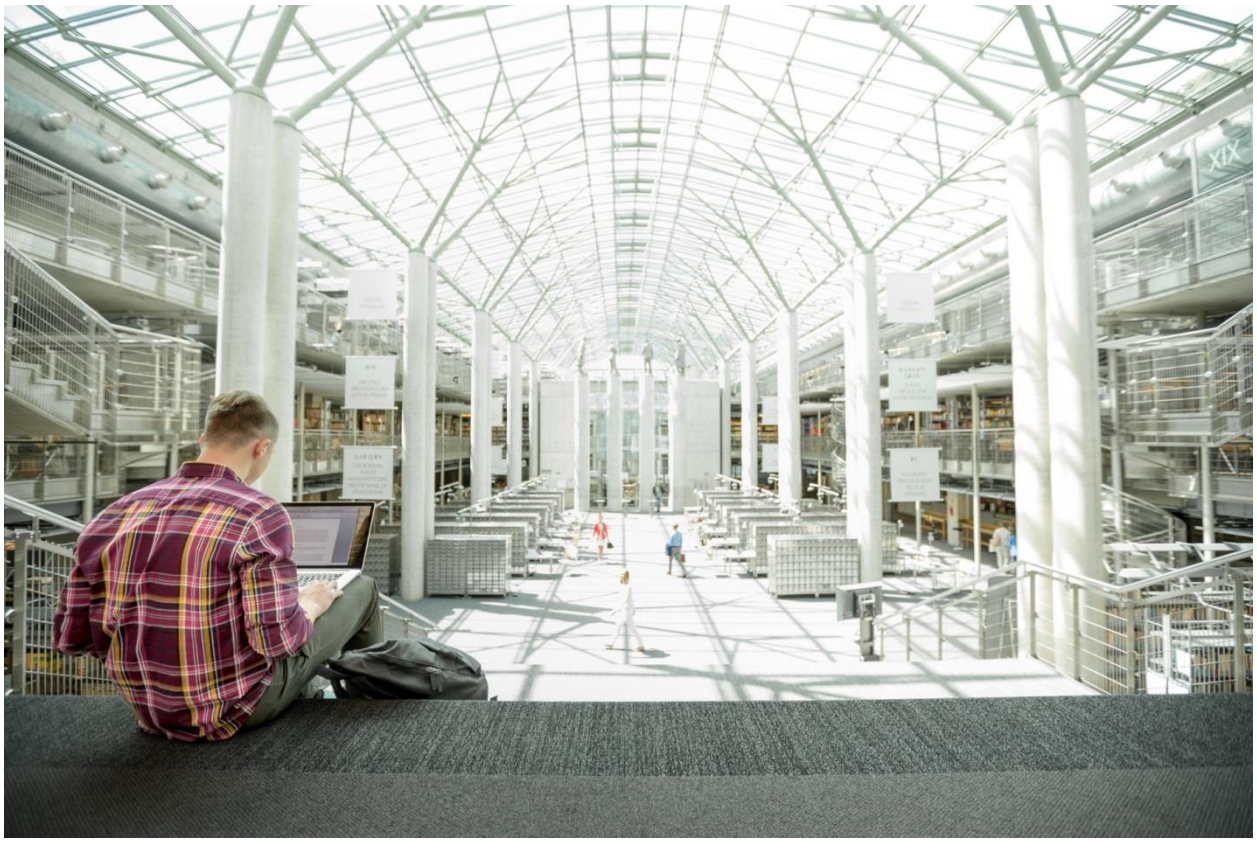


Cisco UCS Infrastructure for Red Hat OpenShift Container Platform Design Guide

Cisco UCS Infrastructure for Red Hat OpenShift Container Platform 3.9

Last Updated: July 24, 2018

CISCO VALIDATED DESIGN

About the Cisco Validated Design (CVD) Program

The CVD program consists of systems and solutions designed, tested, and documented to facilitate faster, more reliable, and more predictable customer deployments. For more information visit

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 Internetwork 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, GigaDrive, HomeLink, Internet Quotient, IOS, iPhone, iQuick Study, LightStream, Linksys, MediaTone, MeetingPlace, MeetingPlace Chime Sound, MGX, Networkers, Networking Academy, Network Registrar, PCNow, PIX, PowerPanels, 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.

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)

© 2018 Cisco Systems, Inc. All rights reserved.

Table of Contents

Executive Summary

Business Challenges

Businesses are increasingly expecting technology to be a centerpiece of every new change. Traditional approaches to leverage technology are too slow in delivering innovation at the pace at which business ecosystems are changing.

To stay competitive, organizations need customized software applications to meet their unique needs – from customer engagements to new product and services development. Therefore, the need to speed up application development, testing, delivery, and deployment is becoming a necessary business competency.

In order to keep up with new technologies or stay one step ahead, enterprises will have to overcome key challenges to accelerate product development, add value and compete better at lower cost.

Key Challenges:

- Policy Management: Large enterprises have redundant processes and policies in place, and are reluctant to adopt new technologies due to fear of breaking compliance causing delays in new development/release cycle.
- Portability: Applications have dependencies around OS versions, libraries, Java versions, etc. Any changes to these dependencies can break portability which means applications developed on a specific environment may behave differently on a production environment hosted on-premises or cloud. Changing code and rebuilding/testing code leads to delay in product or service offering and loss of market share.
- Agility: Enterprises have many legacy applications, tools and complex development process slowing innovation significantly as product release cycle takes days to weeks due to complex flow from development to production.
- Resource Utilization: Engineering and hardware resources in large enterprises are not used efficiently due to various organizations operating in silos. These silos that worked in the past are causing a huge overhead in development/release cycle and resource under-utilization as technology changes rapidly.

Our Solution

Deployment-centric application platform and DevOps initiatives are driving benefits for organizations in their digital transformation journey. Though still early in maturity, Docker format container packaging and Kubernetes container orchestration are emerging to cater to the rapid digital transformation. Docker format container packaging and Kubernetes container orchestration are emerging as industry standards for state-of-the-art PaaS solutions.

Containers have brought a lot of excitement and value to IT by establishing predictability between building and running applications. Developers can trust and know that their application will perform the same when

it's run on a production environment as it did when it was built, while operations and admins have the tools to operate and maintain applications seamlessly.

Red Hat® OpenShift® Container Platform provides a set of container-based open source tools enabling digital transformation, which accelerates application development while making optimal use of infrastructure. Professional developers utilize fine-grained control of all aspects of the application stack, with application configurations enabling rapid response to unforeseen events. Availability of highly secure operating systems assists in standing up an environment capable of withstanding continuously changing security threats, helping deployment with highly secure applications.

Red Hat OpenShift Container Platform helps organizations use the cloud delivery model and simplify continuous delivery of applications and services on Red Hat OpenShift Container Platform, the cloud-native way. Built on proven open source technologies, Red Hat OpenShift Container Platform also provides development teams multiple modernization options to enable a smooth transition to microservices architecture and the cloud for existing traditional applications.

Cisco Unified Computing System™ (Cisco UCS®) servers adapt to meet rapidly changing business needs, including just-in-time deployment of new computing resources to meet requirements and improve business outcomes. With Cisco UCS, you can tune your environment to support the unique needs of each application while powering all your server workloads on a centrally managed, highly scalable system. Cisco UCS brings the flexibility of non-virtualized and virtualized systems in a way that no other server architecture can, lowering costs and improving your return on investment (ROI).

Cisco UCS M5 servers built on Intel's powerful Intel® Xeon Scalable processors are unified yet modular, scalable, high-performing, built on infrastructure-as-code for powerful integrations and continuous delivery of distributed applications.

Cisco, Intel and Red Hat have joined hands to develop a best-in-class solution for delivering PaaS solution to the enterprise with ease. And also, to provide the ability to develop, deploy, and manage containers in an on-prem, Private/ Public cloud environments by bringing automation to the table with a robust platform such as Red Hat OpenShift Container Platform.

Implementation Overview

This design guide will provide an insight on preparing, provisioning, deploying, and managing a Red Hat OpenShift Container Platform 3.9–based on-prem environment with additional container-native storage component on Cisco UCS M5 B- and C-Series servers to cater to the stateful application's persistent storage needs.

Solution Benefits

Some of the key benefits of this solution include:

- Red Hat OpenShift
  - Strong, role-based access controls, with integrations to enterprise authentication systems.
  - Powerful, web-scale container orchestration and management with Kubernetes.
  - Integrated Red Hat Enterprise Linux® Atomic Host, optimized for running containers at scale with Security-Enhanced Linux (SELinux) enabled for strong isolation.

  - Integration with public and private registries.
  - Integrated CI/CD tools for secure DevOps practices.
  - A new model for container networking.
  - Modernize application architectures toward microservices.
  - Adopt a consistent application platform for hybrid cloud deployments.
  - Support for remote storage volumes.
  - Persistent storage for stateful cloud-native containerized applications.
- Cisco UCS
  - Reduced datacenter complexities through Cisco UCS infrastructure with a single management control plane for hardware lifecycle management.
  - Easy to deploy and scale the solution.
  - Superior scalability and high-availability.
  - Compute form factor agnostic.
  - Better response with optimal ROI.
  - Optimized hardware footprint for production and dev/test deployments.

Audience

The intended audience for this CVD is system administrators or system architects. Some experience with Cisco UCS, Docker and Red Hat OpenShift technologies might be helpful, but is not required.

Purpose of this Document

This document highlights the benefits of using Cisco UCS M5 servers for Red Hat OpenShift Container Platform 3.9 to efficiently deploy, scale, and manage a production-ready application container environment for enterprise customers. While Cisco UCS infrastructure provides a platform for compute, network and storage needs, Red Hat OpenShift Container Platform provides a single development platform for creating new cloud-native applications and transitioning from monoliths to microservices. The goal of this document is to demonstrate the value that Cisco UCS brings to the data center, such as single-point hardware lifecycle management, highly available compute, network, storage infrastructure and the simplicity of deployment of application container using Red Hat OpenShift Container Platform.

Solution Overview

Introduction

Red Hat OpenShift Container Platform 3.9 is built around a core of application containers powered by Docker, with orchestration and management provided by Kubernetes, on a foundation of Red Hat Enterprise Linux Atomic Host. It provides many enterprise-ready features, like enhanced security features, multitenancy, simplified application deployment, and continuous integration/ continuous deployment tools. With Cisco UCS M5 servers, provisioning and managing the OpenShift Container Platform 3.9 infrastructure becomes practically effortless with stateless computing and single-plane of management, thus giving a resilient solution.

This document describes the system architecture for the OpenShift Container Platform based on Cisco UCS B- and C-Series M5 servers and Cisco Nexus 9000 Series switches. These servers are powered by the Intel® Xeon® Scalable platform, which provides the foundation for a powerful data center platform that creates an evolutionary leap in agility and scalability. Disruptive by design, this innovative processor sets a new level of platform convergence and capabilities across compute, storage, memory, network, and security. This document provides detail of the hardware requirements to support various OpenShift node roles. It also describes the network and storage architecture for OpenShift Container Platform. The hardware/software components required for building the OpenShift cluster is provided, in addition to the design details.

Technology Overview

This section provides a brief introduction of the various hardware/ software components used in this solution.

Cisco Unified Computing System

The Cisco Unified Computing System is a next-generation solution for blade and rack server computing. The system integrates a low-latency; lossless 10 Gigabit Ethernet unified network fabric with enterprise-class, x86-architecture servers. The system is an integrated, scalable, multi-chassis platform in which all resources participate in a unified management domain. The Cisco Unified Computing System accelerates the delivery of new services simply, reliably, and securely through end-to-end provisioning and migration support for both virtualized and non-virtualized systems. Cisco Unified Computing System provides:

- Comprehensive Management
- Radical Simplification
- High Performance

The Cisco Unified Computing System consists of the following components:

- Compute - The system is based on an entirely new class of computing system that incorporates rack mount and blade servers based on Intel® Xeon® scalable processors product family.
- Network - The system is integrated onto a low-latency, lossless, 40-Gbps unified network fabric. This network foundation consolidates Local Area Networks (LAN's), Storage Area Networks (SANs), and high-performance computing networks which are separate networks today. The unified fabric lowers costs by reducing the number of network adapters, switches, and cables, and by decreasing the power and cooling requirements.
- Virtualization - The system unleashes the full potential of virtualization by enhancing the scalability, performance, and operational control of virtual environments. Cisco security, policy enforcement, and diagnostic features are now extended into virtualized environments to better support changing business and IT requirements.
- Storage access - The system provides consolidated access to both SAN storage and Network Attached Storage (NAS) over the unified fabric. It is also an ideal system for Software defined Storage (SDS). Combining the benefits of single framework to manage both the compute and Storage servers in a single pane, Quality of Service (QOS) can be implemented if needed to inject IO throttling in the system. In addition, the server administrators can pre-assign storage-access policies to storage resources, for simplified storage connectivity and management leading to increased productivity. In addition to external storage, both rack and blade servers have internal storage which can be accessed through built-in hardware RAID controllers. With storage profile and disk configuration policy configured in Cisco UCS Manager, storage needs for the host OS and application data gets fulfilled by user defined RAID groups for high availability and better performance.

- Management - the system uniquely integrates all system components to enable the entire solution to be managed as a single entity by the Cisco UCS Manager. The Cisco UCS Manager has an intuitive graphical user interface (GUI), a command-line interface (CLI), and a powerful scripting library module for Microsoft PowerShell built on a robust application programming interface (API) to manage all system configuration and operations.

Cisco Unified Computing System (Cisco UCS) fuses access layer networking and servers. This high-performance, next-generation server system provides a data center with a high degree of workload agility and scalability.

Cisco UCS Manager

Cisco Unified Computing System (UCS) Manager provides unified, embedded management for all software and hardware components in the Cisco UCS. Using Single Connect technology, it manages, controls, and administers multiple chassis for thousands of virtual machines. Administrators use the software to manage the entire Cisco Unified Computing System as a single logical entity through an intuitive GUI, a command-line interface (CLI), or an XML API. The Cisco UCS Manager resides on a pair of Cisco UCS 6300 Series Fabric Interconnects using a clustered, active-standby configuration for high-availability.

UCS Manager offers unified embedded management interface that integrates server, network, and storage. UCS Manager performs auto-discovery to detect inventory, manage, and provision system components that are added or changed. It offers comprehensive set of XML API for third part integration, exposes 9000 points of integration and facilitates custom development for automation, orchestration, and to achieve new levels of system visibility and control.

Service profiles benefit both virtualized and non-virtualized environments and increase the mobility of non-virtualized servers, such as when moving workloads from server to server or taking a server offline for service or upgrade. Profiles can also be used in conjunction with virtualization clusters to bring new resources online easily, complementing existing virtual machine mobility.

