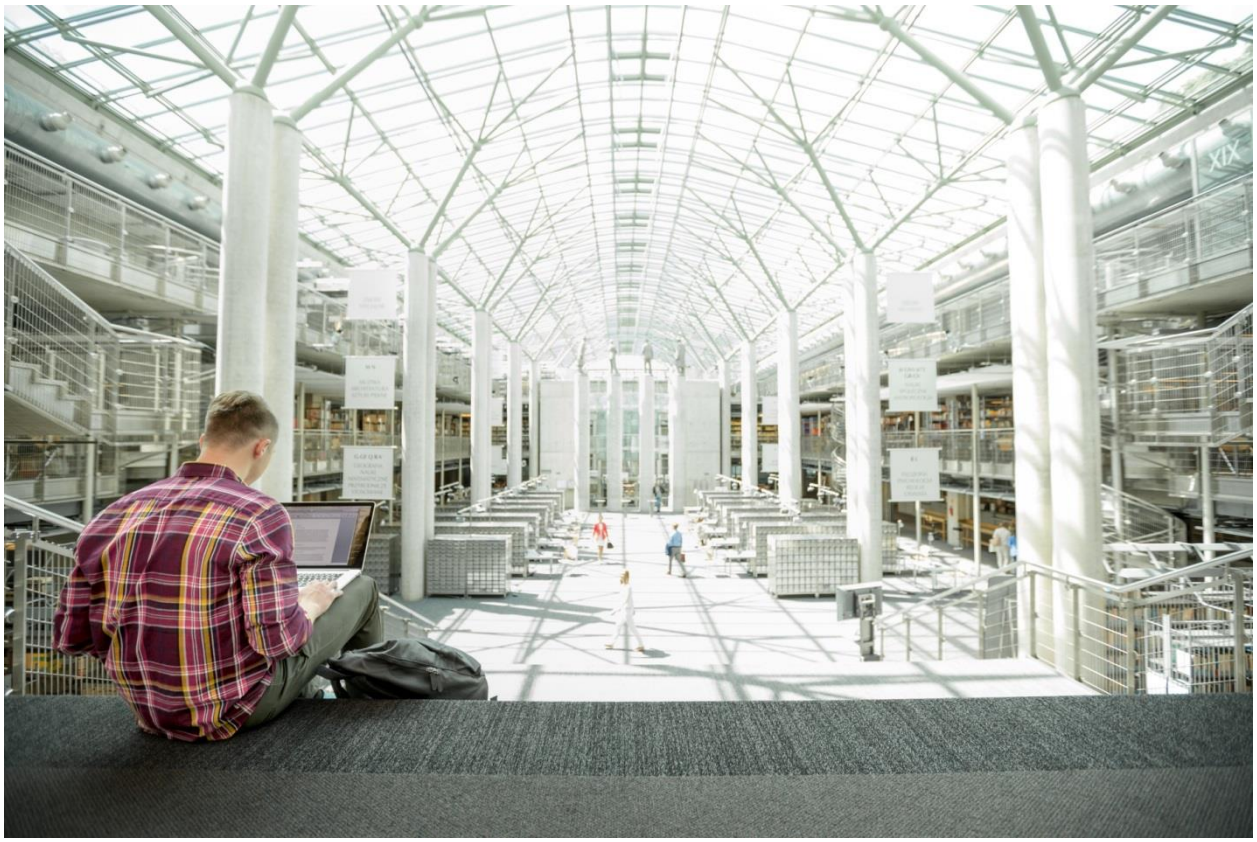


Cisco UCS S3260 Storage Server with SwiftStack Software Defined Object Storage

Design and Deployment Guide

Last Updated: November 28, 2017


CISCO
VALIDATED
DESIGN

About Cisco Validated Designs

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.

Table of Contents

Executive Summary

Cisco Validated Design program consist of systems and solutions that are designed, tested, and documented to facilitate and improve customer deployments. These designs incorporate a wide range of technologies and products into a portfolio of solutions that have been developed to address the business needs of our customers.

Most of the modern data centers are moving away from traditional file system type storage, to object storages. Object storage offers simple management, unlimited scalability and custom metadata for objects. With its low cost per gigabyte of storage, Object storage systems are suited for archive, backup, Life sciences, video surveillance, healthcare, multimedia, message and machine data, etc.

SwiftStack object storage is a massively scalable software defined storage system that can achieve enterprise class reliability, scale-out capacity and lower costs with industry standard server solution.

The Cisco UCS S3260 Storage Server, originally designed for the data center, makes it an excellent fit for unstructured data workloads such as backup, archive, and cloud data. The S3260 delivers a complete infrastructure with exceptional scalability for computing and storage resources together with 40 Gigabit Ethernet networking.

Cisco and SwiftStack are collaborating to offer Customers, object storage solutions. The power of Cisco UCS management framework for S3260 nodes complements, simple but reliable framework from SwiftStack for object storage,

The reference architecture described in this document is a realistic use case for deploying SwiftStack object storage on Cisco UCS storage server S3260 and rack mounted server C220. The document covers step by step instructions for setting UCS hardware for SwiftStack Object and Controller nodes, installing Red Hat Linux Operating system, Installing SwiftStack Software along with Performance data collected to provide scale up and scale down guidelines, issues and workarounds evolved during installation, what needs to be done to leverage High Availability from both hardware and software for Business continuity, lessons learnt, best practices evolved while validating the solution, etc.

Solution Overview

Introduction

Object storage is a highly scalable system for organizing and storing data objects. Object storage does not use a file system structure, instead it ingests data as objects with unique keys into a flat directory structure and the metadata is stored with the objects instead of hierarchical journal or tree. Search and retrieval is performed via these unique keys for searching. Most of the newly generated data is unstructured today. With about 80 percent of data being unstructured, new approaches using x86 servers are proving to be more cost effective, providing storage that can be expanded as easily as your data grows. Object storage is the newest approach for handling massive amounts of data.

SwiftStack based on open source Swift, enables applications to store data that is usable by object applications and cloud-native applications alike.

Together with Cisco UCS, SwiftStack Storage can deliver a fully enterprise-ready solution that can manage different workloads and still remain flexible. The Cisco UCS S3260 Storage Server is an excellent platform to use with the main types of Swift workloads, such as capacity-optimized and performance-optimized workloads. It is also excellent for workloads with a large number of I/O operations per second and scales well for varying work load and block sizes.

This document describes the architecture, design and deployment procedures of SwiftStack object storage on Cisco UCS S3260 servers with 2 x C220M4 rack servers.

Audience

The audience for this document includes, but is not limited to, sales engineers, field consultants, professional services, IT managers, partner engineers, IT architects, and customers who want to take advantage of an infrastructure that is built to deliver IT efficiency and enable IT innovation. The reader of this document is expected to have the necessary training and background to install and configure Red Hat Enterprise Linux, Cisco Unified Computing System (UCS) and Cisco Nexus Switches as well as a high-level understanding of Object storage, Swift and SwiftStack. External references are provided where applicable and it is recommended that the reader be familiar with these documents.

Readers are also expected to be familiar with the infrastructure, network and security policies of the customer installation.

Purpose of this Document

This document describes the step by step installation of SwiftStack on Cisco UCS platform. It also covers High Availability use cases, Performance and Scalability tests, workarounds, if any evolved while validating the design along with Operational best practices.

Solution Summary

This solution is focused on SwiftStack storage on Red Hat Linux 7 on Cisco Unified Computing System. The advantages of Cisco UCS and SwiftStack combine to deliver an object storage solution that is simple to install, scalable and performant. The configuration uses the following components for the deployment:

Cisco Unified Computing System (UCS)

- Cisco UCS 6332 Series Fabric Interconnects
- Cisco UCS S3260 storage servers.
- Cisco S3260 system IO controller with VIC 1380
- Cisco C220M4 servers with VIC 1387

Cisco Nexus C9332PQ Series Switches

SwiftStack storage 5.x.

Red Hat Enterprise Linux 7.x

The solution includes the following features:

Infrastructure for large scale object storage

Design and Implementation of a Swift Object Storage solution on Cisco UCS S3260 Storage Server

Simplified infrastructure management with Cisco UCS Manager

Architectural scalability – linear scaling based on network, storage, and compute requirements

The scope is limited to the infrastructure pieces of the solution. However, an attempt has been made to add any discoveries made as part of the validation.

Technology Overview

Cisco Unified Computing System

The Cisco Unified Computing System (Cisco UCS) is a state-of-the-art data center platform that unites computing, network, storage access, and virtualization into a single cohesive system.

The main components of Cisco Unified Computing System are:

Computing - The system is based on an entirely new class of computing system that incorporates rack-mount and blade servers based on Intel Xeon Processor E5 and E7. The Cisco UCS servers offer the patented Cisco Extended Memory Technology to support applications with large datasets and allow more virtual machines (VM) per server.

Network - The system is integrated onto a low-latency, lossless, 40-Gbps unified network fabric. This network foundation consolidates LANs, 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. By unifying the storage access, the Cisco Unified Computing System can access storage over Ethernet (NFS or iSCSI), Fibre Channel, and Fibre Channel over Ethernet (FCoE). This provides customers with choice for storage access and investment protection. In addition, the server administrators can pre-assign storage-access policies for system connectivity to storage resources, simplifying storage connectivity, and management for increased productivity.

The Cisco Unified Computing System is designed to deliver:

A reduced Total Cost of Ownership (TCO) and increased business agility.

Increased IT staff productivity through just-in-time provisioning and mobility support.

A cohesive, integrated system which unifies the technology in the data center.

Industry standards supported by a partner ecosystem of industry leaders.

Cisco UCS S3260 Storage Server

The Cisco UCS® S3260 Storage Server (Figure 1) is a modular, high-density, highly-available dual node rack server well suited for service providers, enterprises, and industry-specific environments. It addresses the need for dense cost effective storage for the ever-growing data needs. Designed for a new class of cloud-scale applications, it is simple to deploy and excellent for big data applications, software-defined storage environments and other unstructured data repositories, media streaming, and content distribution.

Figure 1 Cisco UCS S3260 Storage Server

Extending the capability of the Cisco UCS C3000 portfolio, the Cisco UCS S3260 helps you achieve the highest levels of data availability. With dual-node capability that is based on the Intel® Xeon® processor E5-2600 v4 series, it features up to 600 TB of local storage in a compact 4-rack-unit (4RU) form factor. All hard-disk drives can be asymmetrically split between the dual-nodes and are individually hot-swappable. The drives can be built-in in an enterprise-class Redundant Array of Independent Disks (RAID) redundancy or be in a pass-through mode.

This high-density rack server comfortably fits in a standard 32-inch depth rack, such as the Cisco® R42610 Rack.

The Cisco UCS S3260 Server can be deployed as standalone server or as part of the Cisco Unified Computing System™ (Cisco UCS) in both bare metal or virtualized environments. Its modular architecture reduces total cost of ownership (TCO) by allowing you to upgrade individual components over time and as use cases evolve, without having to replace the entire system.

The Cisco UCS S3260 uses a modular server architecture that, using Cisco's blade technology expertise, allows you to upgrade the computing or network nodes in the system without the need to migrate data migration from one system to another. It delivers:

- Dual server nodes
- Up to 36 computing cores per server node
- Up to 60 drives mixing a large form factor (LFF) with up to 28 solid-state disk (SSD) drives plus 2 SSD SATA boot drives per server node
- Up to 1 TB of memory per server node (2 terabyte [TB] total)
- Support for 12-Gbps serial-attached SCSI (SAS) drives
- A system I/O Controller with Cisco VIC 1300 Series Embedded Chip supporting Dual-port 40Gbps
- High reliability, availability, and serviceability (RAS) features with tool-free server nodes, system I/O controller, easy-to-use latching lid, and hot-swappable and hot-pluggable components

Cisco UCS C220 M4 Rack Server

The Cisco UCS® C220 M4 Rack Server (Figure 2) is the most versatile, general-purpose enterprise infrastructure and application server in the industry. It is a high-density two-socket enterprise-class rack server that delivers industry-leading performance and efficiency for a wide range of enterprise 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.

Figure 2 Cisco UCS C220 M4 Rack Server

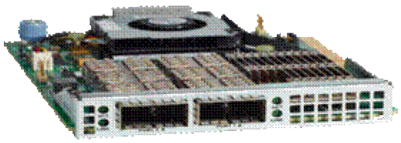
The enterprise-class Cisco UCS C220 M4 server extends the capabilities of the Cisco UCS portfolio in a 1RU form factor. It incorporates the Intel® Xeon® processor E5-2600 v4 and v3 product family, next-generation DDR4 memory, and 12-Gbps SAS throughput, delivering significant performance and efficiency gains. The Cisco UCS C220 M4 rack server delivers outstanding levels of expandability and performance in a compact 1RU package:

- Up to 24 DDR4 DIMMs for improved performance and lower power consumption
- Up to 8 Small Form-Factor (SFF) drives or up to 4 Large Form-Factor (LFF) drives
- Support for 12-Gbps SAS Module RAID controller in a dedicated slot, leaving the remaining two PCIe Gen 3.0 slots available for other expansion cards
- A modular LAN-on-motherboard (mLOM) slot that can be used to install a Cisco UCS virtual interface card (VIC) or third-party network interface card (NIC) without consuming a PCIe slot
- Two embedded 1Gigabit Ethernet LAN-on-motherboard (LOM) ports

Cisco UCS Virtual Interface Card 1387

The Cisco UCS Virtual Interface Card (VIC) 1387 (Figure 3) is a Cisco® innovation. It provides a policy-based, stateless, agile server infrastructure for your data center. This dual-port Enhanced Quad Small Form-Factor Pluggable (QSFP) half-height PCI Express (PCIe) modular LAN-on-motherboard (mLOM) adapter is designed exclusively for Cisco UCS C-Series and 3260 Rack Servers. The card supports 40 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. The card can present more than 256 PCIe standards-compliant interfaces to the host, and these can be dynamically configured as either network interface cards (NICs) or host bus adapters (HBAs). In addition, the VIC supports Cisco Data Center Virtual Machine Fabric Extender (VM-FEX) technology. This technology extends the Cisco UCS fabric interconnect ports to virtual machines, simplifying server virtualization deployment.

Figure 3 Cisco UCS Virtual Interface Card 1387



The Cisco UCS VIC 1387 provides the following features and benefits:

Stateless and agile platform: The personality of the card is determined dynamically at boot time using the service profile associated with the server. The number, type (NIC or HBA), identity (MAC address and World Wide Name [WWN]), failover policy, bandwidth, and quality-of-service (QoS) policies of the PCIe interfaces are all determined using the service profile. The capability to define, create, and use interfaces on demand provides a stateless and agile server infrastructure

Network interface virtualization: Each PCIe interface created on the VIC is associated with an interface on the Cisco UCS fabric interconnect, providing complete network separation for each virtual cable between a PCIe device on the VIC and the interface on the fabric interconnect

Cisco UCS 6300 Series Fabric Interconnect

The Cisco UCS 6300 Series Fabric Interconnects are a core part of Cisco UCS, providing both network connectivity and management capabilities for the system (Figure 4). The Cisco UCS 6300 Series offers line-rate, low-latency, lossless 10 and 40 Gigabit Ethernet, Fibre Channel over Ethernet (FCoE), and Fibre Channel functions.

Figure 4 Cisco UCS 6300 Series Fabric Interconnect

The Cisco UCS 6300 Series provides the management and communication backbone for the Cisco UCS B-Series Blade Servers, 5100 Series Blade Server Chassis, and C-Series Rack Servers managed by Cisco UCS. All servers attached to the fabric interconnects become part of a single, highly available management domain. In addition, by supporting unified fabric, the Cisco UCS 6300 Series provides both LAN and SAN connectivity for all servers within its domain.

From a networking perspective, the Cisco UCS 6300 Series uses a cut-through architecture, supporting deterministic, low-latency, line-rate 10 and 40 Gigabit Ethernet ports, switching capacity of 2.56 terabits per second (Tbps), and 320 Gbps of bandwidth per chassis, independent of packet size and enabled services. The product family supports Cisco® low-latency, lossless 10 and 40 Gigabit Ethernet unified network fabric capabilities, which increase the reliability, efficiency, and scalability of Ethernet networks. The fabric interconnect supports multiple traffic classes over a lossless Ethernet fabric from the server through the fabric interconnect. Significant TCO savings can be achieved with an FCoE optimized server design in which network interface cards (NICs), host bus adapters (HBAs), cables, and switches can be consolidated.

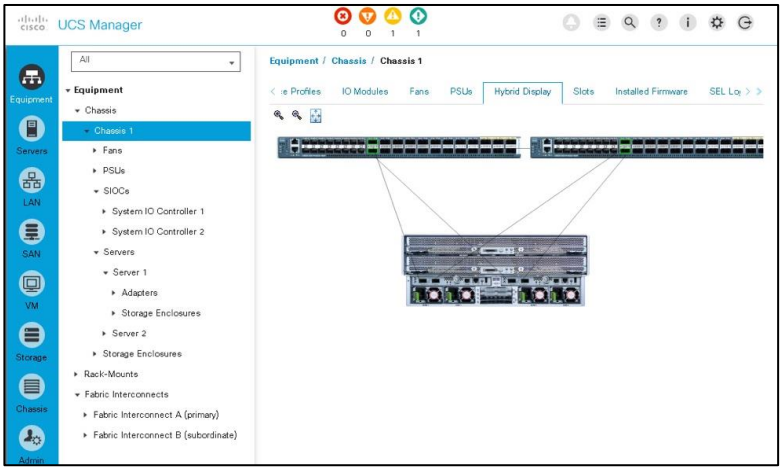
The Cisco UCS 6332 32-Port Fabric Interconnect is a 1-rack-unit (1RU) Gigabit Ethernet, and FCoE switch offering up to 2.56 Tbps throughput and up to 32 ports. The switch has 32 fixed 40-Gbps Ethernet and FCoE ports.

Both the Cisco UCS 6332UP 32-Port Fabric Interconnect and the Cisco UCS 6332 16-UP 40-Port Fabric Interconnect have ports that can be configured for the breakout feature that supports connectivity between 40 Gigabit Ethernet ports and 10 Gigabit Ethernet ports. This feature provides backward compatibility to existing hardware that supports 10 Gigabit Ethernet. A 40 Gigabit Ethernet port can be used as four 10 Gigabit Ethernet ports. Using a 40 Gigabit Ethernet SFP, these ports on a Cisco UCS 6300 Series Fabric Interconnect can connect to another fabric interconnect that has four 10 Gigabit Ethernet SFPs. The breakout feature can be configured on ports 1 to 12 and ports 15 to 26 on the Cisco UCS 6332UP fabric interconnect. Ports 17 to 34 on the Cisco UCS 6332 16-UP fabric interconnect support the breakout feature.

Cisco UCS Manager

Cisco UCS® Manager (Figure 5) provides unified, embedded management of all software and hardware components of the Cisco Unified Computing System™ (Cisco UCS) across multiple chassis, rack servers and thousands of virtual machines. It supports all Cisco UCS product models, including Cisco UCS B-Series Blade Servers, C-Series Rack Servers, and M-Series composable infrastructure and Cisco UCS Mini, as well as the associated storage resources and networks. Cisco UCS Manager is embedded on a pair of Cisco UCS 6300 or 6200 Series Fabric Interconnects using a clustered, active-standby configuration for high availability. The manager participates in server provisioning, device discovery, inventory, configuration, diagnostics, monitoring, fault detection, auditing, and statistics collection.

Figure 5 Cisco UCS Manager



An instance of Cisco UCS Manager with all Cisco UCS components managed by it forms a Cisco UCS domain, which can include up to 160 servers. In addition to provisioning Cisco UCS resources, this infrastructure management software provides a model-based foundation for streamlining the day-to-day processes of updating, monitoring, and managing computing resources, local storage, storage connections, and network connections. By enabling better automation of processes, Cisco UCS Manager allows IT organizations to achieve greater agility and scale in their infrastructure operations while reducing complexity and risk. The manager provides flexible role- and policy-based management using service profiles and templates.

Cisco UCS Manager manages Cisco UCS systems through an intuitive HTML 5 or Java user interface and a command-line interface (CLI). It can register with Cisco UCS Central Software in a multi-domain Cisco UCS environment, enabling centralized management of distributed systems scaling to thousands of servers. UCS Manager can be integrated with Cisco UCS Director to facilitate orchestration and to provide support for converged infrastructure and Infrastructure as a Service (IaaS).

The Cisco UCS XML API provides comprehensive access to all Cisco UCS Manager Functions. The API provides Cisco UCS system visibility to higher-level systems management tools from independent software vendors (ISVs) such as VMware, Microsoft, and Splunk as well as tools from BMC, CA, HP, IBM, and others. ISVs and in-house developers can use the XML API to enhance the value of the Cisco UCS platform according to their unique requirements. Cisco UCS PowerTool for UCS Manager and the Python Software Development Kit (SDK) help automate and manage configurations within UCS Manager.

Cisco Nexus C9332PQ Switch

The Cisco Nexus® 9000 Series Switches include both modular and fixed-port switches that are designed to overcome these challenges with a flexible, agile, low-cost, application-centric infrastructure.

Figure 6 Cisco 9332PQ

The Cisco Nexus 9300 platform consists of fixed-port switches designed for top-of-rack (ToR) and middle-of-row (MoR) deployment in data centers that support enterprise applications, service provider hosting, and cloud computing environments. They are Layer 2 and 3 nonblocking 10 and 40 Gigabit Ethernet switches with up to 2.56 terabits per second (Tbps) of internal bandwidth.

The Cisco Nexus 9332PQ Switch (Figure 6) is a 1-rack-unit (1RU) switch that supports 2.56 Tbps of bandwidth and over 720 million packets per second (mpps) across thirty-two 40-Gbps Enhanced QSFP+ ports

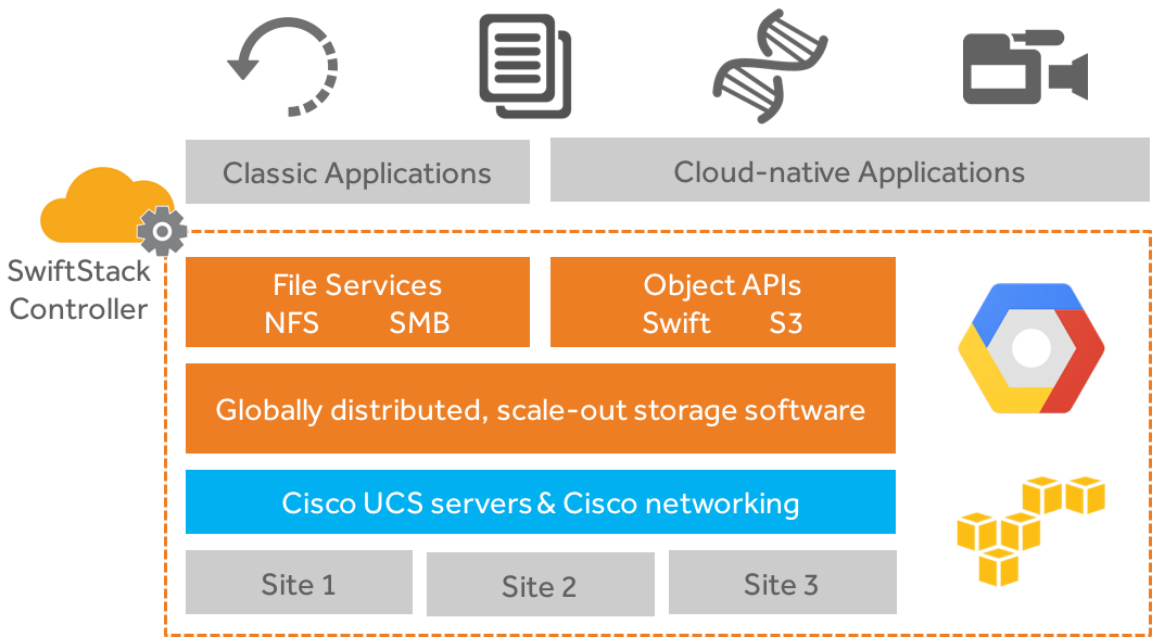
All the Cisco Nexus 9300 platform switches use dual- core 2.5-GHz x86 CPUs with 64-GB solid-state disk (SSD) drives and 16 GB of memory for enhanced network performance.

With the Cisco Nexus 9000 Series, organizations can quickly and easily upgrade existing data centers to carry 40 Gigabit Ethernet to the aggregation layer or to the spine (in a leaf-and-spine configuration) through advanced and cost-effective optics that enable the use of existing 10 Gigabit Ethernet fiber (a pair of multimode fiber strands).

Cisco provides two modes of operation for the Cisco Nexus 9000 Series. Organizations can use Cisco® NX-OS Software to deploy the Cisco Nexus 9000 Series in standard Cisco Nexus switch environments. Organizations also can use a hardware infrastructure that is ready to support Cisco Application Centric Infrastructure (Cisco ACI™) to take full advantage of an automated, policy-based, systems management approach.

SwiftStack Object Storage 5.x

With SwiftStack software running on Cisco UCS S-Series servers, you get hybrid cloud storage enabling freedom to move workloads between clouds with universal access to data across on-premises and public infrastructure. SwiftStack was built from day one to have the fundamental attributes of the cloud–like a single namespace across multiple geographic locations, policy-driven placement of data, and consumption-based pricing.



SwiftStack storage is optimized for unstructured data, which is growing at an ever-increasing rate inside most thriving enterprises. When media assets, scientific research data, and even backup archives live in a multi-tenant storage cloud, utilization of this valuable data increases while driving out unnecessary costs.

SwiftStack is a fully-distributed storage system that horizontally scales to hold your data today and tomorrow. It scales linearly, allowing you to add additional capacity and performance independently...whatever your applications need.

While scaling storage is typically complex, it's not with SwiftStack. No advanced configuration is required. It takes only a few simple commands to install software on a new UCS S3260 server and deploy it in the cluster. Load balancing capabilities are fully integrated, allowing applications to automatically take advantage of the distributed cluster.

Powered by OpenStack Swift at the core, with SwiftStack, you get to utilize what drives some of the largest storage clouds and leverage the power of a vibrant community. SwiftStack is the lead contributor to the Swift project that has over 220 additional contributors worldwide. Having an engine backed by this community and deployed in demanding customer environments makes SwiftStack the most proven, enterprise-grade object storage software available.

Key SwiftStack features for an active archive:

- Starts as small as 120TB, and scales to 100s of PB
- Spans multiple data centers while still presenting a single namespace
- Handles data according to defined policies that align to the needs of different applications
- Uses erasure coding and replicas in the same cluster to protect data
- Offers multi-tenant support with authentication via Active Directory, LDAP, and Keystone
- Supports file protocols (SMB, NFS) and object APIs (S3, Swift) simultaneously

- Automatically synchronizes to Google Cloud Storage and Amazon S3 with the Cloud Sync feature
- Encrypts data and metadata at rest
- Manages highly scalable storage infrastructure via centralized out-of-band controller
- Ensures all functionality touching data is open by leveraging an open-source core
- Optimizes TCO with pay-as-you-grow licensing with support and maintenance included

Red Hat Enterprise Linux 7.x

Red Hat® Enterprise Linux® is a high-performing operating system that has delivered outstanding value to IT environments for more than a decade. More than 90% of Fortune Global 500 companies use Red Hat products and solutions including Red Hat Enterprise Linux. As the world's most trusted IT platform, Red Hat Enterprise Linux has been deployed in mission-critical applications at global stock exchanges, financial institutions, leading telcos, and animation studios. It also powers the websites of some of the most recognizable global retail brands.

Red Hat Enterprise Linux:

- Delivers high performance, reliability, and security
- Is certified by the leading hardware and software vendors
- Scales from workstations, to servers, to mainframes
- Provides a consistent application environment across physical, virtual, and cloud deployments

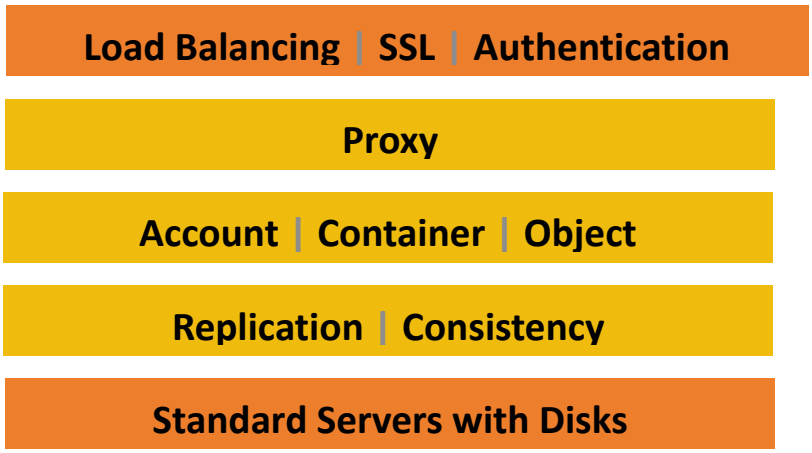
Designed to help organizations make a seamless transition to emerging datacenter models that include virtualization and cloud computing, Red Hat Enterprise Linux includes support for major hardware architectures, hypervisors, and cloud providers, making deployments across physical and different virtual environments predictable and secure. Enhanced tools and new capabilities in this release enable administrators to tailor the application environment to efficiently monitor and manage compute resources and security.

Solution Design

SwiftStack Core Storage Architecture

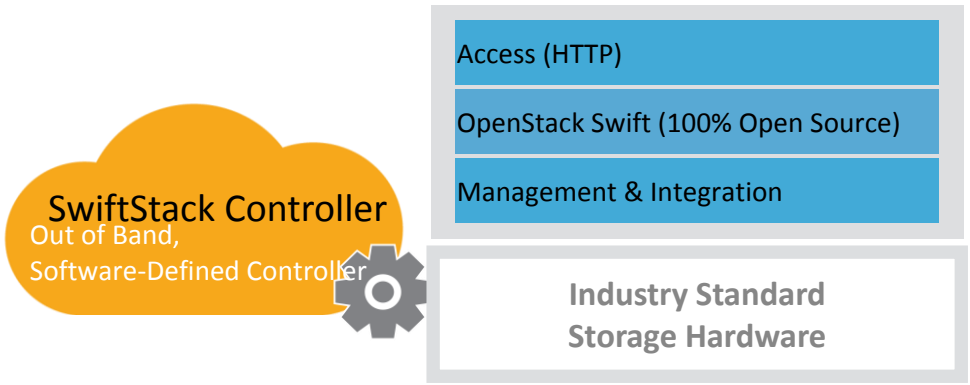
SwiftStack provides native Object API (S3 and Swift) to access the data stored in the SwiftStack Cluster. The design provides linear scalability, extreme durability with no single-point of Failure. It has no limitation on the hardware and uses any standard server and Linux system. SwiftStack clusters also supports multi-region data center architecture.

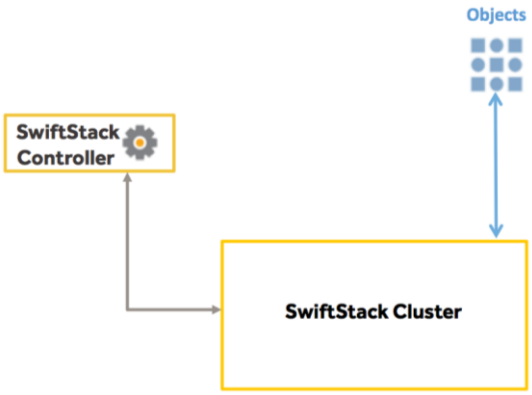
SwiftStack Nodes includes 4 different roles to handle different services in SwiftStack Cluster (Group of Swift Nodes) called as PACO – P: Proxy, A: Account, C: Container and O: Object. In most deployments, all four services are deployd and run on a single physical node.



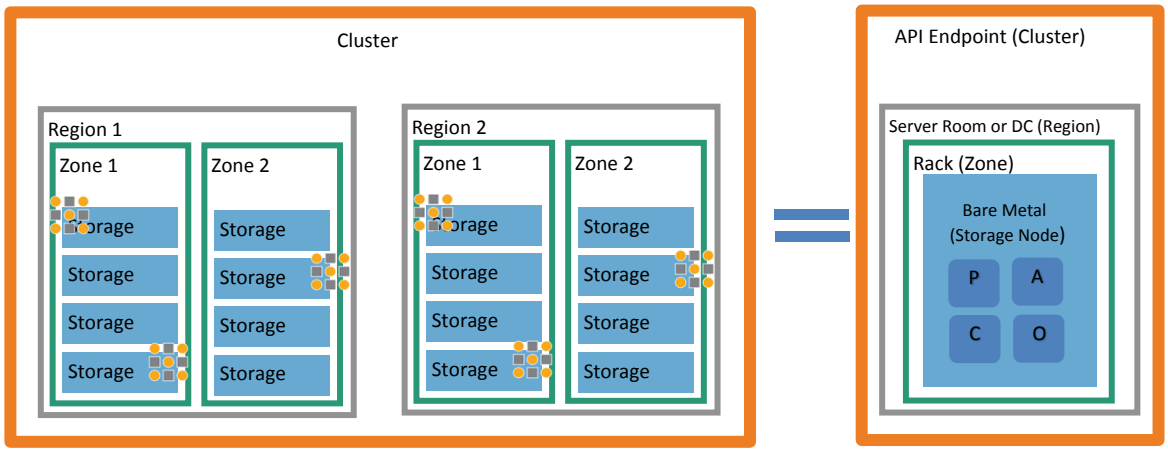
SwiftStack Architecture

SwiftStack solution is enterprise-grade object storage, with OpenStack Swift at the core. It has been deployed at 100s of companies with massive amounts of data stored. It includes two major components, SwiftStack Storage Nodes and the SwiftStack Controller which is an out-of-band management system that manages one of more SwiftStack storage clusters.

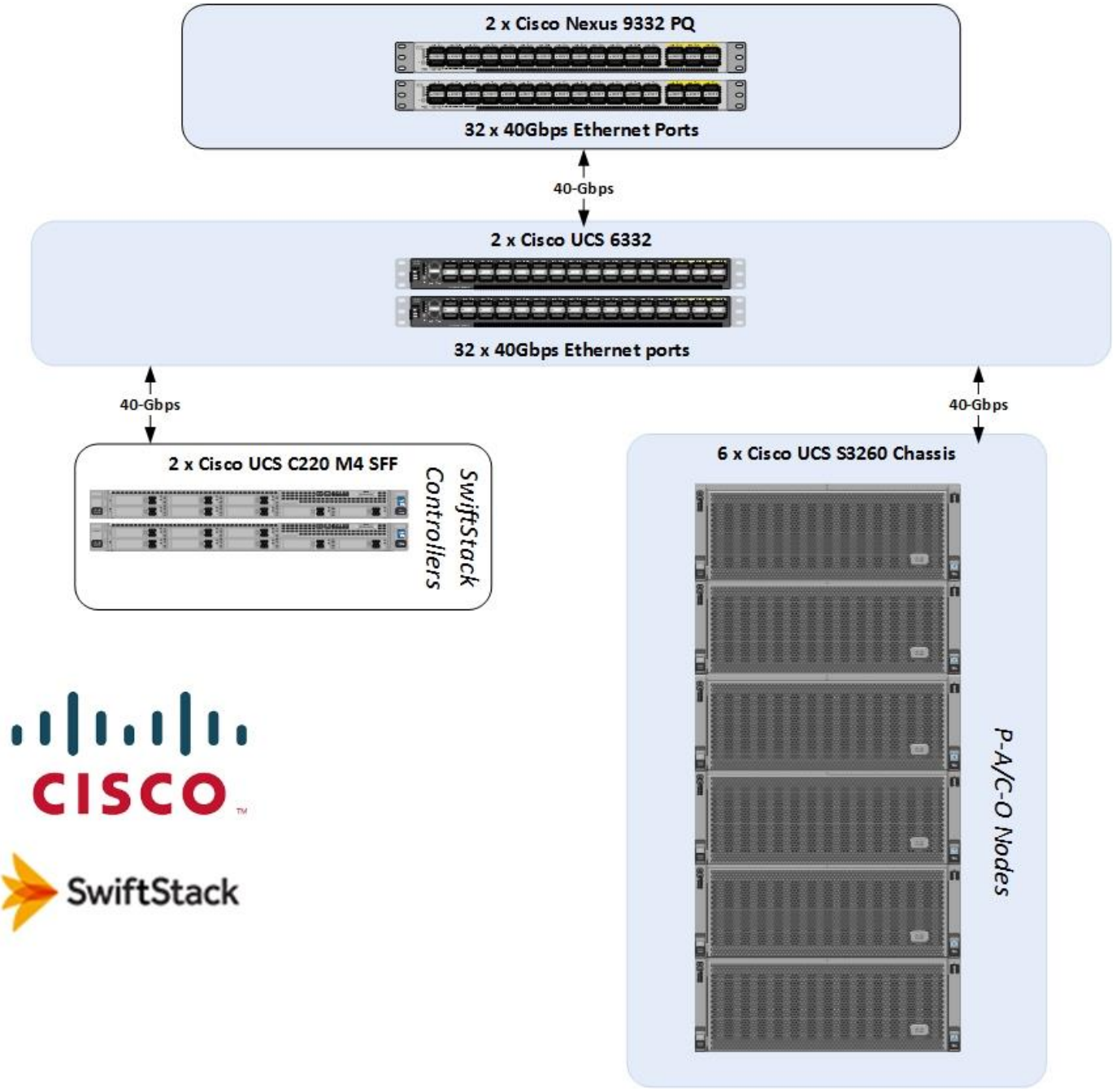




The SwiftStack Cluster Abstract Concept is illustrated below:



Logical Diagram



Physical Topology

Based on the logical flows, the following physical topology was built to validate the design.

Figure 7 illustrates the physical topology of this solution.

Figure 7 Physical Topology

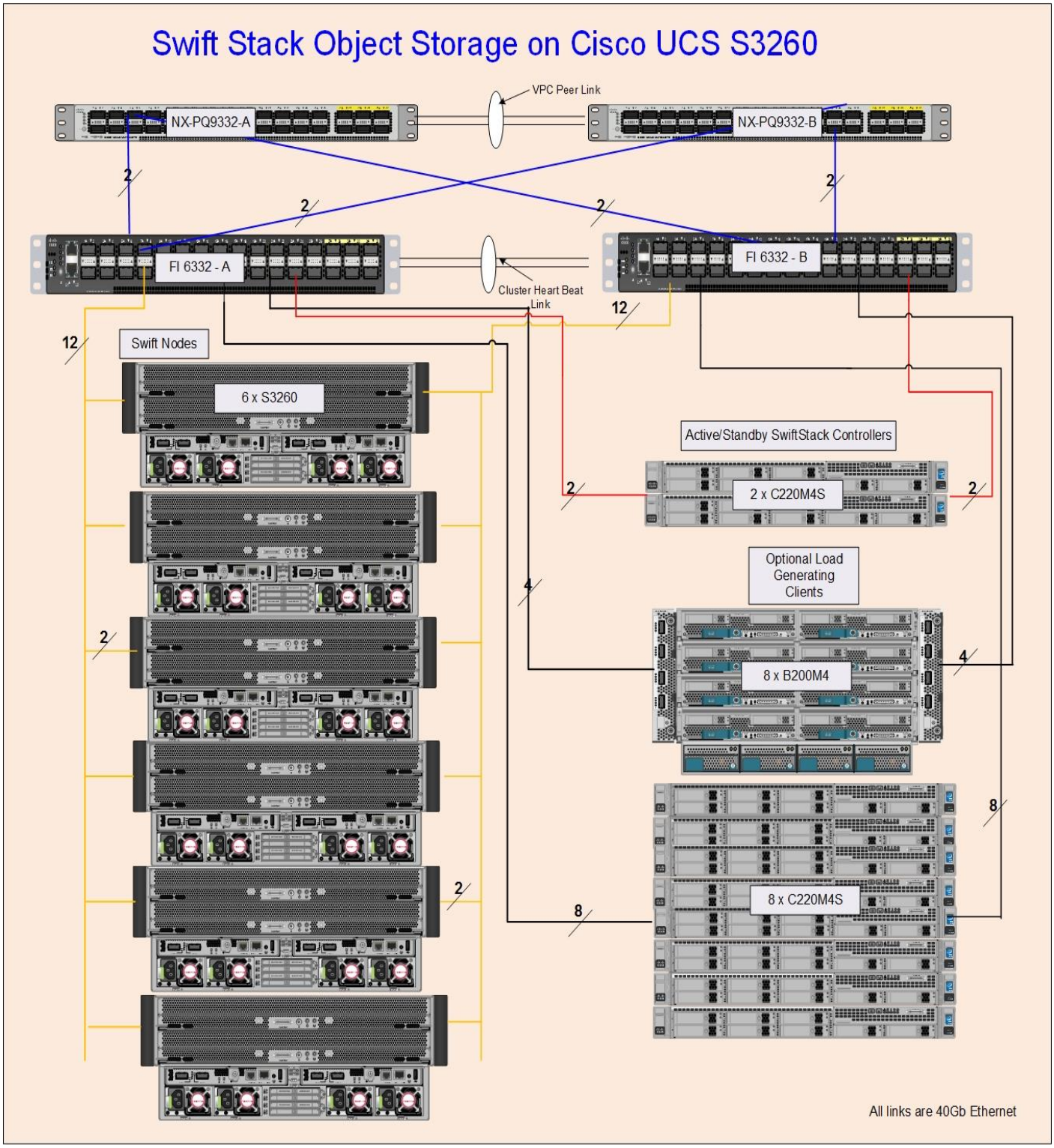


Figure 8 Cisco UCS S3260 Connectivity

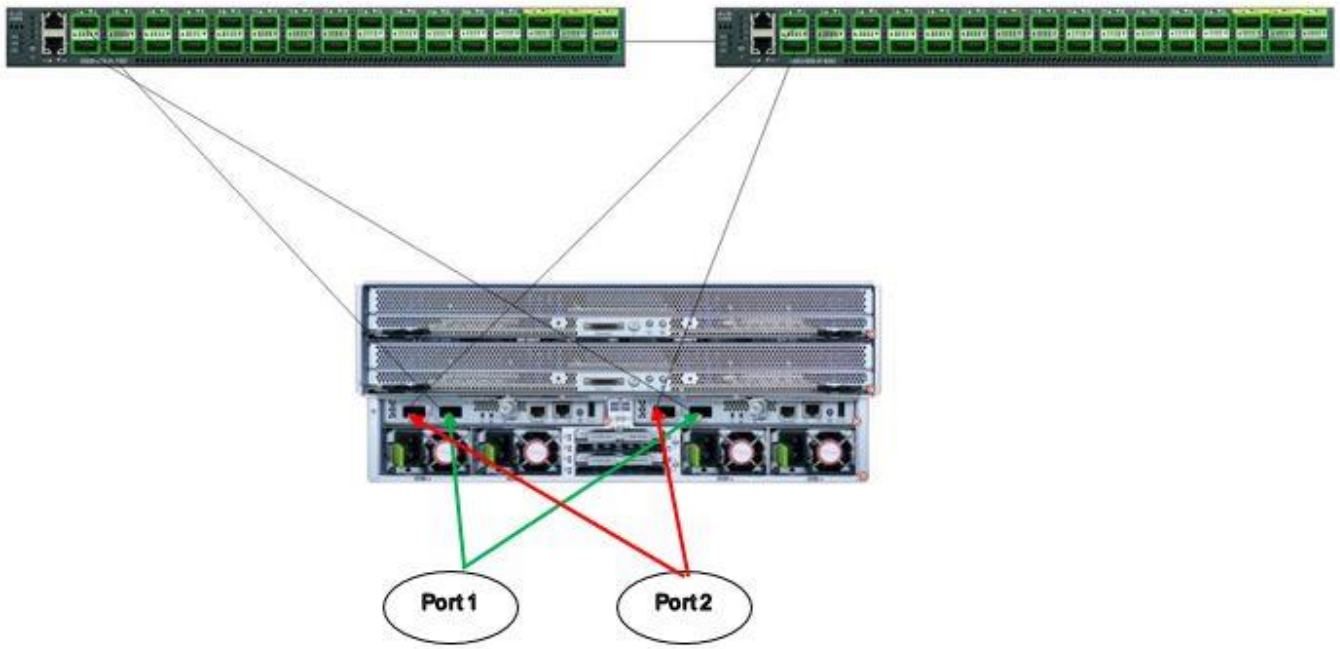
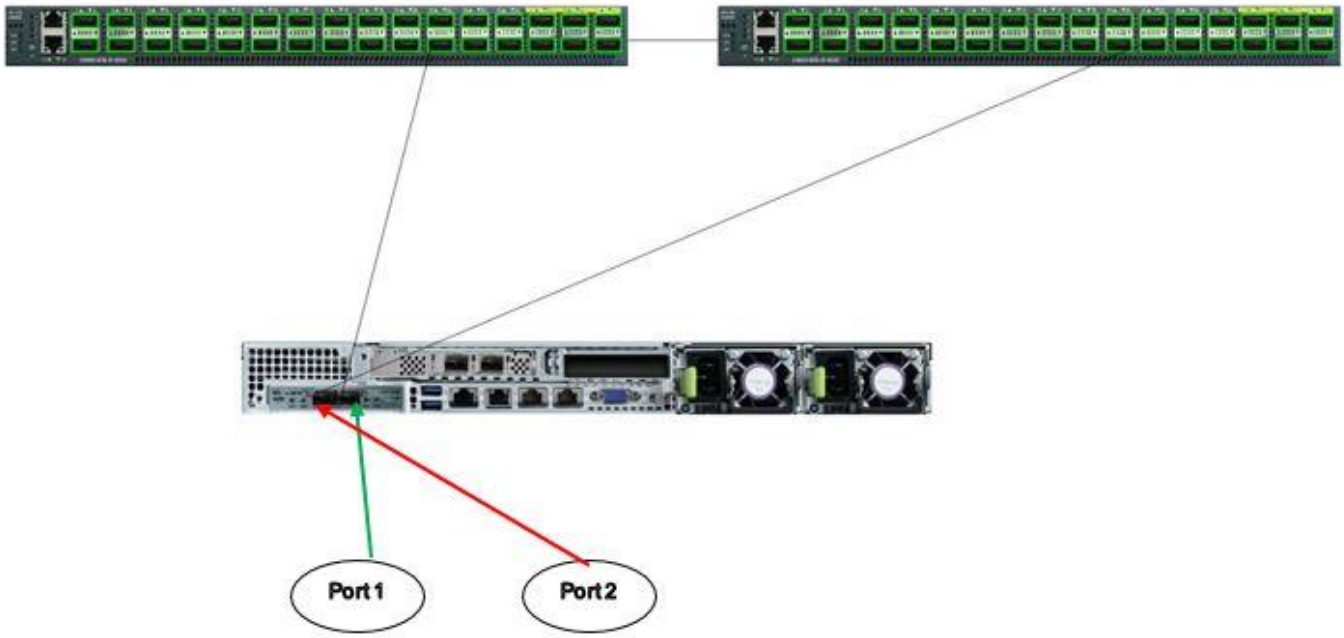


Figure 9 Cisco UCS C220 Connectivity



The configuration consists of 6 x S3260 chassis as PACO (Proxy, Account and Container and object services) nodes along with 2 x C220M4S as Active/Standby Controller nodes. The PACO and Controller nodes were connected to a pair of Fabric Interconnects and Cisco Nexus switches as shown in Figure 9.

