Hadoop as a Service (HaaS) with Cisco UCS Common Platform Architecture (CPA v2) for Big Data and OpenStack

Last Updated: August 1, 2014

Cisco Validated Design

Building Architectures to Solve Business Problems

About the Authors

Raghunath Nambiar

Raghunath Nambiar, Distinguished Engineer, Data Center Business Group (Cisco Systems)

Raghunath Nambiar is a Distinguished Engineer at Cisco's Data Center Business Group. His current responsibilities include emerging technologies and big data strategy.

Karthik Kulkarni

Karthik Kulkarni, Technical Marketing Engineer, Data Center Business Group (Cisco Systems)

Karthik Kulkarni is a Technical Marketing Engineer with role as a BigData Solutions Architect in the Data Center Solutions Group at Cisco Systems. His main focus is on Architecture and Solutions on Big data related technologies, Infrastructure and performance.

Manankumar Trivedi

Manankumar Trivedi, Performance Engineer, Data Center Business Group (Cisco Systems)

Manankumar Trivedi is a Performance Engineer in the Data Center Solutions Group at Cisco Systems. He is part of solution engineering team focusing on big data infrastructure and performance.

About Cisco Validated Design (CVD) Program

The CVD program consists of systems and solutions designed, tested, and documented to facilitate faster, more reliable, and more predictable customer deployments. For more information visit: http://www.cisco.com/go/designzone.

CVD Document Title

Acknowledgment

The authors acknowledge Ashwin Manjunatha, Ajay Singh, Mehul Bhatt, Samantha Jian-Pielak, Marc Solanas Tarre, Debo Dutta and Sindhu Sudhir for their contributions in developing this document.

Hadoop as a Service (HaaS) with Cisco UCS Common Platform Architecture (CPA v2) for Big Data and OpenStack

Introduction

Hadoop has become a strategic data platform embraced by mainstream enterprises as it offers a path for businesses to unlock value in big data while maximizing existing investments. The Hortonworks Data Platform 2.0 (HDP 2.0) is a 100% open source distribution of Apache Hadoop that is built, tested and hardened with enterprise rigor. The combination of HDP and Cisco UCS provides an industry-leading platform for Hadoop based application deployments. Hadoop as a Service is new to the industry but is gaining traction in many Service Providers and IT Organizations. This CVD focuses on setting up OpenStack on Ubuntu to deploy and manage Hadoop as a Service on Cisco UCS Common Platform Architecture version 2 (CPA v2).

Audience

This document describes the architecture and deployment procedures of Hadoop as a Service (HaaS) with OpenStack and Hortonworks Data Platform 2.0 (HDP 2.0) for Hadoop on a 64-node cluster based on CPA v2 for Big Data. The intended audience of this document include, but is not limited to, sales engineers, field consultants, professional services, IT managers, partner engineering and customers who want to deploy Hadoop as a Service with HDP 2.0 on Cisco UCS CPA v2 for Big Data.

Big Data

The Cisco UCS solution for HDP 2.0 is based on CPA v2 for Big Data, is a highly scalable architecture designed to meet a variety of scale-out application demands with seamless data integration and management integration capabilities built using the following components:

- **Cisco UCS 6200 Series Fabric Interconnects**—provide high-bandwidth, low-latency connectivity for servers, with integrated, unified management provided for all connected devices by Cisco UCS Manager. Deployed in redundant pairs, Cisco fabric interconnects offer the full active-active

Corporate Headquarters:
Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA

redundancy, performance, and exceptional scalability needed to support the large number of nodes that are typical in clusters serving big data applications. Cisco UCS Manger enables rapid and consistent server configuration using service profiles, automating ongoing system maintenance activities such as firmware updates across the entire cluster as a single operation. Cisco UCS Manager also offers advanced monitoring with options to raise alarms and send notifications about the health of the entire cluster.

- **Cisco UCS 2200 Series Fabric Extenders**—extend the network into each rack, acting as remote line cards for fabric interconnects and providing highly scalable and extremely cost-effective connectivity for a large number of nodes.
- **Cisco UCS C-Series Rack Mount Servers**—are 2-socket servers based on Intel Xeon E-2600 v2 series processors and supporting up to 768GB of main memory. 24 Small Form Factor (SFF) disk drives are supported in performance optimized option and 12 Large Form Factor (LFF) disk drives are supported in capacity option, along with 4 Gigabit Ethernet LAN-on-motherboard (LOM) ports.
- **Cisco UCS Virtual Interface Cards (VICs)**—unique to Cisco, Cisco UCS Virtual Interface Cards incorporate next-generation converged network adapter (CNA) technology from Cisco, and offer dual 10Gbps ports designed for use with Cisco UCS C-Series Rack-Mount Servers. Optimized for virtualized networking, these cards deliver high performance and bandwidth utilization and support up to 256 virtual devices.
- **Cisco UCS Manager**—resides within the Cisco UCS 6200 Series Fabric Interconnects. It makes the system self-aware and self-integrating, managing all of the system components as a single logical entity. Cisco UCS Manager can be accessed through an intuitive graphical user interface (GUI), a command-line interface (CLI), or an XML application-programming interface (API). Cisco UCS Manager uses service profiles to define the personality, configuration, and connectivity of all resources within Cisco UCS, radically simplifying provisioning of resources so that the process takes minutes instead of days. This simplification allows IT departments to shift their focus from constant maintenance to strategic business initiatives.

Hortonworks Data Platform (HDP 2.0)

Apache Hadoop is an open-source software framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. The Hortonworks Data Platform2.0 (HDP 2.0) is an enterprise-grade, Apache Hadoop distribution that enables storing, processing, and managing large data sets. The HDP 2.0 combines Apache Hadoop and it's related projects into a single tested and certified package.

Hortonworks: Key Features and Benefits

With HDP 2.0, enterprises can retain and process large amounts of data, join new and existing data sets, and lower the cost of data analysis compared to traditional solutions. Hortonworks enables enterprises to implement the following data management principles:

- Retain as much data as possible. Traditional data warehouses age, and over time will eventually store only summary data. Analyzing detailed records is often critical to uncovering useful business insights.
- Join new and existing data sets. Enterprises can build large-scale environments for transactional data with analytic databases, but these solutions are not always well suited to processing nontraditional data sets such as text, images, machine data, and online data. Hortonworks enables enterprises to incorporate both structured and unstructured data in one comprehensive data management system.

- Archive data at low cost. It is not always clear what portion of stored data will be of value for future analysis. Therefore, it can be difficult to justify expensive processes to capture, cleanse, and store that data. Hadoop scales easily, so you can store years of data without much incremental cost, and find deeper patterns that your competitors may miss.
- Access all data efficiently. Data needs to be readily accessible. Apache Hadoop clusters can provide a low-cost solution for storing massive data sets while still making the information readily available. Hadoop is designed to efficiently scan all of the data, which is complimentary to databases that are efficient at finding subsets of data.
- Apply data cleansing and data cataloging. Categorize and label all data in Hadoop with enough descriptive information (metadata) to make sense of it later, and to enable integration with transactional databases and analytic tools. This greatly reduces the time and effort of integrating with other data sets, and avoids a scenario in which valuable data is eventually rendered useless.
- Integrate with existing platforms and applications. There are many business intelligence (BI) and analytic tools available, but they may not be compatible with your particular data warehouse or DBMS. Hortonworks connects seamlessly with many leading analytic, data integration, and database management tools.

The HDP 2.0 is the foundation for the next-generation enterprise data architecture – one that addresses both the volume and complexity of today's data.

Canonical Ubuntu 12.04 LTS Release

Also known by its code name "Precise Pangolin", Ubuntu 12.04 is a Long Term Support (LTS) release from Canonical. The support for Ubuntu 12.04 is expected to continue till April 2017, hence providing a long term robust support framework for customers. For open source projects, support is a crucial component, and Canonical provides enterprise level scale, stability and support for underlying Operating System as well as OpenStack components for cloud deployment on Ubuntu.

Canonical Ubuntu OpenStack Architecture benefits

Canonical OpenStack Platform on Canonical Ubuntu 12.04 provides the foundation to build a private or public Infrastructure as a Service (IaaS) for cloud-enabled workloads. It allows organizations to leverage OpenStack, the largest and fastest growing open source cloud infrastructure project, while maintaining the security, stability, and enterprise readiness of a platform built on Canonical Ubuntu 12.04.

Canonical Ubuntu OpenStack Platform gives organizations a open framework for hosting cloud workloads. In conjunction with other Ubuntu technologies, Canonical Ubuntu OpenStack Platform allows organizations to move from traditional workloads to cloud-enabled workloads on their own terms and timelines, as their applications require.

Canonical Ubuntu OpenStack Platform provides a certified ecosystem of hardware, software, and services, an enterprise lifecycle that extends the community OpenStack release cycle, and Canonical support on both the OpenStack modules and their underlying Linux dependencies.

Hadoop as a Service (HaaS) Architecture Overview

The Architecture for Hadoop as a Service is outlined in this section.

Canonical Ubuntu 12.04 LTS is both the host OS and guest OS on top of OpenStack. the tested configuration is based on Cisco UCS CPA v2 for Big Data optimized for Capacity with Flash Memory. http://www.cisco.com/c/dam/en/us/solutions/collateral/borderless-networks/advanced-services/common_platform_architecture.pdf. OpenStack release used for this solution is Havana. OpenStack components used are Keystone for Identity Service, Glance for VM Image service, Nova for compute (nova-compute uses KVM as the hypervisor), Storage is ephemeral (storage that is local to the VM and is deleted when the VM is terminated), networking is nova-network, which is a FlatNetwork and Horizon is used for OpenStack dashboard. Hortonworks 2.0 is installed manually on the guest VMs.

Note This CVD goes with Ephemeral storage for data storage. One of the reasons to go with Ephemeral as storage was to have compute local to storage as this is fundamental to Hadoop, else this would cause unwanted network traffic slowing the performance. Data redundancy is managed inherently by HDFS (Hadoop Filesystem). There are other solutions on top of OpenStack, which can provide storage to be local to compute or provide an alternate to HDFS with hooks to Hadoop MapReduce which are beyond the scope of this CVD.

OpenStack and Hadoop Layout

Of the 64 nodes (physical servers), one of the node is going to be Controller node for OpenStack (A controller node basically runs all services necessary to manage compute nodes which host VMs. Compute nodes run only minimal services and refer to Controller node for all querying and managing VM lifecycle within the individual servers. Typically a controller node does not host VMs. In other words, controller node does not run nova-compute; however, in this architecture Hadoop Namenode is run as a Single VM on the controller node. Hence, to run this single VM we need to run nova-compute services on controller node as well.

In order to host Hadoop Master services (namely Namenode and Resource Manager), two nodes run single VM to host Hadoop Namenode and Hadoop Resource Manager/Secondary Namenode (and other Master services discussed later). All other compute nodes host multiple VMs which will be used for data and tasks.

Note If running a smaller cluster, say less than 16 nodes, all Hadoop Master Services can be placed in a single VM on one node, which is the OpenStack controller node. However, Hadoop data and task (job) services are not run on this VM.

Solution Overview

The current version of the Cisco UCS CPA v2 for Big Data offers the following configuration depending on the compute and storage requirements:

Table 1 Cisco UCS CPA v2 Configuration Details

Performance and Capacity Balanced	Capacity Optimized	Capacity Optimized with Flash Memory
16 Cisco UCS C240 M3 Rack Servers, each with: • 2 Intel Xeon processors E5-2660 v2 • 256GB of memory • LSI MegaRaid 9271CV 8i card • 24 1TB 7.2K SFF SAS drives (384TB total)	16 Cisco UCS C240 M3 Rack Servers, each with: • 2 Intel Xeon processors E5-2660 v2 • 128GB of memory • LSI MegaRaid 9271CV 8i card • 12 4TB 7.2K LFF SAS drives (786TB total)	16 Cisco UCS C240 M3 Rack Servers, each with: • 2 Intel Xeon processors E5-2660 v2 • 128GB of memory • Cisco UCS Nytro MegaRAID 200GB Controller • 12 4TB 7.2K LFF SAS drives (786TB total)

Note This CVD describes the install process for a 64 node Capacity Optimized with Flash Memory Cluster Configuration with 256GB Memory.

The Capacity Optimized with Flash memory cluster configuration consists of the following:

- Two Cisco UCS 6296UP Fabric Interconnects
- Eight Cisco Nexus 2232PP Fabric Extenders (two per rack)
- 64 UCS C240 M3 Rack-Mount servers (16 per rack)
- Four Cisco R42610 standard racks
- Eight Vertical Power distribution units (PDUs) - Country Specific

Rack and PDU Configuration

Each rack consists of two vertical PDUs. The master rack consists of two Cisco UCS 6296UP Fabric Interconnects, two Cisco Nexus 2232PP Fabric Extenders and sixteen Cisco UCS C240M3 Servers, connected to each of the vertical PDUs for redundancy; thereby, ensuring availability during power source failure. The expansion racks consists of two Cisco Nexus 2232PP Fabric Extenders and sixteen Cisco UCS C240M3 Servers are connected to each of the vertical PDUs for redundancy; thereby, ensuring availability during power source failure, similar to the master rack.

Note Please contact your Cisco representative for country specific information.

Table 2 and Table 3 describe the rack configurations of rack 1 (master rack) and racks 2-4 (expansion racks).

Table 2 Rack 1 - Master Rack

Cisco 42U Rack	Master Rack
42	Cisco UCS FI 6296UP
41	
40	Cisco UCS FI 6296UP
39	
38	Cisco Nexus FEX 2232PP
37	Cisco Nexus FEX 2232PP
36	Unused
35	Unused
34	Unused
33	Unused
32	Cisco UCS C240 M3
31	Cisco UCS C240 M3
30	Cisco UCS C240 M3
29	Cisco UCS C240 M3
28	Cisco UCS C240 M3
27	Cisco UCS C240 M3
26	Cisco UCS C240 M3
25	Cisco UCS C240 M3
24	Cisco UCS C240 M3
23	Cisco UCS C240 M3
22	Cisco UCS C240 M3
21	Cisco UCS C240 M3
20	Cisco UCS C240 M3
19	Cisco UCS C240 M3
18	Cisco UCS C240 M3
17	Cisco UCS C240 M3
16	Cisco UCS C240 M3
15	Cisco UCS C240 M3
14	Cisco UCS C240 M3
13	Cisco UCS C240 M3
12	Cisco UCS C240 M3
11	Cisco UCS C240 M3
10	Cisco UCS C240 M3
9	Cisco UCS C240 M3
8	Cisco UCS C240 M3

Table 2 Rack 1 - Master Rack

Cisco 42U Rack	Master Rack
7	Cisco UCS C240 M3
6	Cisco UCS C240 M3
5	Cisco UCS C240 M3
4	Cisco UCS C240 M3
3	Cisco UCS C240 M3
2	Cisco UCS C240 M3
1	Cisco UCS C240 M3

Table 3 Rack 2-4 - Expansion Racks

Cisco UCS 42U Rack	Expansion Rack
42	Unused
41	Unused
40	Unused
39	Unused
38	Cisco Nexus FEX 2232PP
37	Cisco Nexus FEX 2232PP
36	Unused
35	Unused
34	Unused
33	Unused
32	Cisco UCS 240 M3
31	
30	Cisco UCS 240 M3
29	
28	Cisco UCS 240 M3
27	
26	Cisco UCS 240 M3
25	
24	Cisco UCS 240 M3
23	
22	Cisco UCS 240 M3
21	
20	Cisco UCS 240 M3
19	

Table 3 Rack 2-4 - Expansion Racks

Cisco UCS 42U Rack	Expansion Rack
18	Cisco UCS 240 M3
17	
16	Cisco UCS 240 M3
15	
14	Cisco UCS 240 M3
13	
12	Cisco UCS 240 M3
11	
10	Cisco UCS 240 M3
9	
8	Cisco UCS 240 M3
7	
6	Cisco UCS 240 M3
5	
4	Cisco UCS 240 M3
3	
2	Cisco UCS 240 M3
1	

Server Configuration and Cabling

This CVD focuses on the architecture for Canonical Ubuntu OpenStack on UCS platform using Cisco UCS C-Series Servers for both compute and storage. Cisco UCS C240 M3 servers are used as compute, storage and controller nodes (from OpenStack perspective). UCS C-Series Servers are managed by UCS Manager, which provides ease of infrastructure management, and built-in network high availability.

The C240 M3 rack server is equipped with Intel Xeon E5-2660 v2 processors, 256GB of memory, Cisco UCS Virtual Interface Card 1225, Cisco LSI Nytro MegaRAID 8110-4i with 200GB Flash storage controller and 12 x 4TB 7.2K SATA disk drives.

Figure 1 illustrates the ports on the Cisco Nexus 2232PP Fabric Extender connecting to the Cisco UCS C240 M3 Servers. Sixteen Cisco UCS C240 M3 servers are used in Master rack configurations.

Figure 1 Topology Diagram of Cisco UCS C240 M3 Server

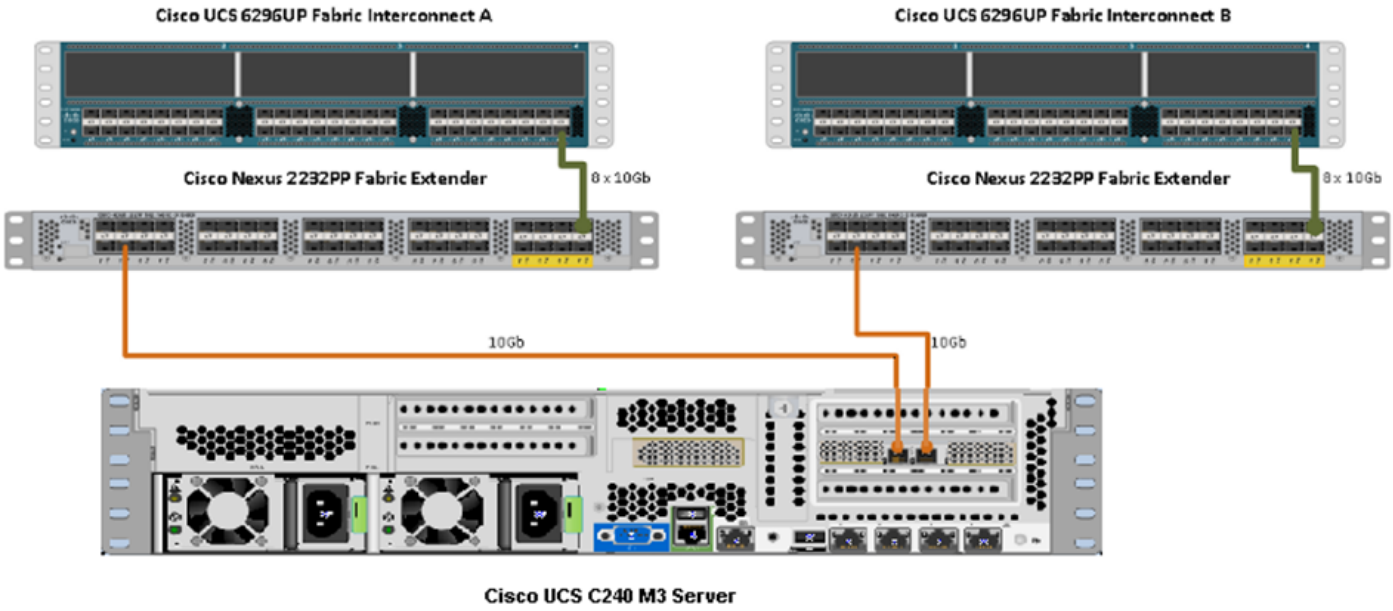
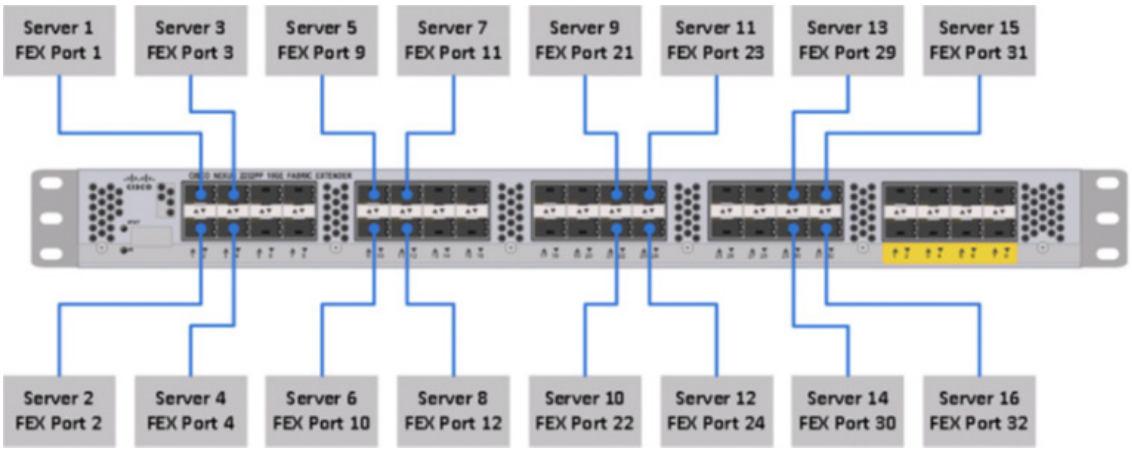


Figure 2 illustrates the port connectivity between the Cisco Nexus 2232PP fabric extender and Cisco UCS C240M3 server.

Figure 2 Connectivity Diagram of Cisco Nexus 2232PP FEX and Cisco UCS C240 M3 Servers



For more information on physical connectivity and single-wire management see:

http://www.cisco.com/en/US/docs/unified_computing/ucs/c-series_integration/ucsm2.1/b_UCSM2-1_C-Integration_chapter_010.html

For more information on physical connectivity illustrations and cluster setup, see:

http://www.cisco.com/en/US/docs/unified_computing/ucs/c-series_integration/ucsm2.1/b_UCSM2-1_C-Integration_chapter_010.html#reference_FE5B914256CB4C47B30287D2F9CE3597

Figure 3 depicts a 64 node cluster. Each link in the figure represents 8 x 10 Gigabit links.

Figure 3 64 Node Cluster Configuration

Software Distributions and Versions

The software distributions required versions are listed below.

Hortonworks Data Platform (HDP 2.0)

The Hortonworks Data Platform supported is HDP 2.0. For more information visit http://www.hortonworks.com

Canonical Ubuntu 12.04 LTS Release

The operating system supported is Ubuntu 12.04.4 Server. For more information visit http://www.ubuntu.com

Software Versions

The software versions tested and validated in this document are listed in Table 4.

Table 4 Software Versions

Layer	Component	Version or Release
Compute	Cisco UCS C240 M3	1.5.4f

Table 4 Software Versions

Layer	Component	Version or Release
Network	Cisco UCS FI 6296UP	UCS 2.2(1b)A
	Cisco UCS VIC 1225 Firmware	2.2(1b)
	Cisco UCS VIC 1225 Driver	2.1.1.41
	Cisco Nexus 2232PP FEX	5.2(3)N2(2.21b)
Storage	UCS Nytro MegaRAID 200GB Controller Firmware	23.26.0.04
	UCS Nytro MegaRAID 200GB Controller Driver	06.602.03.00
	Ubuntu Linux Server (x86_64)	12.04.4LTS Server
	Cisco UCS Manager	2.2(1b)
Software	HDP	2.0.6
	OpenStack	Havana

Note The latest drivers can be downloaded from the link below: http://software.cisco.com/download/release.html?mdfid=284296254&flowid=31743&softwareid=283853158&release=1.5.1&relind=AVAILABLE&rellifecycle=&reltype=latest

Fabric Configuration

This section provides details for configuring a fully redundant, highly available Cisco UCS 6296 fabric configuration.

1. Initial setup of the Fabric Interconnect A and B.
2. Connect to IP address of Fabric Interconnect A using web browser.
3. Launch UCS Manager.
4. Edit the chassis discovery policy.
5. Enable server and uplink ports.
6. Create pools and polices for service profile template.
7. Create service profile template and 64 service profiles.
8. Start discover process.
9. Associate to server.

Note Make sure that the links, L1 of Fabric A is connected to L1 of Fabric B and L2 of Fabric A is connected to L2 of Fabric B.

Performing Initial Setup of Cisco UCS 6296 Fabric Interconnects

This section describes the steps to perform initial setup of the Cisco UCS 6296 Fabric Interconnects A and B.

Configure Fabric Interconnect A

Follow these steps to configure Fabric Interconnect A:

1. Connect to the console port on the first Cisco UCS 6296 Fabric Interconnect.
2. At the prompt to enter the configuration method, enter console to continue.
3. If asked to either perform a new setup or restore from backup, enter setup to continue.
4. Enter y to continue to set up a new Fabric Interconnect.
5. Enter y to enforce strong passwords.
6. Enter the password for the admin user.
7. Enter the same password again to confirm the password for the admin user.
8. When asked if this fabric interconnect is part of a cluster, answer y to continue.
9. Enter A for the switch fabric.
10. Enter the cluster name for the system name.
11. Enter the Mgmt0 IPv4 address.
12. Enter the Mgmt0 IPv4 netmask.
13. Enter the IPv4 address of the default gateway.
14. Enter the cluster IPv4 address.
15. To configure DNS, answer y.
16. Enter the DNS IPv4 address.
17. Answer y to set up the default domain name.
18. Enter the default domain name.
19. Review the settings that were printed to the console, and if they are correct, answer yes to save the configuration.
20. Wait for the login prompt to make sure the configuration has been saved.

Configure Fabric Interconnect B

Follow these steps to configure Fabric Interconnect B:

1. Connect to the console port on the second Cisco UCS 6296 Fabric Interconnect.
2. When prompted to enter the configuration method, enter console to continue.
3. The installer detects the presence of the partner Fabric Interconnect and adds this fabric interconnect to the cluster. Enter y to continue the installation.
4. Enter the admin password that was configured for the first Fabric Interconnect.
5. Enter the Mgmt0 IPv4 address.
6. Answer yes to save the configuration.
7. Wait for the login prompt to confirm that the configuration has been saved.

For more information on configuring Cisco UCS 6200 Series Fabric Interconnect, see:

http://www.cisco.com/en/US/docs/unified_computing/ucs/sw/gui/config/guide/2.0/b_UCSM_GUI_Configuration_Guide_2_0_chapter_0100.html

Logging Into Cisco UCS Manager

Follow these steps to login to Cisco UCS Manager.

Open a Web browser and navigate to the Cisco UCS 6296 Fabric Interconnect cluster address.

Click the Launch link to download the Cisco UCS Manager software.

If prompted to accept security certificates, accept as necessary.

When prompted, enter admin for the username and enter the administrative password.

Click Login to log in to the Cisco UCS Manager.

Upgrading UCS Manager Software to Version 2.2(1b)

This document assumes the use of UCS 2.2(1b). Make sure you have upgraded the Cisco UCS Manager software and Cisco UCS 6296UP Fabric Interconnect to version 2.2(1b)

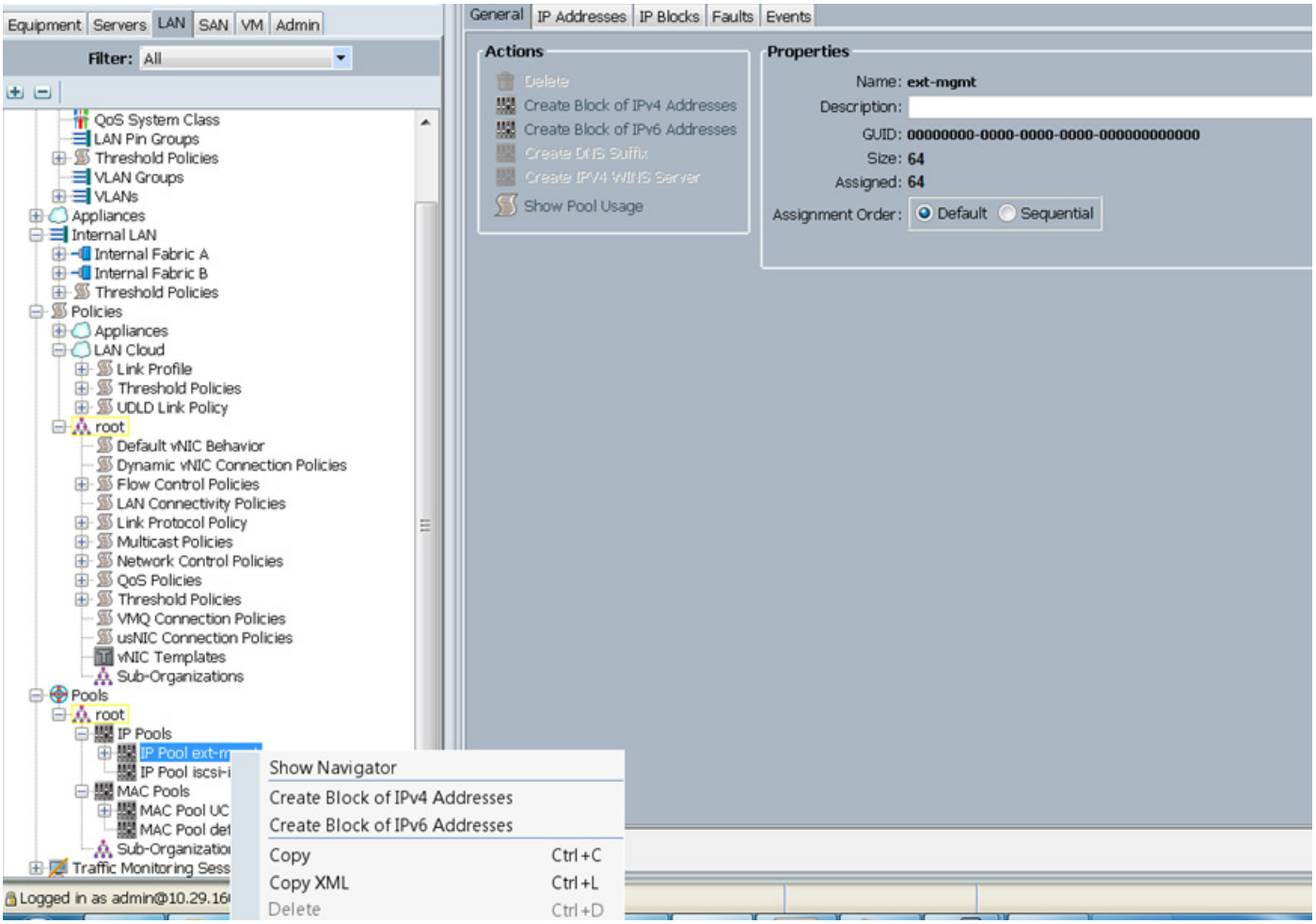
Adding Block of IP Addresses for KVM Access

The procedure discussed here provides you the details for creating a block of KVM IP addresses for server access in the Cisco UCS environment.

Follow these steps to create a block of IP addresses:

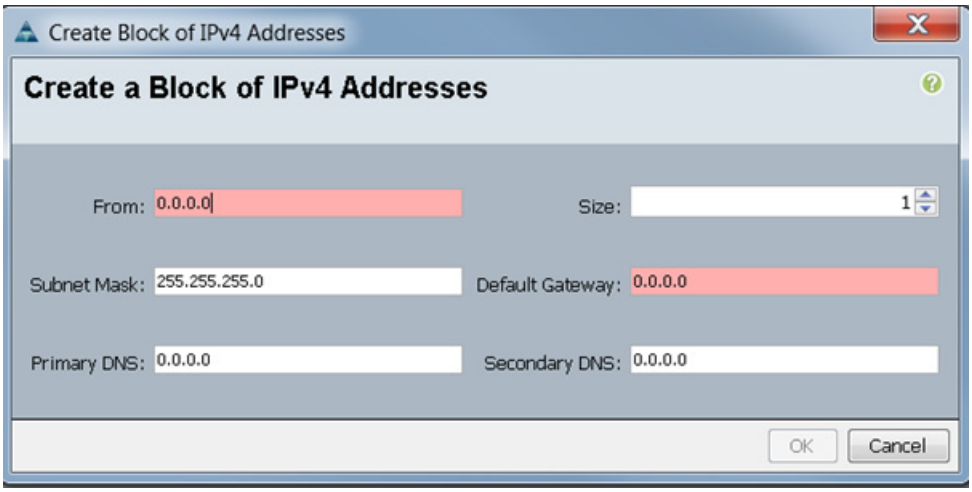
1. Select the **LAN** tab at the top of the left window.
2. Select **Pools > IpPools > Ip Pool ext-mgmt**.
3. Right-click on **IP Pool ext-mgmt**.
4. Select **Create Block of IPv4 Addresses**.

Figure 4 Adding a Block of IPv4 Addresses for KVM Access - Part 1



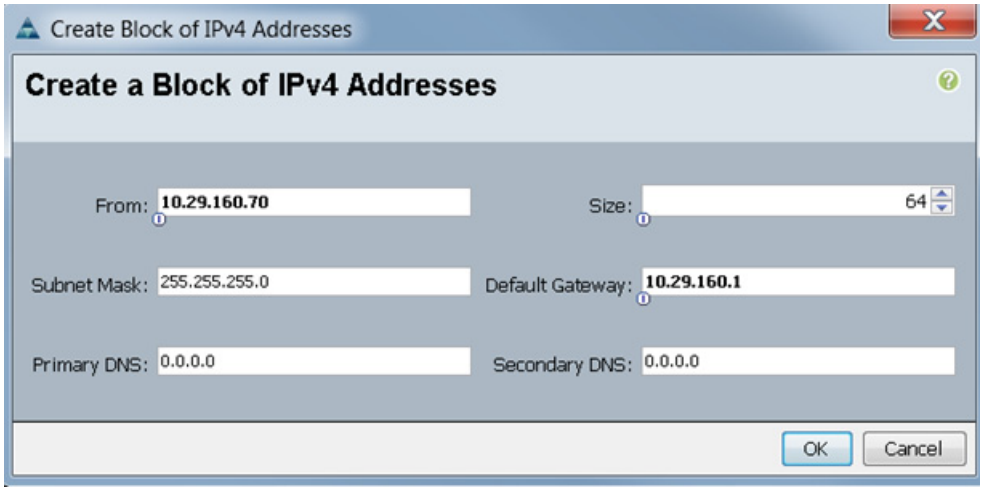
5. Enter the starting IP address of the block and number of IPs needed, as well as the subnet and gateway information.

Figure 5 Adding Block of IPv4 Addresses for KVM Access - Part 2



6. Click **OK** to create the IP block.
7. Click **OK** in the message box.

Figure 6 Adding Block of IPv4 Addresses for KVM Access - Part 3



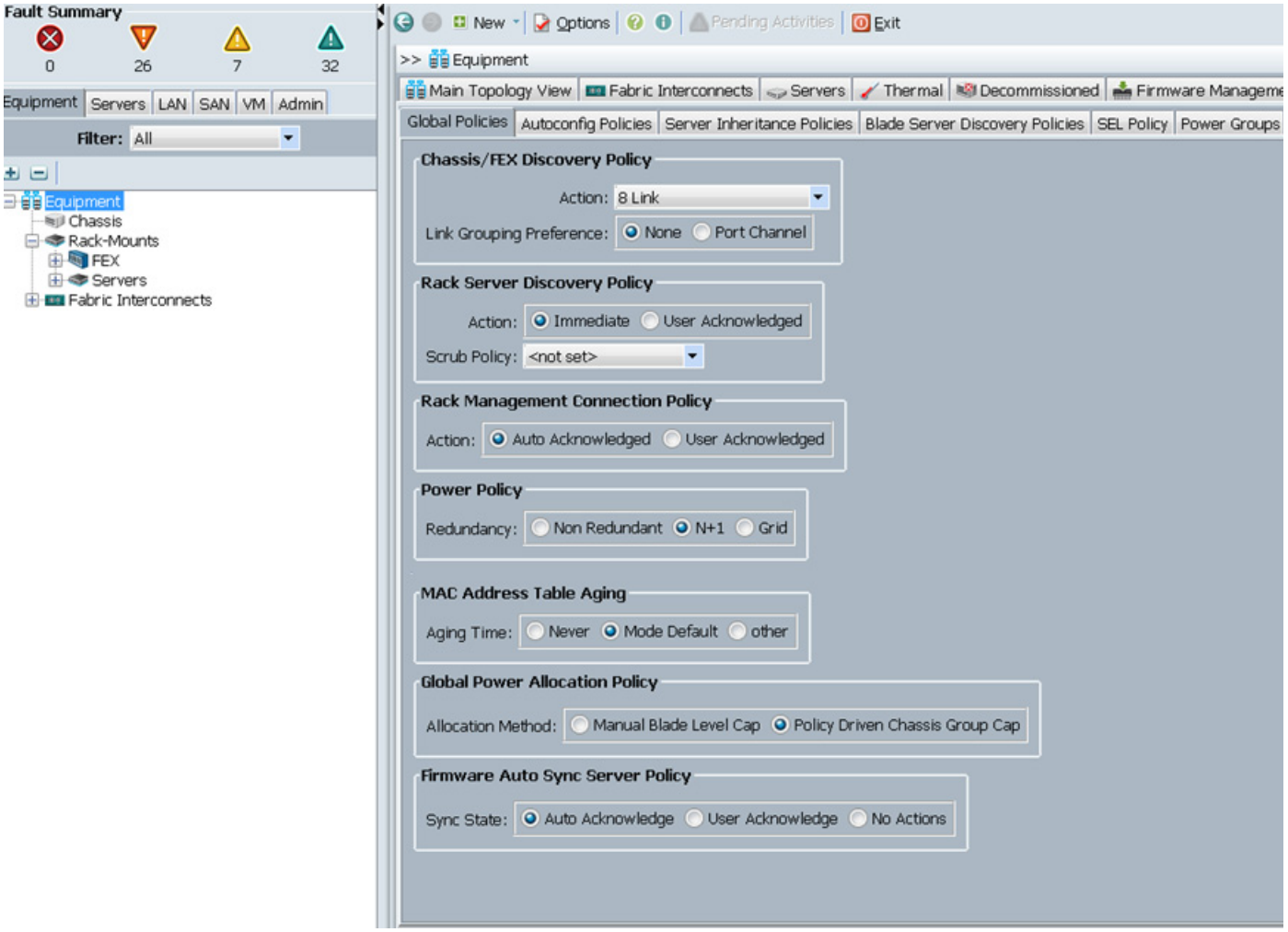
Editing Chassis/FEX Discovery Policy

The procedure discussed here provides you the details for modifying the chassis discovery policy. Setting the discovery policy now will simplify server for the future B-Series UCS Chassis and additional Cisco UCS Fabric Extenders for further C-Series connectivity.

Follow these steps to edit the Chassis Discovery Policy:

1. Navigate to the **Equipment** tab in the left pane.
2. In the right pane, click the **Policies** tab.
3. Under Global Policies, change the Chassis/FEX Discovery Policy to 8-link.
4. Click **Save Changes** in the bottom right hand corner.
5. Click **OK**.

Figure 7 Chassis/FEX Discovery Policy



Enabling Server Ports and Uplink Ports

The procedure discussed here provides you the details for enabling server and uplinks ports.

Follow these steps to enable server ports and uplink ports:

1. Select the **Equipment** tab on the top left of the window.
2. Select **Equipment > Fabric Interconnects > Fabric Interconnect A** (primary) **> Fixed Module**.
3. Expand the Unconfigured Ethernet Ports section.
4. Select all the ports that are connected to the Cisco 2232 FEX (8 per FEX), right-click on them, and select **Reconfigure > Configure as a Server Port**.
5. Select **port 1** that is connected to the uplink switch, right-click, then select **Reconfigure > Configure as Uplink Port**.
6. Select **Show Interface** and select **10GB** for Uplink Connection.
7. A pop-up window appears to confirm your selection. Click **Yes** then **OK** to continue.

8. Select **Equipment** > **Fabric Interconnects** > **Fabric Interconnect B** (subordinate) > **Fixed Module**.
9. Expand the Unconfigured Ethernet Ports section.
10. Select all the ports that are connected to the Cisco 2232 Fabric Extenders (8 per Fex), right-click on them, and select **Reconfigure** > **Configure as Server Port**.
11. A prompt displays asking if this is what you want to do. Click **Yes** then **OK** to continue.
12. Select **port 1**, which is connected to the uplink switch, right-click, then select **Reconfigure** > **Configure as Uplink Port**.
13. Select **Show Interface** and select **10GB** for Uplink Connection.
14. A pop-up window appears to confirm your selection. Click **Yes** then **OK** to continue.

Figure 8 Enabling Server Ports

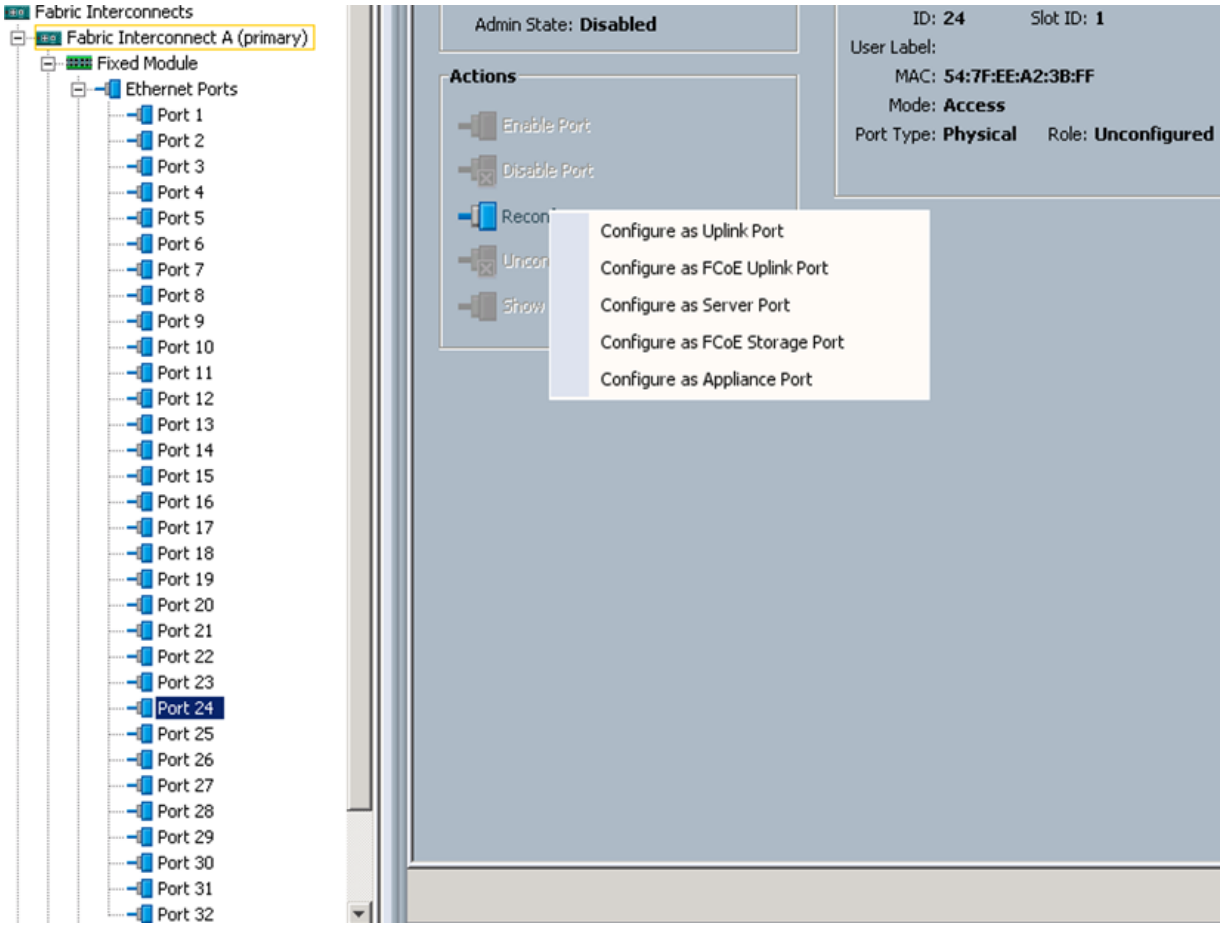
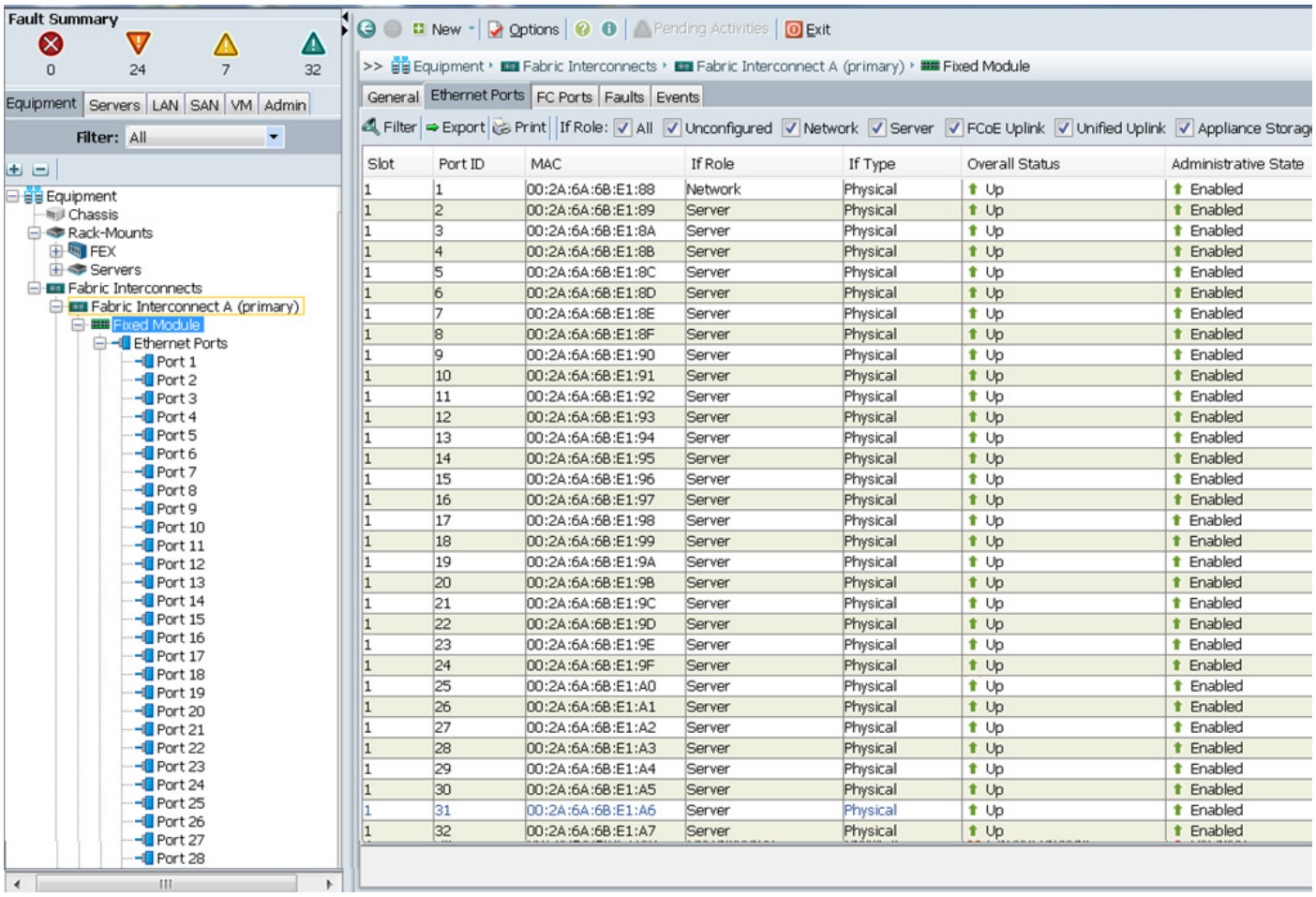


Figure 9 Showing Servers and Uplink Ports



Creating Pools for Service Profile Templates

The procedure discussed in this section provides you the details for creating Organizations, pools, and VLAN configuration for Service profile Templates.

Creating an Organization

Organizations are used as a means to arrange and restrict access to various groups within the IT organization, thereby enabling multi-tenancy of the compute resources. This document does not assume the use of Organizations; however, the necessary steps are provided for future reference.

Follow these steps to configure an organization within the Cisco UCS Manager:

1. Click **New** on the top left corner in the right pane in the UCS Manager GUI.
2. Select **Create Organization** from the options.
3. Enter a name for the organization.
4. (Optional) Enter a description for the organization.

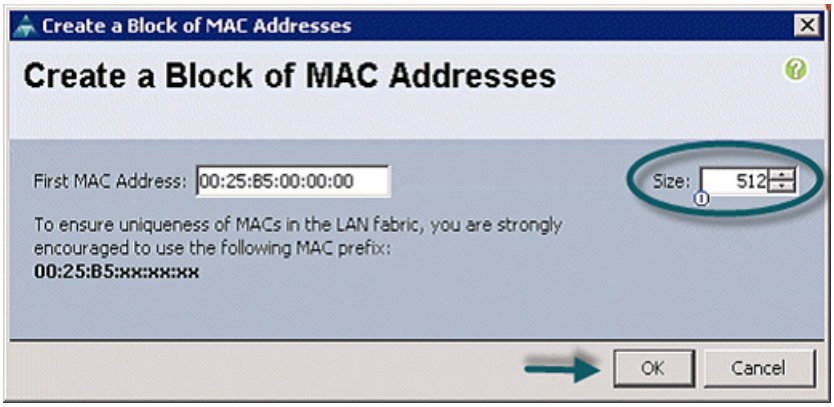
5. Click **OK**.
6. Click **OK** in the success message box.

Creating MAC Address Pools

Follow these steps to create MAC address pools:

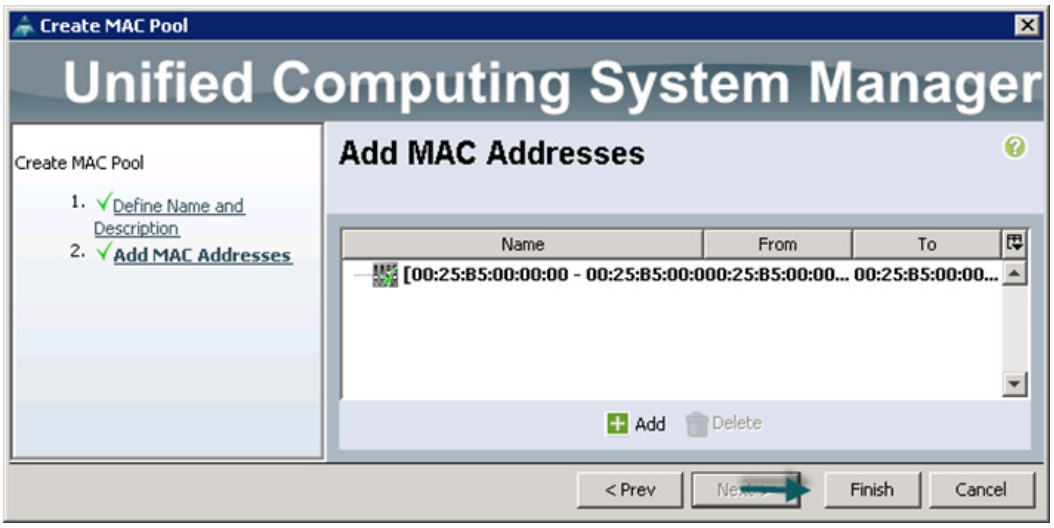
1. Select the **LAN** tab on the left of the window.
2. Select **Pools** > **root**.
3. Right-click on MAC Pools under the root organization.
4. Select **Create MAC Pool** to create the MAC address pool. Enter ucs for the name of the MAC pool.
5. (Optional) Enter a description of the MAC pool.
6. Click **Next**.
7. Click **Add**.
8. Specify a starting MAC address.
9. Specify a size of the MAC address pool, which is sufficient to support the available server resources.
10. Click **OK**.