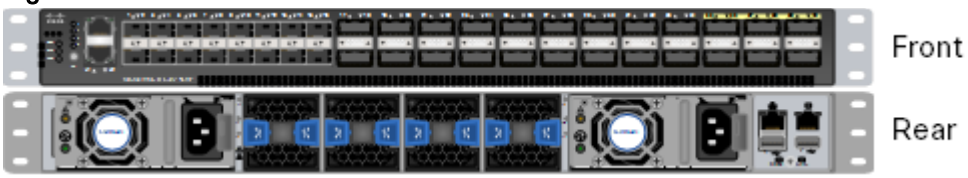
For more Cisco UCS Manager Information, refer to: http://www.cisco.com/c/en/us/products/servers-unified-computing/ucs-manager/index.html

Cisco UCS Fabric Interconnects

The Fabric interconnects provide a single point for connectivity and management for the entire system. Typically deployed as an active-active pair, the system's fabric interconnects integrate all components into a single, highly-available management domain controlled by Cisco UCS Manager. The fabric interconnects manage all I/O efficiently and securely at a single point, resulting in deterministic I/O latency regardless of a server or virtual machine's topological location in the system.

Cisco UCS 6300 Series Fabric Interconnects support the bandwidth up to 2.43-Tbps unified fabric with low-latency, lossless, cut-through switching that supports IP, storage, and management traffic using a single set of cables. The fabric interconnects feature virtual interfaces that terminate both physical and virtual connections equivalently, establishing a virtualization-aware environment in which blade, rack servers, and virtual machines are interconnected using the same mechanisms. The Cisco UCS 6332-16UP is a 1-RU Fabric Interconnect that features up to 40 universal ports that can support 24 40-Gigabit Ethernet, Fiber Channel over Ethernet, or native Fiber Channel connectivity. In addition to this it supports up to 16 1- and 10-Gbps FCoE or 4-, 8- and 16-Gbps Fibre Channel unified ports.

Figure 1 Cisco UCS Fabric Interconnect 6332-16UP



For more information, visit the following link: https://www.cisco.com/c/en/us/products/servers-unified-computing/ucs-6332-16up-fabric-interconnect/index.html

Cisco UCS 5108 Blade Server Chassis

The Cisco UCS 5100 Series Blade Server Chassis is a crucial building block of the Cisco Unified Computing System, delivering a scalable and flexible blade server chassis. The Cisco UCS 5108 Blade Server Chassis is six rack units (6RU) high and can mount in an industry-standard 19-inch rack. A single chassis can house up to eight half-width Cisco UCS B-Series Blade Servers and can accommodate both half-width and full-width blade form factors. Four single-phase, hot-swappable power supplies are accessible from the front of the chassis. These power supplies are 92 percent efficient and can be configured to support non-redundant, N+1 redundant and grid-redundant configurations. The rear of the chassis contains eight hot-swappable fans, four power connectors (one per power supply), and two I/O bays for Cisco UCS 2304 Fabric Extenders. A passive mid-plane provides multiple 40 Gigabit Ethernet connections between blade serves and fabric interconnects. The Cisco UCS 2304 Fabric Extender has four 40 Gigabit Ethernet, FCoE-capable, Quad Small Form-Factor Pluggable (QSFP+) ports that connect the blade chassis to the fabric interconnect. Each Cisco UCS 2304 can provide one 40 Gigabit Ethernet ports connected through the midplane to each half-width slot in the chassis, giving it a total eight 40G interfaces to the compute. Typically configured in pairs for redundancy, two fabric extenders provide up to 320 Gbps of I/O to the chassis.

For more information, please refer to the following link: http://www.cisco.com/c/en/us/products/servers-unified-computing/ucs-5100-series-blade-server-chassis/index.html

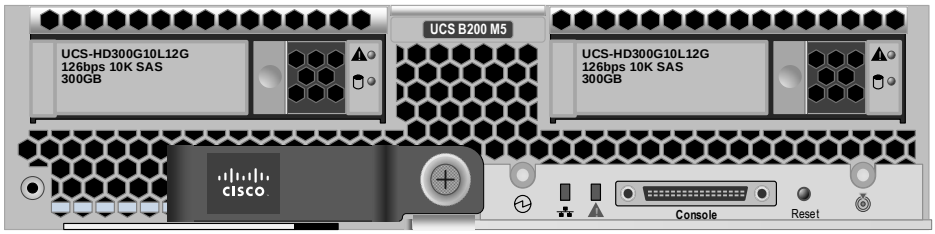
Cisco UCS B200 M5 Blade Server

The Cisco UCS B200 M5 Blade Server delivers performance, flexibility, and optimization for deployments in data centers, in the cloud, and at remote sites. This enterprise-class server offers market-leading performance, versatility, and density without compromise for workloads including Virtual Desktop Infrastructure (VDI), web infrastructure, distributed databases, converged infrastructure, and enterprise applications such as Oracle and SAP HANA. The B200 M5 server can quickly deploy stateless physical and virtual workloads through programmable, easy-to-use Cisco UCS Manager Software and simplified server access through Cisco SingleConnect technology. The Cisco UCS B200 M5 server is a half-width blade. Up to eight servers can reside in the 6-Rack-Unit (6RU) Cisco UCS 5108 Blade Server Chassis, offering one of the highest densities of servers per rack unit of blade chassis in the industry. You can configure the B200 M5 to meet your local storage requirements without having to buy, power, and cool components that you do not need. The B200 M5 provides you these main features:

- Up to two Intel Xeon Scalable CPUs with up to 28 cores per CPU
- 24 DIMM slots for industry-standard DDR4 memory at speeds up to 2666 MHz, with up to 3 TB of total memory when using 128-GB DIMMs

- Modular LAN On Motherboard (mLOM) card with Cisco UCS Virtual Interface Card (VIC) 1340, a 2-port, 40 Gigabit Ethernet, Fibre Channel over Ethernet (FCoE)–capable mLOM mezzanine adapter
- Optional rear mezzanine VIC with two 40-Gbps unified I/O ports or two sets of 4 x 10-Gbps unified I/O ports, delivering 80 Gbps to the server; adapts to either 10- or 40-Gbps fabric connections
- Two optional, hot-pluggable, Hard-Disk Drives (HDDs), Solid-State Disks (SSDs), or NVMe 2.5-inch drives with a choice of enterprise-class RAID or pass-through controllers

Figure 2 Cisco UCS B200 M5 Blade Server



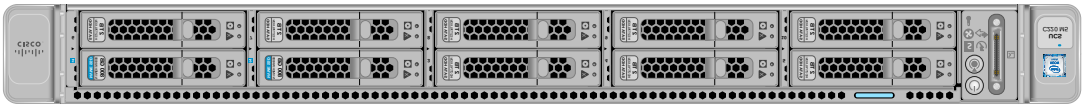
For more information, see: https://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/ucs-b-series-blade-servers/datasheet-c78-739296.html

Cisco UCS C220 M5 Rack-Mount Server

The Cisco UCS C220 M5 Rack Server is among the most versatile general-purpose enterprise infrastructure and application servers in the industry. It is a high-density 2-socket rack server that delivers industry-leading performance and efficiency for a wide range of workloads, including virtualization, collaboration, and bare metal applications. The Cisco UCS C-Series Rack Servers can be deployed as standalone servers or as part of the Cisco Unified Computing System™ (Cisco UCS) to take advantage of Cisco's standards-based unified computing innovations that help reduce customers' Total Cost of Ownership (TCO) and increase their business agility. The Cisco UCS C220 M5 server extends the capabilities of the Cisco UCS portfolio in a 1-Rack-Unit (1RU) form factor. It incorporates the Intel® Xeon® Scalable processors, supporting up to 20 percent more cores per socket, twice the memory capacity, 20 percent greater storage density, and five times more PCIe NVMe Solid-State Disks (SSDs) compared to the previous generation of servers. These improvements deliver significant performance and efficiency gains that will improve your application performance. The C220 M5 delivers outstanding levels of expandability and performance in a compact package, with:

- Latest Intel Xeon Scalable CPUs with up to 28 cores per socket
- Up to 24 DDR4 DIMMs for improved performance
- 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

Figure 3 Cisco UCS C220 M5SX



For more information, see: https://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/ucs-c-series-rack-servers/datasheet-c78-739281.html

Cisco UCS C240 M5 Rack-Mount Server

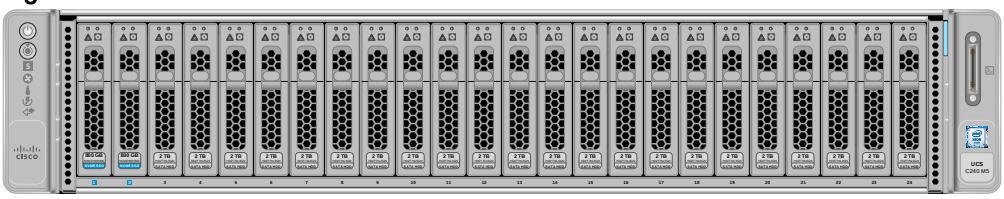
The Cisco UCS C240 M5 Rack Server is a 2-socket, 2-Rack-Unit (2RU) rack server offering industry-leading performance and expandability. It supports a wide range of storage and I/O-intensive infrastructure workloads, from big data and analytics to collaboration. Cisco UCS C-Series Rack Servers can be deployed as standalone servers or as part of a Cisco Unified Computing System™ (Cisco UCS) managed environment to take advantage of Cisco's standards-based unified computing innovations that help reduce customers' Total Cost of Ownership (TCO) and increase their business agility.

In response to ever-increasing computing and data-intensive real-time workloads, the enterprise-class Cisco UCS C240 M5 server extends the capabilities of the Cisco UCS portfolio in a 2RU form factor. It incorporates the Intel® Xeon® Scalable processors, supporting up to 20 percent more cores per socket, twice the memory capacity, and five times more

Non-Volatile Memory Express (NVMe) PCI Express (PCIe) Solid-State Disks (SSDs) compared to the previous generation of servers. These improvements deliver significant performance and efficiency gains that will improve your application performance. The C240 M5 delivers outstanding levels of storage expandability with exceptional performance, with:

- Latest Intel Xeon Scalable CPUs with up to 28 cores per socket
- Up to 24 DDR4 DIMMs for improved performance
- Up to 26 hot-swappable Small-Form-Factor (SFF) 2.5-inch drives, including 2 rear hot-swappable SFF drives (up to 10 support NVMe PCIe SSDs on the NVMe-optimized chassis version), or 12 Large-Form-Factor (LFF) 3.5-inch drives plus 2 rear hot-swappable SFF drives
- 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, supporting dual 10- or 40-Gbps network connectivity
- Dual embedded Intel x550 10GBASE-T LAN-On-Motherboard (LOM) ports
- Modular M.2 or Secure Digital (SD) cards that can be used for boot

Figure 4 Cisco UCS C240 M5SX



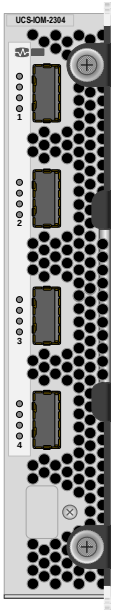
For more information, see: https://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/ucs-c-series-rack-servers/datasheet-c78-739279.html

Cisco UCS Fabric Extenders

Cisco UCS 2304 Fabric Extender brings the unified fabric into the blade server enclosure, providing multiple 40 Gigabit Ethernet connections between blade servers and the fabric interconnect, simplifying diagnostics, cabling, and management. It is a third-generation I/O Module (IOM) that shares the same form factor as the second-generation Cisco UCS 2200 Series Fabric Extenders and is backward compatible with the shipping Cisco UCS 5108 Blade Server Chassis. The Cisco UCS 2304 connects the I/O fabric between the Cisco UCS 6300 Series Fabric Interconnects and the Cisco UCS 5100 Series Blade Server Chassis, enabling a lossless and deterministic Fibre Channel over Ethernet (FCoE) fabric to connect all blades and chassis together. Because the fabric extender is similar to a distributed line card, it does not perform any switching and is managed as an extension of the fabric interconnects. This approach reduces the overall infrastructure complexity and enabling Cisco UCS to scale to many chassis without multiplying the number of switches needed, reducing TCO and allowing all chassis to be managed as a single, highly available management domain.

The Cisco UCS 2304 Fabric Extender has four 40Gigabit Ethernet, FCoE-capable, Quad Small Form-Factor Pluggable (QSFP+) ports that connect the blade chassis to the fabric interconnect. Each Cisco UCS 2304 can provide one 40 Gigabit Ethernet ports connected through the midplane to each half-width slot in the chassis, giving it a total eight 40G interfaces to the compute. Typically configured in pairs for redundancy, two fabric extenders provide up to 320 Gbps of I/O to the chassis.

Figure 5 Cisco UCS 2304 Fabric Extender



For more information, see: https://www.cisco.com/c/en/us/products/collateral/servers-unified-computing/ucs-6300-series-fabric-interconnects/datasheet-c78-675243.html

Cisco VIC Interface Cards

The Cisco UCS Virtual Interface Card (VIC) 1340 is a 2-port 40-Gbps Ethernet or dual 4 x 10-Gbps Ethernet, Fiber Channel over Ethernet (FCoE) capable modular LAN on motherboard (mLOM) designed exclusively for the M4 generation of Cisco UCS B-Series Blade Servers. All the blade servers for both Controllers and Computes will have MLOM VIC 1340 card. Each blade will have a capacity of 40Gb of network traffic. The underlying network interfaces like will share this MLOM card.

The Cisco UCS VIC 1340 enables a policy-based, stateless, agile server infrastructure that can present over 256 PCIe standards-compliant interfaces to the host that can be dynamically configured as either network interface cards (NICs) or host bus adapters (HBAs).

For more information, see: http://www.cisco.com/c/en/us/products/interfaces-modules/ucs-virtual-interface-card-1340/index.html

The Cisco UCS Virtual Interface Card 1385 improves flexibility, performance, and bandwidth for Cisco UCS C-Series Rack Servers. It offers dual-port Enhanced Quad Small Form-Factor Pluggable (QSFP+) 40 Gigabit Ethernet and Fibre Channel over Ethernet (FCoE) in a half-height PCI Express (PCIe) adapter. The 1385 card works with Cisco Nexus 40 Gigabit Ethernet (GE) and 10 GE switches for high-performance applications. The Cisco VIC 1385 implements the Cisco Data Center Virtual Machine Fabric Extender (VM-FEX), which unifies virtual and physical networking into a single infrastructure. The extender provides virtual-machine visibility from the physical network and a consistent network operations model for physical and virtual servers.