The configuration, validation, testing and performance criteria were done on the following:

- 3 Full Chassis with 56 drives (54 HDD’s and 2 SSD’s) with Single Node on each Chassis.

- 3 Full Chassis with 56 drives (27 HDD's and 1 SSD's per node) with Dual Nodes on each Chassis.
- 6 Full Chassis with 56 drives (54 HDD's and 2 SSD's) with Single Node on each Chassis.
- 6 Full Chassis with 56 drives (27 HDD's and 1 SSD's per node) with Dual Nodes on each Chassis.

The system can be scaled up or down with the above configuration as long as the minimumrequirements of the software are met.

System Hardware and Software Specifications

Table 1 lists the Hardware and Software releases used for solution verification.

Table 1 Required Hardware Components

	Hardware	Quantity	Firmware Details
PACO Nodes	Storage (Chassis) UCS S3260	6	C3X60M4 3.0.3
Controllers	Cisco UCS C220M4S Rack Servers	2	C220M4 3.0.3
Fabric Interconnects	FI-6332	2	3.1(3)
Nexus Switches	Nexus – C9332PQ	2	7.0(3)I1(3)

Table 2 Software Specifications

	Software	Version
Operating System	Red Hat Enterprise Linux	7.3
Object Storage Software	SwiftStack	5.2.0.2

Bill of Materials

This section contains the Bill of Materials used in the configuration.

Component	Model	Quantity	Comments
SwiftStack PACO nodes	Cisco UCS S3260M4 Chassis	6	CPU – 2 x E5-2650 V4 per Node Memory – 16 x 8GB 2400 MHz DDR4 DIMM – total of 128G per Node S3260 SIOC with dual port 40G adapter and VIC 1380 per Node Raid Controller – Cisco MRAID 12 G SAS

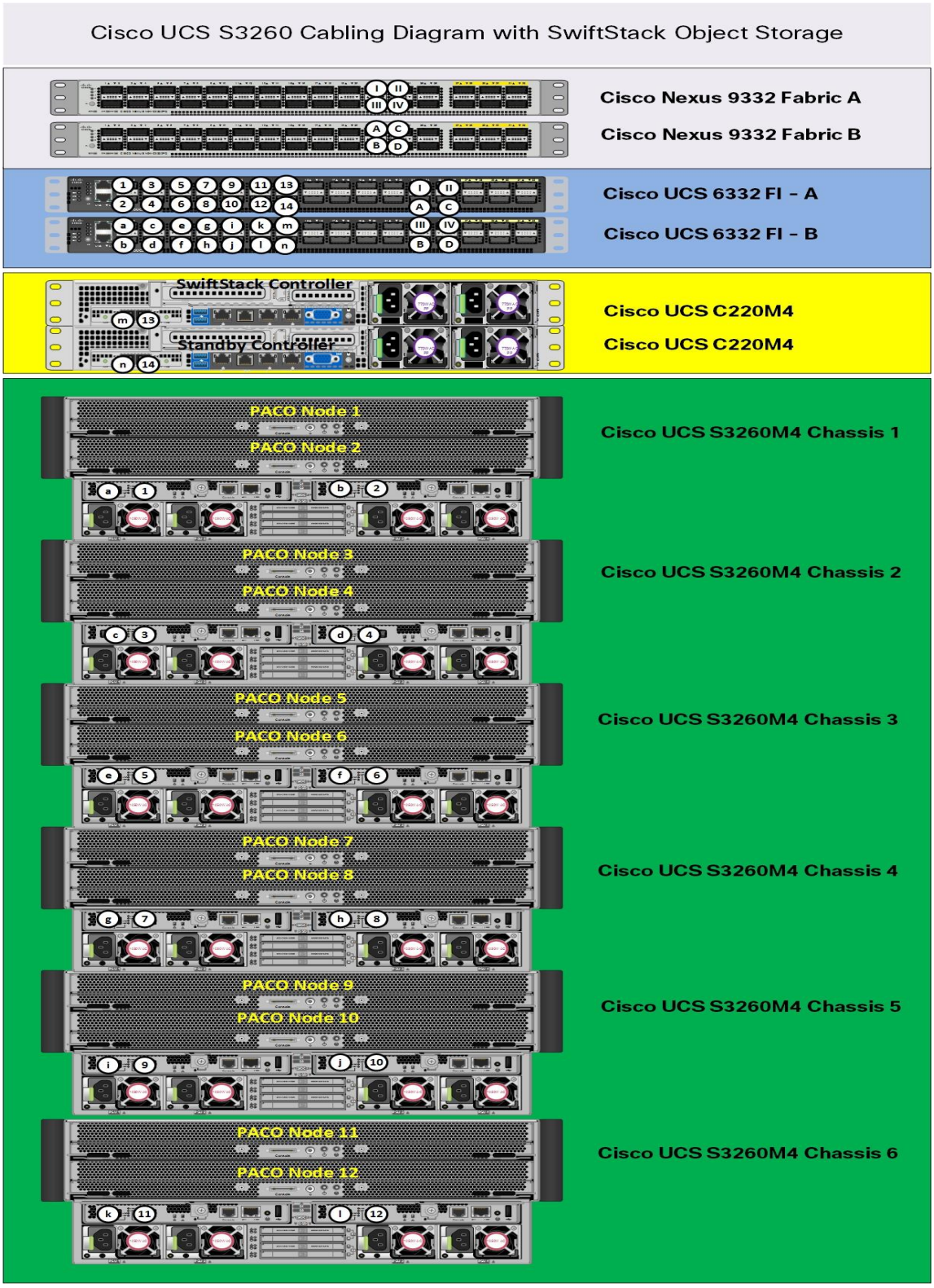
Component	Model	Quantity	Comments
			Controller
			Local Disks – 54x10TB HDD's and 2x400GB SSD's per Chassis. HDD's and SSD's distributed equally per node in Dual node configuration.
SwiftStack Controller Nodes	Cisco UCS C220M4 servers	2	CPU – 2 x E5-2680 V4 with 128GB Memory.
Fabric Interconnects	Cisco UCS 6332 UP Fabric Interconnects	2	
Switches	Cisco Nexus 9372PX Switches	2	

Performance tests have been evaluated on both single and dual node configurations.

Cabling Diagram

The cabling diagram is shown in Figure 10. The cables can be dropped between the identical numbers/letters as shown.

Figure 10 Cabling Diagram



The exact cabling for Cisco UCS S3260 Storage Server, C220 M4S and the Cisco UCS 6332 Fabric Interconnect is illustrated in Table below.

Table 3 Cabling details

Local Device	Local Port	Connection	Remote Device	Remote Port	Cable
Cisco Nexus 9332 Switch A	Eth1/21	40GbE	Cisco UCS Fabric Interconnect A	Eth1/23	40G Twin-Ax
	Eth1/22	40GbE	Cisco UCS Fabric Interconnect B	Eth1/23	40G Twin-Ax
	Eth1/23	40GbE	Cisco UCS Fabric Interconnect A	Eth 1/25	40G Twin-Ax
	Eth1/24	40GbE	Cisco UCS Fabric Interconnect B	Eth 1/25	40G Twin-Ax
	Eth1/30	40GbE	Nexus Peer Link	Eth1/30	40G Twin-Ax
	Eth1/32	40GbE	Nexus Peer Link	Eth1/32	40G Twin-Ax
	MGMT0	1GbE	Top of Rack (Management)	Any	1G RJ 45
Cisco Nexus 9332 Switch B	Eth1/21	40GbE	Cisco UCS Fabric Interconnect A	Eth1/24	40G Twin-Ax
	Eth1/22	40GbE	Cisco UCS Fabric Interconnect B	Eth1/24	40G Twin-Ax
	Eth1/23	40GbE	Cisco UCS Fabric Interconnect A	Eth 1/26	40G Twin-Ax
	Eth1/24	40GbE	Cisco UCS Fabric Interconnect B	Eth 1/26	40G Twin-Ax
	Eth1/30	40GbE	Nexus Peer Link	Eth1/30	40G Twin-Ax
	Eth1/32	40GbE	Nexus Peer Link	Eth1/32	40G Twin-Ax
	MGMT0	1GbE	Top of Rack (Management)	Any	1G RJ 45
Cisco UCS 6332 Fabric Interconnect A	Eth1/1	40GbE	S3260 Chassis 1 – Node1	Port 1	40G Twin-Ax
	Eth1/2	40GbE	S3260 Chassis 1 – Node2	Port 1	40G Twin-Ax
	Eth1/3	40GbE	S3260 Chassis 2 –	Port 1	40G

Local Device	Local Port	Connection	Remote Device	Remote Port	Cable
			Node1		Twin-Ax
	Eth1/4	40GbE	S3260 Chassis 2 – Node2	Port 1	40G Twin-Ax
	Eth1/5	40GbE	S3260 Chassis 3 – Node1	Port 1	40G Twin-Ax
	Eth1/6	40GbE	S3260 Chassis 3 – Node2	Port 1	40G Twin-Ax
	Eth1/7	40GbE	S3260 Chassis 4 – Node1	Port 1	40G Twin-Ax
	Eth 1/8	40GbE	S3260 Chassis 4 – Node2	Port 1	40G Twin-Ax
	Eth 1/9	40GbE	S3260 Chassis 5 – Node1	Port 1	40G Twin-Ax
	Eth 1/10	40GbE	S3260 Chassis 5 – Node2	Port 1	40G Twin-Ax
	Eth 1/11	40GbE	S3260 Chassis 6 – Node1	Port 1	40G Twin-Ax
	Eth 1/12	40GbE	S3260 Chassis 6 – Node2	Port 1	40G Twin-Ax
	Eth 1/13	40GbE	C220M4S –Active Controller	Port 1	40G Twin-Ax
	Eth 1/14	40GbE	C220M4S – Standby Controller	Port 1	40G Twin-Ax
	Eth 1/15-22 used for Clients				
	Eth1/23	40GbE	Nexus 9332A	Eth 1/21	40G Twin-Ax
	Eth1/24	40GbE	Nexus 9332B	Eth 1/21	40G Twin-Ax
	Eth1/25	40GbE	Nexus 9332 A	Eth 1/23	40G Twin-Ax
	Eth1/26	40GbE	Nexus 9332 A	Eth 1/23	40G Twin-Ax
	Eth 1/27 and above either used as Clients or left as spares				

Local Device	Local Port	Connection	Remote Device	Remote Port	Cable
	MGMT0	1GbE	Top of RACK (Management)	Any	1G RJ 45
	L1	1GbE	UCS 6332 Fabric Interconnect B	L1	1G RJ 45
	L2	1GbE	UCS 6332 Fabric Interconnect B	L2	1G RJ 45
Cisco UCS 6332 Fabric Interconnect B	Eth1/1	40GbE	S3260 Chassis 1 – Node1	Port 2	40G Twin-Ax
	Eth1/2	40GbE	S3260 Chassis 1 – Node2	Port 2	40G Twin-Ax
	Eth1/3	40GbE	S3260 Chassis 2 – Node1	Port 2	40G Twin-Ax
	Eth1/4	40GbE	S3260 Chassis 2 – Node2	Port 2	40G Twin-Ax
	Eth1/5	40GbE	S3260 Chassis 3 – Node1	Port 2	40G Twin-Ax
	Eth1/6	40GbE	S3260 Chassis 3 – Node2	Port 2	40G Twin-Ax
	Eth1/7	40GbE	S3260 Chassis 4 – Node1	Port 2	40G Twin-Ax
	Eth 1/8	40GbE	S3260 Chassis 4 – Node2	Port 2	40G Twin-Ax
	Eth 1/9	40GbE	S3260 Chassis 5 – Node1	Port 2	40G Twin-Ax
	Eth 1/10	40GbE	S3260 Chassis 5 – Node2	Port 2	40G Twin-Ax
	Eth 1/11	40GbE	S3260 Chassis 6 – Node1	Port 2	40G Twin-Ax
	Eth 1/12	40GbE	S3260 Chassis 6 – Node2	Port 2	40G Twin-Ax
	Eth 1/13	40GbE	C220M4S –Active Controller	Port 2	40G Twin-Ax
	Eth 1/14	40GbE	C220M4S – Standby Controller	Port 2	40G Twin-Ax

Local Device	Local Port	Connection	Remote Device	Remote Port	Cable
	Eth 1/15-22 used for Clients				
	Eth1/23	40GbE	Nexus 9332A	Eth 1/22	40G Twin-Ax
	Eth1/24	40GbE	Nexus 9332B	Eth 1/22	40G Twin-Ax
	Eth1/25	40GbE	Nexus 9332 A	Eth 1/24	40G Twin-Ax
	Eth1/26	40GbE	Nexus 9332 A	Eth 1/24	40G Twin-Ax
	Eth 1/27 and above either used as Clients or left as spares				
	MGMT0	1GbE	Top of RACK (Management)	Any	1G RJ 45
	L1	1GbE	UCS 6332 Fabric Interconnect B	L1	1G RJ 45
	L2	1GbE	UCS 6332 Fabric Interconnect B	L2	1G RJ 45

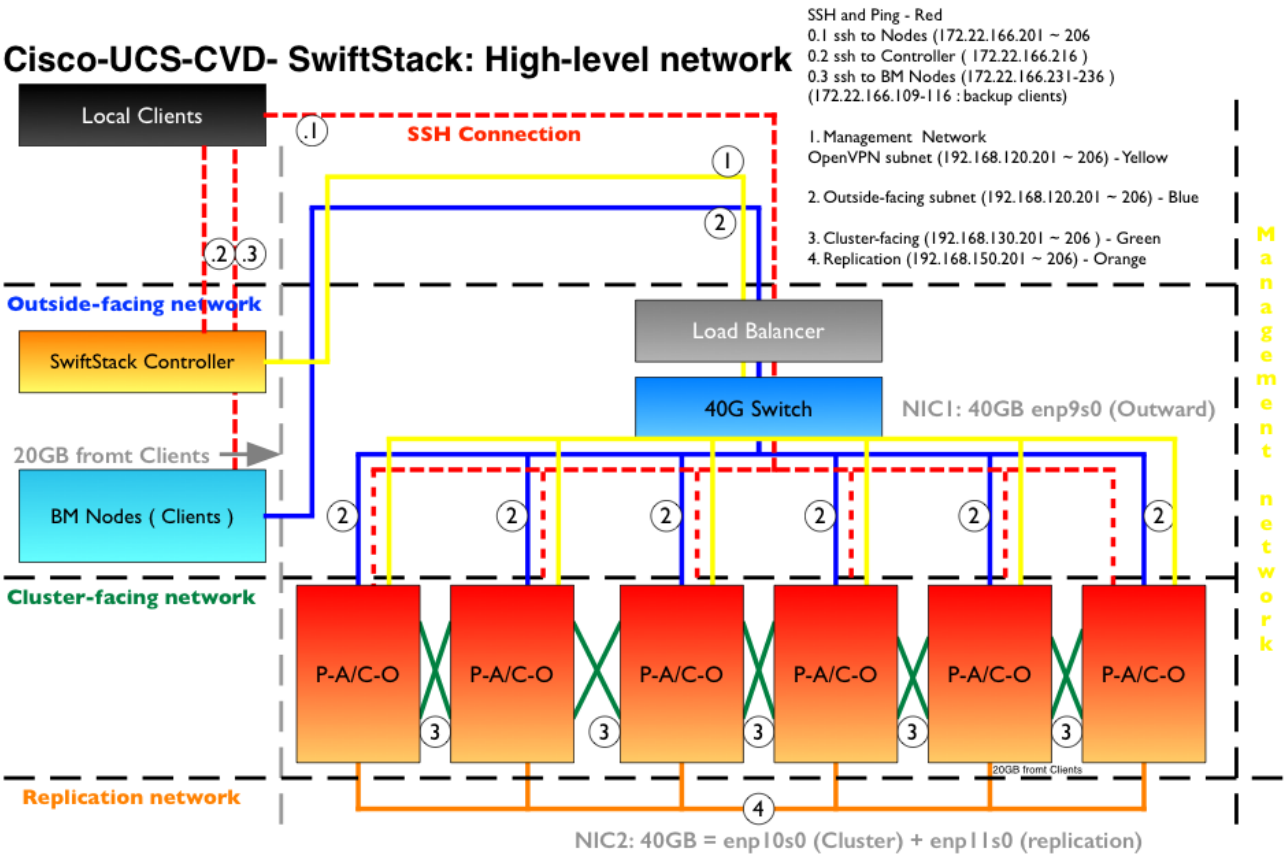
Network Design and Architecture

The following network architecture has been evaluated on the test bed. Each server node has 2x40Gbps ports. Taking the advantage of Cisco's VIC architecture multiple network interfaces are detailed below.

Table 4 Interface Configuration on each Server Node

Interface	Purpose	Physical Port on Adapter	Network Capacity	Network
eth0	PXE Interface for boot (optional)	Port 1	40Gbps	192.168.xxx.xxx
eth1	External/OS interface	Port 1		172.xxx.xxx.xxx
eth2	Client/Outward facing interface	Port 1		192.168.120.xxx
eth3	Cluster network interface	Port 2	40Gbps	192.168.130.xxx
eth4	Replication network interface	Port 2		192.168.150.xxx

Figure 11 Logical Network Flow Diagram



SwiftStack controllers are not in the data path. Configuration of External/Outward facing interface on Controller is optional.

Cisco UCS Hardware Configuration

This section details the deployment of hardware used in this solution. Please refer to the cabling diagram as depicted above and power on the hardware before proceeding further.

Configure Nexus 9332PQ Switch A and B

Both Cisco UCS Fabric Interconnect A and B are connected to two Cisco Nexus 9332PQ switches for connectivity to Upstream Network. The following sections describe the setup of both Cisco Nexus 9332PQ switches.

1. Connect the console port to the Nexus 9332 PQ switch designated for Fabric A:

```
---- Basic System Configuration Dialog VDC: 1 ----
This setup utility will guide you through the basic configuration of the system.
Setup configures only enough connectivity for management of the system.
*Note: setup is mainly used for configuring the system initially, when no
configuration is present. So setup always assumes system defaults and not the
current system configuration values.
Press Enter at anytime to skip a dialog. Use ctrl-c at anytime to skip the
remaining dialogs.
Would you like to enter the basic configuration dialog (yes/no): yes
Do you want to enforce secure password standard (yes/no) [y]:
Create another login account (yes/no) [n]:
Configure read-only SNMP community string (yes/no) [n]:
Configure read-write SNMP community string (yes/no) [n]:
Enter the switch name :N9K-40G-OPS-FAB-A
Continue with Out-of-band (mgmt0) management configuration? (yes/no) [y]:
             Mgmt0 IPv4 address : 192.168.10.8
 Mgmt0 IPv4 netmask : 255.255.255.0
            Configure the default gateway? (yes/no) [y]:
            IPv4 address of the default gateway : 192.168.10.1
            Configure advanced IP options? (yes/no) [n]:
            Enable the telnet service? (yes/no) [n]:
            Enable the ssh service? (yes/no) [y]:
            Type of ssh key you would like to generate (dsa/rsa) [rsa]:
            Number of rsa key bits <1024-2048> [2048]:
            Configure the ntp server? (yes/no) [n]: y
            NTP server IPv4 address : <<ntp_server_ip>>

            Configure CoPP system profile (strict/moderate/lenient/dense/skip)
[strict]:
The following configuration will be applied:
   password strength-check
   switchname N9K-40G-OPS-FAB-A
   vrf context management
   ip route 0.0.0.0/0 192.168.10.8
   exit
   no feature telnet
   ssh key rsa 2048 force
    feature ssh
    ntp server <<var_global_ntp_server_ip>>
    copp profile strict
     interface mgmt0
```

```
ip address 192.168.10.8 255.255.255.0
no shutdown
  Would you like to edit the configuration? (yes/no) [n]: Enter
  Use this configuration and save it? (yes/no) [y]: Enter
  [########################################] 100%
  Copy complete.
```

Login into the switch and verify credentials.

Repeat the this procedure for the Nexus Switch B.

Enable Features on Nexus 9332PQ Switch A and B

To enable the features UDLD, VLAN, HSRP, LACP, VPC, and Jumbo Frames, connect to the management interface via ssh on both switches and complete the following steps on both Switch A and B:

Switch A

```
N9k-Fab-A# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
N9k-Fab-A(config)# feature udld
N9k-Fab-A(config)# feature interface-vlan
N9k-Fab-A(config)# feature hsrp
N9k-Fab-A(config)# feature lacp
N9k-Fab-A(config)# feature vpc
N9k-Fab-A(config)# system jumbomtu 9216
N9k-Fab-A(config)# exit
N9k-Fab-A(config)# copy running-config startup-config
```

Switch B

```
N9k-Fab-B# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
N9k-Fab-B(config)# feature udld
N9k-Fab-B(config)# feature interface-vlan
N9k-Fab-B(config)# feature hsrp
N9k-Fab-B(config)# feature lacp
N9k-Fab-B(config)# feature vpc
N9k-Fab-B(config)# system jumbomtu 9216
N9k-Fab-B(config)# exit
N9k-Fab-B(config)# copy running-config startup-config
```

Configuring VLANs on Nexus 9332PQ Switch A and B

To configure the VLANs like External, Outward Facing, Cluster and Replication Networks, follow the steps below.

Switch A

From the command line, enter the following and change the VLAN and Ethernet ports per your setup.

```
vlan 10
name UCSM-VLAN
no shut
exit
vlan 120
name Swift-Client-Network
no shut
```

```
exit
vlan 130
name Swift-Cluster-Network
no shut
exit
vlan 150
name Swift-Replication-Network
no shut
exit
vlan 166
name External-Network
no shut
exit

interface vlan 1
exit

interface Vlan10
  description UCSM VLAN
  no shutdown
  no ip redirects
  ip address 192.168.10.253/24
  no ipv6 redirects
  hsrp version 2
  hsrp 100
    preempt
    priority 110
    ip 192.168.10.1
  exit
exit
interface Vlan120
  description swift-client
  no shutdown
  no ip redirects
  ip address 192.168.120.253/24
  no ipv6 redirects
  hsrp version 2
  hsrp 120
    preempt
    priority 120
    ip 192.168.120.1
  exit
exit
interface Vlan130
  description swift-cluster
  no shutdown
  no ip redirects
  ip address 192.168.130.253/24
  no ipv6 redirects
  hsrp version 2
  hsrp 130
    preempt
    priority 130
    ip 192.168.130.1
  exit
exit
interface Vlan150
  description swift-replication
```

```
  no shutdown
  no ip redirects
  ip address 192.168.150.253/24
  no ipv6 redirects
  hsrp version 2
  hsrp 150
    preempt
    priority 10
    ip 192.168.150.1
  exit
exit
interface Vlan166
  description External_Network
  no shutdown
  no ip redirects
  ip address 172.22.166.249/24
  no ipv6 redirects
exit
interface Ethernet1/1
  switchport mode trunk
  switchport trunk allowed vlan 10,100,166
  no shutdown
exit
```

copy running-config startup-config

Switch B

```
vlan 10
  name UCSM-VLAN
  no shut
exit
vlan 120
  name Swift-Client-Network
  no shut
exit
vlan 130
  name Swift-Cluster-Network
  no shut
exit
vlan 150
  name Swift-Replication-Network
  no shut
exit
vlan 166
  name External-Network
  no shut
exit


interface Vlan1
no shut
exit

interface Vlan10
  description UCSM vlan
  no shutdown
  no ip redirects
```

```
  ip address 192.168.10.254/24
  no ipv6 redirects
  hsrp version 2
  hsrp 100
    preempt
    priority 110
    ip 192.168.10.1
  exit
exit

interface Vlan120
  description swift-client
  no shutdown
  no ip redirects
  ip address 192.168.120.254/24
  no ipv6 redirects
  hsrp version 2
  hsrp 120
    preempt
    priority 120
    ip 192.168.120.1
  exit
exit

interface Vlan130
  description swift-cluster
  no shutdown
  no ip redirects
  ip address 192.168.130.254/24
  no ipv6 redirects
  hsrp version 2
  hsrp 130
    preempt
    priority 130
    ip 192.168.130.1
  exit
exit

interface Vlan150
  description swift-replication
  no shutdown
  no ip redirects
  ip address 192.168.150.254/24
  no ipv6 redirects
  hsrp version 2
  hsrp 150
    preempt
    priority 10
    ip 192.168.150.1
  exit
exit

interface Vlan166
  description External_Network
  no shutdown
  no ip redirects
  no ipv6 redirects
exit
```

Configure vPC and Port Channels on Nexus C9332PQ Switch A and B

To enable the vPC and Port Channels on both Switch A and B, complete the following steps:

VPC and Port Channel for Peer Link

```
N9K-40G-OPS-FAB-B# conf terminal
Enter configuration commands, one per line. End with CNTL/Z.
N9K-40G-OPS-FAB-B(config)# vpc domain 101
N9K-40G-OPS-FAB-B(config-vpc-domain)#peer-keepalive destination 192.168.10.8
Note:
 --------:: Management VRF will be used as the default VRF ::--------
N9K-40G-OPS-FAB-B(config-vpc-domain)# peer-gateway
N9K-40G-OPS-FAB-B(config-vpc-domain)# exit
N9K-40G-OPS-FAB-B(config)# interface port-channel 1
N9K-40G-OPS-FAB-B(config-if)# switchport mode trunk
N9K-40G-OPS-FAB-B(config-if)# spanning-tree port type network
N9K-40G-OPS-FAB-B(config-if)# vpc peer-link
N9K-40G-OPS-FAB-B(config-if)# exit

N9K-40G-OPS-FAB-B(config)# interface ethernet 1/31-32
N9K-40G-OPS-FAB-B(config-if)# switchport mode trunk
N9K-40G-OPS-FAB-B(config-if)# switchport trunk allowed vlan 10,100,166
N9K-40G-OPS-FAB-B(config-if)# speed 40000
N9K-40G-OPS-FAB-B(config-if)# channel-group 1 mode active
N9K-40G-OPS-FAB-B(config-if)# exit
N9K-40G-OPS-FAB-B# copy running-config startup-config
[########################################] 100%
Copy complete.
```

Repeat these steps on the second switch.

VPC and Port Channel for Fabric Interconnects

```
N9K-40G-OPS-FAB-A# conf terminal
N9K-40G-OPS-FAB-A(config)# interface port-channel 20
N9K-40G-OPS-FAB-A(config-if)#  switchport mode trunk
N9K-40G-OPS-FAB-A(config-if)# switchport trunk allowed vlan 1,10,120,130,150,166
N9K-40G-OPS-FAB-A(config-if)# spanning-tree port type edge trunk
Warning: Edge port type (portfast) should only be enabled on ports connected to a
single host. Connecting hubs, concentrators, switches, bridges, etc... to this
 interface  when edge port type (portfast) is enabled, can cause temporary
bridging loops.
 Use with CAUTION
N9K-40G-OPS-FAB-A(config-if)# mtu 9216
N9K-40G-OPS-FAB-A(config-if)# vpc 20
N9K-40G-OPS-FAB-A(config-if)# no shutdown
N9K-40G-OPS-FAB-A(config-if)# exit
N9K-40G-OPS-FAB-A(config)# interface port-channel 21
N9K-40G-OPS-FAB-A(config-if)#  switchport mode trunk
N9K-40G-OPS-FAB-A(config-if)# switchport trunk allowed vlan 1,10,120,130,150,166
N9K-40G-OPS-FAB-A(config-if)# spanning-tree port type edge trunk
Warning: Edge port type (portfast) should only be enabled on ports connected to a
single host. Connecting hubs, concentrators, switches, bridges, etc... to this
```

```
 interface  when edge port type (portfast) is enabled, can cause temporary
bridging loops.
 Use with CAUTION
N9K-40G-OPS-FAB-A(config-if)# mtu 9216
N9K-40G-OPS-FAB-A(config-if)# vpc 21
N9K-40G-OPS-FAB-A(config-if)# no shutdown
N9K-40G-OPS-FAB-A(config-if)# exit
```

On Switch A configure the ethernet interfaces for VPC:

```
interface Ethernet1/21
  description UCS Fabric A port 23
  switchport mode trunk
  switchport trunk allowed vlan 1,10,120,130,150,166
  mtu 9216
  channel-group 20 mode active

interface Ethernet1/22
  description UCS Fabric B port 23
  switchport mode trunk
  switchport trunk allowed vlan 1,10,120,130,150,166
  mtu 9216
  channel-group 21 mode active

interface Ethernet1/23
  description UCS Fabric A port 25
  switchport mode trunk
  switchport trunk allowed vlan 1,10,120,130,150,166
  mtu 9216
  channel-group 20 mode active

interface Ethernet1/24
  description UCS Fabric B port 25
  switchport mode trunk
  switchport trunk allowed vlan 1,10,120,130,150,166
  mtu 9216
  channel-group 21 mode active
```

On Switch B configure the ethernet interfaces for VPC:

```
interface Ethernet1/21
  description UCS Fabric A port 24
  switchport mode trunk
  switchport trunk allowed vlan 1,10,120,130,150,166
  mtu 9216
  channel-group 20 mode active

interface Ethernet1/22
  description UCS Fabric B port 24
  switchport mode trunk
  switchport trunk allowed vlan 1,10,120,130,150,166
  mtu 9216
  channel-group 21 mode active

interface Ethernet1/23
  description UCS Fabric A port 26
  switchport mode trunk
  switchport trunk allowed vlan 1,10,120,130,150,166
```

```
  mtu 9216
  channel-group 20 mode active

interface Ethernet1/24
  description UCS Fabric B port 26
  switchport mode trunk
  switchport trunk allowed vlan 1,10,120,130,150,166
  mtu 9216
  channel-group 21 mode active
```

Health Checks of Nexus C9332PQ Configuration for Switch A and B

```
N9K-40G-OPS-FAB-B# show vpc br
Legend:
                (*) - local vPC is down, forwarding via vPC peer-link

vPC domain id                     : 101
Peer status                       : peer adjacency formed ok
vPC keep-alive status             : peer is alive
Configuration consistency status  : success
Per-vlan consistency status       : success
Type-2 consistency status         : success
vPC role                          : primary, operational secondary
Number of vPCs configured         : 2
Peer Gateway                      : Enabled
Dual-active excluded VLANs        : -
Graceful Consistency Check        : Enabled
Auto-recovery status              : Disabled
Delay-restore status              : Timer is off.(timeout = 30s)
Delay-restore SVI status          : Timer is off.(timeout = 10s)

vPC Peer-link status
---------------------------------------------------------------------
id   Port   Status Active vlans
--   ----   ------ --------------------------------------------------
1    Po1    up     1,10,120,130,150,166

vPC status
----------------------------------------------------------------------------
id   Port   Status Consistency Reason                     Active vlans
--   ----   ------ ----------- ------                     ------------
20   Po20   up     success     success                    1,10,120,13
                                                         0,150,166
21   Po21   up     success     success                    1,10,100,12
                                                         0,130,150,1
                                                         66
N9K-40G-OPS-FAB-B#


N9K-40G-OPS-FAB-A# show vpc br
Legend:
                (*) - local vPC is down, forwarding via vPC peer-link

vPC domain id                     : 101
Peer status                       : peer adjacency formed ok
vPC keep-alive status             : peer is alive
Configuration consistency status  : success
```

```
Per-vlan consistency status       : success
Type-2 consistency status         : success
vPC role                          : secondary, operational primary
Number of vPCs configured         : 2
Peer Gateway                      : Enabled
Dual-active excluded VLANs        : -
Graceful Consistency Check        : Enabled
Auto-recovery status              : Disabled
Delay-restore status              : Timer is off.(timeout = 30s)
Delay-restore SVI status          : Timer is off.(timeout = 10s)

vPC Peer-link status
---------------------------------------------------------------------
id   Port   Status Active vlans
--   ----   ------ --------------------------------------------------
1    Po1    up     1,10,120,130,150,166

vPC status
----------------------------------------------------------------------------
id   Port   Status Consistency Reason                     Active vlans
--   ----   ------ ----------- ------                     ------------
20   Po20   up     success     success                    1,10,120,13
                                                         0,150,166
21   Po21   up     success     success                    1,10,100,12
                                                         0,130,150,1
                                                         66
```

Ping Default Gateway and any IP's to make sure that Fabric Interconnects and servers can connect later.

```
N9K-40G-OPS-FAB-A# ping 172.22.166.1
PING 172.22.166.1 (172.22.166.1): 56 data bytes
64 bytes from 172.22.166.1: icmp_seq=0 ttl=254 time=0.78 ms
64 bytes from 172.22.166.1: icmp_seq=1 ttl=254 time=0.638 ms
64 bytes from 172.22.166.1: icmp_seq=2 ttl=254 time=0.553 ms
64 bytes from 172.22.166.1: icmp_seq=3 ttl=254 time=0.782 ms
64 bytes from 172.22.166.1: icmp_seq=4 ttl=254 time=0.687 ms

--- 172.22.166.1 ping statistics ---
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min/avg/max = 0.553/0.687/0.782 ms
N9K-40G-OPS-FAB-A# ping 172.22.166.204
PING 172.22.166.204 (172.22.166.204): 56 data bytes
64 bytes from 172.22.166.204: icmp_seq=0 ttl=63 time=0.624 ms
64 bytes from 172.22.166.204: icmp_seq=1 ttl=63 time=0.425 ms
64 bytes from 172.22.166.204: icmp_seq=2 ttl=63 time=0.409 ms
64 bytes from 172.22.166.204: icmp_seq=3 ttl=63 time=0.366 ms
64 bytes from 172.22.166.204: icmp_seq=4 ttl=63 time=0.478 ms

--- 172.22.166.204 ping statistics ---
5 packets transmitted, 5 packets received, 0.00% packet loss
round-trip min/avg/max = 0.366/0.46/0.624 ms
```

Configure Cisco UCS Fabric Interconnects

Configure the Fabric Interconnects after the cabling is complete. Hook up the console port on the Fabrics, and complete the following steps:

Please replace the appropriate addresses for your setup.

Cisco UCS 6332FI Switch A

Connect the console port to the UCS 6332 Fabric Interconnect switch designated for Fabric A:

```
Enter the configuration method: console
Enter the setup mode; setup newly or restore from backup.(setup/restore)? setup
You have chosen to setup a new fabric interconnect? Continue? (y/n): y
Enforce strong passwords? (y/n) [y]: y
Enter the password for "admin": <password>
Enter the same password for "admin": <password>
Is this fabric interconnect part of a cluster (select 'no' for standalone)?
(yes/no) [n]:y
Which switch fabric (A|B): A
Enter the system name: UCS-SwiftStack-Fabric
Physical switch Mgmt0 IPv4 address: 192.168.10.21
Physical switch Mgmt0 IPv4 netmask: 255.255.255.0
IPv4 address of the default gateway: 192.168.10.1
Cluster IPv4 address: 10.22.100.20
Configure DNS Server IPv4 address? (yes/no) [no]: y
DNS IPv4 address: <<var_nameserver_ip>>
Configure the default domain name? y
Default domain name: <<var_dns_domain_name>>
Join centralized management environment (UCS Central)? (yes/no) [n]: Press Enter
You will be prompted to review the settings.
If they are correct, answer yes to apply and save the configuration. Wait for the
login prompt to make sure that the configuration has been saved.
```

Cisco UCS 6332FI Switch B

Connect the console port to Peer UCS 6332 Fabric Interconnect switch designated for Fabric B.

Make sure that L1/L2 ports are connected before proceeding.

```
Enter the configuration method: console
Installer has detected the presence of a peer Fabric interconnect. This Fabric
interconnect will be added to the cluster. Do you want to continue {y|n}? y
Enter the admin password for the peer fabric interconnect: <password>
Physical switch Mgmt0 IPv4 address: 192.168.10.22
Apply and save the configuration (select “no” if you want to re-enter)? (yes/no):
yes
```

Verify the connectivity. After completing the FI configuration, verify the connectivity as below by logging to one of the Fabrics or the VIP address and checking the cluster state or extended state as shown below:

```
UCS-40G-Openstack-dom2-Fab-A# show cluster extended-state
Cluster Id: 0x48f6d6ca658111e6-0x924b0078883f3a3b

Start time: Mon Jul  3 20:41:21 2017
Last election time: Thu Jul  6 13:36:14 2017

A: UP, PRIMARY
B: UP, SUBORDINATE

A: memb state UP, lead state PRIMARY, mgmt services state: UP
B: memb state UP, lead state SUBORDINATE, mgmt services state: UP
   heartbeat state PRIMARY_OK

INTERNAL NETWORK INTERFACES:
eth1, UP
eth2, UP

HA READY
Detailed state of the device selected for HA storage:
Chassis 1, serial: FOX1948G99W, state: inactive
Chassis 2, serial: FOX1540GN14, state: active
Server 1, serial: FCH1904V31U, state: active
```

Log into Cisco UCS Manager

To login to Cisco UCS Manager, complete the following steps:

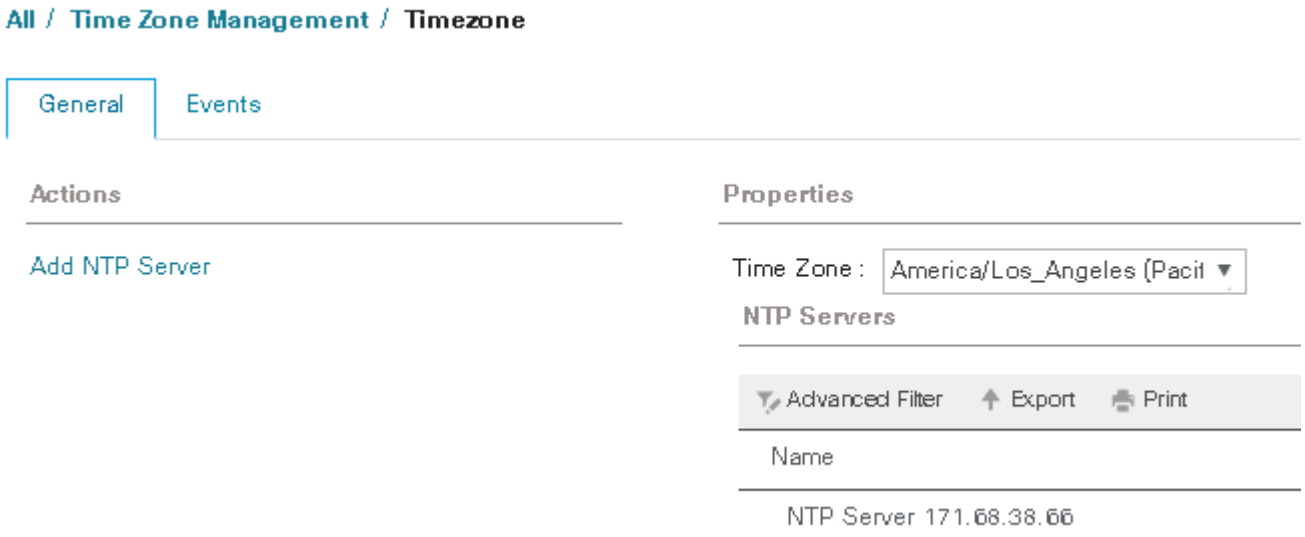
1. Open a Web browser and navigate to the Cisco UCS 6332 Fabric Interconnect cluster address.
2. Click the Launch link to download the Cisco UCS Manager software.
3. If prompted to accept security certificates, accept as necessary.
4. Click Launch UCS Manager HTML.
5. When prompted, enter **admin** for the username and enter the administrative password.
6. Click Login to log in to the Cisco UCS Manager.

Configure NTP Server

To configure the NTP server for the Cisco UCS environment, complete the following steps:

1. Select **Admin** tab on the left site.
2. Select Time Zone Management.
3. Select `Time Zone`.
4. Under **Properties** select your time zone.
5. Select Add NTP Server.
6. Enter the IP address of the NTP server.

7. Select OK.

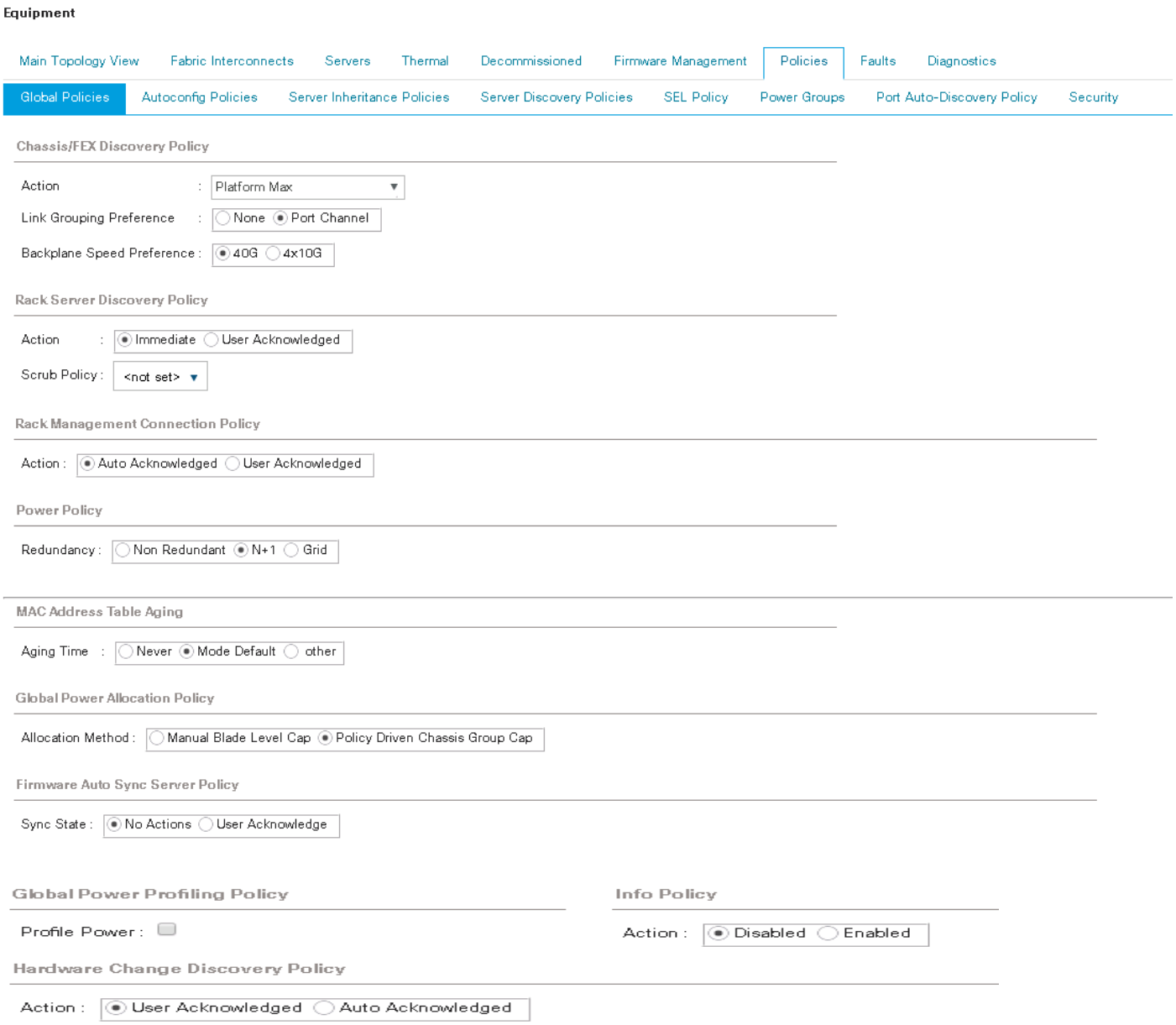


Initial Setup of the Environment

Configure the Cisco UCS Global Policies

To configure the Global policies, log into UCS Manager GUI, and complete the following steps:

1. Select the Equipment tab on the left site of the window.
2. Select Policies on the right site.
3. Select Global Policies.
4. Under Chassis/FEX Discovery Policy select Platform Max under Action.
5. Select 40G under Backplane Speed Preference.
6. Under Rack Server Discovery Policy select Immediate under Action.
7. Under Rack Management Connection Policy select Auto Acknowledged under Action.
8. Under Power Policy select Redundancy N+1.
9. Under Global Power Allocation Policy select Policy Driven.
10. Select Save Changes.



Configure Server Ports for Discovery

Navigate to each Fabric Interconnect and configure the server ports on Fabric Interconnects. Complete the following steps:

1. Under Equipment > Fabric Interconnects > Fabric Interconnect A > Fixed Module > Ethernet Ports;

    a. Select the ports (Port 1 to 14 per your requirement), right-click and select Configure as Server Port. In case you have clients like Cisco UCS blade or RACK servers connected to FI, you may have to configure additional ports on FI as server ports.

    b. Select the ports (23-26) right click and configure them as Network Ports.

    c. Click Save Changes to save the configuration.

    d. Repeat the above steps on Fabric Interconnect B too and save the configuration.