Figure 10 Specifying first MAC Address and Size



11. Click **Finish**.

Figure 11 Adding MAC Addresses



12. When the message box displays, click **OK**.

Configuring VLANs

VLANs are configured as in shown in Table 5.

Table 5 VLAN Configurations

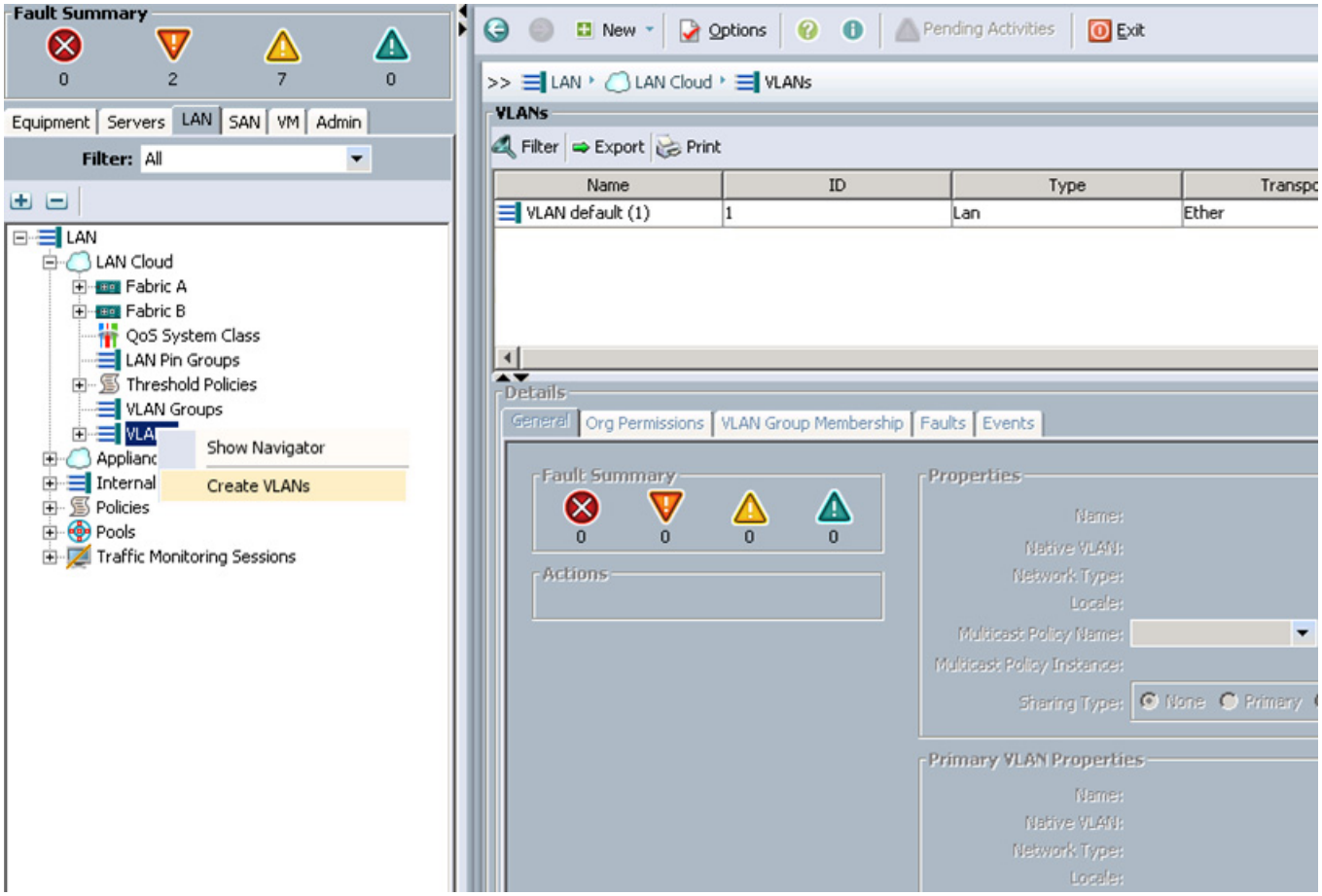
VLAN	Fabric	NIC Port	Function	Failover
vlan160_mgnt	A	eth0	Management, User connectivity	Fabric failover to B
vlan12_HDFS	B	eth1	Hadoop	Fabric failover to A

All of the VLANs created need to be trunked to the upstream distribution switch connecting the fabric interconnects. For this deployment vlan160_mgmt is configured for management access and user connectivity, vlan12_HDFS is configured for Hadoop interconnect traffic.

Follow these steps to configure VLANs in the Cisco UCS Manager:

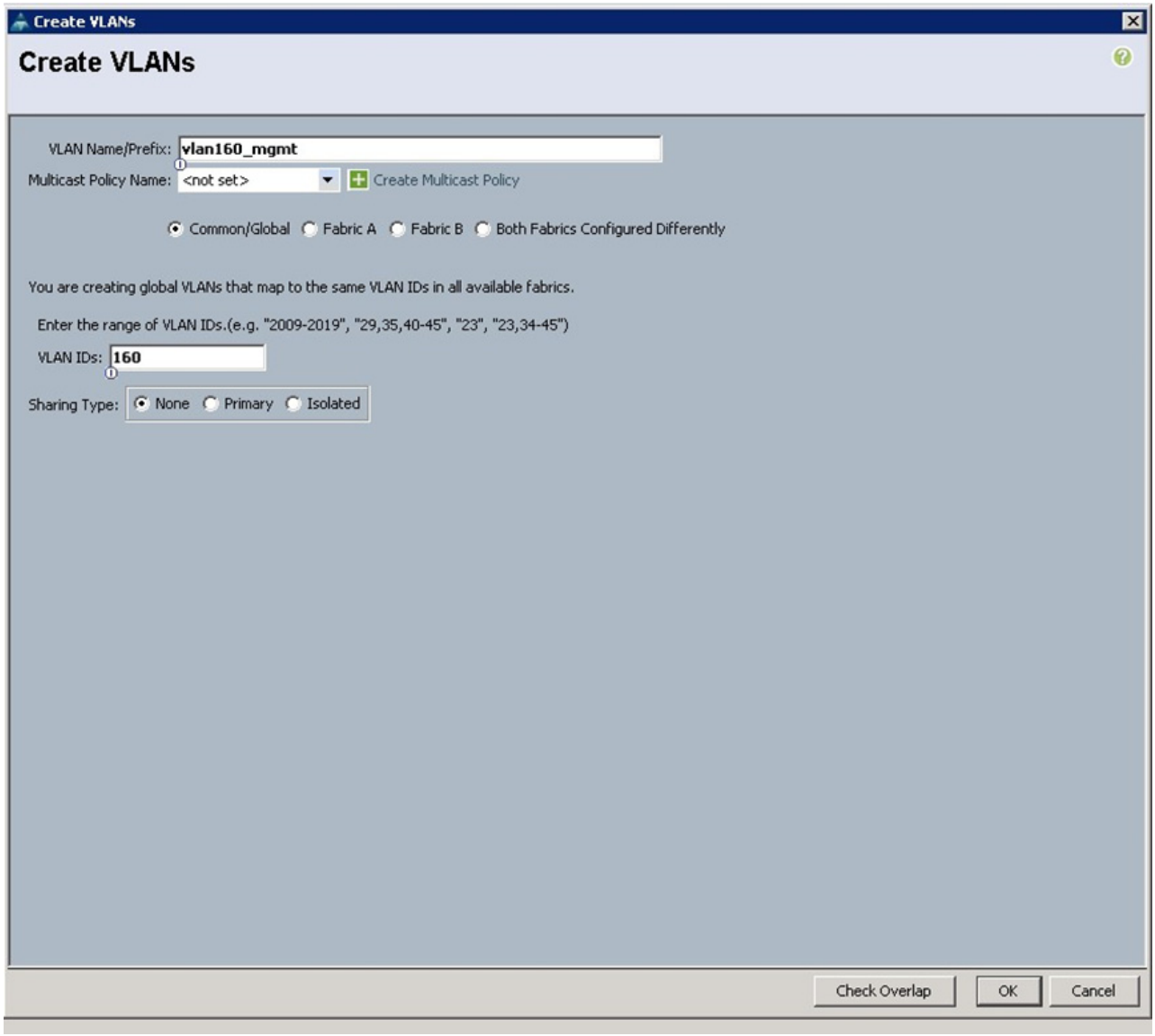
1. Select the **LAN** tab in the left pane in the UCS Manager GUI.
2. Select **LAN** > **VLANs**.
3. Right-click on the VLANs under the root organization.
4. Select **Create VLANs** to create the VLAN.

Figure 12 Creating VLAN



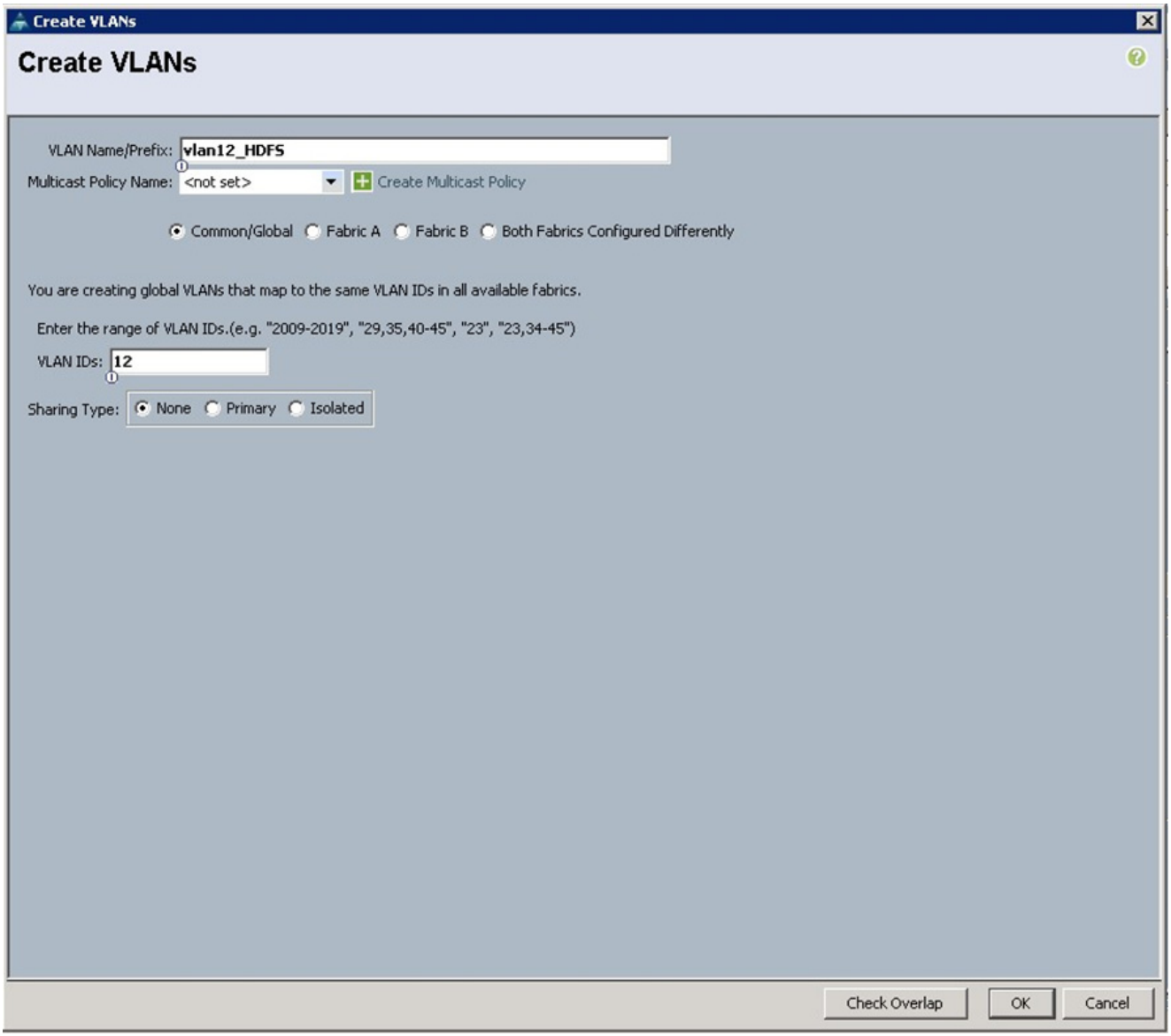
5. Enter vlan160_mgmt for the VLAN Name.
6. Select **Common/Global** for vlan160_mgmt.
7. Enter 160 on VLAN IDs of the Create VLAN IDs.
8. Click **OK** and then, click **Finish**.
9. Click **OK** in the success message box.

Figure 13 Creating Management VLAN



10. Select the **LAN** tab in the left pane again
11. Select **LAN > VLANs**.
12. Right-click on the VLANs under the root organization.
13. Select Create VLANs to create the VLAN.
14. Enter vlan12_HDFS for the VLAN Name.
15. Select **Common/Global** for the vlan12_HDFS.
16. Enter 12 on VLAN IDs of the Create VLAN IDs.
17. Click **OK** and then, click **Finish**.

Figure 14 Creating VLAN for Hadoop Data



Creating Server Pool

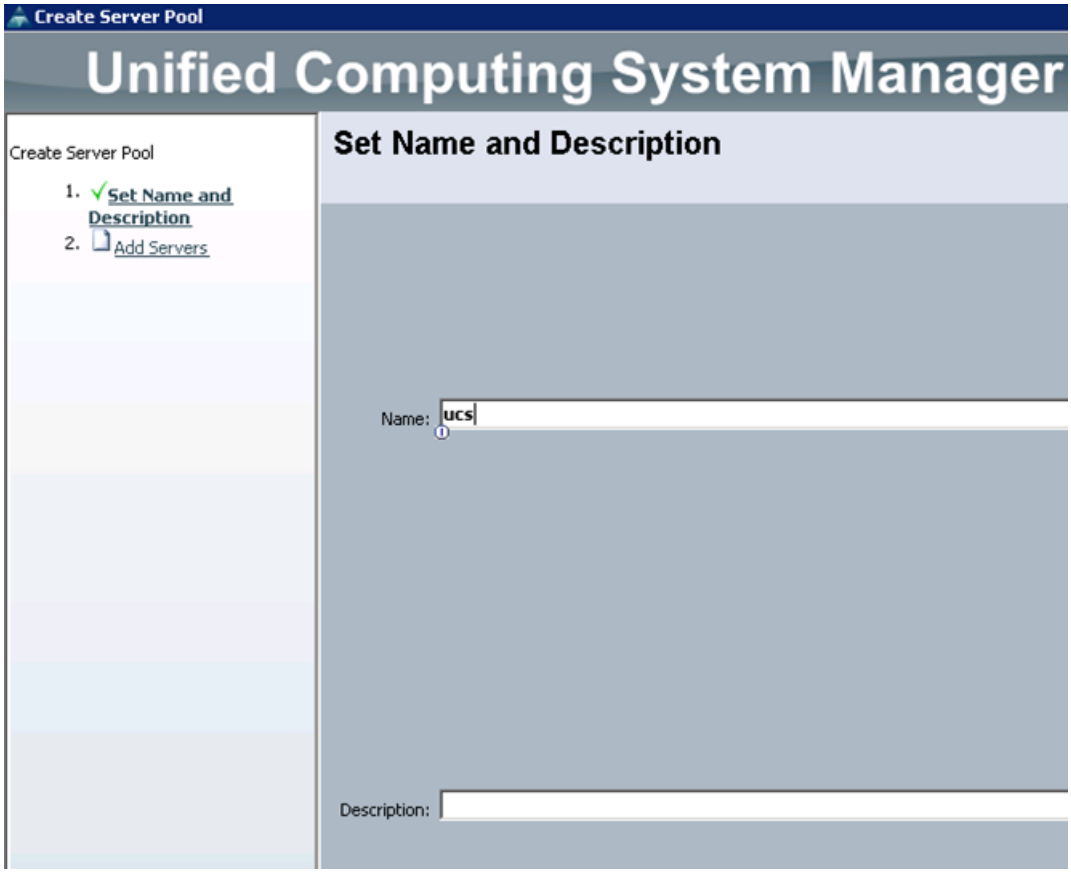
A server pool contains a set of servers. These servers typically share the same characteristics in terms of location in the chassis, server type, amount of memory, local storage, type of CPU, or local drive configuration. You can manually assign a server to a server pool, or use server pool policies and server pool policy qualifications to automate the assignment.

Follow these steps to configure the server pool within the Cisco UCS Manager:

1. Select the **Servers** tab in the left pane in the UCS Manager GUI.
2. Select **Pools** > **root**.
3. Right-click on the Server Pools.

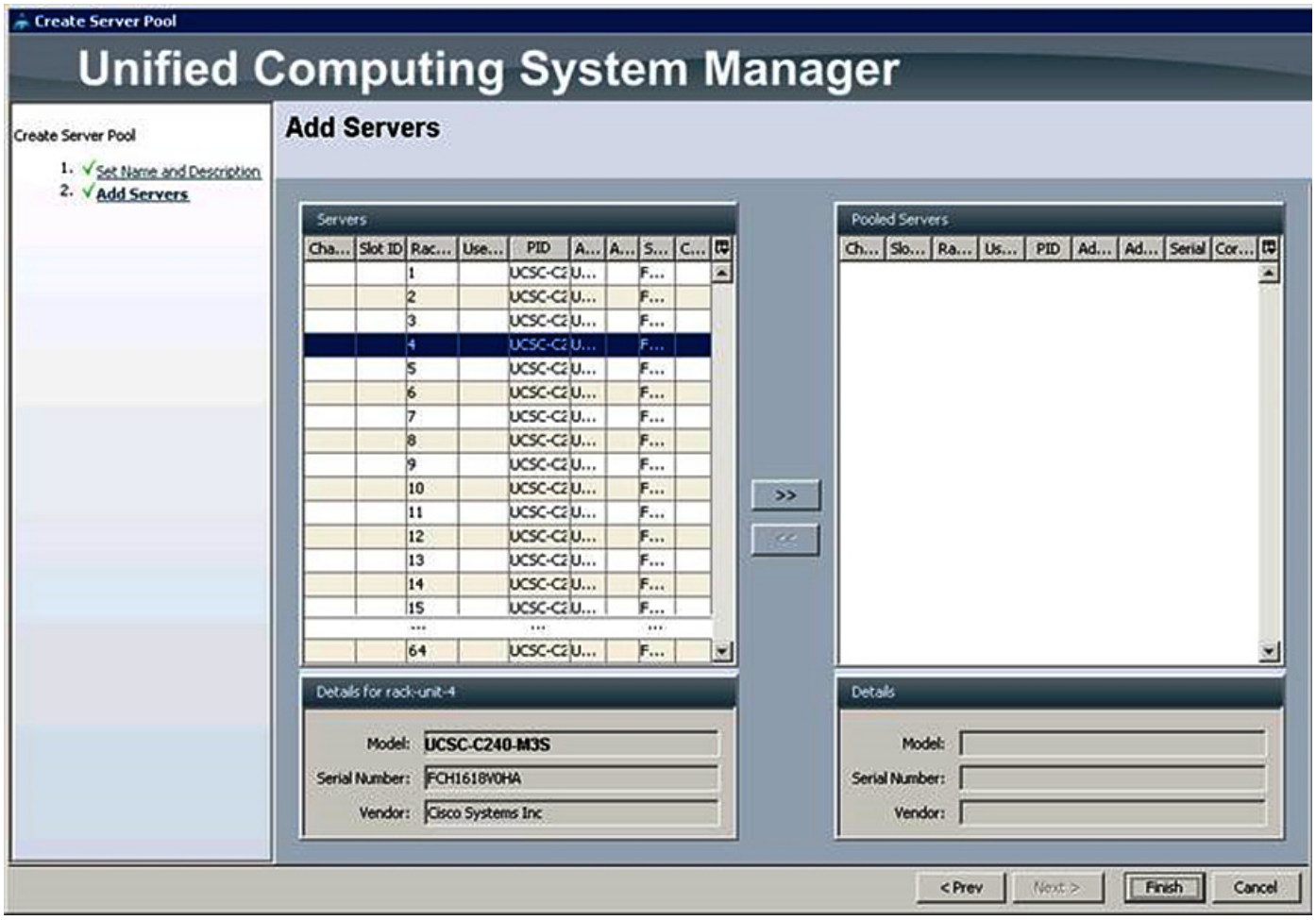
4. Select **Create Server Pool**.
5. Enter your required name (ucs) for the Server Pool in the name text box.
6. (Optional) enter a description for the organization
7. Click **Next** to add the servers.

Figure 15 Setting Name and Description of the Server Pool



8. Select all the Cisco UCS C240 M3 servers to be added to the server pool that you have previously created (ucs), then click **>>** to add them to the pool.
9. Click **Finish**.
10. Click **OK** and then click **Finish**.

Figure 16 Adding Servers to the Server Pool



Creating Policies for Service Profile Templates

Creating Host Firmware Package Policy

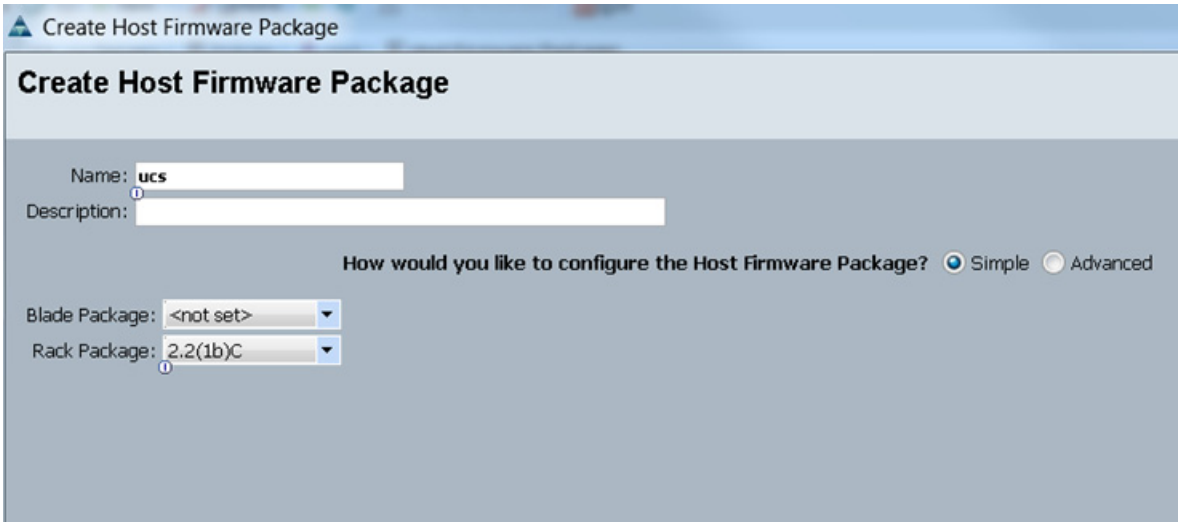
Firmware management policies allow the administrator to select the corresponding packages for a given server configuration. These include adapters, BIOS, board controllers, FC adapters, HBA options, ROM and storage controller properties as applicable.

Follow these steps to create a firmware management policy for a given server configuration using the Cisco UCS Manager:

1. Select the **Servers** tab in the left pane in the UCS Manager GUI.
2. Select **Policies > root**.
3. Right-click on the Host Firmware Packages.
4. Select **Create Host Firmware Package**.

5. Enter your required Host Firmware package name (ucs).
6. Select **Simple** radio button to configure the Host Firmware package.
7. Select the appropriate Rack package that you have.
8. Click **OK** to complete creating the management firmware package.
9. Click **OK**.

Figure 17 Creating Host Firmware Package



Creating QoS Policies

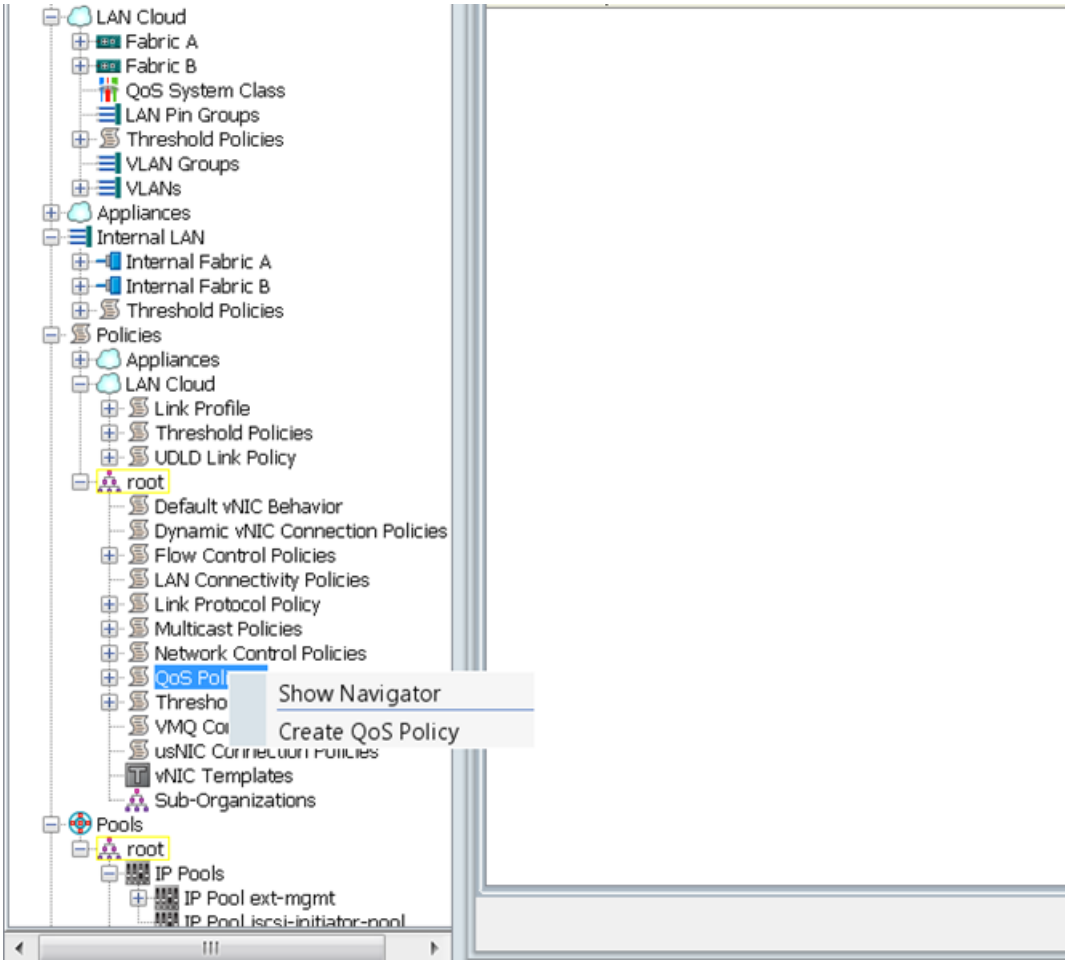
Follow these steps to create the QoS policy for a given server configuration using the Cisco UCS Manager:

Best Effort Policy

Follow these steps for setting the best effort policy:

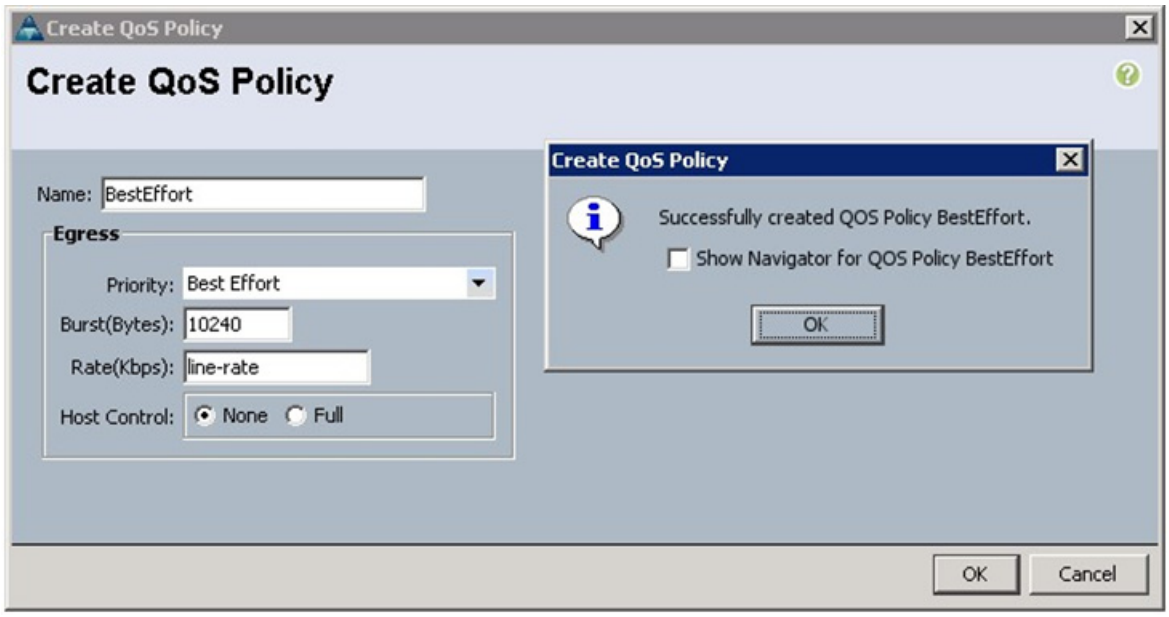
1. Select the **LAN** tab in the left pane in the UCS Manager GUI.
2. Select **Policies** > **root**.
3. Right-click on the QoS Policies.
4. Select **Create QoS Policy**.

Figure 18 Creating QoS Policy



5. Enter BestEffort as the name of the policy.
6. Select **BestEffort** from the drop down menu.
7. Keep the Burst (Bytes) field as default (10240).
8. Keep the Rate (Kbps) field as default (line-rate).
9. Keep Host Control radio button as default (none).
10. Once the pop-up window appears, click **OK** to complete the creation of the policy.

Figure 19 Creating BestEffort QoS Policy

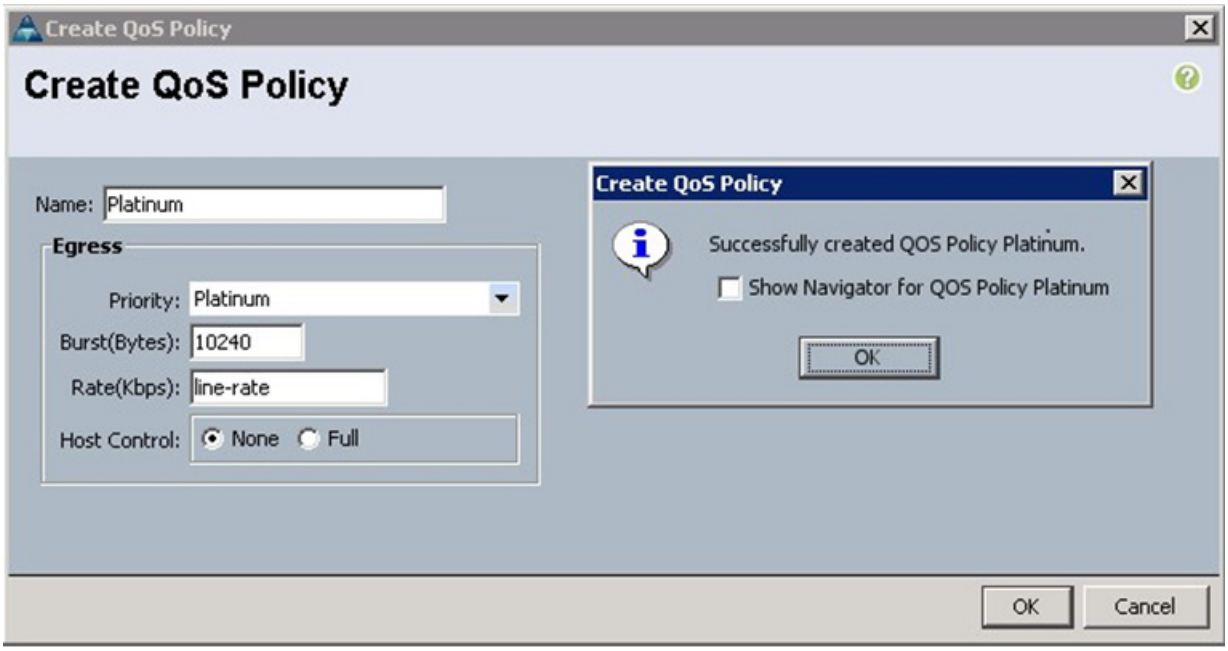


Platinum Policy

Follow these steps for setting platinum policy:

1. Select the **LAN** tab in the left pane in the UCS Manager GUI.
2. Select **Policies** > **root**.
3. Right-click on the QoS Policies.
4. Select **Create QoS Policy**.
5. Enter Platinum as the name of the policy.
6. Select Platinum from the drop down menu.
7. Keep the Burst (Bytes) field as default (10240).
8. Keep the Rate (Kbps) field as default (line-rate).
9. Keep Host Control radio button as default (none).
10. Once the pop-up window appears, click **OK** to complete the creation of the policy.

Figure 20 Creating Platinum QoS Policy

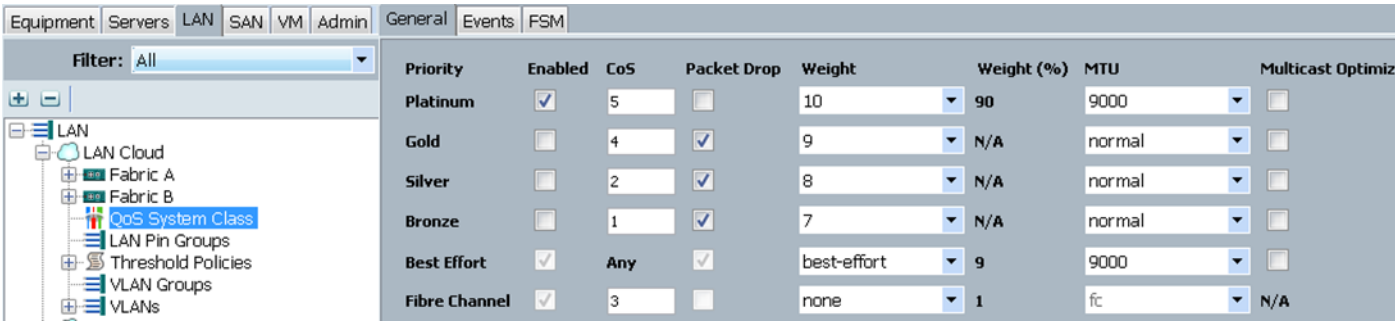


Setting Jumbo Frames

Follow these steps for setting Jumbo frames and enabling QoS:

1. Select the **LAN** tab in the left pane in the UCS Manager GUI.
2. Select **LAN Cloud** > **QoS System Class**.
3. In the right pane, select the **General** tab
4. In the Platinum row, enter 9000 for MTU.
5. Check the Enabled Check box next to Platinum.
6. In the Best Effort row, select **best-effort** for weight and **9000** for MTU.
7. In the Fiber Channel row, select **none** for weight.
8. Click **Save Changes**.
9. Click **OK**.

Figure 21 Setting Jumbo Frames



Creating Local Disk Configuration Policy

Follow these steps to create local disk configuration in the Cisco UCS Manager:

1. Select the **Servers** tab on the left pane in the UCS Manager GUI.
2. Go to **Policies** > **root**.
3. Right-click on the Local Disk Configuration Policies.
4. Select **Create Local Disk Configuration Policy**.
5. Enter ucs as the local disk configuration policy name.
6. Change the Mode to Any Configuration. Uncheck the **Protect Configuration** check box.
7. Keep the FlexFlash State field as default (Disable).
8. Keep the FlexFlash RAID Reporting State field as default (Disable).
9. Click **OK** to complete the creation of the Local Disk Configuration Policy.
10. Click **OK**.

Figure 22 Configuring Local Disk Policy

Creating Server BIOS Policy

The BIOS policy feature in Cisco UCS automates the BIOS configuration process. The traditional method of setting the BIOS is done manually and is often error-prone. By creating a BIOS policy and assigning the policy to a server or group of servers, you can enable transparency within the BIOS settings configuration.

BIOS settings can have a significant performance impact, depending on the workload and the applications. The BIOS settings listed in this section is for configurations optimized for best performance which can be adjusted based on the application, performance and energy efficiency requirements.

Follow these steps to create a server BIOS policy using the Cisco UCS Manager:

1. Select the **Servers** tab in the left pane in the UCS Manager GUI.
2. Select **Policies** > **root**.
3. Right-click on the BIOS Policies.
4. Select **Create BIOS Policy**.
5. Enter your preferred BIOS policy name (ucs).
6. Change the BIOS settings as per the Figure 23, Figure 24, Figure 25.

Figure 23 Creating Server BIOS Policy

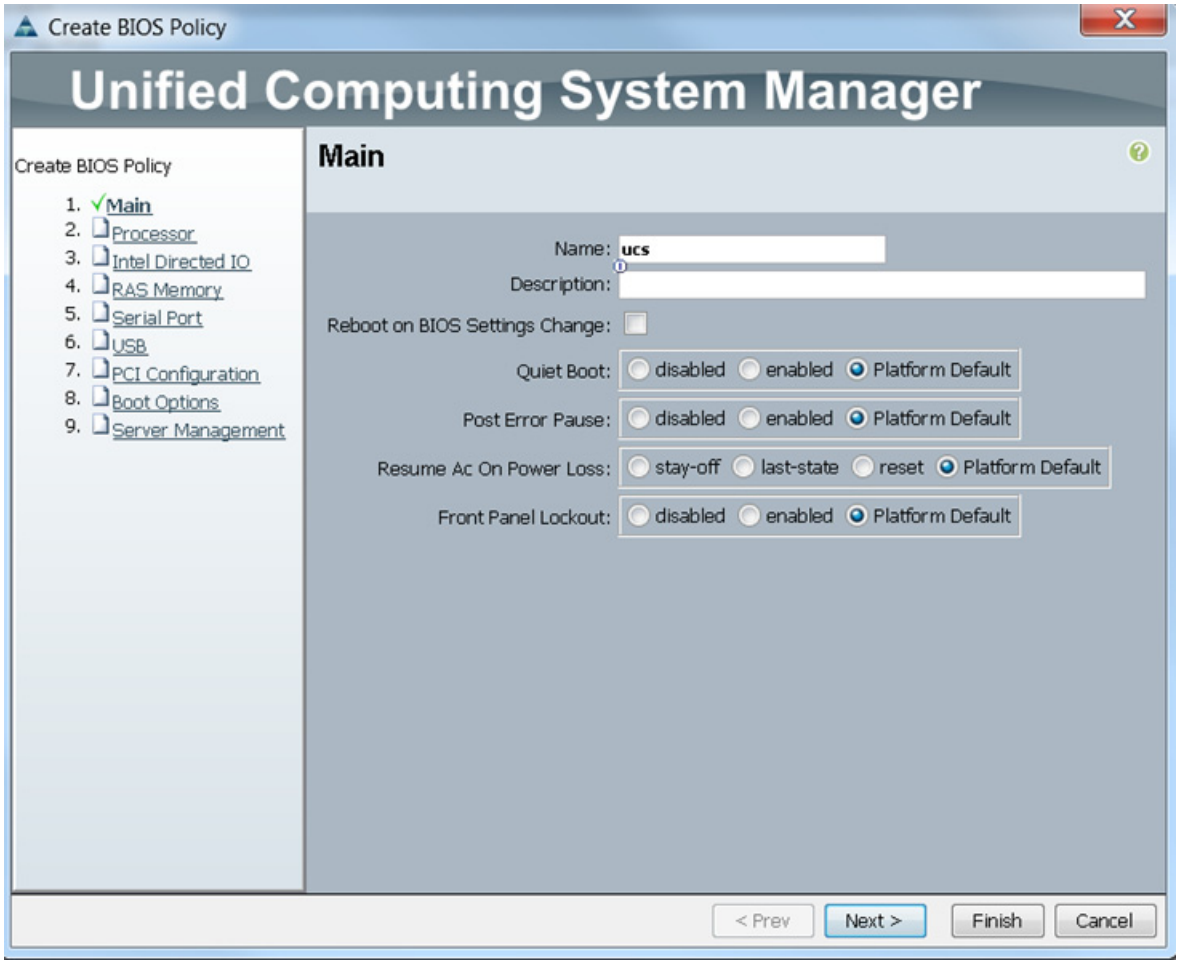


Figure 24 Creating Server BIOS Policy for Processor

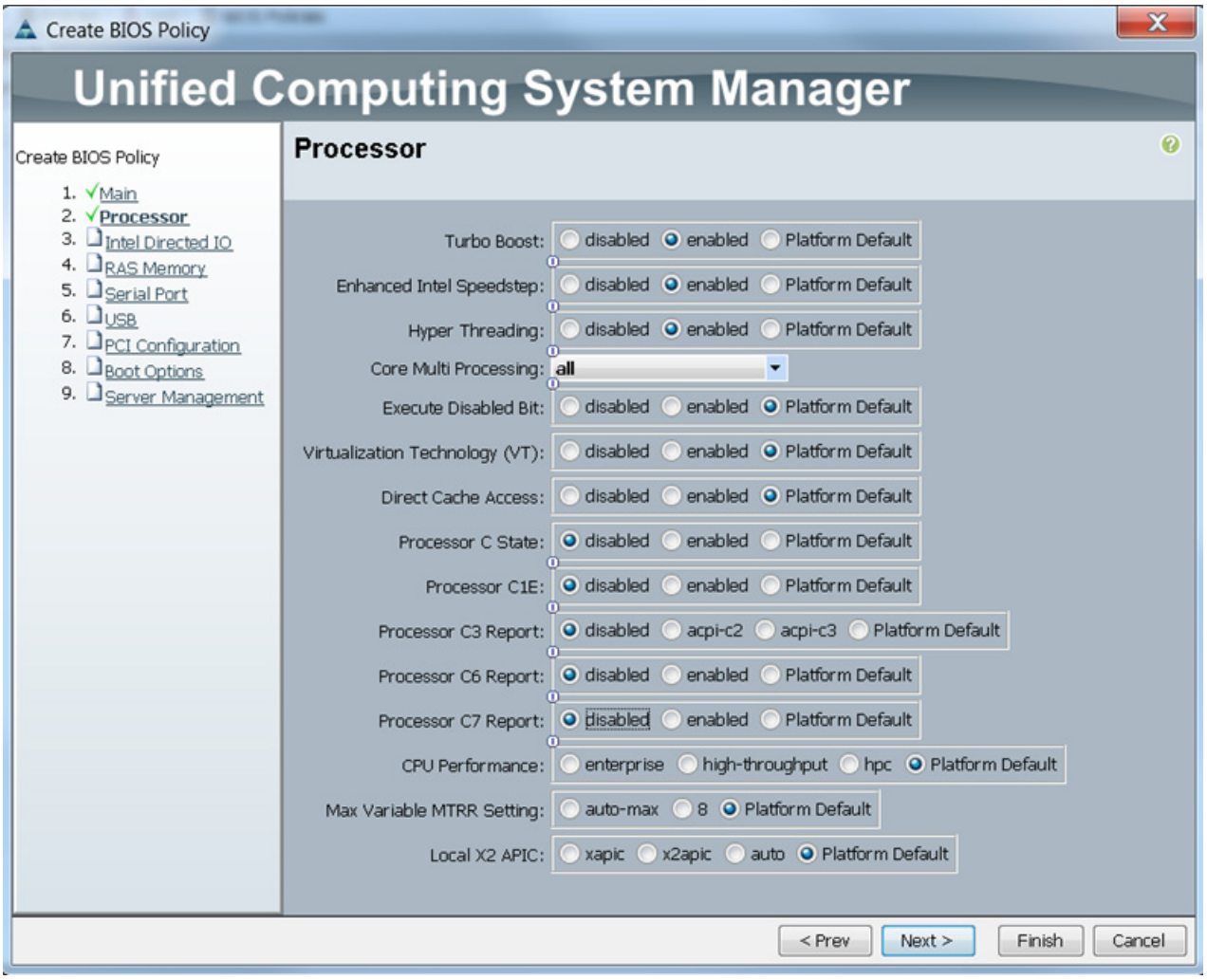
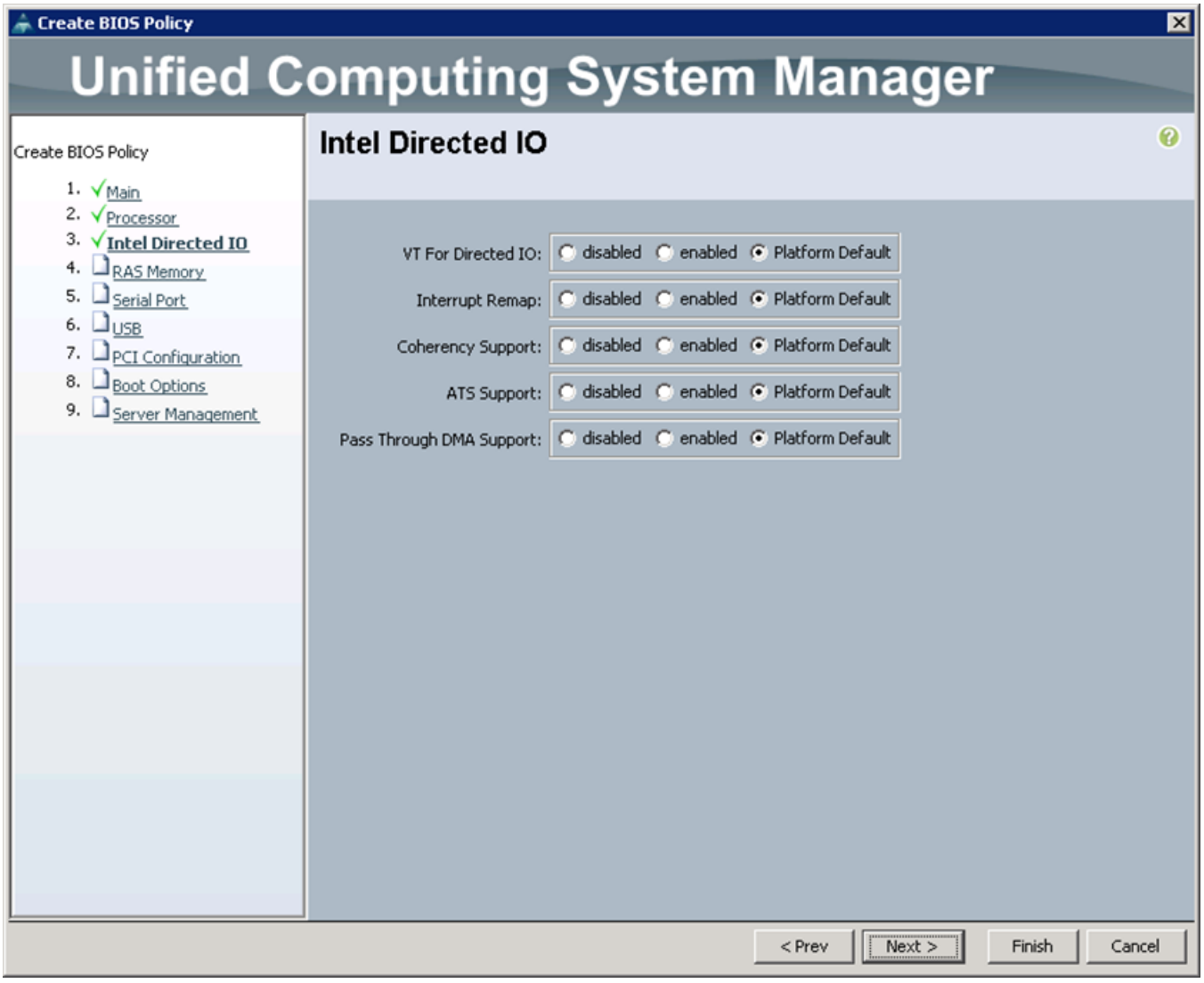


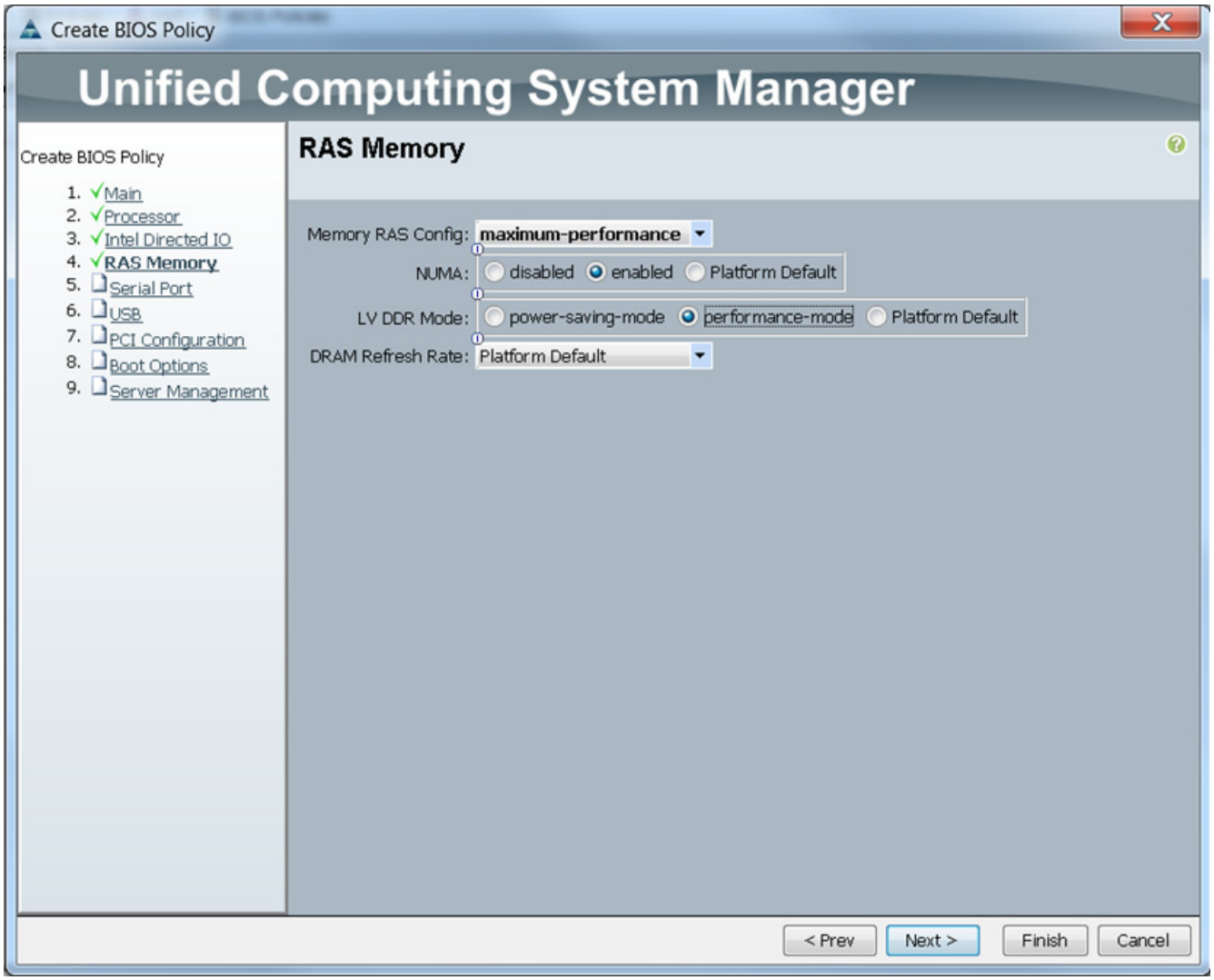
Figure 25 Creating Server BIOS Policy for Intel Directed IO



Click **Finish** to complete creating the BIOS policy.

Click **OK**.