For more information, see: https://www.cisco.com/c/en/us/products/interfaces-modules/ucs-virtual-interface-card-1385/index.html

Cisco UCS Differentiators

Cisco's Unified Compute System is revolutionizing the way servers are managed in data-center. Following are the unique differentiators of UCS and UCS Manager:

1. Embedded Management –In UCS, the servers are managed by the embedded firmware in the Fabric Inter-connects, eliminating need for any external physical or virtual devices to manage the servers.
2. Unified Fabric –In UCS, from blade server chassis or rack servers to FI, there is a single Ethernet cable used for LAN, SAN and management traffic. This converged I/O results in reduced cables, SFPs and adapters which in turn reduce capital and operational expenses of the overall solution.
3. Auto Discovery –By simply inserting the blade server in the chassis or connecting rack server to the fabric interconnect, discovery and inventory of compute resource occurs automatically without any management intervention. The combination of unified fabric and auto-discovery enables the wire-once architecture of UCS, where compute capability of UCS can be extended easily while keeping the existing external connec-tivity to LAN, SAN and management networks.
4. Policy Based Resource Classification –Once a compute resource is discovered by UCS Manager, it can be automatically classified to a given resource pool based on policies defined. This capability is useful in multi-tenant cloud computing. This CVD showcases the policy based resource classification of UCS Manager.

5. Combined Rack and Blade Server Management –UCS Manager can manage B-Series blade servers and C-Series rack server under the same UCS domain. This feature, along with stateless computing makes compute resources truly hardware form factor agnostic.
6. Model based Management Architecture –UCS Manager Architecture and management database is model based and data driven. An open XML API is provided to operate on the management model. This enables easy and scalable integration of UCS Manager with other management systems.
7. Policies, Pools, Templates –The management approach in UCS Manager is based on defining policies, pools and templates, instead of cluttered configuration, which enables a simple, loosely coupled, data driven approach in managing compute, network and storage resources.
8. Loose Referential Integrity –In UCS Manager, a service profile, port profile or policies can refer to other policies or logical resources with loose referential integrity. A referred policy cannot exist at the time of authoring the referring policy or a referred policy can be deleted even though other policies are referring to it. This provides different subject matter experts to work independently from each-other. This provides great flexibility where different experts from different domains, such as network, storage, security, server and virtualization work together to accomplish a complex task.
9. Policy Resolution –In UCS Manager, a tree structure of organizational unit hierarchy can be created that mimics the real-life tenants and/or organization relationships. Various policies, pools and templates can be defined at different levels of organization hierarchy. A policy referring to another policy by name is resolved in the organization hierarchy with closest policy match. If no policy with specific name is found in the hierarchy of the root organization, then special policy named "default" is searched. This policy resolution practice enables automation friendly management APIs and provides great flexibility to owners of different organizations.
10. Service Profiles and Stateless Computing –a service profile is a logical representation of a server, carrying its various identities and policies. This logical server can be assigned to any physical compute resource as far as it meets the resource requirements. Stateless computing enables procurement of a server within minutes, which used to take days in legacy server management systems.
11. Built-in Multi-Tenancy Support –The combination of policies, pools and templates, loose referential integrity, policy resolution in organization hierarchy and a service profiles based approach to compute resources makes UCS Manager inherently friendly to multi-tenant environment typically observed in private and public clouds.
12. Extended Memory – the enterprise-class Cisco UCS B200 M5 blade server extends the capabilities of Cisco's Unified Computing System portfolio in a half-width blade form factor. The Cisco UCS B200 M5 harnesses the power of the latest Intel® Xeon® scalable processors product family CPUs with up to 3 TB of RAM– allowing huge VM to physical server ratio required in many deployments, or allowing large memory operations required by certain architectures like Big-Data.
13. Virtualization Aware Network –VM-FEX technology makes the access network layer aware about host virtualization. This prevents domain pollution of compute and network domains with virtualization when virtual network is managed by port-profiles defined by the network administrators' team. VM-FEX also off-loads hypervisor CPU by performing switching in the hardware, thus allowing hypervisor CPU to do more virtualization related tasks. VM-FEX technology is well integrated with VMware vCenter, Linux KVM and Hyper-V SR-IOV to simplify cloud management.
14. Simplified QoS –Even though Fiber Channel and Ethernet are converged in UCS fabric, built-in support for QoS and lossless Ethernet makes it seamless. Network Quality of Service (QoS) is simplified in UCS Manager by representing all system classes in one GUI panel.

Cisco Nexus 9000 Switches

The Cisco Nexus 9000 Series delivers proven high performance and density, low latency, and exceptional power efficiency in a broad range of compact form factors. Operating in Cisco NX-OS Software mode or in Application Centric Infrastructure (ACI) mode, these switches are ideal for traditional or fully automated data center deployments.

The Cisco Nexus 9000 Series Switches offer both modular and fixed 10/40/100 Gigabit Ethernet switch configurations with scalability up to 30 Tbps of non-blocking performance with less than five-microsecond latency, 1152 x 10 Gbps or 288 x 40 Gbps non-blocking Layer 2 and Layer 3 Ethernet ports and wire speed VXLAN gateway, bridging, and routing.

Figure 6 Cisco UCS Nexus 9396PX

For more information, see: https://www.cisco.com/c/en/us/products/collateral/switches/nexus-9000-series-switches/datasheet-c78-736967.html

Intel Scalable Processor Family

Intel® Xeon® Scalable processors provide a new foundation for secure, agile, multi-cloud data centers. This platform provides businesses with breakthrough performance to handle system demands ranging from entry-level cloud servers to compute-hungry tasks including real-time analytics, virtualized infrastructure, and high performance computing. This processor family includes technologies for accelerating and securing specific workloads.

- Intel® Xeon® Scalable processors are now available in four feature configurations:
- Intel® Xeon® Bronze Processors with affordable performance for small business and basic storage.
- Intel® Xeon® Silver Processors with essential performance and power efficiency
- Intel® Xeon® Gold Processors with workload-optimized performance, advanced reliability
- Intel® Xeon® Platinum Processors for demanding, mission-critical AI, analytics, hybrid-cloud workloads

Figure 7 Intel® Xeon® Scalable Processor Family

Intel® SSD DC S4500 Series

Intel® SSD DC S4500 Series is a storage inspired SATA SSD optimized for read-intensive workloads. Based on TLC Intel® 3D NAND Technology, these larger capacity SSDs enable data centers to increase data stored per rack unit. Intel® SSD DC S4500 Series is built for compatibility in legacy infrastructures so it enables easy storage upgrades that minimize the costs associated with modernizing data center. This 2.5” 7mm form factor offers wide range of capacity from 240 GB up to 3.8 TB.

Figure 8 Intel® SSD DC S4500

Red Hat OpenShift Architecture

The Red Hat OpenShift cluster platform is managed by the Kubernetes container orchestrator, which manages containerized applications across a cluster of systems running the Docker container runtime. The physical configuration of the Red Hat OpenShift Container Platform is based on the Kubernetes cluster architecture. This Red Hat OpenShift RA contains five types of nodes: bastion, master, infrastructure, storage, and application.

- Bastion Node:

  This is a dedicated node that serves as the main deployment and management server for the Red Hat OpenShift cluster. It is used as the logon node for the cluster administrators to perform the system deployment and management operations, such as running the Ansible® OpenShift deployment playbooks. The bastion node runs Red Hat Enterprise Linux 7.5.

- OpenShift Master Nodes:

  The OpenShift Container Platform master is a server that performs control functions for the whole cluster environment. It is responsible for the creation, scheduling, and management of all objects specific to Red Hat OpenShift. It includes API, controller manager, and scheduler capabilities in one OpenShift binary. It is also a common practice to install an etcd key-value store on OpenShift masters to achieve a low-latency link between etcd and OpenShift masters. It is recommended that you run both Red Hat OpenShift masters and etcd in highly available environments. This can be achieved by running multiple OpenShift masters in conjunction with an external active-passive load balancer and the clustering functions of etcd. The OpenShift master node runs Red Hat Enterprise Linux Atomic Host 7.5.

- OpenShift Infrastructure Nodes:

  The OpenShift infrastructure node runs infrastructure specific services: Docker Registry*, HAProxy router, and Heketi. Docker Registry stores application images in the form of containers. The HAProxy router provides routing functions for Red Hat OpenShift applications. It currently supports HTTP(S) traffic and TLS-enabled traffic via Server Name Indication (SNI). Heketi provides management API for configuring Container-native persistent storage. Additional applications and services can be deployed on OpenShift infrastructure nodes. The OpenShift infrastructure node runs Red Hat Enterprise Linux Atomic Host 7.5.

- OpenShift Application Nodes:

  The OpenShift application nodes run containerized applications created and deployed by developers. An OpenShift application node contains the OpenShift node components combined into a single binary, which can be used by OpenShift masters to schedule and control containers. A Red Hat OpenShift application node runs Red Hat Enterprise Linux Atomic Host 7.5.

- OpenShift Storage Nodes:

  The OpenShift storage nodes run CNS services, which configure persistent volumes for application containers that require data persistence. Persistent volumes may be created manually by a cluster administrator or automatically by storage class objects. An OpenShift storage node is also capable of running containerized applications. A Red Hat OpenShift storage node runs Red Hat Enterprise Linux Atomic Host 7.5.

The following software components are used in designing this CVD.

Red Hat Enterprise Linux Atomic Host

Red Hat Enterprise Linux Atomic Host is a lightweight variant of the Red Hat Enterprise Linux operating system designed to run Linux containers. By combining the modular capabilities of Linux containers and Red Hat Enterprise Linux, containers can be more securely and effectively deployed and managed across public, private, and hybrid cloud infrastructures.

Red Hat OpenShift Container Platform

The Red Hat OpenShift Container Platform is a complete container application platform that includes the application development process in one consistent solution across multiple infrastructure footprints. Red Hat OpenShift integrates the architecture, processes, platforms, and services needed to help development and

operations teams traverse traditional siloed structures and produce applications that help businesses succeed.

Container-native Storage Solution from Red Hat

Container-native storage (CNS) solution from Red Hat makes OpenShift Container Platform a fully hyperconverged infrastructure where storage containers co-reside with the compute containers. Storage plane is based on containerized Red Hat Gluster® Storage services, which controls storage devices on every storage server. Heketi is a part of the container-native storage architecture and controls all of the nodes that are members of storage cluster. Heketi also provides an API through which storage space for containers can be easily requested. While Heketi provides an endpoint for storage cluster, the object that makes calls to its API from OpenShift clients is called a Storage Class. It is a Kubernetes and OpenShift object that describes the type of storage available for the cluster and can dynamically send storage requests when a persistent volume claim is generated.

Docker

Red Hat OpenShift Container Platform uses Docker runtime engine for containers.

Kubernetes

Red Hat OpenShift Container Platform is a complete container application platform that natively integrates technologies like Docker and Kubernetes - a powerful container cluster management and orchestration system. In our solution we have used Kubernetes to orchestrate and manage containerized applications.

Etcd

Etcd is a key-value store used in OpenShift Container Platform cluster. Etcd data store provides complete cluster and endpoint states to the OpenShift API servers (part of the master binary). Etcd data store furnishes information to API servers about node status, network configurations, secrets etc.

Open vSwitch

Open vSwitch is an open-source implementation of a distributed virtual multilayer switch. It is designed to enable effective network automation through programmatic extensions, while supporting standard management interfaces and protocols such as 802.1ag, SPAN, LACP and NetFlow. Open vSwitch provides software-defined networking (SDN)-specific functions in the OpenShift Container Platform environment.

HAProxy

HAProxy is an open source software that provides a high availability load balancer and proxy server for TCP and HTTP-based applications that spreads requests across multiple servers. In this solution, HAProxy provides routing and load-balancing functions for Red Hat OpenShift applications.

Keepalived

Keepalived is routing software which provides simple and robust facilities for load balancing and high-availability to Linux based infrastructure. In this solution, keepalived is used to provide virtual IP management for HAProxy instances to ensure highly available OpenShift Container Platform cluster.

Red Hat Ansible Automation

Red Hat Ansible Automation is a powerful IT automation tool. It is capable of provisioning numerous types of resources and deploying applications. It can configure and manage devices and operating system components. Due to the simplicity, extensibility, and portability, this OpenShift solution is based largely on Ansible Playbooks.

Ansible is mainly used for installation and management of the OpenShift Container Platform deployment.

Solution Design

This section provides an overview of the hardware and software components used in this solution, as well as the design factors to be considered in order to make the system work as a single, highly available solution.

Red Hat OpenShift System Architectural Overview

Red Hat OpenShift Infrastructure Components

OpenShift Container Platform is a Platform-as-a-Service (PaaS) offering from Red Hat that brings together Docker and Kubernetes, and provides an API to manage these services. OpenShift Container Platform allows you to create and manage containers. Containers are standalone processes that run within their own environment, independent of operating system and the underlying infrastructure. OpenShift 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. OpenShift Container Platform has a microservices-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. Those services are broken down by function:

- REST APIs, which expose each of the core objects.
- Controllers, which read those APIs, apply changes to other objects, and report status or write back to the object.