Equipment / Fabric Interconnects / Fabric Interconnect A (primary) / Fixed Module / Ethernet Ports

Ethernet Ports

Advanced Filter | Export | Print | All | Unconfigured | Network | Server | FCoE Uplink | Unified Uplink | Appliance Storage | FCoE Storage | Unified Storage | Monitor

Slot	Aggr. Port ID	Port ID	MAC	If Role	If Type	Overall Status	Admin State
1	0	1	00:78:88:3F:3A:42	Server	Physical	Up	Enabled
1	0	2	00:78:88:3F:3A:46	Server	Physical	Up	Enabled
1	0	3	00:78:88:3F:3A:4A	Server	Physical	Up	Enabled
1	0	4	00:78:88:3F:3A:4E	Server	Physical	Up	Enabled
1	0	5	00:78:88:3F:3A:52	Server	Physical	Up	Enabled
1	0	6	00:78:88:3F:3A:56	Server	Physical	Up	Enabled
1	0	7	00:78:88:3F:3A:5A	Server	Physical	Up	Enabled
1	0	8	00:78:88:3F:3A:5E	Server	Physical	Up	Enabled
1	0	9	00:78:88:3F:3A:62	Server	Physical	Up	Enabled
1	0	10	00:78:88:3F:3A:66	Server	Physical	Up	Enabled
1	0	11	00:78:88:3F:3A:6A	Server	Physical	Up	Enabled
1	0	12	00:78:88:3F:3A:6E	Server	Physical	Up	Enabled
1	0	13	00:78:88:3F:3A:72	Server	Physical	Up	Enabled
1	0	14	00:78:88:3F:3A:73	Server	Physical	Up	Enabled

Ethernet Ports

Advanced Filter | Export | Print | All | Unconfigured | Network | Server | FCoE Uplink | Unified Uplink | Appliance Storage | FCoE Storage | Unified Storage | Monitor

Slot	Aggr. Port ID	Port ID	MAC	If Role	If Type	Overall Status	Admin State
1	0	23	00:78:88:3F:3A:94	Network	Physical	Up	Enabled
1	0	24	00:78:88:3F:3A:98	Network	Physical	Up	Enabled
1	0	25	00:78:88:3F:3A:9C	Network	Physical	Up	Enabled
1	0	26	00:78:88:3F:3A:A0	Network	Physical	Up	Enabled

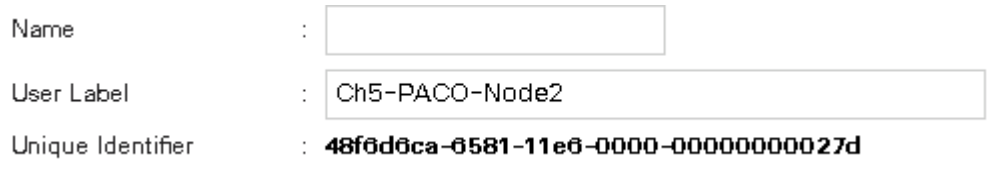
Power on the Servers

Power on the Cisco UCS S3260 and Cisco UCS C220 servers along with any other client servers connected to Fabric Interconnects and the let the discovery complete.

Label Servers

To label each server (for better identification) with the following steps:

1. Select the Equipment tab on the left site.
2. Select Chassis > Chassis 1 > Server 1.
3. In the Properties section on the right go to User Label and add Ch1-PACO-Node1 to the field. In case of server 2 (dual node config) you may make it as Ch1-PACO-Node2 and so on.
4. Similarly the RACK servers can be named for a better identification.

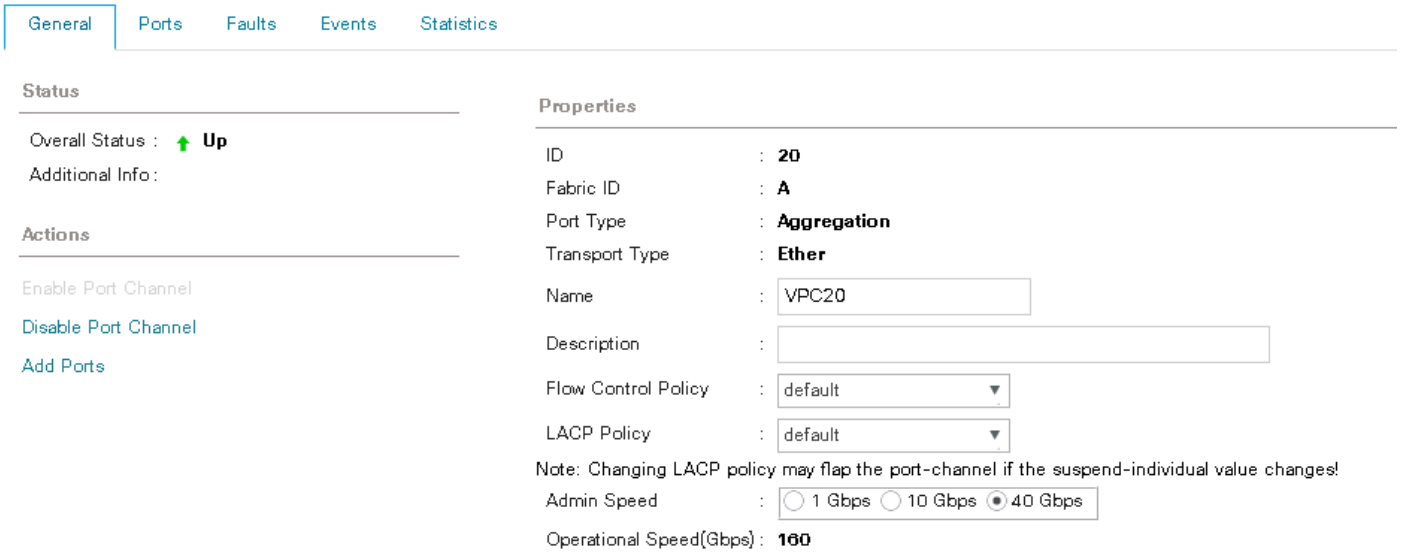


Name	:	SSContrroller1
User Label	:	SSContrroller-Active-25RU
Unique Identifier	:	**48f6d6ca-6581-11e6-0000-00000000022f**
Service Profile	:	org-root/ls-Swift-Controller-1

Creating Port Channel for FI Uplinks

To create Port Channels to the connected Nexus 9332PQ switches, complete the following steps:

1. Select the **LAN** tab in the left pane of the UCSM GUI on Fabric Interconnect A
2. Go to LAN > LAN Cloud > Fabric A > Port Channels and right-click Create Port Channel.
3. Type in **ID 20** and Click Next.
4. Select the available ports on the left **23-26** (Network Ports configured in the FI) and assign them with >> to **Ports in the Port Channel**.
5. Create Port Channel.
6. Click Finish and then OK.
7. Repeat the procedure for Fabric Interconnect B.



Configure UCS Servers

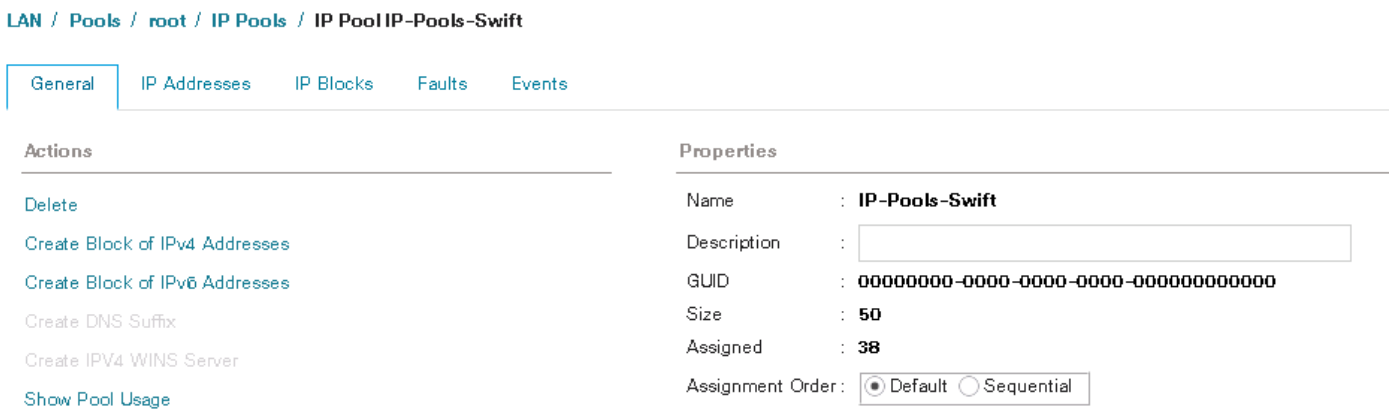
Configure Pools and Policies

Create KVM Pools

To create a KVM IP Pool, complete the following:

1. Select the LAN tab on the left site.
2. Go to LAN > Pools > root > IP Pools > IP Pool ext-mgmt.
3. Right-click Create Block of IPv4 Addresses.
4. Enter an IP Address in the From field.
5. Enter Size 50
6. Enter your Subnet Mask.
7. Fill in your Default Gateway.
8. Enter your Primary DNS and Secondary DNS if needed.
9. Click OK.

The size of the pool depends on the total number of servers, clients, and controllers.

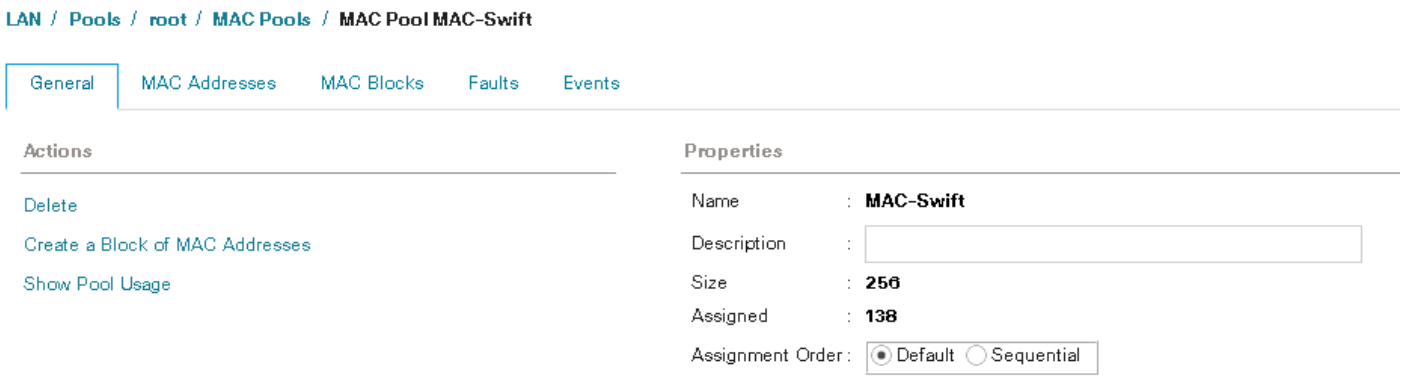


Create MAC Pools

To create a MAC Pool, complete the following steps:

1. Select the LAN tab on the left site.
2. Go to LAN > Pools > root > Mac Pools and right-click Create MAC Pool.

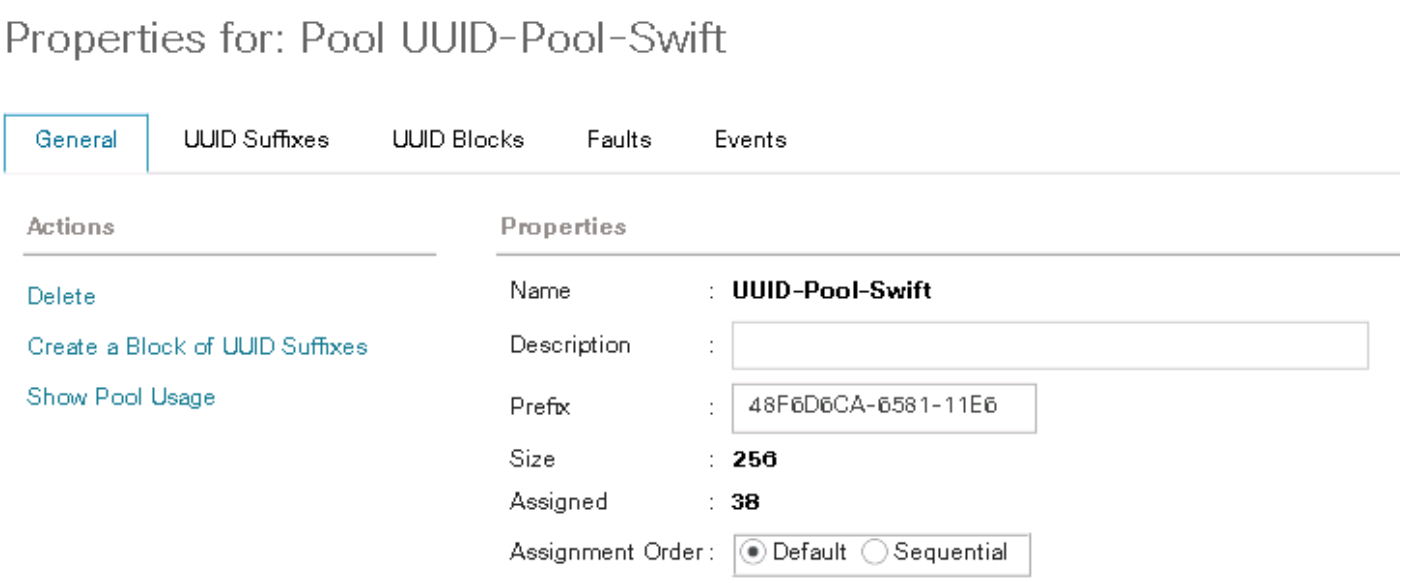
3. Type in MAC-Swift for Name.
4. (Optional) Enter a Description of the MAC Pool.
5. Click Next.
6. Click Add.
7. Specify a starting MAC address.
8. Specify the size of the MAC address pool, which is sufficient to support the available resources say 256.



Create UUID Pools

To create a UUID Pool, complete the following steps:

1. Select the Servers tab on the left site.
2. Go to Servers > Pools > root > UUID Suffix Pools and right-click Create UUID Suffix Pool.
3. Type in UUID-Pool-Swift for Name.
4. (Optional) Enter a Description of the UUID Pool.
5. Click Next.
6. Click Add.
7. Specify a starting UUID Suffix.
8. Specify a size of the UUID suffix pool, which is sufficient to support the available server resources, i.e. 256.
9. Click OK.
10. Click Finish and then OK.



Create VLANs

As mentioned earlier, it is important to separate the network traffic with VLANs for outward facing, Cluster and Replication traffics. The following table shows the configured VLANs.

Table 5 Configured VLANs

VLAN	Name	Function
10	UCSM-VLAN	Optional if UCSM is not in private VLAN.
120	Swift-Client	Outward Client Facing Network.
130	Swift-Cluster	Cluster Network
150	Swift-Replication	Replication Network
166	External	External Network for OS to download and yum updates

To Configure VLAN's in Cisco UCS Manager GUI, complete the following steps:

1. Select LAN in the left pane in the UCSM GUI.
2. Select LAN > LAN Cloud > VLANs and right-click Create VLANs.
3. Enter Swift-Cluster for the VLAN Name.
4. Keep Multicast Policy Name as <not set>.
5. Select Common/Global for Public.
6. Enter 120 in the VLAN IDs field.

7. Click OK and then Finish.

Repeat these steps for the rest of the VLAN's.

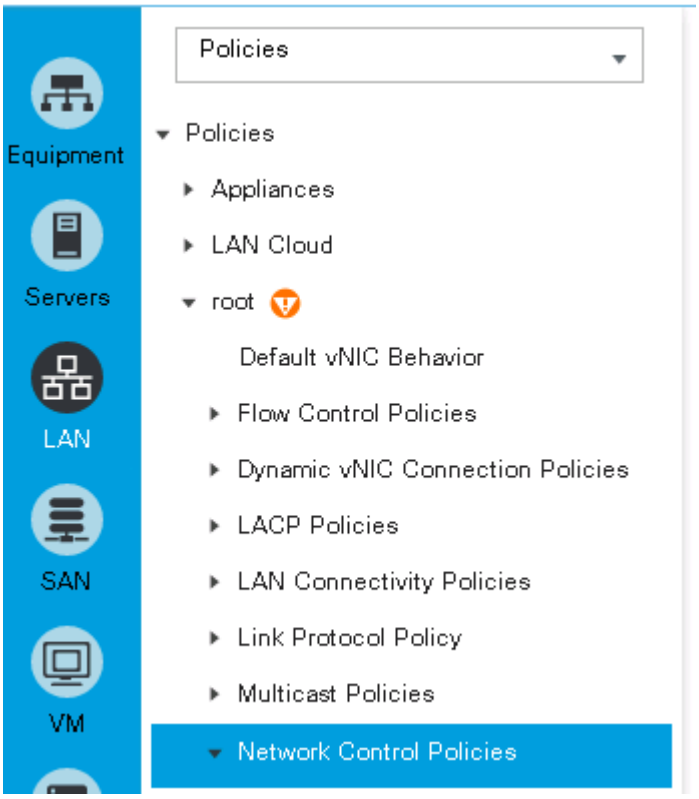
Enable CDP

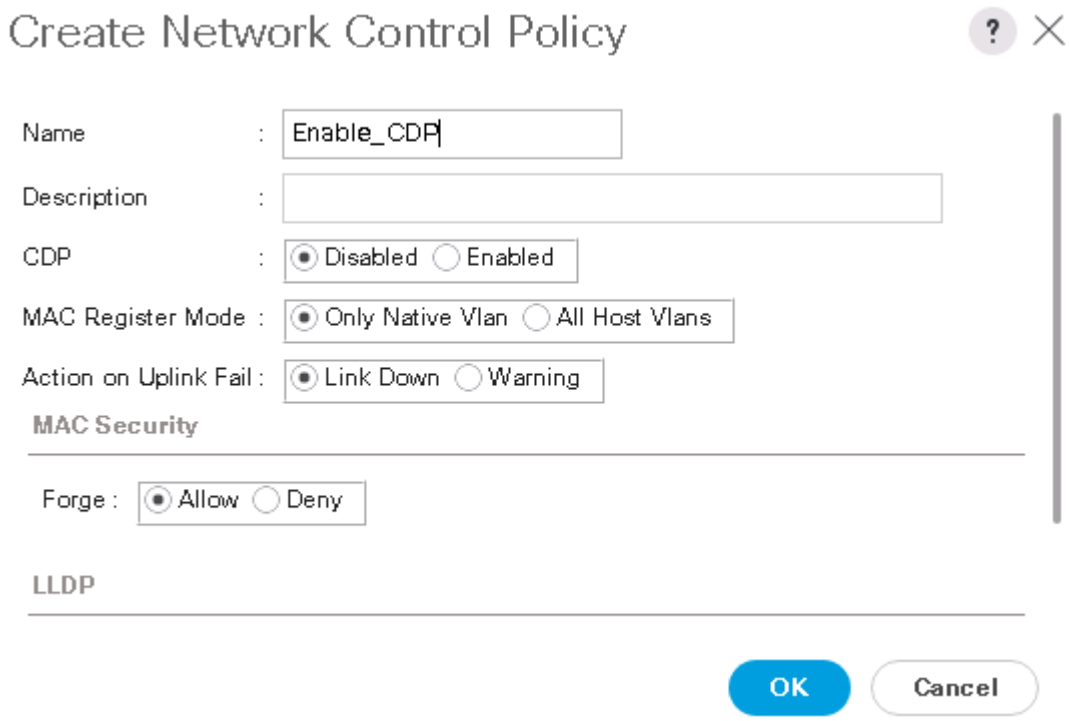
To enable Network Control Policies, complete the following steps:

1. Select the LAN tab in the left pane of the UCSM GUI.
2. Go to LAN > Policies > root > Network Control Policies and right-click Create Network-Control Policy.
3. Type in Enable-CDP in the Name field.
4. (Optional) Enter a description in the Description field.
5. Click Enabled under CDP.
6. Click All Hosts VLANs under MAC Register Mode.

7. Leave everything else untouched and click OK.

8. Click OK.

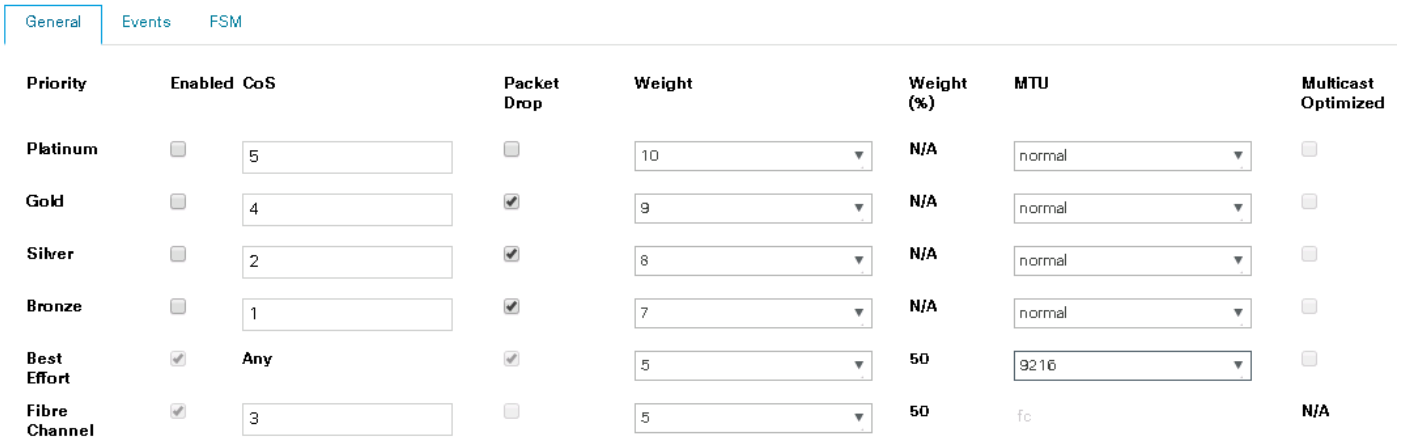




QOS System Class

Make sure to have 9216 as MTU for Best Effort classes in QOS System Class.

1. Select the LAN tab in the left pane of the UCSM GUI.
2. Go to LAN > LAN Cloud > QoS System Class.
3. Set Best Effort MTU as 9216.
4. Set Fibre Channel Weight to None.
5. Click Save Changes and then OK.



vNIC Template Setup

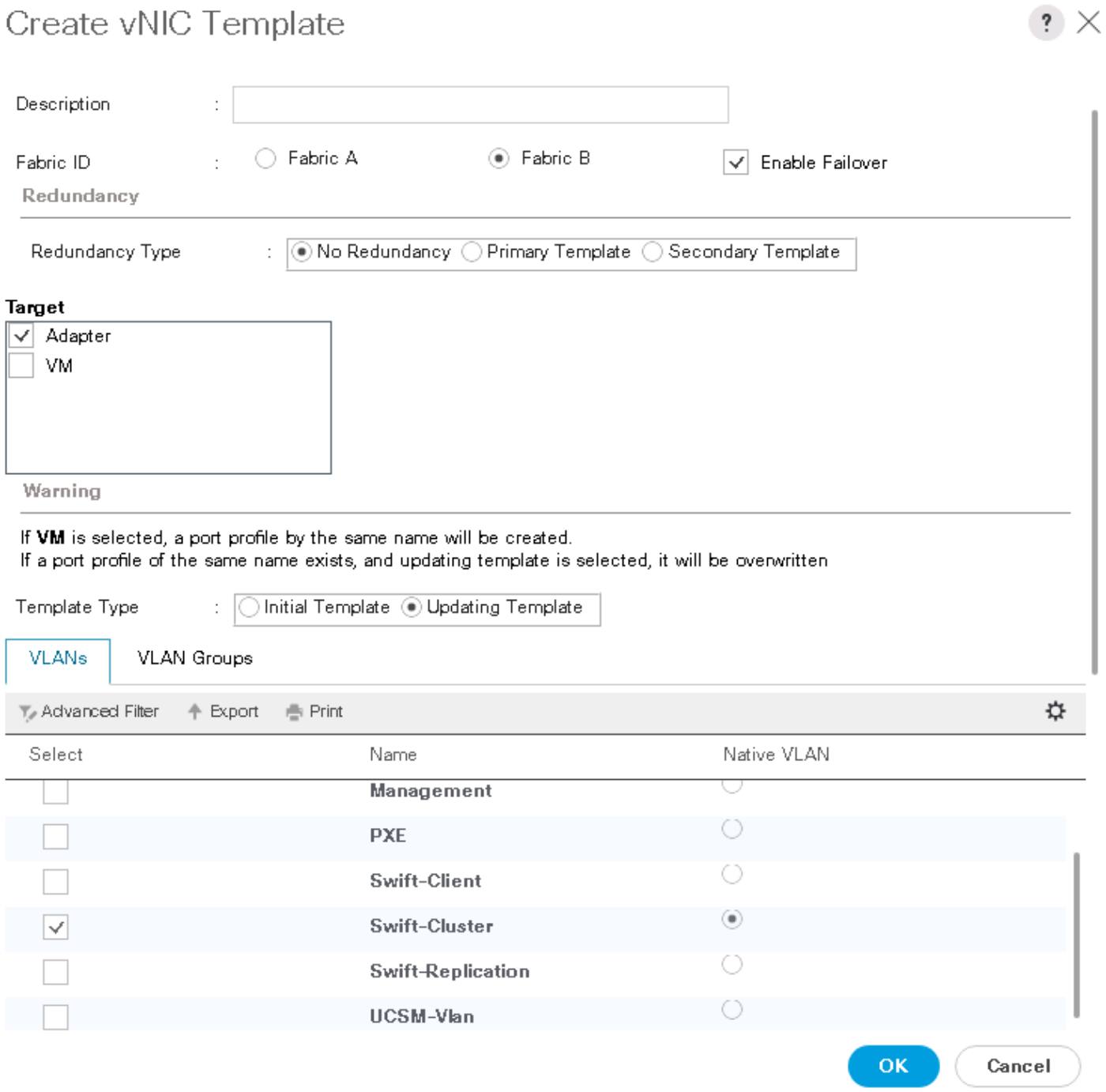
Based on the previous section to create VLANs, the next step is to create the appropriate vNIC templates. For SwiftStack Storage we need to create four different vNICs, depending on the role of the server. Table 6 provides an overview of the configuration.

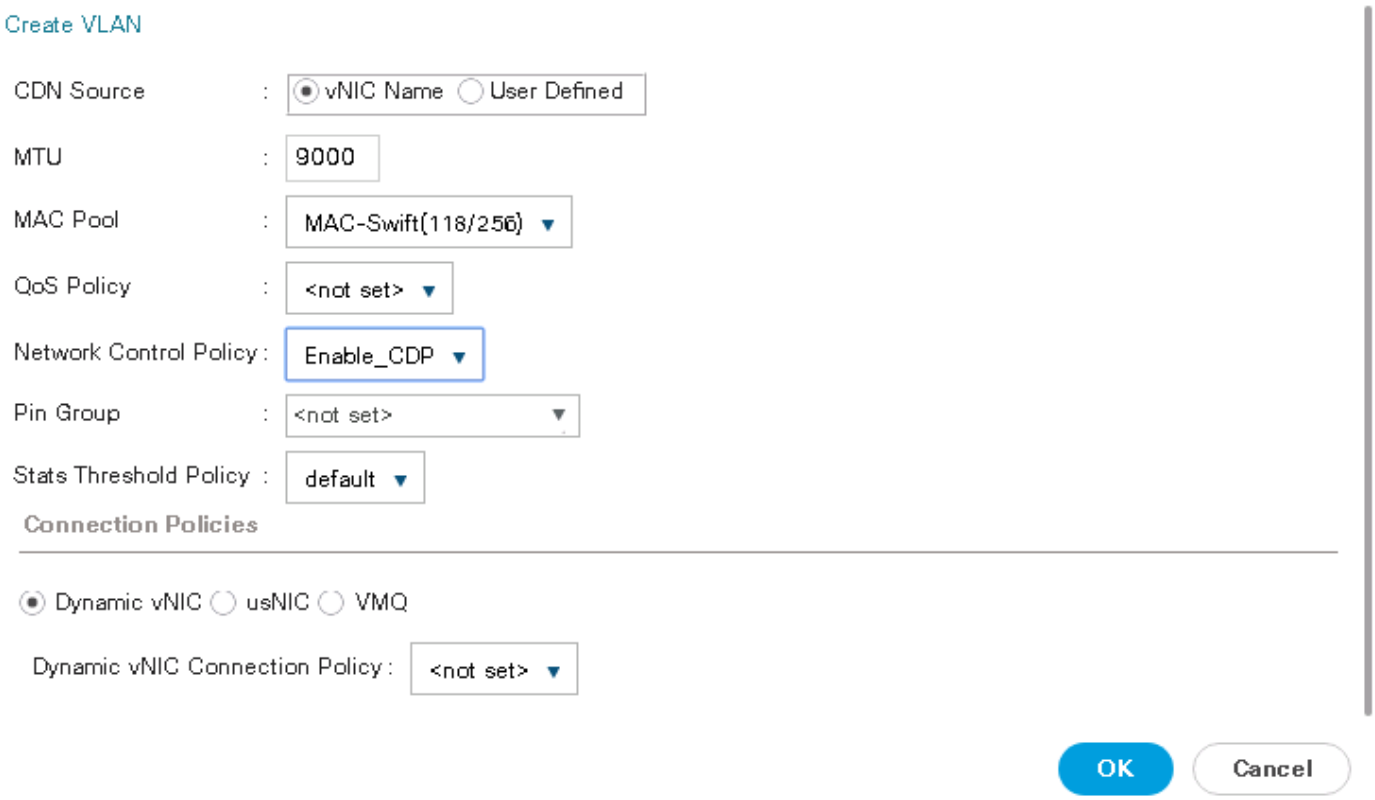
Table 6 Configuration Overview

Name	vNIC Name	Fabric Interconnect	Failover	VLAN	MTU Size
Outward Facing	Swift-Client	A	Yes	Swift-Client	9000
Cluster Network	Swift-Cluster	B	Yes	Swift-Cluster	9000
Replication Network	Swift-Repl	B	Yes	Swift-Repl	9000
External Network	External	A	Yes	External	1500

To create the appropriate vNICs, complete the following steps:

1. Select the LAN tab in the left pane of the UCSM GUI.
2. Go to LAN > Policies > root > vNIC Templates and right-click Create vNIC Template.
3. Type in Swift-Cluster in the Name field.
4. (Optional) Enter a description in the Description field.
5. Click Fabric B as Fabric ID and enable failover.
6. Select Swift-Cluster as VLANs and click Native VLAN.
7. Select MAC-Swift as MAC Pool.
8. Select Swift-Cluster as QoS Policy.
9. Select Enable-CDP as Network Control Policy.
10. Click OK and then OK.


Create vNIC Template
Description :
Fabric ID : Fabric A Fabric B Enable Failover
Redundancy
Redundancy Type : No Redundancy Primary Template Secondary Template
Target
Adapter
VM
Warning
If VM is selected, a port profile by the same name will be created.
If a port profile of the same name exists, and updating template is selected, it will be overwritten
Template Type : Initial Template Updating Template
VLANs
VLAN Groups
Advanced Filter
Export
Print
Select
Name
Native VLAN
Management
PXE
Swift-Client
Swift-Cluster
Swift-Replication
UCSM-Vlan
OK
Cancel



Repeat these steps for the remaining vNIC's, taking care of MTU size and the vLAN names.

Ethernet Adapter Policy Setup

By default, Cisco UCS provides a set of Ethernet adapter policies. These policies include the recommended settings for each supported server operating system. Operating systems are sensitive to the settings in these policies.

Cisco UCS best practice is to enable Jumbo Frames MTU 9000 for any Storage facing Networks (Outward facing Network, Cluster Network and Replication Network). For Jumbo Frames MTU9000, you can use default Ethernet Adapter Policy predefined as Linux. Once the jumbo frame setting are made in the vNIC template as above the ether interface created on the Operating System will get its MTU as 9000.

You can also create a specific adapter policy and it is beneficial in particular if you are using MTU 1500 say for outward facing network.

To create a specific adapter policy for Red Hat Enterprise Linux, complete the following steps:

1. Select the Server tab in the left pane of the UCSM GUI.
2. Go to Servers > Policies > root > Adapter Policies and right-click Create Ethernet Adapter Policy.
3. Type in RHEL in the Name field.
4. (Optional) Enter a description in the Description field.
5. Under Resources type in the following values:

Transmit Queues: 8

Ring Size: 4096

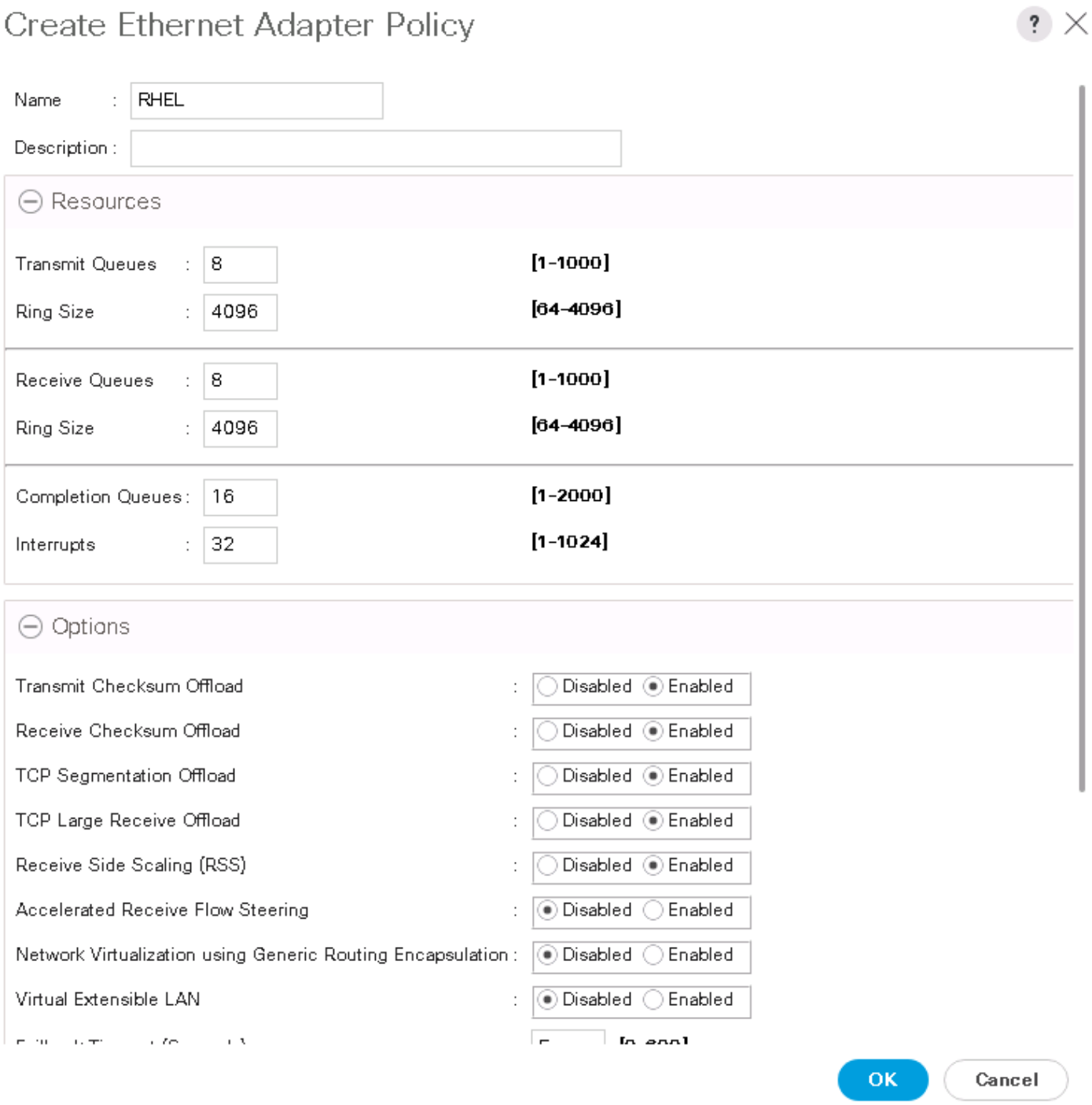
Receive Queues: 8

Ring Size: 4096

Completion Queues: 16

Interrupts: 32

6. Under Options enable Receive Side Scaling (RSS).
7. Click OK and then OK.

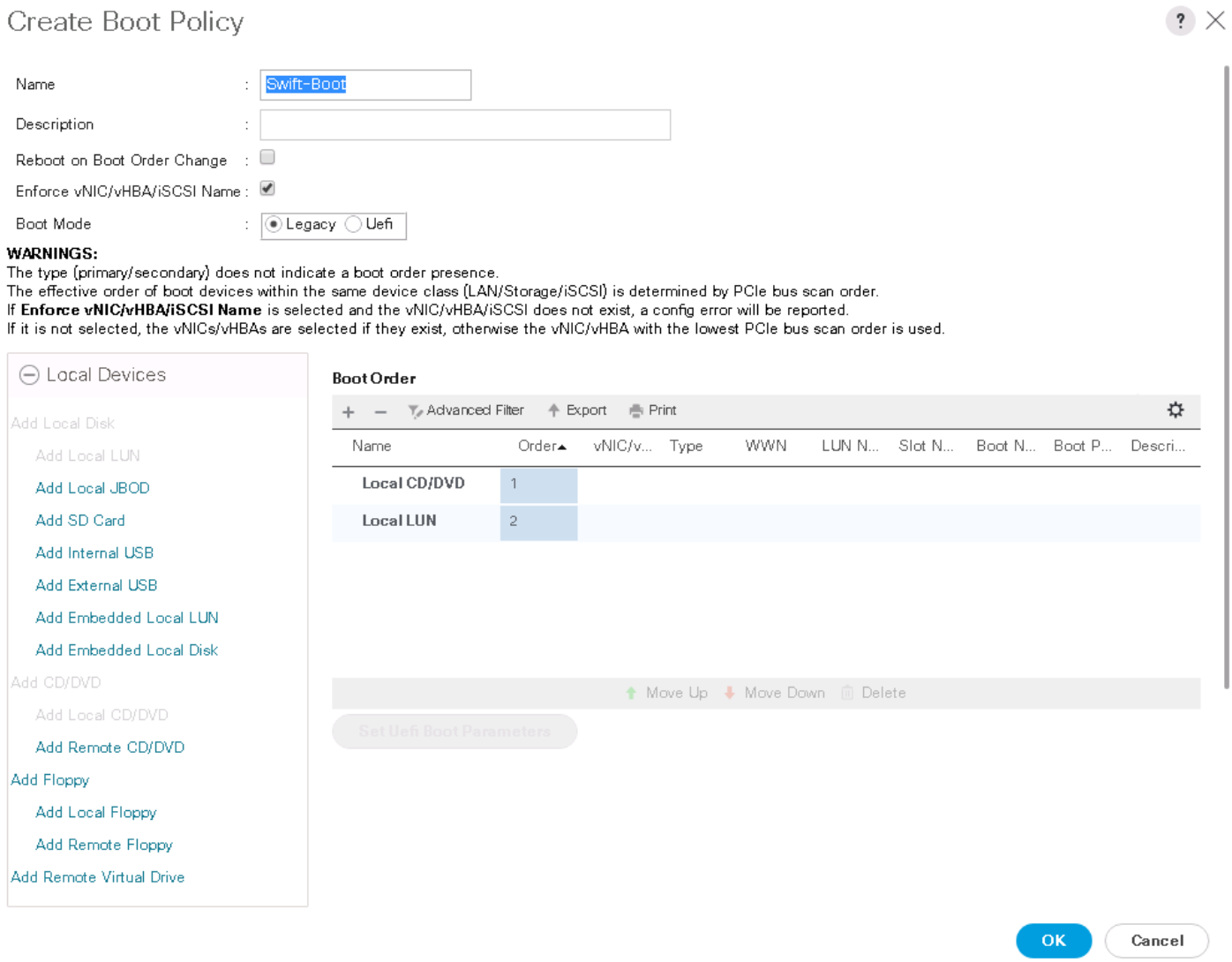


Boot Policy Setup

To create a Boot Policy, complete the following steps :

1. Select the Servers tab in the left pane.
2. Go to Servers > Policies > root > Boot Policies and right-click Create Boot Policy.
3. Type in a Swift-Boot in the Name field.

4. (Optional) Enter a description in the Description field.
5. Click Local Devices > Add Local CD/DVD and Click OK.
6. Click Local Devices > Add Local LUN and Set Type as “Any” and Click OK.
7. Click OK.

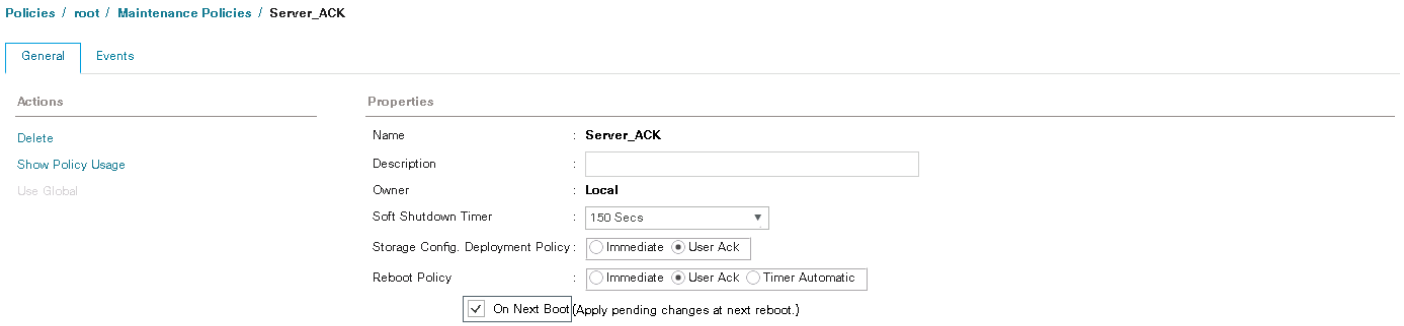


Create Maintenance Policy Setup

To setup a Maintenance Policy, complete the following steps :

1. Select the Servers tab in the left pane.
2. Go to Servers > Policies > root > Maintenance Policies and right-click Create Maintenance Policy.
3. Type in a Server-Maint in the Name field.

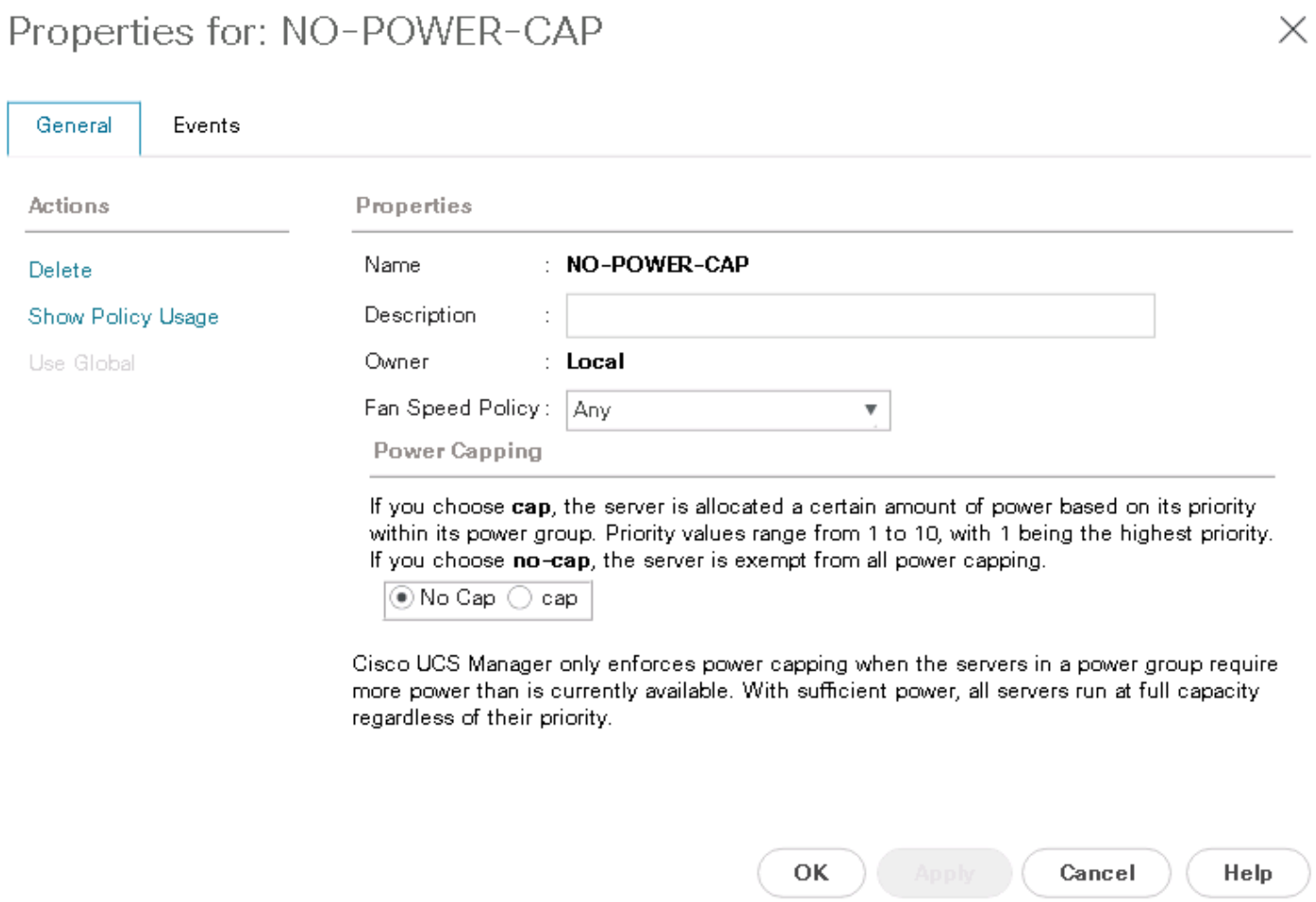
4. (Optional) Enter a description in the Description field.
5. Click User Ack under Reboot Policy.
6. Click OK and then OK.