Figure 26 Creating Server BIOS Policy for Memory

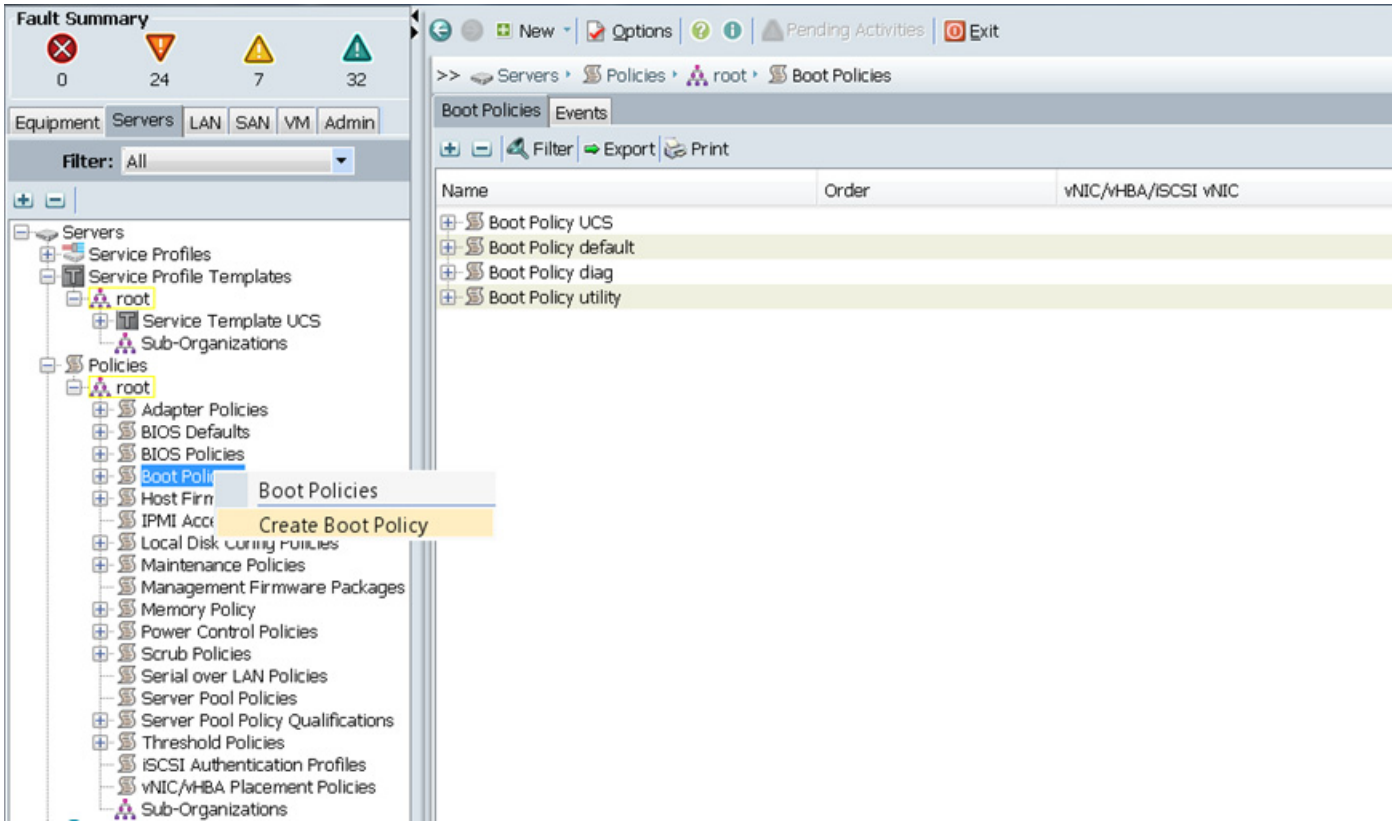


Creating Boot Policy

Follow these steps to create boot policies within the Cisco UCS Manager:

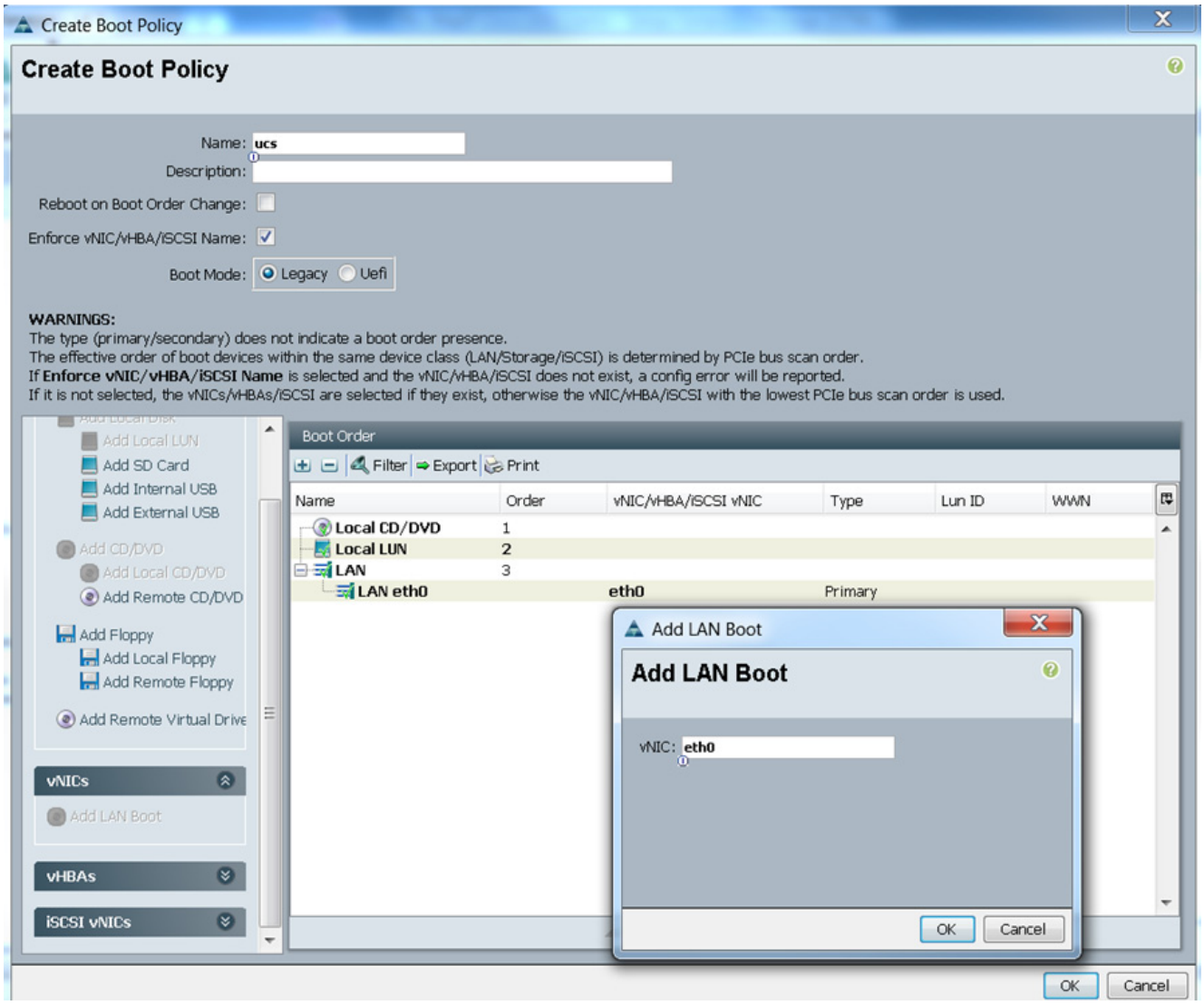
1. Select the **Servers** tab in the left pane in the UCS Manager GUI.
2. Select **Policies** > **root**.
3. Right-click on the Boot Policies.
4. Select **Create Boot Policy**.

Figure 27 Creating Boot Policy - Part 1



5. Enter ucs as the boot policy name.
6. (Optional) enter a description for the boot policy.
7. Keep the Reboot on **Boot Order Change** check box unchecked.
8. Keep **Enforce vNIC/vHBA/iSCSI Name** check box checked.
9. Keep Boot Mode Default (Legacy).
10. Expand Local Devices > Add CD/DVD and select **Add Local CD/DVD**.
11. Expand Local Devices > Add Local Disk and select **Add Local LUN**.
12. Expand vNICs and select **Add LAN Boot** and enter eth0.
13. Click **OK** to add the Boot Policy.
14. Click **OK**.

Figure 28 Creating Boot Policy - Part 2

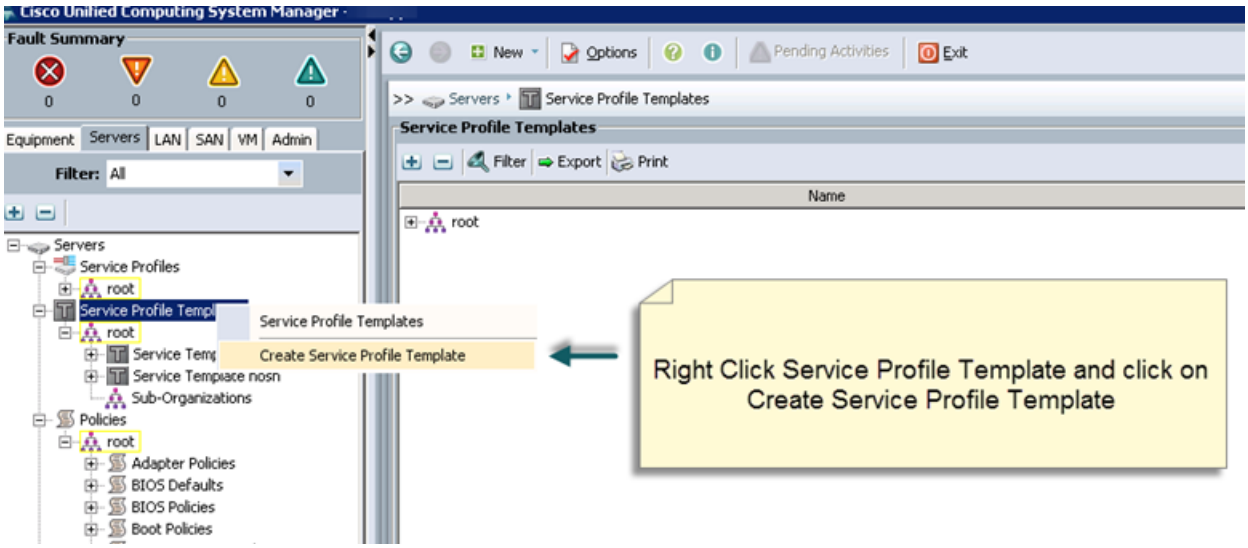


Creating Service Profile Template

To create a service profile template, follow these steps:

1. Select the **Servers** tab in the left pane in the UCSM GUI.
2. Right-click on the Service Profile Templates.
3. Select **Create Service Profile Template**.

Figure 29 Creating Service Profile Template



4. The Create Service Profile Template window appears.

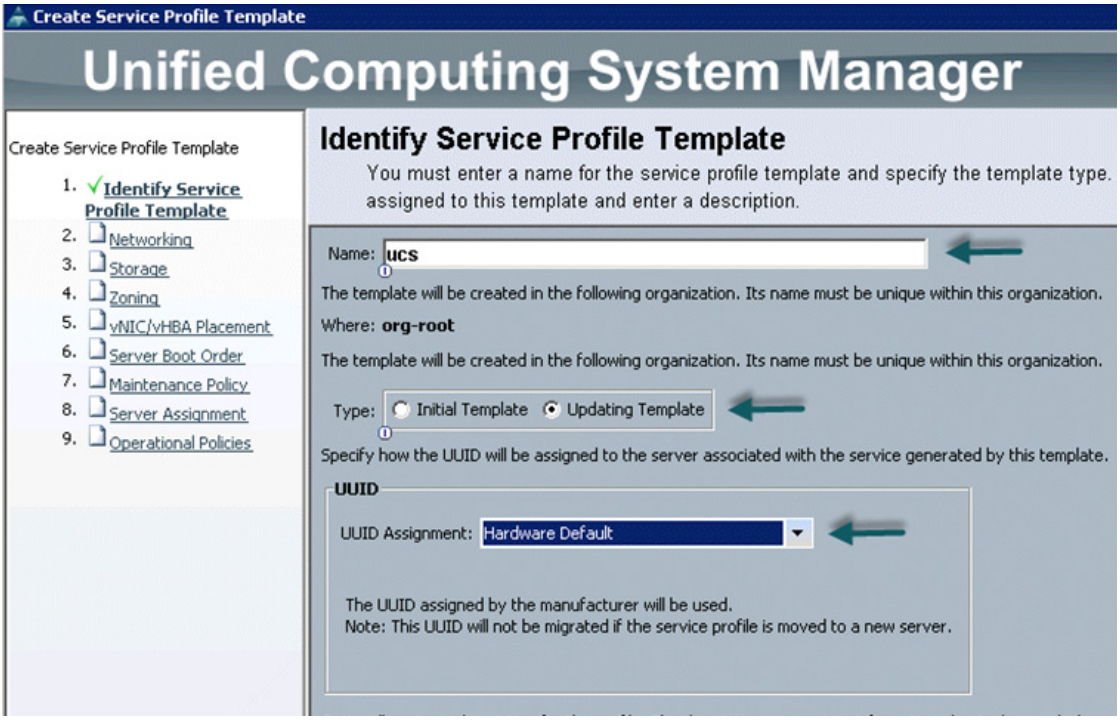
   The following steps provide a detailed configuration procedure to identify the service profile template:

   a. Name the service profile template as ucs. Select the **Updating Template** radio button.

   b. In the UUID section, select **Hardware Default** as the UUID pool.

   c. Click **Next** to continue to the next section.

Figure 30 Creating Service Profile Template - Identify

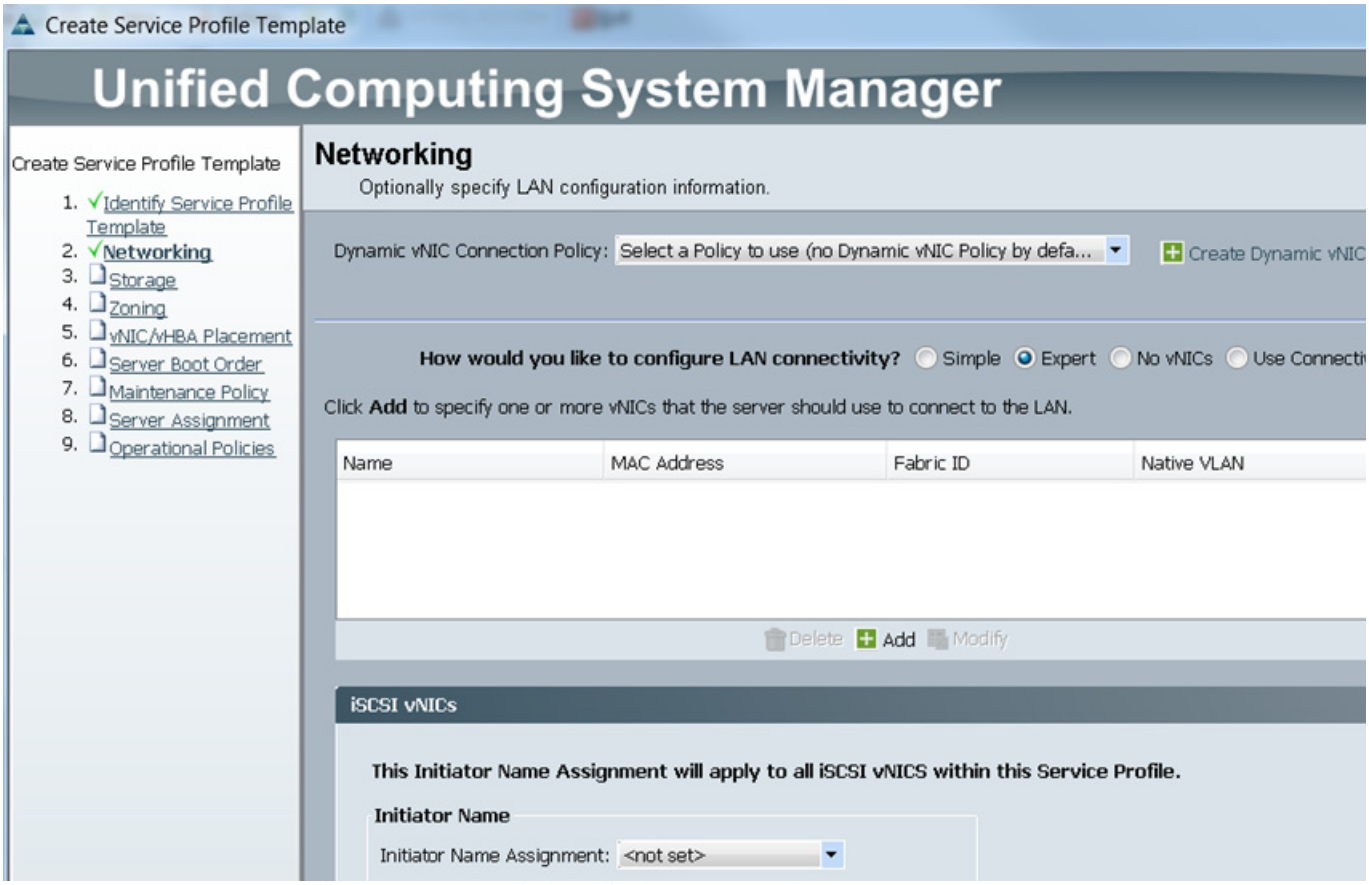


Configuring Network Settings for the Template

For network setting, follow these steps:

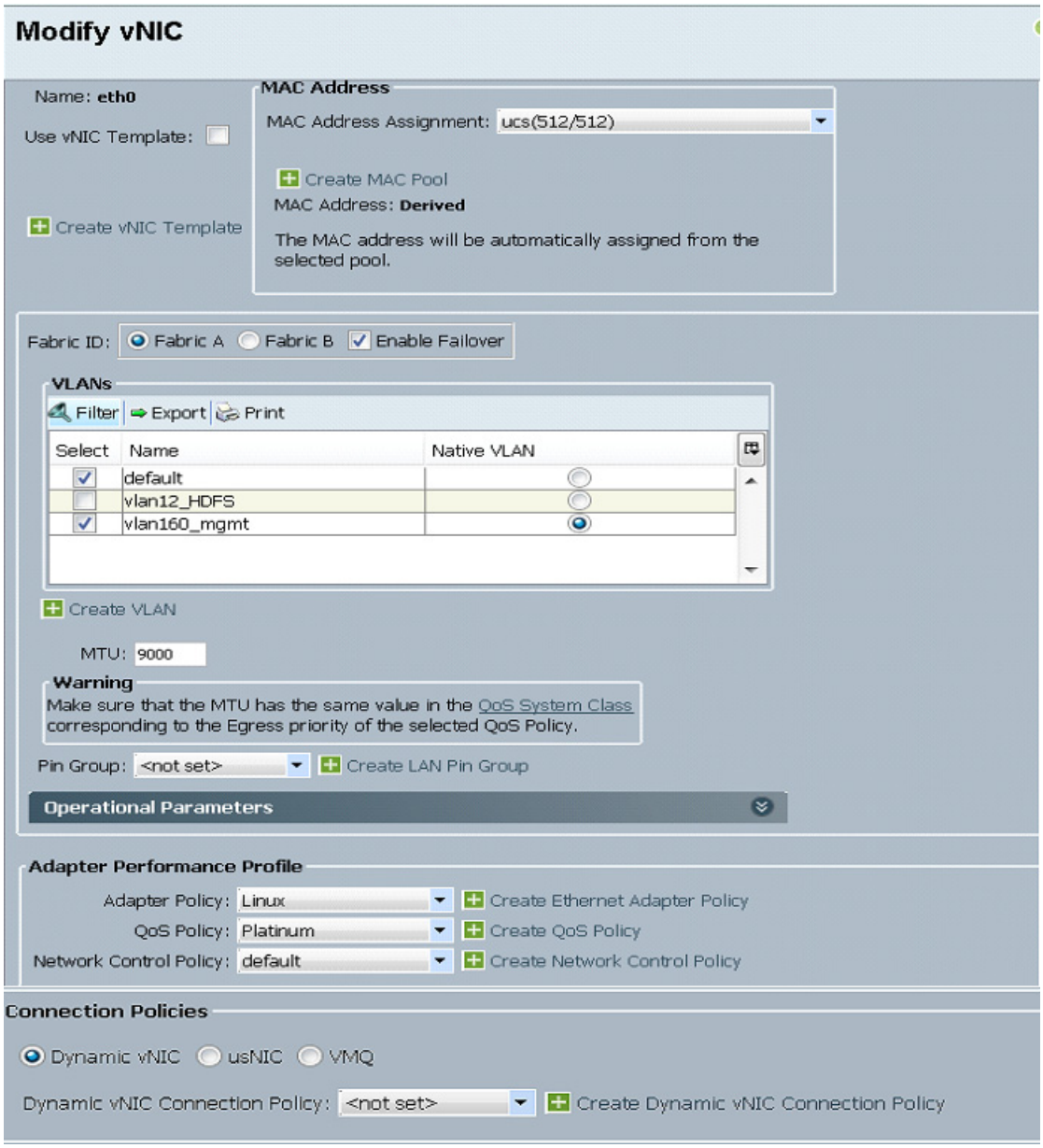
1. Keep the Dynamic vNIC Connection Policy field at the default.
2. Select **Expert** radio button for the option how would you like to configure LAN connectivity?
3. Click **Add** to add a vNIC to the template.

Figure 31 Creating Service Profile Template - Networking



4. The Create vNIC window displays. Name the vNIC as eth0.
5. Select **ucs** in the Mac Address Assignment pool.
6. Select the **Fabric A** radio button and check the **Enable Failover** check box for the Fabric ID.
7. Check the **vlan160_mgmt** check box for VLANs and select the **Native VLAN** radio button.
8. Select MTU size as **9000**.
9. Select adapter policy as **Linux**.
10. Select QoS Policy as **BestEffort**.
11. Keep the Network Control Policy as **Default**.
12. Keep the Connection Policies as **Dynamic vNIC**.
13. Keep the Dynamic vNIC Connection Policy as **<not set>**.
14. Click **OK**.

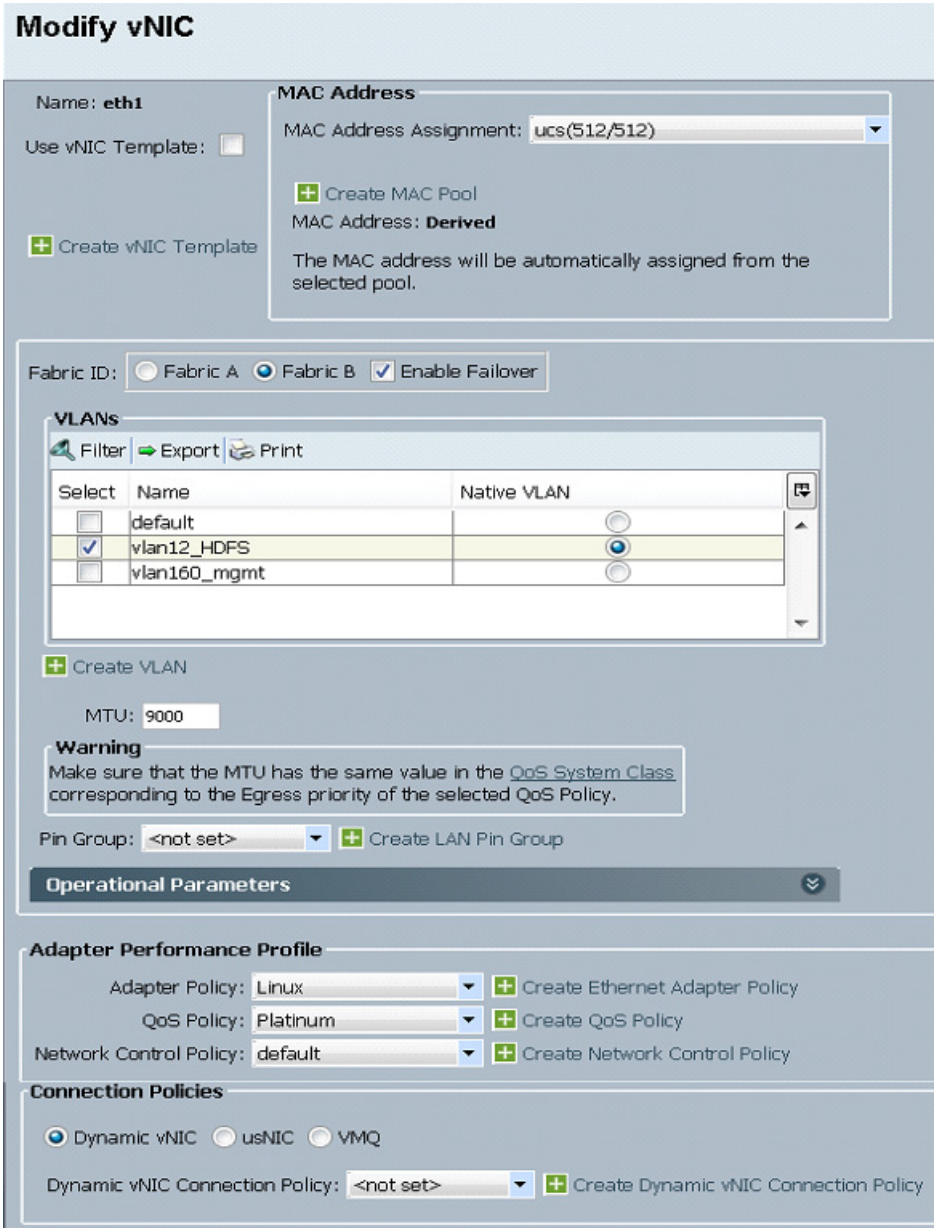
Figure 32 Configuring vNIC eth0



15. The Create vNIC window appears. Name the vNIC as eth1.
16. Select **ucs** in the Mac Address Assignment pool.
17. Select **Fabric B** radio button and check the **Enable Failover** check box for the Fabric ID.
18. Check the **vlan12_HDFS** check box for VLANs and select the **Native VLAN** radio button
19. Select MTU size as **9000**.
20. Select adapter policy as **Linux**.
21. Select QoS Policy as **Platinum**.
22. Keep the Network Control Policy as **Default**.
23. Keep the Connection Policies as **Dynamic vNIC**.

24. Keep the Dynamic vNIC Connection Policy as **<not set>**.
25. Click **OK**.

Figure 33 Configuring vNIC eth1



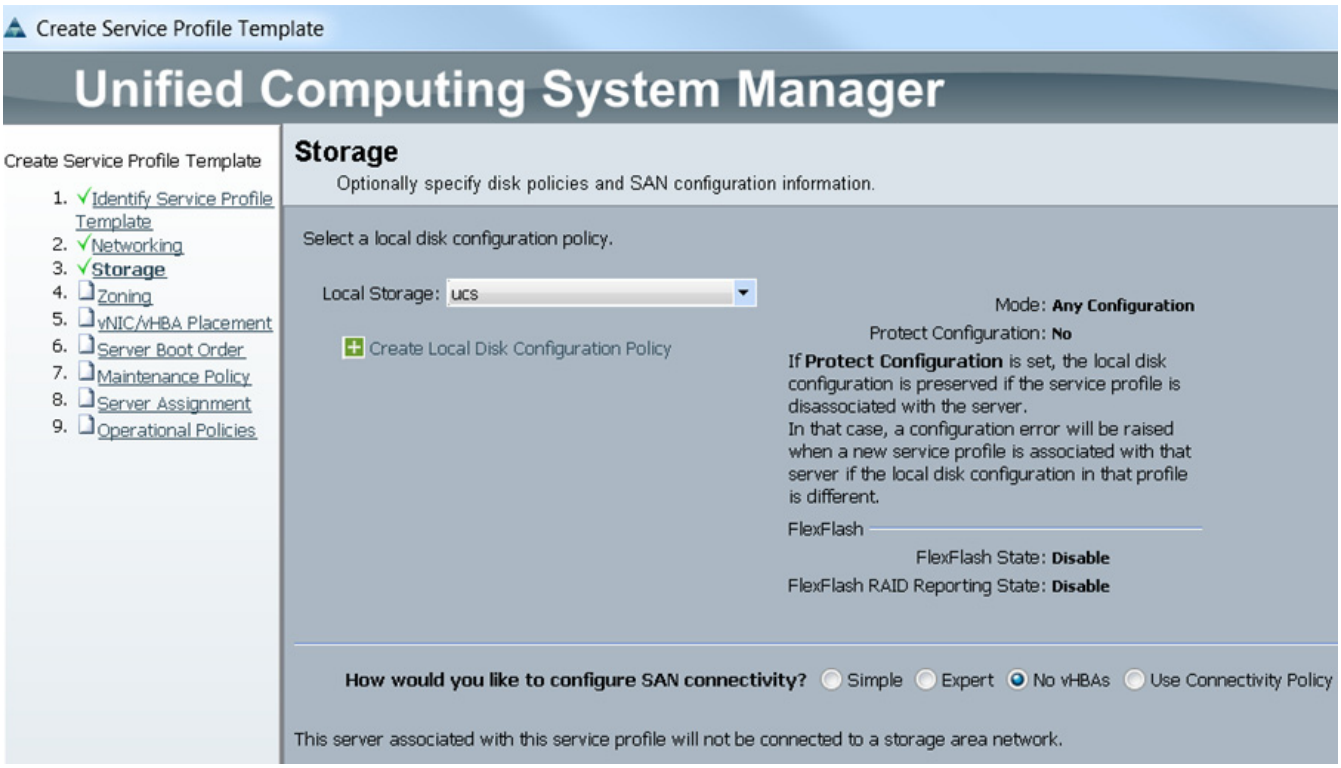
Configuring Storage Policy for the Template

For configuring storage policy, follow these steps:

1. Select **ucs** for the local disk configuration policy.
2. Select the **No vHBAs** radio button for the option for How would you like to configure SAN connectivity?

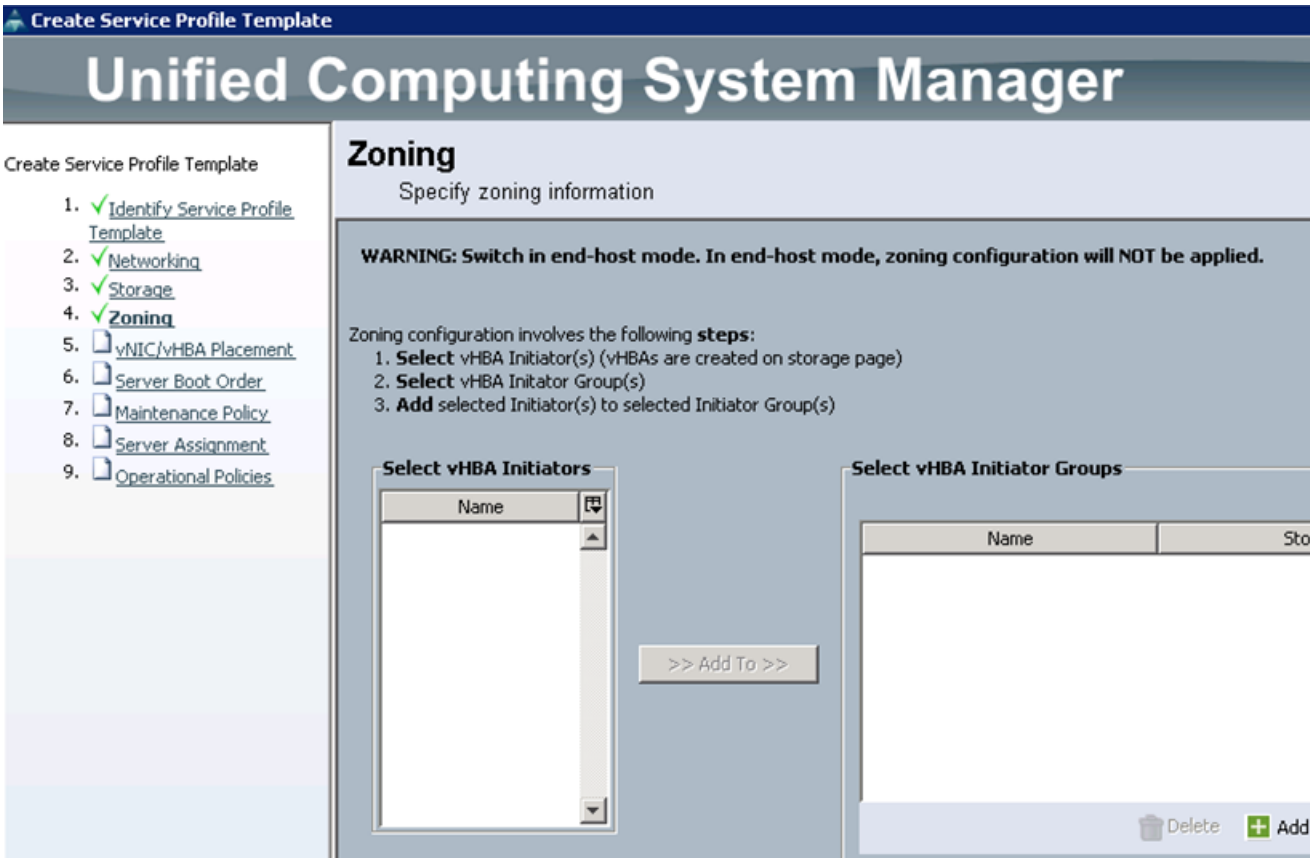
3. Click **Next** to continue to the next section.

Figure 34 Creating Service Profile Template - Storage



4. Click **Next** to continue. Click **Next**, in the zoning window to go to the next section.

Figure 35 Creating Service Profile Template - Zoning

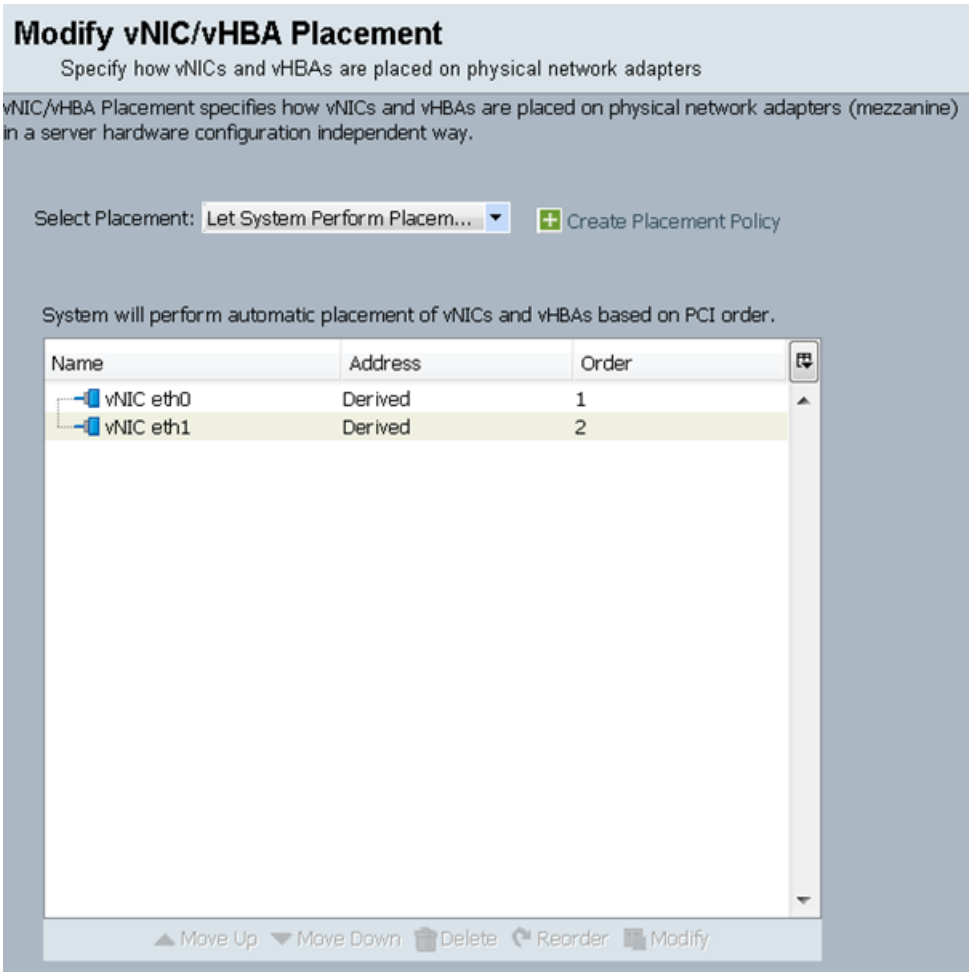


Configuring vNIC/vHBA Placement for the Template

For configuring vNIC/vHBA placement policy, follow these steps:

1. Select the **Default Placement Policy** option for the Select Placement field.
2. Select eth0 and eth1 assign the vNICs in the following order:
   a. eth0
   b. eth1
3. Review to make sure that all of the vNICs were assigned in the appropriate order.
4. Click **Next** to continue to the next section.

Figure 36 Creating Service Profile Template - vNIC/vHBA Placement

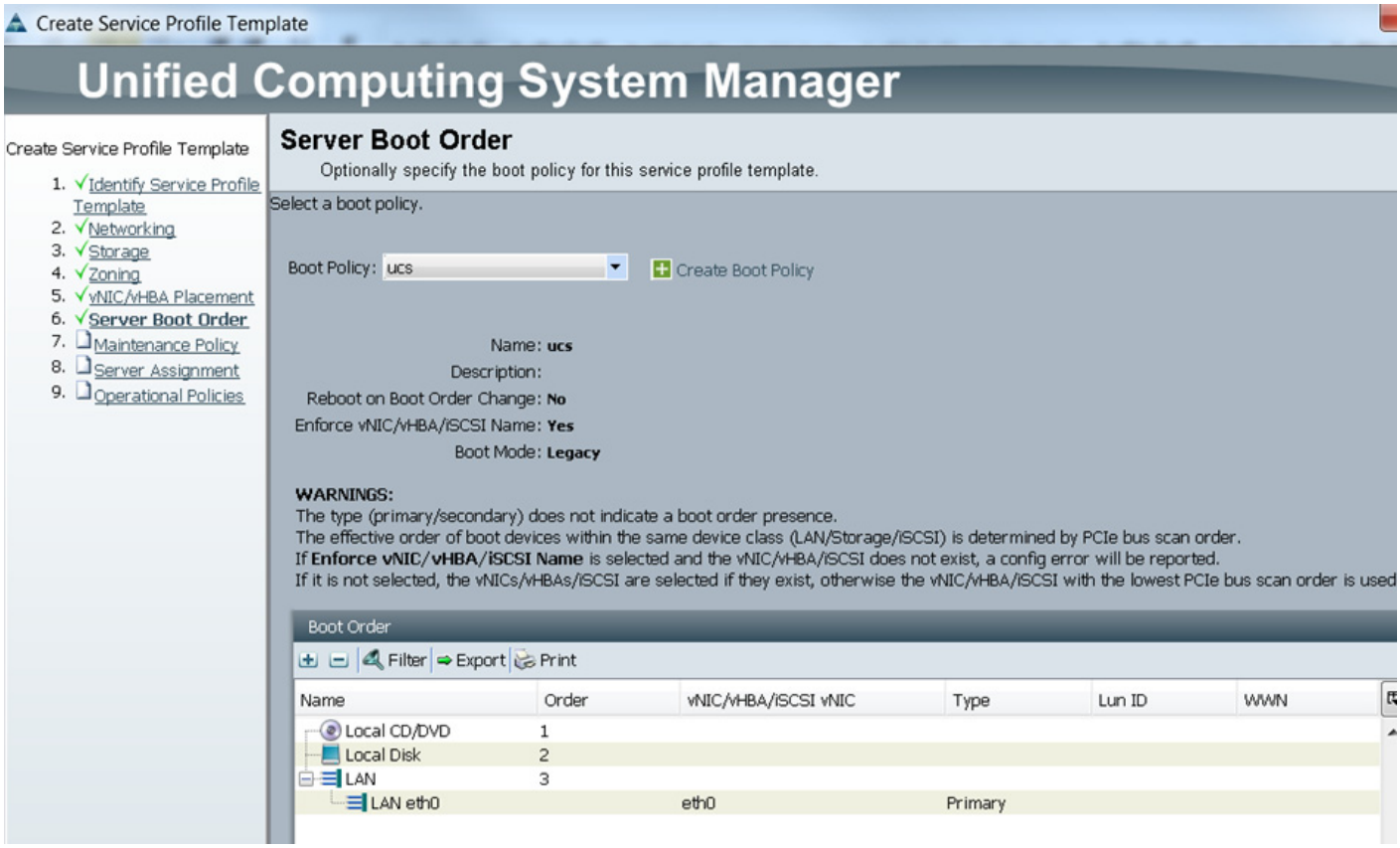


Configuring Server Boot Order for the Template

For setting the server boot order, follow these steps:

1. Select **ucs** in the Boot Policy name field.
2. Check the **Enforce vNIC/vHBA/iSCSI Name** check box.
3. Review to make sure that all of the boot devices were created and identified.
4. Verify that the boot devices are in the correct boot sequence.
5. Click **OK**.
6. Click **Next** to continue to the next section.

Figure 37 Creating Service Profile Template - Server Boot Order



For applying the maintenance policy, follow these steps:

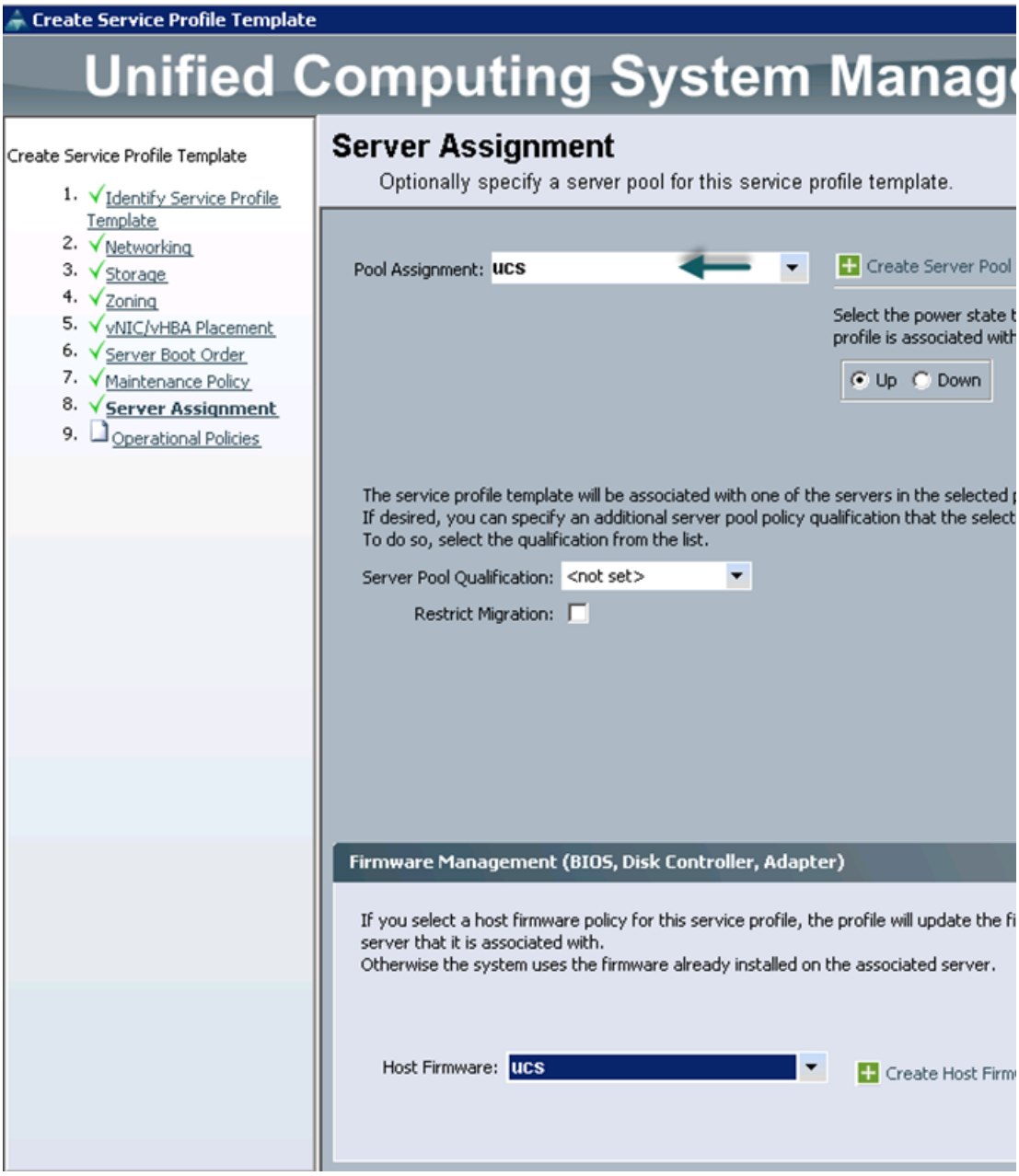
1. Keep the Maintenance policy at **no policy used** by default.
2. Click **Next** to continue to the next section.

Configuring Server Assignment for the Template

For assigning servers to the pool, follow these steps:

1. Select **ucs** for the Pool Assignment field.
2. Keep the Server Pool Qualification field at default.
3. Select **ucs** in Host Firmware Package **.**

Figure 38 Creating Service Profile Template - Server Assignment

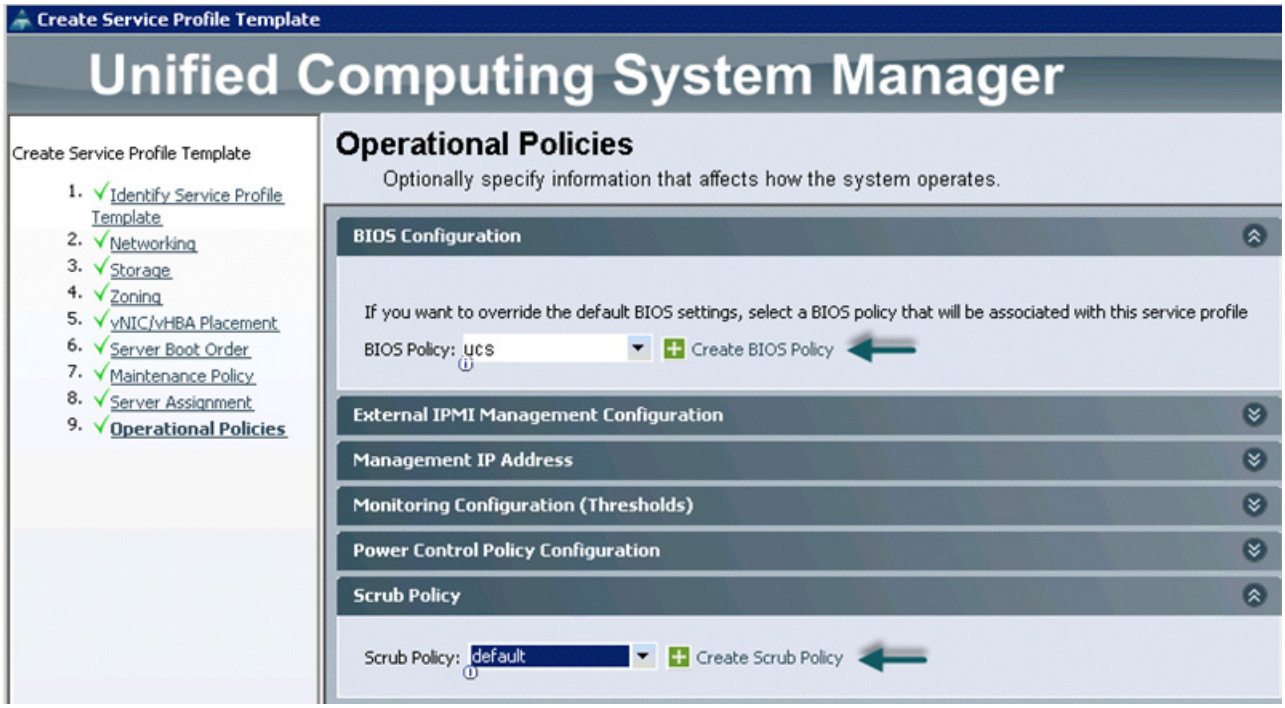


Configuring Operational Policies for the Template

For configuring operational Policies, follow these steps:

1. Select **ucs** in the BIOS Policy field.
2. Keep the Scrub Policy as default.
3. Click **Finish** to create the Service Profile template.
4. Click **OK** in the pop-up window to proceed.

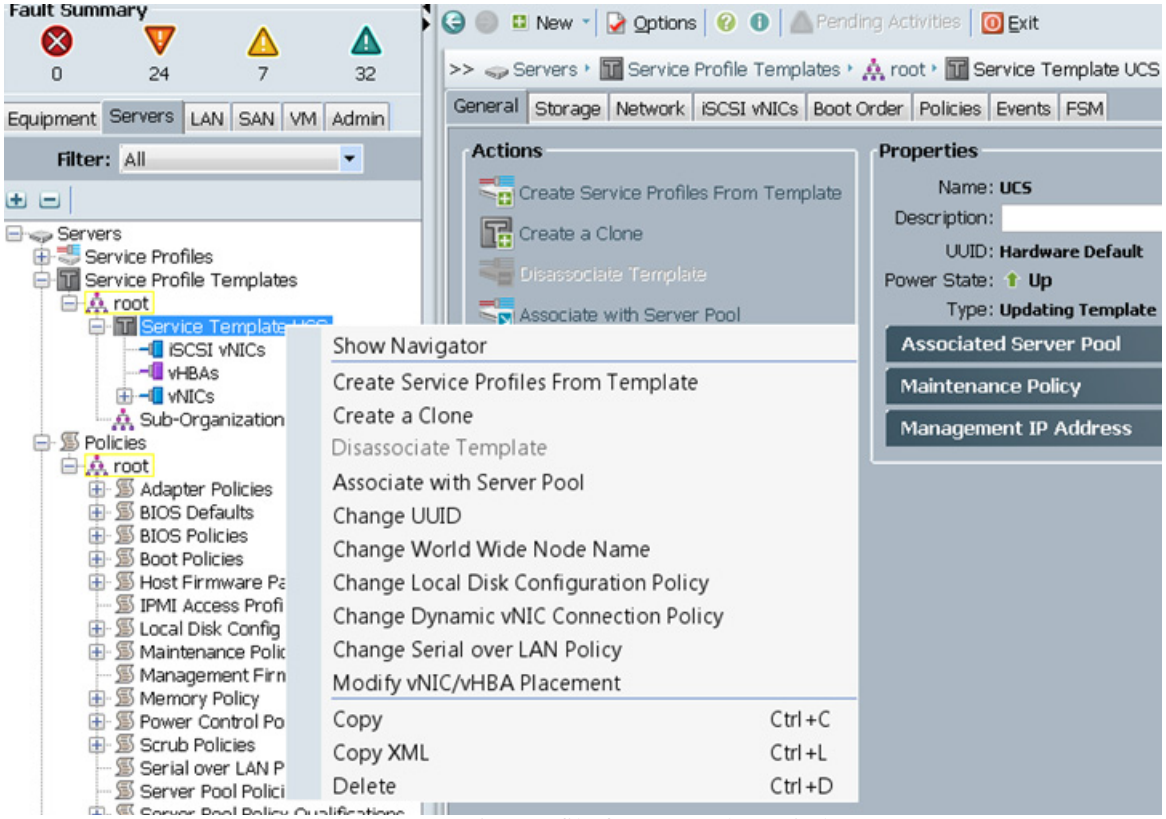
Figure 39 Creating Service Profile Template - BIOS Configuration for Operational Policies



Creating Service Profiles from Templates

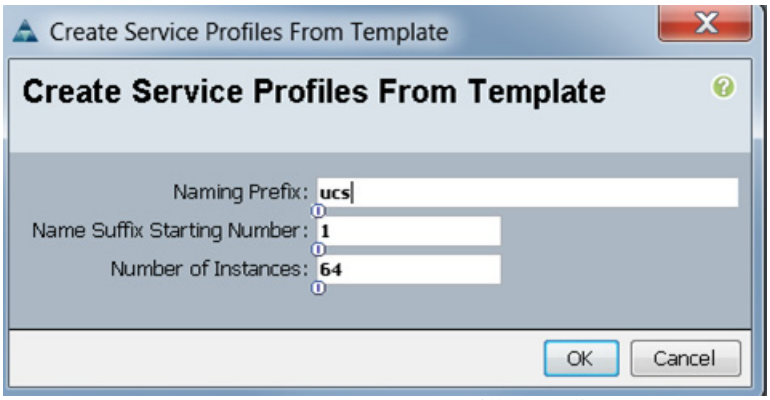
1. Select the **Servers** tab in the left pane of the UCS Manager GUI.
2. Go to **Service Profile Templates > root .**
3. Right-click on the **Service Profile Templates**.
4. Select **Create Service Profiles From Template**.