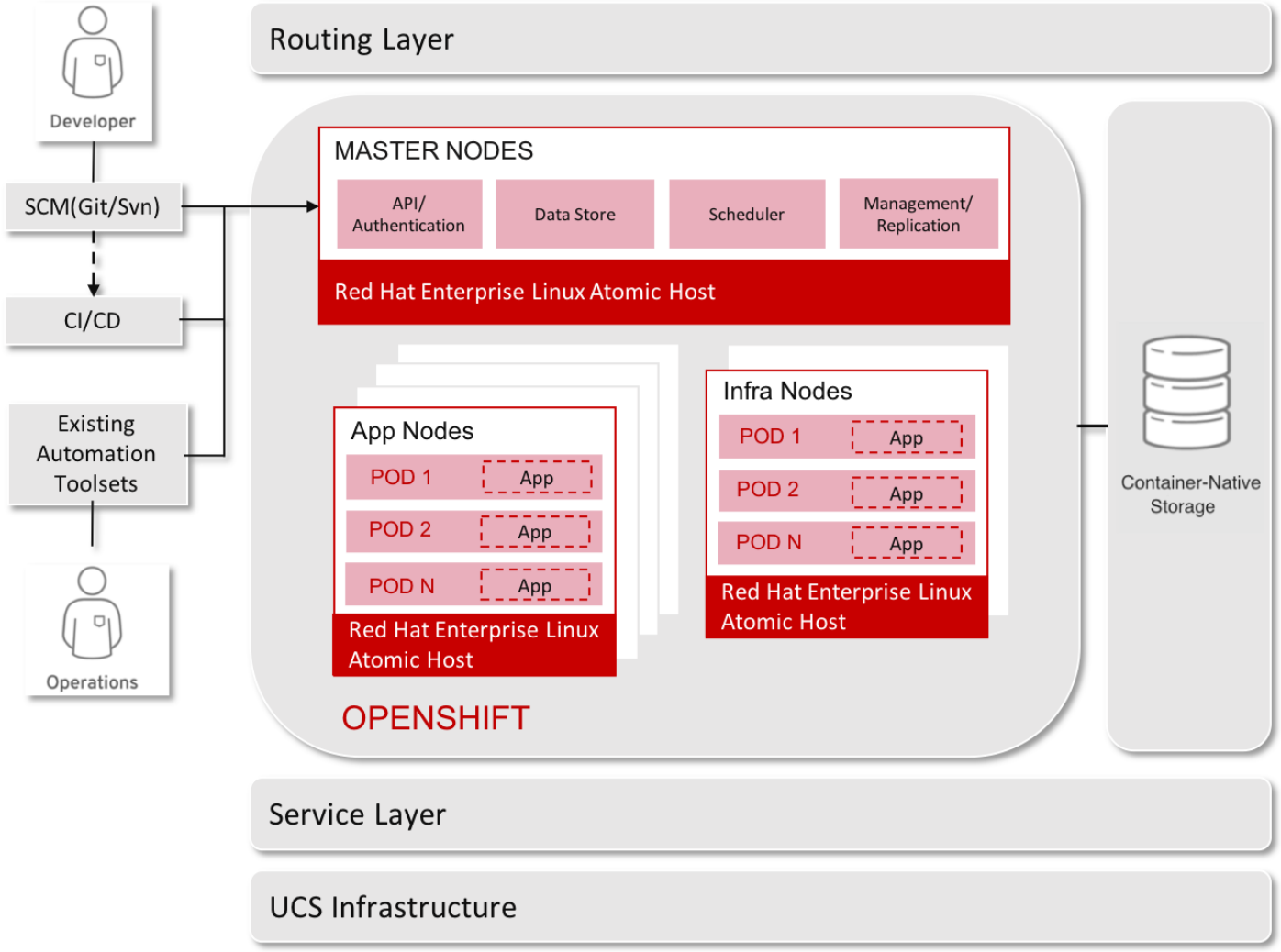
OpenShift is a layered system designed to expose underlying Docker-formatted container image and Kubernetes concepts as accurately as possible, with a focus on easy composition of applications by a developer. For example, install Ruby, push code, and add MySQL. The concept of an application as a separate object is removed in favor of more flexible composition of "services", allowing two web containers to reuse a database or expose a database directly to the edge of the network.

The Docker service provides the abstraction for packaging and creating Linux-based, lightweight container images. Kubernetes provides the cluster management and orchestrates containers on multiple hosts.

OpenShift Container Platform adds:

- Source code management, build, and deployment for developers
- Managing and promoting images at scale as they flow through the system
- Application management at scale
- Team and user tracking for organizing a large developer organization
- Networking infrastructure that supports the cluster

Figure 9 Red Hat OpenShift Container Platform Architecture Overview



Kubernetes Infrastructure

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

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

High Availability Masters

Masters are the hosts that contain the master components, including the API server, controller manager server, and etcd - the kv state data store. The master manages nodes in the Kubernetes cluster and schedules the pods (single/group of containers) to run on the application nodes.

Table 1 Master Components

Components	Description

API Server	The Kubernetes API server validates and configures the data for pods, services, and replication controllers. It also assigns pods to nodes and synchronizes pod information with service configuration. Can be run as a standalone process.
etcd	etcd stores the persistent master state while other components watch etcd for changes to bring themselves into the desired state. etcd can be optionally configured for high availability, typically deployed with 2n+1 peer services.
Controller Manager Server	The controller manager server watches etcd for changes to replication controller objects and then uses the API to enforce the desired state. Can be run as a standalone process. Several such processes create a cluster with one active leader at a time.
HAProxy	Optional, used when configuring highly-available masters with the native method to balance load between API master endpoints. The advanced installation method can configure HAProxy for you with the native method. Alternatively, you can use the native method but pre-configure your own load balancer of choice.

While in a single master configuration, the availability of running applications remains if the master or any of its services fail. However, failure of master services reduces the ability of the system to respond to application failures or creation of new applications.

To mitigate concerns about availability of the master, the solution design uses a high-availability solution to configure masters and ensure that the cluster has no single point of failure. When using the native HA method with HAProxy, master components have the following availability:

Table 2 Availability Matrix with HAProxy

Roles	Style	Description
etcd	Active-active	Fully redundant deployment with load balancing
API Server	Active-active	Managed by HAProxy
Controller Manager Server	Active-passive	One instance is elected as a cluster leader at a time
HAProxy	Active-passive	Balances load between API master endpoints

Application Nodes

An Application node provides the runtime environments for containers. Each node in a Kubernetes cluster has the required services to be managed by the master. Nodes also have the required services to run pods, including the Docker service, a kubelet, and a service proxy.

Kubelet

Each node has a kubelet that updates the node as specified by a container manifest, which is a YAML file that describes a pod. The kubelet uses a set of manifests to ensure that its containers are started and that they continue to run.

Service Proxy

Each node also runs a simple network proxy that reflects the services defined in the API on that node. This allows the node to do simple TCP and UDP stream forwarding across a set of back ends.

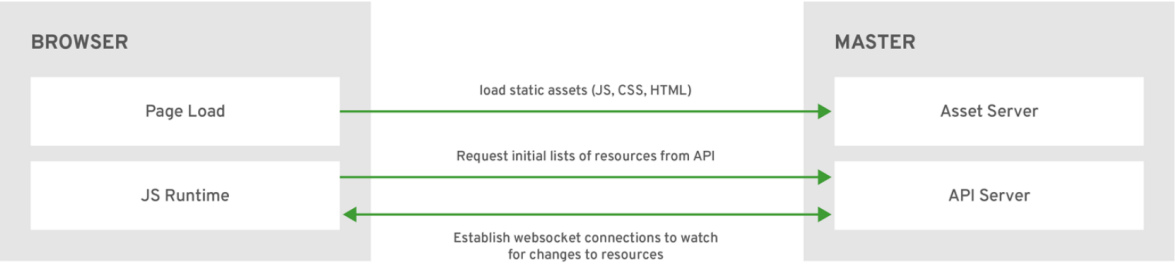
Web Console

The OpenShift Container Platform web console is a user interface accessible from a web browser. Developers can use the web console to visualize, browse, and manage the contents of projects.

The web console is started as part of the master. All static assets required to run the web console are served from the openshift binary. Administrators can also customize the web console using extensions, which let you run scripts and load custom stylesheets when the web console loads.

On accessing the web console from a browser, it first loads all required static assets. It then makes requests to the OpenShift Container Platform APIs using the values defined from the openshift start option --public-master, or from the related master configuration file parameter masterPublicURL. The web console uses WebSockets to maintain a persistent connection with the API server and receive updated information as soon as it is available.

Figure 10 Web Console Request Architecture



Cluster Node Types

Below table shows functions and roles each class of node in perform in the reference design of OpenShift Container Platform cluster –

Table 3 Type of nodes in Red Hat OpenShift Container Platform cluster and their roles

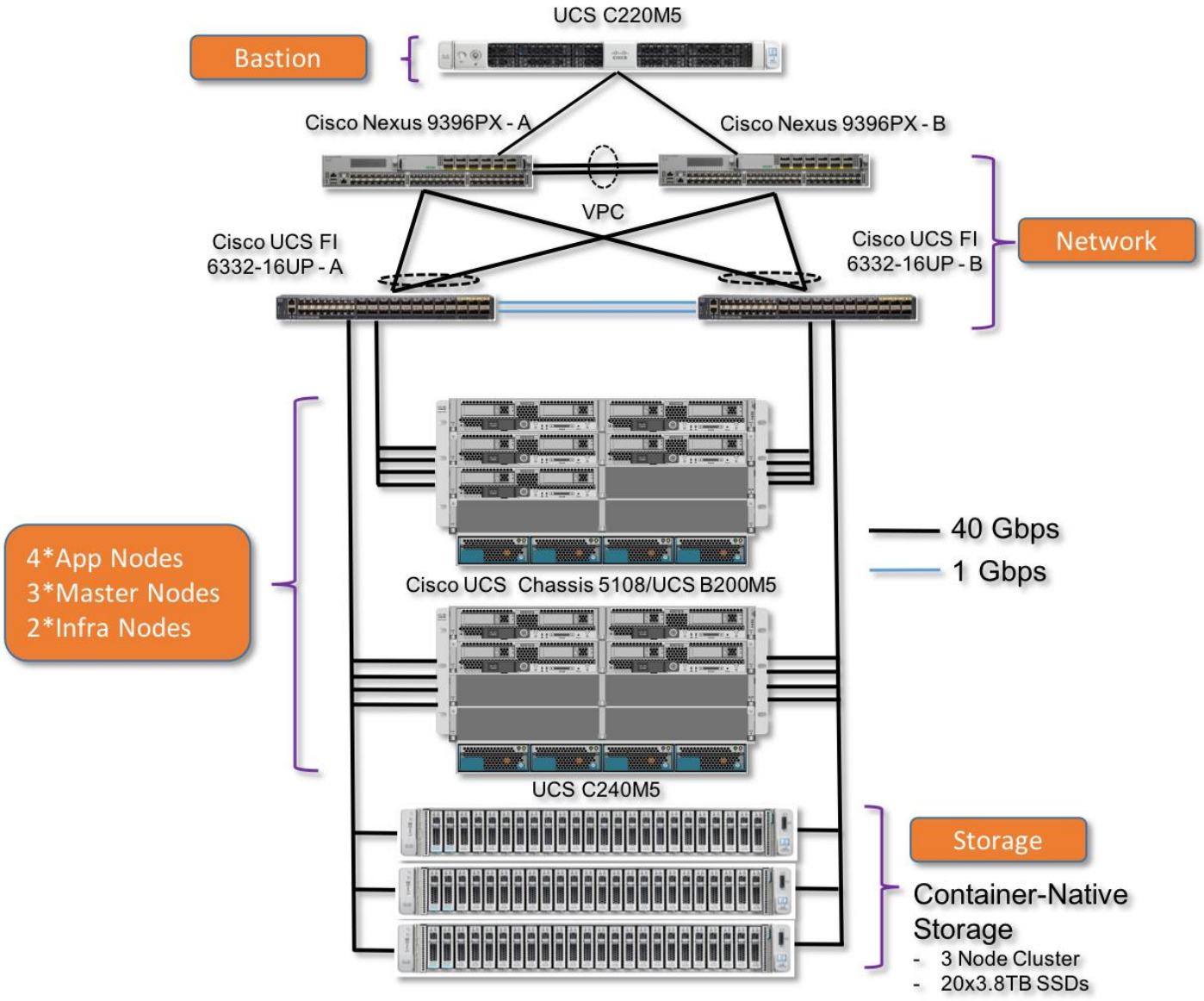
Node	Roles
Bastion Node	- System deployment and Management Operations - Runs DNS services for the Red Hat OpenShift Container Platform cluster
Master Nodes	- Kubernetes services - Etcd data store - Controller Manager & Scheduler
Infrastructure Nodes	- Container Registry - HA Proxy - Heketi - KeepAlived
Application Nodes	- Application Containers PODs - Docker Runtime
Storage Nodes	- Red Hat Gluster Storage - Container-native storage services

Red Hat OpenShift on UCS Reference Architecture - Physical Topology

Physical Topology Reference Architecture Production Use Case

The Physical Topology for Architecture – I: High-End Production use case, includes 9x Cisco UCS B200M5s, 1xCisco UCS C220M5 and 3x Cisco UCS C240M5 blade and rack servers, UCS Fabric Interconnects 6332-16UP and Cisco Nexus 9396PX switches. All the nodes are provisioned with SSDs as storage devices including CNS nodes for persistent storage provisioning.

Figure 11 Reference Architecture – Production Use Case

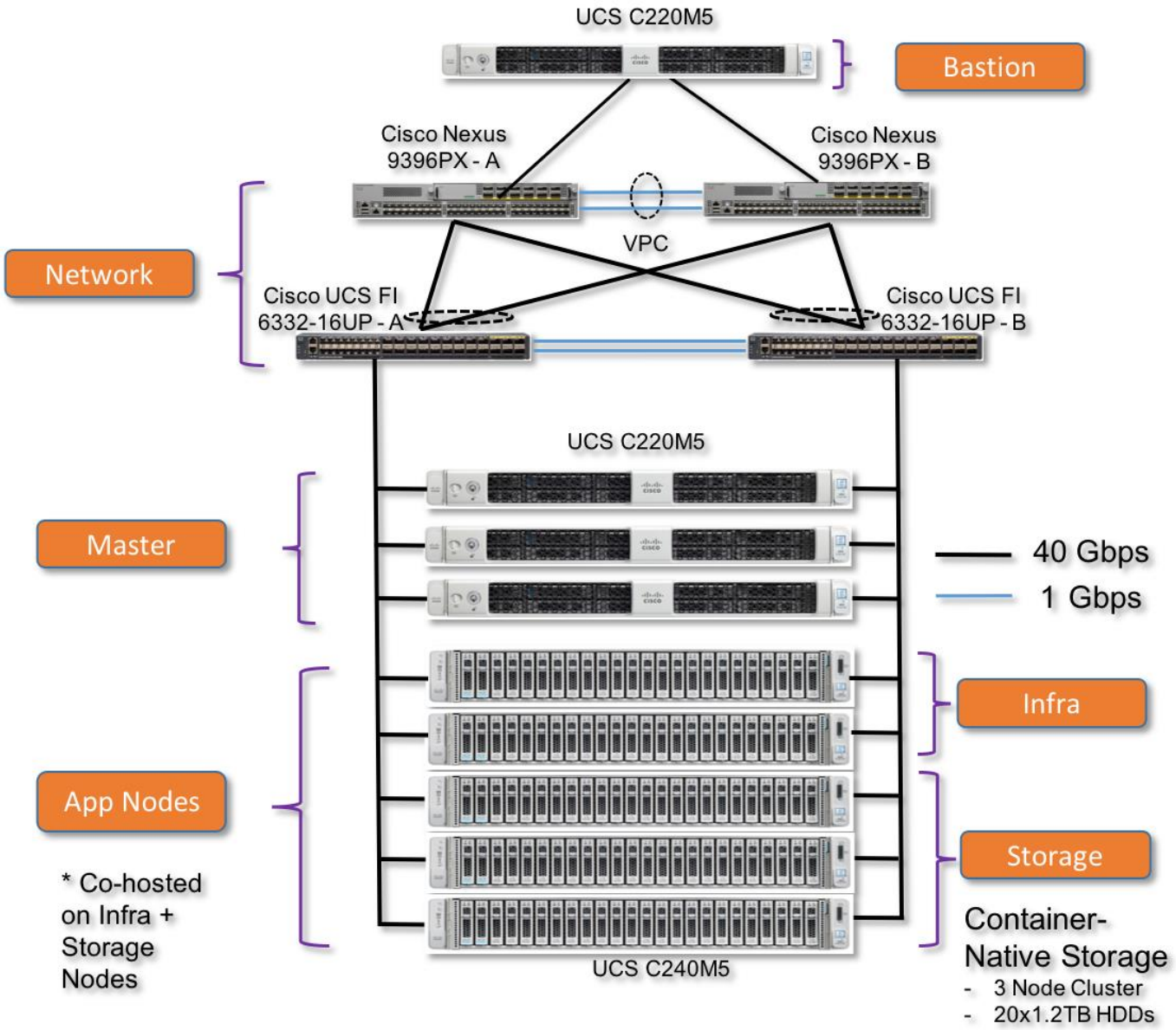


Physical Topology Reference Architecture Dev/Test Use Case

The Physical Topology for Architecture – II: Starter Dev/Test use case, includes 5xCisco UCS C240M5 and 4xCisco UCS C220M5 rack servers, UCS Fabric Interconnects 6332-16UP and Cisco Nexus 9396PX

switches. All the nodes are provisioned with HDDs as storage devices including storage nodes for CNS. App nodes are co-located with 2-node Infra as well 3-node storage cluster on Cisco UCS C240M5 servers.