Create Power Control Policy Setup

To create a Power Control Policy, complete the following steps:

1. Select the Servers tab in the left pane.
2. Go to Servers > Policies > root > Power Control Policies and right-click Create Power Control Policy.
3. Type in No-Power-Cap in the Name field.
4. (Optional) Enter a description in the Description field.
5. Click No Cap and Click OK.



Creating Chassis Profile

The Chassis Profile is required to assign specific disks to a particular server node in a Cisco UCS S3260 Storage Server as well as upgrading to a specific chassis firmware package. The disks can be asymmetrically assigned to both the nodes as desired. However the disks were equally distributed amongst the two nodes in dual node configuration.

This section covers the Disk zoning policy that is attached to the chassis profile. Whether to allocate all the 56 disks to a single node or 28 disks per node in a dual configuration is defined by the Disk Zoning Policy.

Create Chassis Maintenance Policy

To create a Chassis Maintenance Policy, complete the following steps:

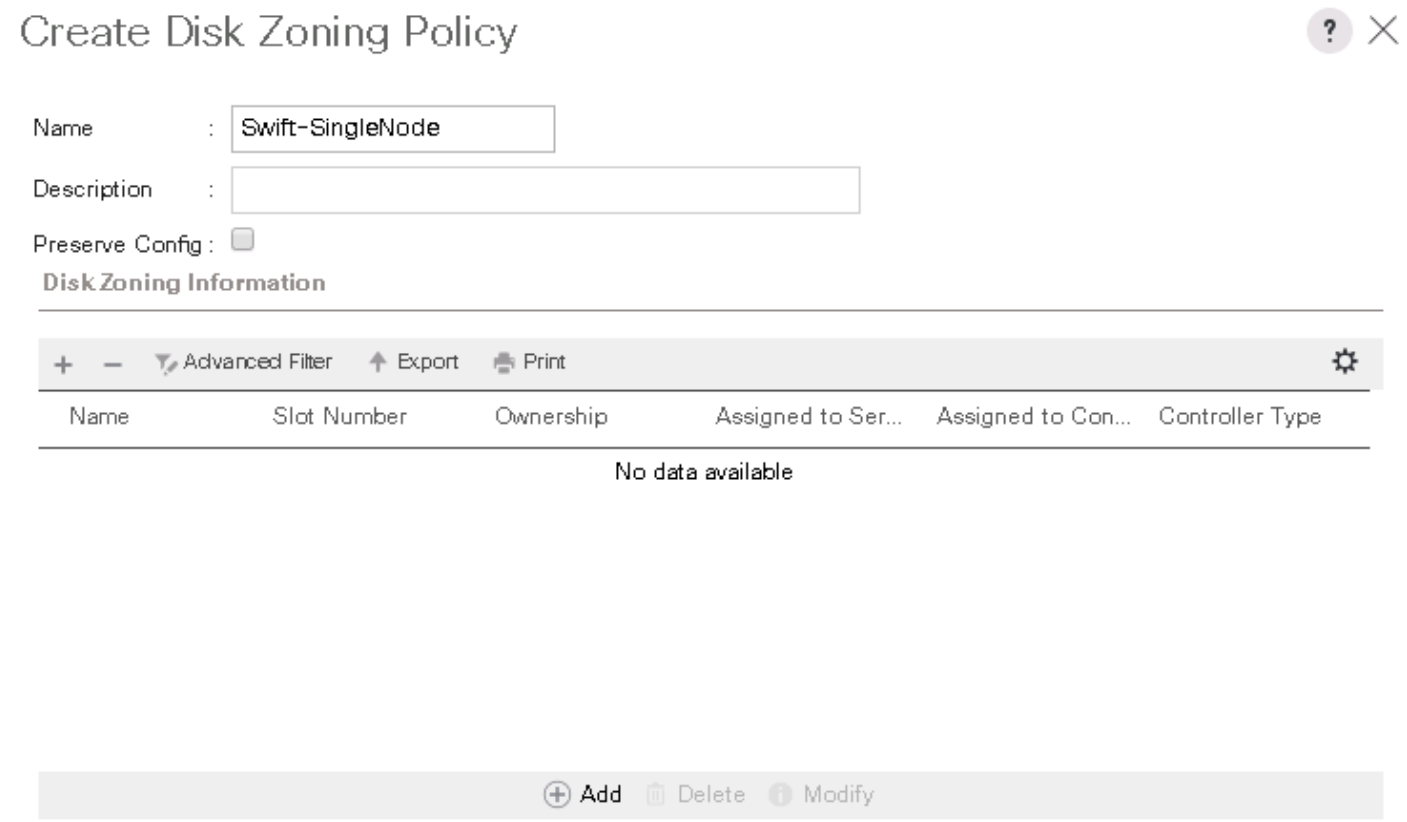
1. Select the **`Chassis`** tab in the left pane of the Cisco UCSM GUI.
2. Go to Chassis > Policies > root > Chassis Maintenance Policies and right-click Create Chassis Maintenance Policy.
3. Type in ***`Swift`*** in the **`Name`** field.

4. (Optional) Enter a description in the **Description** field.
5. Click **OK** and then **OK**.

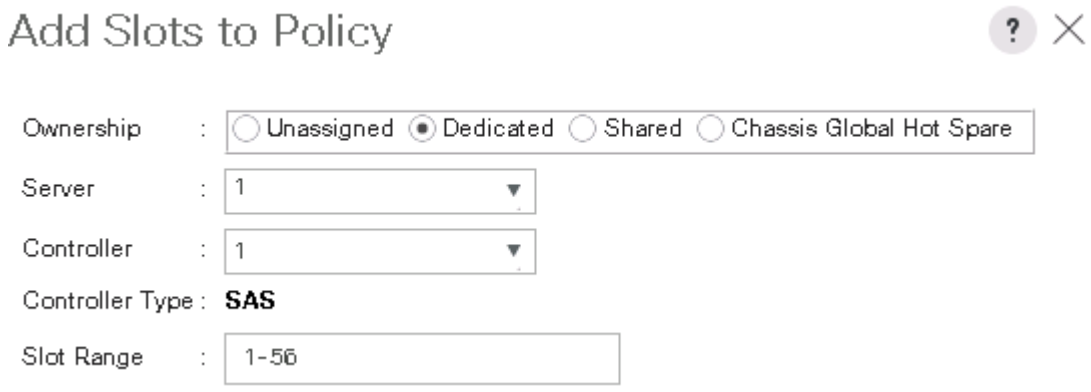
Create Disk Zoning Policy

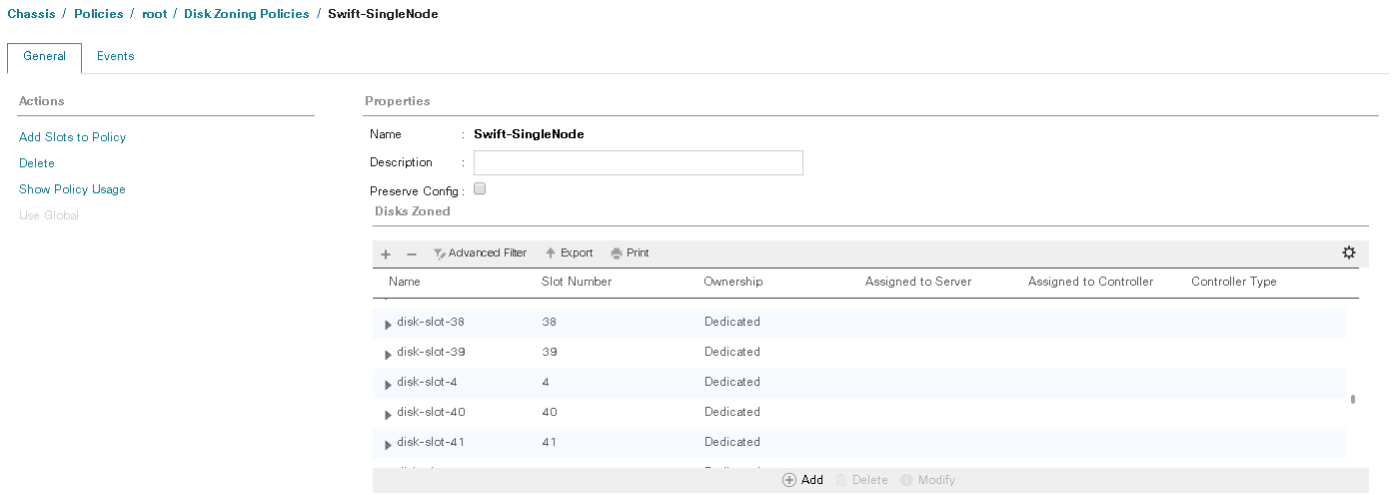
To create the disk zoning policy, complete the following steps:

1. Two polices one with Single Node and one with Dual Node are shown below. You may be needing only one out of the two depending on the installation.
2. Single Node Disk Zoning Policy:
    a. Select the **Chassis** tab in the left pane of the Cisco UCSM GUI.
    b. Go to Chassis > Policies > root > Disk Zoning Policies and right-click Create Disk Zoning Policy.
    c. Enter Swift-SingleNode in the Name field
    d. (Optional) Enter a description in the **Description** field.



3. In the Disk Zoning Information above, Click Add and add all the 56 disks as dedicated to Node 1 and Controller 1.





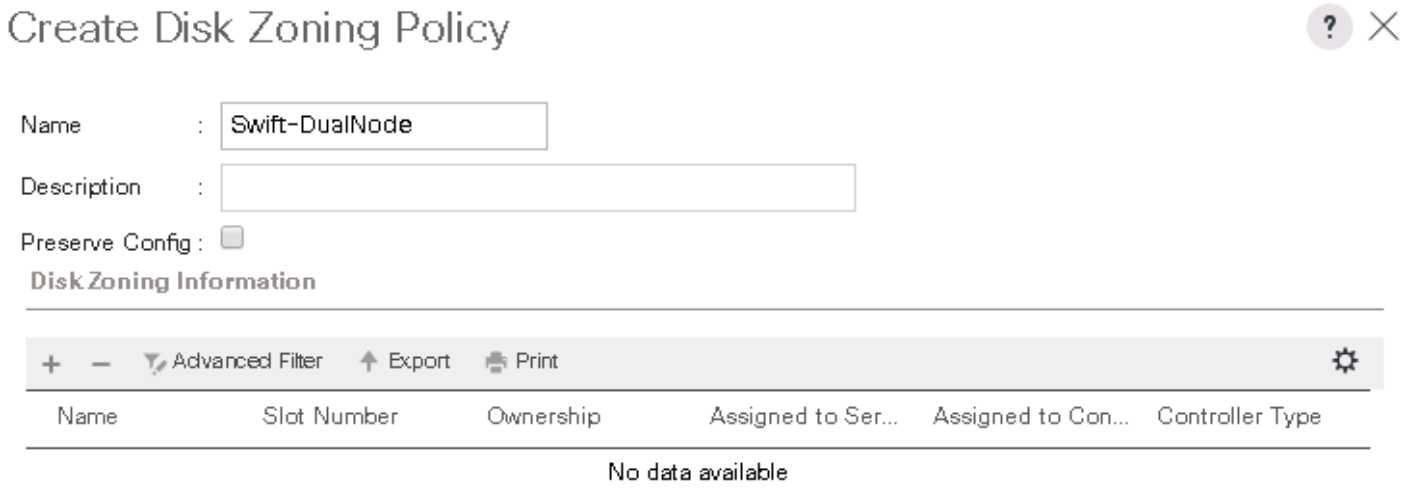
4. Dual Node Disk Zoning Policy:

    a. Select the **`Chassis`** tab in the left pane of the Cisco UCSM GUI.

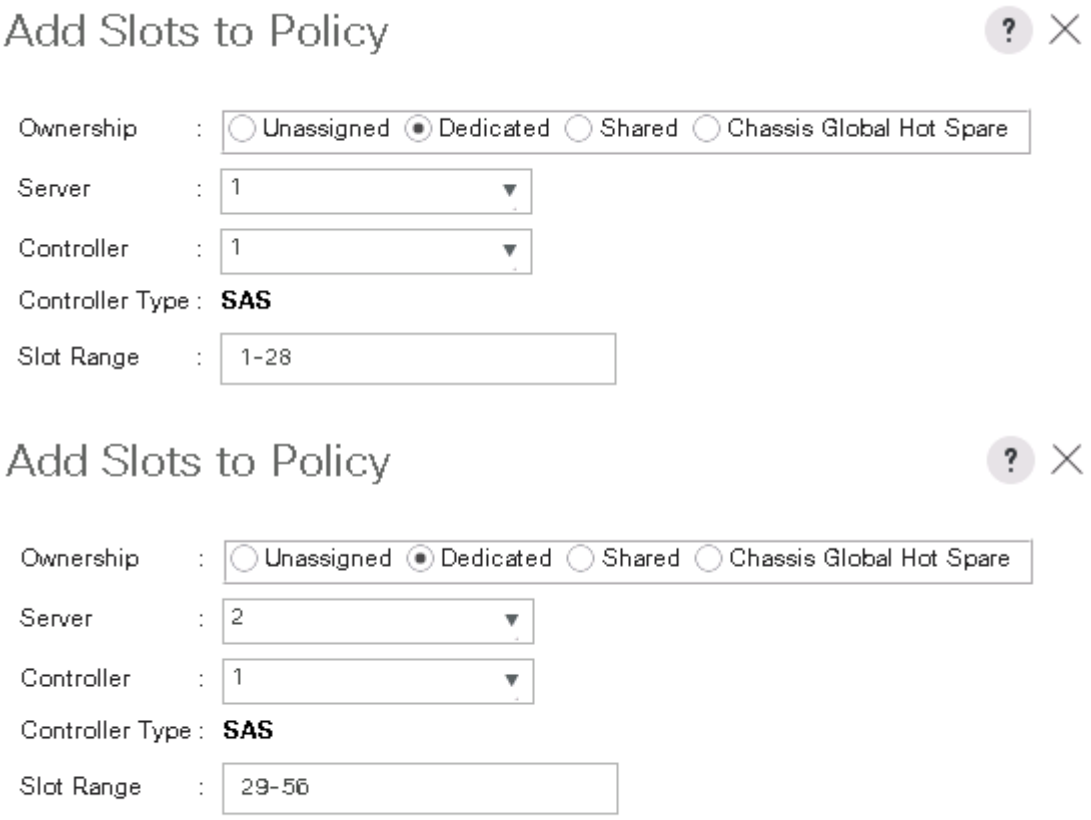
    b. Go to Chassis > Policies > root > Disk Zoning Policies and right-click Create Disk Zoning Policy.

    c. Enter Swift-DualNode in the Name field

    d. (Optional) Enter a description in the **`Description`** field.



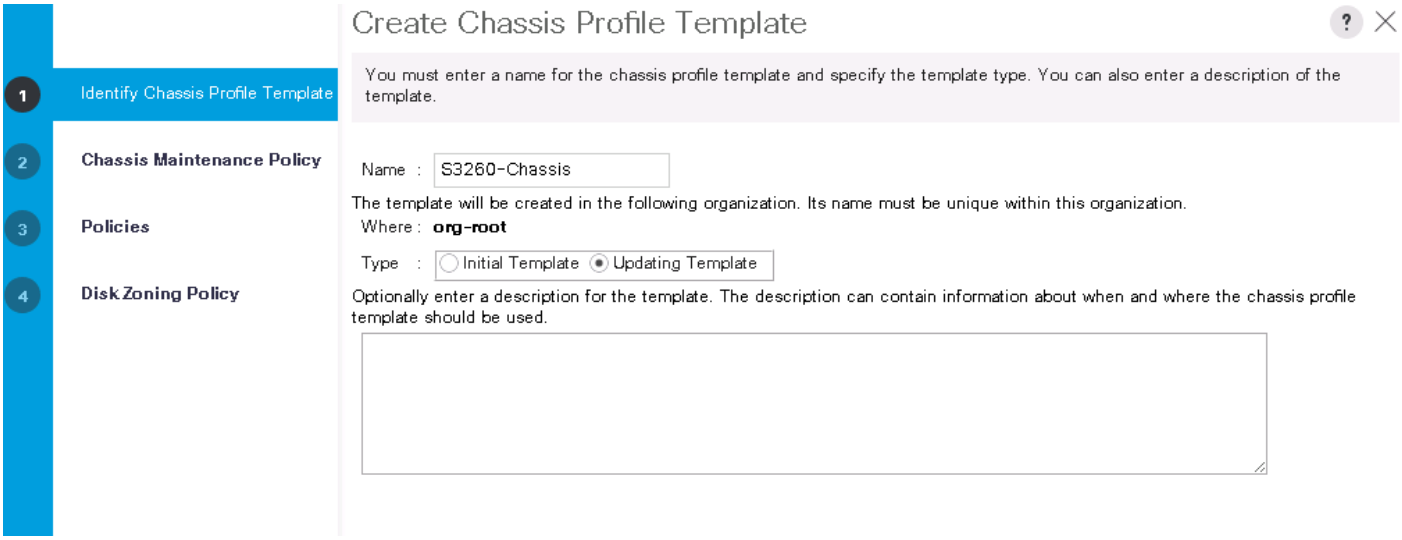
5. In the Disk Zoning Information above, Click Add and add 28 disks to Node 1 and Controller 1 and the rest 28 disks to Node2 and Controller 1.



Create Chassis Profile Template

To create a Chassis Profile Template, complete the following steps:

1. Select the Chassis tab in the left pane of the Cisco UCSM GUI.
2. Go to Chassis > Chassis Profile Templates and right-click Create Chassis Profile Template.
3. Type in S3260-Chassis in the Name field.
4. Under Type, select Updating Template.
5. (Optional) Enter a description in the Description field.

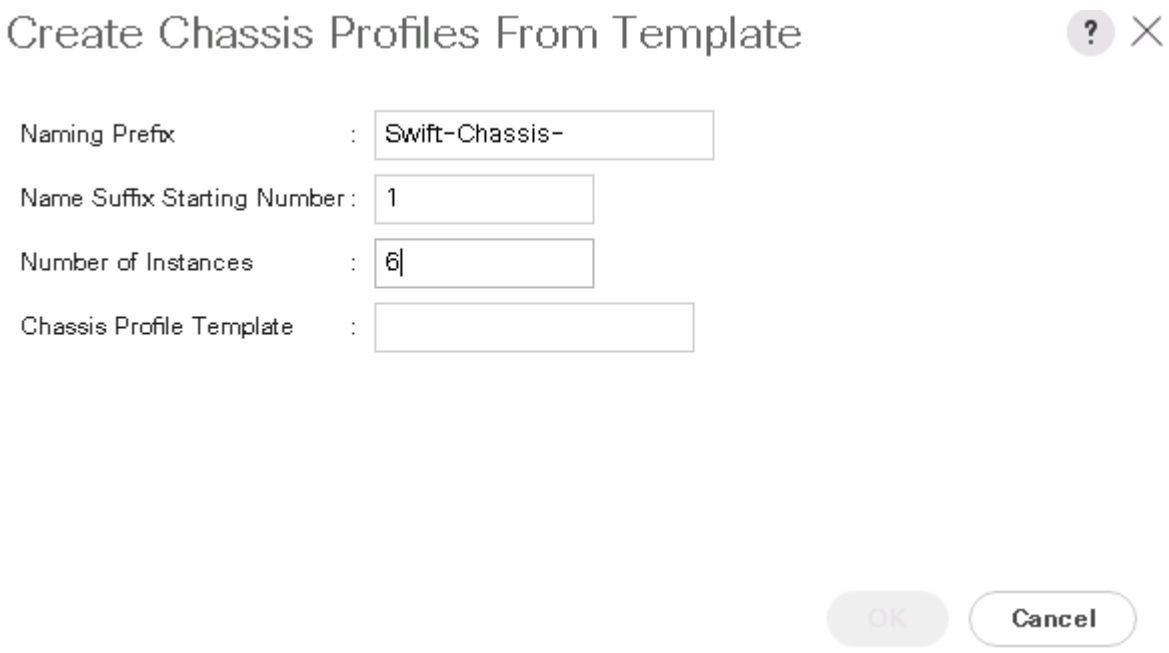


6. Select Chassis Maintenance Policy as ‘Swift’ created earlier.
7. Under Disk Zoning Policy, Select the zoning policy created earlier: SingleNode or DualNode.
8. Click Finish to complete the Chassis Profile Template creation.

Create Chassis Profile from Template

To create the Chassis Profiles from the previous created Chassis Profile Template, complete the following steps:

1. Select the Chassis tab in the left pane of the Cisco UCSM GUI.
2. Go to Chassis > Chassis Profiles and right-click Create Chassis Profiles from Template.
3. Type in Swift-Chassis- in the Name field.
4. Leave the Name Suffix Starting Number untouched.
5. Enter 6 for the Number of Instances for all connected Cisco UCS S3260 Storage Server.
6. Choose your previously created Chassis Profile Template.
7. Click OK.



This will create 6 chassis profiles that can be associated later.

Associate Chassis Profile

To associate all previous created Chassis Profile, complete the following steps:

1. Select the Chassis tab in the left pane of the Cisco UCSM GUI.
2. Go to Chassis > Chassis Profiles and select Swift-Chassis-1.
3. Right-click Change Chassis Profile Association.
4. Under Chassis Assignment, select existing Chassis.
5. Under Available Chassis, select the first chassis.
6. Click OK and then OK.
7. Repeat these steps for the other four Chassis Profiles by selecting the IDs 2 – 6.
8. Associate Chassis Profiles

After the association of the Chassis profile along with Disk zoning policy, the disks distribution between the nodes may get corrected in few minutes.

Convert the Disks to Unconfigured Good

The boot disks on both Cisco UCS C220 and Cisco UCS S3260 should be Unconfigured Good before installing the operating system. If not, complete the following steps to make them Unconfigured Good.

1. For S3260 Server:

a. Select Chassis -> Servers -> Server1.

b. In the right pane, select Inventory-> Storage-> Disks.

c. Select the Bootable SSD disks, right click and make them as Unconfigured Good.

Disk 201	456809	BTWA639000BJ480FGN
Disk 202	456809	BTWA63860865480FGN

2. Repeat these steps for Server2 and all other Cisco UCS S3260 servers in the cluster.

3. Repeat the same for all Cisco UCS C220 servers.

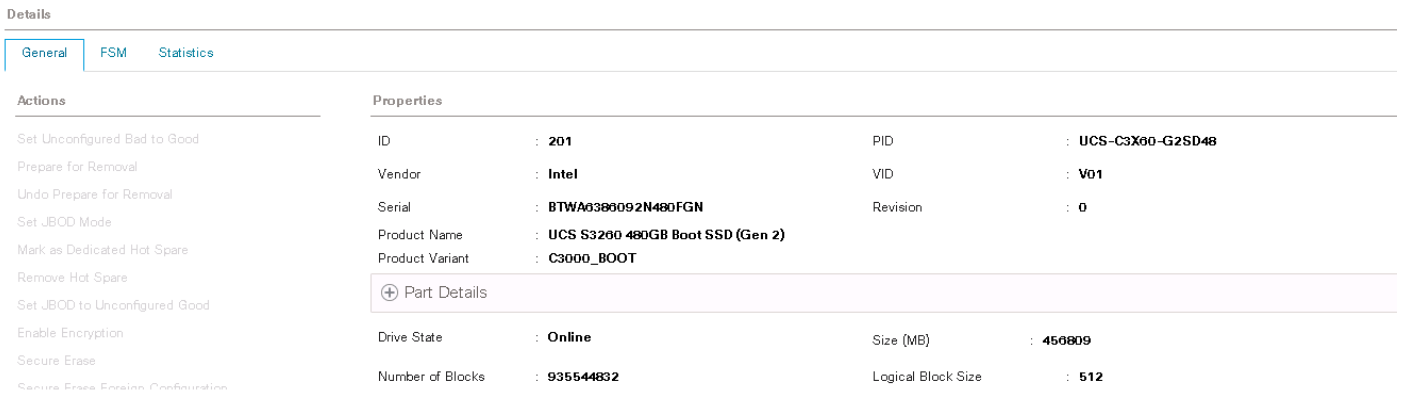
In case of using any automation like PXE boot etc. for S3260, you may have to convert all other disks into unconfigured good too. The Boot LUN created (after applying service profiles) for OS will be the only LUN visible to PXE server where OS will be installed. The non-boot disks can be turned back to JBOD after OS installation either through UCSM GUI or utilities like storcli.

Create Storage Profiles

Storage Profile for Cisco UCS S3260 Storage Server

1. Identify the boot disk numbers for the servers by navigating to the server inventory. Navigate to Server1 and/or Server2 under Chassis, Inventory tab, Storage Tab and then expand the disks under SAS Storage Controller.

Disk 201	456809	BTWA6386092N480FGN
Disk 202	456809	BTWA63860862480FGN

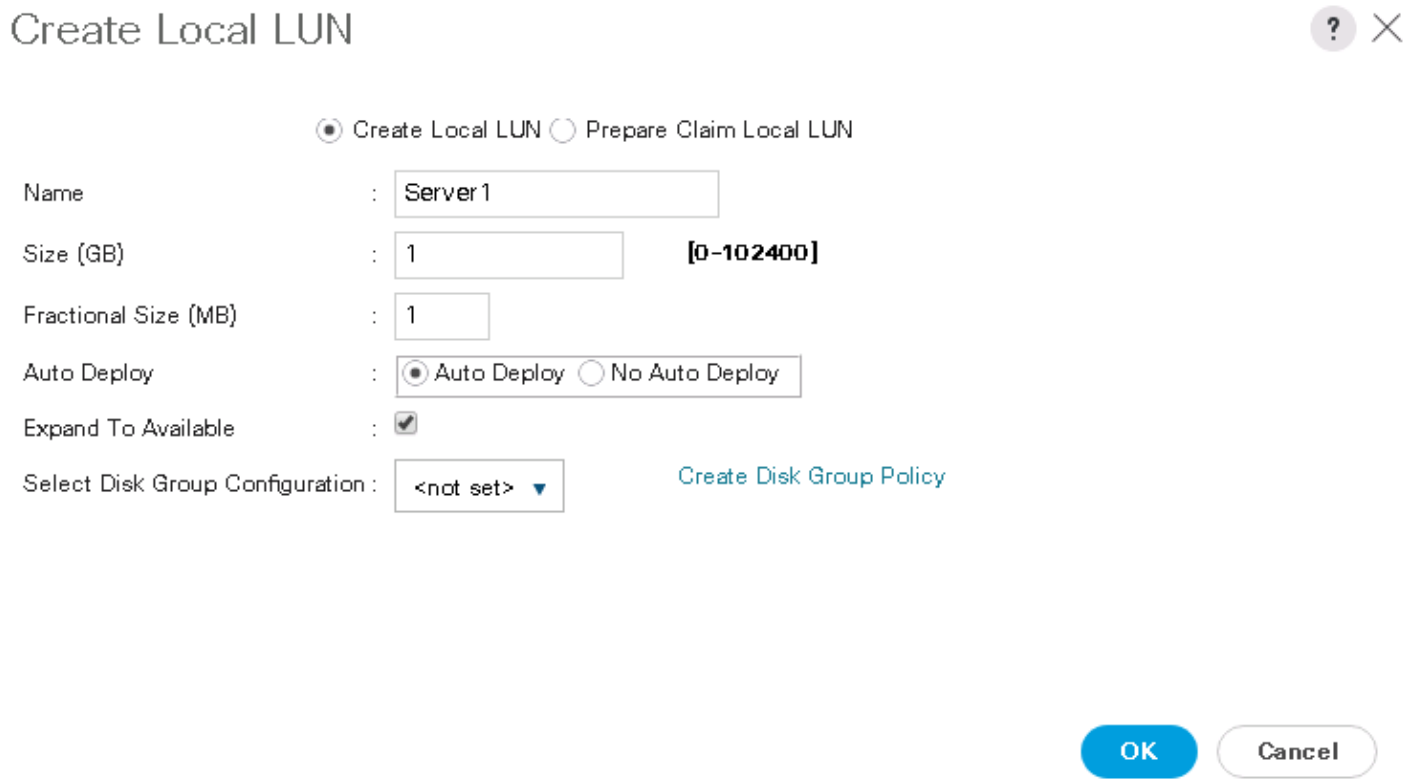


2. Similarly, obtain the disk ID numbers for the boot disks for Server 2 (203 and 204) before proceeding to the following steps.

3. Storage Profile for the first Server:

a. Select **`Storage`** in the left pane of the UCSM GUI.

b. Go to Storage > Storage Profiles and right-click Create Storage Profile.
c. Type in Server1 in the **Name** field.
d. (Optional) Enter a description in the **Description** field.
e. Click **Add**.
f. Create Local LUN.
g. Type in **Swift-Server1** in the **Name** field.
h. Size (GB) = 1
i. Fractional Size (MB) = 0
j. Auto Deploy.
k. Select Expand To Available.

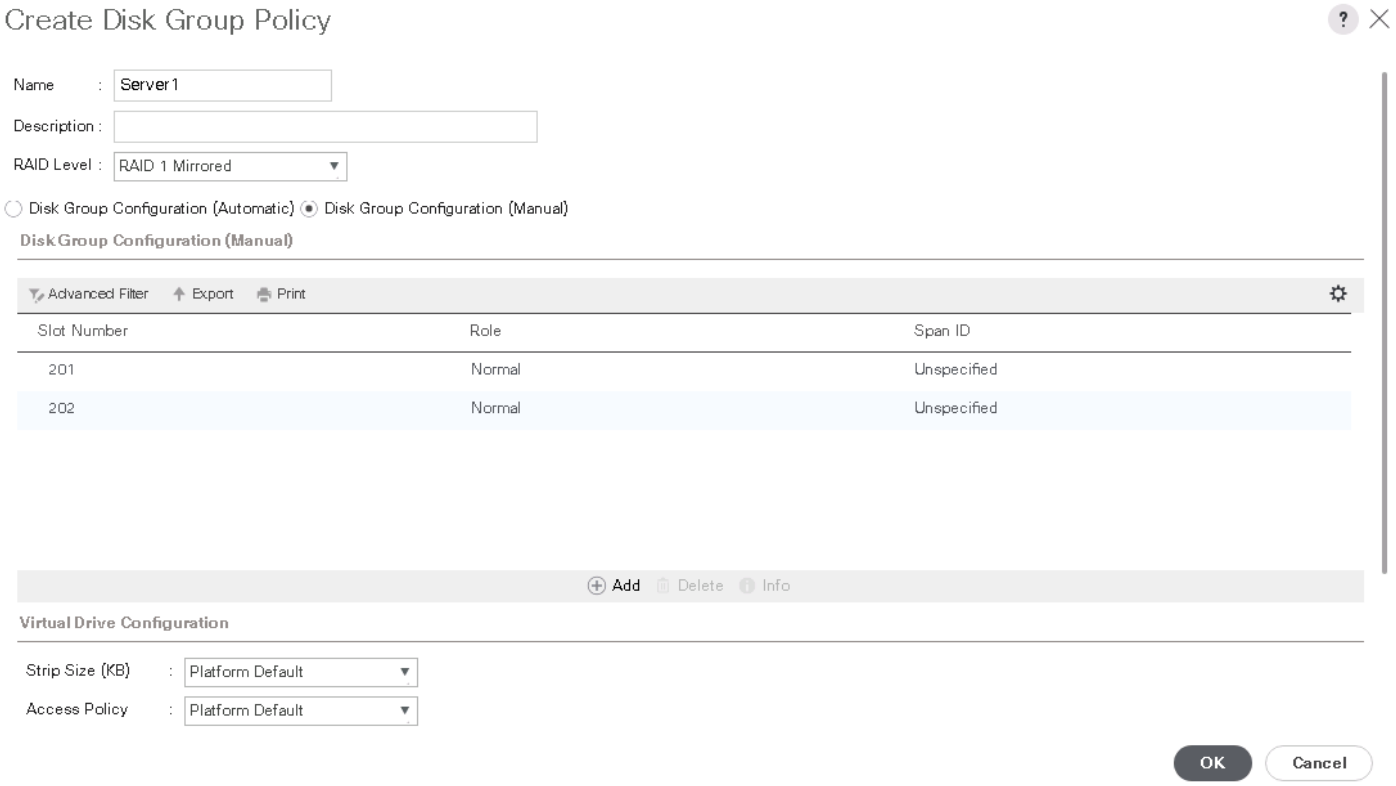


l. Click Create Disk Group Policy
m. Type in **Server1** in the Name field.
n. (Optional) Enter a description in the **Description** field.
o. RAID Level = RAID 1 Mirrored.
p. Select Disk Group Configuration (Manual).
q. Click **Add**.
r. Type in **201** for **Slot Number**.
s. Click **OK** and then again **Add**.

t. Type in **202** for **Slot Number**.

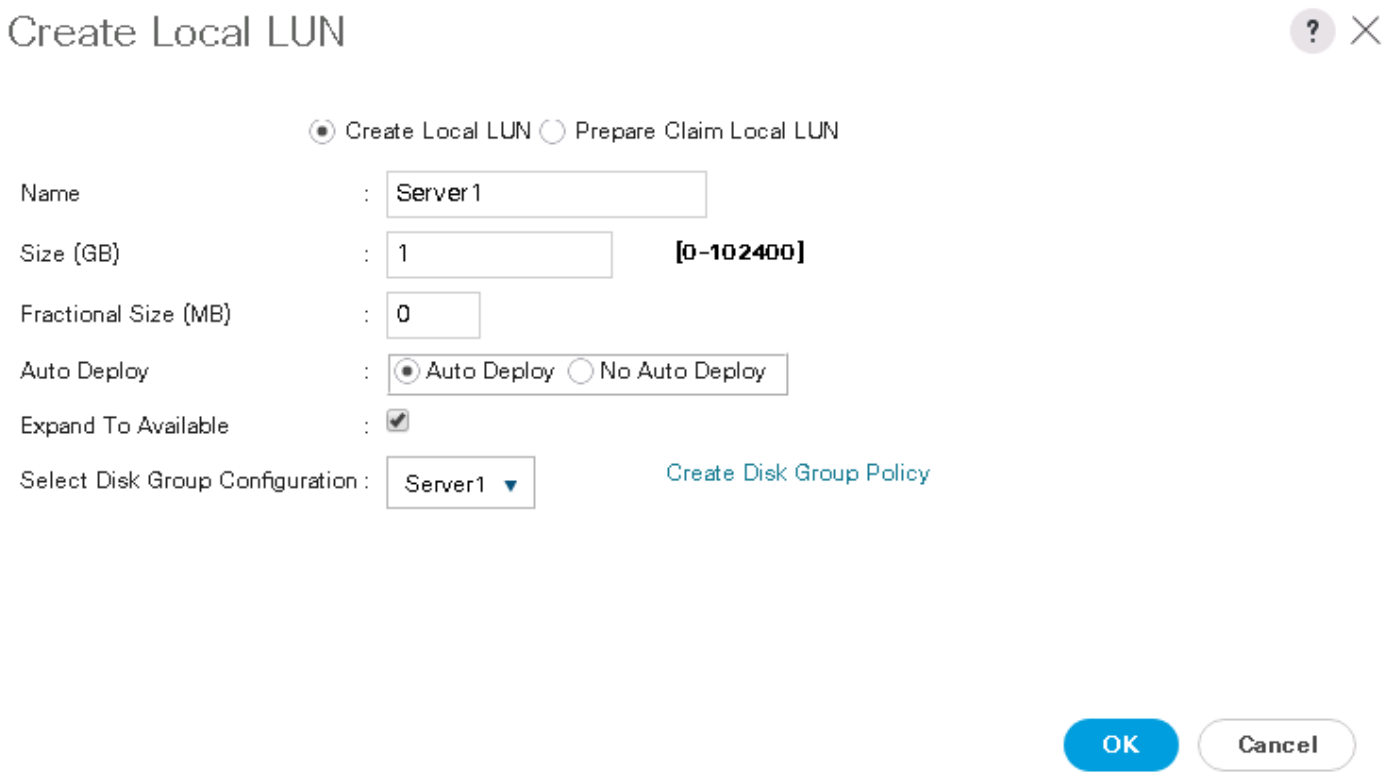
u. Leave everything else untouched.

v. Click **OK** and then **OK**.



w. Select your previously created Disk Group Policy for the Boot SSDs with the radio button under **Select Disk Group Configuration**.

x. Select Disk Group Configuration



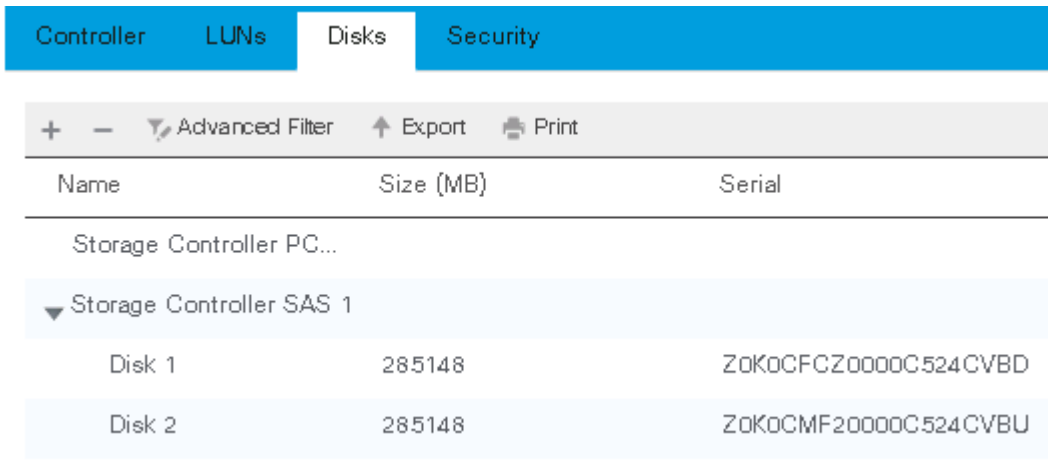
y. Click **OK** and then **OK** and again **OK**.

4. Storage Profile for the second Server:

   Follow the exact steps as above for creating a storage profile for the second server in the chassis. Add slot numbers as 203 and 204 while creating the Disk Group Policy say Server2. The Storage Profile Server1 is for the top Server and Server 2 for the bottom Server of S3260.

Storage Profile for Cisco UCS C220M4S Rack Server

1. Create another Storage profile for Swift Controller Boot Disks. Identify the disk numbers from Server Inventory and Create another Storage profile say Control following the similar steps as above.



Create Service Profile Templates

Creating Service Profile Templates for Cisco UCS S3260 Storage Servers

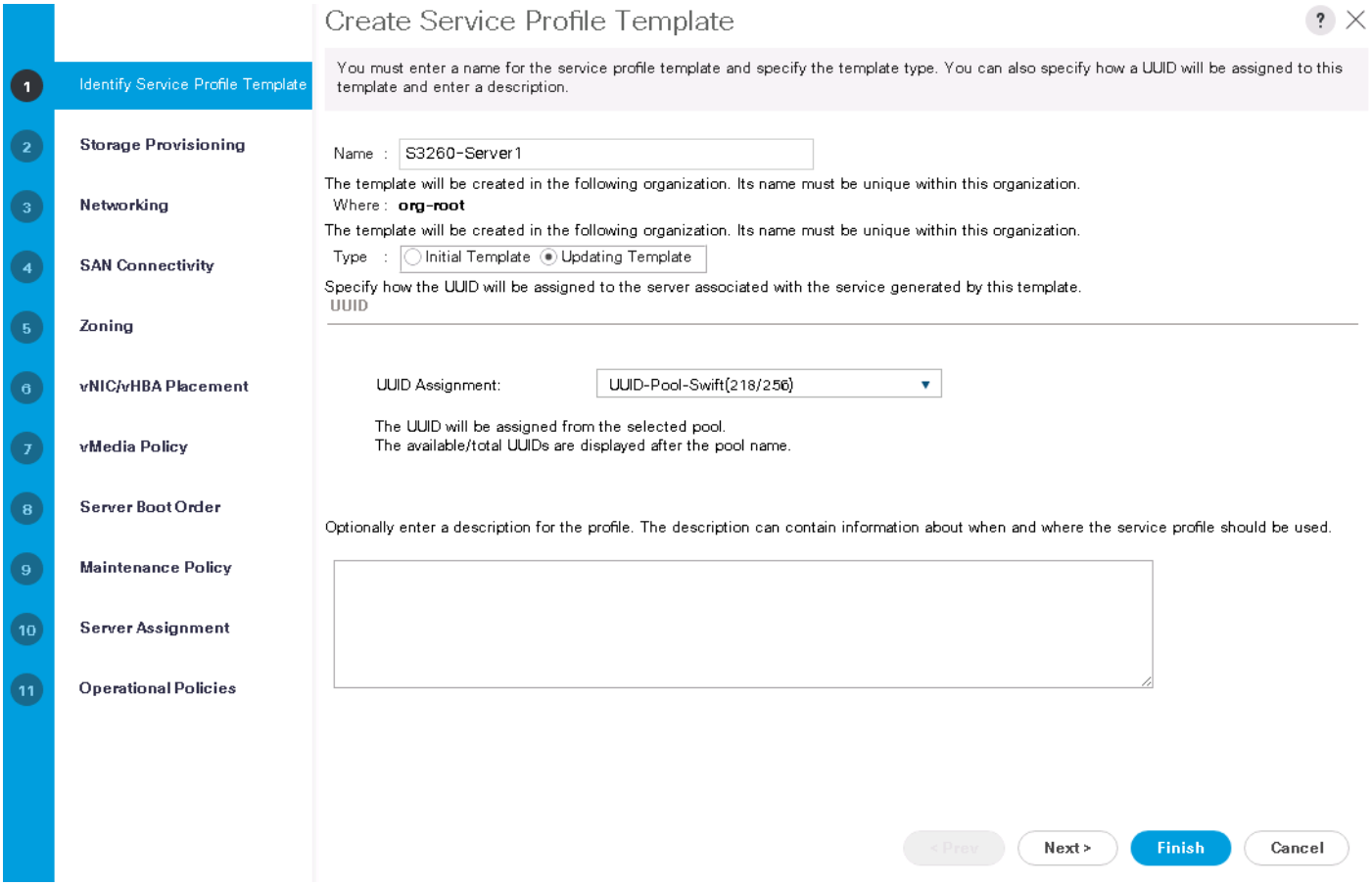
Depending on whether you will be deploying a single or a dual node configuration you may need one or two Service profile templates for S3260 server.

A single service profile template with all the 56 disks should suffice in case of single node configuration. On the other hand you may need to two templates, one for the top and other for the bottom server in case of dual node configuration.

To create a Service Profile Template, complete the following steps:

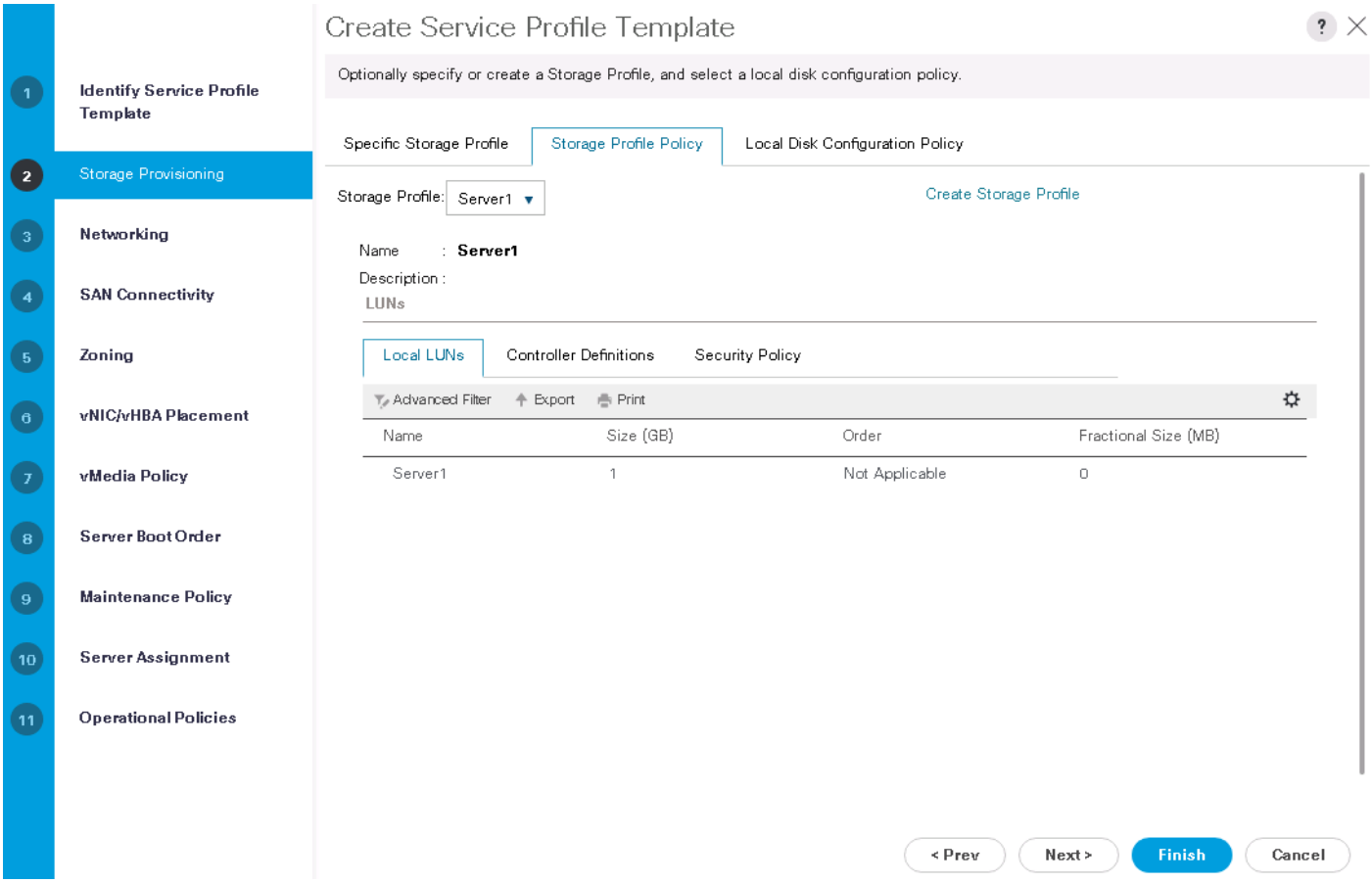
Identify Service Profile Template

1. Select Servers in the left pane of the UCSM GUI.
2. Go to Servers > Service Profile Templates > root and right-click Create Service Profile Template.
3. Enter the Name of the template say S3260-Server1.
4. Check as Updating Template.
5. The UUID pool created earlier for Swift and click Next.