Figure 40 Creating Service Profiles from Template



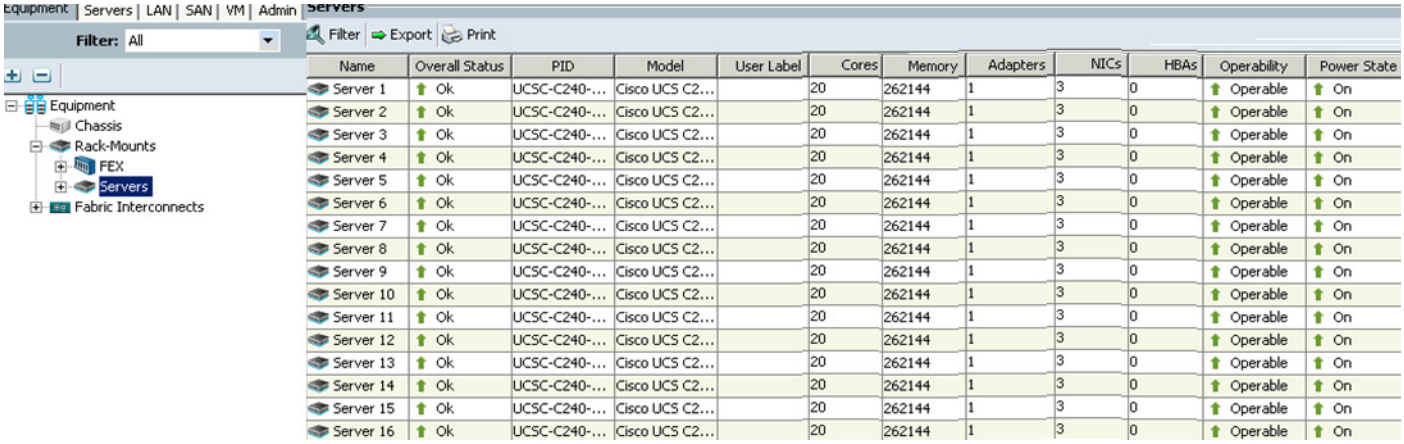
5. The Create Service Profile from Template window appears.

Figure 41 Creating Service Profiles



6. UCS Manager will then discover the servers. Association of the Service Profiles will happen automatically.

Figure 42 UCS Manager Showing all the Discovered Servers



Configuring Nytro Flash on all node for installing OS

This section details the RAID configuration of Nytro Flash on all the nodes for installing the operating system. The Nytro Flash on RAID controller is configured as boot volume with 40 GB size with RAID1 configuration for redundancy.

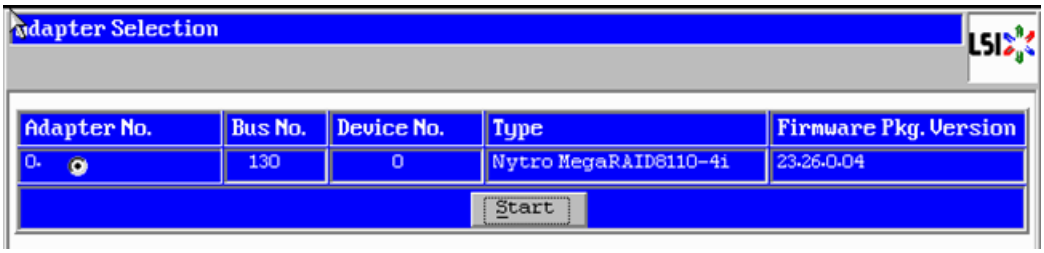
Note Installing OS on the Cisco Nytro MegaRaid card has the advantage of having all the 12 4TB Disk drives fully available for data.

There are several ways to configure RAID: using LSI WebBIOS Configuration Utility embedded in the MegaRAID BIOS, booting DOS and running MegaCLI commands, using Linux based MegaCLI commands, or using third party tools that have MegaCLI integrated. For this deployment, the Nytro Flash and the disk drives are configured using LSI WebBIOS Configuration Utility.

Follow these steps to create RAID1 on the Nytro Flash to install the operating system:

1. When the server is booting, the following text appears on the screen:
   a. Press **<Ctrl><H>** to launch the WebBIOS.
   b. Press **Ctrl+H** immediately.
2. The Adapter Selection window appears. Click **Start** to continue.

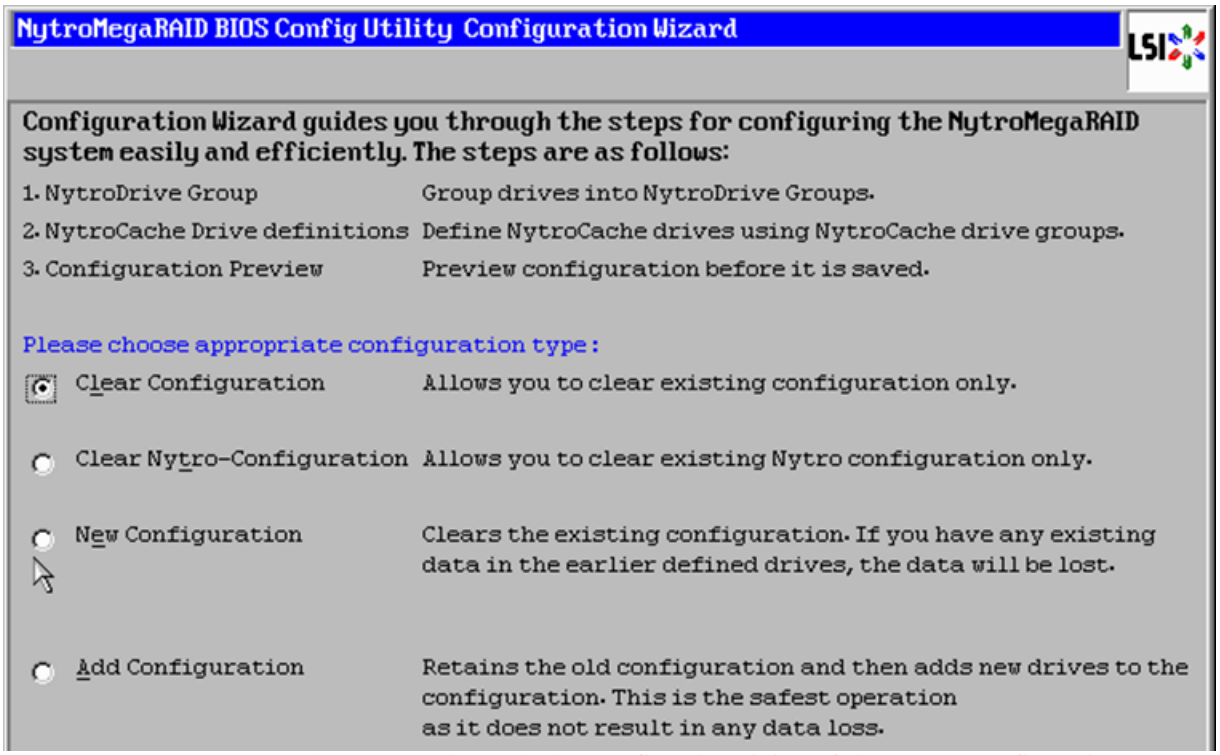
Figure 43 Adapter Selection for RAID Configuration



3. Click **Configuration Wizard**.

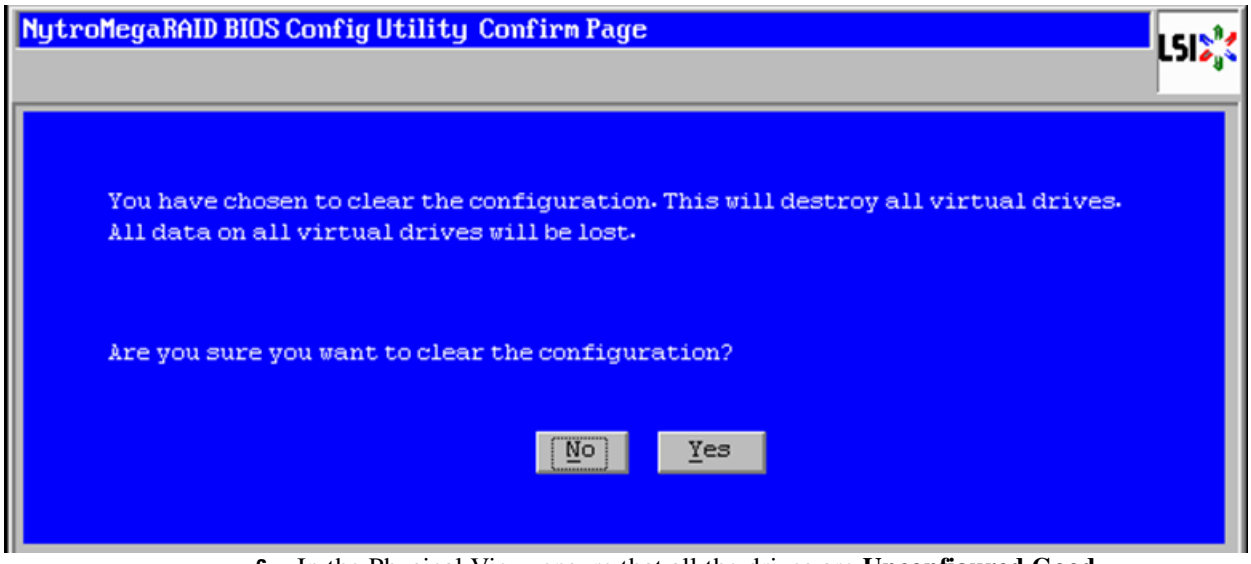
4. In the configuration wizard window, choose **Clear Configuration** and click **Next** to clear the existing configuration.

Figure 44 Clearing Current Configuration on the Controller



5. Choose **Yes** when asked to confirm the wiping of the current configuration.

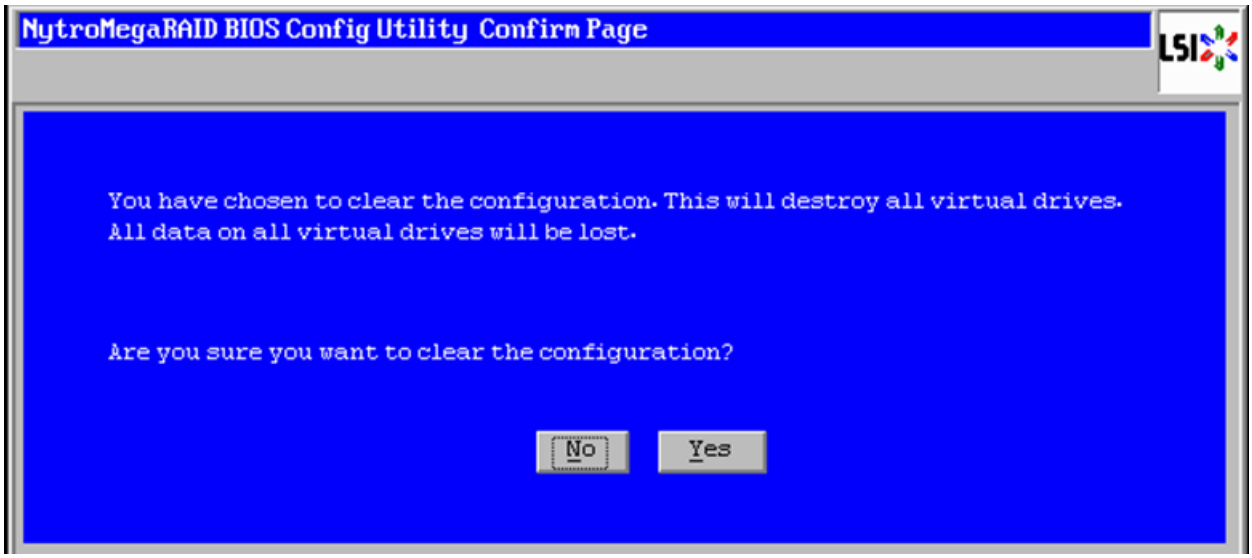
Figure 45 Confirming Clearance of the Previous Configuration on the Controller



6. In the Physical View, ensure that all the drives are **Unconfigured Good**.
7. Click **Configuration Wizard**.

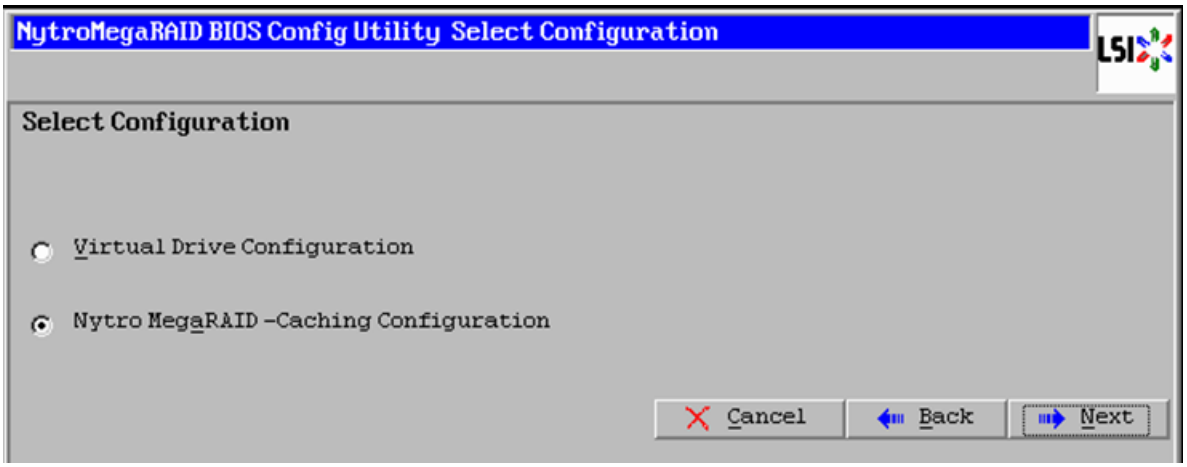
8. In the Configuration Wizard window choose the configuration type as **New Configuration** and click **Next**.

Figure 46 Creating a New Configuration



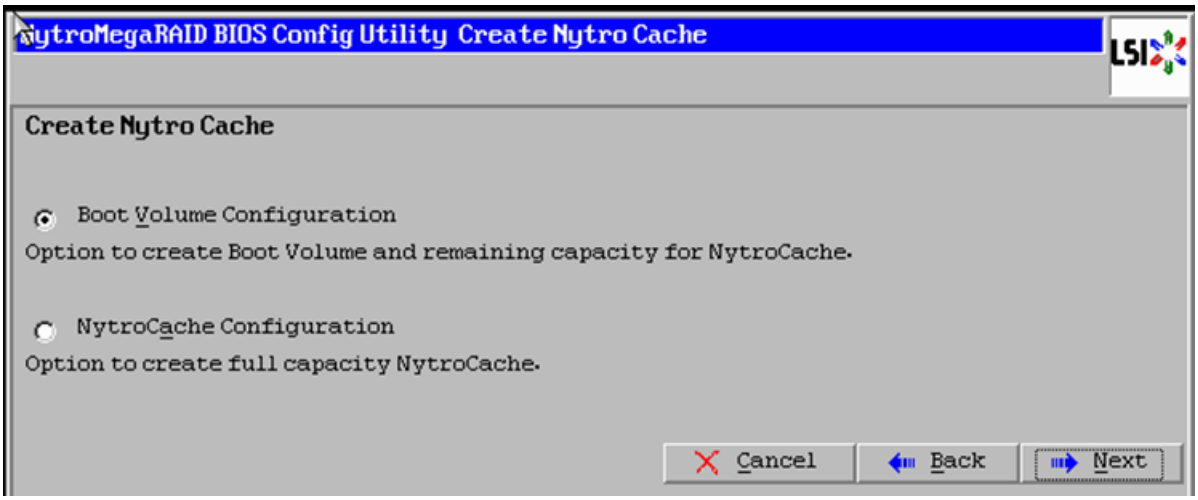
9. Choose **Yes** when asked to confirm the clear the current configuration.
10. Select the **Nytro MegaRAID Caching Configuration** radio button.

Figure 47 Selecting Nytro MegaRAID-Caching Configuration



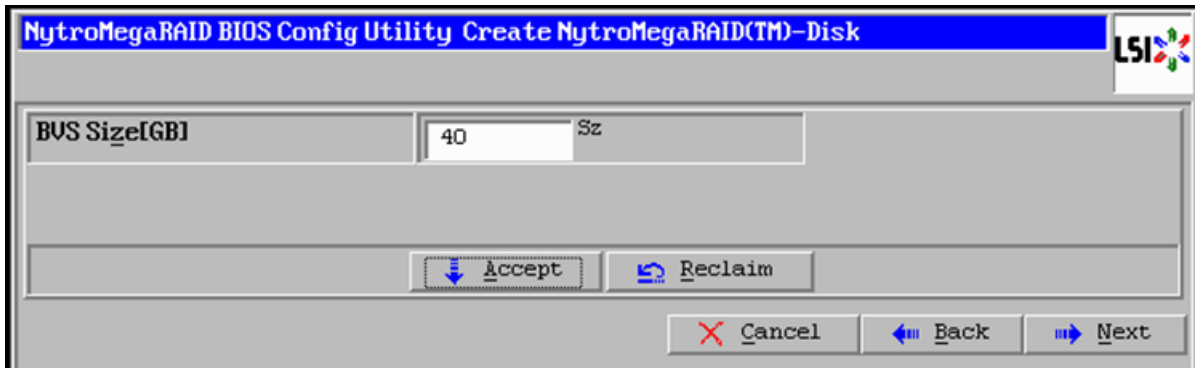
11. Select **Boot Volume Configuration** to create a Boot Volume for installing Operating system and the remaining capacity on Nytro Flash is used for NytroCache.

Figure 48 Selecting Boot Volume Configuration



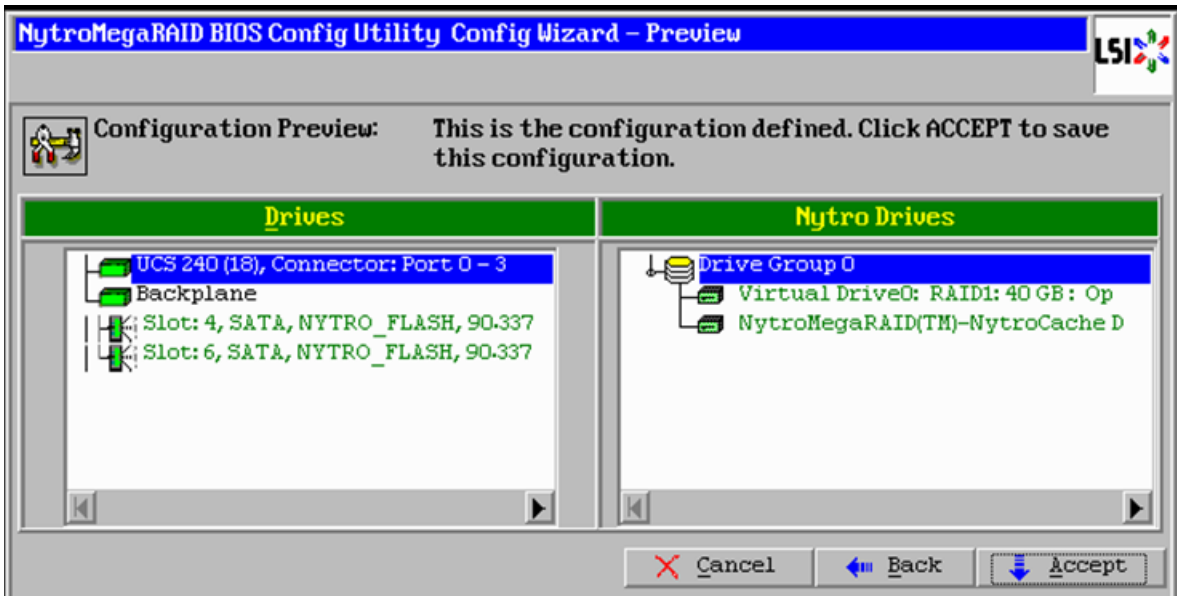
12. Type 40 for BVS Size [GB]. Click **Accept** and **Next**.

Figure 49 Creating Boot Volume of 40GB



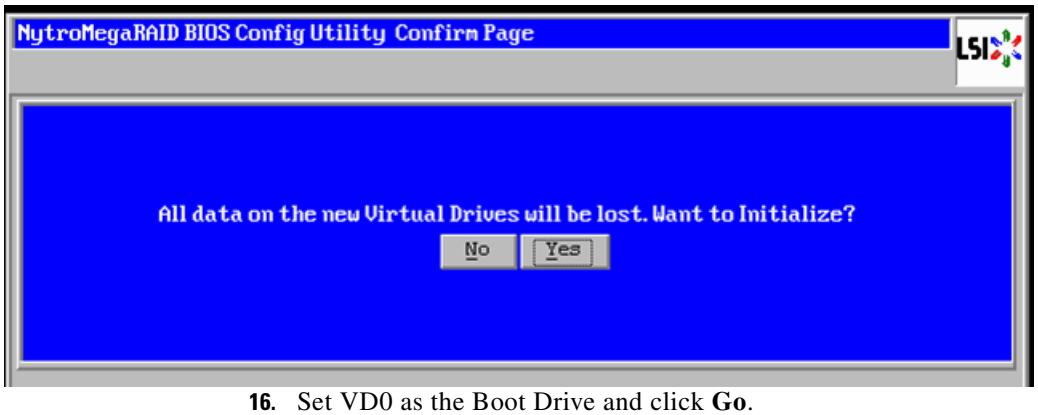
13. Verify the configuration and click **Accept** to save the Configuration.

Figure 50 Accept the Configuration



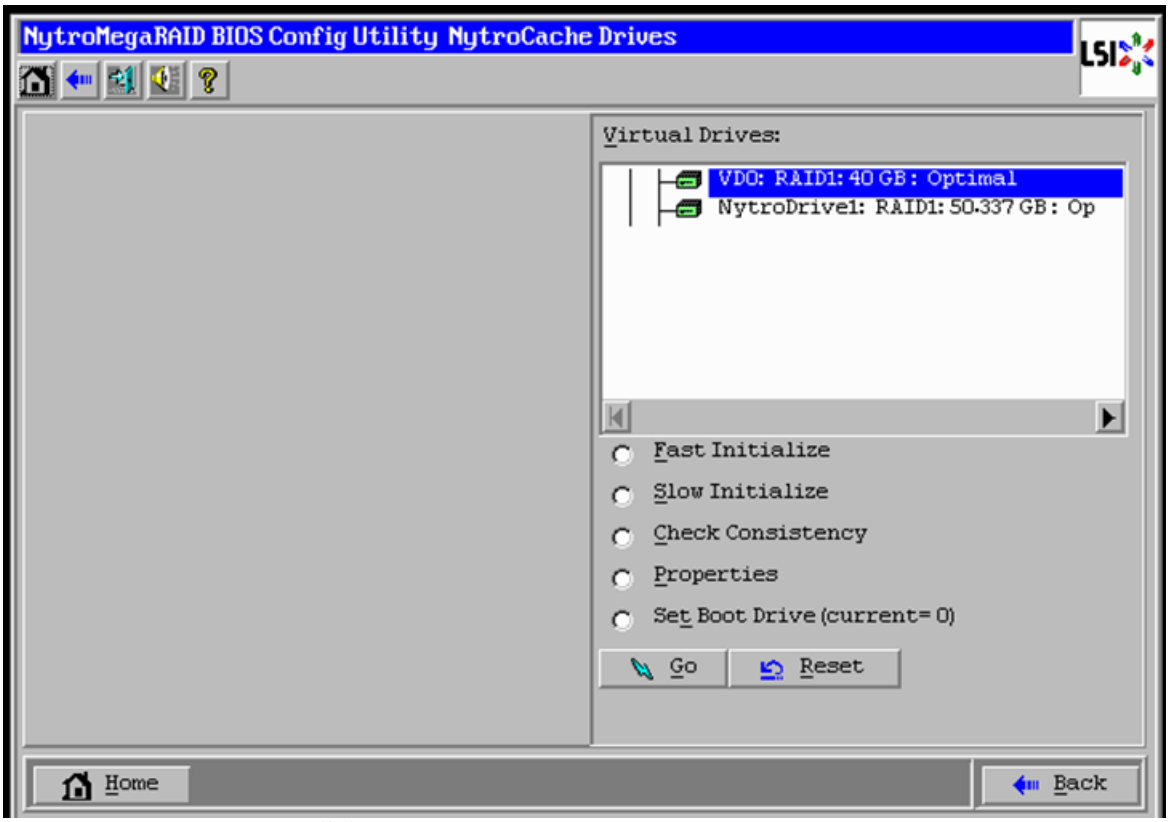
14. Click **Yes** to save the configuration.
15. Click **Yes**. When asked to confirm the initialization.

Figure 51 Confirm Saving Configuration



16. Set VD0 as the Boot Drive and click **Go**.

Figure 52 Setting Virtual Drive as Boot Drive



17. Click **Home**.
18. Review the Configuration and click **Exit.**
19. Confirm Exit and reboot the server.

Configuring Disk Drives on all nodes

This section details the configuration of disk drives on all nodes. The disk drives are configured as single RAID5 with 1MB stripe size, read ahead cache and write cache is enabled while battery is in use.

There are several ways in which you can configure RAID:

- Using LSI WebBIOS Configuration Utility embedded in the MegaRAID BIOS
- Booting DOS and running MegaCLI commands
- Using Linux based MegaCLI commands
- Using third party tools that have MegaCLI integrated

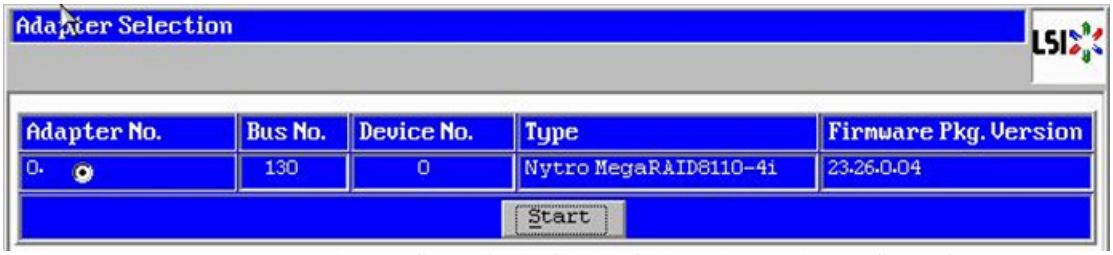
For this deployment, the disk drives are configured using LSI WebBIOS Configuration Utility.

Follow these steps to create RAID5 on all the 12 disk drive:

1. When the server is booting, the following text appears on the screen:
   a. Press **<Ctrl><H>** to launch the WebBIOS.
   b. Press **Ctrl+H** immediately.
2. The Adapter Selection window appears. Click **Start** to continue.

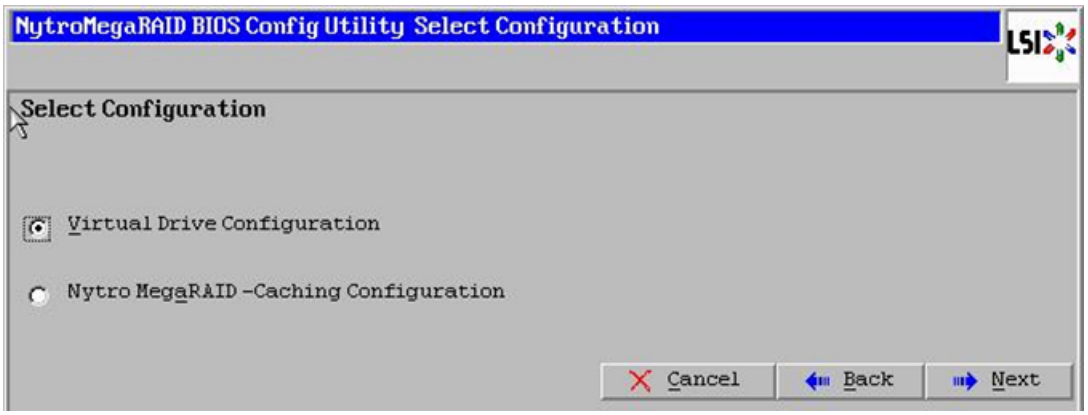
3. Click **Configuration Wizard**.

Figure 53 Adapter Selection for RAID Configuration



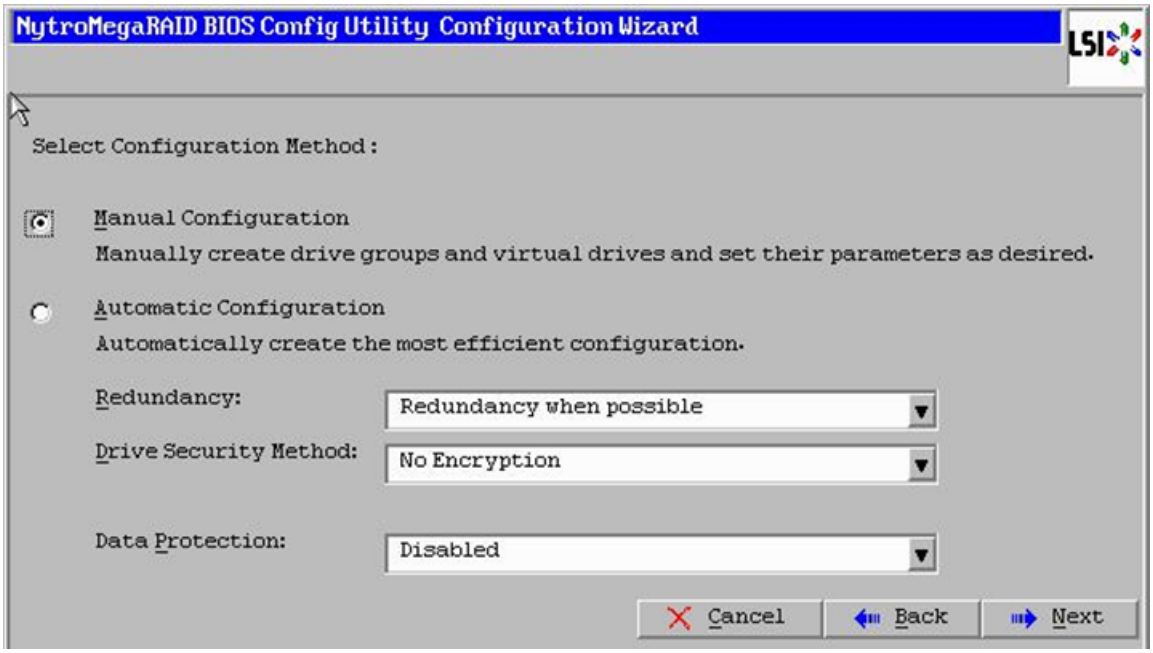
4. In the Configuration Wizard window choose the configuration type as **add Configuration** and click **Next**.

Figure 54 Setting Virtual Drive as Boot Drive



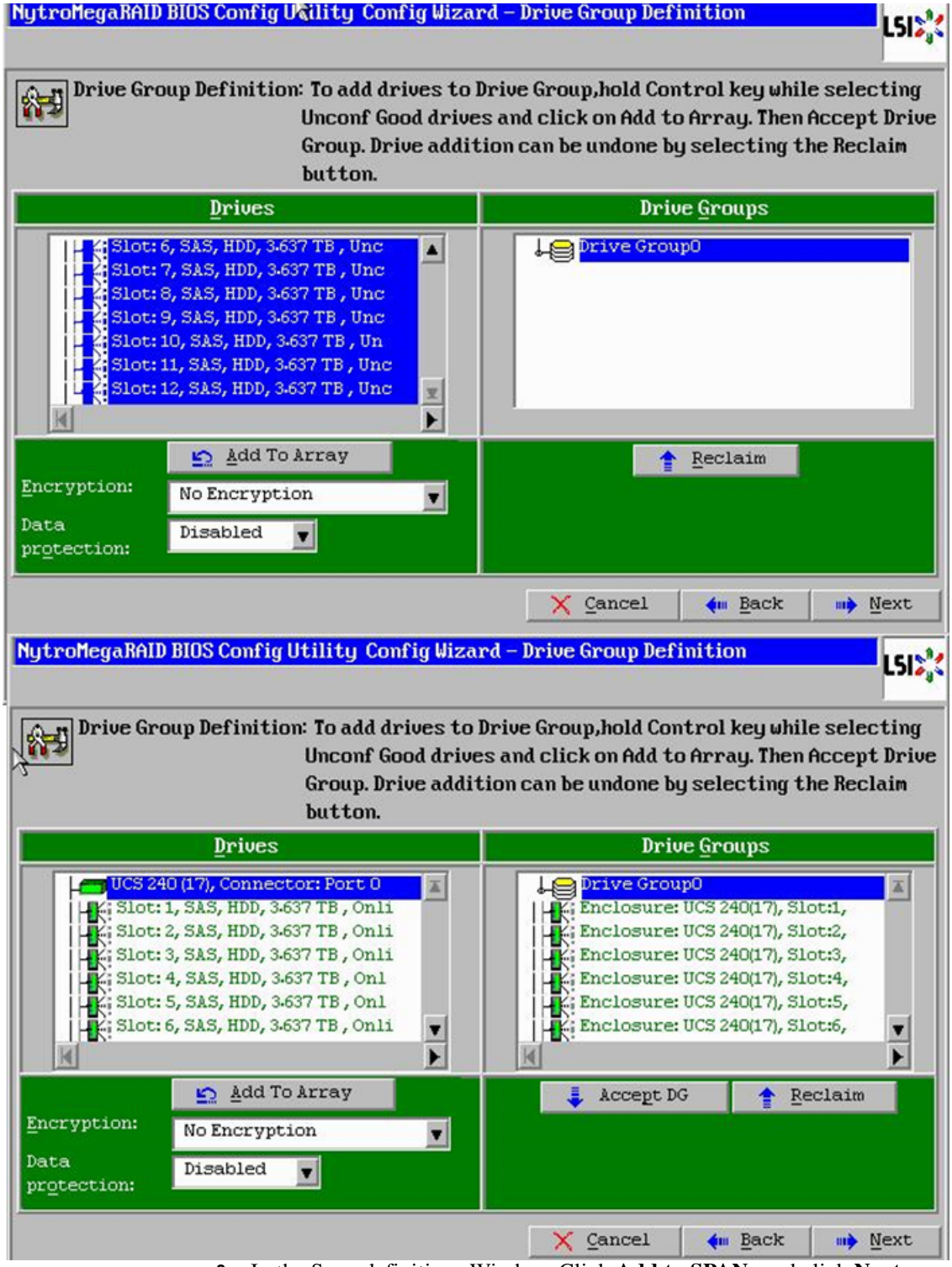
5. Select the configuration method as **Manual Configuration**; this enables you to have complete control over all attributes of the new storage configuration, such as, the drive groups, virtual drives and the ability to set their parameters.
6. Click **Next .**

Figure 55 Choosing Manual Configuration Method



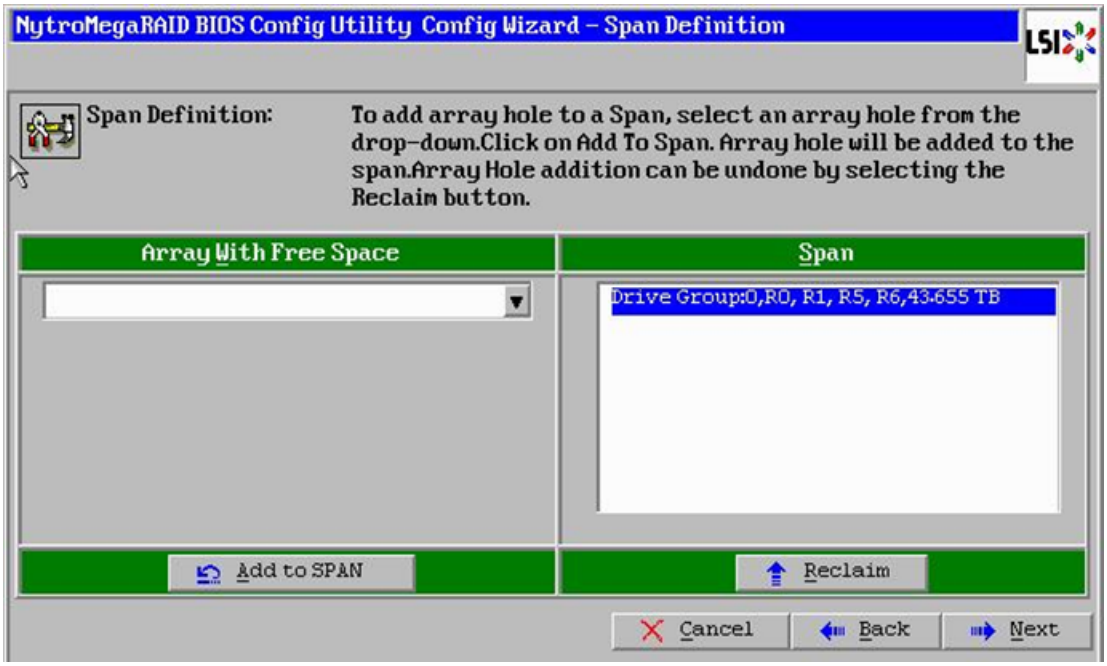
7. The Drive Group Definition window appears. Use this window to choose all the 12 drives to create drive groups.
8. Click **Add to Array** to move the drives to a proposed drive group configuration in the Drive Groups pane. Click **Accept DG** and then, click **Next.**

Figure 56 Selecting all Drives and Adding to Drive Group



9. In the Span definitions Window, Click **Add to SPAN** and click **Next.**

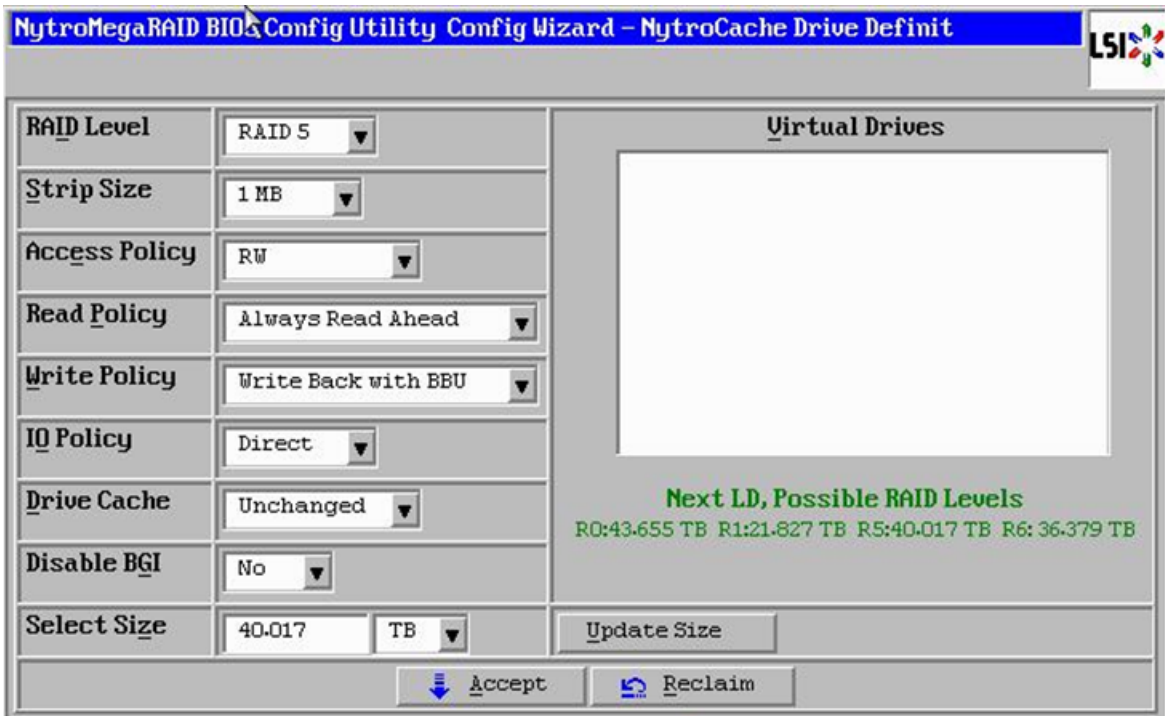
Figure 57 Span Definition Window - Adding to Span



10. In the Virtual Drive definitions window, follow these steps to change the virtual drive definition:
    a. Click **Update Size**.
    b. Change Strip Size to **1MB**. A larger strip size produces higher read performance
    c. From the Read Policy drop-down list, choose **Always Read Ahead.**
    d. From the Write Policy drop-down list, choose **Write Back with BBU.**
    e. Make sure the RAID Level is set to **RAID5**.
    f. Click **Accept** to accept all the changes to the virtual drive definitions.
    g. Click **Next.**

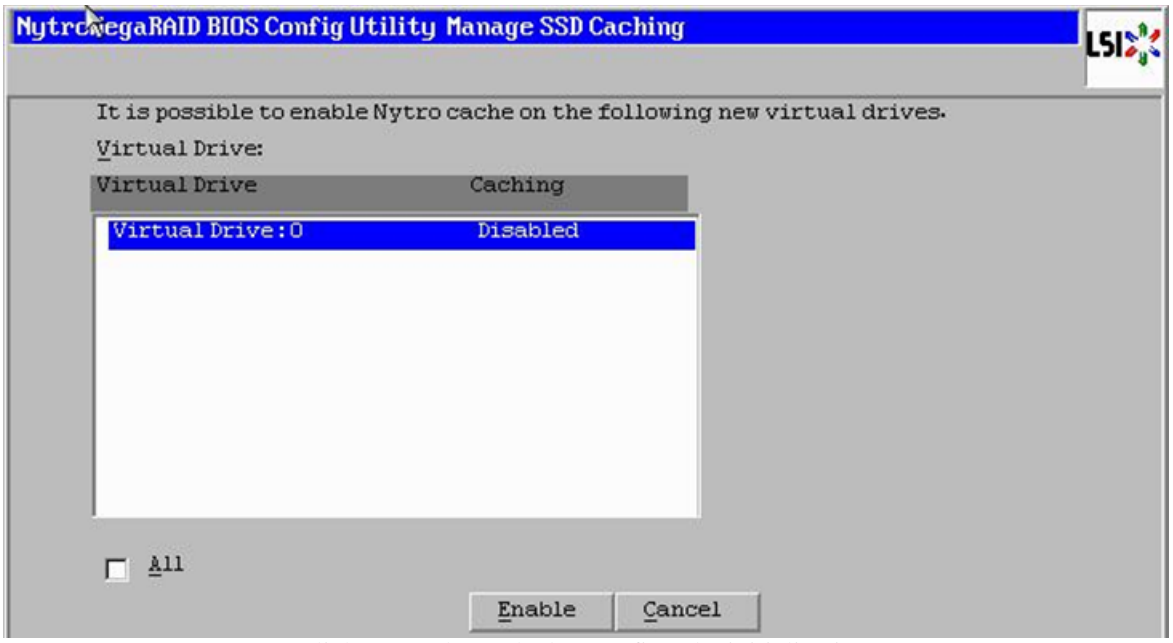
Note Clicking the **Update Size** might change some of the settings in the window. Make sure all the settings are correct before accepting.

Figure 58 Virtual Drive Definition Window



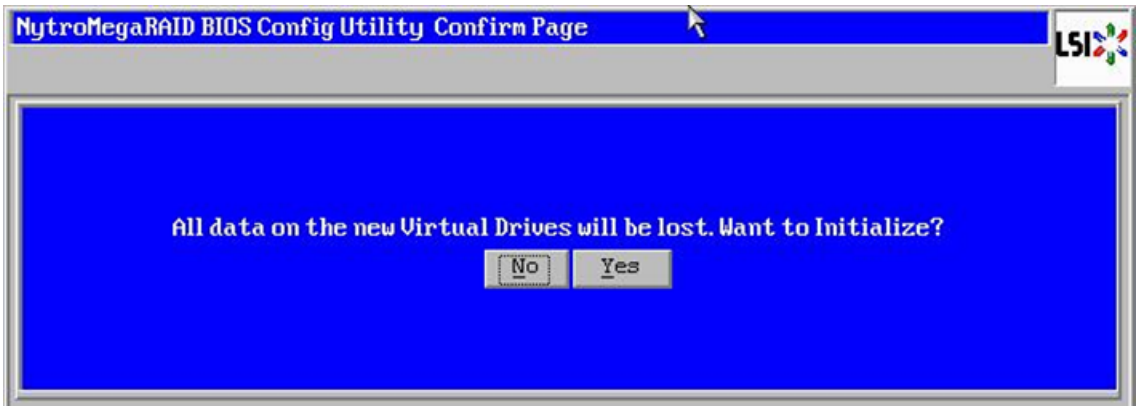
11. After the virtual drive definitions are done, click **Next**. The Configuration Preview window appears showing VD1.
12. Check the virtual drive configuration in the Configuration Preview window and click **Accept** to accept the configuration.
13. Click **Yes** to save the configuration.
14. In the managing SSD Caching Window, Click **Cancel.**

Figure 59 SSD Caching Window



15. Click **Yes**. When asked to confirm the initialization.

Figure 60 Initializing Virtual Drive Window



16. Click **Home**.
17. Review the Configuration and click **Exit.**

Installing Ubuntu Server 12.04.4 LTS with KVM

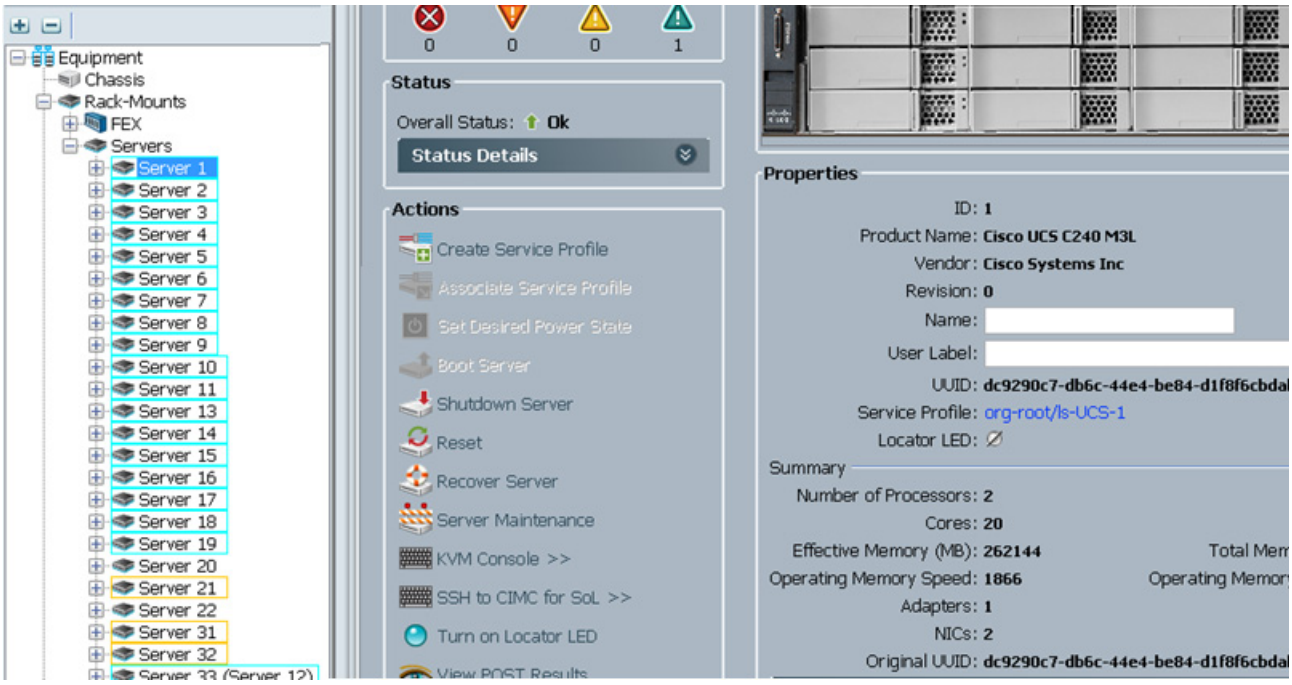
The following section provides detailed procedures for installing Ubuntu Server 12.04.4 LTS.

There are multiple methods to install Ubuntu Server OS. The installation procedure described in this deployment guide uses KVM console and virtual media from Cisco UCS Manager.

1. Log in to the Cisco UCS 6296UP Fabric Interconnect and launch the Cisco UCS Manager application.
2. Select the **Equipment** tab.

3. In the navigation pane expand **Rack-Mounts** and **Servers**.
4. Right-click on the server requiring OS Installation and select **KVM Console**.

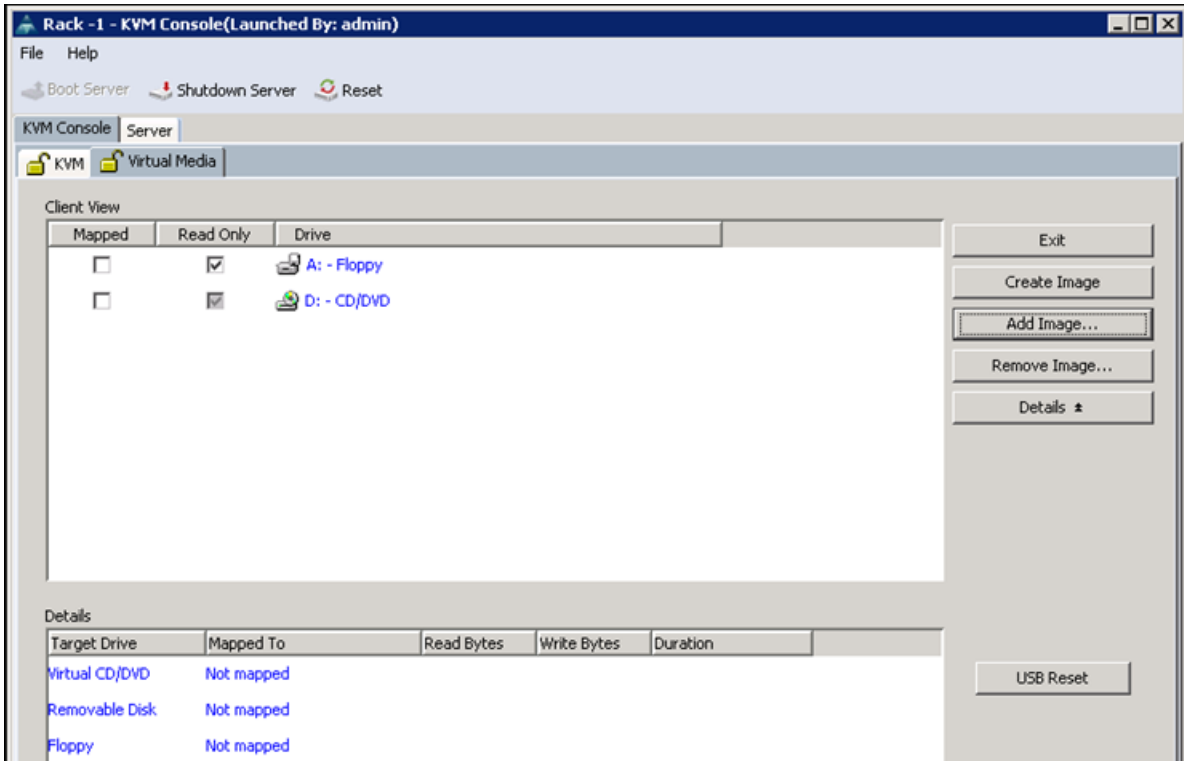
Figure 61 Opening KVM Console of the Server



5. In the KVM window, select the **Virtual Media** tab.
6. Click Add Image... in the Virtual Media selection window.
7. Browse to the Ubuntu Server 12.04.4 LTS installer ISO image file.