Figure 12 Reference Architecture – Dev/Test Use Case

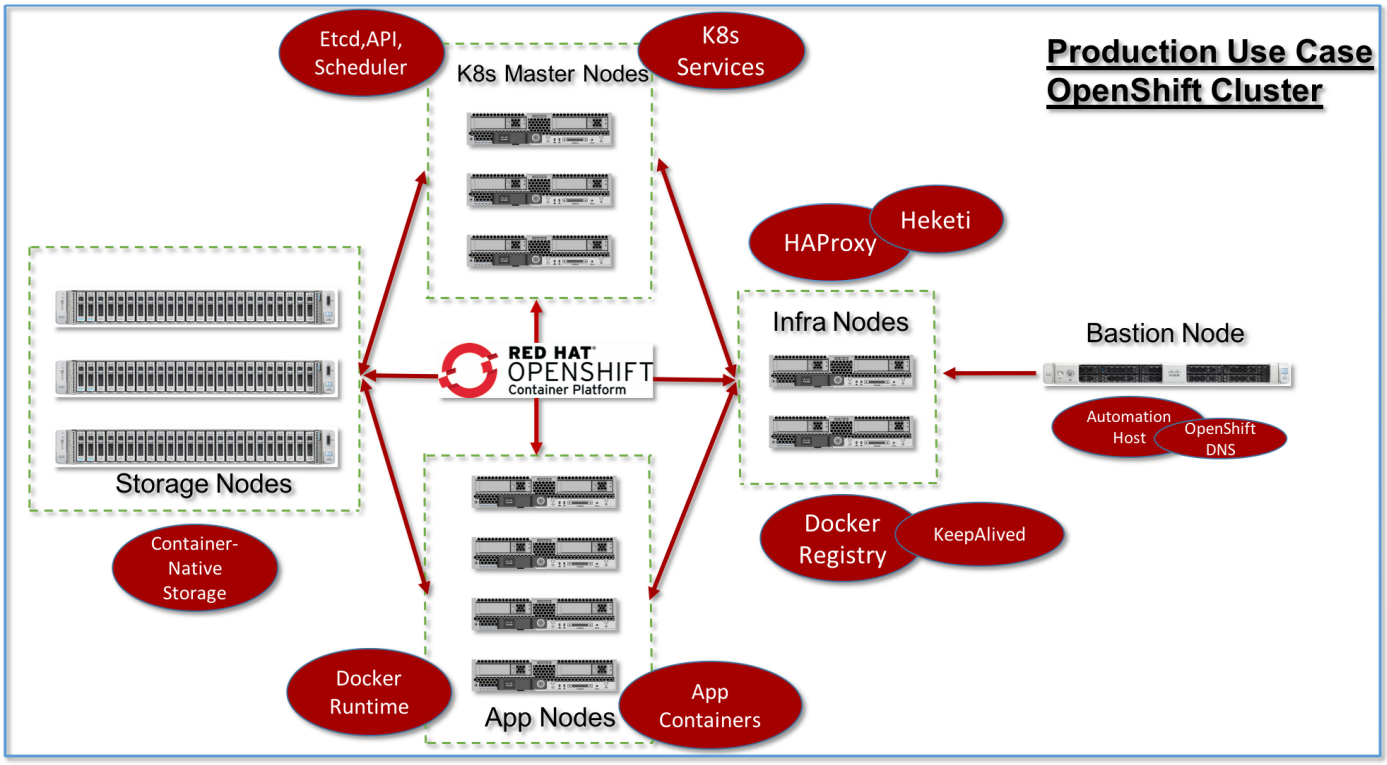


Red Hat OpenShift on UCS Reference Architecture - Logical Topology

Logical Topology Reference Architecture Production Use Case

The cluster node roles and various services they provide are shown in the diagram below.

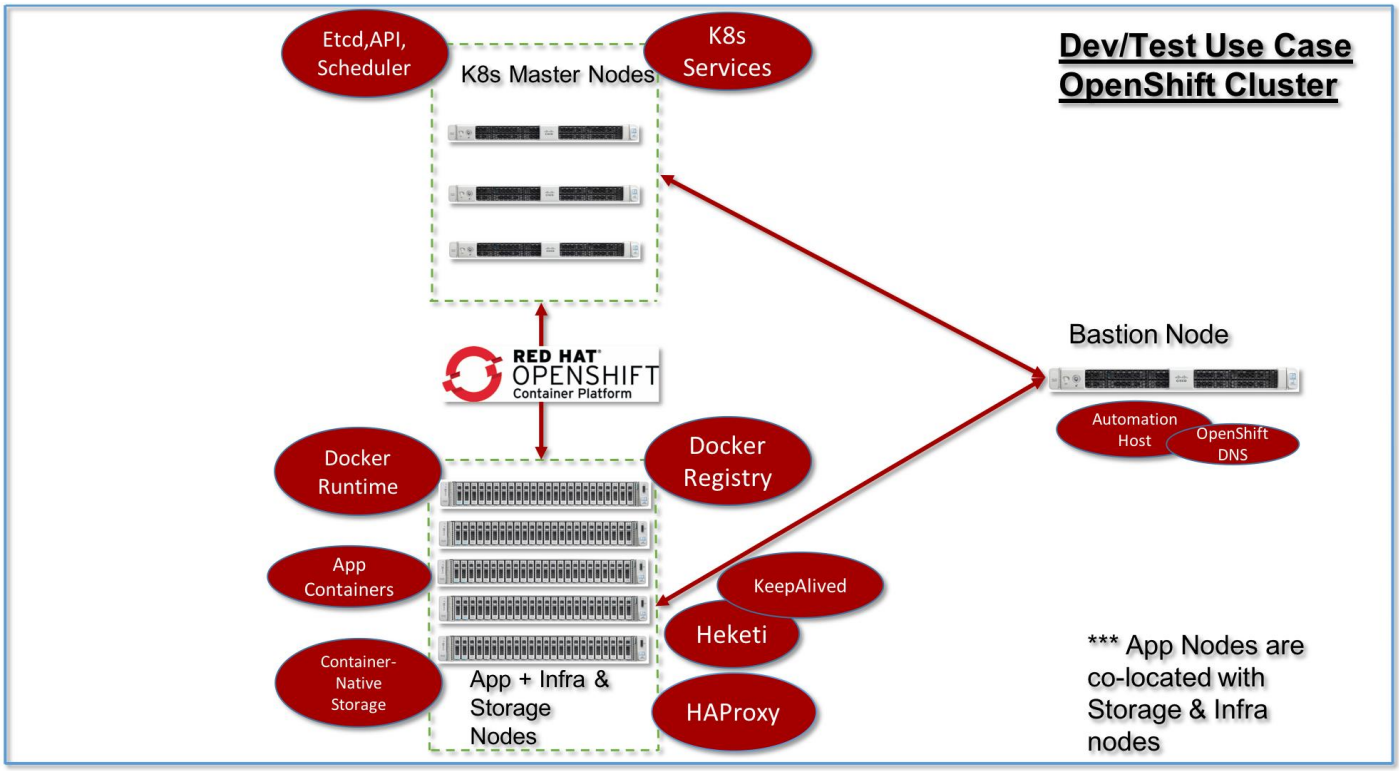
Figure 13 Logical Reference Architecture for Production Use Case



Logical Topology Reference Architecture Dev/Test Use Case

In this architecture option, 2-nodes are dedicated for Infra services and 2-nodes for CNS services. In this architecture Infra services and functions are co-hosted on 2 of the Application nodes and CNS services are co-hosted on the rest of the 3 Application nodes. Master and Bastion nodes remain dedicated for the services and roles they perform in the cluster.

Figure 14 Logical Reference Architecture for Dev/Test Use Case



Networking Overview

Network Architecture

OpenShift Container Platform supports the Kubernetes Container Network Interface (CNI) as the interface between the OpenShift Container Platform and Kubernetes. Software defined network (SDN) plug-ins are a powerful and flexible way to match network capabilities for networking needs.

The solution design relies on Kubernetes to ensure the pods are able to network with each other, and get an IP address from an internal network for each pod. This ensures all containers within the pod behave as if they were on the same host. Giving each pod its own IP address means that pods can be treated like physical hosts or virtual machines in terms of port allocation, networking, naming, service discovery, load balancing, application configuration, and migration. Creating links between pods is not necessary, and it is not recommended as pods communicate with each other directly using the IP address. In order to interact with the pods, services are created and using services, communication patterns between the pods can be defined.

OpenShift Container Platform DNS

OpenShift Container Platform has a built-in DNS so the services can be reached by the service DNS as well as the service IP/port. OpenShift Container Platform supports split DNS by running SkyDNS on the master that answers DNS queries for services. The master listens to port 53 by default. DNS services are required for the following use cases:

- Multiple services such as frontend and backend services running multiple pods, where frontend pods need to communicate with backend pods
- Services are deleted and re-created again, new set of IP address can be assigned but service names should remain un-changed, in order to communicate between the service types, without having a need to change the code. Inter service communication can take place with names rather than ip addresses

OpenShift SDN

OpenShift Container Platform deploys a software-defined networking (SDN) approach for connecting pods in an OpenShift Container Platform cluster. The OpenShift SDN connects all pods across all node hosts, providing a unified cluster network. OpenShift SDN is installed and configured by default as part of the Ansible-based installation procedure.

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).

OpenShift SDN provides three SDN plug-ins for configuring the pod network, such as:

- The ovs-subnet plug-in is the original plug-in: which provides a "flat" pod network where every pod can communicate with every other pod and service.
- The ovs-multitenant plug-in: project-level isolation for pods and services. Each project receives a unique Virtual Network ID (VNID) that identifies traffic from pods assigned to the project. Pods from different projects cannot send packets to or receive packets from pods and services of a different project. However, projects that receive VNID 0 are more privileged in that they are allowed to communicate with all other pods, and all other pods can communicate with them. In OpenShift Container Platform clusters, the default project has VNID 0. This facilitates certain services, such as the load balancer, to communicate with all other pods in the cluster and vice versa.
- The ovs-networkpolicy plug-in: allows project administrators to configure their own isolation policies using NetworkPolicy objects.

 This CVD uses the first two SDN plug-ins: ovs-subnet and ovs-multitenant

OpenShift SDN – Functions at Masters

On an OpenShift Container Platform master, OpenShift SDN maintains a registry of nodes, stored in etcd. When the system administrator registers a node, OpenShift SDN allocates an unused subnet from the cluster network and stores this subnet in the registry. When a node is deleted, OpenShift SDN deletes the subnet from the registry and considers the subnet available to be allocated again.

In the default configuration, the cluster network is the 10.128.0.0/14 network (i.e. 10.128.0.0 - 10.131.255.255), and nodes are allocated /23 subnets (i.e., 10.128.0.0/23, 10.128.2.0/23, 10.128.4.0/23, and so on). This means that the cluster network has 512 subnets available to assign to nodes, and a given node is allocated 510 addresses that it can assign to the containers running on it. The size and address range of the cluster network are configurable, as is the host subnet size.

Note that the OpenShift SDN on a master does not configure the local (master) host to have access to any cluster network. Consequently, a master host does not have access to pods via the cluster network, unless it is also running as a node.

When using the ovs-multitenant plug-in, the OpenShift SDN master also watches for the creation and deletion of projects, and assigns VXLAN VNIDs to them, which are later used by the nodes to isolate traffic correctly.

OpenShift SDN – Functions at Nodes

On a node, OpenShift SDN first registers the local host with the SDN master in the aforementioned registry so that the master allocates a subnet to the node.

Next, OpenShift SDN creates and configures three network devices:

- `br0`: the OVS bridge device that pod containers will be attached to. OpenShift SDN also configures a set of non-subnet-specific flow rules on this bridge.
- `tun0`: an OVS internal port (port 2 on `br0`). This gets assigned the cluster subnet gateway address, and is used for external network access. OpenShift SDN configures netfilter and routing rules to enable access from the cluster subnet to the external network via NAT.
- `vxlan_sys_4789`: The OVS VXLAN device (port 1 on `br0`), which provides access to containers on remote nodes. Referred to as `vxlan0` in the OVS rules.

Each time a pod is started on the host, OpenShift SDN:

- Assigns the pod a free IP address from the node's cluster subnet.
- Attaches the host side of the pod's veth interface pair to the OVS bridge `br0`.
- Adds OpenFlow rules to the OVS database to route traffic addressed to the new pod to the correct OVS port.
- In the case of the ovs-multitenant plug-in, adds OpenFlow rules to tag traffic coming from the pod with the pod's VNID, and to allow traffic into the pod if the traffic's VNID matches the pod's VNID (or is the privileged VNID 0). Non-matching traffic is filtered out by a generic rule.

OpenShift SDN nodes also watch for subnet updates from the SDN master. When a new subnet is added, the node adds OpenFlow rules on `br0` so that packets with a destination IP address the remote subnet go to `vxlan0` (port 1 on `br0`) and thus out onto the network. The ovs-subnet plug-in sends all packets across the VXLAN with VNID 0, but the ovs-multitenant plug-in uses the appropriate VNID for the source container.

Network Isolation

The ovs-multitenant plug-in provides network isolation. When a packet exits a pod assigned to a non-default project, the OVS bridge br0 tags that packet with the project's assigned VNID. If the packet is directed to another IP address in the node's cluster subnet, the OVS bridge only allows the packet to be delivered to the destination pod if the VNIDs match.

If a packet is received from another node via the VXLAN tunnel, the Tunnel ID is used as the VNID, and the OVS bridge only allows the packet to be delivered to a local pod if the tunnel ID matches the destination pod's VNID.

Packets destined for other cluster subnets are tagged with their VNID and delivered to the VXLAN tunnel with a tunnel destination address of the node owning the cluster subnet.