Storage Provisioning

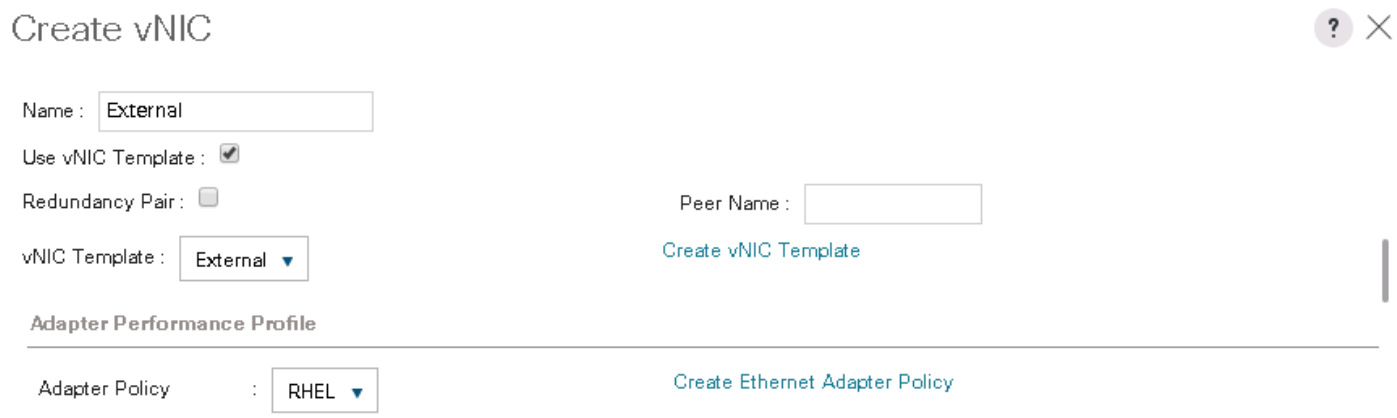
1. Go to Storage Profile Policy Tab and select the Storage profile created. For this particular Storage Template for the Top Servers select server1 created earlier and then click Next.



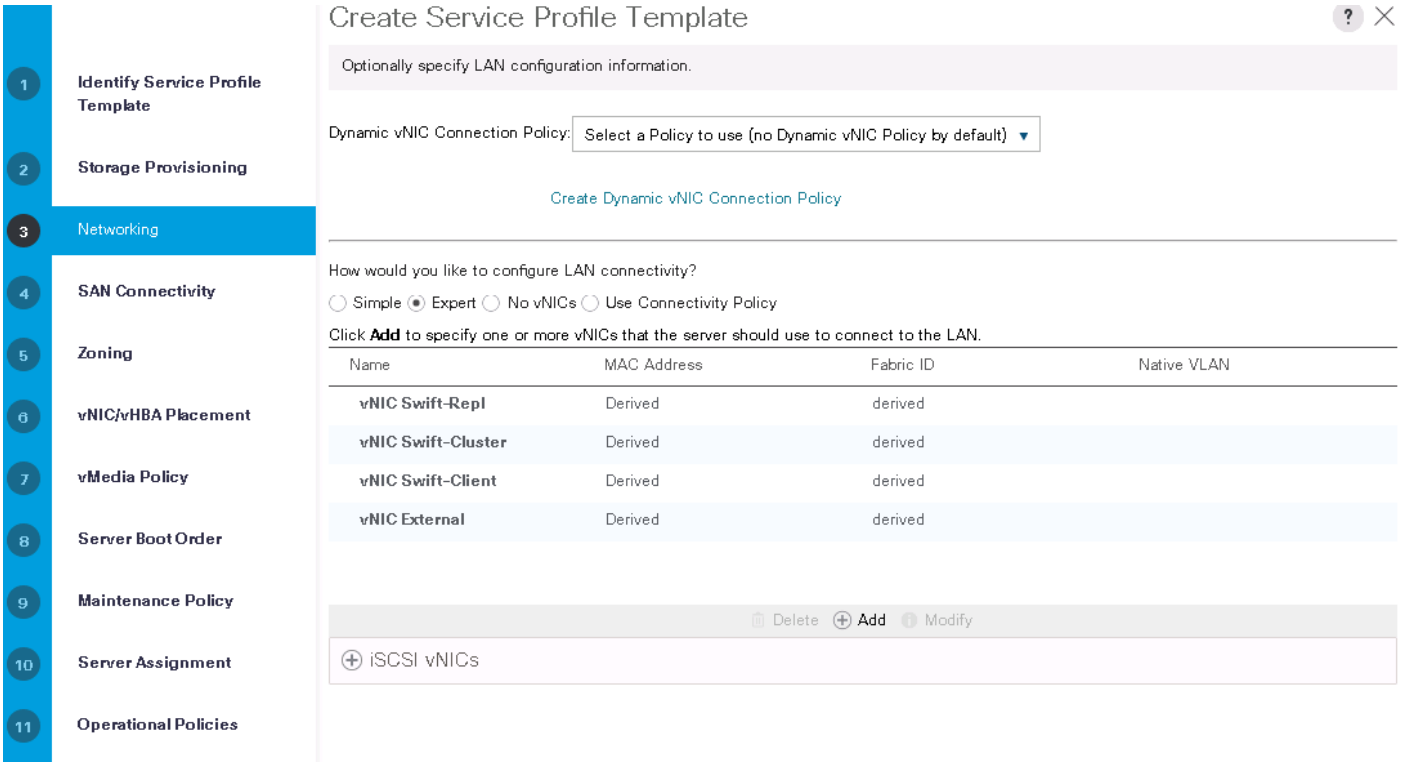
Networking

In the Networking tab, complete the following steps:

1. Select the expert Mode and Click Add for Creating vNIC's.
2. Enter, External in the name field and check the vnic template flag.
3. Select name of the vNIC Template as 'External' created earlier.
4. Select the adapter policy 'RHEL' created earlier for adapter policies.
5. Click OK.



6. Add other Ethernet Interfaces, the Outward facing, Cluster and Replication to the list as below before clicking Next.



Ignore SAN Connectivity and Zoning by clicking Next.

SAN Connectivity

1. Select No vHBA’s in SAN Connectivity tab.

How would you like to configure SAN connectivity?

◯ Simple ◯ Expert ◉ No vHBAs ◯ Use Connectivity Policy

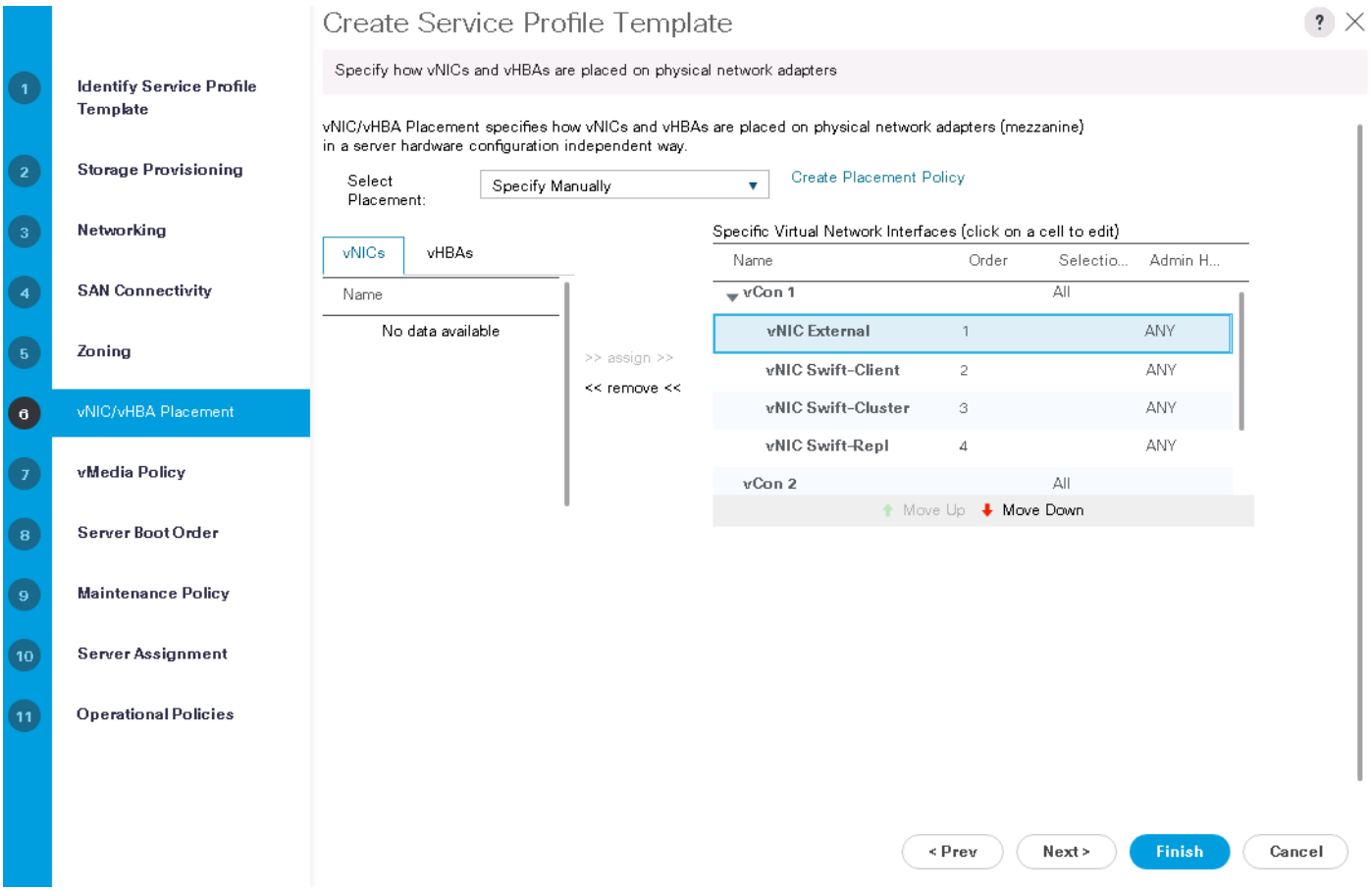
This server associated with this service profile will not be connected to a storage area network.

Zoning

1. Click Next and ignore Zoning.

vNIC/vHBA Placement

1. In the Select Placement Drop Down, Specify Manually.
2. Highlight vCon1 and then assign the interfaces in the following order – External, Swift-Client, Swift-Cluster, Swift-Repl and then click Next.



Server Boot Order

Select the Boot Policy Swift-Boot Created earlier and click Next.

Maintenance Policy

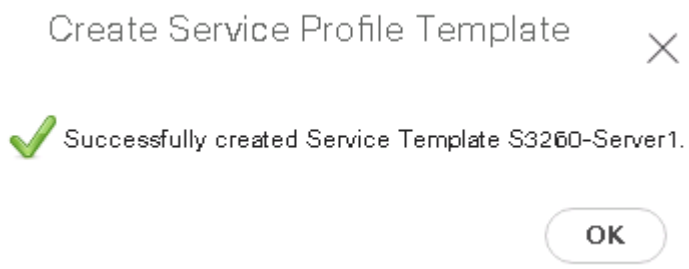
Select maintenance Policy Server_ACK in the drop down, that was created earlier. Click Next.

Server Assignment

Leave everything as default and click Next.

Operational Policies

Select Power Control Policy NO-POWER-CAP create earlier and finish the configuration.

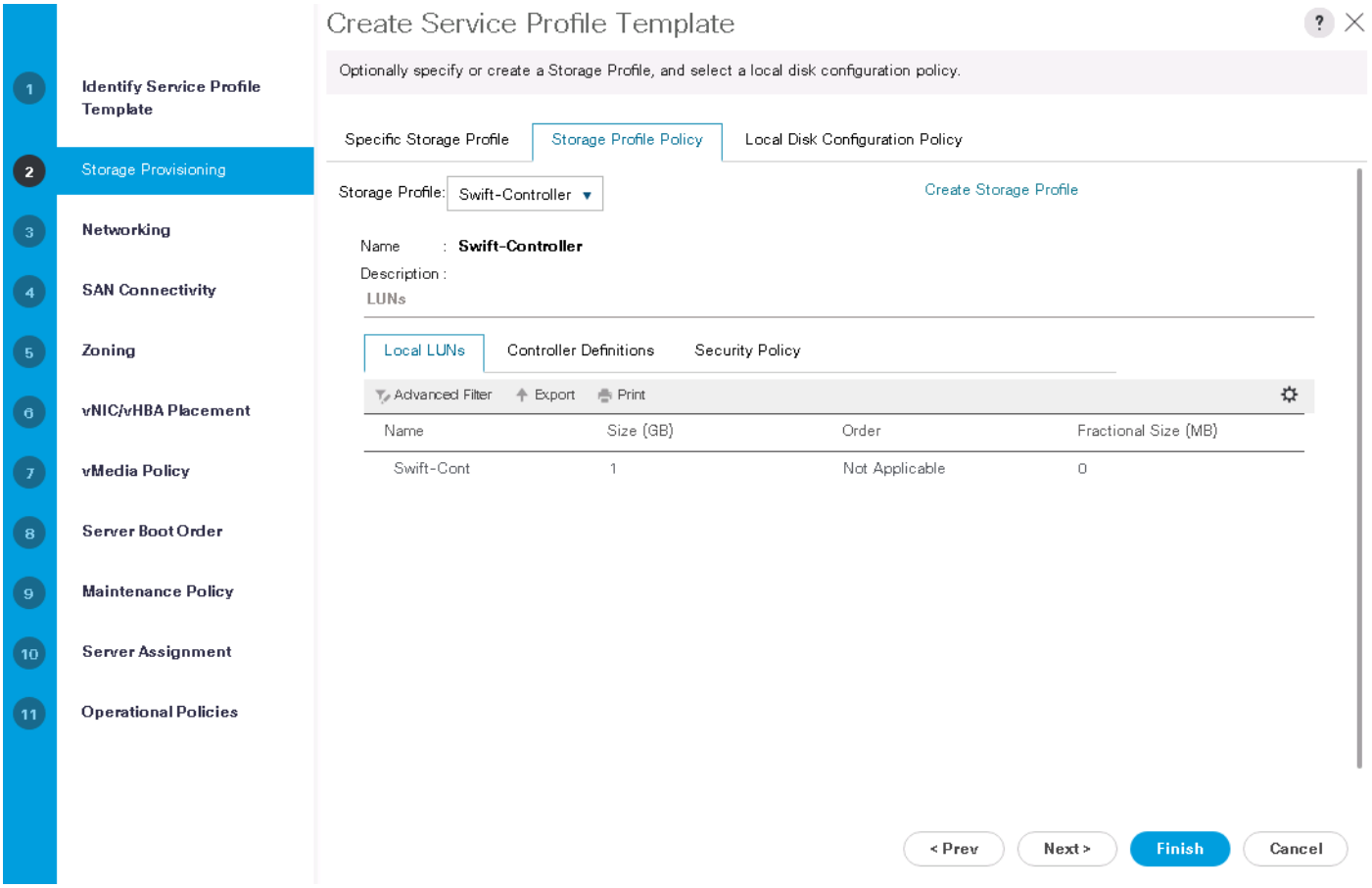


In case of Dual Node Configuration you may have to create one more template S3260-Server2 that attaches Server2 Storage Policy (with lun id's 203 and 204).

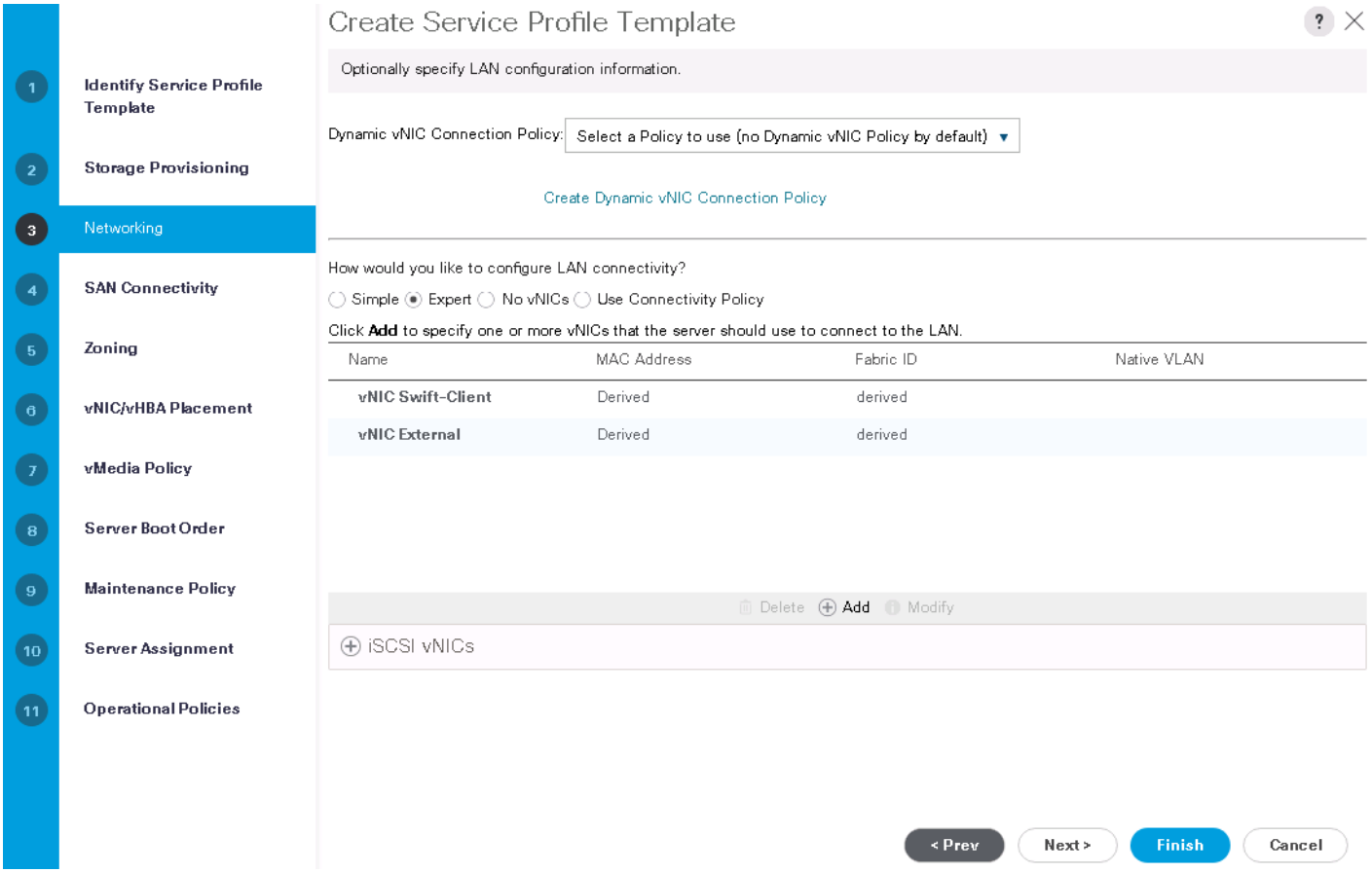
Create Service Profile Template for Cisco UCS C220 M4S

The Service Profile for the Rack-Mount Servers for Swift Controllers (Active and Standby) is very similar to the above created for the S3260. The only differences are with the Storage Profiles, Networking and vNIC/vHBA Placement. The changes are listed here:

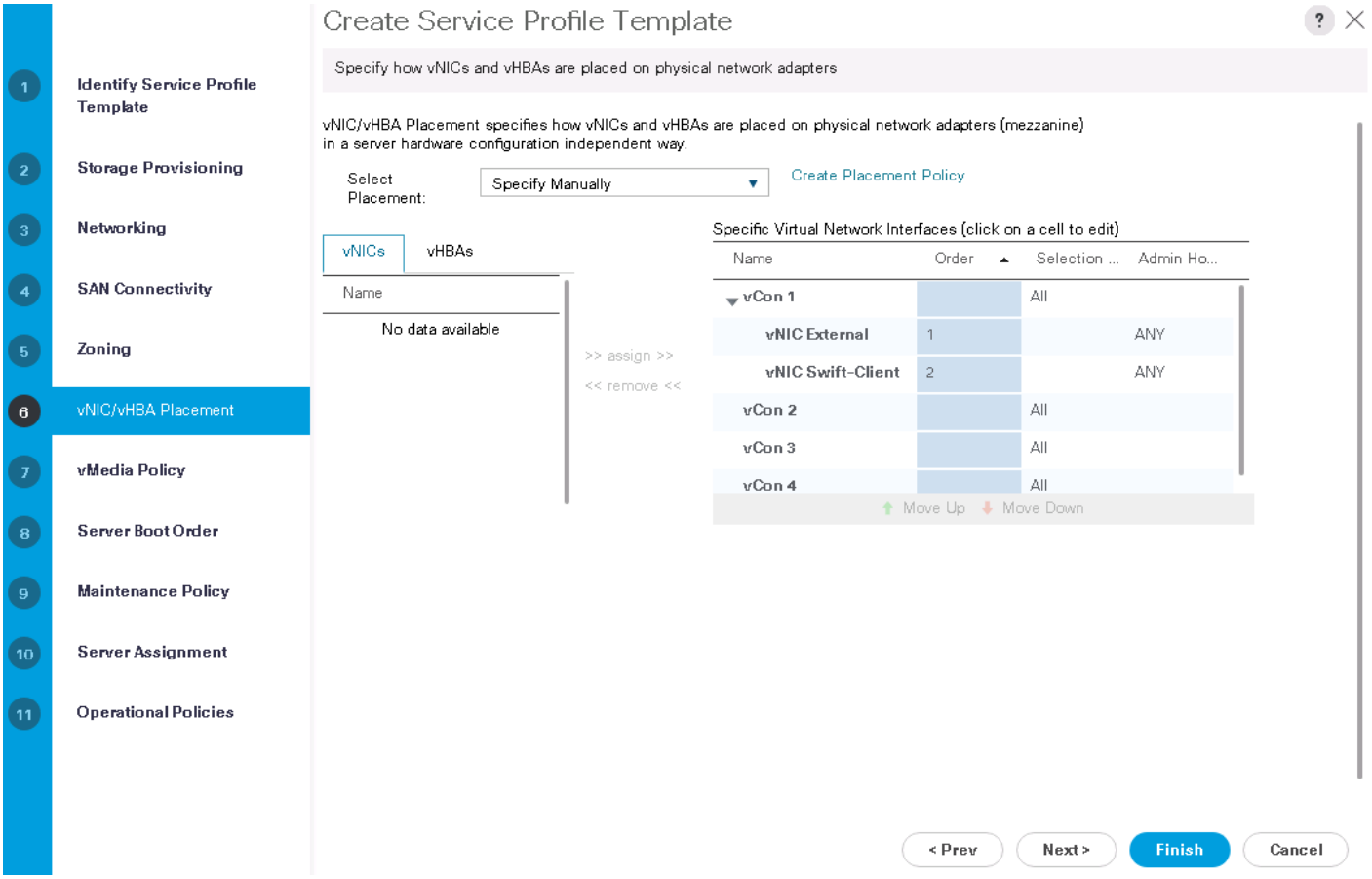
1. In the **`Storage Provisioning`** tab choose the appropriate Storage Profile for the Cisco UCS C220 M4S you created earlier.



The Networking tab will have only two Interfaces one for the nodes External communication and the other for outward facing network. In the setup used for the CVD, the outward facing network made available to the controller though this is not necessary.



The vNIC/vHBA Placement tab will list the first interface as external and the second interface as the outward facing network.



2. Walk through the remaining tabs similar to S3260 by adding Maintenance Policies and complete the creation of Service Profile Template C220.

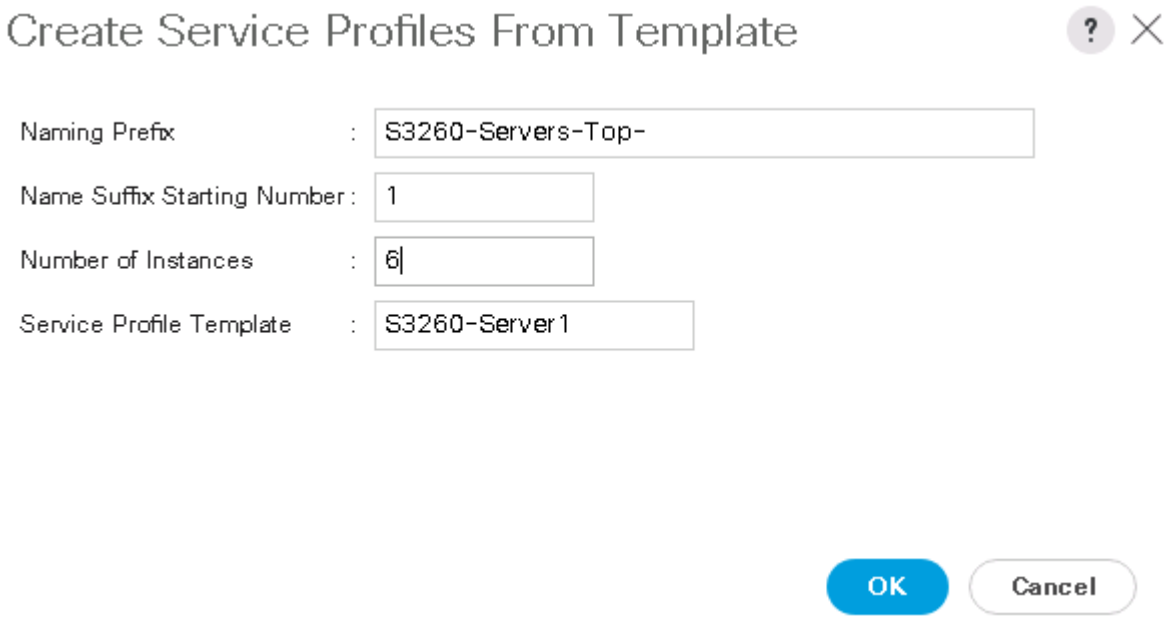
Summary

There will be three service profile templates, one for each server of S3260 and another for SwiftStack Controllers. In case of a single node configuration, you should have only two Service Profile templates; one for S3260 and the other for Controller. If there are other blade or Rack mounted servers that you would like to connect to these SwiftStack servers acting as clients, create the necessary Service Profile templates and profiles.

Create Service Profiles from Templates

Create Service Profiles for S3260 Storage Servers

1. Select Servers from the left pane of the UCSM GUI.
2. Go to Servers > Service Profiles > root and right-click Create Service Profiles from Template.
3. Similar to the above create one Service Profile for Active Controller and the other for Standby Controller.
4. Add the name, starting prefix and Number of instances and select the Service Profile Template S3260-Server1 created above.



8. Repeat these steps for the second server for the dual node configuration.

Create Service Profiles for Cisco UCS C220 M4 Swift Controllers

1. Select **Servers** from the left pane of the UCSM GUI.
2. Go to Servers > Service Profiles and right-click Create Service Profiles from Template.
3. Add the name, starting prefix and Number of Instances as 1.

Summary

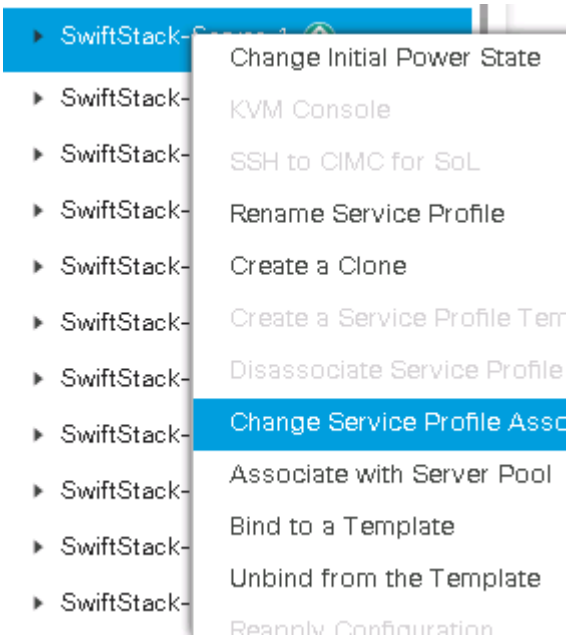
You will have 6 Service Profiles one for each Chassis in a Single Node Configuration.

You will have 12 Service Profiles two for each Chassis in a dual Node Configuration.

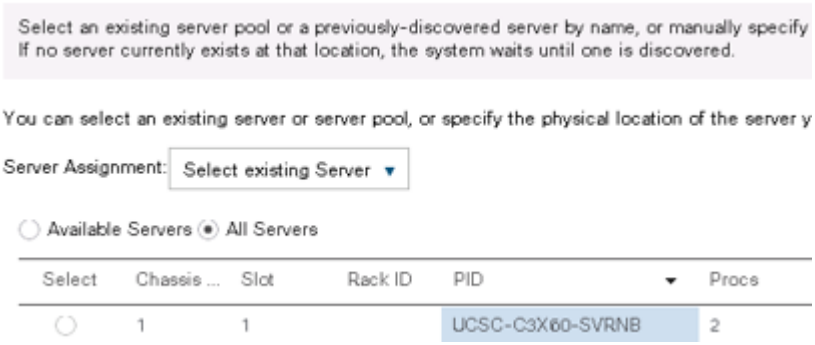
You will have 2 Service Profiles for the SwiftStack controllers, one for Active and the other for Standby.

Associate Service Profiles to the Servers

1. Select **Servers** from the left pane of the UCSM GUI.
2. Go to Servers > Service Profiles and right click on the service profile that you would like to associate with.



3. Click Change Service Profile Association and the next page select existing server and choose the appropriate server in the chassis.



4. Repeat these steps for all the SwiftStack Storage Servers and the Swift Controllers.

Post Cisco UCS Configuration Health Checks

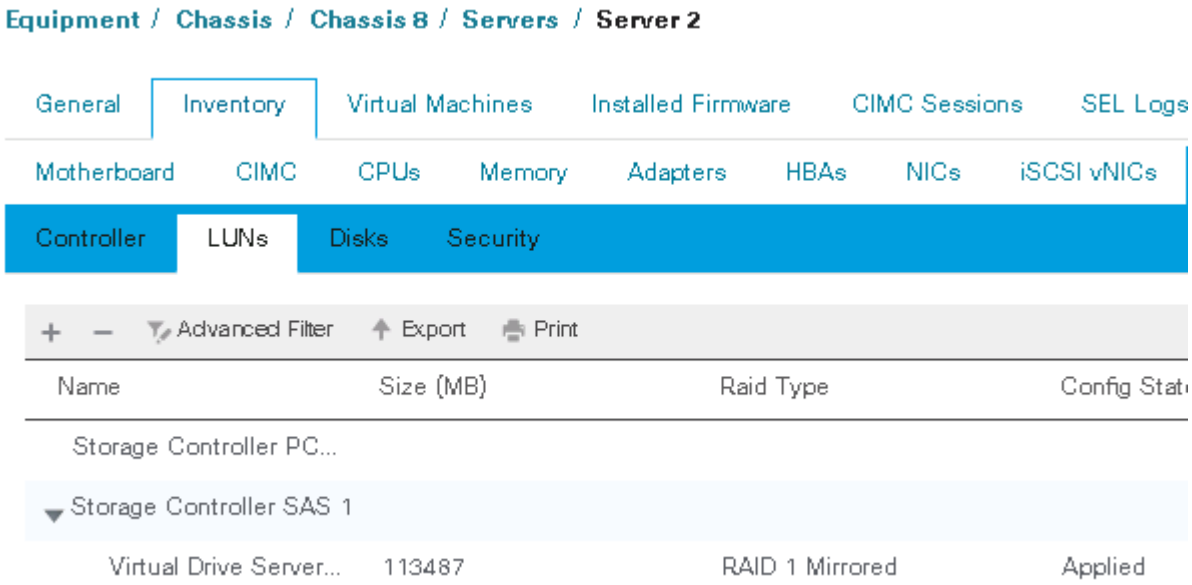
The following is a list of health checks to be carried out before Installing Operating System and SwiftStack software.

1. Go to Equipment Tab, Fabric Interconnects, Ethernet ports and check the status of Server and Network Ports. Alternatively open a putty session to Fabric Interconnect, connect nxos a (or b) and check the status of ports with show interface brief.

2. Repeat this step by logging into Nexus Switches and check the status of ports and port-channels if any.

3. For each server node, the S3260 Storage Server Node(s) or the Swift Controller check the Inventory tab for CPU and Memory details and Critical Faults.

4. Under the same Inventory tab for the server under storage tab and then in Luns check the status of the boot lun. After successful application of the service profile, this should show as RAID1 Mirrored and in applied status as below.



5. Under the same Inventory tab and in Disks, make sure that other SSD and HDD disks are available. If installing OS manually through Red Hat UI, you will select the Virtual Drive as shown in step 4 above and the rest if any jbods can be left as is. If automated through pxe install the other HDD and SSD's should be made as unconfigured too.

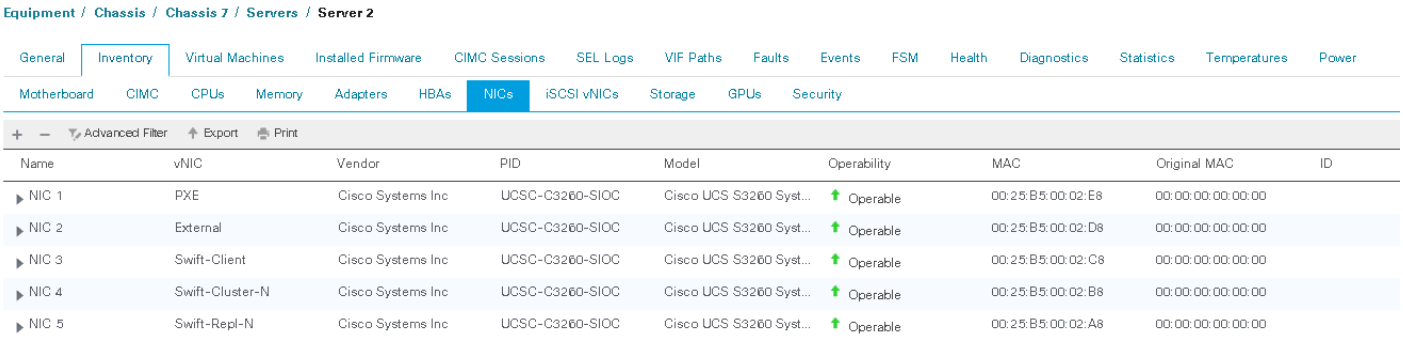
6. Under the same inventory tab check the number of NIC's are same as desired for that server. There should be a minimum of 4 NIC's for Storage Nodes.

Install the Operating System on the Swift Nodes

Install Operating System

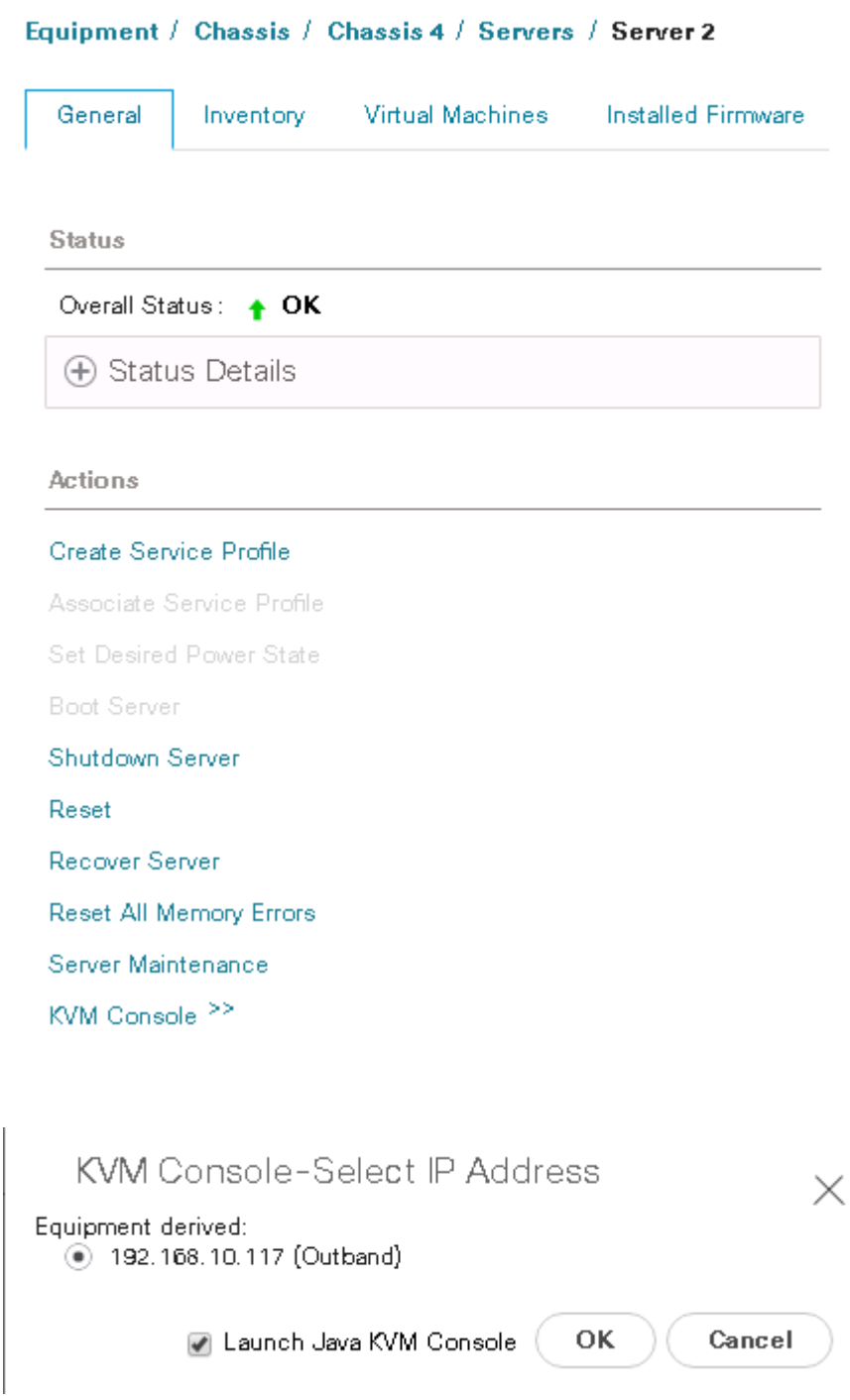
This section describes the Installation of the Operating System on the nodes. The installation procedure is the same for both the Controller and Server Nodes. The operating system can be installed either manually through the Red Hat Installer or through the kick-start installer. Make sure to have the appropriate MAC address for the interfaces and have the MAC addresses from Cisco UCS.

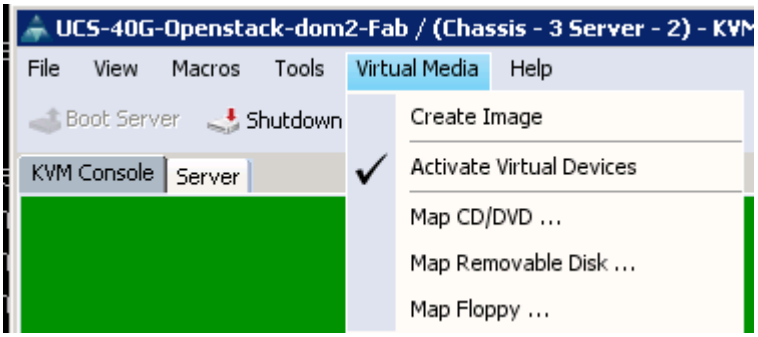
1. From Cisco UCS Manager, navigate to Server > Inventory and NICS to get the MAC address details for the servers.



2. Login to UCS Manager GUI.
3. Click the specific Server.
4. Launch the Java KVM Console.
5. Activate the Virtual Devices.
6. Map the ISO file.
7. Reboot the server.
8. Press F6 to select the boot option from CVD and launch the installer.

Red Hat Operating System 7.3 was installed on the test bed.


Equipment / Chassis / Chassis 4 / Servers / Server 2
General
Inventory
Virtual Machines
Installed Firmware
Status
Overall Status : OK
Status Details
Actions
Create Service Profile
Associate Service Profile
Set Desired Power State
Boot Server
Shutdown Server
Reset
Recover Server
Reset All Memory Errors
Server Maintenance
KVM Console >>
KVM Console-Select IP Address
Equipment derived:
192.168.10.117 (Outband)
Launch Java KVM Console
OK
Cancel



9. When the Installer opens, install the Red Hat Operating system 7.3. You may select the minimal install from the drop-down menu.

10. Create /home file system and reserve around 100GB for root file system to take care for /var/logs etc.

Post Operating System Installation on the Nodes

1. Check and ping the default gateway for North bound connection. This is needed to register the servers.

2. Register the server and attach subscriptions

```
sudo subscription-manager --release=7.3 register
(enter username/password when prompted)
sudo subscription-manager attach --pool=<pool name>
sudo subscription-manager repos --disable=*
sudo subscription-manager repos --enable=rhel-7-server-rpms --enable=rhel-7-
server-optional-rpms --enable=rhel-7-server-extras-rpms
yum clean all
```

3. Update hostname, proxies if any and resolv.conf

   hostnamectl set-hostname swiftstack-server1.cisco.com
   /etc/resolv.conf

```
domain cisco.com
nameserver <name server1>
nameserver <name server2>
nameserver <name server3>
options timeout:1 attempts:1
```

4. Update dns server and /etc/hosts as needed in your setup for name resolution.

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4

192.168.120.201 swiftstack-server1 swiftstack-server1.cisco.com
192.168.120.202 swiftstack-server2 swiftstack-server2.cisco.com
192.168.120.203 swiftstack-server3 swiftstack-server3.cisco.com
……
```

5. Update /etc/rhsm/rhsm.conf with proxy entries in case you are behind the proxy server.

6. Yum update the server.

   yum update –y

7. Install additional packages. Not all packages were needed but they were installed for operational activities.

```
yum install dstat net-tools sysstat wget ntp rng-tools bind bind-utils –y
```

8. Configure named.conf.

9. Bind-utils is installed for SwiftStack Load Balancer and it is not needed if not using SwiftStack load balancer.

10. Add an entry in /etc/named.conf as shown below:

```
zone "cisco.com" {
type master;
notify no;
file "cisco.com";
```

```
};
```

11. Create a file with the zone name, in this case as cisco.com and make entries as shown below:

```
;
; BIND data file for cisco.com
;

$TTL    604800
@       IN      SOA     swiftstack-cluster.cisco.com. swiftstack1-
cluster.cisco.com. (
                              1             ; Serial
                              1D          ; Refresh
                              1S           ; Retry
                              4W       ; Expire
                              1D        ; Default TTL
)

@ IN      NS      swiftstack-cluster.cisco.com.

;
swiftstack-cluster IN      A       192.168.120.220
swiftstack-cluster IN      A       192.168.120.221
swiftstack-cluster IN      A       192.168.120.222
swiftstack-cluster IN      A       192.168.120.223
swiftstack-cluster IN      A       192.168.120.224
swiftstack-cluster IN      A       192.168.120.225
swiftstack-cluster IN      A       192.168.120.226
swiftstack-cluster IN      A       192.168.120.227
swiftstack-cluster IN      A       192.168.120.228
swiftstack-cluster IN      A       192.168.120.229
swiftstack-cluster IN      A       192.168.120.230
swiftstack-cluster IN      A       192.168.120.231
;
```

Here `swiftstack-cluster` is the name of Load balancer to which the clients will connect and they will be round robin amongst the SwiftStack PACO nodes on Outward facing network.

12. Restart the named daemon:

```
systemctl restart named.service
systemctl enable named.service
```

13. Configure rngd.

Rngd is used to speed up the build and config as it builds the entropy on the Controller. rng-tools not needed in case you select ‘fake entropy’ with the controller install.

```
Update /etc/sysconfig/rngd as
EXTRAOPTIONS="-i -o /dev/random -r /dev/urandom -t 10 -W 2048"
```

14. Restart rngd service.

```
systemctl restart rngd.service
systemctl enable rngd.service
```

15. Configure ntp.

16. Update ntp server address in /etc/ntp.conf and enable ntp service.