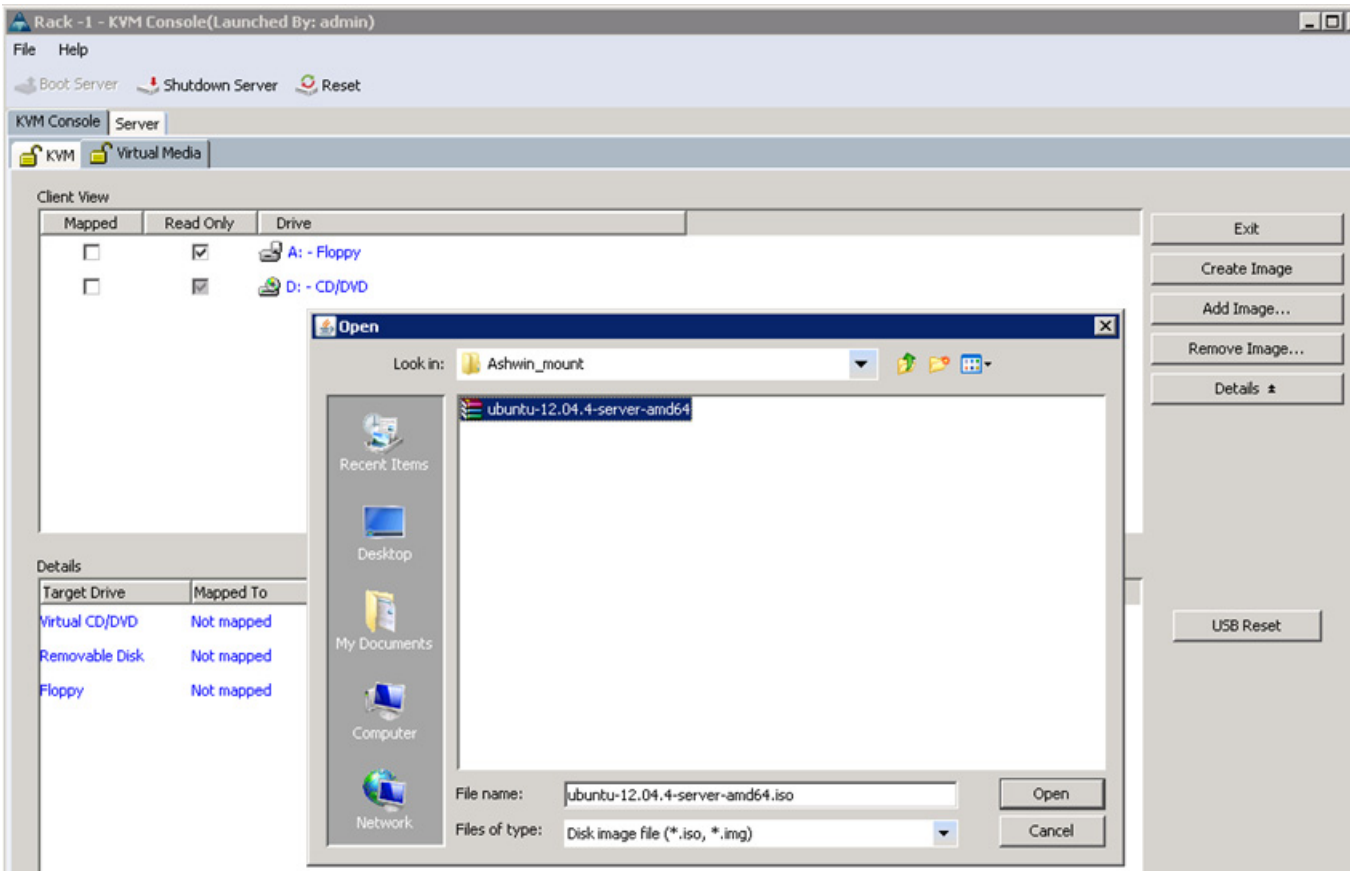
Note The Ubuntu Server 12.04.4 LTS ISO can be downloaded from: http://www.ubuntu.com/download/server/thank-you?country=US&version=12.04.4&architecture=amd64.

Figure 62 Adding an ISO Image



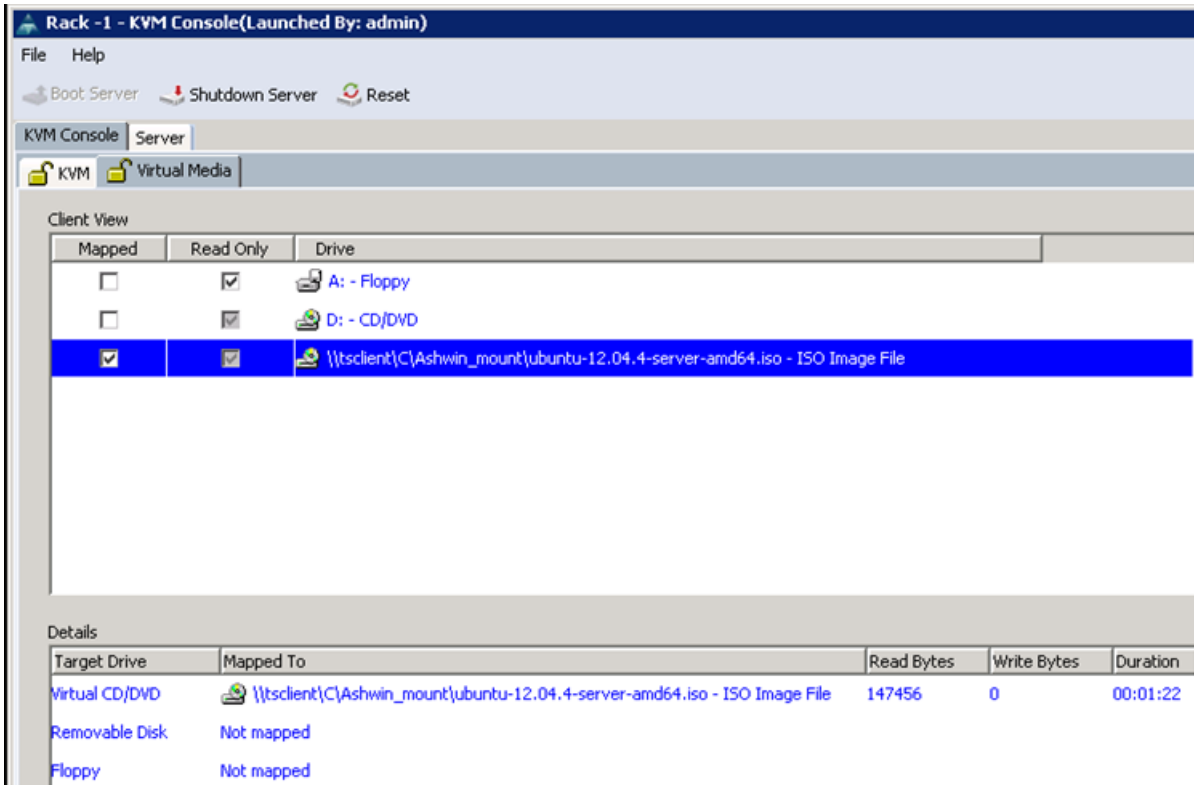
8. Select Ubuntu Server 12.04.4 LTS installer ISO image file and click Open to add the image to the list of virtual media.

Figure 63 Browse to Ubuntu Server 12.04.4 LTS ISO Image



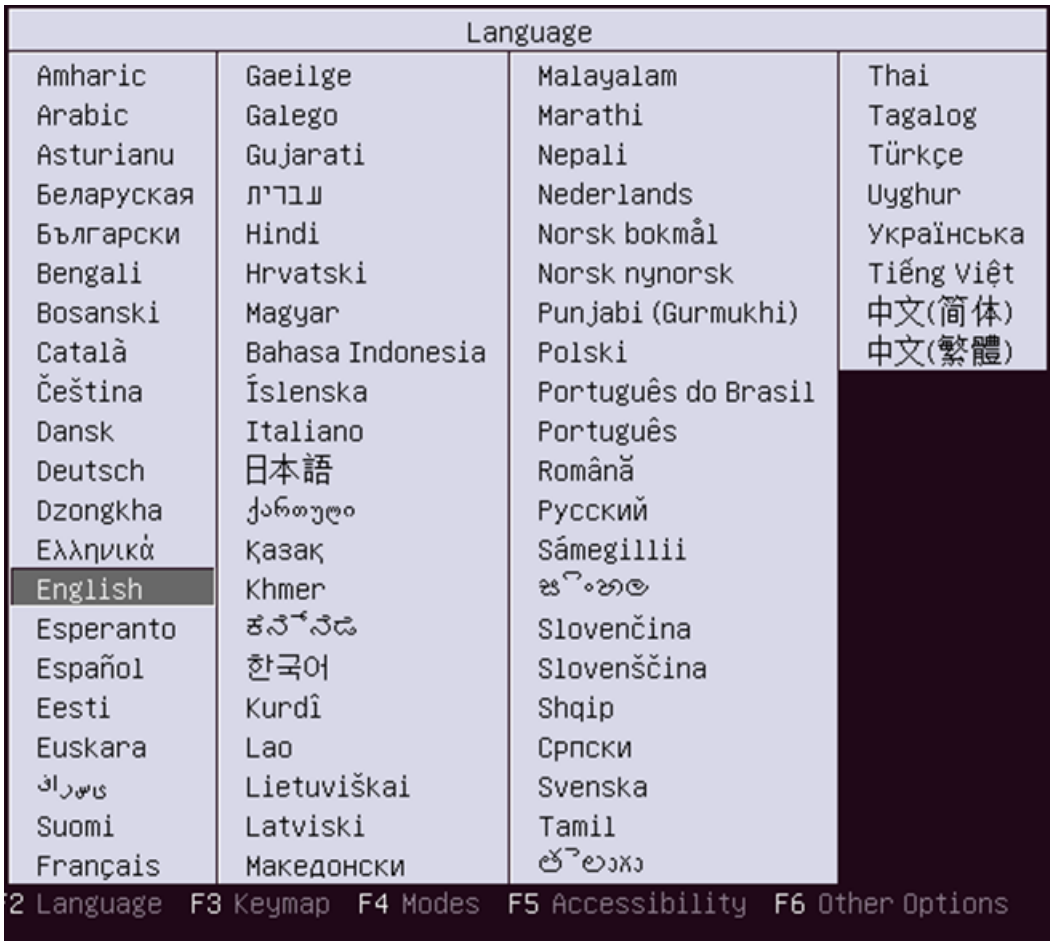
9. Check the **Mapped** check box, next to the entry corresponding to the image you just added.
10. In the KVM window, select the **KVM** tab to monitor during boot.
11. In the KVM window, select Boot Server.
12. Click **OK**.
13. Click **OK** to reboot the system.

Figure 64 Mapping ISO Image



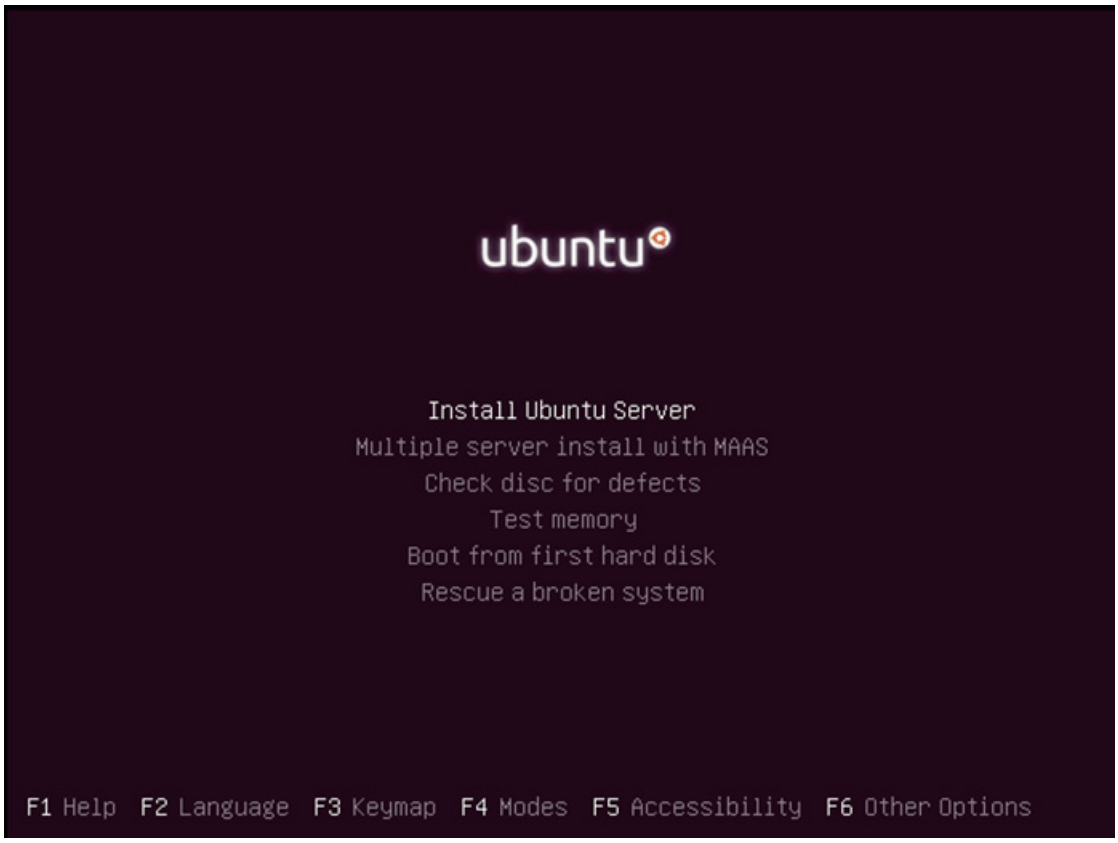
14. On reboot, the machine detects the presence of the Ubuntu Server 12.04.4 LTS ISO install media. Choose **English** from the list.

Figure 65 Selecting Language



15. Select the **Install Ubuntu Server** option.

Figure 66 Selecting Install Ubuntu Server Option

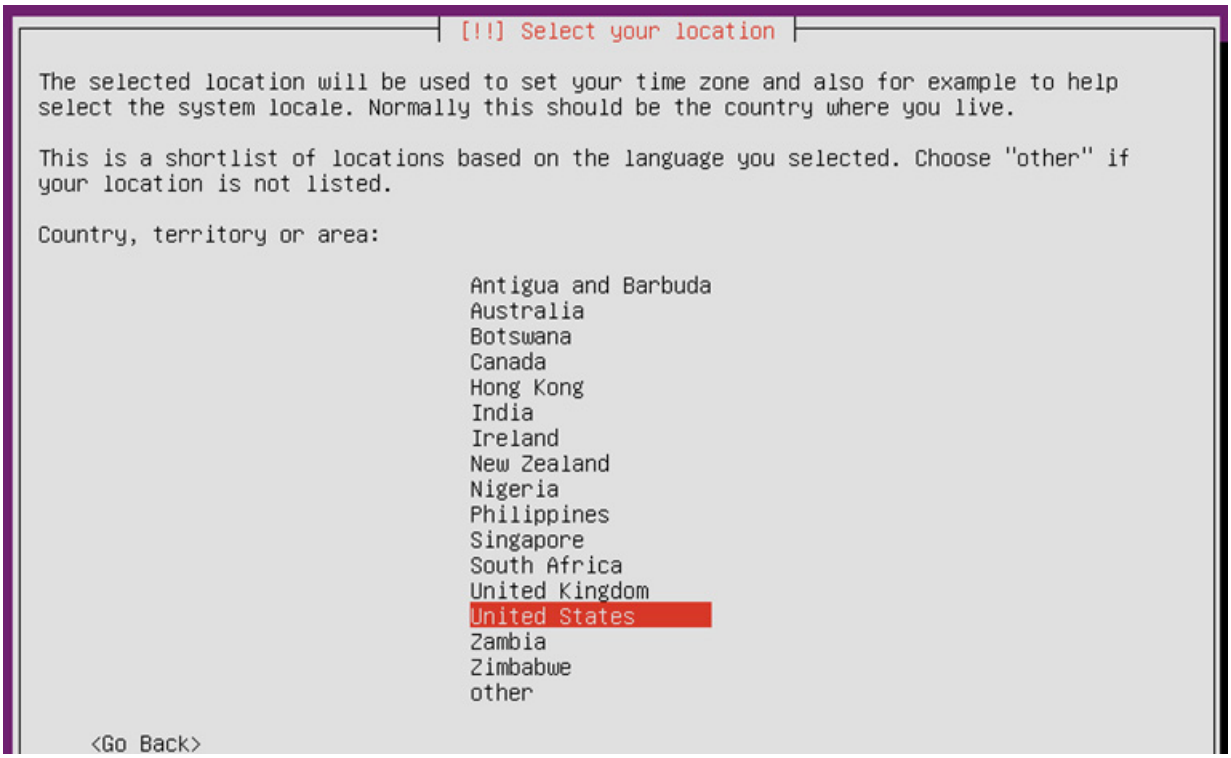


16. Choose the language for Installation process and the default language for the installed server.

Figure 67 Selecting Language for Installation and Default Language for the Server

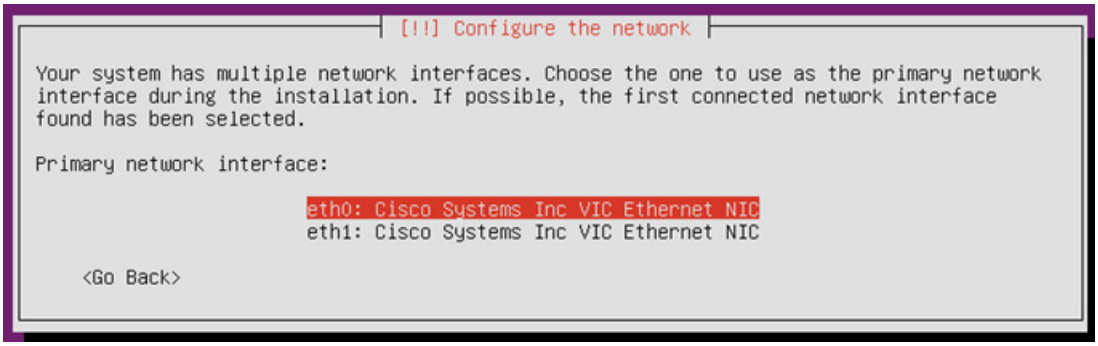
17. Select your location from the list.

Figure 68 Selecting Location



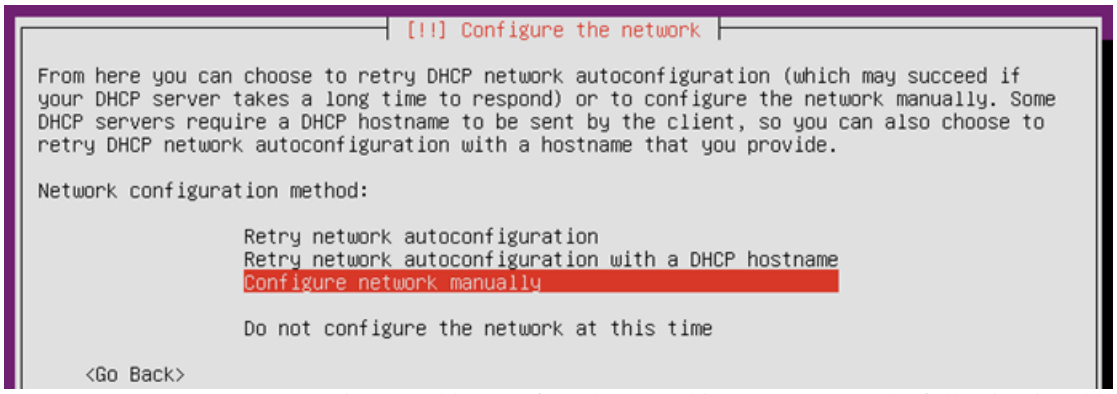
18. Click **No** when asked to detect keyboard layout.
19. Choose your keyboard layout from the list.
20. Select Network Interfaces eth0 for assigning IP address.

Figure 69 Configuring Network for eth0



21. Select **Configure network manually**.

Figure 70 Selecting Network Configuration Method



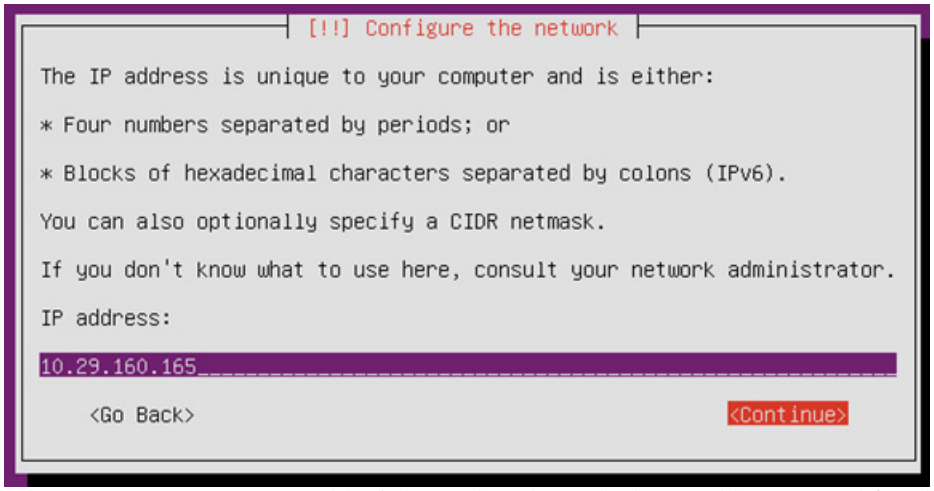
22. Assign IP addresses for eth0. For this setup we use the following ip address:

    IP Address:10.29.160.165

    Netmask: 255.255.255.0

    Gateway: 10.29.160.1

Figure 71 Entering IP Address



23. Table 6 shows the eth0 and eth1 IP addresses used for all the servers in the cluster.

Table 6 Hostnames and IP Addresses

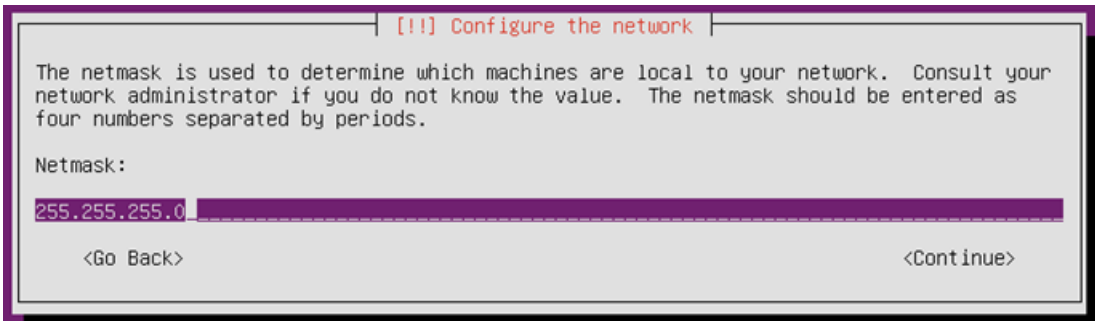
Hostname	eth0	eth1
Ubuntu1	10.29.160.165	192.168.10.165
Ubuntu2	10.29.160.166	192.168.10.166
Ubuntu3	10.29.160.167	192.168.10.167
Ubuntu4	10.29.160.168	192.168.10.168
Ubuntu5	10.29.160.169	192.168.10.169
Ubuntu6	10.29.160.170	192.168.10.170
Ubuntu7	10.29.160.171	192.168.10.171

Table 6 Hostnames and IP Addresses

Hostname	eth0	eth1
Ubuntu8	10.29.160.172	192.168.10.172
Ubuntu9	10.29.160.173	192.168.10.173
Ubuntu10	10.29.160.174	192.168.10.174
Ubuntu11	10.29.160.175	192.168.10.175
Ubuntu12	10.29.160.176	192.168.10.176
Ubuntu13	10.29.160.177	192.168.10.177
Ubuntu14	10.29.160.178	192.168.10.178
Ubuntu15	10.29.160.179	192.168.10.179
Ubuntu16	10.29.160.180	192.168.10.180
..	..	..
Ubuntu64	10.29.160.228	192.168.10.228

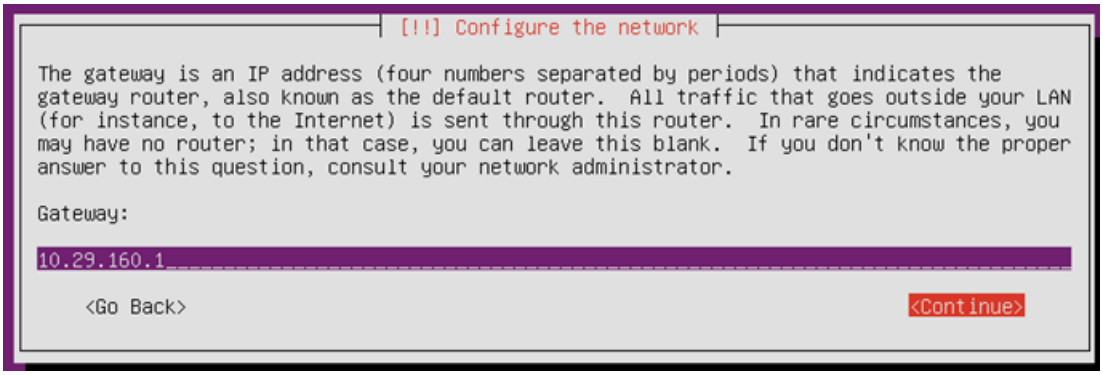
24. Enter netmask as **255.255.255.0** and click **Continue**.

Figure 72 Entering Netmask



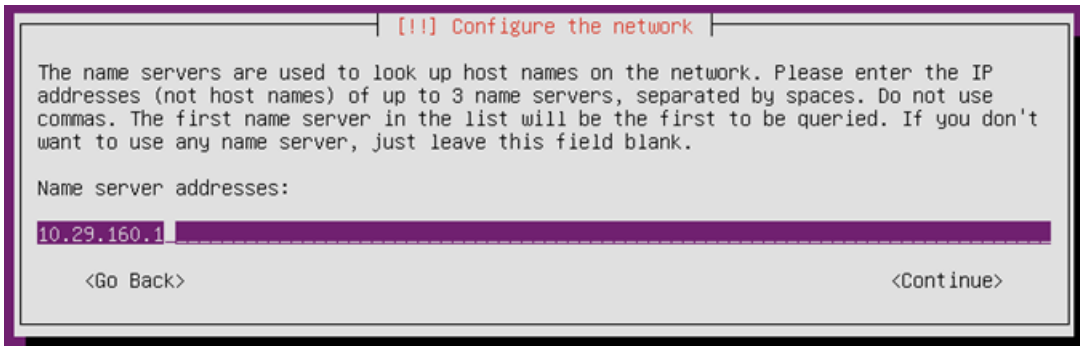
25. Enter your gateway address and click **Continue**.

Figure 73 Entering Gateway



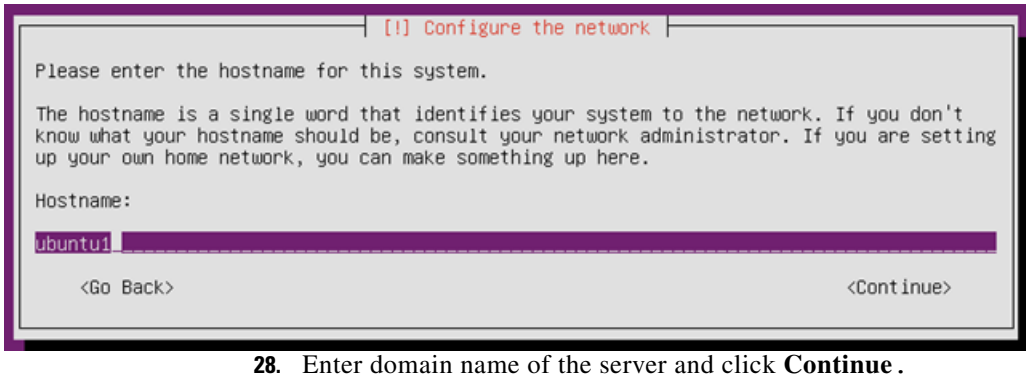
26. Enter your name server addresses and click **Continue**.(Optional)

Figure 74 Entering Name Server Address



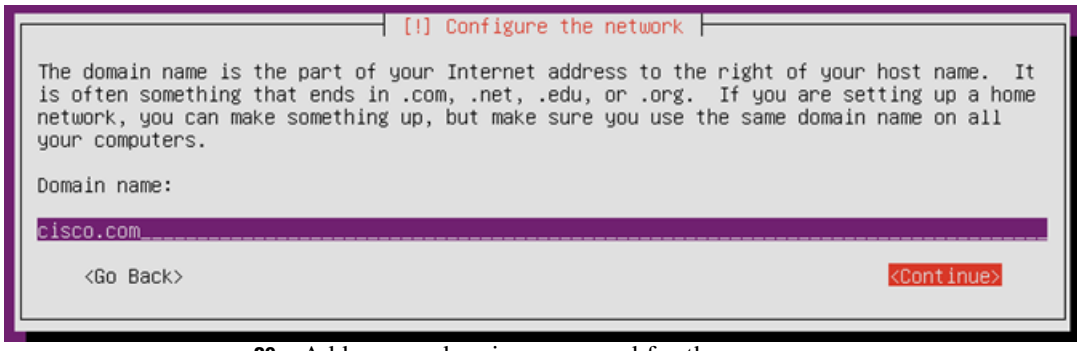
27. Enter hostname of the server and click **Continue**.

Figure 75 Entering Hostname for the Server



28. Enter domain name of the server and click **Continue**.

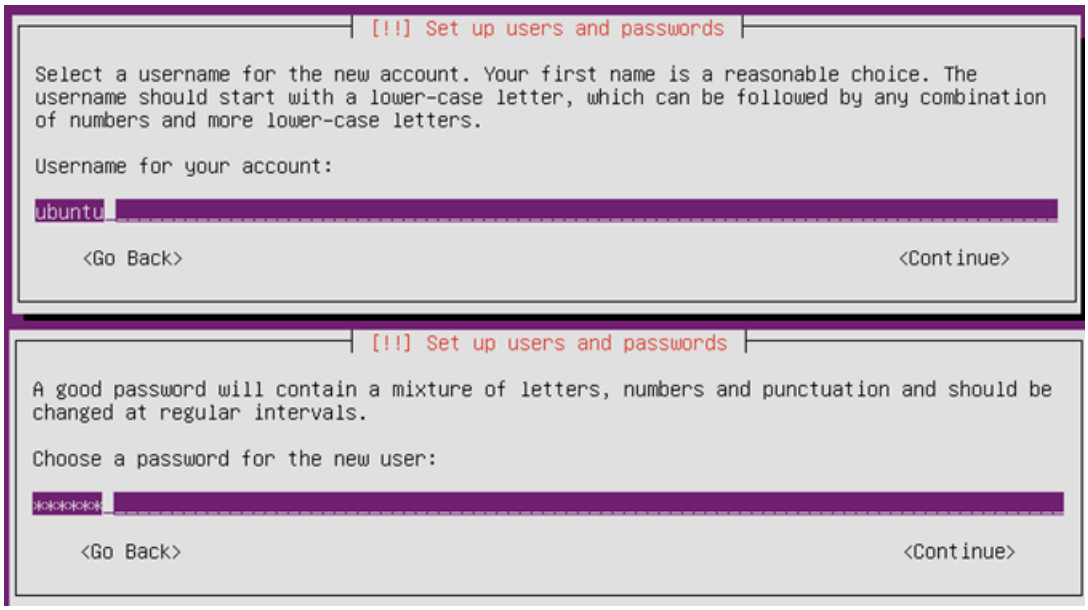
Figure 76 Entering Domain Name for the Server



29. Add user and assign password for the user.

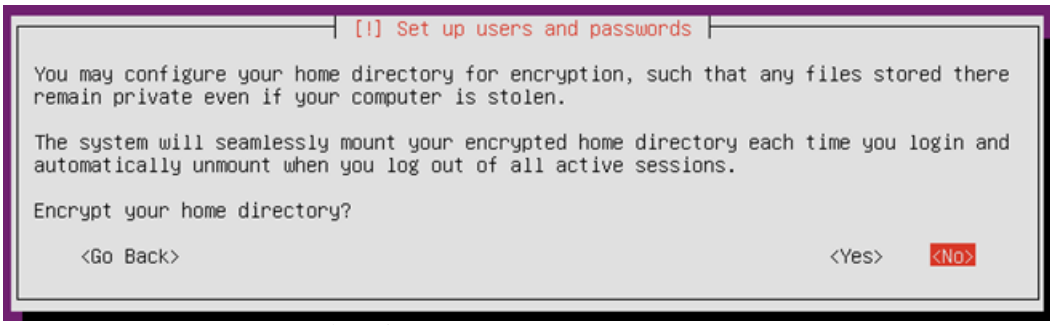
Figure 77 Setting up User for the Server

Figure 78 Setting up Username and Password



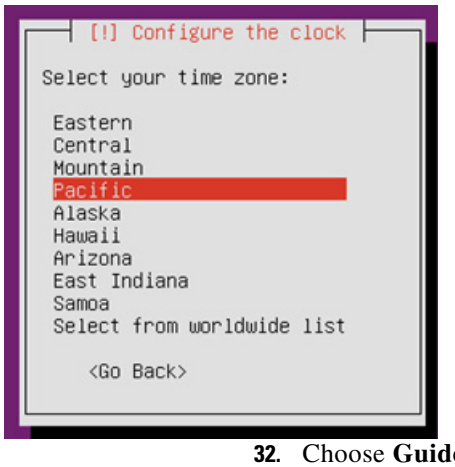
30. Choose **No** to disable encryption on home directory.

Figure 79 Configuring Encryption for Home Directory



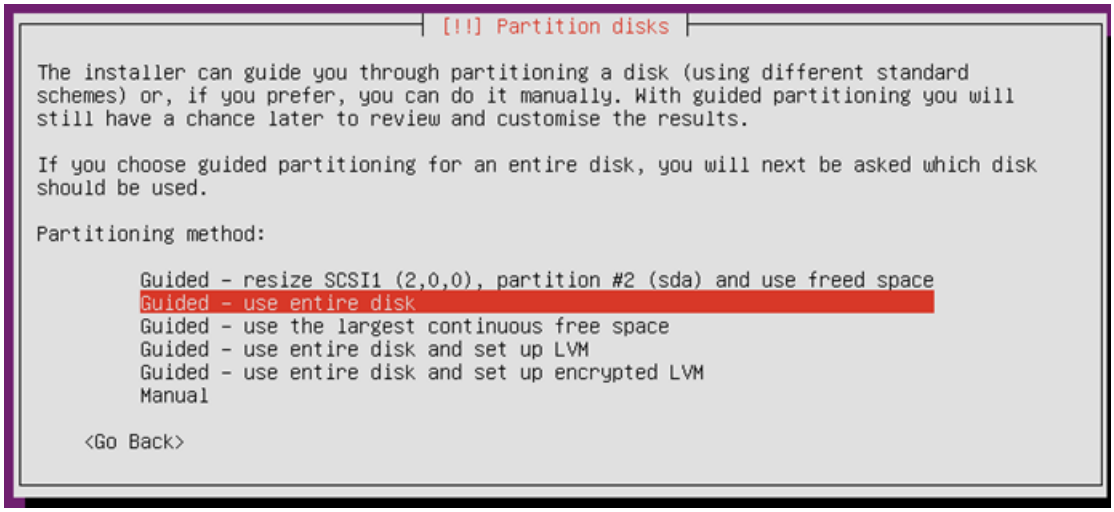
31. Select time zone.

Figure 80 Selecting Time Zone



32. Choose **Guided-use entire disk**. All the data on the disk will be erased.

Figure 81 Choosing Partitioning Method

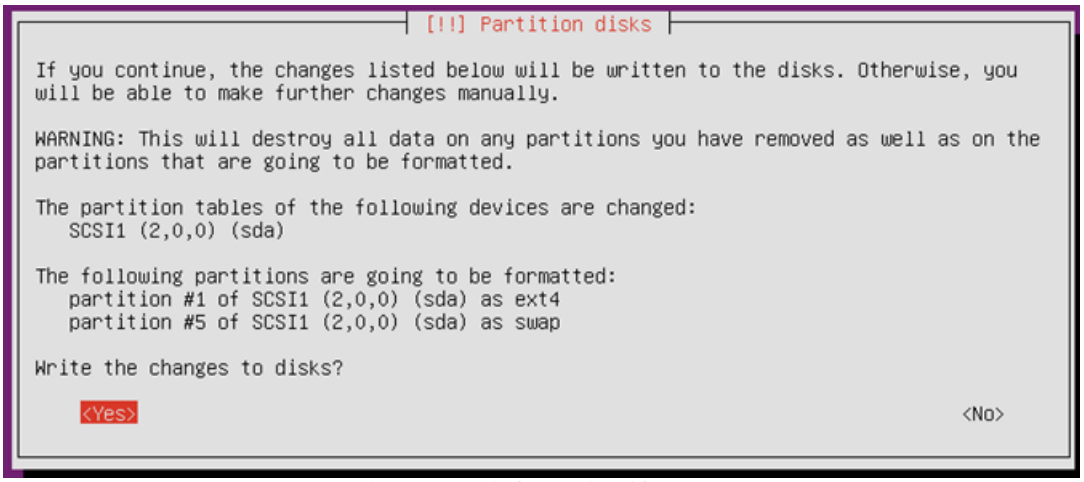


33. Select 42.9 GB LSI NMR8110-4i from the list

Figure 82 Partitioning Nytro Flash VD for Installing Operating System

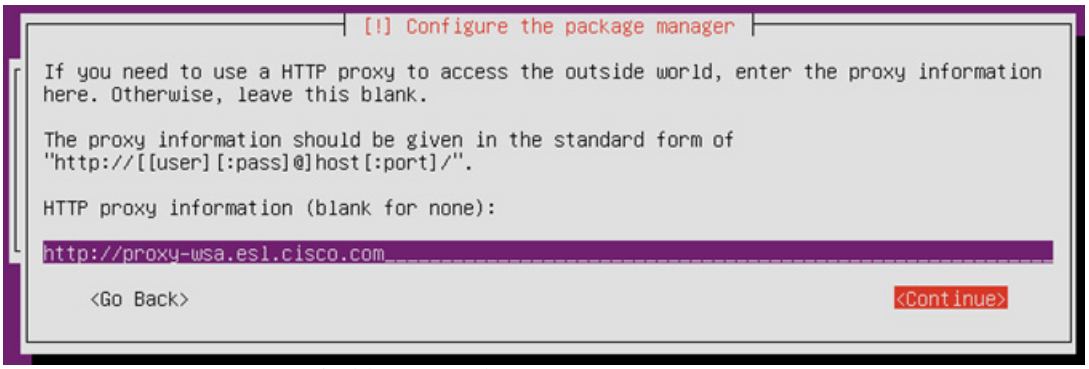
34. Choose **Yes** to confirm partitioning.

Figure 83 Confirm Write Changes



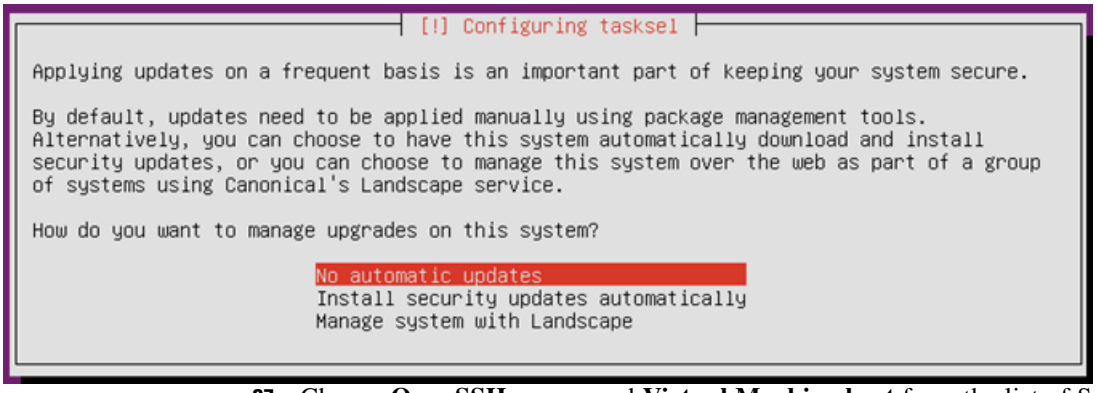
35. Enter http proxy information if needed.

Figure 84 Configuring http Proxy



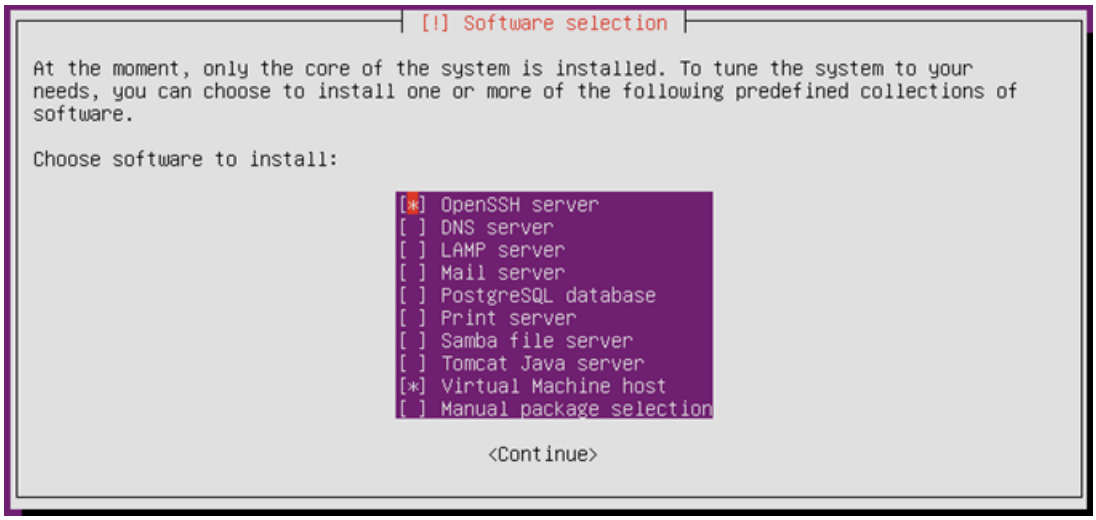
36. Disable auto update.

Figure 85 Selecting Updating Options



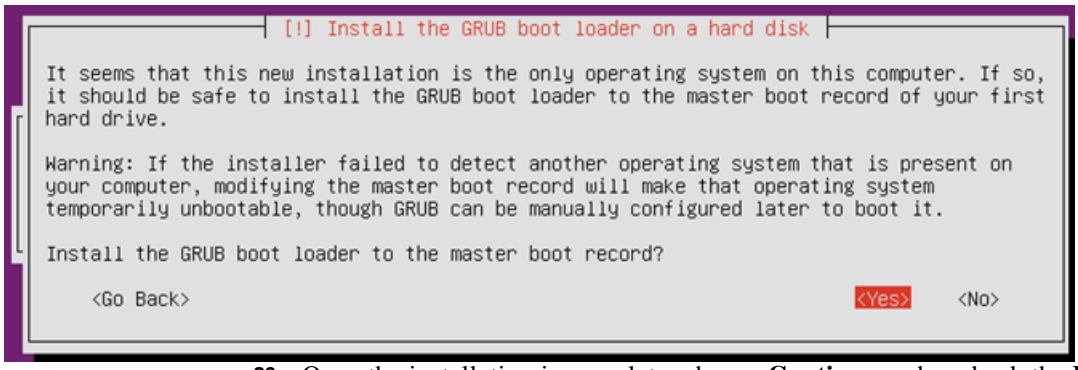
37. Choose **OpenSSH server** and **Virtual Machine host** from the list of Software for Installation.

Figure 86 Selecting Software for Installation



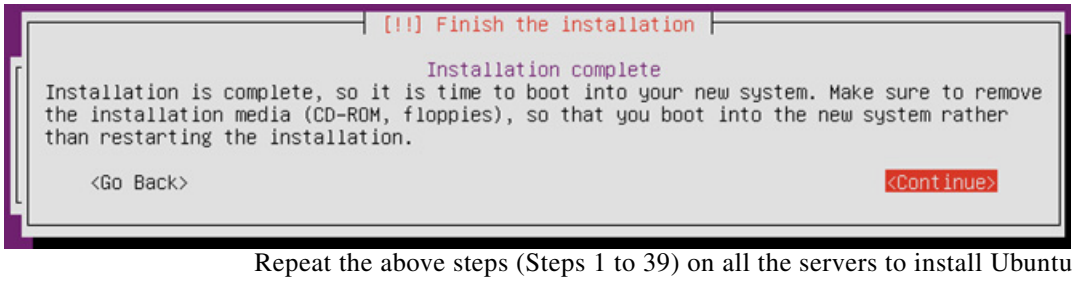
38. Installing GRUB boot loader to Master Boot Record.

Figure 87 Installing GRUB Boot Loader



39. Once the installation is complete, choose **Continue** and uncheck the **Ubuntu Server 12.04.4 LTS installer** ISO image file from KVM Virtual Media tab.

Figure 88 Finishing the Installation



Repeat the above steps (Steps 1 to 39) on all the servers to install Ubuntu Server 12.04.4 LTS. The OS installation and configuration of the nodes that is mentioned above can be automated through PXE boot or third party tools.

Note The hostnames and their corresponding IP addresses are shown in Table 6.

Post OS Install Configuration

Choose one of the nodes of the cluster or a separate node as Admin Node for management of all the other nodes, node ubuntu20 for the purpose of this CVD.

Note

- All Nodes in the system are assumed to have access to Internet.
- Most commands in this CVD are run as “sudo”. To avoid this, one could change profile as “sudo su” and run the commands without “sudo” as this is with full root privileges.

Most of the following configurations are done from the admin node.

Update Proxy for apt

On the admin node add the proxy to apt

```
cat /etc/apt/apt.conf
Acquire::http::Proxy "http://proxy-wsa.esl.cisco.com:80";
```

Run the following command after updating the file as above

```
sudo apt-get update
```

Note This apt.conf should be updated on all nodes and this will be done after clush is installed as per the instructions given in the “Setup Parallel Shell - clush” section on page 80.

Dnsmasq

On the admin node install dnsmasq as follows:

```
sudo apt-get install dnsmasq
```

Password-less login

On the admin node run command ssh-keygen:

```
ssh-keygen
```

Run the following command from the admin node to copy the public key id_rsa.pub to all the nodes of the cluster. ssh-copy-id appends the keys to the remote-host's .ssh/authorized_key.

```
for IP in {165..228}; do echo -n "$IP -> "; ssh-copy-id -i ~/.ssh/id_rsa.pub
10.29.160.$IP; done

Enter yes for Are you sure you want to continue connecting (yes/no)?
```

Setup Parallel Shell - clush

Install clustershell as follows

```
sudo apt-get install clustershell
```

Update clustershell config to identify all nodes in the cluster

```
$ cat /etc/clustershell/groups
all: ubuntu[1-64]
```

Note Provide “all” in the above file relates to all nodes to be included in “-a” option for clush

For all cluster nodes add the following lines in the end of /etc/sudoers to prevent Ubuntu asking for Sudo password in order for clush to work

```
ubuntu@ubuntu20:~$ sudo tail -3 /etc/sudoers
#includedir /etc/sudoers.d
ubuntu ALL=NOPASSWD: ALL

Copy /etc/apt/apt.conf on all nodes

clush -b -a -c /etc/apt/apt.conf --dest=/home/ubuntu
clush -b -a sudo mv /home/ubuntu/apt.conf /etc/apt/
clush -b -a cat /etc/apt.conf
clush -b -a sudo apt-get update
```

Nameserver

On Admin node move the link - /etc/resolv.conf to static file for backing up:

```
sudo mv /etc/resolv.conf /etc/resolv.conf.orig
sudo vi /etc/resolv.conf
nameserver 8.8.8.8

clush -b -a -c /etc/resolv.conf --dest=/home/ubuntu
clush -b -a sudo mv /home/ubuntu/resolv.conf /etc/
clush -b -a cat /etc/resolv.conf
```

Ping www.ubuntu.com to confirm

Configuring /etc/hosts

Follow these steps to create the host file across all the nodes in the cluster:

1. Populate the host file with IP addresses and corresponding hostnames on the Admin node (ubuntu20).

```
vi /etc/hosts
127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4
::1 localhost localhost.localdomain localhost6 localhost6.localdomain6
10.29.160.165 ubuntu1.cisco.com ubuntu1
10.29.160.166 ubuntu2.cisco.com ubuntu2
10.29.160.167 ubuntu3.cisco.com ubuntu3
10.29.160.168 ubuntu4.cisco.com ubuntu4
10.29.160.169 ubuntu5.cisco.com ubuntu5
10.29.160.170 ubuntu6.cisco.com ubuntu6
10.29.160.171 ubuntu7.cisco.com ubuntu7
10.29.160.172 ubuntu8.cisco.com ubuntu8
10.29.160.173 ubuntu9.cisco.com ubuntu9
10.29.160.174 ubuntu10.cisco.com ubuntu10
10.29.160.175 ubuntu11.cisco.com ubuntu11
10.29.160.176 ubuntu12.cisco.com ubuntu12
10.29.160.177 ubuntu13.cisco.com ubuntu13
```

```
10.29.160.178 ubuntu14.cisco.com ubuntu14
10.29.160.179 ubuntu15.cisco.com ubuntu15
10.29.160.180 ubuntu16.cisco.com ubuntu16
10.29.160.181 ubuntu17.cisco.com ubuntu17
10.29.160.182 ubuntu18.cisco.com ubuntu18
10.29.160.183 ubuntu19.cisco.com ubuntu19
10.29.160.184 ubuntu20.cisco.com ubuntu20
...
10.29.160.228 ubuntu64.cisco.com ubuntu64
```

2. Deploy **/etc/hosts** from the admin node (ubuntu20) to all the nodes via the following pscp command:

```
clush –a –b –c /etc/hosts --dest=/home/ubuntu/
clush –a –b sudo cp /home/ubuntu/hosts /etc/
```

Domain name

For setting the domain name, follow these instructions:

```
clush –a –b sudo domainname cisco.com
Ensure hostname –f shows cisco.com or appropriate domain name set
```

NTP Configuration

The Network Time Protocol (NTP) is used to keep all the cluster nodes time-synchronized. The Network Time Protocol daemon (ntpd) sets and maintains the system time in sync with the timeserver located in the admin node (ubuntu20). Configuring NTP is critical for any Hadoop cluster because if the server clocks in the cluster go out of sync, serious problems will occur with HBase and other services.