HAProxy Router Plug-in

A router is one way to get traffic into the cluster. The HAProxy Template Router plug-in is one of the available router plugins. The template router has two components:

- A wrapper that watches endpoints and routes and causes a HAProxy reload based on changes.
- A controller that builds the HAProxy configuration file bases on routes and endpoints.

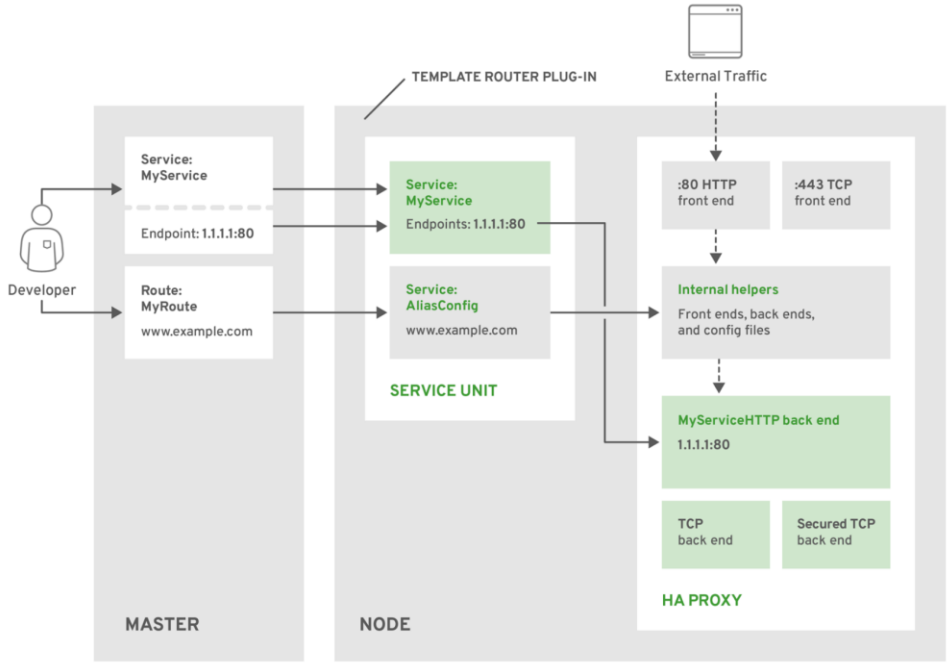
The controller and HAProxy are housed inside a pod, which is managed by a deployment configuration. The process of setting up the router is automated by the `oc adm router` command.

The controller watches the routes and endpoints for changes, as well as HAProxy's health. When a change is detected, it builds a new haproxy-config file and restarts HAProxy. The haproxy-config file is constructed based on the router's template file and information from OpenShift Container Platform.

The HAProxy template file can be customized as needed to support features that are not currently supported by OpenShift Container Platform. The HAProxy manual describes all of the features supported by HAProxy.

The following diagram illustrates how data flows from the master through the plug-in and finally into an HAProxy configuration:

Figure 15 HAProxy Router Data Flow



Physical Network Connectivity

Following figures illustrates cabling details for the two architectures.

Figure 16 Production Use Case

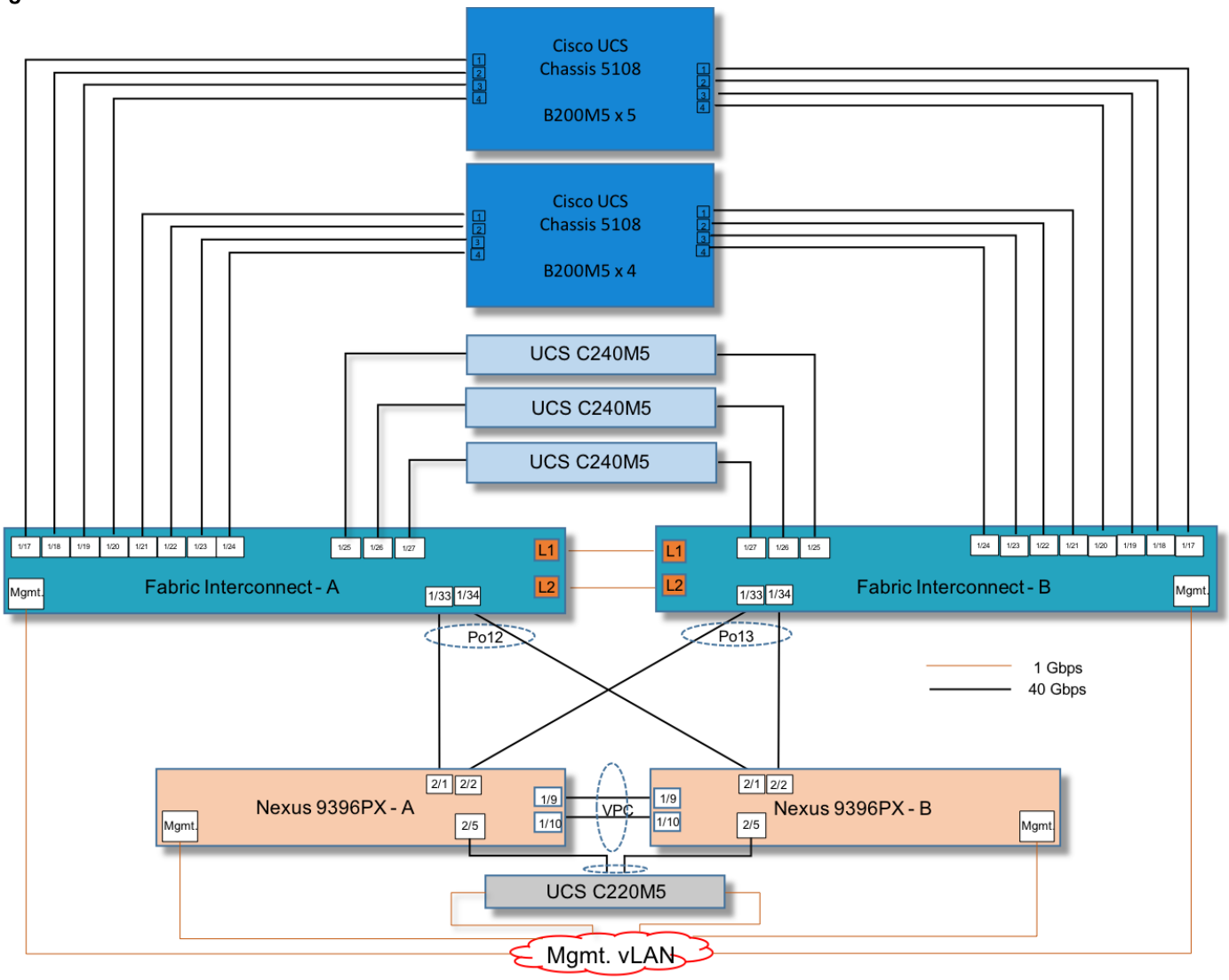
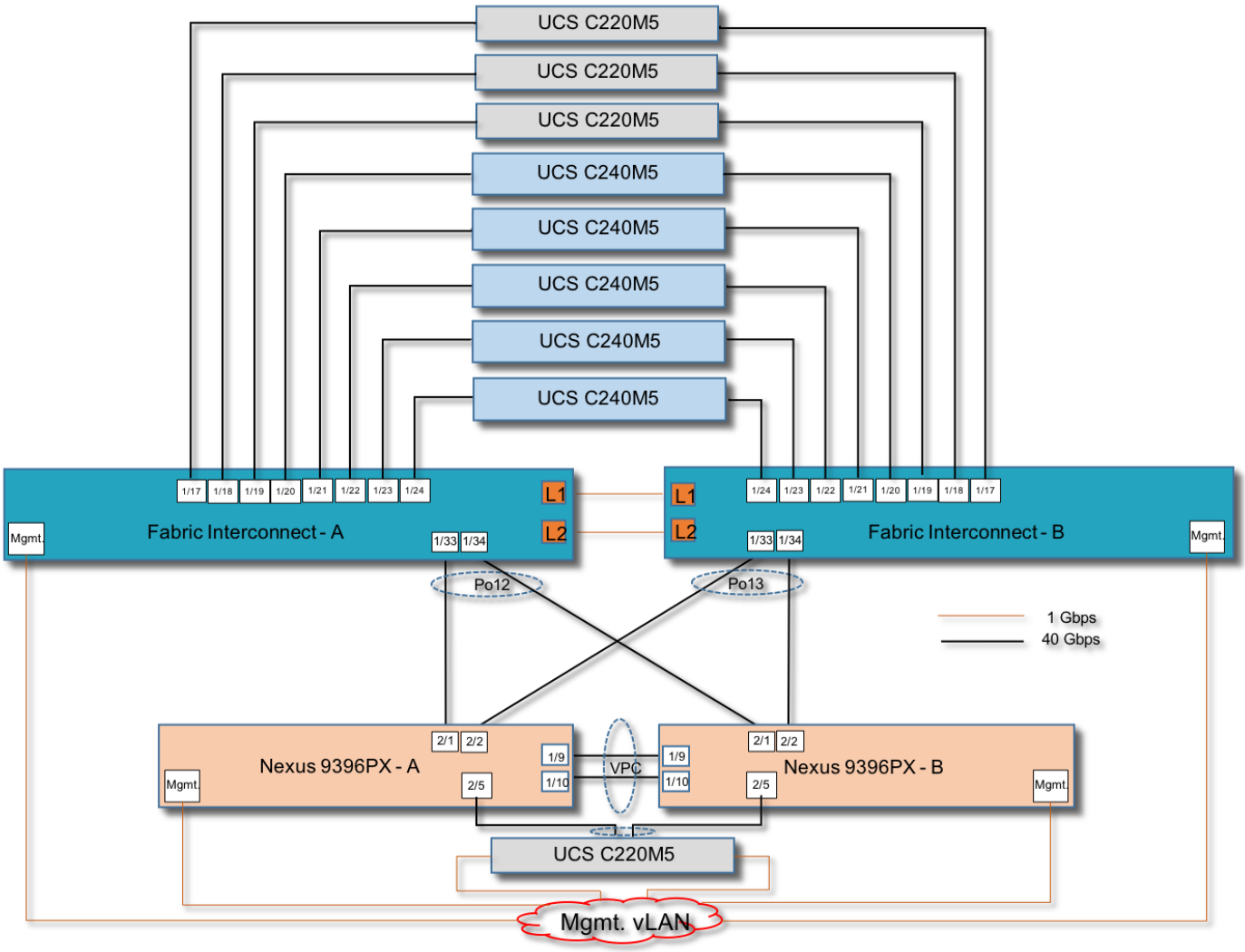


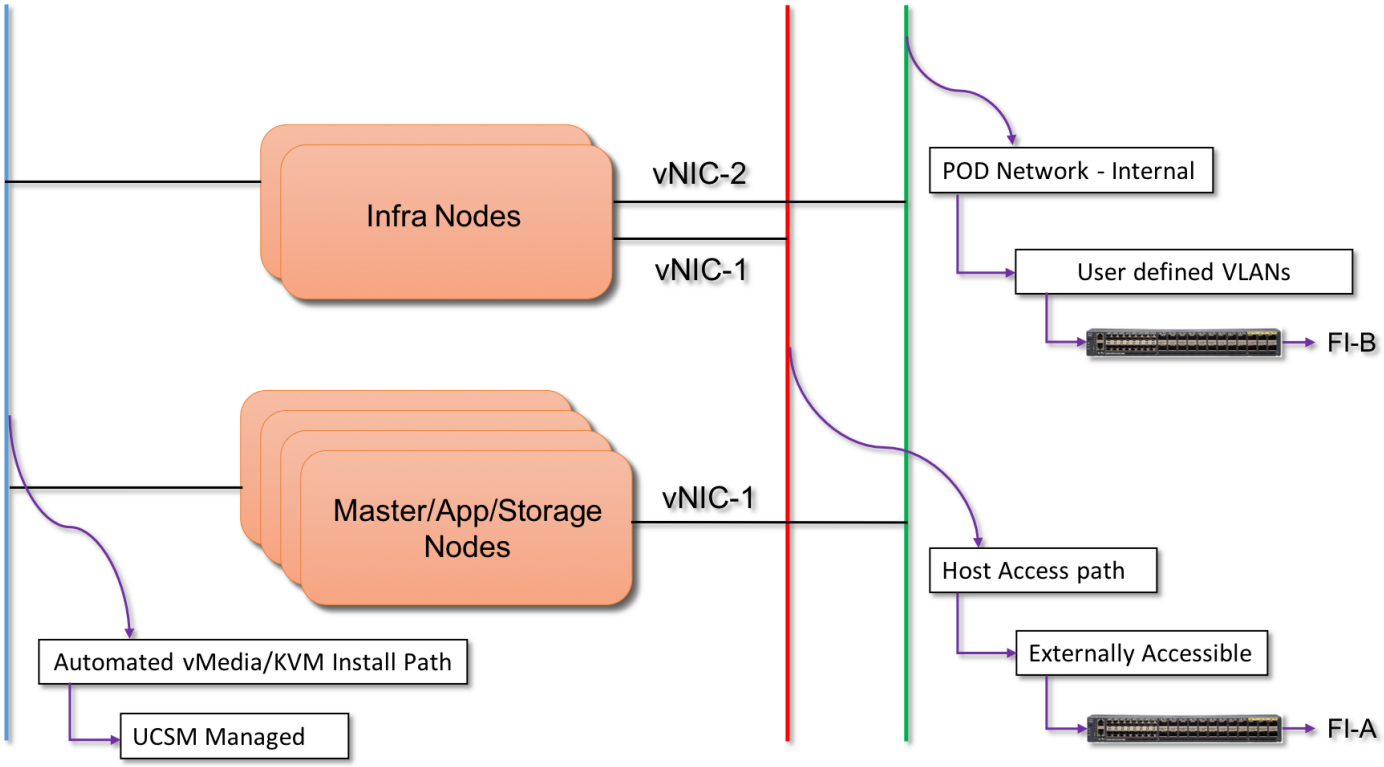
Figure 17 Dev/Test Use Case



Logical Network Connectivity

This section provides details on how Cisco VIC enables application containers to use dedicated I/O path for network access in a secured environment with multi-tenancy. With Cisco VIC, containers get a dedicated physical path to a classic L2 VLAN topology with better line rate efficiency. To further enhance the value of Cisco UCS by optimizing the infrastructure utilization, the traffic paths were configured to segregate all management/control traffic through fabric interconnect A, and container data traffic through fabric interconnect B.