```
systemctl restart ntpd.service
systemctl enable ntpd.service
```

17. Update enic drivers.

It is recommended to update the enic drivers on the host certified for the UCSM version.

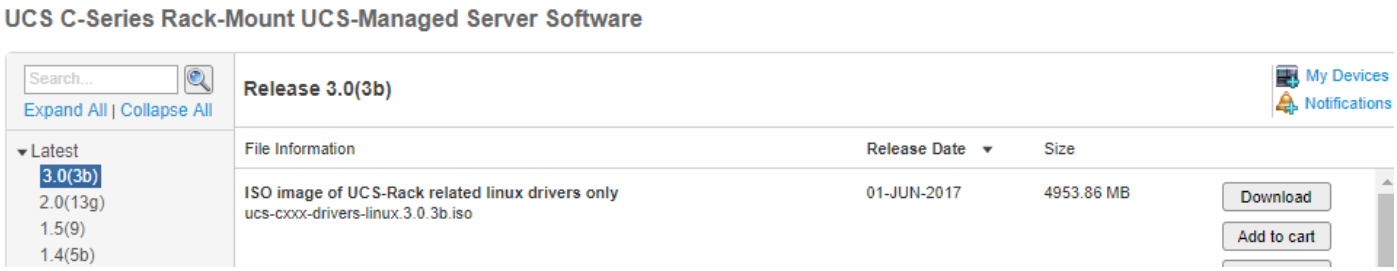
18. Login to http://software.cisco.com.

19. Click Software Download.

20. In the products page select Unified Computing.

21. Select UCS ‘C’ series RACK-Mount UCS Managed Server Software.

22. Download the linux drivers for your UCSM version.



23. Mount the iso file and extract the enic drivers.

mount ucs-cxxx-drivers-linux.<version>.iso /mnt
cd /mnt/Network/Cisco/VIC/RHEL/RHEL7.3/

24. Copy the enic driver to say /tmp and install the enic driver as rpm –ivh <rpm name>

25. Run modinfo to check that the drivers are in place.

```
[root@swiftstack-server2 ~]# modinfo enic
filename:       /lib/modules/3.10.0-514.21.1.el7.x86_64/weak-
updates/enic/enic.ko
version:        2.3.0.31
license:        GPL v2
author:         Scott Feldman <scofeldm@cisco.com>
description:    Cisco VIC Ethernet NIC Driver
```

```
rhelversion:    7.3
```

Health Checks

After completing the steps detailed in the previous section on all the servers, make sure that the connectivity works amongst the servers and if using jumbo frames do a ping test with a jumbo packet size.

Also make sure by logging to each server and checking through /proc/partitions that both HDD ad SSD disks are visible to the operating system. If any of these are left out in 'unconfigured' state, turn to JBOD mode either through Cisco UCS Manager or through storcli.

Install SwiftStack Software

Pre-installation Checks

Make sure that SSH Trust is enabled amongst all the controller and storage nodes.

Below is an example of setting up the ssh-trust:

```
ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
…..

Use ssh-copy-id to copy the files.
ssh-copy-id root@swiftstack-server2
```

Establish trust between all the nodes before proceeding with the installation.

Install On-Premise Controller Software

There are two modes of installation for SwiftStack controller. The controller can be hosted on SwiftStack site and can be installed via http://platform.swiftstack.com. You can also install Controller On-Premise and the current scope in this CVD is limited to On-Premise controller only. Please Contact SwiftStack for getting the On-Premise Controller software and verify your hardware requirements.

For detailed information about the software, log in to portal.swifstack.com and check for on-premise controller installation under the Admin section. Only high-level installation steps are presented.

Install Software

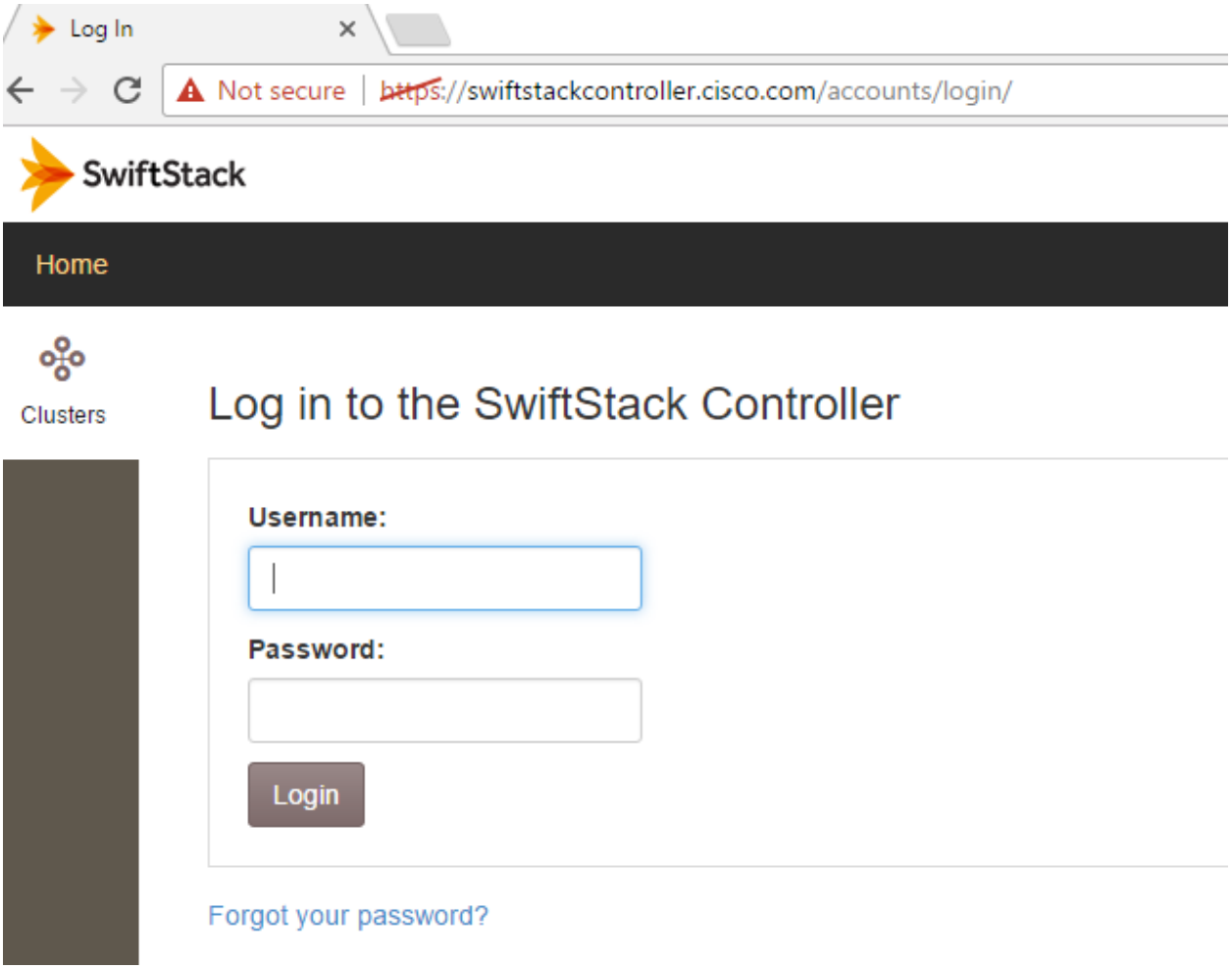
Obtain the Installer software and run the script in a linux shell as shown below:

```
root@swiftcontroller ~]# ./SwiftStack-Controller-5.2.0.2-installer.sh
Extracting SwiftStack Controller 5.2.0.2
.....................................................................
997580 blocks

…..
…..
....

Installing SwiftStack Controller...
Preparing...                          ########################################
Updating / installing...
swiftstack-controller-5.2.0.2-1.el7   ########################################
**** SwiftStack Controller install details will be logged to /tmp/install-2017-
04-06-14:01:57.log
```

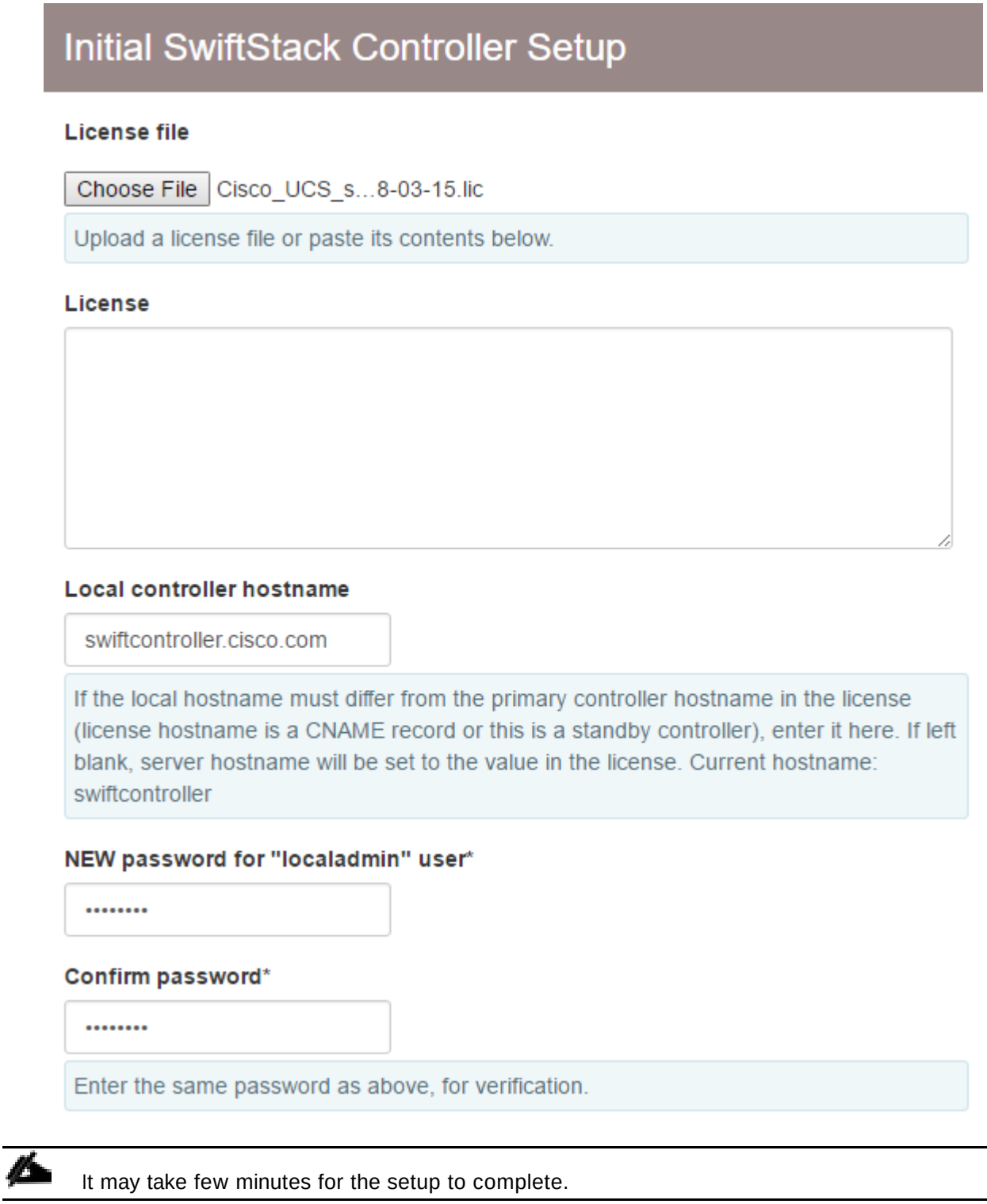
```
Controller install succeeded!

You must now complete setup by pointing a web browser at this server
using HTTPS and the default port of 443.  E.g.

    https://172.22.166.216/

Log in with the username "localadmin" and the default password, "password"
```

Post Software Install Configuration

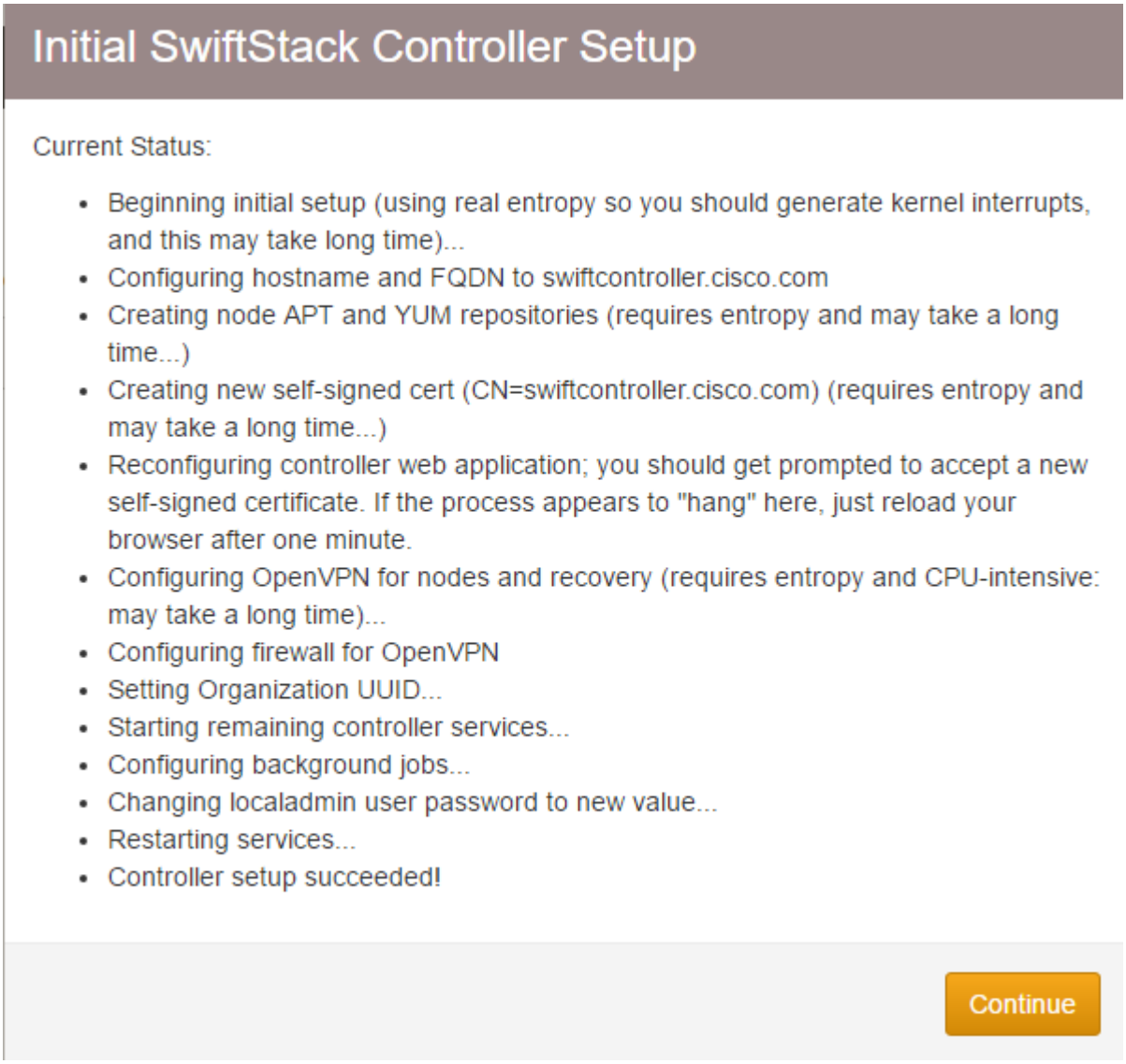
1. Log in to the URL pointed out after the install, with the user as localadmin and default password.



2. Enter license key obtained from SwiftStack and enter the hostname and new password to proceed. You may leave the other values to default and click Submit.



It may take few minutes for the setup to complete.



3. After re-logging with the new password, the system will prompt you to add the nodes and create the cluster. It may display as shown below to run the curl command on the storage nodes:

```
curl https://swiftcontroller.cisco.com:443/install | sudo bash
```

Install SwiftStack Software on Storage Nodes

Before running the installer software make sure that SSL certificates are installed. They could either be from commercial CA or Self-Signed.

Self-Signed certificate will be generated on the controller node and needs to be copied to all the server nodes.

1. On Controller Node:

```
[root@swiftcontroller ~]# cd /opt/ss/etc/
[root@swiftcontroller etc]# ls -l ssman.crt
```

```
-rw-r--r-- 1 root 668 Jun  6 11:16 ssman.crt
```

2. Copy this certificate to all Server nodes.

```
scp ssman.crt root@swiftstack-node:/etc/pki/ca-trust/source/anchors/
```

3. Once copied, run update-ca-trust extract as root user on storage node.

```
[root@swiftstack-server28-2 .ssh]# update-ca-trust extract
[root@swiftstack-server28-2 .ssh]#
```

4. Run the Curl command on the storage node.

```
curl https://swiftcontroller.cisco.com:443/install | sudo bash
```

After completing this command, the system will print the claim URL. This can also be printed by running ssclaimurl command on the server as shown below:

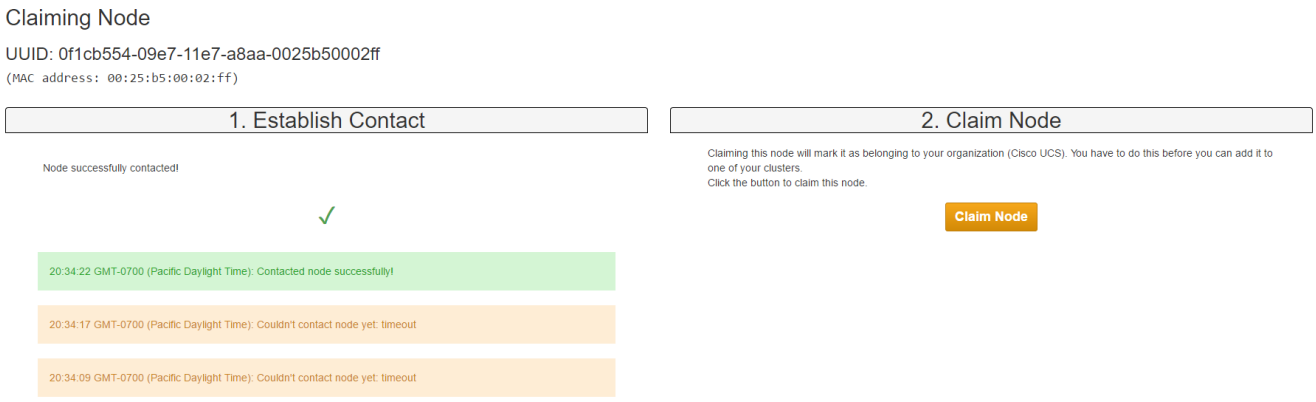
```
[root@swiftstack-server2 ~]# ssclaimurl
+-------------------------------------------------------------------------------+
|                                                                               |
| Your claim URL is:                                                            |
| https://swiftcontroller.cisco.com:443/claim/9723dbc2-61e1-11e7-8fcb-0025b500024f |
|                                                                               |
+-------------------------------------------------------------------------------+
```

5. Run the Curl Command on each storage node and get the ssclaimurl's. The nodes have to be claimed through http request to the controller.

Configure SwiftStack Controller for Nodes

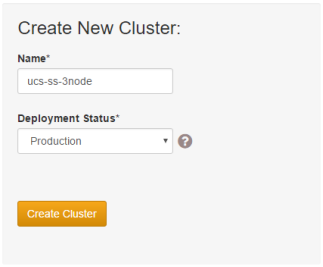
1. Claim Nodes

2. Run the ClaimURL in a browser to claim the nodes.



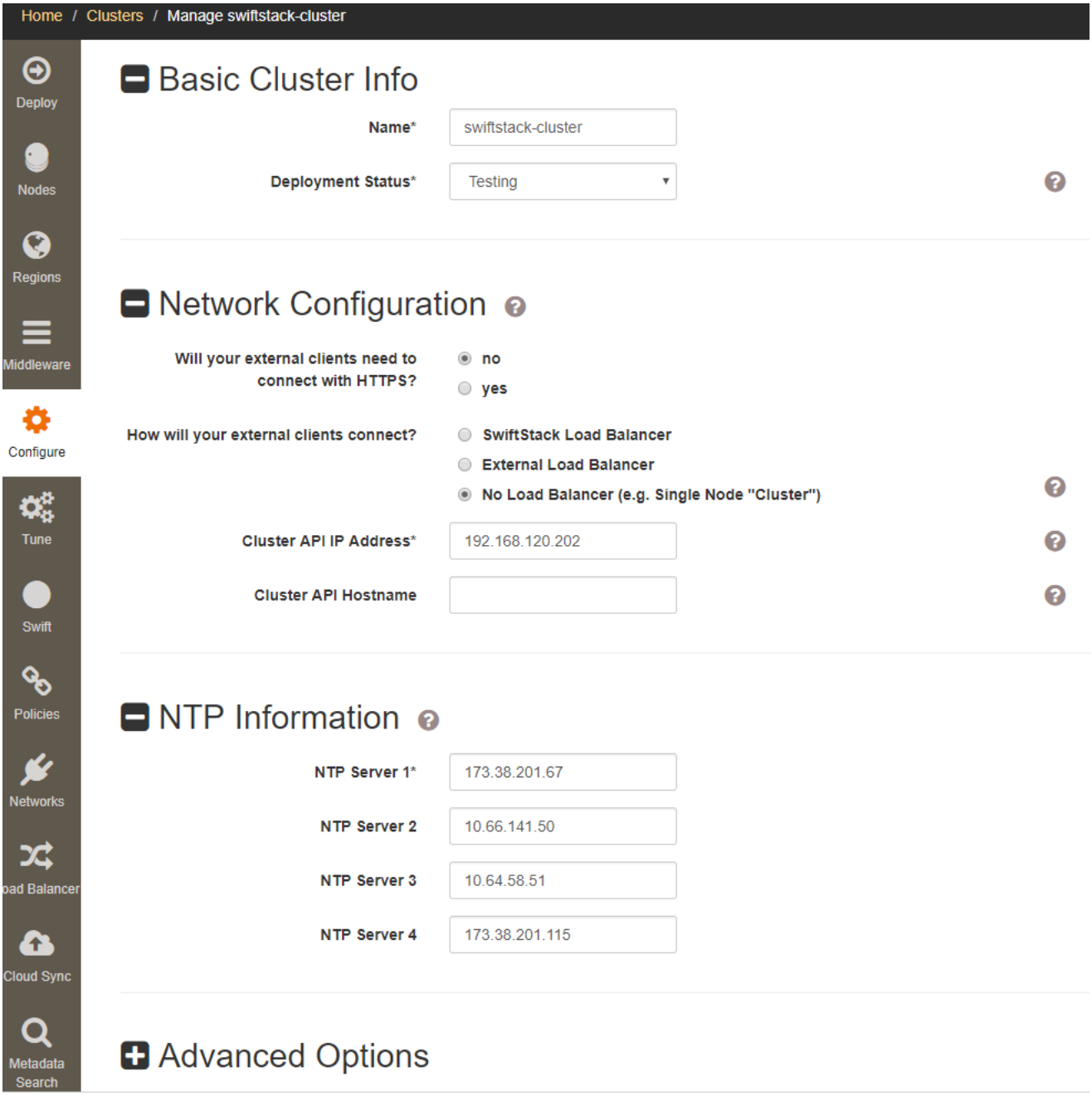
Create and Configure the Cluster

1. Enter the cluster name and create the cluster.



Configure the Basic Settings

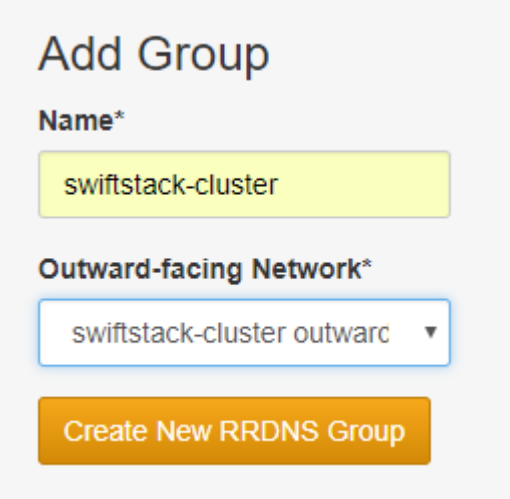
1. Click the Configure button and configure the basic settings as shown and submit the changes.



In the screenshot above, for Network Configuration, ‘No Load Balancer’ is used. In case you would like to use a SwiftStack load balancer, enter the name of the Load Balancer configured. For details on configuring SwiftStack Load Balancer, please refer to the SwiftStack documentation for details. A snippet is provided below.

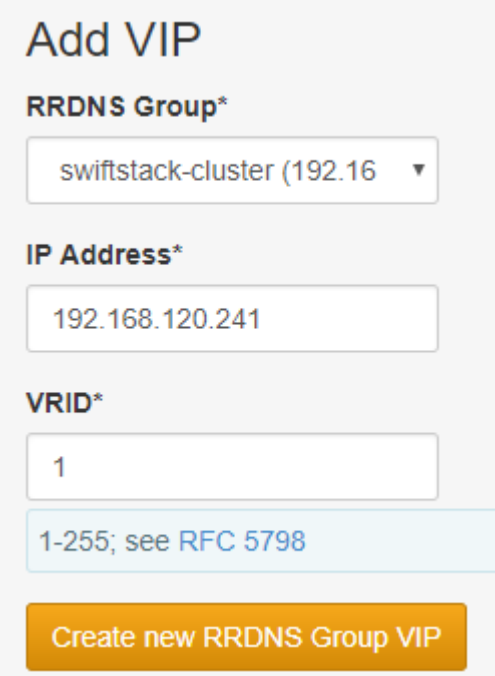
1. Click Home > Clusters > Manage <name of the Cluster>.

2. Click Load Balancer and Add Group name and the outward facing network.

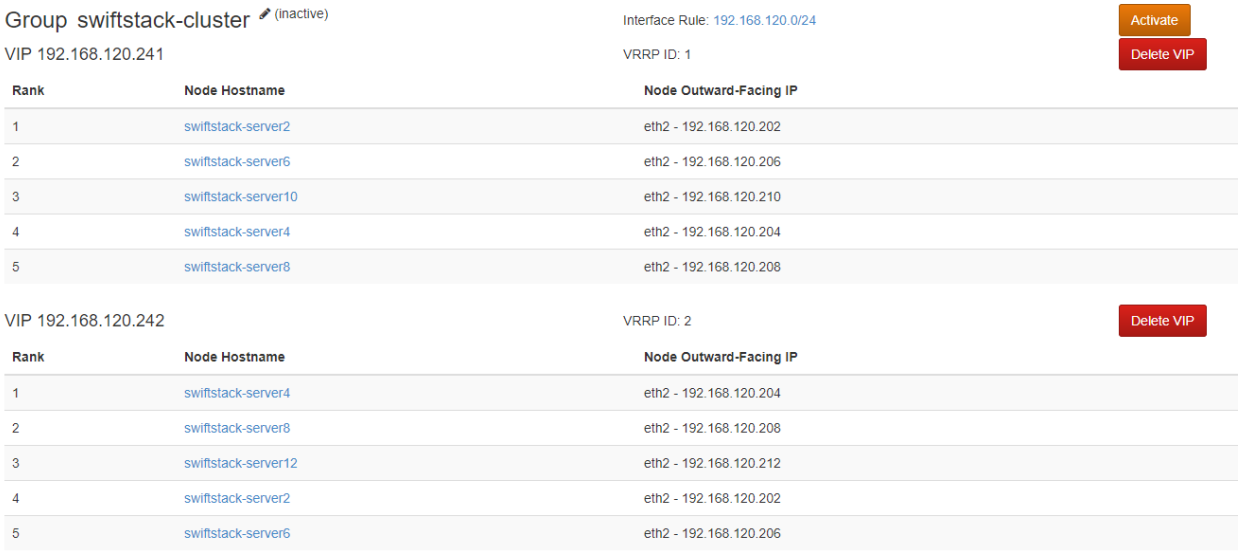


In this example, the name is same as the cluster name configured in bind in your /var/named/<zone file>.

3. Click Create New RRDNS Group, and provide a VIP with IP address again per your zone file and click Create new RRDNS Group VIP.



4. Add another VIP and repeat for all the VIP's per your named zone file. This should create multiple entries as shown below.

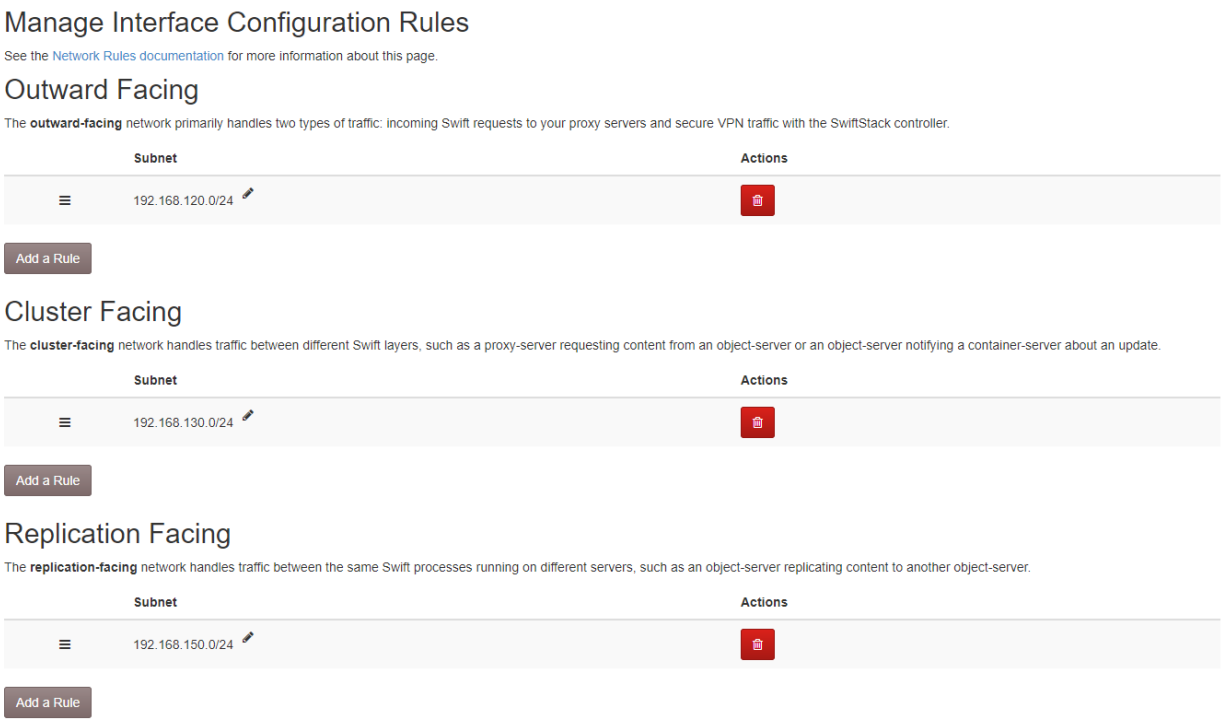


Repeat this procedure for as many VIP's as nodes in your cluster and Activate the Group. In the network page make sure to enter the name of this group for the SwiftStack Load balancer.

Once configured and configuration deployed, this Load Balancer should be pingable from all the server nodes.

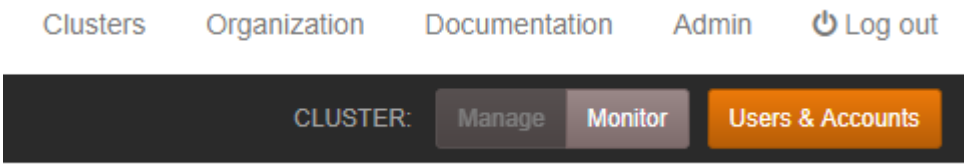
Manage Interface Configuration Rules

1. Click Networks and provide the three network configurations and save them as rules. Every node ingested will inherit these rules by default.



Create User and Accounts

1. Click User and Accounts on the right side of the page to create Swift users. These are Swift Cluster accounts and a minimum of one account is needed for the cluster.



2. Create new user as needed. Create a backup user say bk.

Ingest the Nodes

1. Click Nodes on the left pane and ingest the nodes.

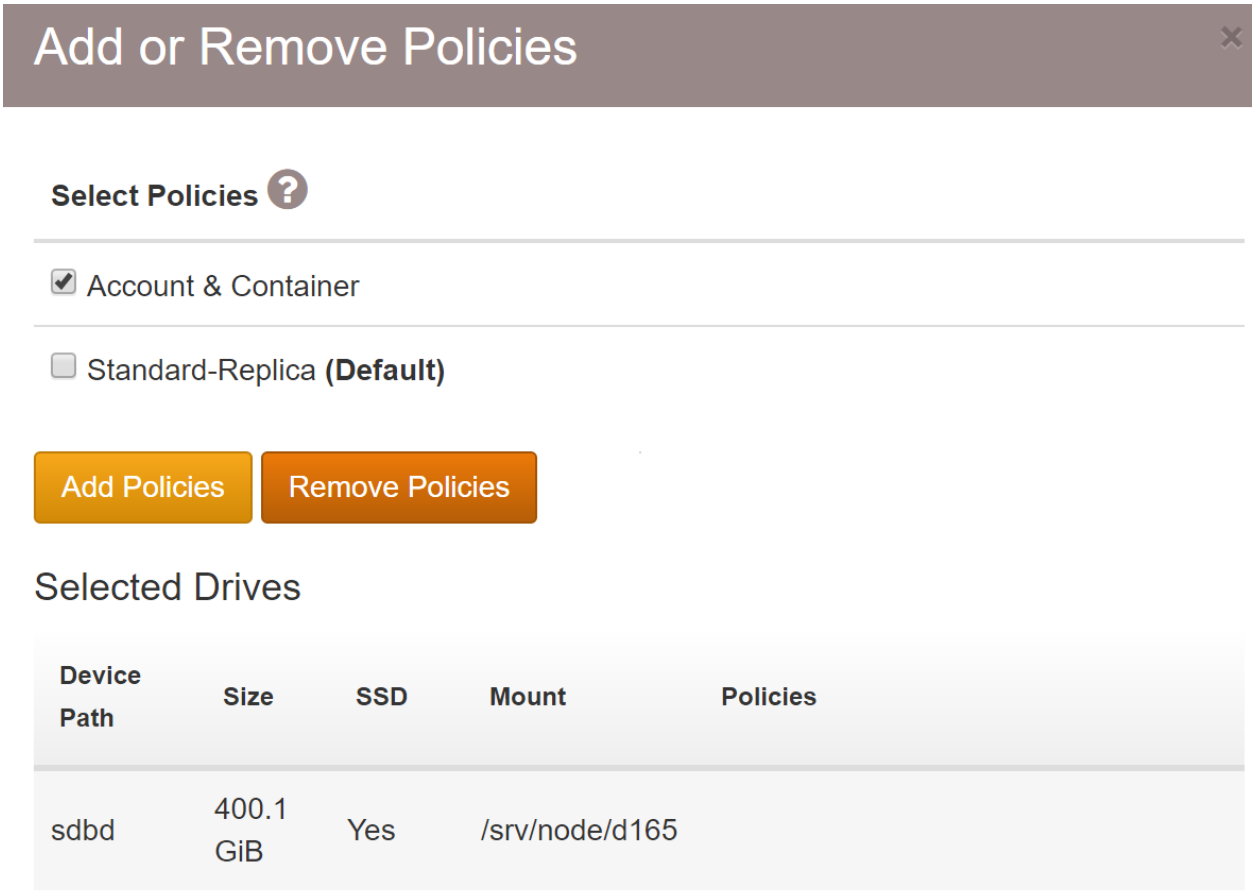
2. Confirm the networks when prompted by the system for outward facing, cluster and replication networks.

Setup the Nodes

1. Click Setup of each node.

    a. Select all the disks and format the drives.

    b. Select SSD disks (2 disks in single node and 1 disk in dual node configurations), click Add Policies and select Account and Container as shown below.



c. Select HDD disks (54 disks in Single Node and 27 disks in dual node configurations), Click on Add Policies and select Standard-Replica.

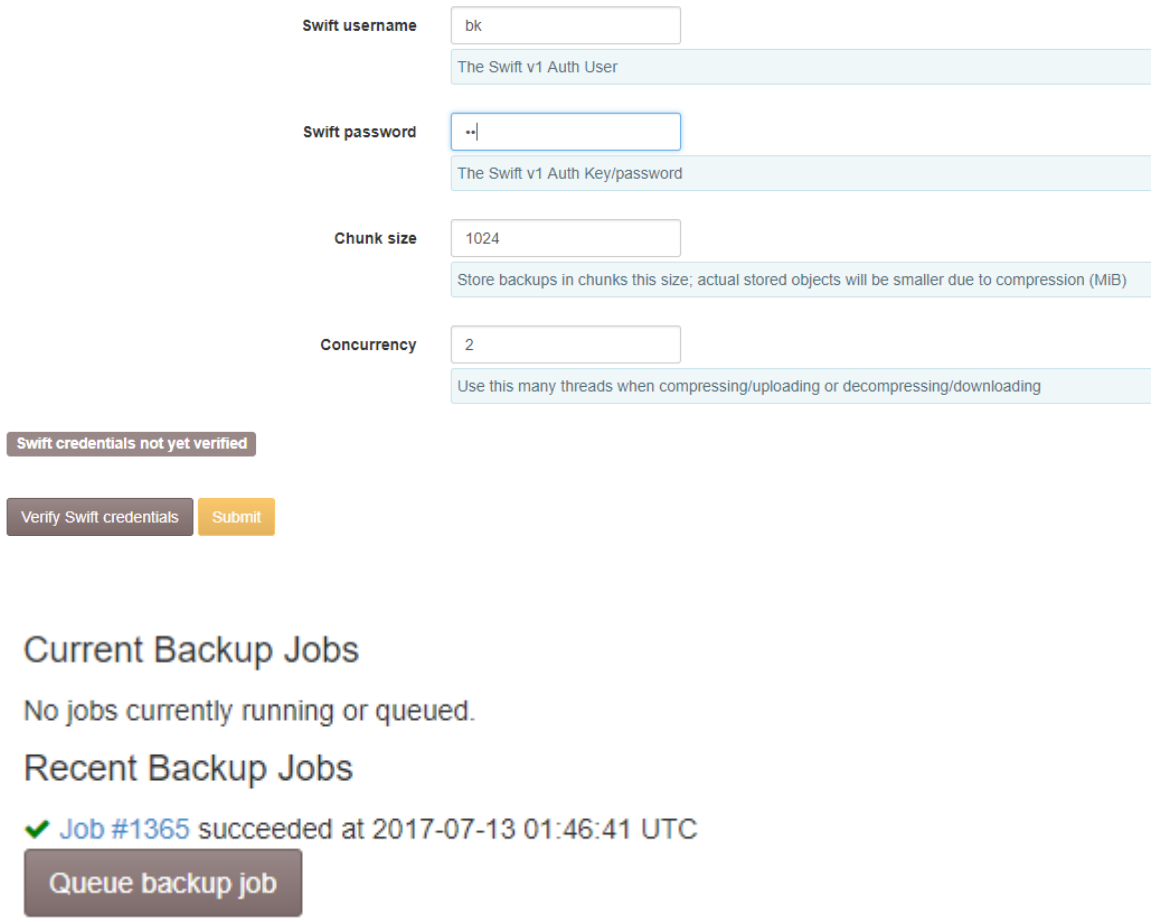
d. Once Completed validate that the policies are displayed correctly on the respective disks as below.

e. In the left menu, enable the node now.

f. This completes the setup of one node. Repeat the procedure for all the storage nodes.

Deploy the Configuration

1. Go to Home > Clusters > Deploy and click Deploy Config to Swift Nodes.
2. Click the Admin Tab on the right top corner and update the default values as needed in your setup.
3. Click backups in Admin page.
4. Enable Save backups to Swift, enter the backup user and password created earlier and click Validate Swift Credentials and submit the changes and then queue a backup job.



Configure Standby Controller for Nodes

The Standby Controller is always in passive node. The primary and standby are not in Active/Active, but in Active/Passive mode. Hence the standby controller is configured and left as is. Refer to the HA section in this document on how to activate the Standby Controller in case of failure of Primary Controller.

Installation

The software installation on standby is the same as on the primary.

Run `./SwiftStack-Controller-5.2.0.2-installer.sh`
The installation should complete with the message as

```
You must now complete setup by pointing a web browser at this server
using HTTPS and the default port of 443.  E.g.