Note Installing an internal NTP server keeps your cluster synchronized even when an outside NTP server is inaccessible.

```
Configure /etc/ntp.conf on the admin node with the following contents:
vi /etc/ntp.conf
driftfile /var/lib/ntp/drift
restrict 127.0.0.1
restrict -6 ::1
server 127.127.1.0
fudge 127.127.1.0 stratum 10
includefile /etc/ntp/crypto/pw
keys /etc/ntp/keys
```

Create `/home/ubuntu/ntp.conf` on the admin node and copy it to all nodes except admin node (Don't overwrite ntp.conf on admin node).

```
vi /root/ntp.conf
server 10.29.160.184    <- Admin-node ip/NTP server IP
driftfile /var/lib/ntp/drift
restrict 127.0.0.1
restrict -6 ::1
includefile /etc/ntp/crypto/pw
keys /etc/ntp/keys
```

Install ntp on all the nodes by running the following commands:

```
clush –b –a sudo apt-get –y install ntp

clush –b –a sudo service ntp status
---------------
```

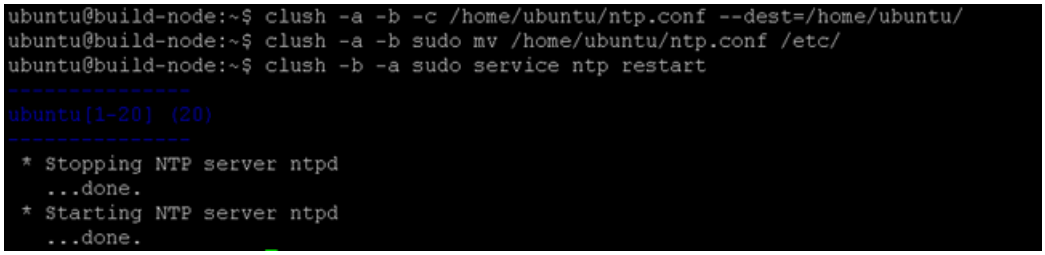
```
ubuntu[1-20] (20)
---------------
* NTP server is running
```

Copy ntp.conf file from the admin node to /etc of all the nodes by executing the following command in the admin node (ubuntu20).

```
clush –b –a –c /home/ubuntu/ntp.conf –-dest=/home/ubuntu
clush –b –w ubuntu[1-19,21-64] sudo mv /home/ubuntu/ntp.conf /etc/
```

Restart NTP on all the nodes including the admin node:

```
clush –b –a sudo service ntp restart
```



Disable IPTables

Disable IPTables (Firewall) by running the following command on the admin node.

```
clush –a –b sudo ufw disable
sudo ufw status
Status: inactive
```

Installing xfsprogs

Install xfsprogs on all the nodes for xfs filesystem.

```
clush –a –b sudo apt-get install xfsprogs
```

Configuring the Filesystem

In order to format the data partition which will be used by OpenStack for instances and data, ensure OS is on /dev/sda on all nodes and the data partition is on /dev/sdb by running the following commands

```
clush –a –b df –h
clush –a –b sudo cat /proc/partitions
```

On the Admin node, create a file containing the following script driveconf.sh

1. To create partition tables and file systems on the local disks supplied to each of the nodes, run the following script as the root user on each node.

```
vi /root/driveconf.sh
#!/bin/bash
X=/dev/sdb
Y=${X##*/}1
/sbin/parted  -s  ${X} mklabel gpt quit
/sbin/parted  -s  ${X} mkpart 1 6144s 100% quit
```

```
echo /sbin/mkfs.xfs -f -q -l size=65536b,lazy-count=1,su=256k -d
sunit=1024,swidth=6144 -r extsize=256k -L ${Y} ${X}1
/sbin/mkfs.xfs -f -q -l size=65536b,lazy-count=1,su=256k -d sunit=1024,swidth=6144
-r extsize=256k -L ${Y} ${X}1
(( $? )) && continue
/bin/mkdir  -p  /DATA/
(( $? )) && continue
/bin/mount  -t xfs  -o allocsize=128m,noatime,nobarrier,nodiratime  ${X}1  /DATA/
(( $? )) && continue
echo "LABEL=${Y} /DATA/ xfs allocsize=128m,noatime,nobarrier,nodiratime 0 0"
echo "LABEL=${Y} /DATA/ xfs allocsize=128m,noatime,nobarrier,nodiratime 0 0" >>
/etc/fstab
```

Note This script formats /dev/sdb which is considered as the data partition (single volume of all 12 4TB drives which is RAID5). If OS is on /dev/sdb, that will be wiped out.

2. Run the following command to copy `driveconf.sh` to all the datanodes.

```
clush -b -a -c /home/ubuntu/driveconf.sh --dest=/home/ubuntu
clush -b -a sudo chmod 755 /home/ubuntu/driveconf.sh
```

3. Run the following command from the admin node to run the script across all data nodes.

```
clush -a -b /home/ubuntu/driveconf.sh
```

OpenStack Pre-requisites

The following section provides information on setting up OpenStack.

Controller Node

In this CVD, the node ubuntu20 is used as the controller node. The controller node in OpenStack runs all OpenStack API services and OpenStack schedulers.

Networking

For an OpenStack production deployment, most of the nodes must have these network interface cards:

- One network interface card for external network traffic
- Another card to communicate with other OpenStack nodes.

In this CVD, eth0 is 10.29.160.x - public interface (with access to internet). eth1 is 192.168.10.x - private/internal interface for communication between VMs. We can add additional eth1 ip as follows:

Update /etc/network/interface on each host.

```
$ sudo cat /etc/network/interfaces
# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
```

```
auto eth0
iface eth0 inet static
    address 10.29.160.184
    netmask 255.255.255.0
    network 10.29.160.0
    broadcast 10.29.160.255
    gateway 10.29.160.1
auto eth1
iface eth1 inet dhcp

dns-search cisco.com
..
```

Run command “service networking restart” on all nodes as follows

```
clush –b –a sudo service networking restart
```

```
ubuntu@build-node:~$ clush -a -b sudo service networking restart
---------------
ubuntu[1-20] (20)
---------------
networking stop/waiting
networking start/running
```

```
clush -b -a "sudo ifconfig  | grep \"inet addr:\" "
```

Passwords

The various OpenStack services and the required software like the database and the Messaging server have to be password protected. These passwords are needed when configuring a service and then again to access the service. For more information, see:

http://docs.openstack.org/havana/install-guide/install/apt/content/basics-passwords.html

For simplicity, we use “ubuntu” as the password for all services in this CVD.

This guide uses the conventions, SERVICE_PASS as the password to access the service with SERVICE as the database name and SERVICE_DBPASS as the database password.

Table provides the list of passwords used in this CVD.

Table 7 List of Passwords

Passwords	Description
RABBIT_PASS	Password of user guest of RabbitMQ
KEYSTONE_DBPASS	Database password of Identity service
ADMIN_PASS	Password of user `admin`
GLANCE_DBPASS	Database password for Image Service
GLANCE_PASS	Password of Image Service user glance
NOVA_DBPASS	Database password for Compute service
NOVA_PASS	Password of Compute service user nova
DASH_DBPASS	Database password for the dashboard

Installing MySQL

OpenStack services require a database to store information. This CVD uses MySQL for database which will be installed on the controller node and all the other nodes need MySQL client software to be installed for accessing MySQL.

Install MySQL Server on the Controller Node

```
sudo apt-get installmysql-server python-mysqldb
```

```
ubuntu@ubuntu20:~$ sudo apt-get install mysql-server python-mysqldb
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18 libterm-readkey-perl mysql-client-5.5 mysql-cli
  mysql-server-5.5 mysql-server-core-5.5
Suggested packages:
  libclone-perl libmldbm-perl libnet-daemon-perl libplrpc-perl libsql-statement-perl libipc-sharedcache-perl tinyca ma
  python-mysqldb-dbg
The following NEW packages will be installed:
  libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18 libterm-readkey-perl mysql-client-5.5 mysql-cli
  mysql-server mysql-server-5.5 mysql-server-core-5.5 python-mysqldb
0 upgraded, 12 newly installed, 0 to remove and 100 not upgraded.
Need to get 26.8 MB of archives.
After this operation, 95.7 MB of additional disk space will be used.
Do you want to continue [Y/n]? Y
```

During the install, you will be prompted for the mysql root password. Enter a password of your choice and verify it.

Edit **/etc/mysql/my.cnf** and set the bind-address to the internal IP address of the controller, to enable access from outside the controller node.

```
[mysqld]
…
bind-address            = 10.29.160.184
```

```
ubuntu@ubuntu20:~$ cat /etc/mysql/my.cnf | grep bind
bind-address            = 10.29.160.184
```

Restart the MySQL service to apply the changes:

```
# sudo service mysql restart
```

```
ubuntu@ubuntu20:~$ sudo service mysql restart
mysql stop/waiting
mysql start/running, process 9746
```

Securing MySQL Server

All anonymous users that are created when the database is first started are to be deleted. Otherwise, database connection problems occur while following the CVD. To do this, use the **mysql_secure_installation** command.

```
# sudo mysql_install_db
# sudo mysql_secure_installation
```

```
ubuntu@ubuntu20:~$ sudo mysql_install_db
Installing MySQL system tables...
140411 12:28:06 [Warning] Using unique option prefix key_buffer instead of key_buffer_size
the full name instead.
OK
Filling help tables...
140411 12:28:06 [Warning] Using unique option prefix key_buffer instead of key_buffer_size
the full name instead.
OK

To start mysqld at boot time you have to copy
support-files/mysql.server to the right place for your system
ubuntu@ubuntu20:~$ sudo mysql_secure_installation



By default, a MySQL installation has an anonymous user, allowing anyone
to log into MySQL without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] Y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] Y
 ... Success!
```

This command presents a number of options to secure the database installation. Respond **yes** to all prompts unless there is a good reason to do otherwise.

Install MySql Client on all the Other Nodes

On all the nodes, install the MySQL client and the MySQL Python library on any of the systems that does not host a MySQL database:

```
# clush -b -a sudo apt-get -y install python-mysqldb
```

```
ubuntu@build-node:~$ clush -a -b sudo apt-get -y install python-mysqldb
---------------
ubuntu[1-20] (20)
---------------
Reading package lists...
Building dependency tree...
Reading state information...
python-mysqldb is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 100 not upgraded.
```

OpenStack Repository - Ubuntu Cloud Archive for Havana

1. Install the Ubuntu Cloud Archive for Havana:

```
# clush -a -b sudo apt-get -y install python-software-properties
# clush -a -b sudo add-apt-repository cloud-archive:havana

Ensure Havana repository is properly set by running command
```

```
$ sudo cat /etc/apt/sources.list.d/cloudarchive-havana.list
deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/havana main
deb-src http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/havana
main
```

```
ubuntu@build-node:~$ clush -a -b sudo apt-get -y install python-software-properties
---------------
ubuntu[1-20] (20)
---------------
Reading package lists...
Building dependency tree...
Reading state information...
python-software-properties is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 100 not upgraded.
ubuntu@build-node:~$ clush -a -b cat /etc/apt/sources.list.d/cloudarchive-havana.list
---------------
ubuntu[1-20] (20)
---------------
deb http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/havana main
deb-src http://ubuntu-cloud.archive.canonical.com/ubuntu precise-updates/havana main
```

2. Update the package database, upgrade your system, and reboot for all changes to take effect:

```
# clush -a -b uname -r
3.11.0-15-generic

# clush -a -b sudo apt-get -y update
# clush -a -b sudo apt-get -y dist-upgrade

clush -a -b ls /boot/initrd*
---------------
ubuntu[1-20] (20)
---------------
/boot/initrd.img-3.11.0-15-generic
/boot/initrd.img-3.11.0-19-generic
```

```
ubuntu@build-node:~$ clush -a -b ls /boot/initrd*
---------------
ubuntu[1-20] (20)
---------------
/boot/initrd.img-3.11.0-15-generic
/boot/initrd.img-3.11.0-19-generic
```

```
# clush -a -b sudo reboot

The above command upgrades kernel from 3.11.0-15 to 3.11.0-19

# clush -a -b uname -r
```

```
ubuntu@build-node:~$ clush -a -b uname -r
---------------
ubuntu[1-20] (20)
---------------
3.11.0-19-generic
```

Messaging server (RabbitMQ)

Install the messaging queue server RabbitMQ on the controller node.

```
# sudo apt-get -y install rabbitmq-server
```

Note The rabbitmq-server package configures the RabbitMQ service to start automatically and creates a `guest` user with a default `guest` password. This guide uses the `guest` account, and it is strongly advised to change its default password, especially if you have IPv6 available: by default the RabbitMQ server enables anyone to connect to it by using guest as login and password, and with IPv6, it is reachable from the outside.

To change the default guest password of RabbitMQ:

```
# sudo rabbitmqctl change_password guest RABBIT_PASS
# sudo rabbitmqctl change_password guest ubuntu
Changing password for user "guest"..
...done.
```

Install Apache on the Controller Node

To install Apache on the controller node, run the following command:

```
# sudo apt-get install apache2
```

OpenStack Services

OpenStack provides an Infrastructure as a Service (IaaS) solution through a set of interrelated services. Each service offers an Application Programming Interface (API) that facilitates this integration. Depending on the needs, some or all services can be installed.

Note For more information, see:
http://docs.openstack.org/havana/install-guide/install/apt/content/ch_overview.html

The following table describes the OpenStack services that make up the OpenStack architecture:

Table 8 OpenStack Services

Service	Project Name	Description
Dashboard	Horizon	Provides a web-based self-service portal to interact with underlying OpenStack services, such as launching an instance, assigning IP addresses and configuring access controls.
Compute	Nova	Manages the lifecycle of compute instances in an OpenStack environment. Responsibilities include spawning, scheduling and decomissioning of machines on demand.

Table 8 OpenStack Services

Service	Project Name	Description
Networking	Neutron	Enables network connectivity as a service for other OpenStack services, such as OpenStack Compute. Provides an API for users to define networks and the attachments into them. Has a pluggable architecture that supports many popular networking vendors and technologies.
Storage		
Object Storage	Swift	Stores and retrieves arbitrary unstructured data objects via a RESTful, HTTP based API. It is highly fault tolerant with its data replication and scale out architecture. Its implementation is not like a file server with mountable directories.
Block Storage	Cinder	Provides persistent block storage to running instances. Its pluggable driver architecture facilitates the creation and management of block storage devices.
Shared Services		
Identity Service	Keystone	Provides an authentication and authorization service for other OpenStack services. Provides a catalog of endpoints for all OpenStack services.
Image Service	Glance	Stores and retrieves virtual machine disk images. OpenStack Compute makes use of this during instance provisioning.
Telemetry	Ceilometer	Monitors and meters the OpenStack cloud for billing, benchmarking, scalability, and statistical purposes.

Table 8 OpenStack Services

Service	Project Name	Description
Higher-level Services		
Orchestration	Heat	Orchestrates multiple composite cloud applications by using either the native HOT template format or the AWS CloudFormation template format, through both an OpenStack-native REST API and a CloudFormation-compatible Query API.

Canonical Ubuntu OpenStack Havana Software Components

This CVD focuses on Canonical OpenStack software components based on the upstream "Havana" OpenStack release. Ubuntu is the popular Linux flavor to deploy OpenStack among Service Providers and Enterprise customers. Following few subsections cover key software components involved in OpenStack. For more information, see:

https://wiki.openstack.org/wiki/UnderstandingFlatNetworking#Multiple_nodes.2C_multiple_adapters

Identity Service ("Keystone")

This is a central authentication and authorization mechanism for all OpenStack users and services. It supports multiple forms of authentication including standard username and password credentials and it can also integrate with existing directory services such as LDAP.

Endpoints, Tenants, Tokens and Roles

The Identity service catalog lists all of the services deployed in an OpenStack cloud and manages authentication for them through endpoints. An endpoint is a network address where a service listens for requests. The Identity service provides each OpenStack service – such as Image, Compute, or Block Storage -- with one or more endpoints.

The Identity service uses tenants to group or isolate resources. By default users in one tenant can't access resources in another even if they reside within the same OpenStack cloud deployment or physical host. The Identity service issues tokens to authenticated users. The endpoints validate the token before allowing user access. User accounts are associated with roles that define their access credentials. Multiple users can share the same role within a tenant.

Image Service ("Glance")

This service discovers, registers, and delivers virtual machine images. They can be copied via snapshot and immediately stored as the basis for new instance deployments. Stored images allow OpenStack users and administrators to provision multiple servers quickly and consistently.

By default the Image Service stores images in the /var/lib/glance/images directory of the local server's filesystem where Glance is installed.

Compute Service ("Nova")

OpenStack Compute provisions and manages large networks of virtual machines. It is the backbone of OpenStack's IaaS functionality. OpenStack Compute scales horizontally on standard hardware enabling the favorable economics of cloud computing. Users and administrators interact with the compute fabric via a web interface and command line tools.

OpenStack Compute provides distributed and asynchronous architecture, allowing scale out fault tolerance for virtual machine instance management.

OpenStack Compute is composed of many services that work together to provide the full functionality. The openstack-nova-cert and openstack-nova-consoleauth services handle authorization. The openstack-nova-api responds to service requests and the openstack-nova-scheduler dispatches the requests to the message queue. The openstack-nova-conductor service updates the state database, which limits direct access to the state database by compute nodes for increased security. The openstacknova-compute service creates and terminates virtual machine instances on the compute nodes. Finally, openstack-nova-novncproxy provides a VNC proxy for console access to virtual machines via a standard web browser.

Ephemeral Storage

Deploying only the OpenStack Compute Service (nova), users do not have access to any form of persistent storage by default. The disks associated with VMs are "Ephemeral", meaning that (from the user's point of view) they effectively disappear when a virtual machine is terminated (when VM is deleted).

Networking (nova-network)

Networking for the VMs in this CVD employs nova-network (Legacy OpenStack Network), which is a FlatNetwork. FlatNetworking primarily involves compute nodes. It uses Ethernet adapters configured as bridges to allow network traffic to transit between all the various nodes. This setup can be done with a single adapter on the physical host, or multiple.

For more information, see:
https://wiki.openstack.org/wiki/UnderstandingFlatNetworking#Multiple_nodes.2C_multiple_adapters

Dashboard ("Horizon")

The OpenStack Dashboard is an extensible web-based application that allows cloud administrators and users to control and provision compute, storage, and networking resources. Administrators can use the Dashboard to view the state of the cloud, create users, assign them to tenants, and set resource limits.

The OpenStack Dashboard runs as an Apache HTTP server via the httpd service.

Note Both the Dashboard and command line tools can be used to manage an OpenStack environment. This CVD focuses on the command line tools because they offer more granular control and insight into the OpenStack functionality.

Installation of OpenStack

OpenStack basically has three roles for the nodes underneath.

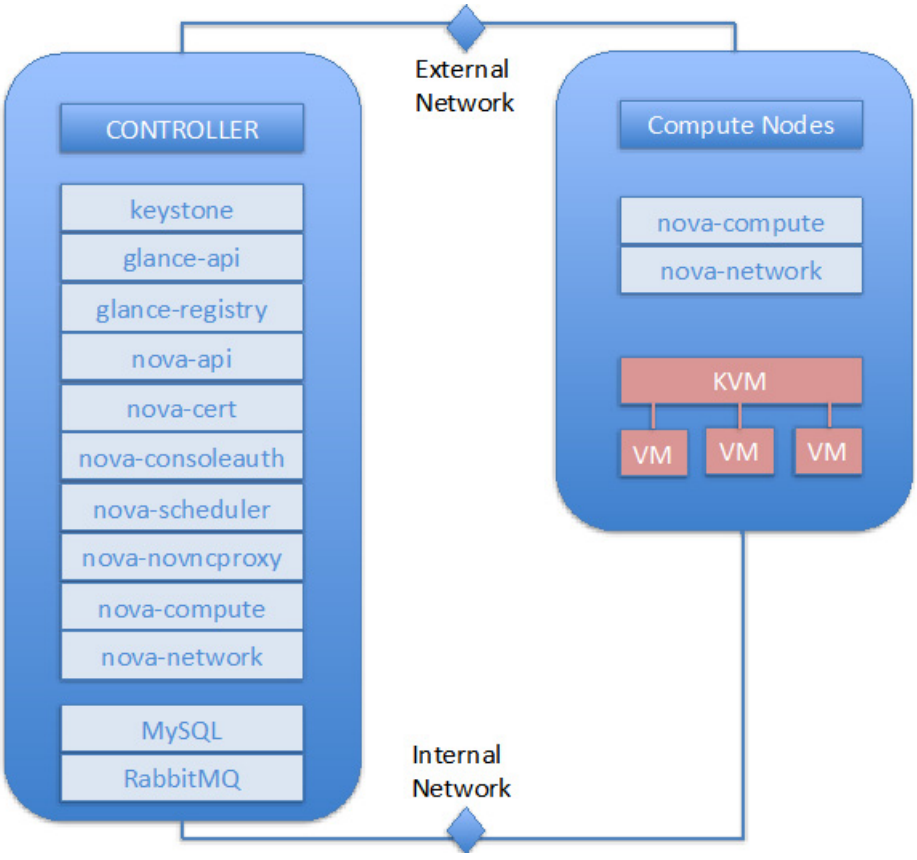
- Controller node – It is the main management for OpenStack which controls compute and storage node. This runs all OpenStack API Services and OpenStack schedulers.
- Compute node – These nodes are hosts to the VMs spawned.
- Storage node – These nodes hosts the storage for VMs.

Note In the architecture discussed in this CVD, the storage is Ephemeral, which is local to VMs. Hence compute nodes do the job of storage nodes as well and there are no separate storage nodes.

Following OpenStack services are run on different components.

- Controller node - Identity Service (keystone), Image Service (Glance), Dashboard (Horizon), and management portion of Compute, with the associated API services, MySQL databases, and messaging system. In this CVD, we have Nova compute and Nova network also running on the controller node.
- Compute node - Tenant virtual machines on a hypervisor. By default, Nova compute uses KVM as the hypervisor. Compute also provisions and operates tenant networks and implements security groups.

Figure 89 Controller and Compute Nodes



Installing Havana OpenStack Services

This section details the installation of Havana OpenStack components on the cluster.

Identity Service (Keystone)

The Identity Service is used for the following functions:

- User management: To manage User Credentials.
- Service catalog: Provides available services along with their API endpoints.

Identity Service definitions:

- **User** - Digital representation of a person, system, or service who uses OpenStack cloud services. Users have a login and may be assigned tokens to access resources. Users can be directly assigned to a particular tenant and behave as if they are contained in that tenant.
- **Token** - An arbitrary bit of text that is used to access resources. Each token has a scope which describes which resources are accessible with it. A token may be revoked at any time and is valid for a finite duration.
- **Tenant** - A container used to group or isolate resources and/or identity objects. Depending on the service operator, a tenant may map to a customer, account, organization, or project.
- **Service** - An OpenStack service such as Compute (Nova), or Image Service (Glance). Provides one or more endpoints through which users can access resources and perform operations.
- **Endpoint** - A network-accessible address, usually described by a URL, from where you access a service.
- **Role** - A role includes a set of rights and privileges. A user assuming that role inherits those rights and privileges. In the Identity Service, a token that is issued to a user includes the list of roles that user has. Services that are being called by that user determine how they interpret the set of roles a user has and to which operations or resources each role grants access.

Note For more information on Identity Service, see Install the Identity Service (keystone).

1. Install the OpenStack Identity Service on the controller node. The following command will also install python-keystoneclient (which is a dependency):

```
# sudo apt-get –y install keystone
```

2. The Identity Service uses MySQL to store information. Specify the location of the database in the configuration file. In this CVD we use MySQL database on the controller node (ubuntu20 in this CVD) with the username **keystone**. Replace **KEYSTONE_DBPASS** with a suitable password for the database user. As mentioned earlier for simplicity, we use “ubuntu” as the password for all services in this CVD. Replace controller with the hostname of the controller.

```
Edit /etc/keystone/keystone.conf and change the [sql] section.
$ sudo less /etc/keystone/keystone.conf | grep sql
[sql]
connection = mysql://keystone:<KEYSTONE_DBPASS>@<controller>/keystone
```

```
ubuntu@ubuntu20:~$ sudo less /etc/keystone/keystone.conf | grep sql
[sql]
connection = mysql://keystone:ubuntu@ubuntu20/keystone
```

3. Delete the default sqllite database keystone.db file created in the /var/lib/keystone/ directory.

```
sudo rm –f /var/lib/keystone/keystone.db
```

4. Login to MySQL as root. Create a keystone database user:

```
# mysql -u root -p
mysql> CREATE DATABASE keystone;
mysql> GRANT ALL PRIVILEGES ON keystone.* TO \ 'keystone'@'localhost'  IDENTIFIED
BY 'KEYSTONE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \ IDENTIFIED BY
'KEYSTONE_DBPASS';
```

```
mysql> CREATE DATABASE keystone;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
    -> IDENTIFIED BY 'ubuntu';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' \
    ->
    -> IDENTIFIED BY 'ubuntu';
Query OK, 0 rows affected (0.00 sec)
```

5. Create the database tables for the Identity Service:

```
# sudo keystone-manage db_sync
```

6. Define an authorization token to use as a shared secret between the Identity Service and other OpenStack services. Use **openssl** to generate a random token and store it in the configuration file:

```
# openssl rand -hex 10
```

```
ubuntu@ubuntu20:~$ openssl rand -hex 10
8c2a6ed462e6429ca745
```

```
Edit /etc/keystone/keystone.conf and change the [DEFAULT] section,
replacing ADMIN_TOKEN with the results of the command.
[DEFAULT]
# A "shared secret" between keystone and other openstack services
admin_token = ADMIN_TOKEN
```

Note Make sure there are no trailing spaces before admin_token.

```
ubuntu@ubuntu20:~$ sudo head /etc/keystone/keystone.conf
[DEFAULT]
# A "shared secret" between keystone and other openstack services
 admin_token = 8c2a6ed462e6429ca745

# The IP address of the network interface to listen on
# bind_host = 0.0.0.0

# The port number which the public service listens on
# public_port = 5000
```

7. Restart the Identity Service:

```
# sudo service keystone restart
```

Define Users, Tenants, and Roles

After you install the Identity Service, set up users, tenants, and roles to authenticate against. These are used to allow access to services and endpoints.

At this point, we have not created any users, so we have to use the authorization token created in an earlier step. We have set OS_SERVICE_TOKEN, as well as OS_SERVICE_ENDPOINT to specify where the Identity Service is running. Replace ADMIN_TOKEN with your authorization token above.

```
# export OS_SERVICE_TOKEN=<ADMIN_TOKEN>
# export OS_SERVICE_ENDPOINT=http://<controller>:35357/v2.0
```

```
ubuntu@ubuntu20:~$ export OS_SERVICE_TOKEN=8c2a6ed462e6429ca745
ubuntu@ubuntu20:~$ export OS_SERVICE_ENDPOINT=http://ubuntu20:35357/v2.0
```

First, create a tenant for an administrative user and a tenant for other OpenStack services to use.

```
# keystone tenant-create --name=admin --description="Admin Tenant"
# keystone tenant-create --name=service --description="Service Tenant"
```

```
ubuntu@ubuntu20:~$ sudo keystone tenant-create --name=admin --description="Admin Tenant"
Expecting an auth URL via either --os-auth-url or env[OS_AUTH_URL]
```

Note Ensure to run the above commands without “sudo”, else it throws error as above as the export values are created for a user Ubuntu and not “sudo/root”

Next, create an administrative user called admin. Choose a password for the admin user and specify an email address for the account.

```
# keystone user-create --name=admin --pass=ADMIN_PASS \
-email=admin@cisco.com
```

```
ubuntu@ubuntu20:~$ keystone user-create --name=admin --pass=ubuntu --email=admin@cisco.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |         admin@cisco.com          |
| enabled  |               True               |
|    id    | f33bd76628704fc9976c9a4f4d7bba76 |
|   name   |              admin               |
+----------+----------------------------------+
ubuntu@ubuntu20:~$ keystone tenant-create --name=admin --description="Admin Tenant"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           Admin Tenant           |
|   enabled   |               True               |
|      id     | a688d1fba0cb4adc96f9051fff53e5b5 |
|     name    |              admin               |
+-------------+----------------------------------+
ubuntu@ubuntu20:~$ keystone tenant-create --name=service --description="Service Tenant"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |          Service Tenant          |
|   enabled   |               True               |
|      id     | 79599c252d9b40c6b1868dd02a594335 |
|     name    |             service              |
+-------------+----------------------------------+
```

Create a role for administrative tasks called admin. Any roles you create should map to roles specified in the policy.json files of the various OpenStack services. The default policy files use the admin role to allow access to most services.

```
# keystone role-create --name=admin
```

```
ubuntu@ubuntu20:~$ keystone role-create --name=admin
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 7031b648d5d64019bb42c54e31b75f54 |
|   name   |              admin               |
+----------+----------------------------------+
```

Finally, add roles to users. Users always log in with a tenant, and roles are assigned to users within tenants. Add the `admin` role to the `admin` user when logging in with the `admin` tenant.

```
# keystone user-role-add --user=admin --tenant=admin --role=admin
```

Define services and API endpoints

For Identity Service to track which OpenStack services are installed and where they are located on the network, services must be registered in the OpenStack installation. To register a service, run these commands:

- **keystone service-create -** Describes the service.
- **keystone endpoint-create** - Associates API endpoints with the service.

Identity Service itself must be registered. Use the **OS_SERVICE_TOKEN** environment variable, as set previously, for authentication.

1. Create a service entry for the Identity Service:

```
# keystone service-create --name=keystone --type=identity \
--description="Keystone Identity Service"
```

The service ID is randomly generated and is different from the one shown below.

```
ubuntu@ubuntu20:~$ keystone service-create --name=keystone --type=identity \
> --description="Keystone Identity Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |    Keystone Identity Service     |
|      id     | 644669f68c584cd5a20ac50d67cf8592 |
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
```

2. Specify an API endpoint for the Identity Service by using the returned service ID. When you specify an endpoint, you provide URLs for the public API, internal API, and admin API. In this guide, the **controller** host name is used. Note that the Identity Service uses a different port for the admin API.

```
# keystone endpoint-create \
--service-id=<the_service_id_above> \
--publicurl=http://<controller>:5000/v2.0 \
--internalurl=http://<controller>:5000/v2.0 \
--adminurl=http://<controller>:35357/v2.0
```

```
ubuntu@ubuntu20:~$ keystone service-create --name=keystone --type=identity \
> --description="Keystone Identity Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |    Keystone Identity Service     |
|      id     | 644669f68c584cd5a20ac50d67cf8592 |
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
```

3. As services are added to the OpenStack installation, call these commands to register the services with the Identity Service.

Verify the Identity Service installation

To verify the Identity Service is installed and configured correctly, first unset the **OS_SERVICE_TOKEN** and **OS_SERVICE_ENDPOINT** environment variables. These were only used to bootstrap the administrative user and register the Identity Service.

```
$ unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```

Now use the regular username-based authentication. Request an authentication token using the admin user and the password you chose during the earlier administrative user-creation step.

```
$ keystone --os-username=admin --os-password=ADMIN_PASS \
--os-auth-url=http://<controller>:35357/v2.0 token-get
```

```
ubuntu@ubuntu20:~$ keystone endpoint-create --service-id=644669f68c584cd5a20ac50d67cf8592
> --publicurl=http://ubuntu20:5000/v2.0 \
> --internalurl=http://ubuntu20:5000/v2.0 \
> --adminurl=http://ubuntu20:35357/v2.0
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |    http://ubuntu20:35357/v2.0    |
|      id     | d104d96ce0414f7e8fbd8ae589cf1e1d |
| internalurl |    http://ubuntu20:5000/v2.0     |
|  publicurl  |    http://ubuntu20:5000/v2.0     |
|    region   |            regionOne             |
|  service_id | 644669f68c584cd5a20ac50d67cf8592 |
+-------------+----------------------------------+
```

You should receive a token in response, paired with your user ID. This verifies that keystone is running on the expected endpoint, and that the user account is established with the expected credentials.

Next, verify that authorization is behaving as expected by requesting authorization on a tenant.

```
$ keystone --os-username=admin --os-password=ADMIN_PASS \
   --os-tenant-name=admin --os-auth-url=http://<controller>:35357/v2.0 token-get
```

```
ubuntu@ubuntu20:~$ keystone --os-username=admin --os-password=ubuntu \
> --os-auth-url=http://ubuntu20:35357/v2.0 token-get
+----------+-----------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
| Property |
```

You should receive a new token in response, this time including the ID of the tenant you specified. This verifies that your user account has an explicitly defined role on the specified tenant, and that the tenant exists as expected.

You can also set your **--os-*** variables in your environment to simplify command-line usage. Set up an **openrc.sh** file with the admin credentials and admin endpoint.

```
#cat ~/openrc.sh
export OS_USERNAME=admin
export OS_PASSWORD=ADMIN_PASS
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://<controller>:35357/v2.0
```

```
ubuntu@ubuntu20:~$ cat ~/openrc.sh
export OS_USERNAME=admin
export OS_PASSWORD=ubuntu
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://ubuntu20:35357/v2.0
```

Source this file to read in the environment variables.

```
$ source openrc.sh
```

Verify that **openrc.sh** file is configured correctly by performing the same command as above, but without the **--os-*** arguments.

```
$ keystone token-get
```

The command returns a token and the ID of the specified tenant. This verifies that you have configured your environment variables correctly.

Finally, verify that your admin account has authorization to perform administrative commands.

```
$ keystone user-list
+----------------------------------+---------+--------------------+---------+ |
id                | enabled | email              |  name  |
+----------------------------------+---------+--------------------+---------+
| a4c2d43f80a549a19864c89d759bb3fe | True    | admin@example.com  | admin  |
```

This verifies that the user account has the **admin** role, which matches the role used in the Identity Service **policy.json** file.

Image Service (Glance)

The OpenStack Image Service enables users to discover, register, and retrieve virtual machine images. Users can add new images or take a snapshot of an image from an existing server for immediate storage. Use snapshots for back up and as templates to launch new servers. Virtual machine images made available through the Image Service can be stored in a variety of locations from simple file systems to object-storage systems like OpenStack Object Storage.

Note This guide configures the Image Service to use the file backend. By default this directory is in **/var/lib/glance/images/;** however, this will be changed.

The Image Service includes the following components:

- **glance-api** - Accepts Image API calls for image discovery, retrieval, and storage.
- **glance-registry -** Stores, processes, and retrieves metadata about images. Metadata includes size, type, and so on.
- Database - Stores image metadata. Database used can be changed depending on the preference. Most deployments use MySQL or SQlite.
- Storage repository for image files - The Image Service supports normal file systems, RADOS block devices, Amazon S3, and HTTP.

A number of periodic processes run on the Image Service to support caching. The Image Service is central to the overall IaaS picture. It accepts API requests for images or image metadata from end users or Compute components.

Install the Image Service

This section details installing Glance, the image service in OpenStack Havana, on the controller node.

1. Install the Image Service on the controller node:

   ```
   # apt-get install glance python-glanceclient
   ```

2. The Image Service stores information about images in a database. The examples in this guide use the MySQL database that is used by other OpenStack services.

Configure the location of the database. The Image Service provides the **glance-api** and **glance-registry** services, each with its own configuration file. Both the services configuration files would be updated throughout this section. Replace **GLANCE_DBPASS** with your Image Service database password.

Edit **/etc/glance/glance-api.conf** and **/etc/glance/glance-registry.conf** and change the **[DEFAULT]** section.

```
...
[DEFAULT]
...
# SQLAlchemy connection string for the reference implementation
# registry server. Any valid SQLAlchemy connection string is fine.
#See:http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/connections.html#sqlal
chemy.create_engine
sql_connection = mysql://glance:GLANCE_DBPASS@controller/glance
```

```
ubuntu@ubuntu20:~$ sudo cat /etc/glance/glance-api.conf | grep sql_connection
sql_connection = mysql://glance:ubuntu@ubuntu20/glance
ubuntu@ubuntu20:~$ sudo cat /etc/glance/glance-registry.conf | grep sql_connection
sql_connection = mysql://glance:ubuntu@ubuntu20/glance
```

3. By default, the Ubuntu packages create an SQLite database. Delete the **glance.sqlite** file created in the **/var/lib/glance/** directory so that it does not get used by mistake.

```
# sudo rm -f /var/lib/glance/glance.sqlite
```

4. Use the password you created to log in as root and create a `glance` database user:

```
# mysql -u root -p
mysql> CREATE DATABASE glance;
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \ IDENTIFIED BY
'GLANCE_DBPASS';
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \ IDENTIFIED BY
'GLANCE_DBPASS';
```

```
mysql>
mysql>
mysql> CREATE DATABASE glance;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
    -> IDENTIFIED BY 'ubuntu';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' \
    -> IDENTIFIED BY 'ubuntu';
Query OK, 0 rows affected (0.00 sec)
```

5. Create the database tables for the Image Service:

```
# sudo glance-manage db_sync
```

Note Make sure the command is run with sudo here, else tables in database glance will not be created.

6. Create a glance user that the Image Service can use to authenticate with the Identity Service. Choose a password and specify an email address for the glance user. Use the service tenant and give the user the **admin** role.

```
# keystone user-create --name=glance --pass=GLANCE_PASS \
--email=admin@cisco.com
# keystone user-role-add --user=glance --tenant=service --role=admin
```

```
ubuntu@ubuntu20:~$ keystone user-create --name=glance --pass=ubuntu \
>
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |                                  |
| enabled  |               True               |
|    id    | 4acb00c747ec444182058b31f75ed096 |
|   name   |              glance              |
+----------+----------------------------------+
```

7. Configure the Image Service to use the Identity Service for authentication.

   Edit the **/etc/glance/glance-api.conf** and **/etc/glance/glance-registry.conf** files. Replace **GLANCE_PASS** with the password you chose for the **glance** user in the Identity Service.

   a. Add the following keys under the **[keystone_authtoken]** section:

   ```
   [keystone_authtoken]
   ...
   auth_uri = http://controller:5000
   auth_host = controller
   auth_port = 35357
   auth_protocol = http
   admin_tenant_name = service
   admin_user = glance
   admin_password = GLANCE_PASS
   ```

   b. Add the following key under the **[paste_deploy]** section:

   ```
   [paste_deploy]
   ...
   flavor = keystone
   ```

```
ubuntu@ubuntu20:~$ sudo grep -A8 keystone_authtoken /etc/glance/glance-api.conf
[keystone_authtoken]
auth_uri = http://ubuntu20:5000
auth_host = ubuntu20
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = ubuntu
```

```
ubuntu@ubuntu20:~$ sudo grep -A18 keystone_authtoken /etc/glance/glance-registry.co
[keystone_authtoken]
auth_uri = http://ubuntu20:5000
auth_host = ubuntu20
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = glance
admin_password = ubuntu

[paste_deploy]
# Name of the paste configuration file that defines the available pipelines
#config_file = glance-registry-paste.ini

# Partial name of a pipeline in your paste configuration file with the
# service name removed. For example, if your paste section name is
# [pipeline:glance-registry-keystone], you would configure the flavor below
# as 'keystone'.
flavor=keystone
```

8. Add the credentials to the **/etc/glance/glance-api-paste.ini** and **/etc/glance/glance-registry-paste.ini files**.

   Edit each file to set the following options in the **[filter:authtoken]** section and leave any other existing option as it is.

```
[filter:authtoken]
paste.filter_factory=keystoneclient.middleware.auth_token:filter_factory
auth_host=controller
admin_user=glance
admin_tenant_name=service
admin_password=GLANCE_PASS
```

```
ubuntu@ubuntu20:~$ sudo grep -A7 filter:authtoken /etc/glance/glance-api-paste.ini
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
delay_auth_decision = true
auth_host=ubuntu20
admin_user=glance
admin_tenant_name=service
admin_password=ubuntu
ubuntu@ubuntu20:~$ sudo grep -A7 filter:authtoken /etc/glance/glance-registry-paste.ini
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host=ubuntu20
admin_user=glance
admin_tenant_name=service
admin_password=ubuntu
```

9. Register the Image Service with the Identity Service so that other OpenStack services can locate it. Register the service and create the endpoint:

```
# keystone service-create --name=glance --type=image \
--description="Glance Image Service"
```

```
ubuntu@ubuntu20:~$ keystone service-create --name=glance --type=image \
> --description="Glance Image Service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Glance Image Service       |
|      id     | e9e678a4fe4e4ecb872e9e79f0fc30f0 |
|     name    |              glance              |
|     type    |              image               |
+-------------+----------------------------------+
```

10. Use the **id** property returned for the service to create the endpoint:

```
# keystone endpoint-create \
--service-id=<the_service_id_above> \
--publicurl=http://<controller>:9292 \
--internalurl=http://<controller>:9292 \
--adminurl=http://<controller>:9292
```

```
ubuntu@ubuntu20:~$ keystone endpoint-create \
> --service-id=e9e678a4fe4e4ecb872e9e79f0fc30f0 \
> --publicurl=http://ubuntu20:9292 \
> --internalurl=http://ubuntu20:9292 \
> --adminurl=http://ubuntu20:9292
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |       http://ubuntu20:9292       |
|      id     | b4140feb219746659e345c95ff2202de |
| internalurl |       http://ubuntu20:9292       |
|  publicurl  |       http://ubuntu20:9292       |
|    region   |            regionOne             |
|  service_id | e9e678a4fe4e4ecb872e9e79f0fc30f0 |
+-------------+----------------------------------+
```

11. Restart the **glance service** with the new settings.

```
# service glance-registry restart
# service glance-api restart
```

```
ubuntu@ubuntu20:~$ sudo service glance-registry restart
glance-registry stop/waiting
glance-registry start/running, process 8578
ubuntu@ubuntu20:~$ sudo service glance-api restart
glance-api stop/waiting
glance-api start/running, process 8589
```

Verify the Image Service installation

To test the Image Service installation, download at least one virtual machine image that is known to work with OpenStack. For example, CirrOS is a small test image that is often used for testing OpenStack deployments (CirrOS downloads). This walk through uses the 64-bit CirrOS QCOW2 image.

CirrOS (test) images

CirrOS is a minimal Linux distribution that was designed for use as a test image on clouds such as OpenStack Compute.

For KVM deployment, images are recommended to be in qcow2 format. The most recent 64-bit qcow2 image as of this writing is cirros-0.3.2-x86_64-disk.img

Note In a CirrOS image, the login account is cirros. The password is cubswin:)

Official Ubuntu images

Canonical maintains an official set of Ubuntu-based images.

Images are arranged by Ubuntu release, and by image release date, with "current" being the most recent. For example, the page that contains the most recently built image for Ubuntu 12.04 "Precise Pangolin" is http://cloud-images.ubuntu.com/precise/current/. At the bottom of the page are links to images that can be downloaded directly.

For KVM deployment, images are recommended to be in qcow2 format. The most recent version of the 64-bit QCOW2 image for Ubuntu 12.04 is precise-server-cloudimg-amd64-disk1.img.

Note

- In a ubuntu image, the login account is ubuntu.
- See http://docs.openstack.org/image-guide/content/ch_obtaining_images.html for links to obtain Virtual machine images of different Operating Systems

In this section we will set up the Ubuntu virtual machine image precise-server-cloudimg-amd64-disk1.img for Glance.

For more information about how to download and build images, see OpenStack Virtual Machine Image Guide. For information about how to manage images, see the OpenStack User Guide.

1. Download the image into a dedicated directory on the Controller node using **wget** or **curl**:

```
$ mkdir images
$ cd images/
$ wget
http://uec-images.ubuntu.com/precise/current/precise-server-cloudimg-amd64-disk1.i
mg
$ wget http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img
```

```
buntu@ubuntu20:~$ mkdir images
buntu@ubuntu20:~$ cd images/
buntu@ubuntu20:~/images$ wget http://uec-images.ubuntu.com/precise/current/precise-server-cloudimg-amd64-disk1.img
-2014-04-14 14:59:48--  http://uec-images.ubuntu.com/precise/current/precise-server-cloudimg-amd64-disk1.img
esolving uec-images.ubuntu.com (uec-images.ubuntu.com)... 91.189.88.140
onnecting to uec-images.ubuntu.com (uec-images.ubuntu.com)|91.189.88.140|:80... connected.
TTP request sent, awaiting response... 200 OK
ength: 260112896 (248M) [application/octet-stream]
aving to: `precise-server-cloudimg-amd64-disk1.img'
```

2. Upload the image to the Image Service:

```
# glance image-create --name=<imageLabel> \
--disk-format=<fileFormat> \
--container-format=<containerFormat> --is-public=<accessValue> \
<imageFile>
```

Where:

imageLabel is the arbitrary label, by which users refer to the image.

fileFormat specifies the format of the image file. Valid formats include qcow2, raw, vhd, vmdk, vdi, iso, aki, ari, and ami.

You can verify the format using the file command:

```
$ file precise-server-cloudimg-amd64-disk1.img
precise-server-cloudimg-amd64-disk1.img: QEMU QCOW Image (v2), 41126400 bytes
```

```
ubuntu@ubuntu20:~/images$ file precise-server-cloudimg-amd64-disk1.img
precise-server-cloudimg-amd64-disk1.img: QEMU QCOW Image (v2), 2361393152 bytes
```

containerFormat specifies the container format. Valid formats include: bare, ovf, aki, ari and ami.

Specify bare to indicate that the image file is not in a file format that contains metadata about the virtual machine. Although this field is currently required, it is not actually used by any of the OpenStack services and has no effect on system behavior. Because the value is not used anywhere, it safe to always specify bare as the container format.

accessValue specifies image access:

- true - All users can view and use the image.
- false - Only administrators can view and use the image.

imageFile specifies the name of your downloaded image file.

For example:

```
#glance image-create --name="Ubuntu-Precise" --disk-format=qcow2 \
--container-format=bare --is-public=true < precise-server-cloudimg-amd64-disk1.img
```

```
ubuntu@ubuntu20:~/images$ sudo glance-manage db_sync
ubuntu@ubuntu20:~/images$ glance image-create --name="Ubuntu-Precise" --disk-format=qcow2 --container-format=bare --is-public=true
< precise-server-cloudimg-amd64-disk1.img
+------------------+--------------------------------------+
| Property         | Value                                |
+------------------+--------------------------------------+
| checksum         | 034618426bcd7c80528775098a086982     |
| container_format | bare                                 |
| created_at       | 2014-04-14T22:36:10                  |
| deleted          | False                                |
| deleted_at       | None                                 |
| disk_format      | qcow2                                |
| id               | 93381385-53d5-4791-8552-905a79597523 |
| is_public        | True                                 |
| min_disk         | 0                                    |
| min_ram          | 0                                    |
| name             | Ubuntu-Precise                       |
| owner            | a688d1fba0cb4adc96f9051fff53e5b5     |
| protected        | False                                |
| size             | 260112896                            |
| status           | active                               |
| updated_at       | 2014-04-14T22:36:11                  |
+------------------+--------------------------------------+
```

or for RHEL

```
glance image-create --name="RHEL 6.4" --disk-format=qcow2 \
--container-format=bare --is-public=true <
rhel-server-x86_64-kvm-6.4_20130130.0-9-sda.qcow2
```

3. Confirm that the image was uploaded and display its attributes:

```
# glance image-list
# nova image-list
```

```
ubuntu@ubuntu20:~/images$ glance image-list
+--------------------------------------+---------------+-------------+------------------+----------+--------+
| ID                                   | Name          | Disk Format | Container Format | Size     | Status |
+--------------------------------------+---------------+-------------+------------------+----------+--------+
| 93381385-53d5-4791-8552-905a79597523 | Ubuntu-Precise | qcow2       | bare             | 260112896 | active |
+--------------------------------------+---------------+-------------+------------------+----------+--------+
```

Compute service (Nova)

The Compute service is a cloud computing fabric controller, which is the main part of an IaaS system. It is a collection of services that enable you to launch virtual machine instances and use it to host and manage cloud-computing systems.

Compute interacts with the Identity Service for authentication, Image Service for images, and the Dashboard for the user and administrative interface. Access to images is limited by project and by user; quotas are limited per project (for example, the number of instances). The Compute service scales horizontally on standard hardware, and downloads images to launch instances as required.

The Compute Service is made up of the following functional areas and their underlying components:

API

- **nova-api service** - Accepts and responds to end user compute API calls. Supports the OpenStack Compute API, the Amazon EC2 API, and a special Admin API for privileged users to perform administrative actions. Also, initiates most orchestration activities, such as running an instance, and enforces some policies.
- **nova-api-metadata service** - Accepts metadata requests from instances. The nova-api-metadata service is generally used only when run in a multi-host mode with nova-network installations.

Compute core

- **nova-compute process** - A worker daemon that creates and terminates virtual machine instances through hypervisor APIs. For example, XenAPI for XenServer/XCP, libvirt for KVM or QEMU, VMwareAPI for VMware, and others.
- **nova-scheduler process** - Takes a virtual machine instance request from the queue and determines on which compute server host it should run.
- **nova-conductor module** - Mediates interactions between nova-compute and the database. Aims to eliminate direct accesses to the cloud database made by nova-compute. The nova-conductor module scales horizontally.

Networking for VMs

- **nova-network worker daemon** - Similar to nova-compute, it accepts networking tasks from the queue and performs tasks to manipulate the network, such as setting up bridging interfaces or changing iptables rules.
- **nova-dhcpbridge script** - Tracks IP address leases and record them in the database by using the dnsmasq dhcp-script facility.

Console interface

- **nova-consoleauth daemon** - Authorizes tokens for users that console proxies provide.
- **nova-novncproxy daemon** - Provides a proxy for accessing running instances through a VNC connection. Supports browser-based novnc clients.
- **nova-xvpnvncproxy daemon** - A proxy for accessing running instances through a VNC connection.
- **nova-cert daemon** - Manages x509 certificates.

Other components

- **The queue** - A central hub for passing messages between daemons. Usually implemented with RabbitMQ, but could be any AMPQ message queue, such as Apache Qpid.
- **SQL database** - Stores most buildtime and runtime states for a cloud infrastructure. Includes available instance types, instances, networks, and projects. The databases widely used are MySQL (as in this CVD), and PostgreSQL.

The Compute Service interacts with other OpenStack services: Identity Service for authentication, Image Service for images, and the OpenStack dashboard for a web interface.

Install Compute controller services

These services can be configured to run on separate nodes or the same node. In this CVD, most services run on the controller node and the service that launches virtual machines runs on a dedicated compute node. This section shows how to install and configure these services on the controller node. Controller also runs nova-compute in order to run a single VM that runs Hadoop Master services such as Namenode. This section details on the services to be installed on the controller node.

1. Install these compute packages, which provide the compute services that run on the controller node.

```
# sudo apt-get -y install nova-novncproxy novnc nova-api \
nova-ajax-console-proxy nova-cert nova-conductor \
nova-consoleauth nova-doc nova-scheduler \
python-novaclient
```

2. Compute stores information in a database. The examples in this guide use the MySQL database that is used by other OpenStack services. Configure the location of the database. Replace NOVA_DBPASS with your compute service password:

   Edit **/etc/nova/nova.conf** file and add these lines to the [database] and [keystone_authtoken] sections:

```
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:NOVA_DBPASS@controller/nova
[keystone_authtoken]
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS
```

3. Configure the compute service to use the **RabbitMQ** message broker by setting these configuration keys in the [DEFAULT] configuration group of the **/etc/nova/nova.conf** file:

```
rpc_backend = nova.rpc.impl_kombu
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

```
ubuntu@ubuntu20:~$ sudo tail -13 /etc/nova/nova.conf
rpc_backend = nova.rpc.impl_kombu
rabbit_host = ubuntu20
rabbit_password = ubuntu
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:ubuntu@ubuntu20/nova
[keystone_authtoken]
auth_host = ubuntu20
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = ubuntu
```

4. By default, the Ubuntu packages create an SQLite database. Delete the **nova.sqlite file** created in the **/var/lib/nova/** directory so that it does not get used by mistake.

```
sudo rm -f /var/lib/nova/nova.sqlite
```

5. Use the password you created previously to log in as root. Create a **nova** database user:

```
# mysql -u root -p
mysql> CREATE DATABASE nova;
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY
'NOVA_DBPASS';
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' \
IDENTIFIED BY 'NOVA_DBPASS';
```

```
mysql> clear
mysql> CREATE DATABASE nova;
Query OK, 1 row affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY 'ubuntu';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY 'ubuntu';
Query OK, 0 rows affected (0.00 sec)
```

6. Create the compute service tables:

```
# sudo nova-manage db sync
```

Note Make sure the command typed is “db sync” and not “db_sync”. Also, this has to be run as sudo.