Figure 18 Logical Network Diagram

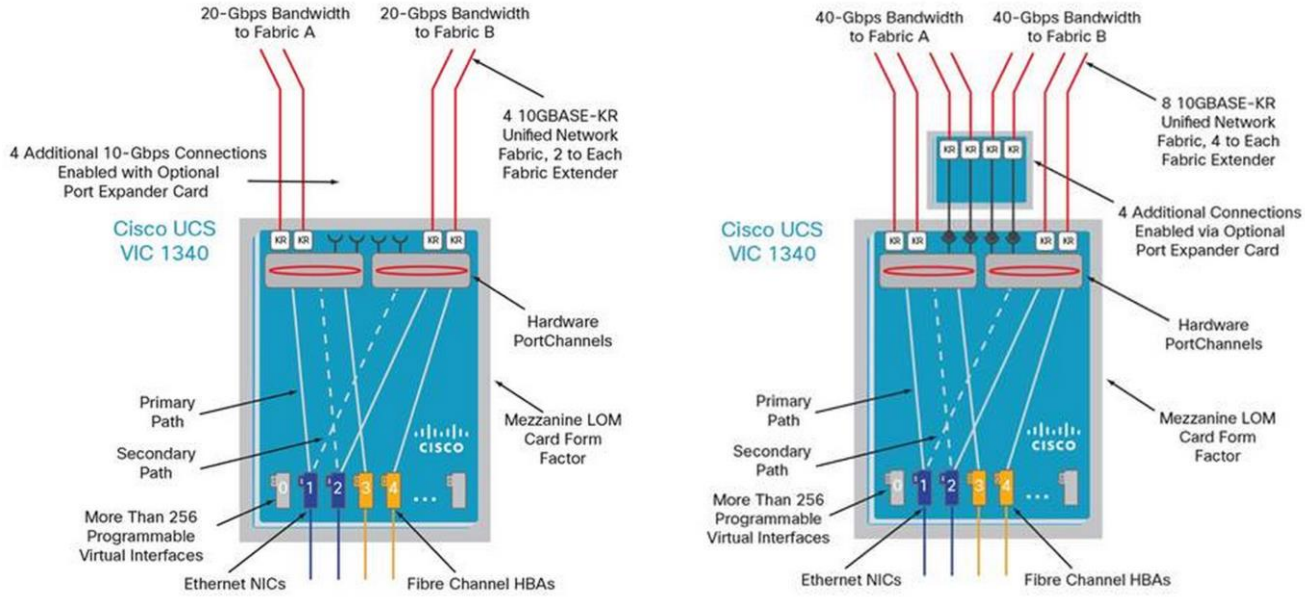


UCS Design – Networking Features

Server Traffic Aggregation

Selection of the FEX, VIC and Mezzanine cards plays an important role in determining the aggregate traffic throughput to and from a server. The following figure shows an overview of backplane connectivity of Cisco VIC 1340 architecture. The number of KR lanes indicates the 10GbE paths available to the chassis and therefore blades. Depending on the models of I/O modules and VICs, traffic aggregation differs. 2204XP enables 2 KR lanes per half-width blade slot whereas the 2208XP enables all four. Similarly, number of KR lanes varies based on selection of VIC 1240/1340, VIC 1240/1340 with Port Expander and VIC 1280/1380.

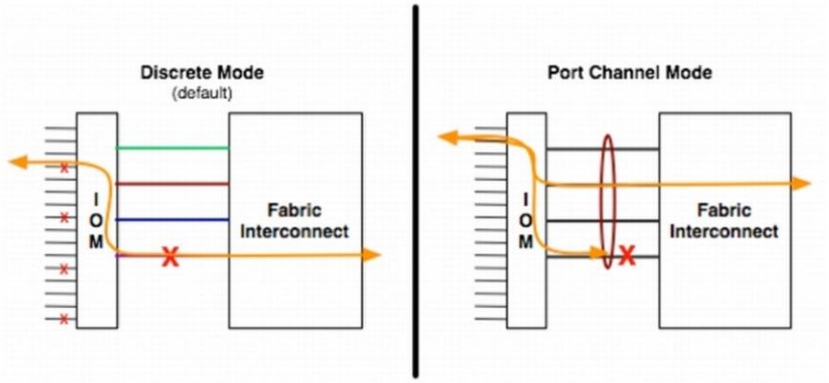
Figure 19 Cisco UCS VIC 1340 Architecture



Chassis/FEX discovery

Cisco Unified Computing System can be configured to discover a chassis using Discrete Mode or the Port-Channel mode. In Discrete Mode each FEX KR connection and therefore server connection is tied or pinned to a network fabric connection homed to a port on the Fabric Interconnect. In the presence of a failure on the external "link" all KR connections are disabled within the FEX I/O module. In Port-Channel mode, the failure of a network fabric link allows for redistribution of flows across the remaining port channel members. Port-Channel mode therefore is less disruptive to the fabric and hence recommended in the FlexPod designs.

Figure 20 Chassis Discovery Policy – Discrete vs. Port Channel

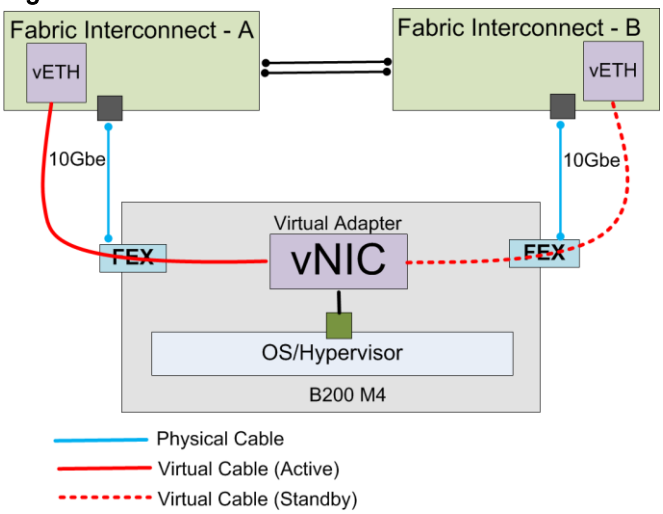


Fabric Failover for Ethernet: Highly Available vNIC

Each adapter in Cisco UCS is a dual-port adapter that connects to both fabrics (A and B). The two fabrics in Cisco UCS provide failover protection in the event of planned or unplanned component downtime in one of the fabrics. Typically, host software such as NIC teaming for Ethernet or multipath I/O (MPIO) for Fibre Channel provides failover across the two fabrics. A vNIC in Cisco UCS is a host-presented PCI device that is centrally managed by Cisco UCS Manager. The fabric-based failover feature, which you enable by selecting the high-availability vNIC option in the service profile definition, allows network interface virtualization (NIV)-

capable adapters and the fabric interconnects to provide active-standby failover for Ethernet vNICs without any NIC-teaming software on the host. For unicast traffic failover, the fabric interconnect in the new path sends Gratuitous Address Resolution Protocols (GARPs). This process refreshes the forwarding tables on the upstream switches.

Figure 21 Fabric Failover

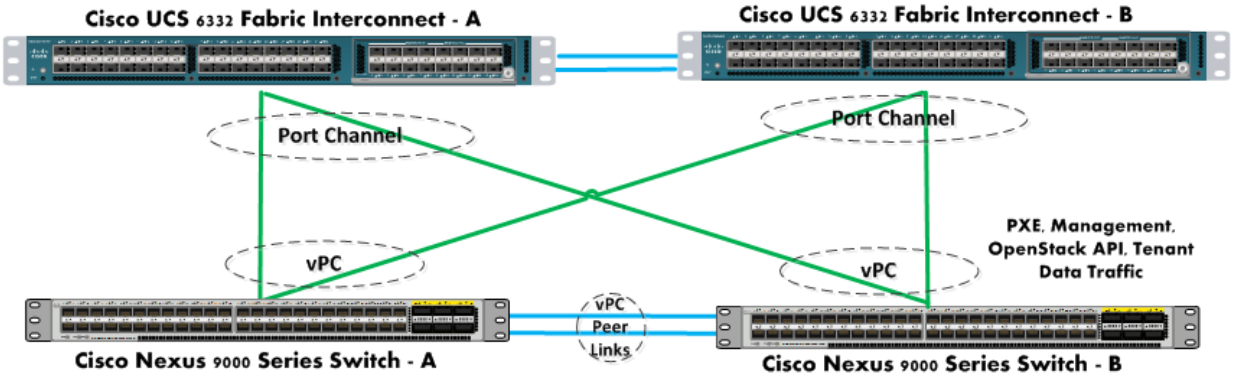


Cisco UCS fabric failover is an important feature because it reduces the complexity of defining NIC teaming software for failover on the host. It does this transparently in the fabric based on the network property that is defined in the service profile.

Cisco UCS Physical Connectivity to Nexus 9000

Cisco UCS Fabric Interconnects are connected with Nexus 93180CY as shown in the Figure 14. It is configured with two port-channels, one from each Cisco UCS Fabric Interconnect to Cisco Nexus 93180CY. These port channels carry all the storage and data traffic. In this validated design, two uplinks from each fabric interconnect to the leaf switches have been utilized for an aggregate bandwidth of 40Gbe (4x10Gbe). However, the number of links can be increased based on the customer's throughput requirements.

Figure 22 Cisco UCS Physical Connectivity to Cisco Nexus 9000 Series Switch



Storage Overview

Container-native storage solution from Red Hat makes OpenShift Container Platform a fully hyperconverged infrastructure where storage containers can co-reside with the compute containers. Storage plane is based on containerized CNS services and controls every storage server. Heketi is a part of the container-native storage architecture and controls all of the nodes that are members of storage cluster. Heketi also provides an API through which storage space for containers can be easily requested. While Heketi provides an endpoint for storage cluster, the object that makes calls to its API from OpenShift clients is called a Storage Class. It is a Kubernetes and OpenShift object that describes the type of storage available for the cluster and can dynamically send storage requests when a persistent volume claim is generated.

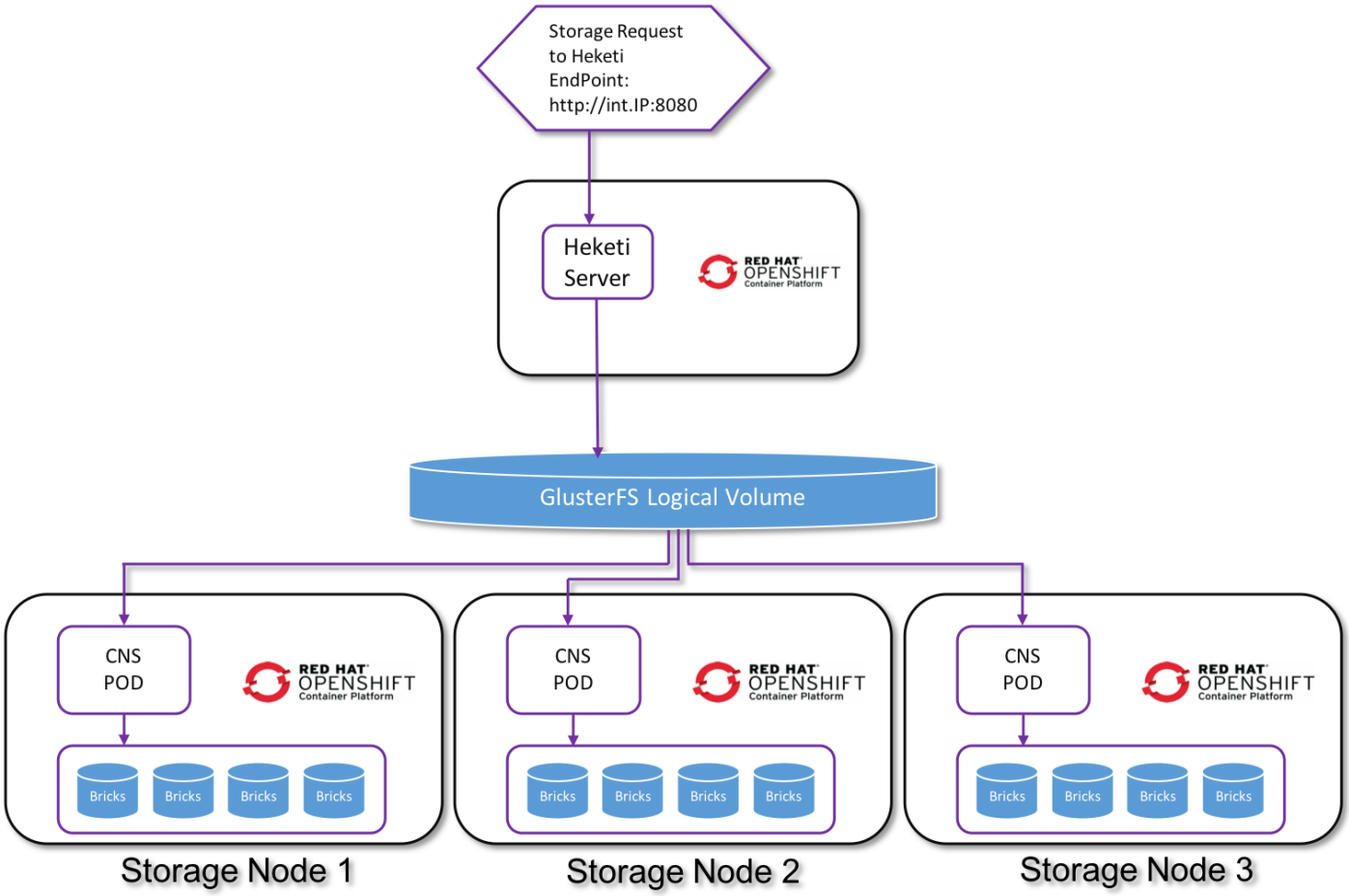
Docker Engine and images are stored on individual nodes for containers and this is achieved by provisioning local storage using device mapper driver in direct LVM mode. Storage profile along with disk configuration policy from UCS manager will be used to carve out boot and Docker data LUNS. Operating system and Docker data will be residing locally on individual cluster nodes and will have RAID configuration as appropriate for B-series and C-series topology.

This solution is designed to take advantage of Container-native storage solution from Red Hat by utilizing its capability of co-residing storage containers and application containers on the same node. This feature is being used for Dev/Test use case architecture, where three of the Application node are co-hosting storage services. While for Production usecase, the solution uses a 3-node dedicated storage cluster with CNS services.

Storage Architecture

The solution design has UCS C240M5 nodes which serve as storage back-end for providing persistent storage to the OpenShift Container Platform. While Production usecase architecture is designed to use all-flash storage devices for high performance and capacity, Dev/Test usecase architecture is designed with hard disk drives. Both the architectures have sufficient infrastructure capacity to scale-out based on user needs. Additional storage capacities can be added to the existing nodes or the storage cluster itself can be scaled-out for future needs.

Figure 23 Persistent Volume Request – Container-native Storage for Red Hat OpenShift Container Platform



Persistent Storage

Persistent Storage As the name suggests, persistent storage allows your saved data and storage resources to exist even if an associated instance is removed.

Managing storage is a distinct problem from managing compute resources. OpenShift Container Platform leverages the Kubernetes persistent volume (PV) framework to allow administrators to provision persistent storage for a cluster. Using persistent volume claims (PVCs), developers can request PV resources without having specific knowledge of the underlying storage infrastructure.