    https://172.22.166.217/

Log in with the username "localadmin" and the default password, "password"
```

1. After entering the URL in the browser, make sure to check ‘This is a Standby Controller’ as below. Enter the name same a the primary controller hostname.

Local controller hostname

If the local hostname must differ from the primary controller hostname in the license (license hostname is a CNAME record or this is a standby controller), enter it here. If left blank, server hostname will be set to the value in the license. Current hostname: swiftcontroller

NEW password for "localadmin" user*

Confirm password*

Enter the same password as above, for verification.

Use insecure fake entropy

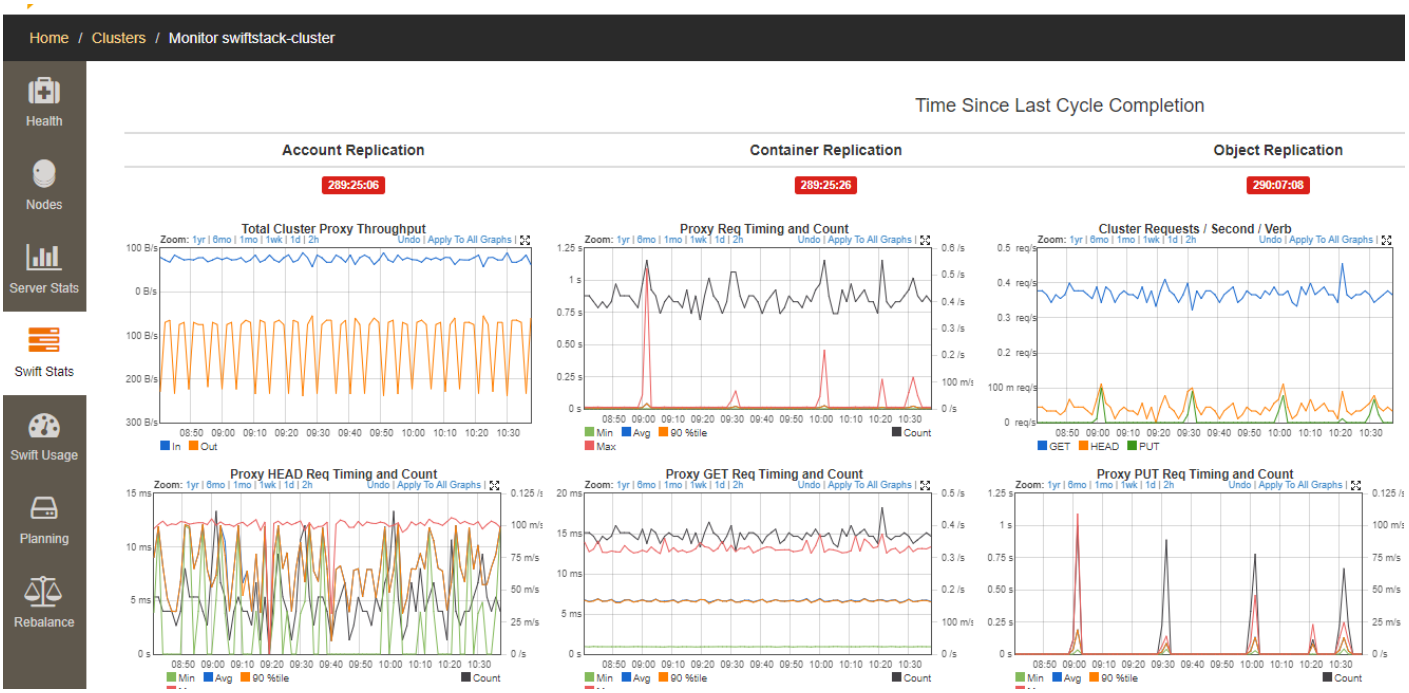
Initial controller setup must create several cryptographic keys. If you want truly secure keys, you must NOT enable this setting and provide true entropy during the initial controller setup process after you submit this form. This usually involves generating interrupts with activity on a physically-attached console keyboard or network traffic. If truly secure keys are not required, you can just enable this setting and keys will be generated quickly.

This is a standby controller

SwiftStack Health Checks

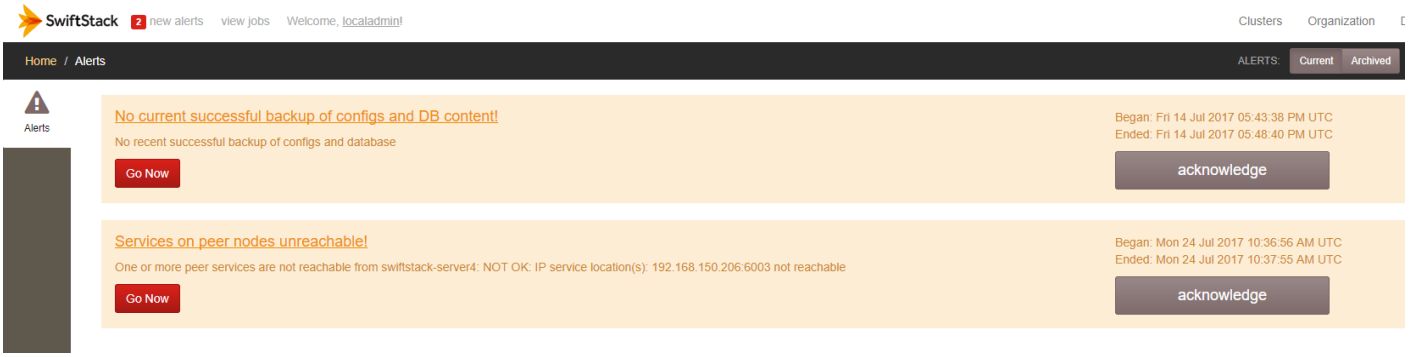
SwiftStack Controller Monitoring and Metrics

SwiftStack Controller metrics are collected in 30 second intervals, with time series data base graphs available for up to 3 years' worth of review. Many basic graphs like CPU, IO and network are available showing overall system health results both at Cluster and Node levels. Swift specific graphs also available showing trends in request handling, error counts that are useful for viewing large changes and troubleshooting.

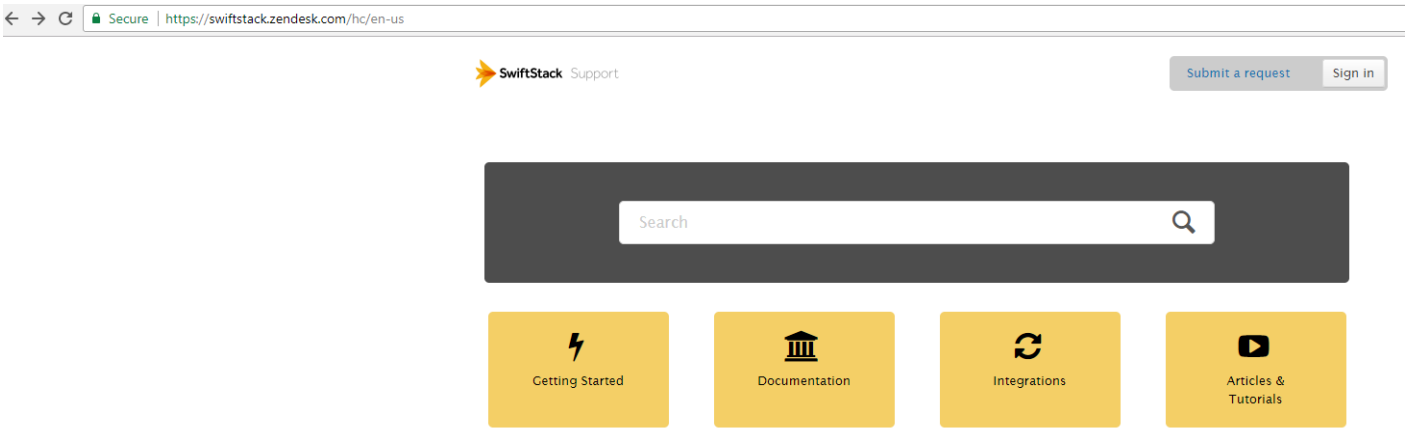


SwiftStack Controller Alerts

Swift Node's alerts can be managed via the controller and alerts can be acknowledged and then archived as desired.



When an alert is received, you can review the alert guide in the SwiftStack Controller documentation https://www.swiftstack.com/docs/. Click on Alerts/Events section to get more details on the alerts.


Secure | https://swiftstack.zendesk.com/hc/en-us
SwiftStack Support
Submit a request
Sign in
Search
Getting Started
Documentation
Integrations
Articles & Tutorials

Load Testing and Performance Evaluation

Load Testing was done on the cluster to evaluate the performance under different configurations. SSBENCH tool was used to test the cluster. The following points were considered while doing the performance testing.

1. All the tests were conducted on default configurations both from Cisco UCS and SwiftStack side to make it as much generic as possible. It is possible that we get better results when Tuning is attempted on both.

2. The purpose of the tests is to get an idea of the performance of the cluster and should nowhere be considered as benchmark values. Most of the effort was spent to get the values of how the cluster will scale up when more chassis/disks are added and also while adding more cpu/mem/network through a separate server node, nor any attempt was made to tune the configuration to its optimal values.

3. All of the tests listed below were conducted in Replicated mode.

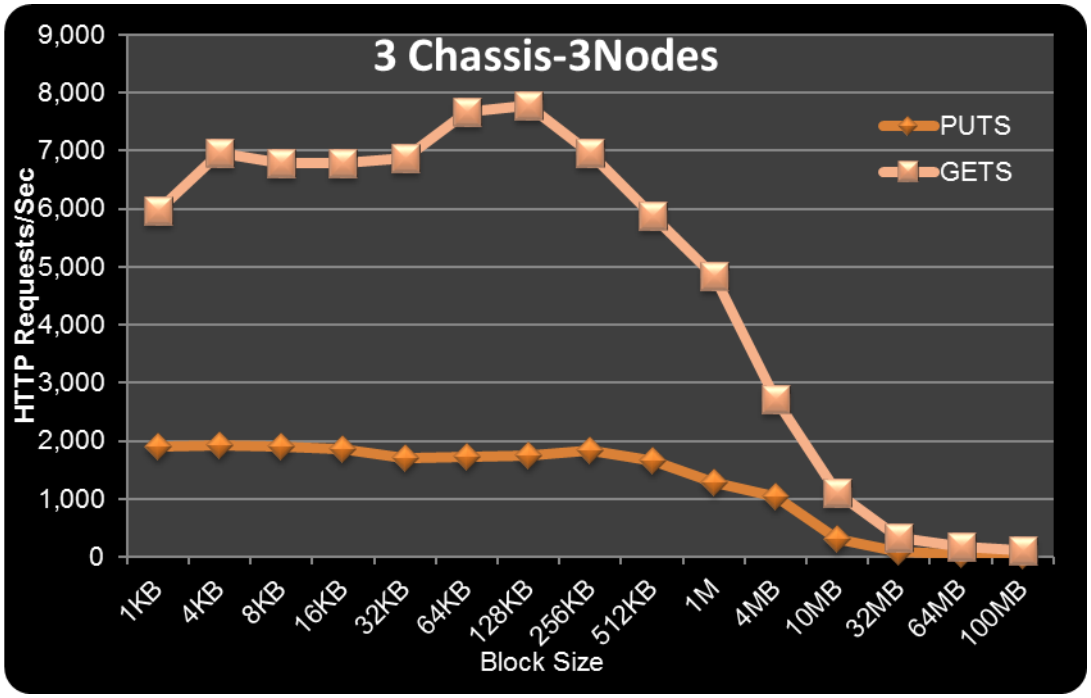
4. A sufficient number of Clients were added to saturate the cluster. Each server had 40 Gb of Network configured for client traffic. Hence in a 12 Node configuration, with a max capacity of 480 Gb of Client network configured on servers, equal number of clients were added to the infrastructure that can push the traffic to servers close to 480 Gb. This is in particular useful when your workload is bandwidth intensive like on block size like 1MB and more.

5. Performance data was gathered on the following configurations:

	Disks (10 TB HDD for objects and 400GB SSD's for A/C and containers)	Network
3 Chassis 3 Nodes	162 HDD + 6 SSD	120Gb client + 120Gb cluster/Repl
3 Chassis 6 Nodes	162 HDD + 6 SSD	240Gb client + 240Gb cluster/repl
6 Chassis 6 Nodes	324 HDD + 12 SSD	240Gb client + 240Gb cluster/repl
6 chassis 12 Node	324 HDD + 12 SSD	480Gb client + 480Gb cluster/repl

3 Chassis - 3Nodes

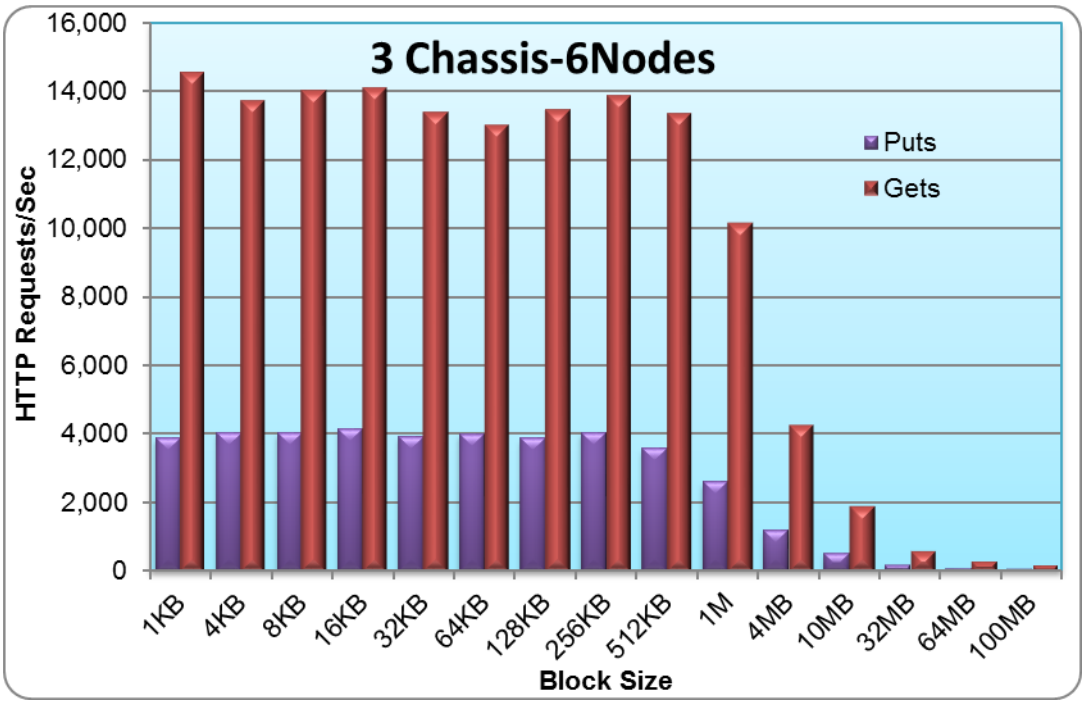
The values obtained for PUTS and GETS are plotted below.



The following points can be concluded:

1. Around 2000 req/sec of PUT requests at 1k and 4k block sizes.
2. More than 7000 req/sec of GET requests observed as peak. This is slightly more than 3 times the PUTS requests. It has to be noted that PUTS aka writes will have an overhead of 3x because of replication.
3. Peak bandwidth of 3000 MBPS for PUTS and 6000 MBPS for GETS.

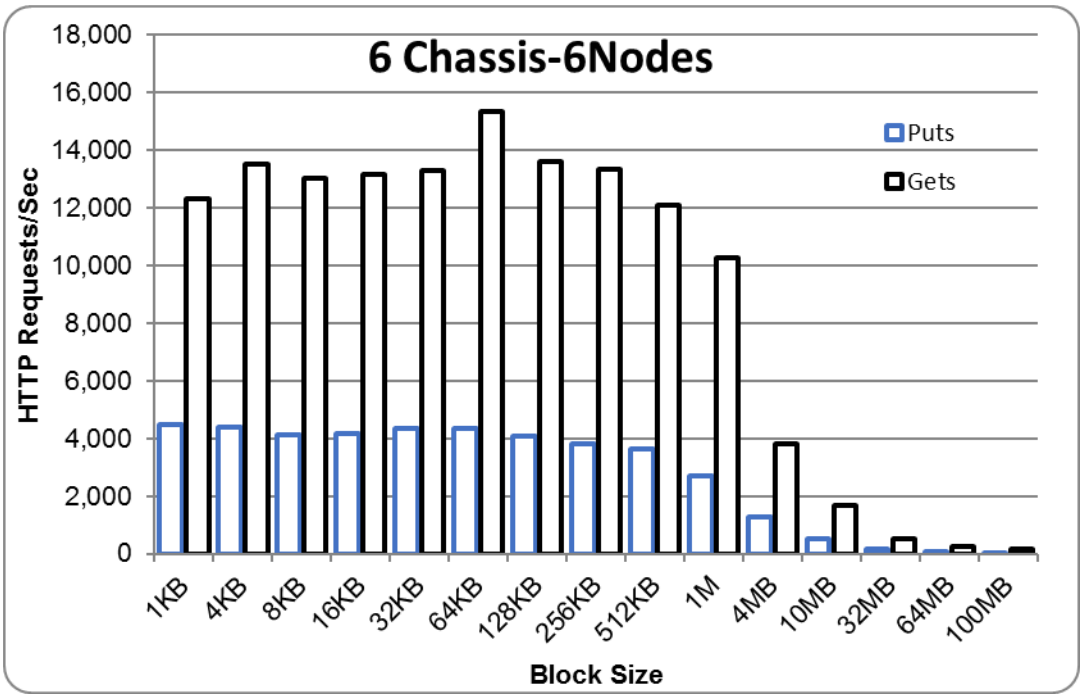
3 Chassis - 6Nodes

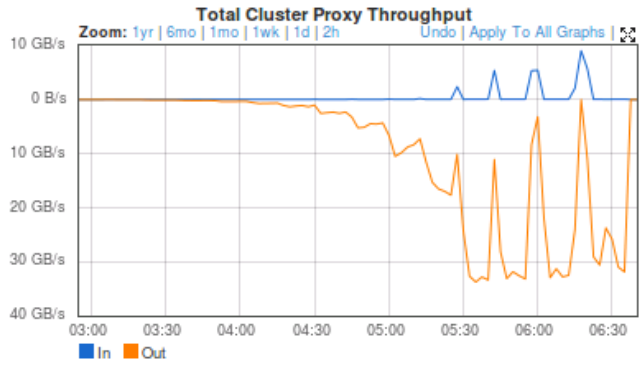


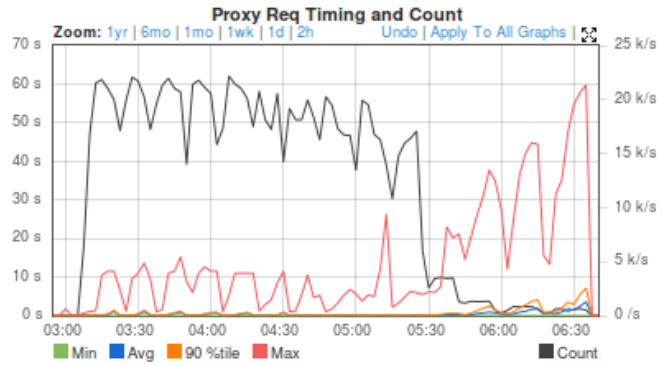
The following points can be concluded:

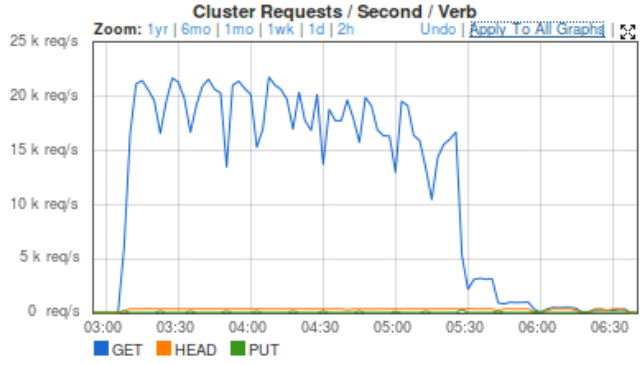
1. Around 4000 req/sec of PUT requests at 1k and 4k block sizes.
2. Around 15000 req/sec of GET requests observed as peak.
3. Bandwidth from client side observed as 5200 MBPS for PUTS while GETS got around 13800 MBPS.

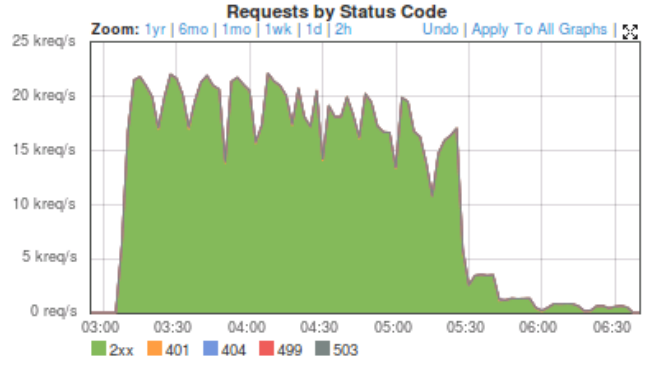
6 Chassis - 6Nodes











The following points can be concluded:

1. Around 4200 PUTS and 14,500 of GETS.

2. Bandwidth from client side observed to be around 5300 MBPS for PUTS and 19000 MBPS for GETS.

3. The graphs collected from the controller show a higher peak values than the average values obtained from ssbench. This shows that for a given period of test from start to end it has averaged.

6 Chassis - 12Nodes

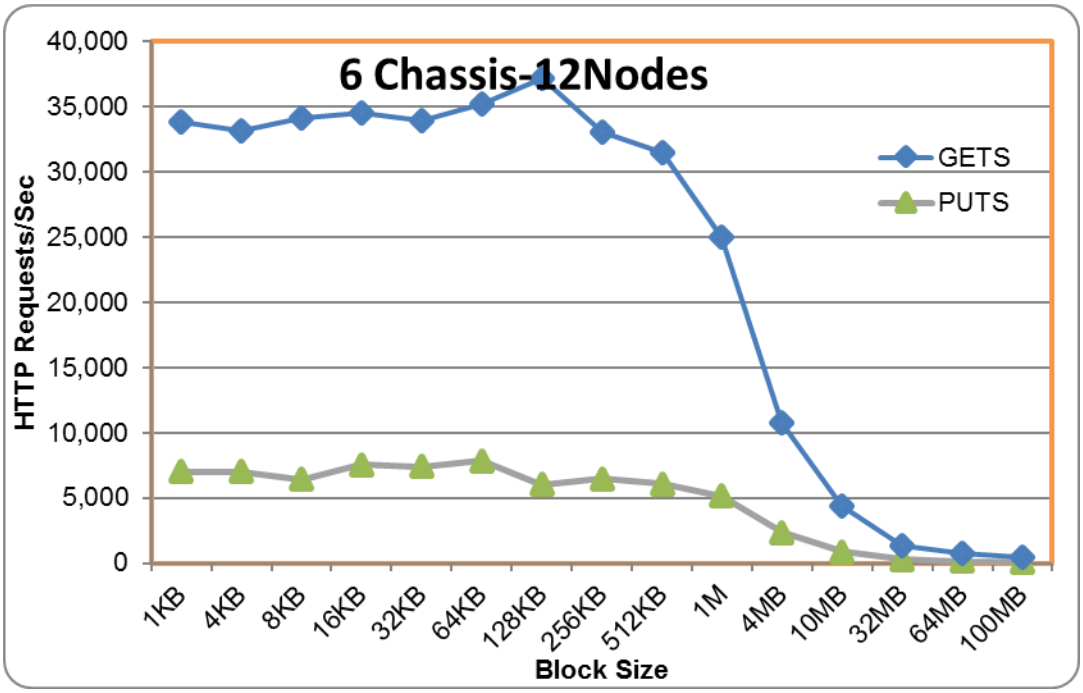


Figure 12 Disk IO for GETS @10MB Block Size

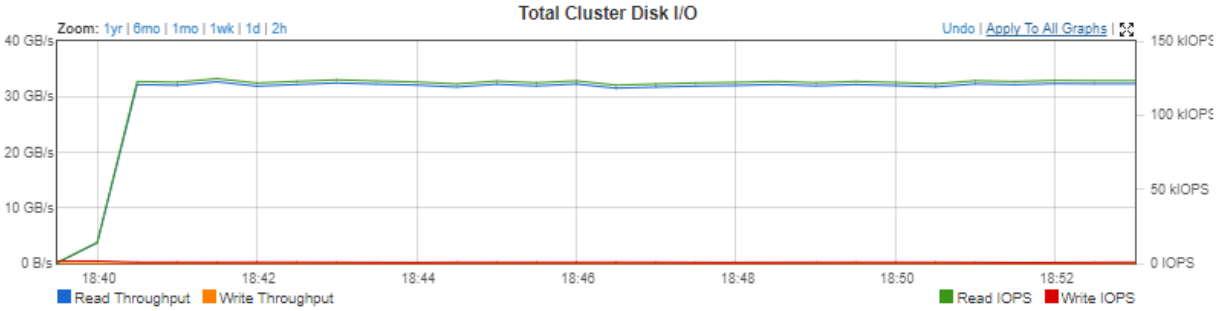
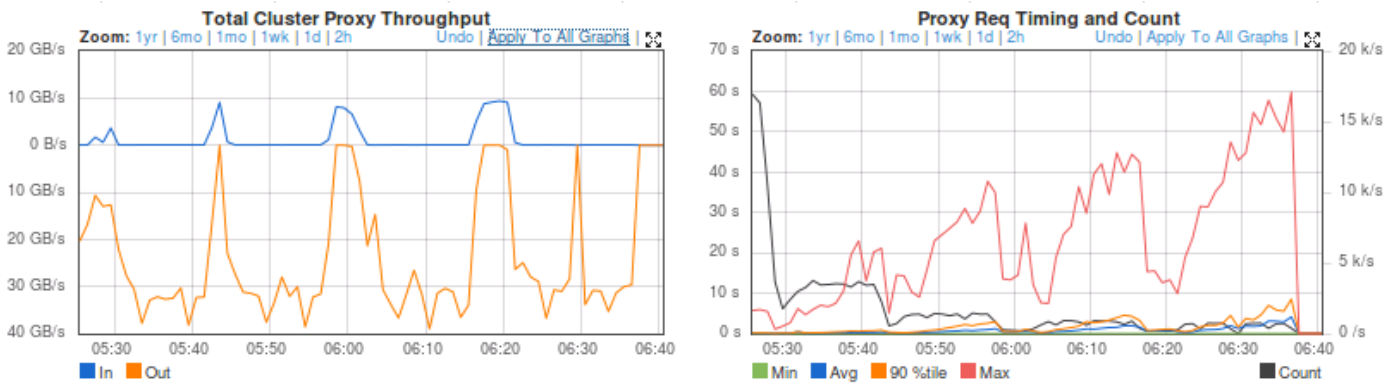


Figure 13 Snapshot from the Controller while Running the Tests



The following points can be concluded:

1. Around 7000 PUTS and 27000 GETS.
2. Bandwidth from client side observed to be around 9400 MBPS for PUTS and 35000 MBPS for GETS.
3. The graphs collected from the controller show a higher peak values than the average values obtained from ssbench. This shows that for a given period of test from start to end it has averaged.

Sizing Guidelines and Scalability of the Cluster

Sizing Guidelines

Choosing the right hardware and number of nodes needed for your application is key. While both Cisco S3260 and SwiftStack give you the option of scaling out the storage as needed, it is best if the requirements are understood correctly and then sized.

The sizing of the cluster depends on several factors, a few are listed below:

The total usable space needed in the cluster.

The type of replication you would like to have for your application – Replicated mode vs Erasure Coded mode.

How many regions you are planning for your cluster.

Network Availability between the cluster nodes and your client nodes to the Cluster.

Number of HTTP requests to or from the cluster to the clients.

For a complete reference of the design and hardware guidelines, it is recommended to go through the SwiftStack documentation: https://www.swiftstack.com/docs/. A few points derived from the test bed based on the performance criteria when using S3260 storage server with 40Gb network is mentioned in this documentation.

Usable Space

The setup used replication. This means every write to the cluster is followed by 2 other writes to two other nodes in the cluster. The effective usable space in the cluster, hence is 1/3rd of the raw space.

Considering 54 x 10TB of disks in a full chassis, the usable space will be around 1PB, in replicated mode for a 6 Chassis cluster. Size your cluster per your usable space requirements.

It is also recommended to leave some free space in the cluster should there be a failure in any one of the nodes. In case of complete failure of a node, SwiftStack will distribute the used blocks to the remaining nodes. In other words in a 6 node cluster, the blocks will be distributed to 5 nodes now. There should be sufficient space left in the cluster for such healthy operations and for business continuity.

HTTP Requests

As mentioned earlier the number of GET and PUT requests and the block size at which this IO happens also matters. In a replicated configuration, PUTS are close to 1/3rd of GETS provided they are no other disk and/or network bottlenecks in the system.

Performance data gathered on four different configurations was gathered to understand how S3260 and SwiftStack would behave when more number of chassis' and nodes were added to the cluster.

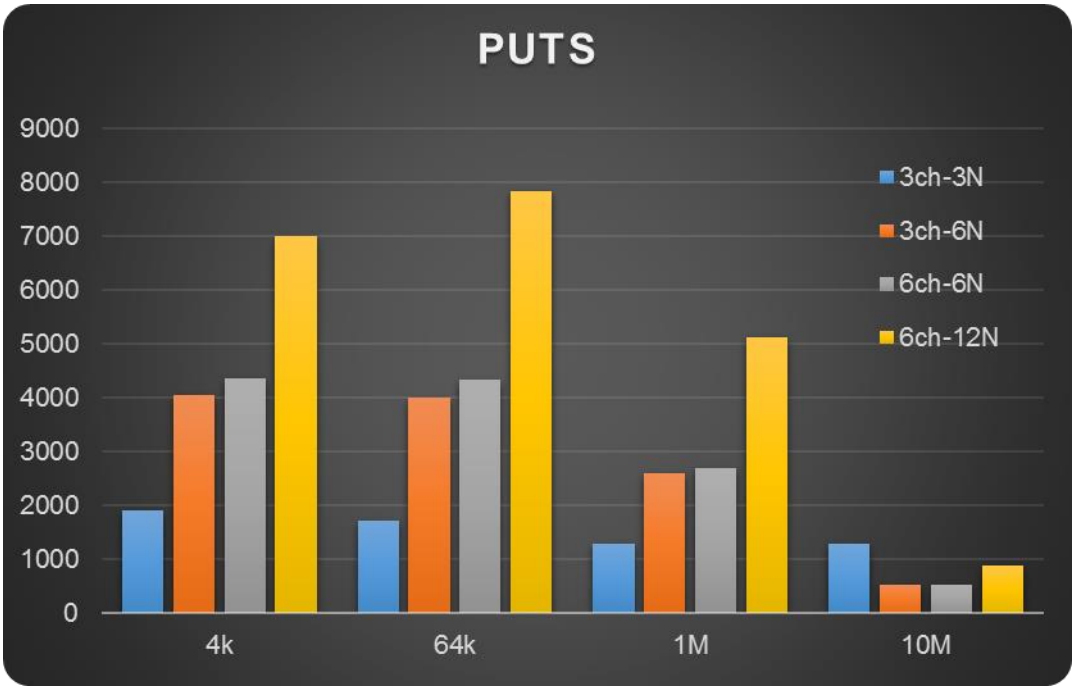
Case Study

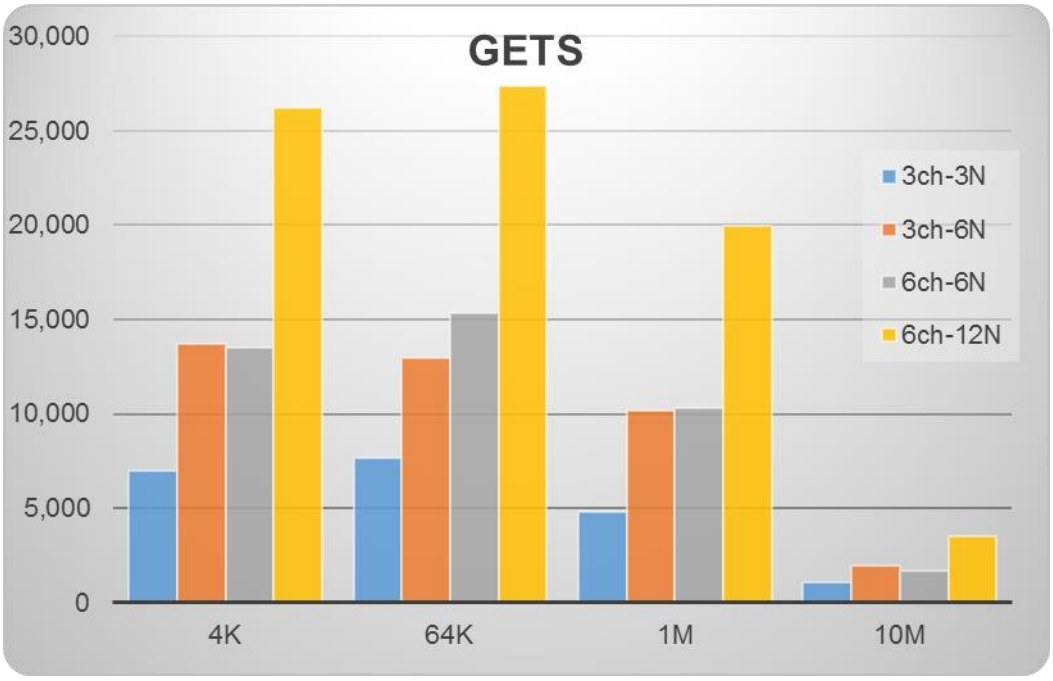
The performance tests were conducted on 3 chassis and 6 chassis with both single and dual node configurations. Below is a case study of 4 block sizes and the GET and PUT requests obtained from the test bed.

	4k		64k		1M		10M		Usable Space in TB***
	PUTS	GETS	PUTS	GETS	PUTS	GETS	PUTS	GETS	Usable Space
3 chassis – 3 Nodes	1918	6962	1729	7677	1294	4844	307	1100	486 TB
3 chassis – 6 Nodes	4046	13714	4012	12983	2599	10153	535	1373	486 TB
6 chassis – 6 Nodes	4369	13508	4346	15322	2698	10284	535	1688	972 TB
6 chassis – 12 Nodes	7008	26176	7838	27359	5119	19937	879	3510	972 TB

*** Approximate Calculations on usable space based on base 2 for storage from disk manufacturers and assuming that only 1/3 of space from raw available because of replication.

Plotting a graph for PUTS and GETS to better analyze the data.





Summary

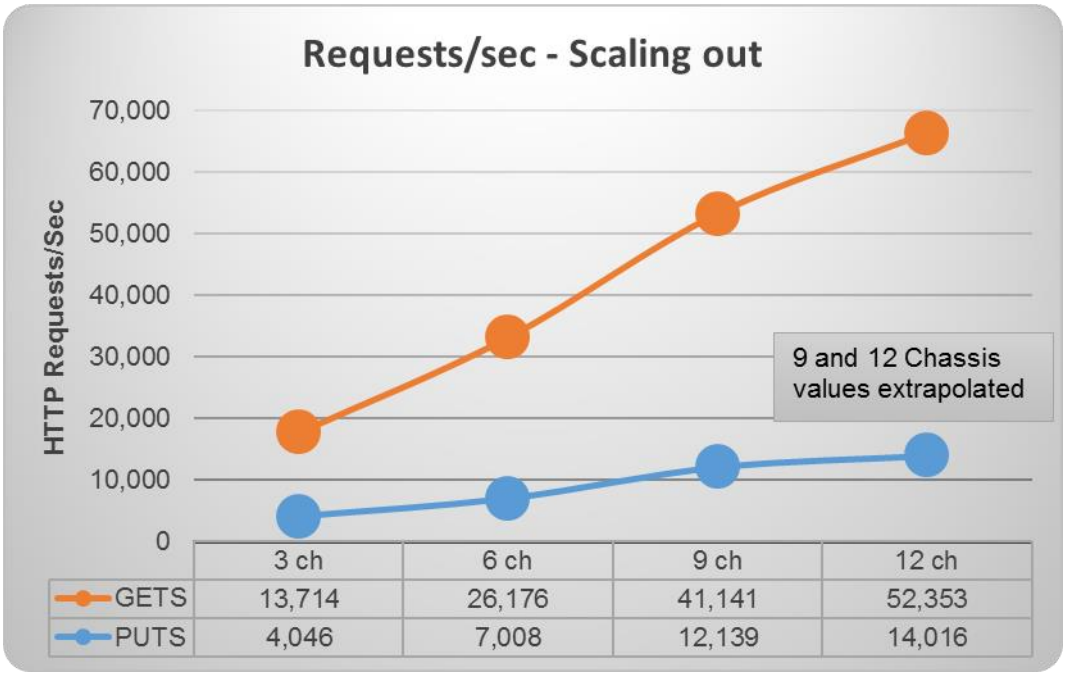
The following conclusions can be made:

1. Decide the number of chassis' required based on the raw and usable storage needed.
    a. Usable space is 1/3rd in a fully replicated mode.
    b. Leave one storage node as spare to take care of failures (N +1 architecture).
    c. Leave 20% operating margin.
    d. Add space for future expansion as well as organic growth.
2. The performance of the cluster increases with number of nodes. More the nodes (the CPU, memory and network) better the performance is.
3. From 3 chassis to 6 chassis, the output is almost doubled (single node to single node and dual node to dual node comparison)
4. Higher blocks sizes seem to get a better benefit by adding additional nodes as the network bandwidth is doubled.

The sizing of the cluster depends on the usable space, number of nodes and performance expectations based on the applications requirements.

Scalability

Considering the data of scalability for a given block size, this is how system could scale by adding more chassis and nodes.



Both PUTS and GETS expected to scale linearly by adding more nodes to the system.

High Availability and Business Continuity

The High Availability of the solution was validated by failing out one of the components of the infrastructure. Following points were considered for Business Continuity.

The Cluster will have reasonable amount of load when the fault is injected. The outputs like Total Cluster Disk I/O or Gets/Puts requested to be gathered before and after Fault injection.

Only one fault injected at any point of time. No double failures considered.

Performance degradation is acceptable but there shouldn't be any business interruption. The underlying infrastructure components should continue to operate with the remaining components.

A few of the HA tests conducted are:

Swift Stack Chassis and Node Failures

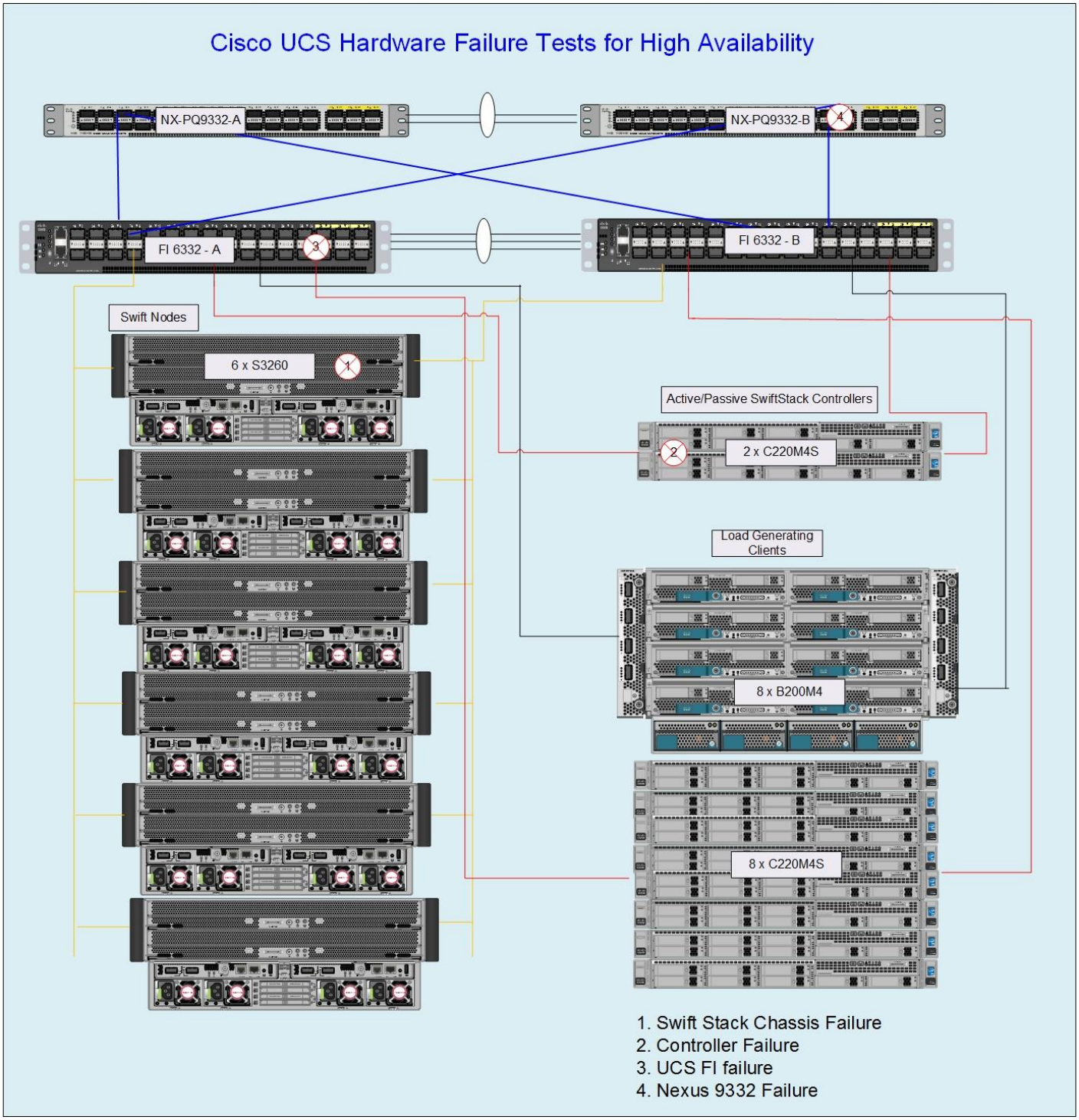
Swift Stack Controller Failures

Cisco UCS Fabric Interconnect Failure

Cisco Nexus Switch Failure

Figure 14 shows the High Availability Tests conducted on the infrastructure

Figure 14 High Availability Tests



SwiftStack Node Failures

Client workload started on the Cluster with SwiftStack Load Balancer.

Sequence of Events

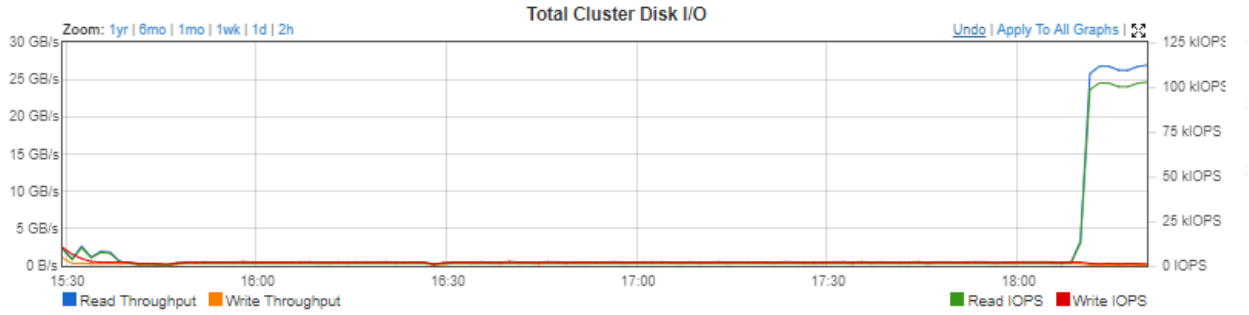
While cluster was running on load, data was gathered on IO, and one chassis was powered off (Nodes pulled out from the chassis), simulating a hardwarefailure. After few minutes of running the cluster in this state, a hard power down of nodes was also issued from Cisco UCS on another chassis. The nodes were powered on and both the chassis' were brought up again after few minutes.

Dstat output gathered on one of the server nodes:

```
[root@swiftstack-server5 ~]# dstat -N eth2,eth3,eth4,total
You did not select any stats, using -cdngy by default.
----total-cpu-usage---- -dsk/total- --net/eth2- --net/eth3- --net/eth4- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send: recv  send: recv  send: recv  send|  in   out | int   csw
  8   3  88   1   0   0|  71M   31M|   0     0 :   0     0 :   0     0 :   0     0 |   0     0 |  33k   37k
 47  17  21   8   0   7|2037M  268k|3336k 1916M:1699M 1798M:1930k 1907k:1704M 3716M|   0     0 |  95k   62k
 43  16  27   7   0   6|1865M  384k|3405k 1966M:1732M 1678M:1972k 1742k:1737M 3645M|   0     0 |  94k   62k
 45  19  21   9   0   7|2142M  156k|3370k 2005M:1681M 1881M: 872k 1469k:1685M 3888M|   0     0 |  93k   59k
 46  19  22   7   0   7|2092M  240k|3401k 2013M:1678M 1774M:1948k 1462k:1684M 3788M|   0     0 |  95k   62k
 43  17  25   7   0   7|1961M 1772k|3649k 2135M:1864M 1710M:1485k 1007k:1669M 3846M|   0     0 |  92k   54k
```

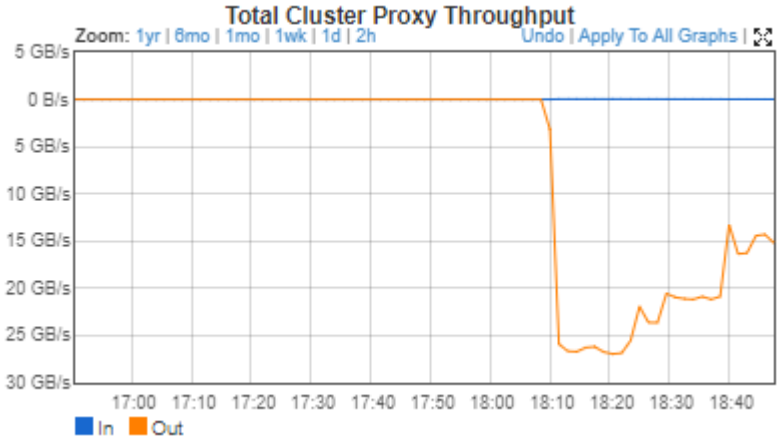
This server was doing disk reads of 2000M, a total Network output of 3800M when the fault was injected.

The total IO of the cluster was around 25 GB/s before fault injection.

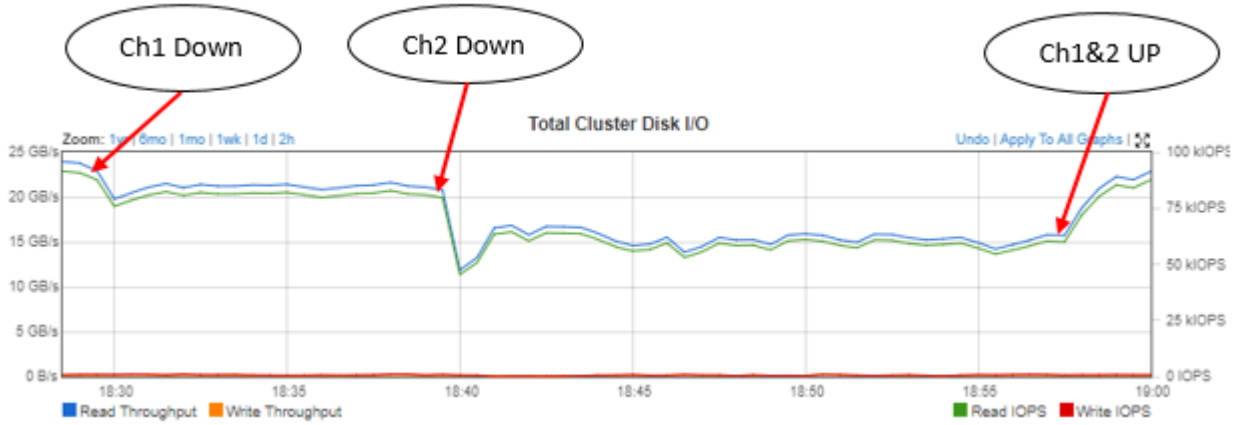


Cisco UCS show Critical Alert that the Chassis cannot be accessed anymore.

Drop in Proxy Throughput observed as below:



The overall activity as recorded by SwiftStack controller is as shown below:



Summary

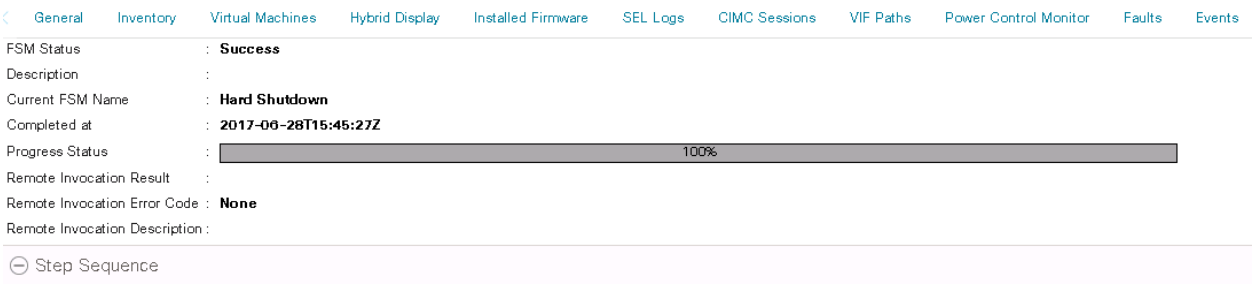
When the first chassis was brought down a drop in IO observed around 18:30. System continues to operate with 20GB/sec.

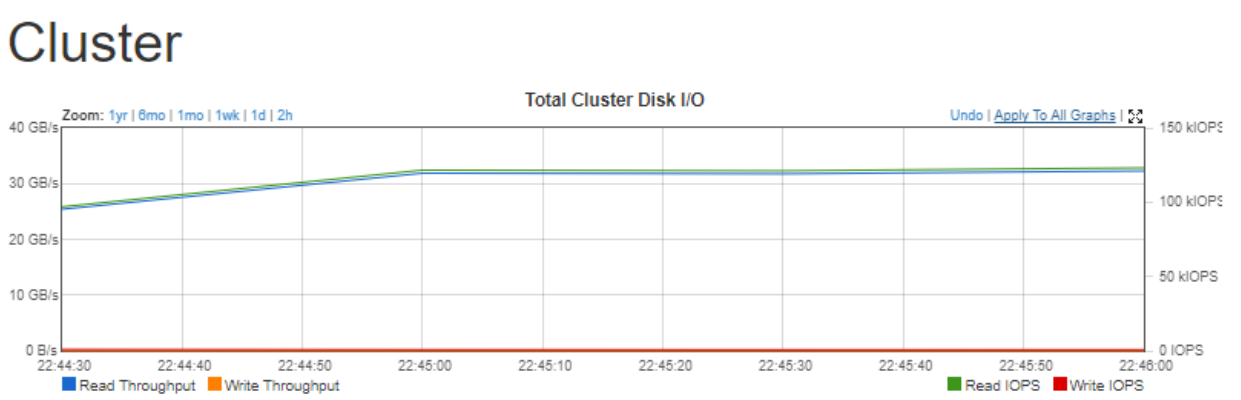
At around 18:40 second chassis was brought down and IO dropped almost to 15GB/sec.

System fully recovers when chassis' were plugged back in.

SwiftStack Controller Failures

SwiftStack controller is not in the data path. No interruption to the client traffic was observed when the Controller node was brought down.





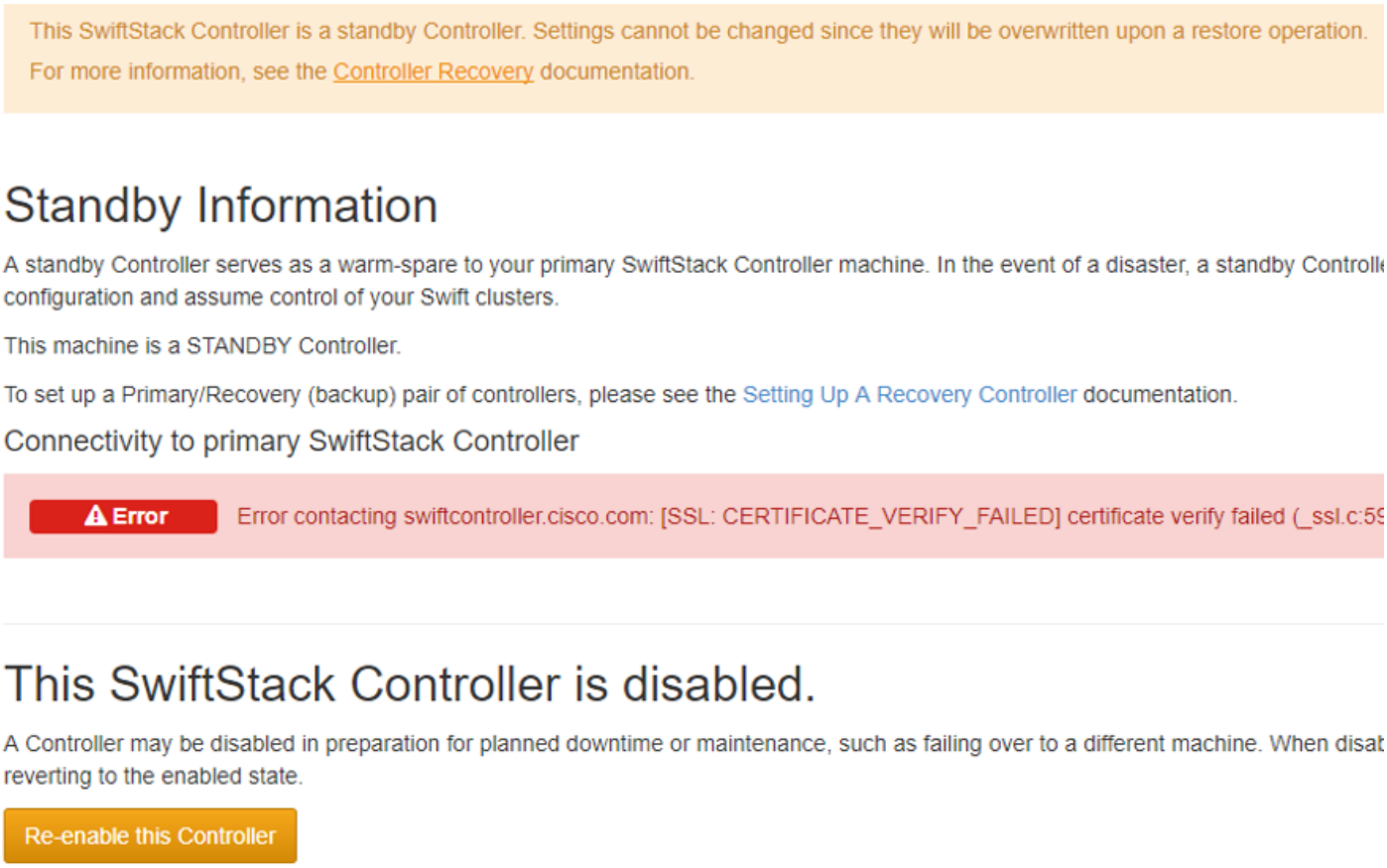
Summary

The clients communicated directly with the PACO nodes and there was no interruption. However, any monitoring, SNMP alerts configured etc. will not work. There is a need to activate the Standby Controller and make it Primary in case it is decided that it's a total hardware failure and cannot be recovered soon. This is covered in the following section.

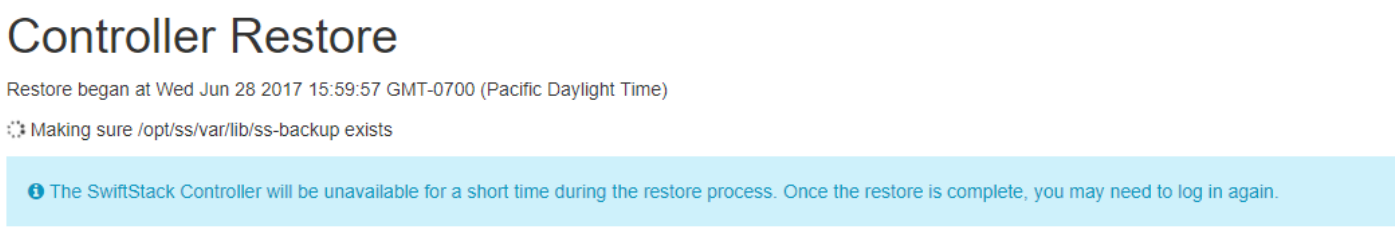
Activate Standby Controller to Primary

1. Update the DNS entry (or /etc/hosts) for the name of the controller to be accessed appropriately.
2. Login to the controller UI, Click on Admin tab and then Recovery or directly login to https://swiftcontroller.cisco.com/recovery/standbys/.

The following information will be displayed:



First restore the controller before enable. As the backup is taken on swift, it will sync up from the database and once completed could be enabled.



The standby controller becomes primary. You may continue to operate in this mode or revert back to the original primary when the problem was fixed.

Cisco UCS Fabric Interconnect Failures

Client workload started on the Cluster.

Sequence of events:

While cluster was running on load data gathered on IO. One of the Fabrics was rebooted and any changes to IO pattern was observed. The reboot was done on Secondary first and then the tests were repeated on Primary.

The tests were conducted for both Gets and Puts operations.

Dstat output gathered on one of the server nodes while running PUTS.