7. Set the my_ip, vncserver_listen, and vncserver_proxyclient_address configuration options to the internal IP address of the controller node:

   Edit the **/etc/nova/nova.conf** file and add these lines to the [DEFAULT] section:

```
[DEFAULT]
...
my_ip=10.29.160.184
vncserver_listen=0.0.0.0
vncserver_proxyclient_address=10.29.160.184
```

8. Configure Nova to use the mounted large Filesystem on **/DATA** for VM instances. This step has to be performed on the controller and must be repeated on the Compute nodes. On the controller node run the following:

```
mkdir /DATA/nova
mkdir /DATA/nova/instances
mkdir /DATA/nova/volumes
sudo –r chown nova:nova /DATA/nova*
```

9. Add this location as the path for Nova in **nova.conf** as follows:

```
state_path=/DATA/nova
state_path=/DATA/nova
volumes_path=/DATA/nova/volumes
```

10. Add another configuration to nova.conf only on controller node and node running resource manager to indicate to use XFS as underneath filesystem.

```
virt_mkfs=["linux=xfs"]
```

11. Create a nova user that Compute uses to authenticate with the Identity Service. Use the service tenant and provide admin role to the user:

```
# keystone user-create --name=nova --pass=NOVA_PASS --email=nova@example.com

# keystone user-role-add --user=nova --tenant=service --role=admin
```

```
ubuntu@ubuntu20:~$ keystone user-create --name=nova --pass=ubuntu --email=admin@cisco.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |         admin@cisco.com          |
| enabled  |               True               |
|    id    | 4443cfba7d6c4150ba3e40ee855f6817 |
|   name   |               nova               |
+----------+----------------------------------+
```

12. Configure Compute to use these credentials with the Identity Service running on the controller. Replace **NOVA_PASS** with your Compute password.

    Edit the [DEFAULT] section in the /etc/nova/nova.conf file to add this key:

```
[DEFAULT]
...
auth_strategy=keystone
```

13. Add the credentials to the **/etc/nova/api-paste.ini** file. Add these options to the [filter:authtoken] section:

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = controller
auth_port = 35357
auth_protocol = http
auth_uri = http://controller:5000/v2.0
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS
```

```
ubuntu@ubuntu20:~$ keystone user-create --name=nova --pass=ubuntu --email=admin@cisco.com
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |         admin@cisco.com          |
| enabled  |               True               |
|    id    | 4443cfba7d6c4150ba3e40ee855f6817 |
|   name   |               nova               |
+----------+----------------------------------+
```

Note Make sure that the **api_paste_config=/etc/nova/api-paste.ini** option is set in the **/etc/nova/nova.conf** file.

14. You must register Compute with the Identity Service so that other OpenStack services can locate it. Register the service and specify the endpoint:

```
# keystone service-create --name=nova --type=compute \
--description="Nova Compute service"
```

```
ubuntu@ubuntu20:~$ keystone service-create --name=nova --type=compute \
> --description="Nova Compute service"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |       Nova Compute service       |
|      id     | 4fc32aa5839040098f9001a2f500c116 |
|     name    |               nova               |
|     type    |             compute              |
+-------------+----------------------------------+
```

15. Use the **id** property that is returned to create the endpoint.

```
# keystone endpoint-create \
--service-id=<the_service_id_above> \
--publicurl=http://controller:8774/v2/%\(tenant_id\)s \
--internalurl=http://controller:8774/v2/%\(tenant_id\)s \
--adminurl=http://controller:8774/v2/%\(tenant_id\)s
```

```
ubuntu@ubuntu20:~$ keystone endpoint-create \
> --service-id=4fc32aa5839040098f9001a2f500c116 \
> --publicurl=http://ubuntu20:8774/v2/%\(tenant_id\)s \
> --internalurl=http://ubuntu20:8774/v2/%\(tenant_id\)s \
> --adminurl=http://ubuntu20:8774/v2/%\(tenant_id\)s
+-------------+-----------------------------------------+
|   Property  |                  Value                  |
+-------------+-----------------------------------------+
|   adminurl  | http://ubuntu20:8774/v2/%(tenant_id)s   |
|      id     |    326c6aa0b7394ca7afd1ff655b2bc916     |
| internalurl | http://ubuntu20:8774/v2/%(tenant_id)s   |
|  publicurl  | http://ubuntu20:8774/v2/%(tenant_id)s   |
|    region   |                regionOne                |
|  service_id |    4fc32aa5839040098f9001a2f500c116     |
+-------------+-----------------------------------------+
```

16. Restart Compute services:

```
# sudo service nova-api restart
# sudo service nova-cert restart
# sudo service nova-consoleauth restart
# sudo service nova-scheduler restart
# sudo service nova-conductor restart
# sudo service nova-novncproxy restart
```

```
ubuntu@ubuntu20:~$ sudo service nova-api restart
nova-api stop/waiting
nova-api start/running, process 14768
ubuntu@ubuntu20:~$ sudo service nova-cert restart
nova-cert stop/waiting
nova-cert start/running, process 14795
ubuntu@ubuntu20:~$ sudo service nova-consoleauth restart
nova-consoleauth stop/waiting
nova-consoleauth start/running, process 14823
ubuntu@ubuntu20:~$ sudo service nova-scheduler restart
nova-scheduler stop/waiting
nova-scheduler start/running, process 14844
ubuntu@ubuntu20:~$ sudo service nova-conductor restart
nova-conductor stop/waiting
nova-conductor start/running, process 14874
ubuntu@ubuntu20:~$ sudo service nova-novncproxy restart
nova-novncproxy stop/waiting
nova-novncproxy start/running, process 14896
```

17. To verify your configuration, list available images:

```
# nova image-list
```

```
ubuntu@ubuntu20:~$ nova image-list
+--------------------------------------+---------------------+--------+--------
--------------------------+
| ID                                   | Name                | Status | Server
                         |
+--------------------------------------+---------------------+--------+--------
--------------------------+
| 8717cfff-598d-4e45-8442-674d54c1cd77 | CirrOS 0.3.1        | ACTIVE |
                         |
| 93381385-53d5-4791-8552-905a79597523 | Ubuntu-Precise      | ACTIVE |
                         |
```

Configure a Compute node

After you configure the compute service on the controller node, the systems need to be configured as a compute node. The compute node receives requests from the controller node and hosts virtual machine instances.

The compute service relies on a hypervisor to run virtual machine instances. OpenStack can use various hypervisors, but this guide uses KVM, which is default for nova-compute.

For this architecture of Hadoop as a Service on OpenStack, all nodes are compute nodes including the controller node. The controller node runs Master hadoop services like Namenode and another node runs Resource manager on their VMs and these don't run either Hadoop tasks or Hadoop datanode/ nodeserver for storing data.

This section also provides details on where the instances should be placed, i.e., in the mounted partition **/DATA/** instead of the default in the OS drive.

Note The final nova.conf screenshot is added at the end of this section.

Following steps are performed on all nodes running nova-compute (including controller node):

1. Make sure the Operating System is configured as follows:

- Ensure eth1 is configured for dhcp.
- Ensure /etc/hosts are set on all nodes.
- Ensure all nodes are NTP synchronized.
- Install the MySQL client libraries. You do not need to install the MySQL database server or start the MySQL service.

2. After you configure the operating system, install the appropriate packages for the compute service.

   Run this command on the controller node (ubuntu20):

```
#sudo apt-get -y install nova-compute-kvm python-guestfs
```

   When prompted to create a supermin appliance, respond **yes**.

```
ubuntu@build-node:~$ clush -a -b sudo apt-get -y install nova-compute-kvm python-guestfs
---------------
ubuntu[1-20] (20)
---------------
Reading package lists...
Building dependency tree...
Reading state information...
python-guestfs is already the newest version.
nova-compute-kvm is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Note This can not be run as a **clush** command on all nodes as this requires the supermin to choose the option **yes**, you can run **clush** command to check if the installation is successful.

3. Due to the bug the current kernel need to be made readable. To make it readable, run:

```
# clush -b -a sudo dpkg-statoverride  --update --add root root 0644
/boot/vmlinuz-$(uname -r)
clush -b -a sudo dpkg-statoverride  --update --add ubuntu ubuntu 0644
/boot/vmlinuz-$(uname -r)
To also enable this override for all future kernel updates, create the file
/etc/kernel/postinst.d/statoverride containing:
#!/bin/sh
version="$1"
# passing the kernel version is required
[ -z "${version}" ] && exit 0
dpkg-statoverride --update --add root root 0644 /boot/vmlinuz-${version}
```

```
untu@build-node:~$ clush -b -a sudo dpkg-statoverride  --update --add root root 0644 /boot/vmlinuz-$(uname -r)
untu@build-node:~$ clush -b -a ls -l /boot/vmlinuz-3.11.0-19-generic
---------------
untu[1-20] (20)
---------------
w------- 1 root root 5909632 Mar 12 14:42 /boot/vmlinuz-3.11.0-19-generic
```

Make sure the file is executable:

```
# chmod +x /etc/kernel/postinst.d/statoverride

create this file locally on the admin node and the copy everywhere
clush -a -b -c ./statoverride --dest=/home/ubuntu
clush -a -b sudo cp /home/ubuntu/statoverride /etc/kernel/postinst.d/
clush -a -b sudo chmod +x /etc/kernel/postinst.d/statoverride
clush -a -b ls -l /etc/kernel/postinst.d/statoverride
```

4. Edit the **/etc/nova/nova.conf** configuration file and add these lines to the appropriate sections:

```
...
[DEFAULT]
...
auth_strategy=keystone
...
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:<NOVA_DBPASS>@<controller>/nova
```

5. Configure the Compute Service to use the RabbitMQ message broker by setting these configuration keys in the **[DEFAULT]** configuration group of the **/etc/nova/nova.conf** file:

```
rpc_backend = nova.rpc.impl_kombu
rabbit_host = controller
rabbit_password = RABBIT_PASS
```

6. Configure Compute/controller node to provide remote console access to instances.

   Edit /etc/nova/nova.conf and add the following keys under the [DEFAULT] section:

```
[DEFAULT]
...
my_ip=<public-interface-ip>
vnc_enabled=True
vncserver_listen=0.0.0.0
vncserver_proxyclient_address=<public-interface-ip>
novncproxy_base_url=http://controller:6080/vnc_auto.html
```

Note Before changing individual **/etc/nova/nova.conf**, ensure to add this specific nova-networking config into **/etc/nova/nova.conf** in [default] to all nodes to avoid updating the individual files again. This CVD will go into details of nova-network in detail in the next section.

```
[default]
network_manager=nova.network.manager.FlatDHCPManager
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
network_size=254
allow_same_net_traffic=False
multi_host=True
send_arp_for_ha=True
share_dhcp_address=True
force_dhcp_release=True
flat_network_bridge=br100
flat_interface=eth1
public_interface=eth0
```

Note eth1 in internal-ip interface for openstack VMs to communicate. eth0 is public-ip interface.

7. Configure Nova to use the mounted large Filesystem on **/DATA** for VM instances. This step has to be done on controller and again on compute nodes. On controller node run the following.

```
clush -b -a sudo mkdir /DATA/nova
clush -b -a sudo mkdir /DATA/nova/instances
```

```
clush -b -a sudo mkdir /DATA/nova/volumes
clush -b -a sudo chown -R nova:nova /DATA/nova*
```

8. Add this location as the path for Nova in **nova.conf** as follows on all nodes.

```
state_path=/DATA/nova
state_path=/DATA/nova
volumes_path=/DATA/nova/volumes
```

Note This step is important to ensure all VM instances are in **/DATA** partition which has huge storage compared to the OS drive which would be default

9. Add another configuration to nova.conf to indicate to use XFS as underneath filesystem.

```
virt_mkfs=["linux=xfs"]
```

10. Specify the host that runs the Image Service. Edit **/etc/nova/nova.conf** file and add these lines to the **[DEFAULT]** section on all compute nodes:

```
[DEFAULT]
...
glance_host=controller
```

11. Edit the **/etc/nova/api-paste.ini** file to add the credentials to the **[filter:authtoken]** section:

```
[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory
auth_host = controller
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = NOVA_PASS
```

Changes to many nodes can be done by making changes on one node and copying the file to all the other compute nodes while ensuring not to overwriting the controller node as in following example:

```
clush -b -w ubuntu[1-19,21-64] sudo cp /home/ubuntu/api-paste.ini
/etc/nova/api-paste.ini
```

12. On Controller node only add the following to **/etc/nova/nova.conf** file. This specific filter option "scheduler_default_filters" is needed to have greater control on where to place the VMs, especially the Larger size VMs for the Master Services.

```
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,ComputeFilt
er,ComputeCapabilitiesFilter,ImagePropertiesFilter,SameHostFilter,DifferentHostFil
ter
```

Note If troubleshooting, the option "debug=true" can be set in the controller node **/etc/nova/nova.conf** file. However, this should be removed when in production to prevent performance slowdown. All debug logs will be available in **/var/log/nova/nova-*.log**.

13. Restart the Compute service on all nodes.

```
# clush –a –b sudo service nova-compute restart
```

14. Only on the controller node run the following command:

```
# sudo service nova-scheduler restart
```

15. Remove the SQLite database created by the packages:

```
# clush -a -b sudo rm -f /var/lib/nova/nova.sqlite
```

File nova.conf

```
virt_mkfs=["linux=xfs"]
lock_path=/var/lock/nova
force_dhcp_release=True
iscsi_helper=tgtadm
libvirt_use_virtio_for_bridges=True
connection_type=libvirt
root_helper=sudo nova-rootwrap /etc/nova/rootwrap.conf
verbose=True
default_floating_pool = cisco_CentOS
floating_range = 10.29.161.2/24
auto_assign_floating_ip=True
ec2_private_dns_show_ip=True
api_paste_config=/etc/nova/api-paste.ini
volumes_path=/DATA/nova/volumes
enabled_apis=ec2,osapi_compute,metadata
rpc_backend = nova.rpc.impl_kombu
rabbit_host = ubuntu20
rabbit_password = ubuntu
my_ip=10.29.160.184
vncserver_listen=0.0.0.0
vncserver_proxyclient_address=10.29.160.184
auth_strategy=keystone
network_manager=nova.network.manager.FlatDHCPManager
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
network_size=254
allow_same_net_traffic=False
multi_host=True
send_arp_for_ha=True
share_dhcp_address=True
force_dhcp_release=True
flat_network_bridge=br100
flat_interface=eth1
public_interface=eth0
scheduler_default_filters=RetryFilter,AvailabilityZoneFilter,RamFilter,ComputeFilter
,ComputeCapabilitiesFilter,ImagePropertiesFilter,SameHostFilter,DifferentHostFilter
[database]
# The SQLAlchemy connection string used to connect to the database
connection = mysql://nova:ubuntu@ubuntu20/nova
[keystone_authtoken]
auth_host = ubuntu20
auth_port = 35357
auth_protocol = http
admin_tenant_name = service
admin_user = nova
admin_password = ubuntu
```

Note Make sure MTU on both eth0 and eth1 are 9000. Without MTU set to 9000 the compute of nova-compute cannot register with controller through rabbitmq. To check if it is working fine, see if the **ping -s 9000 <controller-node>** is reachable.

Enable Networking (nova-networking)

This section discusses configuring FlatNetworking in OpenStack compute, that takes care of DHCP.

This set up uses multi-host functionality. Networking is configured to be highly available by distributing networking functionality across multiple hosts. As a result, no single network controller acts as a single point of failure. This process configures each compute node for networking.

1. Install the appropriate packages for compute networking on all the nova-compute nodes (compute nodes and controller node controller node also because we are installing nova-compute on controller).

   So that the nova-network service can forward metadata requests on each compute node, each compute node must install the **nova-api-metadata** service, as follows:

   ```
   # clush –a –b sudo apt-get –y install nova-network nova-api-metadata
   ```

```
ubuntu@build-node:~$ clush -a -b sudo apt-get -y install nova-network nova-api-metadata
---------------
ubuntu[1-20] (20)
---------------
Reading package lists...
Building dependency tree...
Reading state information...
nova-api-metadata is already the newest version.
nova-network is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

2. Edit the **/etc/nova/nova.conf** file to define the networking mode and add these lines to the **[DEFAULT]** section:

Note This was already done in the above section. To avoid updating the individual files, edit nova.conf.

```
[default]
network_manager=nova.network.manager.FlatDHCPManager
firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
network_size=254
allow_same_net_traffic=False
multi_host=True
send_arp_for_ha=True
share_dhcp_address=True
force_dhcp_release=True
flat_network_bridge=br100
flat_interface=eth1
public_interface=eth0
```

Note This must be done on the controller node as well.

3. Restart the network service:

   ```
   # clush –a –b sudo service nova-network restart
   ```

4. Create a network that virtual machines can use. Do this once for the entire installation and not on each compute node. Run the **nova network-create** command on the controller.

Note Change the fixed-range-v4 based on your network environment

```
# source openrc.sh
# nova network-list
```

```
# sudo nova-manage network create vmnet --fixed_range_v4=192.168.10.0/24
--bridge=br100 --multi_host=T
# nova network-list
```

```
ubuntu@ubuntu20:~$ nova network-list
+--------------------------------------+-------+-----------------+
| ID                                   | Label | Cidr            |
+--------------------------------------+-------+-----------------+
| 78a78a24-5a3e-4c2f-ad0e-13bb4cf358bb | vmnet | 192.168.10.0/24 |
+--------------------------------------+-------+-----------------+
```

Note Troubleshooting Note: Due to a bug, the "nova-api service" is automatically removed after you trying to install "nova-compute". This causes an error while running the above command such as **"HTTPConnectionPool(host='10.0.0.0', port=8774): Max retries exceeded with url"**

Solution:

Reinstall **nova-api** as follows on the controller:

```
sudo apt-get install nova-novncproxy novnc nova-api
```

Restart all the nova services as:

```
service nova-api restart
service nova-cert restart
service nova-consoleauth restart
service nova-scheduler restart
service nova-conductor restart
service nova-novncproxy restart
```

5. On controller node, to ensure all the nova-compute nodes are registered fine, run the following command. Ensure all the compute-nodes are listed.

```
#nova hypervisor-list
```

```
ubuntu@ubuntu20:~$ nova hypervisor-list
+----+---------------------+
| ID | Hypervisor hostname |
+----+---------------------+
| 1  | ubuntu20.cisco.com  |
| 2  | ubuntu2.cisco.com   |
| 3  | ubuntu3.cisco.com   |
| 4  | ubuntu11.cisco.com  |
| 5  | ubuntu14.cisco.com  |
| 6  | ubuntu4.cisco.com   |
| 7  | ubuntu8.cisco.com   |
| 8  | ubuntu17.cisco.com  |
| 9  | ubuntu5.cisco.com   |
| 10 | ubuntu13.cisco.com  |
| 11 | ubuntu1.cisco.com   |
| 12 | ubuntu12.cisco.com  |
| 13 | ubuntu10.cisco.com  |
| 14 | ubuntu16.cisco.com  |
| 15 | ubuntu19.cisco.com  |
| 16 | ubuntu6.cisco.com   |
| 17 | ubuntu18.cisco.com  |
| 18 | ubuntu7.cisco.com   |
| 19 | ubuntu9.cisco.com   |
| 20 | ubuntu15.cisco.com  |
+----+---------------------+
```

#sudo nova-manage service list

```
ubuntu@ubuntu20:~$ sudo nova-manage service list
Binary           Host                                 Zone             Status     State Updated_At
nova-cert        ubuntu20                             internal         enabled    :-)   2014-04-16 02:13:43
nova-consoleauth ubuntu20                             internal         enabled    :-)   2014-04-16 02:13:39
nova-scheduler   ubuntu20                             internal         enabled    :-)   2014-04-16 02:13:37
nova-conductor   ubuntu20                             internal         enabled    :-)   2014-04-16 02:13:43
nova-compute     ubuntu20                             nova             enabled    :-)   2014-04-16 02:13:41
nova-network     ubuntu20                             internal         enabled    :-)   2014-04-16 02:13:40
nova-compute     ubuntu2                              nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu13                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu6                              nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu12                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu14                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu18                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu1                              nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu11                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu3                              nova             enabled    :-)   2014-04-16 02:13:40
nova-compute     ubuntu16                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu17                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu4                              nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu9                              nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu10                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu8                              nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu5                              nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu19                             nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu7                              nova             enabled    :-)   2014-04-16 02:13:41
nova-compute     ubuntu15                             nova             enabled    :-)   2014-04-16 02:13:41
```

Launch an instance

After configuring the compute services, instances can be launched. An instance is a virtual machine that OpenStack provisions on compute servers.

1. Generate a keypair that consists of a private and public key to be able to launch instances on OpenStack. These keys are injected into the instances to make password-less SSH access to the instance. This depends on the way the necessary tools are bundled into the images. For more details, see the OpenStack Admin User Guide.

```
$ ssh-keygen
$ cd .ssh
$ nova keypair-add mykey > mykey.pem
```

   The **id_rsa** private key is saved locally in **~/.ssh**, which can be used to connect to an instance launched by using mykey as the keypair. To view available keypairs:

```
$ nova keypair-list
```

```
ubuntu@ubuntu20:~$ cd .ssh/
ubuntu@ubuntu20:~/.ssh$ nova keypair-add --pub_key id_rsa.pub mykey
ubuntu@ubuntu20:~/.ssh$ nova keypair-list
+-------+-------------------------------------------------+
| Name  | Fingerprint                                     |
+-------+-------------------------------------------------+
| mykey | 30:a0:3d:89:b8:67:0b:4d:c4:d4:d2:4e:3b:d8:b0:42 |
+-------+-------------------------------------------------+
```

2. To launch an instance, you must specify the ID for the flavor you want to use for the instance. A flavor is a resource allocation profile. For example, it specifies how many virtual CPUs and how much RAM your instance gets. To see a list of the available profiles:

```
$ nova flavor-list

nova flavor-create [--ephemeral <ephemeral>] [--swap <swap>] \
[--rxtx-factor <factor>] [--is-public <is-public>]  \
<name> <id> <ram> <disk> <vcpus>
```

   where:

   - --ephemeral <ephemeral> (optional) is the ephemeral space size in GB (the default is 0).
   - --swap <swap> (optional) is the swap size in MB (the default is 0).
   - --rxtx-factor <factor> is the RX/TX factor (the default is 1).
   - --is-public <is-public> makes the flavor accessible to the public (the default is true).
   - <name> is the name of the new flavor.
   - <id> is the unique integer ID for the new flavor.
   - <ram> is the memory size in MB.
   - <disk> is the disk size in GB.
   - <vcpus> is the number of vCPUs.

   To create a hadoop slave (data-node and task-tracker) flavor for a VM where 8 VMs would be hosted on C240M3 server with 20 Physical cores and 256GB RAM, run the following command to create a VM flavor with 2vCPUs, 50GB OS disk space, 28250 MB Memory and 2TB local ephemeral disk space for each of the VM.

```
nova flavor-create hadoop.8vm.ephemeral --ephemeral 2000  8 28250 50 2
```

To Create a hadoop Master (Namenode and Job-tracker/Resource Manager) flavor for a VM where 1 VM would be hosted on C240M3 server with 20 Physical cores and 256GB RAM, run the following command to create a VM flavor with 16vCPUs, 50GB OS disk space, 226000 MB Memory and 20TB local ephemeral disk space for each of the VM.

```
nova flavor-create hadoop.master --ephemeral 20000 9 226000 50 16
```

Similarly,

```
nova flavor-create cirr.small.eph --ephemeral 10 10 2000 2 1
```

```
ubuntu@ubuntu1:~$ nova flavor-create hadoop.8vm.ephemeral --ephemeral 2000 8 28250 50 2
+----+----------------------+-----------+------+-----------+------+-------+-------------+-----------
| ID | Name                 | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public
+----+----------------------+-----------+------+-----------+------+-------+-------------+-----------
| 8  | hadoop.8vm.ephemeral | 28250     | 50   | 2000      |      | 2     | 1.0         | True
+----+----------------------+-----------+------+-----------+------+-------+-------------+-----------
ubuntu@ubuntu1:~$ nova flavor-create hadoop.master --ephemeral 20000 9 226000 50 16
+----+---------------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name          | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+---------------+-----------+------+-----------+------+-------+-------------+-----------+
| 9  | hadoop.master | 226000    | 50   | 20000     |      | 16    | 1.0         | True      |
+----+---------------+-----------+------+-----------+------+-------+-------------+-----------+
```

nova flavor-list

```
ubuntu@ubuntu20:~$ nova flavor-list
+----+----------------------+-----------+------+-----------+------+-------+-------------+-----------+
| ID | Name                 | Memory_MB | Disk | Ephemeral | Swap | VCPUs | RXTX_Factor | Is_Public |
+----+----------------------+-----------+------+-----------+------+-------+-------------+-----------+
| 1  | m1.tiny              | 512       | 1    | 0         |      | 1     | 1.0         | True      |
| 10 | cirr.small.eph       | 2000      | 2    | 10        |      | 1     | 1.0         | True      |
| 11 | hadoop.4vm.ephemeral | 56500     | 50   | 4000      |      | 4     | 1.0         | True      |
| 12 | hadoop.2vm.ephemeral | 113000    | 50   | 8000      |      | 8     | 1.0         | True      |
| 13 | hadoop.1vm.ephemeral | 226000    | 50   | 16000     |      | 16    | 1.0         | True      |
| 14 | hadoop.4vm.ephemeral2| 28250     | 50   | 4000      |      | 4     | 1.0         | True      |
| 2  | m1.small             | 2048      | 20   | 0         |      | 1     | 1.0         | True      |
| 3  | m1.medium            | 4096      | 40   | 0         |      | 2     | 1.0         | True      |
| 4  | m1.large             | 8192      | 80   | 0         |      | 4     | 1.0         | True      |
| 5  | m1.xlarge            | 16384     | 160  | 0         |      | 8     | 1.0         | True      |
| 8  | hadoop.8vm.ephemeral | 28250     | 50   | 2000      |      | 2     | 1.0         | True      |
| 9  | hadoop.master        | 226000    | 50   | 20000     |      | 16    | 1.0         | True      |
+----+----------------------+-----------+------+-----------+------+-------+-------------+-----------+
```

Note Flavors can be created from Dashboard Horizon as well.

3. Get the ID of the image to use for the instance:

```
$ nova image-list
```

```
ubuntu@ubuntu20:~$ nova image-list
+--------------------------------------+---------------+--------+--------+
| ID                                   | Name          | Status | Server |
+--------------------------------------+---------------+--------+--------+
| 8717cfff-598d-4e45-8442-674d54c1cd77 | CirrOS 0.3.1  | ACTIVE |        |
| 93381385-53d5-4791-8552-905a79597523 | Ubuntu-Precise | ACTIVE |        |
+--------------------------------------+---------------+--------+--------+
ubuntu@ubuntu20:~$ glance image-list
+--------------------------------------+---------------+-------------+------------------+----------+--------+
| ID                                   | Name          | Disk Format | Container Format | Size     | Status |
+--------------------------------------+---------------+-------------+------------------+----------+--------+
| 8717cfff-598d-4e45-8442-674d54c1cd77 | CirrOS 0.3.1  | qcow2       | bare             | 13147648 | active |
| 93381385-53d5-4791-8552-905a79597523 | Ubuntu-Precise | qcow2       | bare             | 260112896 | active |
+--------------------------------------+---------------+-------------+------------------+----------+--------+
```

```
$nova network-list
```

```
ubuntu@ubuntu20:~$ nova network-list
+--------------------------------------+-------+-----------------+
| ID                                   | Label | Cidr            |
+--------------------------------------+-------+-----------------+
| 78a78a24-5a3e-4c2f-ad0e-13bb4cf358bb | vmnet | 192.168.10.0/24 |
+--------------------------------------+-------+-----------------+
```

4. To enable SSH and ping and all TCP (used in hadoop), you must configure security group rules.

```
# nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
# nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
# nova secgroup-add-rule default tcp 1 65535 0.0.0.0/0

nova secgroup-list
nova secgroup-list-rules default
```

```
ubuntu@ubuntu20:~$ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| tcp         | 22        | 22      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
ubuntu@ubuntu20:~$ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
+-------------+-----------+---------+-----------+--------------+
| IP Protocol | From Port | To Port | IP Range  | Source Group |
+-------------+-----------+---------+-----------+--------------+
| icmp        | -1        | -1      | 0.0.0.0/0 |              |
+-------------+-----------+---------+-----------+--------------+
```

5. Before launching the VMs we need to increase the default quota on each node.

```
#nova quota-show
```

Since all these are run as admin (see openrc.sh file create earlier which was sourced), get the tenant-id of admin to update RAM for VM instances.

```
#cat openrc.sh
export OS_USERNAME=admin
export OS_PASSWORD=ubuntu
export OS_TENANT_NAME=admin
export OS_AUTH_URL=http://ubuntu20:35357/v2.0
```

Update RAM to be 230000, this is slightly over to 8VMs each of 28500MB RAM, as in the hadoop.8vm.slave flavor created. As mentioned earlier, all nodes are 256GB RAM.

Note

- All the quotas are for the project (identified by tenant-id and in this case for project admin) on the entire cluster on which openstack is deployed. So in-order to update instances, ram or cores, we need to consider how many per node and how many nodes and provide the total number for quota-update. In this config, each node has 20 physical core and 256 GB RAM, of which 16 cores are provided for Openstack and 230000MB for Openstack Memory and roughly 8 instances per node.
- The command **keystone tenant-list** gives the tenant-id for admin.

```
#keystone tenant-list
#nova quota-update --ram 5000000 <admin-tenant-id>
#nova quota-update --instances 200 < admin-tenant-id>
#nova quota-update --cores 400 < admin-tenant-id>
#nova quota-update --key-pairs 200 < admin-tenant-id>
#nova quota-update --floating-ips 200 < admin-tenant-id>

#nova quota-show
```

```
ubuntu@ubuntu20:~$ nova quota-show
+-----------------------------+---------+
| Quota                       | Limit   |
+-----------------------------+---------+
| instances                   | 200     |
| cores                       | 400     |
| ram                         | 5000000 |
| floating_ips                | 200     |
| fixed_ips                   | -1      |
| metadata_items              | 128     |
| injected_files              | 5       |
| injected_file_content_bytes | 10240   |
| injected_file_path_bytes    | 255     |
| key_pairs                   | 200     |
| security_groups             | 10      |
| security_group_rules        | 20      |
+-----------------------------+---------+
```

These can be updated from Horizon dashboard too.

6. **Creating VMs on all the Host Servers**: Before placing the VMs from Ubuntu to host Hadoop, we need to ensure that the Hadoop master VMs are placed in both the nodes specially configured for master nodes with different RAID and will be a single VM running on the node with 226000 MB RAM.

   In order to achieve this, we need to launch the cirros instance on nodes giving a filter of Different_host so as to get the ids of cirros to pass it as option to launch Ubuntu VMs.

```
#nova boot --flavor 1 --key_name mykey --image
8717cfff-598d-4e45-8442-674d54c1cd77 --security_group default cirros
```

```
ubuntu@ubuntu20:~$ nova boot --flavor 1 --key_name mykey --image 8717cfff-598d-4
e45-8442-674d54c1cd77 --security_group default cirros
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
| OS-EXT-STS:task_state                | scheduling                           |
| image                                | CirrOS 0.3.1                         |
| OS-EXT-STS:vm_state                  | building                             |
| OS-EXT-SRV-ATTR:instance_name        | instance-0000000d                    |
| OS-SRV-USG:launched_at               | None                                 |
| flavor                               | m1.tiny                              |
| id                                   | 6cb4df7f-8ea1-457a-83ab-a479a42beb5f |
| security_groups                      | [{u'name': u'default'}]              |
| user_id                              | f33bd76628704fc9976c9a4f4d7bba76     |
| OS-DCF:diskConfig                    | MANUAL                               |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| progress                             | 0                                    |
| OS-EXT-STS:power_state               | 0                                    |
| OS-EXT-AZ:availability_zone          | nova                                 |
| config_drive                         |                                      |
| status                               | BUILD                                |
| updated                              | 2014-04-16T21:47:51Z                 |
| hostId                               |                                      |
| OS-EXT-SRV-ATTR:host                 | None                                 |
| OS-SRV-USG:terminated_at             | None                                 |
| key_name                             | mykey                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                 |
| name                                 | cirros                               |
| adminPass                            | 2tfzc22DxmbL                         |
| tenant_id                            | a688d1fba0cb4adc96f9051fff53e5b5     |
| created                              | 2014-04-16T21:47:51Z                 |
| os-extended-volumes:volumes_attached | []                                   |
| metadata                             | {}                                   |
+--------------------------------------+--------------------------------------+
```

To list the VMs launched run the following command.

```
#nova list

#nova show 6cb4df7f-8ea1-457a-83ab-a479a42beb5f
```

```
buntu@ubuntu20:~$ nova list
+--------------------------------------+--------+--------+------------+-------------+------------------+
| ID                                   | Name   | Status | Task State | Power State | Networks         |
+--------------------------------------+--------+--------+------------+-------------+------------------+
| 6cb4df7f-8ea1-457a-83ab-a479a42beb5f | cirros | ACTIVE | None       | Running     | vmnet=192.168.10.2
+--------------------------------------+--------+--------+------------+-------------+------------------+
buntu@ubuntu20:~$ nova show 6cb4df7f-8ea1-457a-83ab-a479a42beb5f
+--------------------------------------+----------------------------------------------------------+
| Property                             | Value                                                    |
+--------------------------------------+----------------------------------------------------------+
| status                               | ACTIVE                                                   |
| updated                              | 2014-04-16T21:48:02Z                                     |
| OS-EXT-STS:task_state                | None                                                     |
| OS-EXT-SRV-ATTR:host                 | ubuntu7                                                  |
| key_name                             | mykey                                                    |
| image                                | CirrOS 0.3.1 (8717cfff-598d-4e45-8442-674d54c1cd77)      |
| vmnet network                        | 192.168.10.2                                             |
| hostId                               | 8b4a46ba4002da64b6909ae7aa4223d92cd3ad1a6e8d058220e8b66e |
| OS-EXT-STS:vm_state                  | active                                                   |
| OS-EXT-SRV-ATTR:instance_name        | instance-0000000d                                        |
| OS-SRV-USG:launched_at               | 2014-04-16T21:48:02.000000                               |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | ubuntu7.cisco.com                                        |
| flavor                               | m1.tiny (1)                                              |
```

We now know that this VM was placed in node ubuntu7, so rename the VM in-order to use this information later.

```
#nova rename cirros cirros-ubuntu7
```

```
ubuntu@ubuntu20:~$ nova rename cirros cirros-ubuntu7
ubuntu@ubuntu20:~$ nova list
+--------------------------------------+---------------+--------+------------+-------------+------------------+
| ID                                   | Name          | Status | Task State | Power State | Networks         |
+--------------------------------------+---------------+--------+------------+-------------+------------------+
| 6cb4df7f-8ea1-457a-83ab-a479a42beb5f | cirros-ubuntu7 | ACTIVE | None       | Running     | vmnet=192.168.10.2
+--------------------------------------+---------------+--------+------------+-------------+------------------+
```

Now create another cirros instance on a different node by passing a hint **different_host** option as:

```
#nova boot --flavor 1 --key_name mykey --image
8717cfff-598d-4e45-8442-674d54c1cd77 --security_group default --hint
different_host=6cb4df7f-8ea1-457a-83ab-a479a42beb5f cirros
```

```
ubuntu@ubuntu20:~$ nova boot --flavor 1 --key_name mykey --image 8717cfff-598d-4e
45-8442-674d54c1cd77 --security_group default --hint different_host=6cb4df7f-8ea1
-457a-83ab-a479a42beb5f cirros
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
| OS-EXT-STS:task_state                | scheduling                           |
| image                                | CirrOS 0.3.1                         |
| OS-EXT-STS:vm_state                  | building                             |
| OS-EXT-SRV-ATTR:instance_name        | instance-0000000e                    |
| OS-SRV-USG:launched_at               | None                                 |
| flavor                               | m1.tiny                              |
| id                                   | 31457a87-d314-4c02-8a05-90e2932ed352 |
| security_groups                      | [{u'name': u'default'}]              |
| user_id                              | f33bd76628704fc9976c9a4f4d7bba76     |
| OS-DCF:diskConfig                    | MANUAL                               |
| accessIPv4                           |                                      |
```

```
#nova list
#nova show 6cb4df7f-8ea1-457a-83ab-a479a42beb5f
```

```
buntu@ubuntu20:~$ nova list
+--------------------------------------+----------------+--------+------------+-------------+-------------------+
| ID                                   | Name           | Status | Task State | Power State | Networks          |
+--------------------------------------+----------------+--------+------------+-------------+-------------------+
| 31457a87-d314-4c02-8a05-90e2932ed352 | cirros         | ACTIVE | None       | Running     | vmnet=192.168.10.3|
| 6cb4df7f-8ea1-457a-83ab-a479a42beb5f | cirros-ubuntu7 | ACTIVE | None       | Running     | vmnet=192.168.10.2|
+--------------------------------------+----------------+--------+------------+-------------+-------------------+
buntu@ubuntu20:~$ nova show 31457a87-d314-4c02-8a05-90e2932ed352
+-------------------------------------+----------------------------------------------------------+
| Property                            | Value                                                    |
+-------------------------------------+----------------------------------------------------------+
| status                              | ACTIVE                                                   |
| updated                             | 2014-04-16T21:53:12Z                                     |
| OS-EXT-STS:task_state               | None                                                     |
| OS-EXT-SRV-ATTR:host                | ubuntu8                                                  |
| key_name                            | mykey                                                    |
| image                               | CirrOS 0.3.1 (8717cfff-598d-4e45-8442-674d54c1cd77)      |
| vmnet network                       | 192.168.10.3                                             |
```

7. We now know that this VM was placed in node ubuntu8, so rename the VM in-order to use this information later.
8. Repeat the same so as to get a VM on all the physical hosts using **different_host=<img-id1>,<img-id2>..**.

```
#nova list
```

```
ubuntu@ubuntu20:~$ nova list
+--------------------------------------+----------------+--------+------------+-------------+--------------------+
| ID                                   | Name           | Status | Task State | Power State | Networks           |
+--------------------------------------+----------------+--------+------------+-------------+--------------------+
| ca44ffe4-6abe-4b9f-b4b2-99bda322799d | cirros-ubuntu1  | ACTIVE | None       | Running     | vmnet=192.168.10.15|
| 0af29cf7-3a6c-443f-9e05-be9fe388447c | cirros-ubuntu10 | ACTIVE | None       | Running     | vmnet=192.168.10.12|
| d4e147d0-a877-4323-80c1-8cb3f040e849 | cirros-ubuntu11 | ACTIVE | None       | Running     | vmnet=192.168.10.16|
| d559ad38-1e27-4aab-a8d2-b4e49a8e3d2d | cirros-ubuntu12 | ACTIVE | None       | Running     | vmnet=192.168.10.18|
| 6ccd3fab-8e21-498b-806e-36897b3287ce | cirros-ubuntu13 | ACTIVE | None       | Running     | vmnet=192.168.10.20|
| e3845563-7e01-45cb-ae4f-5816486748d8 | cirros-ubuntu14 | ACTIVE | None       | Running     | vmnet=192.168.10.5 |
| b6ff6172-6b8d-417a-a1bf-129c8b0cf0cf | cirros-ubuntu15 | ACTIVE | None       | Running     | vmnet=192.168.10.10|
| eaeba37a-033c-46d9-a0c1-2e41e640c5f2 | cirros-ubuntu16 | ACTIVE | None       | Running     | vmnet=192.168.10.14|
| 4a590988-32ff-4f55-ae90-fc1b4fca99c7 | cirros-ubuntu17 | ACTIVE | None       | Running     | vmnet=192.168.10.7 |
| 2a2955a1-3f6a-4c8b-bb43-56eeee04888b | cirros-ubuntu18 | ACTIVE | None       | Running     | vmnet=192.168.10.19|
| 236d7f23-8dda-4845-9204-e0e94d673ae7 | cirros-ubuntu19 | ACTIVE | None       | Running     | vmnet=192.168.10.17|
| dfc73ee2-c8dc-4a05-a7d7-4e55a991787b | cirros-ubuntu2  | ACTIVE | None       | Running     | vmnet=192.168.10.21|
| 12aed8ed-7e23-4bfe-a5ea-140f8f5eac7d | cirros-ubuntu20 | ACTIVE | None       | Running     | vmnet=192.168.10.4 |
| cd2b503b-7e63-4f01-8673-dd354f708a02 | cirros-ubuntu3  | ACTIVE | None       | Running     | vmnet=192.168.10.11|
| f5d92cad-22b5-49e1-8a53-e0aeab664ee5 | cirros-ubuntu4  | ACTIVE | None       | Running     | vmnet=192.168.10.9 |
| b3f1ab14-a913-4fab-9913-2a76be24a375 | cirros-ubuntu5  | ACTIVE | None       | Running     | vmnet=192.168.10.6 |
| a91655c3-baed-4d80-ab55-9151b0f73528 | cirros-ubuntu6  | ACTIVE | None       | Running     | vmnet=192.168.10.13|
| 6cb4df7f-8ea1-457a-83ab-a479a42beb5f | cirros-ubuntu7  | ACTIVE | None       | Running     | vmnet=192.168.10.2 |
| 31457a87-d314-4c02-8a05-90e2932ed352 | cirros-ubuntu8  | ACTIVE | None       | Running     | vmnet=192.168.10.3 |
| e7e65af8-320d-44c0-b02d-6e929c4a88cf | cirros-ubuntu9  | ACTIVE | None       | Running     | vmnet=192.168.10.8 |
+--------------------------------------+----------------+--------+------------+-------------+--------------------+
```

The Ubuntu VMs can be launched as follows by providing the hing “same_host” with the VM id of the cirros running on the same host as the one we need the VM to be placed in.

```
$ nova boot --flavor flavorType --key_name keypairName --image ID --hint
same_host=<id_of_cirros_on_node_host> <instance-name>
```

Create an instance by using flavor 8 for datanodes VMs. For example:

```
$ nova boot --flavor 8 --key_name mykey --image
93381385-53d5-4791-8552-905a79597523 --security_group default  --hint
same_host=<id_of_cirros_vm_on_ubuntu1> ubuntu1-vm1
```

Create an instance by using flavor 9 for Namenode/ResourceManager VMs on nodes specially configured for that as done earlier. In this CVD, controller and Namenode are in same host.

```
ubuntu@ubuntu1:~$ nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791
nt same_host=47f232af-f916-4060-8f3e-d120ec800d39 ubuntu1-vm1
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
| OS-EXT-STS:task_state                | scheduling                           |
| image                                | Ubuntu-Precise                       |
| OS-EXT-STS:vm_state                  | building                             |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000039                    |
| OS-SRV-USG:launched_at               | None                                 |
| flavor                               | hadoop.8vm.ephemeral                 |
| id                                   | 719bae6e-30c0-4291-931e-b2cc0a37d597 |
| security_groups                      | [{u'name': u'default'}]              |
| user_id                              | f33bd76628704fc9976c9a4f4d7bba76     |
| OS-DCF:diskConfig                    | MANUAL                               |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| progress                             | 0                                    |
| OS-EXT-STS:power_state               | 0                                    |
| OS-EXT-AZ:availability_zone          | nova                                 |
| config_drive                         |                                      |
| status                               | BUILD                                |
```

```
$ nova boot --flavor 9 --key_name mykey --image
93381385-53d5-4791-8552-905a79597523 --security_group default  --hint
same_host=<id_of_cirros_vm_on_ubuntu20> ubuntu20-namenode
```

```
ubuntu@ubuntu20:~$ nova boot --flavor 9 --key_name mykey --image 93381385-53d5-47
91-8552-905a79597523 --security_group default  --hint same_host=12aed8ed-7e23-4bf
e-a5ea-140f8f5eac7d ubuntu20-namenode
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
| OS-EXT-STS:task_state                | scheduling                           |
| image                                | Ubuntu-Precise                       |
| OS-EXT-STS:vm_state                  | building                             |
| OS-EXT-SRV-ATTR:instance_name        | instance-00000045                    |
| OS-SRV-USG:launched_at               | None                                 |
| flavor                               | hadoop.master                        |
| id                                   | 41a6bfde-8404-4768-bf8f-2dcdb6af6745 |
| security_groups                      | [{u'name': u'default'}]              |
| user_id                              | f33bd76628704fc9976c9a4f4d7bba76     |
| OS-DCF:diskConfig                    | MANUAL                               |
| accessIPv4                           |                                      |
| accessIPv6                           |                                      |
| progress                             | 0                                    |
| OS-EXT-STS:power_state               | 0                                    |
| OS-EXT-AZ:availability_zone          | nova                                 |
| config_drive                         |                                      |
| status                               | BUILD                                |
| updated                              | 2014-04-21T23:30:27Z                 |
| hostId                               |                                      |
| OS-EXT-SRV-ATTR:host                 | None                                 |
| OS-SRV-USG:terminated_at             | None                                 |
| key_name                             | mykey                                |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                 |
| name                                 | ubuntu20-namenode                    |
| adminPass                            | PGFe4oaUZrzA                         |
| tenant_id                            | a688d1fba0cb4adc96f9051fff53e5b5     |
| created                              | 2014-04-21T23:30:27Z                 |
| os-extended-volumes:volumes_attached | []                                   |
| metadata                             | {}                                   |
+--------------------------------------+--------------------------------------+
```

Nova show ubuntu20-namenode

```
ubuntu@ubuntu20:~$ nova show ubuntu20-namenode
+--------------------------------------+----------------------
| Property                             | Value
+--------------------------------------+----------------------
| status                               | ACTIVE
| updated                              | 2014-04-21T23:30:32Z
| OS-EXT-STS:task_state                | None
| OS-EXT-SRV-ATTR:host                 | ubuntu20
| key_name                             | mykey
```

```
$ nova boot --flavor 9 --key_name mykey --image
93381385-53d5-4791-8552-905a79597523 --security_group default  --hint
same_host=<id_of_cirros_vm_on_ubuntu19> ubuntu19-resourcemgr
```

Note ssh to Ubuntu VMs won't work unless floating-ips are assigned which will be done in the next section.

9. After the instance launches, use the **nova list** to view its status. The status changes from **BUILD** to **ACTIVE**:

```
$ nova list
```

```
buntu@ubuntu20:~$ nova list
+--------------------------------------+-----------------------+--------+------------+-------------+--------------------+
| ID                                   | Name                  | Status | Task State | Power State | Networks           |
+--------------------------------------+-----------------------+--------+------------+-------------+--------------------+
| ca44ffe4-6abe-4b9f-b4b2-99bda322799d | cirros-ubuntu1        | ACTIVE | None       | Running     | vmnet=192.168.10.15 |
| 0af29cf7-3a6c-443f-9e05-be9fe388447c | cirros-ubuntu10       | ACTIVE | None       | Running     | vmnet=192.168.10.12 |
| d4e147d0-a877-4323-80c1-8cb3f040e849 | cirros-ubuntu11       | ACTIVE | None       | Running     | vmnet=192.168.10.16 |
| d559ad38-1e27-4aab-a8d2-b4e49a8e3d2d | cirros-ubuntu12       | ACTIVE | None       | Running     | vmnet=192.168.10.18 |
| 6ccd3fab-8e21-498b-806e-36897b3287ce | cirros-ubuntu13       | ACTIVE | None       | Running     | vmnet=192.168.10.20 |
| e3845563-7e01-45cb-ae4f-5816486748d8 | cirros-ubuntu14       | ACTIVE | None       | Running     | vmnet=192.168.10.5  |
| b6ff6172-6b8d-417a-a1bf-129c8b0cf0cf | cirros-ubuntu15       | ACTIVE | None       | Running     | vmnet=192.168.10.10 |
| eaeba37a-033c-46d9-a0c1-2e41e640c5f2 | cirros-ubuntu16       | ACTIVE | None       | Running     | vmnet=192.168.10.14 |
| 4a590988-32ff-4f55-ae90-fc1b4fca99c7 | cirros-ubuntu17       | ACTIVE | None       | Running     | vmnet=192.168.10.7  |
| 2a2955a1-3f6a-4c8b-bb43-56eeee04888b | cirros-ubuntu18       | ACTIVE | None       | Running     | vmnet=192.168.10.19 |
| 236d7f23-8dda-4845-9204-e0e94d673ae7 | cirros-ubuntu19       | ACTIVE | None       | Running     | vmnet=192.168.10.17 |
| dfc73ee2-c8dc-4a05-a7d7-4e55a991787b | cirros-ubuntu2        | ACTIVE | None       | Running     | vmnet=192.168.10.21 |
| 12aed8ed-7e23-4bfe-a5ea-140f8f5eac7d | cirros-ubuntu20       | ACTIVE | None       | Running     | vmnet=192.168.10.4  |
| cd2b503b-7e63-4f01-8673-dd354f708a02 | cirros-ubuntu3        | ACTIVE | None       | Running     | vmnet=192.168.10.11 |
| f5d92cad-22b5-49e1-8a53-e0aeab664ee5 | cirros-ubuntu4        | ACTIVE | None       | Running     | vmnet=192.168.10.9  |
| b3f1ab14-a913-4fab-9913-2a76be24a375 | cirros-ubuntu5        | ACTIVE | None       | Running     | vmnet=192.168.10.6  |
| a91655c3-baed-4d80-ab55-9151b0f73528 | cirros-ubuntu6        | ACTIVE | None       | Running     | vmnet=192.168.10.13 |
| 6cb4df7f-8ea1-457a-83ab-a479a42beb5f | cirros-ubuntu7        | ACTIVE | None       | Running     | vmnet=192.168.10.2  |
| 31457a87-d314-4c02-8a05-90e2932ed352 | cirros-ubuntu8        | ACTIVE | None       | Running     | vmnet=192.168.10.3  |
| e7e65af8-320d-44c0-b02d-6e929c4a88cf | cirros-ubuntu9        | ACTIVE | None       | Running     | vmnet=192.168.10.8  |
| 7bd22fde-1098-4a25-97a8-84e2217d728d | ubuntu1-vm1           | ACTIVE | None       | Running     | vmnet=192.168.10.22 |
| 5713ccc4-d750-4e9c-b3e9-9e60691352f3 | ubuntu1-vm2           | ACTIVE | None       | Running     | vmnet=192.168.10.23 |
| 9ee874a8-0424-4445-ac00-e120c6c7ddb6 | ubuntu1-vm3           | ACTIVE | None       | Running     | vmnet=192.168.10.24 |
| ceff83d2-8a97-4da2-b21f-c48d1a18f14c | ubuntu1-vm4           | ACTIVE | None       | Running     | vmnet=192.168.10.25 |
| 87e8a413-b0d8-4281-8bfe-31d10ceef191 | ubuntu1-vm5           | ACTIVE | None       | Running     | vmnet=192.168.10.26 |
| 125dec15-45c4-4ad8-be09-3787bf5adaf3 | ubuntu1-vm6           | ACTIVE | None       | Running     | vmnet=192.168.10.27 |
| 126b6a15-0412-4516-9bff-0c2e31ea0d7c | ubuntu1-vm7           | ACTIVE | None       | Running     | vmnet=192.168.10.28 |
| 8c2b1221-32f2-486f-8681-e76db0837207 | ubuntu1-vm8           | ACTIVE | None       | Running     | vmnet=192.168.10.29 |
| 479513ce-f357-4a75-9f23-91ebc5254883 | ubuntu19-resourcemgr  | ACTIVE | None       | Running     | vmnet=192.168.10.31 |
| 28a9ad5c-d444-44b0-918d-97a66fefeccd | ubuntu20-namenode     | ACTIVE | None       | Running     | vmnet=192.168.10.30 |
+--------------------------------------+-----------------------+--------+------------+-------------+--------------------+
```

10. After the instance boots and initializes and you have configured security groups, you can ssh into the CirrOS instance without a password by using the keypair you specified in the nova boot command. Use the nova list command to get the IP address for the instance. You do not need to specify the private key because it was stored in the default location, **~/.ssh/.id_rsa**, for the ssh client.

Note If using a CirrOS image to spawn an instance you must log in as the cirros, and not the root, user.You can also log in to the cirros account without an ssh key by using the cubswin password: $ ssh cirros@10.0.0.3.

VMs on each node can be instantiated following a script similar to this:

```
nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791-8552-905a79597523
--security_group default  --hint same_host=2a2955a1-3f6a-4c8b-bb43-56eeee04888b
ubuntu18-vm1
nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791-8552-905a79597523
--security_group default  --hint same_host=2a2955a1-3f6a-4c8b-bb43-56eeee04888b
ubuntu18-vm2
nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791-8552-905a79597523
--security_group default  --hint same_host=2a2955a1-3f6a-4c8b-bb43-56eeee04888b
ubuntu18-vm3
nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791-8552-905a79597523
--security_group default  --hint same_host=2a2955a1-3f6a-4c8b-bb43-56eeee04888b
ubuntu18-vm4
nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791-8552-905a79597523
--security_group default  --hint same_host=2a2955a1-3f6a-4c8b-bb43-56eeee04888b
ubuntu18-vm5
nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791-8552-905a79597523
--security_group default  --hint same_host=2a2955a1-3f6a-4c8b-bb43-56eeee04888b
ubuntu18-vm6
```

```
nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791-8552-905a79597523
--security_group default  --hint same_host=2a2955a1-3f6a-4c8b-bb43-56eeee04888b
ubuntu18-vm7
nova boot --flavor 8 --key_name mykey --image 93381385-53d5-4791-8552-905a79597523
--security_group default  --hint same_host=2a2955a1-3f6a-4c8b-bb43-56eeee04888b
ubuntu18-vm8
```

Openstack Dashboard (Horizon)

The OpenStack dashboard, also known as Horizon, is a Web interface that enables cloud administrators and users to manage various OpenStack resources and services.

The dashboard enables web-based interactions with the OpenStack Compute cloud controller through the OpenStack APIs.

Install and configure the dashboard on a node that can contact the Identity Service.

Provide users with the following information so that they can access the dashboard through a web browser on their local machine:

- The public IP address from which they can access the dashboard.
- The user name and password with which they can access the dashboard.

Your web browser, and that of your users, must support HTML5 and have cookies and JavaScript enabled.

Note To use the VNC client with the dashboard, the browser must support HTML5 Canvas and HTML5 WebSockets.

Installing the Dashboard

Following section details installing Horizon dashboard on the controller node:

1. Install the dashboard on the controller node (or a node that can contact the Identity Service as root)

```
# sudo apt-get -y install memcached libapache2-mod-wsgi openstack-dashboard
```

2. Ensure the value of **CACHES['default']['LOCATION']** in **/etc/openstack-dashboard/local_settings.py** match the ones set in **/etc/memcached.conf**.