PVCs are specific to a project and are created and used by developers as a means to use a PV. PV resources on their own are not scoped to any single project; they can be shared across the entire OpenShift Container Platform cluster and claimed from any project. After a PV has been attached to a PVC, that PV cannot then be attached to additional PVCs. This has the effect of scoping a bound PV to a single namespace (that of the binding project).

PVs are defined by a PersistentVolume API object, which represents a piece of existing networked storage in the cluster that has been provisioned by an administrator. It is a resource in the cluster just like a node is a cluster resource. PVs are volume plug-ins like Volumes, but have a lifecycle independent of any individual

pod that uses the PV. PV objects capture the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system.

PVCs are defined by a PersistentVolumeClaim API object, which represents a request for storage by a developer. It is similar to a pod in that pods consume node resources and PVCs consume PV resources. For example, pods can request specific levels of resources (for example, CPU and memory), while PVCs can request specific storage capacity and access modes (for example, they can be mounted once read/write or many times read-only).

A PersistentVolume provided by CNS back-end, as used in this solution, can be mounted with following access modes -

Table 4 PV Access Modes

Access Mode	CLI Abbreviation	Description
ReadWriteOnce	RWO	The Volume can be mounted as read-write by a single node
ReadOnlyMany	ROX	The Volume can be mounted read-only by many nodes
ReadWriteMany	RWX	The Volume can be mounted read-write by many nodes

GlusterFS plugin enables PersistentVolume provisioning in OpenShift Container Platform and supports all the above access modes.

Container-native Storage Solution from Red Hat

This solution integrates CNS deployment and management with OpenShift services. As a result, persistent storage is delivered within an OpenShift pod that provides both compute and file storage.

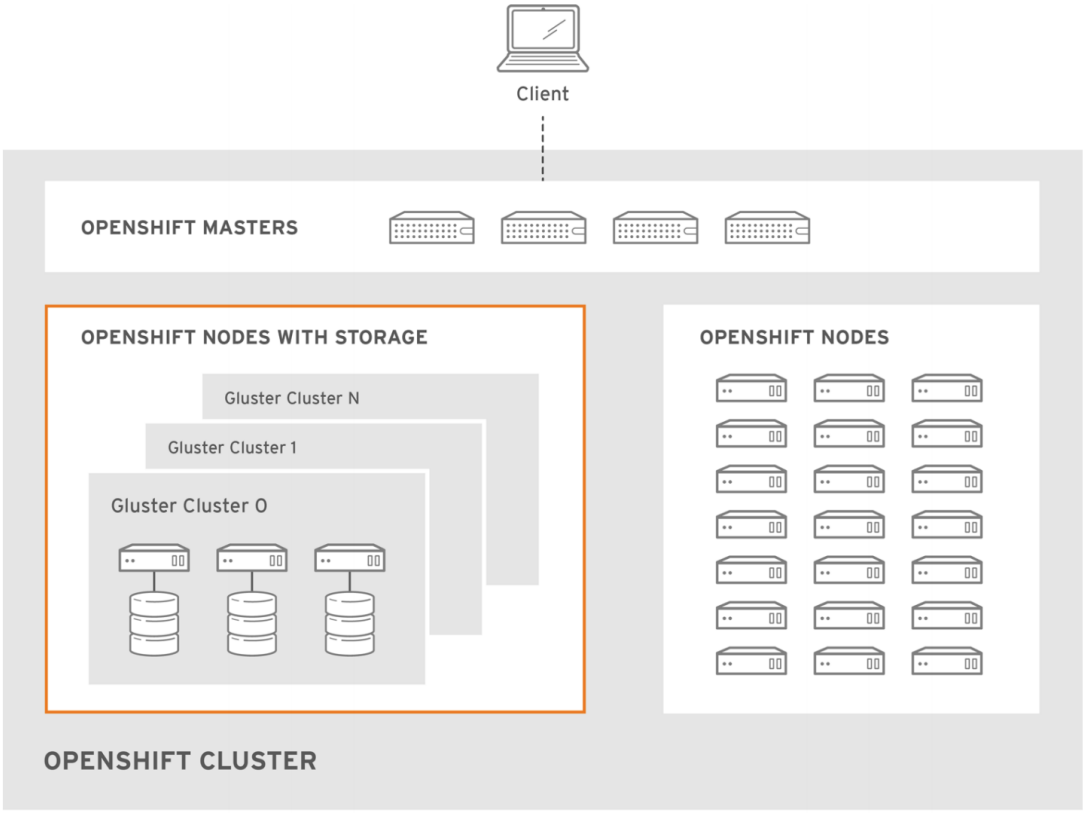
Container-native storage for OpenShift Container Platform is built around three key technologies:

- OpenShift provides the platform as a service (PaaS) infrastructure based on Kubernetes container management. Basic OpenShift architecture is built around multiple master systems where each system contains a set of nodes.
- CNS provides the containerized distributed storage based on Red Hat Gluster Storage. Each CNS volume is composed of a collection of bricks, where each brick is the combination of a node and an export directory.
- Heketi provides the CNS volume life cycle management. It creates the CNS volumes dynamically and supports multiple Red Hat Gluster Storage clusters.

The following list provides the administrators a solution workflow. The administrators can:

- Create multiple persistent volumes (PV) and register these volumes with OpenShift.
- Developers then submit a persistent volume claim (PVC).
- A PV is identified and selected from a pool of available PVs and bound to the PVC.
- The OpenShift pod then uses the PV for persistent storage.

Figure 24 Container-native storage for OpenShift Container Platform



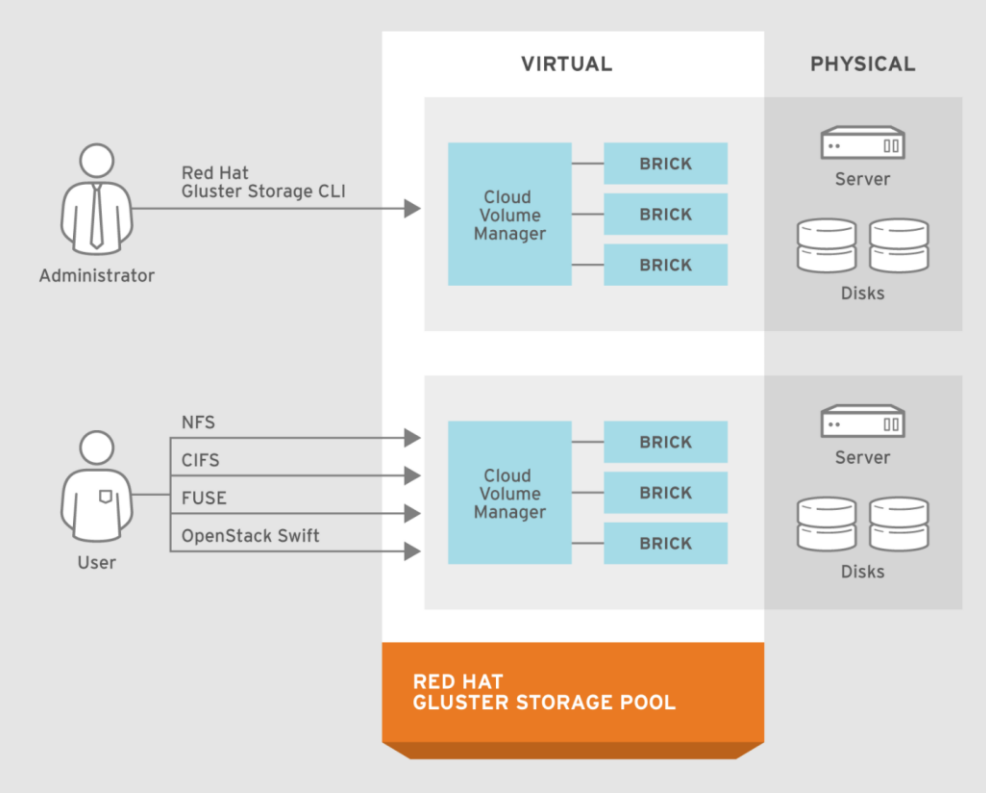
Red Hat Gluster Storage

The Red Hat Gluster Storage design is a completely new method of architecting storage. The result is a system that has immense scalability, is highly resilient, and offers extraordinary performance.

In a scale-out system, one of the biggest challenges is keeping track of the logical and physical locations of data and metadata. Most distributed systems solve this problem by creating a metadata server to track the location of data and metadata. As traditional systems add more files, more servers, or more disks, the central metadata server becomes a performance bottleneck, as well as a central point of failure.

Unlike other traditional storage solutions, Red Hat Gluster Storage does not need a metadata server, and locates files algorithmically using an elastic hashing algorithm. This no-metadata server architecture ensures better performance, linear scalability, and reliability.

Figure 25 Red Hat Gluster Storage Architecture



Software and Hardware Versions

Hardware Components

The following infrastructure components are needed for the two architectural options under the solution.

Table 5 Architecture - I: Production Use Case (high performance)

Component	Model	Quantity	Description
App nodes	B200M5	4	CPU – 2 x 6130@2.1GHz,16 Cores each Memory – 24 x 16GB 2666 DIMM – total of 384G SSDs – 2x960GB 6G SATA -EV (Intel S4500 Enterprise Value) Network Card – 1x1340 VIC + 1xPort Expander for 40Gig network I/O Raid Controller – Cisco MRAID 12 G SAS Controller
Master nodes	B200M5	3	CPU – 2 x 4114@2.2GHz,10Cores each Memory – 12 x 16GB 2400 DIMM – total of 192G SSDs – 2x240GB 6G SATA -EV Network Card – 1x1340 VIC + 1xPort Expander for 40Gig network I/O Raid Controller – Cisco MRAID 12 G SAS Controller
Infra nodes	B200M5	2	CPU – 2 x 4114@2.2GHz,10Cores each Memory – 12 x 16GB 2400 DIMM – total of 192G SSDs – 2x240GB 6G SATA -EV Network Card – 1x1340 VIC + 1xPort Expander for

			40Gig network I/O Raid Controller – Cisco MRAID 12 G SAS Controller
Bastion node	C220M5	1	CPU – 2 x 4114@2.2GHz,10Cores each Memory – 12 x 16GB 2400 DIMM – total of 192G HDDs – 2x300GB12GSAS10KSFF Network Card – 1x1385 VIC RAID Controller - Cisco MRAID 12 G SAS Controller
Storage nodes	C240M5SX	3	CPU – 2 x 6130@2.1GHz,16 Cores each Memory – 12 x 16GB 2666 DIMM – total of 192G SSDs – 2x240GB 6G SATA -EV SSDs - 20x3.8TB SATA (Intel S4500 Enterprise Value) Network Card – 1x1385 VIC Raid Controller – Cisco MRAID 12 G SAS Controller
Chassis	UCS 5108	2	
IO Modules	IOM 2304	4	
Fabric Interconnects	UCS 6332-16UP	2	
Switches	Nexus 9396PX	2	

Table 6 Architecture - II: Dev/Test Use Case

Component	Model	Quantity	Description
Application+Storage nodes co-located	C240M5SX	3	CPU – 2 x 4114@2.2GHz,10Cores each Memory – 12 x 16GB 2400 DIMM – total of 192G HDDs – 2x300GB12GSAS10KSFF(Internal Boot Drives) HDDs – 20x1.2TB12GSAS10KSFF Network Card – 1x1385 VIC Raid Controller – Cisco MRAID 12 G SAS Controller
Application+Infra nodes co-located	C240M5SX	2	CPU – 2 x 4114@2.2GHz,10Cores each Memory – 12 x 16GB 2400 DIMM – total of 192G HDDs – 2x300GB12GSAS10KSFF(Internal Boot Drives) Network Card – 1x1385 VIC Raid Controller – Cisco MRAID 12 G SAS Controller
Master nodes	C220M5	3	CPU – 2 x 4114@2.2GHz,10Cores each Memory – 12 x 16GB 2400 DIMM – total of 192G HDDs – 2x300GB12GSAS10KSFF Network Card – 1x1385 VIC Raid Controller – Cisco MRAID 12 G SAS Controller
Bastion node	C220M5	1	CPU – 2 x 4114@2.2GHz,10Cores each Memory – 12 x 16GB 2400 DIMM – total of 192G HDDs – 2x300GB12GSAS10KSFF Network Card – 1x1385 VIC RAID Controller - Cisco MRAID 12 G SAS Controller
Fabric Interconnects	UCS 6332-16UP	2	
Switches	Nexus 9396PX	2	

Software Components

Following software components are used for this solution in the CVD.

Table 7 Software Components

Component	Version
UCS Manager	3.2.(3d)
Red Hat Enterprise Linux	7.5
Red Hat Enterprise Linux Atomic Host	7.5
Red Hat OpenShift Container Platform	3.9
Container-native storage from Red Hat	3.9
Kubernetes	1.9.1
Docker	1.13.1
Red Hat Ansible Automation	2.4.4
Etcd	3.2.18
Open vSwitch	2.7.3
HAProxy-Router	1.8.1
HAProxy-Loadbalancer	1.5.18
Keepalived	1.3.5
Red Hat Gluster Storage	3.3.0
GlusterFS	3.8.4
Heketi	6.0.0

About the Authors

Rajesh Kharya, Technical Leader, Cisco UCS Solutions Engineering, Cisco Systems Inc.

Rajesh Kharya is Tech lead with Solutions Engineering team, Computing System Product Group. His focus includes Open Source Technologies, Cloud, Software Defined Storage, Containers and Automation.

Sindhu Sudhir, Technical Marketing Engineer, Cisco UCS Solutions Engineering, Cisco Systems Inc.

Sindhu Sudhir is part of Cisco UCS Solutions Engineering team. In her current role she is focusing on Container technologies and infrastructure automation on Cisco UCS platform.

Lukasz Sztachanski, Cloud Solutions Engineer, Data Center Solutions Group, Intel

Focused on developing and designing cloud-orchestration stacks at Intel. His main areas of expertise are Kubernetes, OpenShift and container-native technologies, where he emphasize hardware integration, performance and robustnes. During his over-a-decade career he managed large DC in EMEA region and complex infrastructure for industry-leading SaaS company.

Lukasz Luczaj, Cloud Solutions Engineer, Data Center Solutions Group, Intel

From the beginning of his career he has been working on cloud technologies and software defined architectures with a strong focus on container-based solutions (Docker, Kubernetes, OpenShift). His goal is to optimize performance, provide reliability and make private cloud solutions enterprise ready according to the rule "Build it, Break It, Fix It". He is a strong enthusiast of open source and automation technologies.

Acknowledgements

- Cisco Systems: Vishwanath Jakka, Babu Mahadevan
- Intel: Seema Mehta
- Red Hat: JP Jung, Chris Morgan, Dave Cain