```
You did not select any stats, using -cdngy by default.
----total-cpu-usage---- -dsk/total- --net/eth2- --net/eth3- --net/eth4- -net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send: recv  send: recv  send: recv  send|  in   out | int   csw
 11   5  81   0   0   2| 329M   20M|   0     0 :   0     0 :   0     0 :   0     0 |   0     0 |  60k   25k
 54  30   3   4   0  10|  20M 2148M|  06M   20M:1893M 1775M: 234k  230k:2500M 1796M|   0     0 | 228k   72k
 54  30   2   3   0  11|  10M 2003M|  704M  610k:1955M 1915M: 206k  197k:2659M 1915M|   0     0 | 239k   64k
 57  31   0   1   0  11|  10M 1951M|  767M  7.M:19.2M 2083M: .38k  134k:2729M 2094M|   0     0 | 244k   41k
 57  32   0   1   0  11|  20M 2106M|  655M  585k:1966M 1884M: 138k  132k:2621M 1885M|   0     0 | 235k   38k
 57  32   0   0   0  11|  18M 2126M|  696M  605k:2062M 1818M: 133k  133k:2758M 1819M|   0     0 | 237k   29k
 56  33   0   0   0  11|  12M 2049M|  846M 7956k:1784M 2331M: 164k  151k:2631M 2339M|   0     0 | 241k   31k
```

This server was doing a disk writes of 2148M and handling 750M of Client Traffic while generating around 2000M of Cluster traffic.

Logged into surviving Fabric Interconnects to check the status of the cluster and it was down.

UCS-40G-Openstack-dom2-Fab-B# show cluster extended-state
Cluster Id: 0x48f6d6ca658111e6-0x924b0078883f3a3b

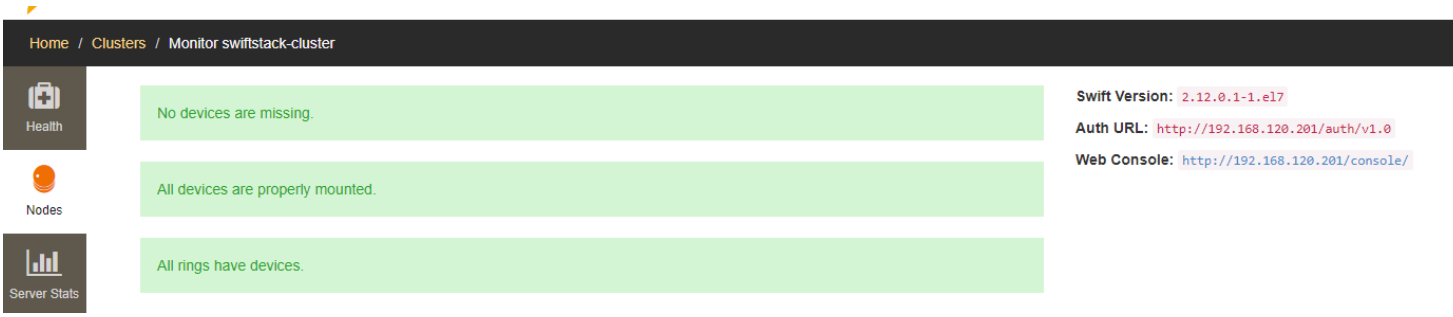
B: UP, SUBORDINATE
A: UNRESPONSIVE, PRIMARY

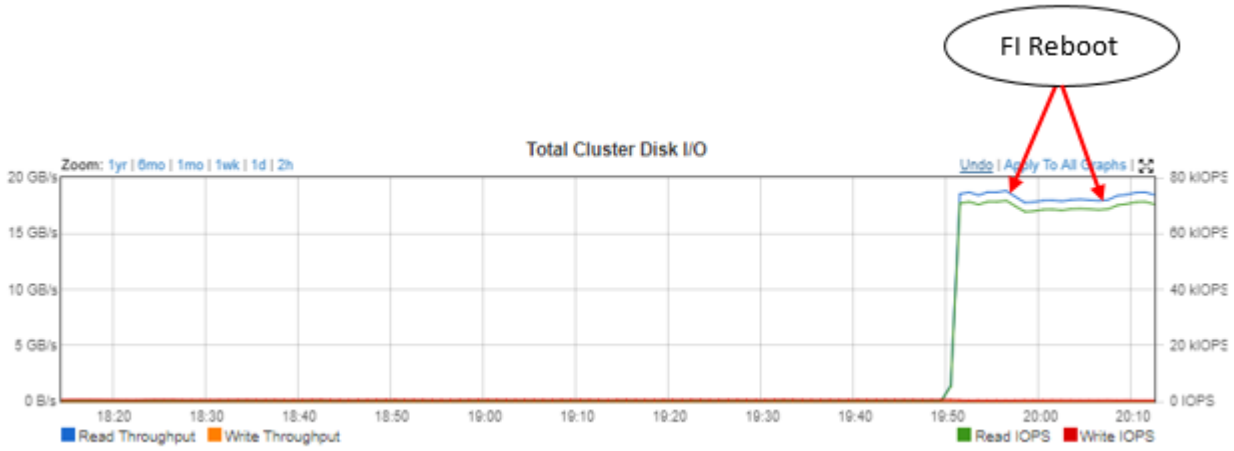
B: memb state UP, lead state SUBORDINATE, mgmt services state: UP
A: memb state UNRESPONSIVE, lead state PRIMARY, mgmt services state: INVALID
heartbeat state SECONDARY_REQUESTED

INTERNAL NETWORK INTERFACES:
eth1, DOWN
eth2, DOWN

HA NOT READY
Peer Fabric Interconnect is down
Waiting for device to come online.
Device count, expected: 3, discovered: 2
Detailed state of the device selected for HA storage:
Chassis 1, serial: FOX1948G99W, state: inactive
Chassis 2, serial: FOX1540GN14, state: active
Server 1, serial: FCH1904V31U, state: active
UCS-40G-Openstack-dom2-Fab-B#

Swift Controller does not point out any errors or devices/nodes when FI is getting rebooted.





Summary:

No interruption of traffic was observed. A small drop in output when the entire traffic is handled by single FI.

The drop in performance also depends on distribution of links and head room available on one Single FI to handle all of the load.

It was around 20GB/sec for puts and 35 GB/s for gets. In both the cases FI handled the traffic, vNIC's failed over to surviving FI, vNIC's failed back and MAC learning completed fine after FI came back and the business was as usual.

Cisco Nexus 9332 Failures

Client workload started on the Cluster.

Sequence of events:

While cluster was running, load data gathered on IO. One of the Nexus Switches was rebooted and any changes to IO pattern was observed.

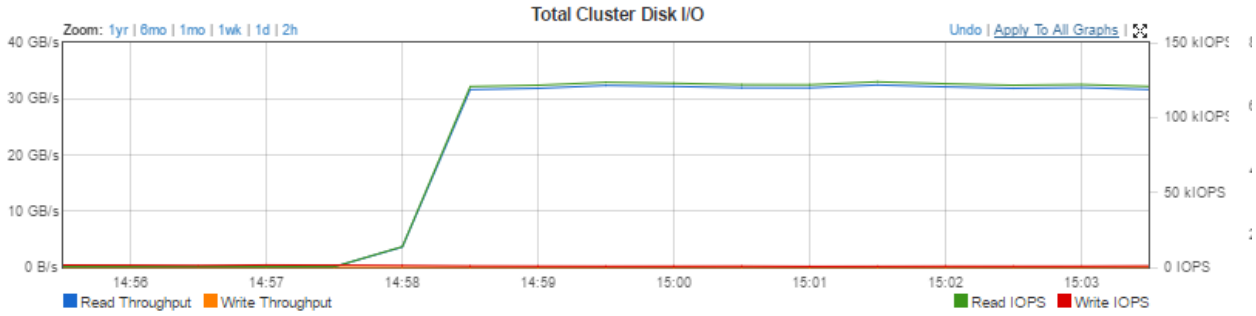
Cisco Nexus Switch Details:

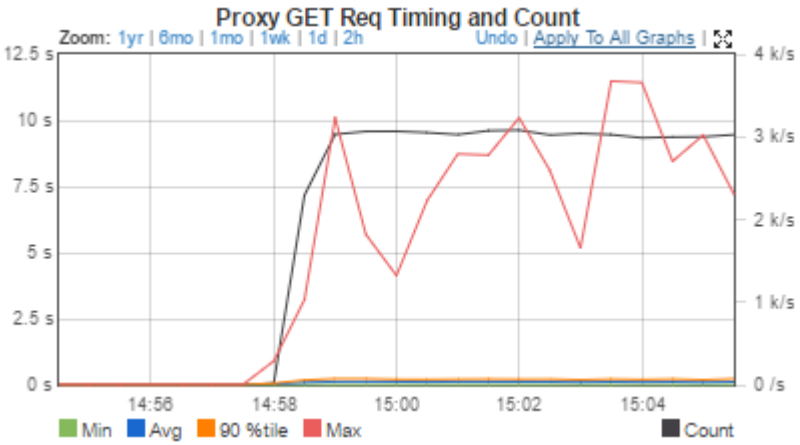
```
  BIOS: version 07.41
  NXOS: version 7.0(3)I1(3)
  BIOS compile time:  10/12/2015
  NXOS image file is: bootflash:///n9000-dk9.7.0.3.I1.3.bin
  NXOS compile time:  8/21/2015 3:00:00 [08/21/2015 10:27:18]


Hardware
  cisco Nexus9000 C9332PQ chassis
  Intel(R) Core(TM) i3- CPU @  with 16402540 kB of memory.
  Processor Board ID SAL2025S8AC

  Device name: N9K-40G-OPS-FAB-B
  bootflash:   51496280 kB
Kernel uptime is 74 day(s), 16 hour(s), 28 minute(s), 15 second(s)
```

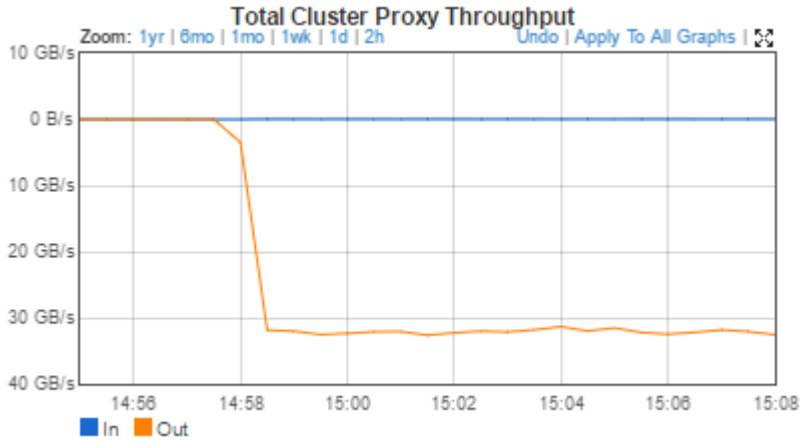
```
Last reset
  Reason: Unknown
  System version: 7.0(3)I1(3)
  Service:

plugin
  Core Plugin, Ethernet Plugin

Active Packages:
N9K-40G-OPS-FAB-B#
```

Make sure that running config is copied as startup configuration and then issue a reboot of the switch.

Before Fault Injection



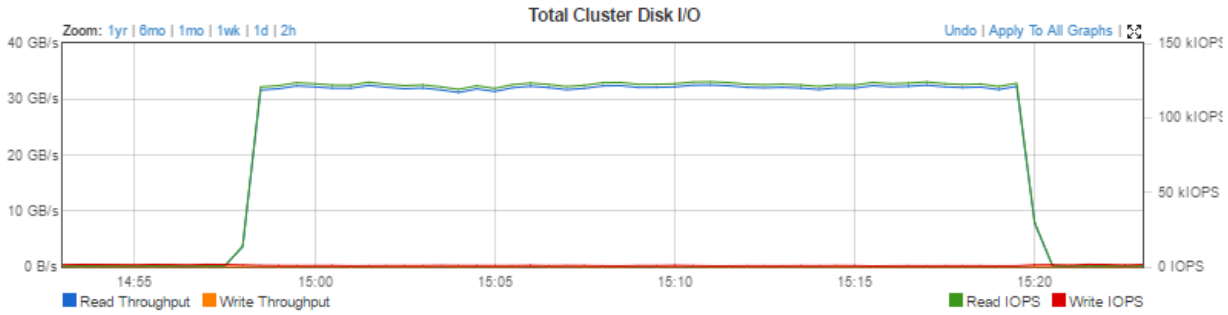




Switch B was reloaded

```
N9K-40G-OPS-FAB-B# show clock
15:06:34.315 UTC Wed Jun 28 2017
N9K-40G-OPS-FAB-B# reload
This command will reboot the system. (y/n)?  [n] y

Rebooted around 15:06 Nexus switch time.

After reboot switch came up fine and no interruption traffic

Hardware
  cisco Nexus9000 C9332PQ chassis
  Intel(R) Core(TM) i3- CPU @  with 16402540 kB of memory.
  Processor Board ID SAL2025S8AC

  Device name: N9K-40G-OPS-FAB-B
  bootflash:   51496280 kB
Kernel uptime is 0 day(s), 0 hour(s), 5 minute(s), 54 second(s)

Last reset at 518842 usecs after  Wed Jun 28 15:06:58 2017

  Reason: Reset Requested by CLI command reload
  System version: 7.0(3)I1(3)
  Service:

plugin
  Core Plugin, Ethernet Plugin

Active Packages:
N9K-40G-OPS-FAB-B#
```

During the entrie time of reload, no interruption of traffic was observed.



Summary:

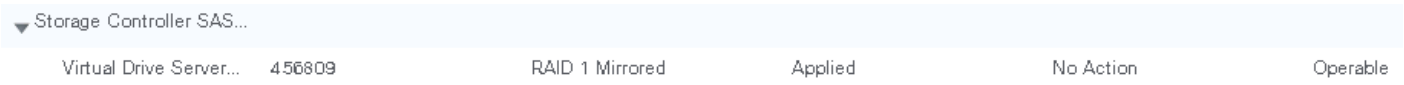
Port Channels were configured on the Nexus Switches as mentioned in the configuration section earlier. Observed Business continuity during through out the tests.

Scaling up the Cluster

This section provides a short description about how to scale up the cluster.

Adding Nodes to Cisco UCS

1. Make sure that the chassis and/or nodes VIC's are connected to the Fabric Interconnects. Power on the chassis. System should do auto-discovery.
2. When the Chassis is discovered, make sure that the inventory (CPU, memory, disks) are fine as expected.
3. Make sure that you have enough room available in UUID, KVM and IP pools. Else create a new pool to add more IP's or UUID's.
4. Create Service Profiles from the Service Profile Templates. Make sure that the disks are in unconfigured state and apply the newly created Service Profiles.
5. After observing that virtual lun is created in the inventory > storage tab after applying the Service Profile, install the operating system.



6. As mentioned in the OS post Install section, update linux kernel with yum and complete any other configurations mentioned in the post-install section.

System is ready to be added as a node to the Cluster.

Adding Nodes to SwiftStack Cluster

1. Copy the ssman.crt file from Controller Node to the new Storage node.

```
scp ssman.crt root@swiftstack-new-node:/etc/pki/ca-trust/source/anchors/
Run update-ca-trust extract on the new storage node.
```

2. Run the Curl Command on the storage node. This will download the necessary packages and installs the software.

   curl https://swiftcontroller.cisco.com:443/install | sudo bash

3. Run the ssclaimurl on a browser and claim the node
4. In case you are using SwiftStack Load Balancer, reconfigure.
5. Ingest the nodes

6. Click on Setup of the nodes – format the disks, add SSD and HDD's to respective policies. SSD's go to Account and Container Policy, while HDD's to Standard-Replica policy.

7. Do not deploy the configuration yet in case you have more nodes to be added now. Complete all the nodes setup upto step 6 above, enable them and then deploy the configuration.

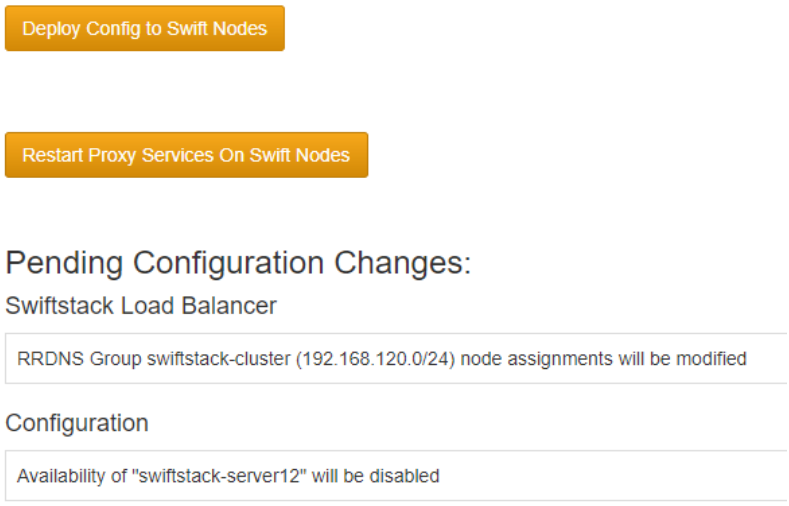
This completes the addition of new node(s) in the cluster. Please note that this could automatically kick in the rebalance activity. The time it takes to distribute used space to the new nodes depends on the used space in the cluster, prior to the addition. While this is a background job, it does consume some network and disk resources. There will be traffic flow through the replication nic configured on the system. This activity can be speeded up or slowed down by tuning the resources in the tuning section. Please contact swiftstack support team that is optimal for your configuration.

Migrating to Dual Node Configuration

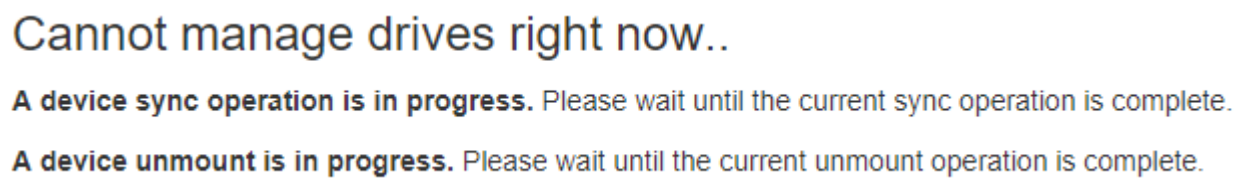
If you have a single node configuration and would like to upgrade to a dual node configuration please follow the steps below.

Delete the chassis with single node

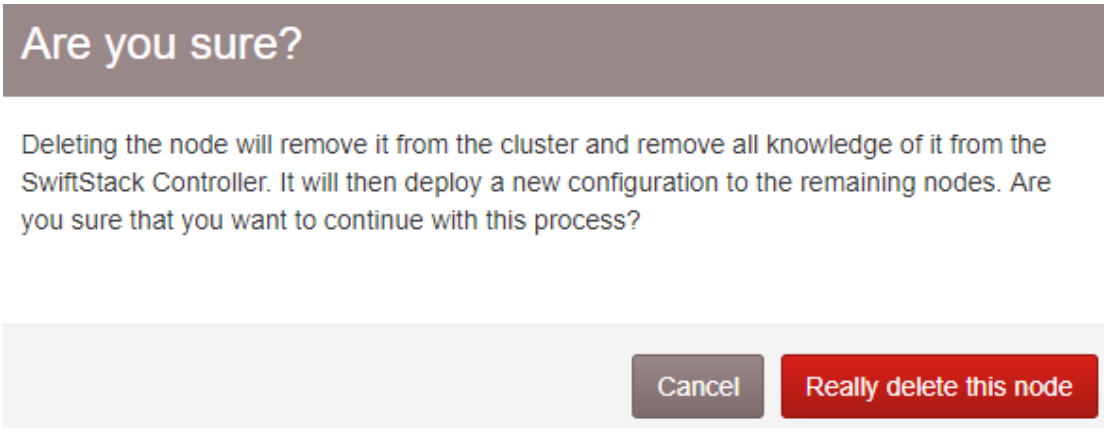
1. Login to controller, Manager SwiftStack Controller and then click Nodes.
2. Click Manage and then disable the node and then deploy the configuration. This will let the Cluster know that one of the nodes has been disabled.



3. Go back to the disabled node in the same manage page (it will be in disabled mode now). Click on setup now, select all drives and then unmount them. You may also monitor the devices by logging into the node and by running df –k.

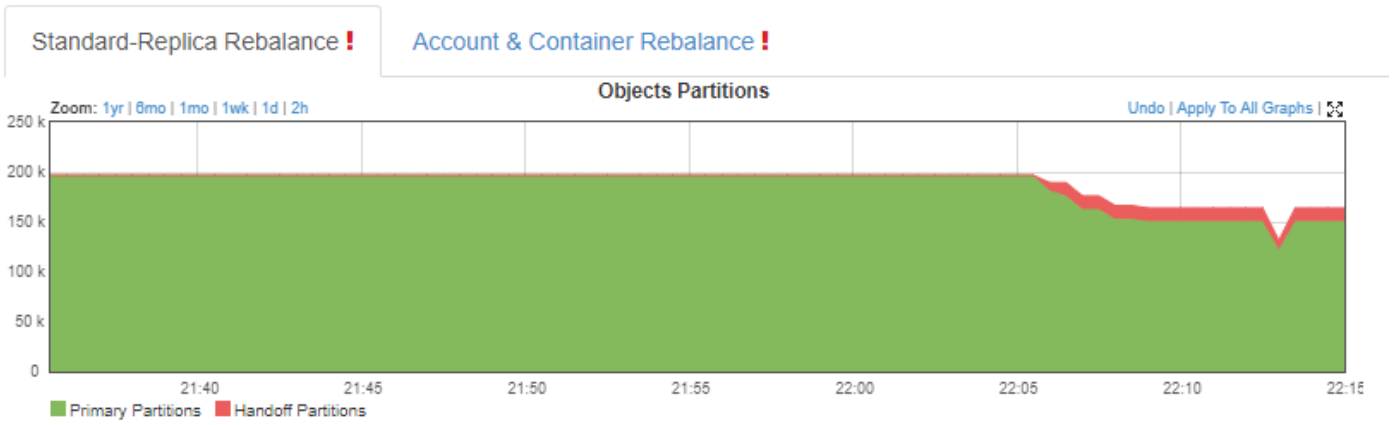


4. In the same page with devices unmounted, select all the drives and click ‘Delete’ now, which will make the disks as unmanaged drives now.
5. Now you can delete this node. This will remove the devices and the node from ring.



6. If not already done, deploy the config to the cluster. This will make the cluster nodes as ‘N-1’ now. This can be verified from clicking on the nodes again.

7. This will also trigger rebalance activity in the cluster for both Account and containers (2 SSD’s deleted) and objects or standard replica.



8. Network activity can be verified by logging into any of the nodes.

```
You did not select any stats, using -cdngy by default.
----total-cpu-usage---- -dsk/total- --net/eth1- --net/eth2- --net/eth3- --net/eth4-  net/total- ---paging-- ---system--
usr sys idl wai hiq siq| read  writ| recv  send: recv  send: recv  send| recv  send: recv  send|  in   out | int   csw
 17   9  72   2   0   1| 111M  224k| 220B  222B: 294B  195B:  36k   64k| 166M  106M: 166M  106M|   0     0 |  55k   52k
 16   9  73   2   0   1| 105M  548k| 348B  396B:  58B   64B: 204B    0 | 141M   98M: 141M   98M|   0     0 |  49k   44k
 16   8  74   2   0   1| 101M 1520k|  68B  222B: 204B    0 :   0     0 | 102M   95M: 102M   95M|   0     0 |  52k   51k
```

This completes the deletion of the node.

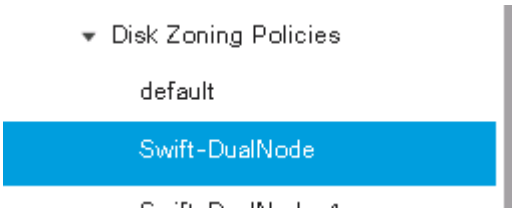
Add the Second Node to Chassis

Add the hardware, the second Server, SIOC and connect the 40G Twin-Ax cables to both the Fabric Interconnects. For details of connectivity, refer to Initial setup and then discover the hardware in Cisco UCS. If needed add necessary KVM IP’s, MAC and UUID’s.

Reconfigure Chassis in Cisco UCS

Associate Chassis Profile

Refer to the section above for creating chassis profile template. Create a new Chassis profile template with ‘dual node’ disk zoning policy as mentioned in this section. Create a chassis profile from this template and then associate to this chassis. The Dual Node disk zoning policy will add the first 28 disks to first node and the rest 28 to the second node. Make sure that each server gets 1 SSD in its distribution else move the SSD disks appropriately within the storage slots.



Associate Service Profile

Create a new Service Profile template if needed and then associate them to the 2 new nodes.

Install Operating System and SwiftStack Software

Follow the Operating system installation steps, post-install checks, and then install the SwiftStack Software on the two nodes as mentioned earlier in this document.

Install SwiftStack software, claim the node, format the disks, add policies and then enable the nodes in the controller.

Final Steps

When the node is added back to the cluster, it may trigger to re-balance the disks again. The rebalance activity on SSD’s and HDD’s depend on the used space within the cluster.

Repeat all of the steps in case you need to make the rest of the chassis also as dual node.

Frequently Asked Questions

Cisco UCS

1. What Cisco UCS Manager version is supported with SwiftStack on Cisco UCS?

   SwiftStack has been validated with UCSM version 3.x and with 40Gb Fabric Interconnects and Switches. It is strongly recommended to run the infrastructure on versions higher than the validation done in this CVD.

2. Can I use 10GB Nexus Switches in the infrastructure?

   This hasn't been validated but might still work. It is strongly recommended again to use 40Gb network between the Storage Nodes. By using 10GB with splitter cables, you might be underutilizing the hardware.

3. How many minimum nodes are recommended?

   This is as per the SDS requirements. The validation was done with minimum of 3 nodes and then scaled up further.

SwiftStack

1. The CVD talks about performance data collected in Replicated mode. Where can I get information on EC (Erasure Coding)?

   Erasure coding and savings in space with a little performance hit are known in SDS world. We will be validating this soon in the infra and will update this CVD.

2. How many nodes can system scale up?

   SwiftStack can scale up to petabytes of usable data. The nodes can be added and/or deleted as mentioned in the scale up section. You may have to check for the availability of the ports on the upstream switches.

Troubleshooting

1. I am unable to claim the node. System continuously spins but never presents me with a ‘Claim Node’ button. What should I do?

   There can be network glitches and it is fine to retry the operation. In case you are not successful, follow these steps:
   From the controller page, delete the node. Login to the server node and run /usr/bin/uninstall-swiftstack-node. This will completely remove the software from the system. Restart by running the curl command to add this node to the controller.

2. What diagnostics can I run in server node to check the status of a storage node?

   Running /usr/bin/ssdiag will spill out any errors. A healthy node may report as below

```
[root@swiftstack-server2 ~]# /usr/bin/ssdiag
SwiftStack Agent Version:        5.2.0.2-1
Swift Package Version:           2.12.0.1-1.el7
……
          Daemons are running: OK
         Resolve API hostname: OK:      (no cluster API hostname defined to check)
           SwiftStack ProxyFS: OK:      (proxyfs not enabled)
       SwiftStack NAS Gateway: OK:      (gateway not enabled)
          VPN link is working: OK
   IP address(es) consistent: OK
SwiftStack Node Connectivity: OK
    SwiftStack agent version: OK
               Swift Services: OK
                  Disk Checks: OK
                  System time: OK
```

3. How do I check the health of a disk in SwiftStack?

   Login to SwiftStack Server node and run utility ‘sdt’ utility provided by SwiftStack.

```
[root@swiftstack-server2 ~]# sdt probe
Probed devices:
DEVICE : LABEL : TYPE :   GB    : BLINK :                  UUID                  : MOUNT POINT
sda    : d44   : xfs  : 10000.8 :   ?   : cda63af3-f835-41dd-83b0-5f6799fcdbcf : /srv/node/d44
sdaa   : d33   : xfs  :  800.2  :   ?   : fe78808c-5242-4fc6-b50a-fa79349ada7d : /srv/node/d33
sdab   : d23   : xfs  : 10000.8 :   ?   : b3dddcc1-066c-42f9-97fb-6119bd0171a5 : /srv/node/d23
```

   The status of individual disks can be queried as below:

```
[root@swiftstack-server2 ~]# sdt health sdab sda
d23 - write-check: OK
d23 - overall-health: OK
d23 - managed-state: OK
d44 - write-check: OK
d44 - overall-health: OK
d44 - managed-state: OK
```

4. Swiftstack shows me an alert about a bad disk that needs to be replaced. How do I replace the correct disk in Cisco UCS?

    a. Go to the Managed Swift Drives page, and on the node tab in UI and then unmount the disk and then delete the disk. Note the device name as pointed out by Swift – say /dev/sdc.

    b. Check the existence and status in cat /proc/partitions and /dev/disk/by-*

    c. You may also have an alert in Cisco UCS. Check for any missing drives in the inventory tab of the server under disks.

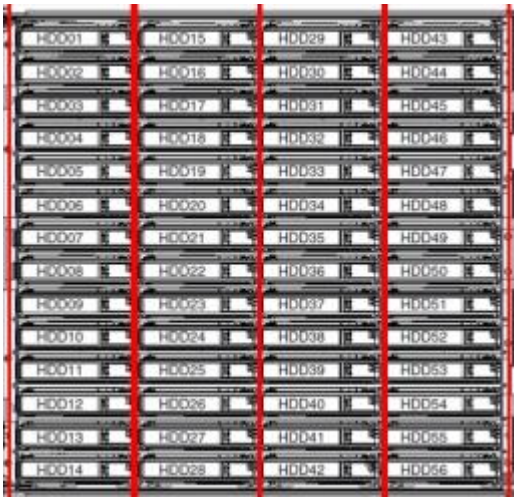
    d. You can also run storcli to confirm the disk number which will report as bad disk.

```
---------------------------------------------------------------------------
EID:Slt DID State DG       Size Intf Med SED PI SeSz Model                Sp
---------------------------------------------------------------------------
37:1     69 JBOD  -  744.125 GB SAS  SSD N   N  512B MZIES800HMHP/003     U
37:2     18 JBOD  -    9.094 TB SAS  HDD N   N  4 KB HUH721010AL4200      U
37:3     46 JBOD  -    9.094 TB SAS  HDD N   N  4 KB HUH721010AL4200      U
37:4     52 UBAD  -    9.094 TB SAS  HDD N   N  4 KB HUH721010AL4200      U
37:5     58 JBOD  -    9.094 TB SAS  HDD N   N  4 KB HUH721010AL4200      U
37:6     47 JBOD  -    9.094 TB SAS  HDD N   N  4 KB HUH721010AL4200      U
```

    e. Check the disk number in storcli or UCS inventory and replace the disk. The disk numbers physically are numbered in 4 rows in S3260 as below. Identify the appropriate disk and replace it.

    f. Format the new drive and add policies as appropriate.

    For further details please visit the page https://www.swiftstack.com/docs/cookbooks/ops/detect_and_replace_failed_drives.html.



Please note that a disk LED locator can also be turned on, in Cisco UCS. However for this the power should be on and cables should be long enough when you pull the disk cabinet.

5. Where to start looking for Swift logs?

    On storage node check /var/log/swift/all.log
    On Controller node check `/opt/ss/var/log/*`

6. How can I use Python Swift Client?

   python-swiftclient is handy to run few CLI commands and also for troubleshooting. Once python swiftclient is installed the parameters like authorization and users can be passed to swift and also can be put in bash or .profile. For instructions on how to install python-swiftclient please visit https://www.swiftstack.com/docs/integration/python-swiftclient.html.

```
[ssbench@swiftstack-client1 ~]$ swift -A http://192.168.120.202/auth/v1.0 -U ssbench -K ssbench stat -v
            StorageURL: http://192.168.120.202/v1/AUTH_ssbench
            Auth Token: AUTH_tk01fbc918567d4ed48f17c9ba287b129b
               Account: AUTH_ssbench
            Containers: 0
               Objects: 0
                 Bytes: 0
       X-Put-Timestamp: 1500895739.20067
           X-Timestamp: 1500895739.20067
            X-Trans-Id: tx016b0f60d2b349f696e05-005975d9fb
          Content-Type: text/plain; charset=utf-8
X-Openstack-Request-Id: tx016b0f60d2b349f696e05-005975d9fb
```

   Alternatively make changes to your .bashrc file as below:

```
export ST_AUTH=http://192.168.120.202/auth/v1.0
export ST_USER=ssbench
export ST_KEY=ssbench
```

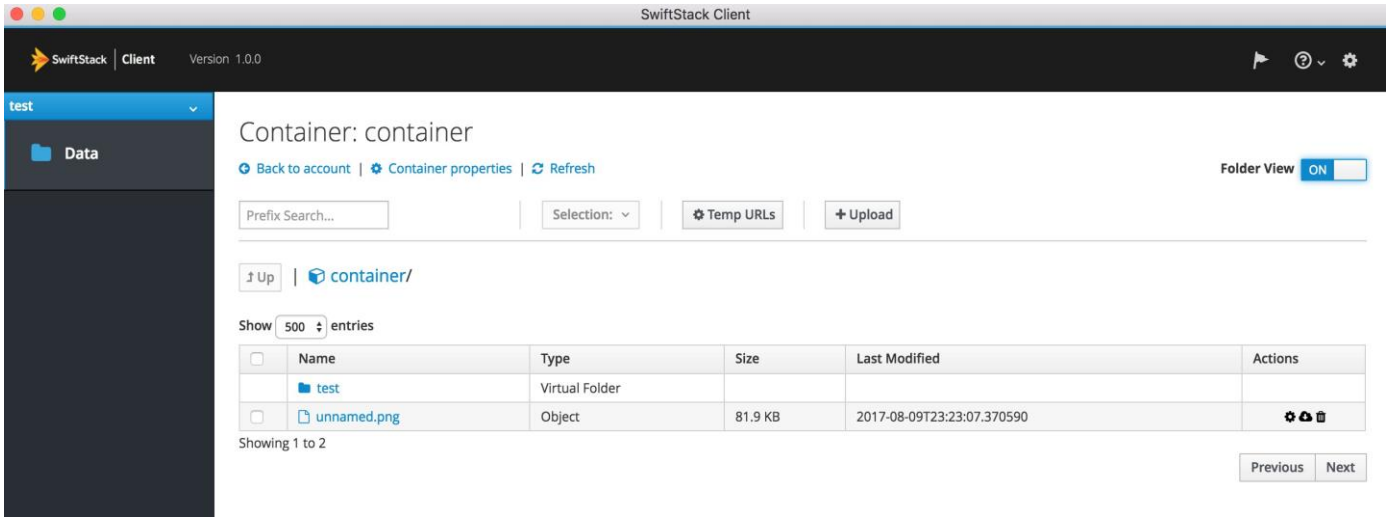
   Additional debug option can also be provided to the above URL

```
[ssbench@swiftstack-client1 ~]$ swift --debug stat
DEBUG:requests.packages.urllib3.connectionpool:Starting new HTTP connection (1): 192.168.120.202
DEBUG:requests.packages.urllib3.connectionpool:http://192.168.120.202:80 "GET /auth/v1.0 HTTP/1.1" 200 0
DEBUG:swiftclient:REQ: curl -i http://192.168.120.202/auth/v1.0 -X GET
DEBUG:swiftclient:RESP STATUS: 200 OK
```

7. Where can I download GUI version of SwiftStack Client?

   You can download it from SwiftStack web site - https://www.swiftstack.com/downloads

   A sample screen shot shown below:



8. There are several Tuning parameters in Controller node. How and what do I tune?

   Most of the times the default settings should suffice. Tuning is a bit iterative and should be done carefully. All the tests done on the test bed were with default parameters. In case you think you could extract better values from the cluster, please contact swiftstack-support team.

Appendix

Cluster.conf

Below is a sample cluster.conf file used on the test bed. The clients may have to be tweaked depending on the number of nodes and chassis's.

```
[ssbench@swiftstack-client1 ansible-ops]$ cat cluster.conf
[benchnodes:vars]
result_folder=/home/ssbench/ansible-ops/result
auth="http://swiftstack-cluster/auth/v1.0"
# If you disable auth middleware in proxy, please provide correct storage_url
# noauth="http://192.168.200.21/v1/AUTH_demo"

# ssbench option (--user_count, -u) 6BM nodes
#concurrency=50, 100
#concurrency= 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000
# for 12 clients
#concurrency= 25, 50, 100, 150, 200, 250, 300, 350, 400, 450, 500
# for 12 clients
concurrency= 600, 700, 800, 900, 1000
# for 24 clients
#concurrency= 13, 25,  50,  75, 100, 125, 150, 175, 200, 225, 250
# for 24 clients
#concurrency= 300, 350, 400, 450, 500
# ssbench option (--worker)
worker=20
# ssbench option (--run_seconds, )
duration=300

# fullness
#concurrency=100, 100, 100, 100, 100, 100, 100, 100
#duration=1800

# Running mode
# - fullness: generates different container name for each run.
#              scenario folder: 'scenarios/fullness'
# - normal:   use default scenario files
#             scenario folder: 'scenarios/normal'
#mode=fullness
mode=normal


[swiftnodes:vars]
# User for couting backend errors and clean disks
ansible_ssh_user=root
ansible_ssh_password=root

# Allow clean all swift disk data
#allow_clean_disks=True
allow_clean_disks=False


[benchnodes]
192.168.120.231 user="ssbench1" key="ssbench"
```

```
192.168.120.232 user="ssbench2" key="ssbench"
192.168.120.233 user="ssbench3" key="ssbench"
192.168.120.234 user="ssbench4" key="ssbench"
192.168.120.235 user="ssbench5" key="ssbench"
192.168.120.236 user="ssbench6" key="ssbench"

192.168.120.109 user="ssbench7" key="ssbench"
192.168.120.110 user="ssbench8" key="ssbench"
192.168.120.111 user="ssbench9" key="ssbench"
192.168.120.112 user="ssbench10" key="ssbench"
192.168.120.113 user="ssbench11" key="ssbench"
192.168.120.114 user="ssbench12" key="ssbench"

192.168.120.131 user="ssbench13" key="ssbench"
192.168.120.132 user="ssbench14" key="ssbench"
192.168.120.137 user="ssbench15" key="ssbench"
192.168.120.138 user="ssbench16" key="ssbench"
192.168.120.139 user="ssbench17" key="ssbench"
192.168.120.140 user="ssbench18" key="ssbench"
192.168.120.141 user="ssbench19" key="ssbench"
192.168.120.142 user="ssbench20" key="ssbench"
192.168.120.143 user="ssbench21" key="ssbench"
192.168.120.144 user="ssbench22" key="ssbench"
192.168.120.145 user="ssbench23" key="ssbench"
192.168.120.146 user="ssbench24" key="ssbench"


[swiftnodes]
192.168.120.201 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.201
192.168.120.202 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.202
192.168.120.203 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.203
192.168.120.204 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.204
192.168.120.205 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.205
192.168.120.206 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.206
192.168.120.207 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.207
192.168.120.208 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.208
192.168.120.209 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.209
192.168.120.210 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.210
192.168.120.211 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.211
192.168.120.212 type=proxy,object,account,container out-
ward_facing_ip=192.168.120.212
[ssbench@swiftstack-client1 ansible-ops]$
```

About the Authors

Ramakrishna Nishtala, Cisco Systems, Inc.

Ramakrishna Nishtala is a Technical Leader in Cisco UCS and Data Center solutions group and has over 20 years of experience in IT infrastructure, Automation, Virtualization and Cloud computing. In his current role at Cisco Systems, he works on best practices, optimization and performance tuning on OpenStack and other Open Source solutions like Swift, Ceph, dockers, etc. on Cisco UCS platforms. Prior to this he was involved in data center migration strategies, compute and storage consolidation, end-to-end performance optimization on databases, application and web servers and solutions engineering.

Acknowledgements

- Jawwad Memon, Cisco Systems, Inc.
- Erik Pounds, SwiftStack
- Hiren Chandramani, SwiftStack
- Johnny Wang, SwiftStack
- Martin Lanner, SwiftStack