```
Open /etc/openstack-dashboard/local_settings.py and look for this line:
CACHES = {
'default': {
'BACKEND' : 'django.core.cache.backends.memcached.MemcachedCache',
'LOCATION' : '127.0.0.1:11211'
}
}
```


Note The address and port must match the ones set in /etc/memcached.conf. If you change the memcached settings, you must restart the Apache web server for the changes to take effect. You can use options other than memcached option for session storage. Set the session back-end through the SESSION_ENGINE option. To change the timezone, use the dashboard or edit the /etc/openstack-dashboard/local_settings.py file. Change the following parameter: TIME_ZONE = "UTC".

3. Update the **ALLOWED_HOSTS in local_settings.py** to include the addresses you wish to access the dashboard from or restrict access from only these nodes.

   Edit **/etc/openstack-dashboard/local_settings.py**:

   ```
   ALLOWED_HOSTS = ['localhost', '<controller-node>']
   Set ALLOWED_HOSTS to '*' if you need to access the dashboard from any other node.
   ALLOWED_HOSTS = '*'
   ```

4. This CVD assumes that you are running the Dashboard on the controller node. You can easily run the dashboard on a separate server, by changing the appropriate settings in local_settings.py.

   Edit **/etc/openstack-dashboard/local_settings.py** and change OPENSTACK_HOST to the hostname of your Identity Service:

   ```
   OPENSTACK_HOST =”<controller>”
   ```

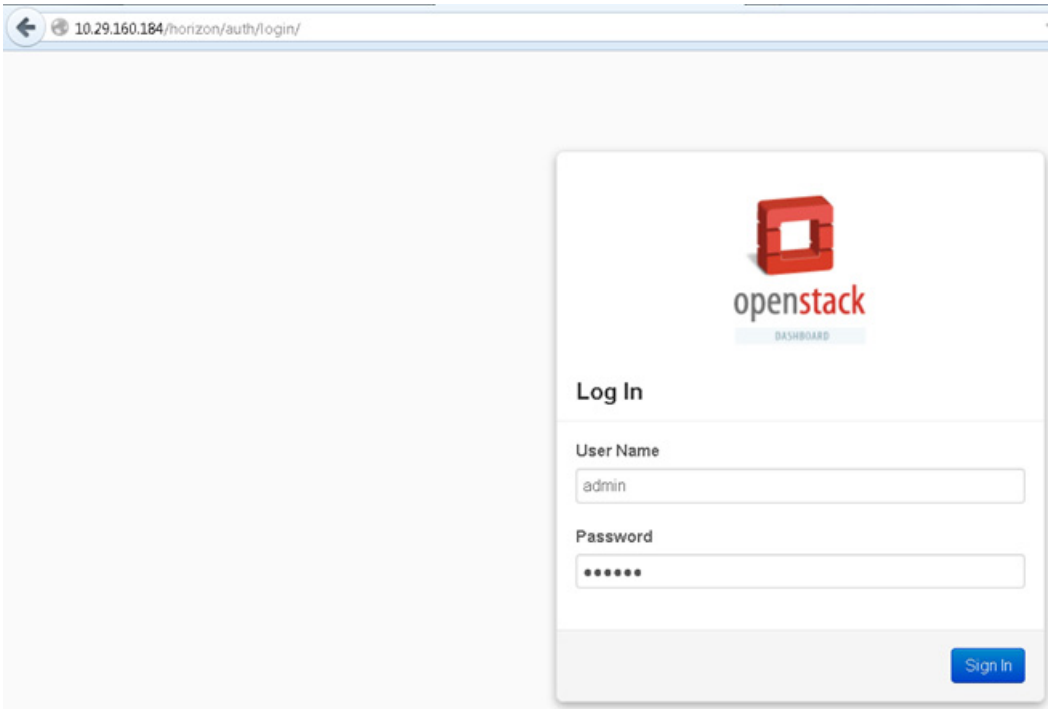
5. Start the Apache web server and memcached:

   ```
   # sudo service apache2 restart
   # sudo service memcached restart
   ```

6. You can now access the dashboard at http://<controller>/horizon.

   Login with credentials for any user that you created with the OpenStack Identity Service, for example, “admin/admin” .

Figure 90 OpenStack Login Page



Dashboard can be further tuned with different backend storage and database. For more information, see: http://docs.openstack.org/havana/install-guide/install/apt/content/dashboard-sessions.html

Allocated floating-ips to the VMs

Allocate floating-ips to the VMs launched in the previous section in-order to be able to access the Ubuntu VMs.

On the controller node run the following command:

```
$sudo nova-manage floating create --ip_range=<public-ip-range>/24 --pool
<pool-name>
```

```
ubuntu@ubuntu20:~$ sudo nova-manage floating create --ip_range=10.29.160.0/24  --pool public_Cisco
```

```
$nova floating-ip-pool-list
```

```
ubuntu@ubuntu20:~$ nova floating-ip-pool-list
+---------------+
| name          |
+---------------+
| public_Cisco  |
+---------------+
```

Go to the horizon page by accessing http://<controller-ip>/horizon

1. Login as admin/<password>

   Access Project > **Access & Security** > Click the **Floating IPs** tab.

2. Click **Allocate IP to Project**.
3. Select the Pool which was created above and Allocate IP.

Figure 91 Managing and Associating IP Address to the Instance

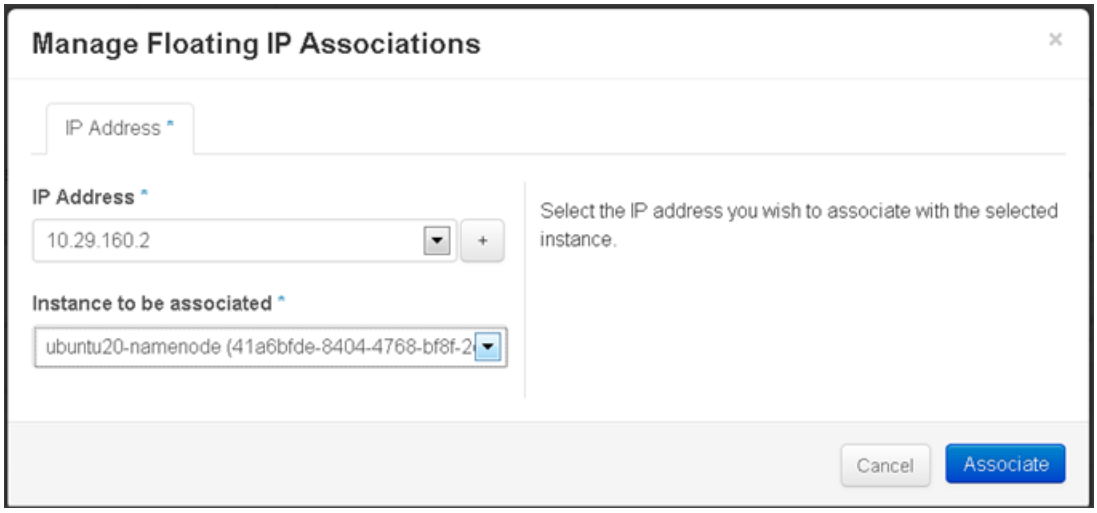


Figure 92 Associated Instances and Their IP Addresses

Instance Name	Image Name	IP Address	Size	Keypair	Status	Task	Power State
ubuntu19-resourcemgr	Ubuntu-Precise	192.168.10.23 10.29.160.3	hadoop.master | 220GB RAM | 16 VCPU | 50.0GB Disk	mykey	Active	None	Running
ubuntu20-namenode	Ubuntu-Precise	192.168.10.15 10.29.160.2	hadoop.master | 220GB RAM | 16 VCPU | 50.0GB Disk	mykey	Active	None	Running

4. To associate the Floating IP to the VM, choose the IP (click **Select** next to the IP) and in the Action plane, click **Associate**.
5. This opens Manage Floating IP Associations page. In this page choose the IP (already selected) and Instance VM to be associated with.
6. Once this is done, if you run the command **nova list** on the controller or in the Horizon page (**Project > Instances**), the VM Instance would have the floating IP associated with it.

Note If a floating ip has to be assigned for all VM (if required in some use-cases), this can be assigned to every VM at the time the VM is being created/spawned by having the following entry in nova.conf to avoid manually assigning floating-ip. This floating ip makes the VM accessible externally (through the internet), hence doing so would require the floating-ip range being assigned to be available (not used). This however may not be preferred for all cases (consider spawning 1000 VMs, then 1000 publicly IP addresses should be available) and this might not be needed for the use-case and here assigning floating-ip manually as and when needed might be ideal.

```
[Default]
…
default_floating_pool = public_Cisco
floating_range = 10.29.160.2/24
auto_assign_floating_ip=True
```

Accessing the VM created

To access the VM, the `mykey.pem` file is needed. This was created in the controller node and so the connection to VM is from the controller node (Copy this file to any other node if access is needed from other nodes).

Run the following set of commands:

```
cd ~/.ssh/
ssh-add mykey.pem

This throws message "Could not open connection to Authentication agent)
eval $(ssh-agent)
This throws message "Agent pid <number>"
chmod 600 mykey.pem
ssh-add mykey.pem
This would throw the message "Identity Added".

Now we can login to the VM as follows
```

```
ssh ubuntu@10.29.160.2 -i mykey.pem
```

```
ubuntu@ubuntu20:~$ cd ~/.ssh
ubuntu@ubuntu20:~/.ssh$ ssh-add mykey.pem
Could not open a connection to your authentication agent.
ubuntu@ubuntu20:~/.ssh$ eval $(ssh-agent)
Agent pid 12905
ubuntu@ubuntu20:~/.ssh$ chmod 600 mykey.pem
ubuntu@ubuntu20:~/.ssh$ ssh-add mykey.pem
Identity added: mykey.pem (mykey.pem)
ubuntu@ubuntu20:~/.ssh$ ssh ubuntu@10.29.160.2 -i mykey.pem
The authenticity of host '10.29.160.2 (10.29.160.2)' can't be established.
ECDSA key fingerprint is 31:ae:00:f2:e1:4d:be:23:6a:f2:0f:43:fd:a5:bc:48.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.29.160.2' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 12.04.4 LTS (GNU/Linux 3.2.0-60-virtual x86_64)
```

Add a password to be able to access directly from the dashboard horizon.

Pre-config of VM cluster for HDP Installation

Setting Up XFS Filesystem Master Nodes

Note On Namenode and Resource manager, the large 20TB filesystem is not mounted, as the VM doesn't have xfs filesystem installed by default. Following are the commands needed to run on Namenode and ResourceManager VM in-order to re-configure as xfs.

Run the following to make sure the partition is visible in the VM:

```
cat /proc/partitions
```

Install xfsprogs to run mkfs.xfs

```
sudo apt-get install xfsprogs
```

```
root@ubuntu20-namenode:/home/ubuntu# sudo apt-get install xfsprogs
```

```
sudo mkfs.xfs -i size=1024 /dev/vdb
sudo mount -t xfs -o allocsize=128m,noatime,nobarrier,nodiratime /dev/vdb /mnt
```

```
root@ubuntu20-namenode:/home/ubuntu# mkfs.xfs -i size=1024 /dev/vdb
meta-data=/dev/vdb               isize=1024   agcount=20, agsize=268435455 blks
         =                       sectsz=512   attr=2, projid32bit=0
data     =                       bsize=4096   blocks=5242880000, imaxpct=5
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0
log      =internal log           bsize=4096   blocks=521728, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
root@ubuntu20-namenode:/home/ubuntu# mount -t xfs -o allocsize=128m,noatime,nobarrier,nodiratime /dev/vdb /mnt/
```

Note If provided Ephemeral storage is less than 4TB, the storage will be automatically mounted as ext3 Filesystem to /mnt when the VM is spawned. If the ephemeral storage is more than 4TB, this won't be done and the above process has to be repeated for data/task VM as well to create the partition and mount it.

Setting Up Password-less Login within VMs

To access all the VMs instantiated from Namenode VM, we need to setup password-less login.

Login to the Namenode VM (running on controller node) as mentioned above and run the following commands:

1. Run the `ssh-keygen` command to create both public and private keys on the admin node.

```
ubuntu@ubuntu20-namenode:~/.ssh$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
/home/ubuntu/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa.
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub.
The key fingerprint is:
a9:53:30:19:cc:c3:0f:59:52:48:02:7f:5f:5f:64:8e ubuntu@ubuntu20-namenode
The key's randomart image is:
+--[ RSA 2048]----+
|  ...=+=o     o  |
|   . .O+     =   |
|    . =+  . E o  |
|     . +.o . .   |
|       S   .     |
|        o        |
|       o         |
|        .        |
|                 |
+-----------------+
```

2. As done above, copy the mykey.pem into ~/.ssh/ only the first time to the namenode VM and run the following from ~/.ssh/

```
sudo chown ubuntu:ubuntu mykey.pem
ssh-add mykey.pem

This throws message "Could not open connection to Authentication agent)
eval $(ssh-agent)
This throws message "Agent pid <number>"
chmod 600 mykey.pem
ssh-add mykey.pem
This would throw the message "Identity Added".
```

3. Run a similar command from the admin node/Namenode VM to copy the public key **id_rsa.pub** to all the nodes of the cluster, since the VMs are all dhcp IP, make sure the IP range is correct. **ssh-copy-id** appends the keys to the remote-host's **.ssh/authorized_key**.

```
for IP in {24..167}; do echo -n "$IP -> "; ssh-copy-id -i ~/.ssh/id_rsa.pub
192.168.10.$IP; done
```

```
Enter yes for Are you sure you want to continue connecting (yes/no)?
```

Note This command does not ask for password as we have already added the password in the console session through mykey.pem.

Clush

Install clush on the Admin VM (VM on controller node) as follows:

```
sudo apt-get install clustershell
```

Update clustershell config to identify all nodes in the cluster

```
$ cat /etc/clustershell/groups
all: 192.168.10[24-167]
```

```
root@ubuntu20-namenode:~# cat /etc/clustershell/groups
all: 192.168.10.[24-167]
```

Note Provide “all” in the above file relates to all nodes to be included in “-a” option for clush.

Ensure that clush is working fine by running the command **clush -a -b pwd**.

/etc/hosts

Update /etc/hosts file on all VMs. From OpenStack, get the VM and their ips as follows:

```
nova list | grep ubuntu | grep -v cirros | awk '{print $12,"   ",$4}' | sed
's/vmnet=//g' | sed 's/,//g' > hosts-vm
```

Copy these vms on hosts-vm to /etc/hosts and copy this to all nodes

```
clush -b -a -c /etc/hosts --dest=/home/ubuntu
clush -b -a sudo mv /home/ubuntu/hosts /etc/
clush -b -a cat /etc/hosts
```

Install NTP on all VMs

Install NTP on all VMs as follows:

```
clush -a -b sudo apt-get –y install ntp
```

The Network Time Protocol (NTP) is used to synchronize the time of all the nodes within the cluster. The Network Time Protocol daemon (ntpd) sets and maintains the system time of day in synchronism with the timeserver located in the admin node (rhel1). Configuring NTP is critical for any Hadoop Cluster. If server clocks in the cluster drift out of sync, serious problems will occur with HBase and other services.

Note Installing an internal NTP server keeps your cluster synchronized even when an external NTP server is inaccessible.

```
Configure /etc/ntp.conf on the admin node with the following contents:
```

```
Sudo vi /etc/ntp.conf
driftfile /var/lib/ntp/drift
restrict 127.0.0.1
restrict -6 ::1
server 127.127.1.0
fudge 127.127.1.0 stratum 10
includefile /etc/ntp/crypto/pw
keys /etc/ntp/keys

Create /home/ubuntu/ntp.conf on the admin node and copy it to all nodes
Sudo vi /home/ubuntu /ntp.conf
server 192.168.10.15    ç Admin VM ip
driftfile /var/lib/ntp/drift
restrict 127.0.0.1
restrict -6 ::1
includefile /etc/ntp/crypto/pw
keys /etc/ntp/keys
```

Install ntp on all the nodes by running the following command:

```
clush –b –a sudo apt-get –y install ntp

clush -b -a sudo service ntp status
---------------
ubuntu[1-20] (20)
---------------
 * NTP server is running
```

Copy ntp.conf file from the admin node to **/etc** of all the nodes by executing the following command in the admin node:

```
clush –b –w 192.168.10.[23-167] -c /home/ubuntu/ntp.conf \
--dest=/home/ubuntu/
clush –b –w 192.168.10.[23-167]
sudo mv /home/ubuntu/ntp.conf /etc/
```

Restart NTP on all the nodes including the admin node:

```
clush –b –a sudo service ntp restart
```

HDP 2.0 Repo for Ubuntu on all VMs

To install HDP 2.0 repo for Ubuntu on all the VMs, run these commands:

```
clush -a -b sudo wget \
http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x/hdp.list -O
/etc/apt/sources.list.d/hdp.list
```

```
ubuntu@ubuntu20-namenode:~$ clush -a -b sudo wget http://public-repo-1.hortonwor
ks.com/HDP/ubuntu12/2.x/hdp.list -O /etc/apt/sources.list.d/hdp.list
ubuntu@ubuntu20-namenode:~$ clush -a -b sudo cat /etc/apt/sources.list.d/hdp.list
---------------
192.168.10.100,192.168.10.101,192.168.10.102,192.168.10.103,192.168.10.104,192.168.10.105
108,192.168.10.109,192.168.10.110,192.168.10.111,192.168.10.112,192.168.10.113,192.168.10
.10.117,192.168.10.118,192.168.10.119,192.168.10.120,192.168.10.121,192.168.10.122,192.16
.168.10.126,192.168.10.127,192.168.10.128,192.168.10.129,192.168.10.130,192.168.10.131,19
,192.168.10.135,192.168.10.136,192.168.10.137,192.168.10.138,192.168.10.139,192.168.10.14
.143,192.168.10.144,192.168.10.145,192.168.10.146,192.168.10.147,192.168.10.148,192.168.1
.10.151,192.168.10.152,192.168.10.153,192.168.10.154,192.168.10.155,192.168.10.156,192.16
.168.10.160,192.168.10.161,192.168.10.162,192.168.10.163,192.168.10.164,192.168.10.165,19
192.168.10.24,192.168.10.25,192.168.10.26,192.168.10.27,192.168.10.28,192.168.10.29,192.1
68.10.33,192.168.10.34,192.168.10.35,192.168.10.36,192.168.10.37,192.168.10.38,192.168.10
.42,192.168.10.43,192.168.10.44,192.168.10.45,192.168.10.46,192.168.10.47,192.168.10.48,1
92.168.10.52,192.168.10.53,192.168.10.54,192.168.10.55,192.168.10.56,192.168.10.57,192.16
8.10.61,192.168.10.62,192.168.10.63,192.168.10.64,192.168.10.65,192.168.10.66,192.168.10.
70,192.168.10.71,192.168.10.72,192.168.10.73,192.168.10.74,192.168.10.75,192.168.10.76,19
2.168.10.80,192.168.10.81,192.168.10.82,192.168.10.83,192.168.10.84,192.168.10.85,192.168
.10.89,192.168.10.90,192.168.10.91,192.168.10.92,192.168.10.93,192.168.10.94,192.168.10.9
8,192.168.10.99 (146)
---------------
deb http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.16/repos/ubuntu12 HDP-UTILS main
deb http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x HDP main
```

Ensure the source list is properly set.

Add gpg keys for all VMs

For Each Ubuntu hosts, add the gpg keys as the root user (ensure the commands are run as “sudo su”).

```
clush -a -b sudo gpg --keyserver pgp.mit.edu --recv-keys B9733A7A07513CAD
clush -a -b "sudo gpg -a --export 07513CAD | sudo apt-key add -"
clush -a -b sudo apt-get update
```

```
root@ubuntu20-namenode:~# clush -a -b sudo "gpg -a --export 07513CAD | apt-key add -"
---------------
192.168.10.100,192.168.10.101,192.168.10.102,192.168.10.103,192.168.10.104,192.168.10
108,192.168.10.109,192.168.10.110,192.168.10.111,192.168.10.112,192.168.10.113,192.16
.10.117,192.168.10.118,192.168.10.119,192.168.10.120,192.168.10.121,192.168.10.122,19
.168.10.126,192.168.10.127,192.168.10.128,192.168.10.129,192.168.10.130,192.168.10.13
,192.168.10.135,192.168.10.136,192.168.10.137,192.168.10.138,192.168.10.139,192.168.1
.143,192.168.10.144,192.168.10.145,192.168.10.146,192.168.10.147,192.168.10.148,192.1
.10.151,192.168.10.152,192.168.10.153,192.168.10.154,192.168.10.155,192.168.10.156,19
.168.10.160,192.168.10.161,192.168.10.162,192.168.10.163,192.168.10.164,192.168.10.16
192.168.10.24,192.168.10.25,192.168.10.26,192.168.10.27,192.168.10.28,192.168.10.29,1
68.10.33,192.168.10.34,192.168.10.35,192.168.10.36,192.168.10.37,192.168.10.38,192.16
.42,192.168.10.43,192.168.10.44,192.168.10.45,192.168.10.46,192.168.10.47,192.168.10.
92.168.10.52,192.168.10.53,192.168.10.54,192.168.10.55,192.168.10.56,192.168.10.57,19
8.10.61,192.168.10.62,192.168.10.63,192.168.10.64,192.168.10.65,192.168.10.66,192.168
70,192.168.10.71,192.168.10.72,192.168.10.73,192.168.10.74,192.168.10.75,192.168.10.7
2.168.10.80,192.168.10.81,192.168.10.82,192.168.10.83,192.168.10.84,192.168.10.85,192
.10.89,192.168.10.90,192.168.10.91,192.168.10.92,192.168.10.93,192.168.10.94,192.168.
8,192.168.10.99 (146)
---------------
OK
```

Fully Qualified domain name (FQDN)

Make sure the fully qualified domain name (FQDN) for each VM host in set properly by running the following command:

```
clush –a hostname -f
```

```
root@ubuntu20-namenode:~# clush -a hostname -f
192.168.10.27: ubuntu1-vm4.novalocal
192.168.10.34: ubuntu2-vm3.novalocal
192.168.10.36: ubuntu2-vm5.novalocal
192.168.10.35: ubuntu2-vm4.novalocal
192.168.10.37: ubuntu2-vm6.novalocal
192.168.10.33: ubuntu2-vm2.novalocal
192.168.10.38: ubuntu2-vm7.novalocal
192.168.10.25: ubuntu1-vm2.novalocal
192.168.10.26: ubuntu1-vm3.novalocal
192.168.10.49: ubuntu3-vm2.novalocal
192.168.10.31: ubuntu1-vm8.novalocal
192.168.10.29: ubuntu1-vm6.novalocal
```

Domain Name Server (DNS)

Make sure the domain name server (DNS) is configured for both forward and reverse lookup.

```
clush –a –b cat /etc/resolv.conf
```

Forward lookup

```
nslookup ubuntu20-namenode
```

Reverse lookup

```
nslookup <ip-of- ubuntu20-namenode>
```

A list of VM ip and their hostname can be generated by running the following command:

```
clush –a echo "\`hostname -i\`   \`hostname\`" | awk '{print $2,"   ",$3}'
```

```
ubuntu@ubuntu20-namenode:~$ nslookup ubuntu20-namenode
Server:         192.168.10.1
Address:        192.168.10.1#53

Name:   ubuntu20-namenode.novalocal
Address: 192.168.10.15
```

Disable SELinux

If SELinux is installed, disable SELinux:

```
clush –a –b "sudo sed -i 's/SELINUX=enforcing/SELINUX=disabled/g'
/etc/selinux/config"
```

Disable IPTables

On Ubuntu VMs, execute the following command to disable IPtables:

```
clush -a -b sudo service ufw stop
```

Install MySQL (resourcemanager VM)

MySQL database instance is needed to store metadata information for Hive and HCatalog services. You can either use an existing MySQL instance or install a new instance of MySQL manually. To install a new instance:

1. Connect to the Resource Manager VM that will be used for Hive and HCatalog (Of the two VMs running Master services one is Namenode VM and the other is Resource-manager VM).
2. Install MySQL server. From a terminal window, type:

```
sudo apt-get -y install mysql-server
```

```
root@ubuntu19-resourcemgr:~# sudo apt-get -y install mysql-server
```

This prompts for mysql root password

3. Start the instance.

```
sudo service mysql start
```

4. Mysql password can be changed as follows:

```
mysqladmin -u root -p`{password}' password $mysqlpassword
```

5. Remove unnecessary information from log and STDOUT.

```
mysqladmin -u root 2>&1 >/dev/null
```

```
root@ubuntu19-resourcemgr:~# mysqladmin -u root 2>&1 >/dev/null
```

6. As root, use mysql (or other client tool) to create the <dbuser>, 'cisco' in this CVD and grant it adequate privileges. This user provides access to the Hive metastore.

```
mysql -u root -p
CREATE USER 'cisco'@'localhost' IDENTIFIED BY 'cisco';
GRANT ALL PRIVILEGES ON *.* TO 'cisco'@'localhost';
CREATE USER 'cisco'@'%' IDENTIFIED BY 'cisco';
GRANT ALL PRIVILEGES ON *.* TO 'cisco'@'%'; FLUSH PRIVILEGES;
```

```
mysql> CREATE USER 'cisco'@'localhost' IDENTIFIED BY 'cisco'; GRANT ALL PRIVILEGES ON *.* TO 'cisco'@'localhost
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

mysql> CREATE USER 'cisco'@'%' IDENTIFIED BY 'cisco';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'cisco'@'%'; FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

7. See if you can connect to the database as the user. You are prompted to enter the password. Enter $dbuserpassword.

```
mysql -u <username > -p
```

```
root@ubuntu19-resourcemgr:~# mysql -u cisco -p cisco
Enter password:
```

8. Install the MySQL connector JAR file for Ubuntu:

```
sudo apt-get -y install mysql-connector-java*
```

```
root@ubuntu19-resourcemgr:~# sudo apt-get -y install mysql-connector-java*
Reading package lists... Done
Building dependency tree
Reading state information... Done
```

Install Java

Install Java (openjdk 1.7) on all VM nodes as follows:

```
clush -a -b sudo apt-get -y install openjdk-7-jdk
clush -a -B java -version
```

Openjdk 1.7 would be installed in /usr/lib/jvm/java-1.7.0-openjdk-amd64. This has to be set as JAVA_HOME.

Create symbolic links (symlinks) to the JDK:

```
clush -a -b sudo mkdir /usr/java
clush -a -b sudo ln -s /usr/lib/jvm/java-1.7.0-openjdk-amd64 /usr/java/default
clush -a -b sudo ln -s /usr/java/default/bin/java /usr/bin/java
```

Add the following in .bash_profile and copy to both ubuntu and root home:

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH

clush -a -b -c /home/ubuntu/.bash_profile --dest=/home/ubuntu
clush -a -b  sudo cp /home/Ubuntu/.bash_profile /root/
clush -a -b source /home/ubuntu/.bash_profile
```

Install Apache2

Install Apache2 on Namenode:

```
sudo apt-get -y install apache2
```

Installing HDP 2.0

HDP 2.0 is an enterprise grade, hardened Hadoop distribution. HDP combines Apache Hadoop and its related projects into a single tested and certified package. It offers the latest innovations from the open source community with the testing and quality you expect from enterprise quality software.

This section details installing Hadoop 2.0 manually on the VMs (which are currently running 8VM per node and two additional VM for Namenode and Resource Manager/Secondary namenode).

Note As of this version, Hadoop 2.0 installation/deployment is not supported on Ubuntu through Ambari and hence the installation is manual. As and when Ambari is supported on Ubuntu, it is the preferred way of installing HDP 2.0.

Download Companion Files

To install hadoop manually download the companion files, which includes script files and configuration files, and use throughout this process. Download and extract the files on all VMs of Hadoop Cluster (all modifications to the hadoop config files will be done mostly in Namenode/Admin VM and copied to all other VMs from the Namenode VM).

```
clush -a -b wget
http://public-repo-1.hortonworks.com/HDP/tools/2.0.6.0/hdp_manual_install_rpm_helper_f
iles-2.0.6.101.tar.gz
```

Unzip and untar the above downloaded file

```
clush -a -b gunzip /home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101.tar.gz

clush -a -b tar -xvf /home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101.tar
```

This has scripts to setup and install hadoop.

```
root@ubuntu20-namenode:/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101# pwd
/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101
root@ubuntu20-namenode:/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101# ls
configuration_files  HDP-CHANGES.txt  readme.txt  scripts
root@ubuntu20-namenode:/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101# ls scripts/
directories.sh  usersAndGroups.sh  yarn-utils.py
root@ubuntu20-namenode:/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101# cd ..
root@ubuntu20-namenode:/home/ubuntu# cd hdp_manual_install_rpm_helper_files-2.0.6.101
root@ubuntu20-namenode:/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101# ls configuration_file
core_hadoop  ganglia  hbase  hive  nagios  oozie  pig  sqoop  webhcat  zookeeper
root@ubuntu20-namenode:/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101#
```

In directory hdp_manual_install_rpm_helper_files-2.0.6.101/scripts/ under hdp_manual_install_rpm_helper_files-2.0.6.101, update the file **directory.sh** pointing to mounted filesystem as follows. Update this on the namenode and copy the same on all the VMs.

```
DFS_NAME_DIR="/mnt/hadoop/hdfs/nn";
DFS_DATA_DIR="/mnt/hadoop/hdfs/dn";
FS_CHECKPOINT_DIR="/mnt/hadoop/hdfs/snn";
YARN_LOCAL_DIR="/mnt/hadoop/yarn/local";
YARN_LOCAL_LOG_DIR="/mnt/hadoop/yarn/logs";
ZOOKEEPER_DATA_DIR="/mnt/hadoop/zookeeper/data";
```

```
root@ubuntu20-namenode:grep mnt directories.sh
DFS_NAME_DIR="/mnt/hadoop/hdfs/nn";
DFS_DATA_DIR="/mnt/hadoop/hdfs/dn";
FS_CHECKPOINT_DIR="/mnt/hadoop/hdfs/snn";
YARN_LOCAL_DIR="/mnt/hadoop/yarn/local";
YARN_LOCAL_LOG_DIR="/mnt/hadoop/yarn/logs";
ZOOKEEPER_DATA_DIR="/mnt/hadoop/zookeeper/data";
```

Copy the file directories.sh to all.

```
clush -b -w 192.168.10.[23-167] -c ./directories.sh
--dest=/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101/scripts/
```

Add the two files in the bash_profile so the value is set for all the nodes. Add this in namenode and copy to all nodes. File **directories.sh** as mentioned above has location for the directories for the **services** and **usersAndGroups.sh** script defines all the users, which will be created later.

```
cat ~/.bash_profile
source
/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101/scripts/directories.sh
source
/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101/scripts/usersAndGroups.sh
umask 0022
```

```
ubuntu@ubuntu20-namenode:clush -a -b cat /home/ubuntu/.bash_profile
---------------
192.168.10.100,192.168.10.101,192.168.10.102,192.168.10.103,192.168.10.104,192.168.10.105,1
108,192.168.10.109,192.168.10.110,192.168.10.111,192.168.10.112,192.168.10.113,192.168.10.1
.10.117,192.168.10.118,192.168.10.119,192.168.10.120,192.168.10.121,192.168.10.122,192.168.
.168.10.126,192.168.10.127,192.168.10.128,192.168.10.129,192.168.10.130,192.168.10.131,192.
,192.168.10.135,192.168.10.136,192.168.10.137,192.168.10.138,192.168.10.139,192.168.10.140,
.143,192.168.10.144,192.168.10.145,192.168.10.146,192.168.10.147,192.168.10.148,192.168.10.
.10.151,192.168.10.152,192.168.10.153,192.168.10.154,192.168.10.155,192.168.10.156,192.168.
.168.10.160,192.168.10.161,192.168.10.162,192.168.10.163,192.168.10.164,192.168.10.165,192.
192.168.10.24,192.168.10.25,192.168.10.26,192.168.10.27,192.168.10.28,192.168.10.29,192.168
68.10.33,192.168.10.34,192.168.10.35,192.168.10.36,192.168.10.37,192.168.10.38,192.168.10.3
.42,192.168.10.43,192.168.10.44,192.168.10.45,192.168.10.46,192.168.10.47,192.168.10.48,192
92.168.10.52,192.168.10.53,192.168.10.54,192.168.10.55,192.168.10.56,192.168.10.57,192.168.
8.10.61,192.168.10.62,192.168.10.63,192.168.10.64,192.168.10.65,192.168.10.66,192.168.10.67
70,192.168.10.71,192.168.10.72,192.168.10.73,192.168.10.74,192.168.10.75,192.168.10.76,192.
2.168.10.80,192.168.10.81,192.168.10.82,192.168.10.83,192.168.10.84,192.168.10.85,192.168.1
.10.89,192.168.10.90,192.168.10.91,192.168.10.92,192.168.10.93,192.168.10.94,192.168.10.95,
8,192.168.10.99 (146)
---------------
source /home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101/scripts/directories.sh
source /home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101/scripts/usersAndGroups.sh
umask 0022
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
export PATH=$JAVA_HOME/bin:$PATH
```

Note Umask should be set to 0022 as explained in "Set Default File and Directory Permissions" section on page 147.

```
clush -b -w 192.168.10.[23-167] -c ~/.bash_profile --dest=/home/ubuntu/
clush -b -a source /home/ubuntu/.bash_profile
```

This can be verified by running:

```
clush -b -a echo $HDFS_USER:$HADOOP_GROUP
```

To create Hadoop Users on all the VMs, run the following script:

Create script **hadoop_users.sh** as follows in /home/ubuntu on Namenode VM to create the users need for HDP services.

```
#!/bin/bash

echo "Create group hadoop and users  on all nodes"
clush -a -B sudo groupadd $HADOOP_GROUP
sleep 1
clush -a -B sudo useradd -G $HADOOP_GROUP $HDFS_USER
```

```
sleep 1
clush -a -B sudo useradd -G $HADOOP_GROUP $YARN_USER
sleep 1
clush -a -B sudo useradd -G $HADOOP_GROUP $MAPRED_USER
sleep 1
clush -a -B sudo useradd -G $HADOOP_GROUP $PIG_USER
sleep 1
clush -a -B sudo useradd -G $HADOOP_GROUP $HIVE_USER
sleep 1
clush -a -B sudo useradd -G $HADOOP_GROUP $WEBHCAT_USER
sleep 1
clush -a -B sudo useradd -G $HADOOP_GROUP $HBASE_USER
sleep 1
clush -a -B sudo useradd -G $HADOOP_GROUP $ZOOKEEPER_USER
sleep 1
```

Run the script from the namenode.

```
sudo chmod 755 hadoop_users.sh
./hadoop_users.sh
```

Role Assignment – Masters/Slaves

This section defines assignment of master services and client services for various VMs that have been selected to appropriate hosts in the cluster. Master services (namenode/resourcemgr/etc) are all assigned to special VMs defined in flavors with more resources. The right column shows the current service assignments by VM.

Reconfigure the service assignment to match Table 9:

Table 9 Service Assignments

Service Name	Virtual Machine
Namenode	namenode-vm
SNamenode	resource-mgr-vm
History Server	resource-mgr-vm
Resouce Manager	resource-mgr-vm
Nagios Server	namenode-vm
Ganglia Collector	namenode-vm
Hive Server2	resource-mgr-vm
HBase Master	resource-mgr-vm
Oozie Server	namenode-vm
ZooKeeper	namenode-vm, resource-mgr-vm, ubuntu1-vm1
Datanode/Tasktrackers	All VMs other than namenode-vm and resource-mgr-vm
NodeManager	All VMs other than namenode-vm and resource-mgr-vm
Region Server	All VMs other than namenode-vm and resource-mgr-vm
Client	All VMs

Create Hadoop Directories

For manually installing HDP 2.0 on the VMs, directories need to be created for different services and appropriate permissions and ownership needs to be set.

Create the NameNode Directories

On the node that hosts the NameNode service, execute the following commands:

```
echo "Create Namenode local dir"
sudo mkdir -p $DFS_NAME_DIR;
sudo chown -R $HDFS_USER:$HADOOP_GROUP $DFS_NAME_DIR;
sudo chmod -R 750 $DFS_NAME_DIR;
```

Create the SecondaryNameNode Directories

Create the directories of Secondary Namenode service on resource-mgr-vm by executing the following commands on Namenode.

```
echo "Create Secondary Namenode local dir"
clush -B -w 192.168.10.23 sudo mkdir -p $FS_CHECKPOINT_DIR;
clush -B -w 192.168.10.23 sudo chown -R $HDFS_USER:$HADOOP_GROUP $FS_CHECKPOINT_DIR;
clush -B -w 192.168.10.23 sudo chmod -R 750 $FS_CHECKPOINT_DIR;
```

Create DataNode and YARN NodeManager Local Directories

Create the directories of Datanodes by executing the following commands on Namenode.

```
echo "Create datanode local dir"
clush -B -a sudo mkdir -p $DFS_DATA_DIR;
clush -B -a sudo chown -R $HDFS_USER:$HADOOP_GROUP $DFS_DATA_DIR;
clush -B -a sudo chmod -R 750 $DFS_DATA_DIR;

echo "Create yarn local dir"
clush -B -a sudo mkdir -p $YARN_LOCAL_DIR;
clush -B -a sudo chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_DIR;
clush -B -a sudo chmod -R 755 $YARN_LOCAL_DIR;

echo "Create yarn local log dir"
clush -B -a sudo mkdir -p $YARN_LOCAL_LOG_DIR;
clush -B -a  sudo chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_LOG_DIR;
clush -B -a sudo chmod -R 755 $YARN_LOCAL_LOG_DIR;
```

Create the Log and PID Directories

Similarly, run the following commands on the namenode:

```
clush -B -a sudo mkdir -p $HDFS_LOG_DIR;
clush -B -a sudo chown -R $HDFS_USER:$HADOOP_GROUP $HDFS_LOG_DIR;
clush -B -a sudo chmod -R 755 $HDFS_LOG_DIR;

clush -B -a sudo mkdir -p $YARN_LOG_DIR;
clush -B -a sudo chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOG_DIR;
clush -B -a sudo chmod -R 755 $YARN_LOG_DIR;

clush -B -a sudo mkdir -p $HDFS_PID_DIR;
clush -B -a sudo chown -R $HDFS_USER:$HADOOP_GROUP $HDFS_PID_DIR;
```

```
clush -B -a sudo chmod -R 755 $HDFS_PID_DIR

clush -B -a sudo mkdir -p $YARN_PID_DIR;
clush -B -a sudo chown -R $YARN_USER:$HADOOP_GROUP $YARN_PID_DIR;
clush -B -a sudo chmod -R 755 $YARN_PID_DIR;

clush -B -a sudo mkdir -p $MAPRED_LOG_DIR;
clush -B -a sudo chown -R $MAPRED_USER:$HADOOP_GROUP $MAPRED_LOG_DIR;
clush -B -a sudo chmod -R 755 $MAPRED_LOG_DIR;

clush -B -a sudo mkdir -p $MAPRED_PID_DIR;
clush -B -a sudo chown -R $MAPRED_USER:$HADOOP_GROUP $MAPRED_PID_DIR;
clush -B -a sudo chmod -R 755 $MAPRED_PID_DIR;
```

Validating Directories

```
#Ensure to evaluate directories, owners and permissions
echo "Validate directories"
sudo ls -ld $DFS_NAME_DIR
clush -B -a sudo ls -ld $YARN_LOCAL_LOG_DIR
clush -B -a sudo ls -ld $YARN_LOCAL_DIR
clush -B -a sudo ls -ld $DFS_DATA_DIR
clush -B -w 192.168.10.[15,23,24] sudo ls -ld $ZOOKEEPER_DATA_DIR
clush -B -w 192.168.10.23 sudo ls -ld $FS_CHECKPOINT_DIR
clush -B -a sudo ls -ld $MAPRED_PID_DIR
clush -B -a sudo ls -ld $MAPRED_LOG_DIR
clush -B -a sudo ls -ld $YARN_PID_DIR
clush -B -a sudo ls -ld $HDFS_PID_DIR
clush -B -a sudo ls -ld $YARN_LOG_DIR
clush -B -a sudo ls -ld $HDFS_LOG_DIR
```

Determine YARN and MapReduce Memory Configuration Settings

Memory configurations can be manually calculated by running the yarn-util script provided in companion files as follows:

```
$ cd /home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101/scripts
```

Given the flavor of each Datanode/Tasktracker VM with 27GB Memory, 2 Cores and 2TB disk space.

```
$ python yarn-utils.py -c 2 -m 27 -d 2 -k True
```


Note The above util gives different values and the util is not updated. The following values will be used for this CVD given 8VMs. For 4VM, only the two values below need to be doubled, the rest of the configuration parameters would be unchanged and if using 2VM, similarly as done in 4VM, the two values used in 4VM would be doubled. Care should be taken to not use the same flavor as used for 8VM when testing 4VM as for 4VM flavor can use more CPU cores, RAM, storage.

```
yarn.scheduler.maximum-allocation-mb=24576
yarn.nodemanager.resource.memory-mb=24576
```

Following will be the memory configuration used:

```
yarn.scheduler.minimum-allocation-mb=3072
yarn.scheduler.maximum-allocation-mb=24576
yarn.nodemanager.resource.memory-mb=24576
mapreduce.map.memory.mb=3072
mapreduce.map.java.opts=-Xmx5500m
mapreduce.reduce.memory.mb=3072
```

```
mapreduce.reduce.java.opts=-Xmx5500m
yarn.app.mapreduce.am.resource.mb=6144
yarn.app.mapreduce.am.command-opts=-Xmx5500m
mapreduce.task.io.sort.mb=1024
mapred.child.java.opts=1024
```

Set Default File and Directory Permissions

Set the default file and directory permissions to 0022 (022). This is the default setting typically for most Linux distributions.

Make sure `umask 0022` is set in .bash_profile as mentioned in the above section. This ensures that the `umask` is set for all terminal sessions that you will use during the installation.

HDP Installation

This section provides information on installing all the necessary hadoop services.

Install the Hadoop Packages

Install the following services on all the cluster VM nodes by running the command on the Namenode VM:

```
clush -a -B sudo apt-get -y install hadoop hadoop-hdfs libhdfs0 libhdfs0-dev
hadoop-yarn hadoop-mapreduce hadoop-client openssl
```

```
buntu@ubuntu20-namenode:clush -a -B sudo apt-get -y install hadoop hadoop-hdfs
ibhdfs0 libhdfs0-dev hadoop-yarn hadoop-mapreduce hadoop-client openssl
lush:   0/146
```

Install Compression Libraries

Follow these steps to make the compression libraries available on all the cluster nodes:

Install Snappy

Complete the following instructions on all the nodes in your cluster:

```
clush -a -B sudo apt-get -y install libsnappy1 libsnappy-dev
```

Install LZO

Execute the following command on all the nodes in your cluster:

```
clush -a -B sudo apt-get -y install liblzo2-2 liblzo2-dev hadoop-lzo
```

Hadoop Configuration

The companion files in

/home/ubuntu/hdp_manual_install_rpm_helper_files-2.0.6.101/configuration_files need to be edited and used in the configuration files for HDFS and MapReduce In directory **/etc/hadoop/conf/core-hadoop.**

Slaves

Create /etc/hadoop/conf/slaves file in namenode VM with the ip of all datanodes. A simple way to get all the ips is to run a gerp command on controller node and filter out all other VMs created such as cirros, namenode VM, resourcemanager VM as shown in the example.

```
nova list | grep ubuntu | grep -v ubuntu19- | grep -v ubuntu20- | grep -v cirros | awk
'{print $4}' ' > slaves
```

Ensure all the needed hostnames are present

```
wc -l slaves
```

Copy the contents of slaves to Namenode VM at **/etc/hadoop/conf/slaves**.

Note Slaves should be hostnames of VMs. Giving IP addresses instead of hostnames will not work.

In Namenode **/etc/hadoop/conf/hadoop-env.sh**, modify **HADOOP_SLAVES** with the value:

```
export HADOOP_SLAVES=/etc/hadoop/conf/slaves
```

core-site.xml

Modify the following properties as:

```
<property>
 <name>fs.defaultFS</name>
 <value>hdfs://<$namenode.full.hostname/ip-address>:8020</value>
 <description>Enter your NameNode hostname</description>
</property>

<property>
 <name>fs.defaultFS</name>
 <value>hdfs://ubuntu20-namenode.novalocal:8020</value>
</property>
```

hdfs-site.xml

Update TODO fields as follows:

Note 192.168.10.15 is IP of Namenode and 192.168.10.23 is IP of Secondary Namenode/ResourceManager.

```
<property>
    <name>dfs.namenode.name.dir</name>
    <value>/mnt/hadoop/hdfs/nn</value>
  </property>

 <property>
    <name>dfs.datanode.data.dir</name>
    <value>/mnt/hadoop/hdfs/dn</value>
  </property>

 <property>
    <name>dfs.namenode.http-address</name>
    <value>ubuntu20-namenode.novalocal:50070</value>
  </property>
```

```
  <property>
    <name>dfs.namenode.secondary.http-address</name>
    <value>ubuntu19-resourcemgr.novalocal:50090</value>
  </property>

  <property>
    <name>dfs.namenode.checkpoint.dir</name>
    <value>/mnt/hadoop/hdfs/snn</value>
  </property>
```

yarn-site.xml

```
<property>
 <name>yarn.resourcemanager.resource-tracker.address</name>
 <value> ubuntu19-resourcemgr.novalocal:8025</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.scheduler.address</name>
 <value>192.168.10.23:8030</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.address</name>
 <value>192.168.10.23:8050</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.admin.address</name>
 <value>192.168.10.23:8141</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.nodemanager.local-dirs</name>
 <value/mnt/hadoop/yarn/logs</value>
 <description>Comma separated list of paths. Use the list of directories from
$YARN_LOCAL_DIR.
               For example,
/grid/hadoop/hdfs/yarn/local,/grid1/hadoop/hdfs/yarn/local.</description>
</property>

<property>
 <name>yarn.nodemanager.log-dirs</name>
 <value>/mnt/hadoop/hdfs/dn</value>
 <description>Use the list of directories from $YARN_LOCAL_LOG_DIR.
       For example, /grid/hadoop/yarn/logs
   /grid1/hadoop/yarn/logs/grid2/hadoop/yarn/logs</description>
</property>

<property>
 <name>yarn.log.server.url</name>
 <value>http://192.168.10.23:19888/jobhistory/logs/</value>
 <description>URL for job history server</description>
</property>

<property>
 <name>yarn.resourcemanager.webapp.address</name>
 <value>192.168.10.23:8088</value>
```

```
 <description>URL for job history server</description>
</property>
```

mapred-site.xml

Modify the following properties:

```
<property>
 <name>mapreduce.jobhistory.address</name>
 <value>192.168.10.23:10020</value>
 <description>Enter your JobHistoryServer hostname.</description>
</property>

<property>
 <name>mapreduce.jobhistory.webapp.address</name>
 <value>192.168.10.23:19888</value>
 <description>Enter your JobHistoryServer hostname.</description>
</property>
```

Update Memory Configuration in yarn-site.xml and mapred-site.xml

```
yarn.scheduler.minimum-allocation-mb=3072
yarn.scheduler.maximum-allocation-mb=24576
yarn.nodemanager.resource.memory-mb=24576
mapreduce.map.memory.mb=3072
mapreduce.map.java.opts=-Xmx5500m
mapreduce.reduce.memory.mb=3072
mapreduce.reduce.java.opts=-Xmx5500m
yarn.app.mapreduce.am.resource.mb=6144
yarn.app.mapreduce.am.command-opts=-Xmx5500m
mapreduce.task.io.sort.mb=1024
mapred.child.java.opts=1024
```

yarn-site.xml

In yarn-site.xml update the following:

```
  <name>yarn.scheduler.minimum-allocation-mb</name>
<value>6144</value>

<name>yarn.scheduler.maximum-allocation-mb</name>
<value>24576</value>

<name>yarn.nodemanager.resource.memory-mb</name>
<value>24576</value>
```

In mapred-site.xml update/add the following

```
 <name>mapreduce.map.memory.mb</name>
   <value>3072</value>

   <name>mapreduce.reduce.memory.mb</name>
   <value>3072</value>

   <name>mapreduce.task.io.sort.mb</name>
   <value>1024</value>

   <name>mapreduce.map.java.opts</name>
   <value>-Xmx5500m</value>

   <name>mapreduce.reduce.java.opts</name>
```

```
        <value>-Xmx5500m</value>

<name>yarn.app.mapreduce.am.resource.mb</name>
    <value>6144</value>

    <name>yarn.app.mapreduce.am.command-opts</name>
    <value>-Xmx5500m</value>

    <name>mapred.child.java.opts</name>
    <value>-Xmx1024m</value>
```

Copy the configuration files

Delete the previous hadoop configuration directory if existing and recreate the directory.

```
clush –a –b echo $HADOOP_CONF_DIR
clush –a –b sudo rm -rf $HADOOP_CONF_DIR
clush –a –b sudo mkdir -p $HADOOP_CONF_DIR

clush -a -b sudo mkdir /home/ubuntu/core-hadoop
clush -a -b sudo chmcod 777 /home/ubuntu/core-hadoop
clush -a -B -c * --dest=/home/ubuntu/core-hadoop
clush -a -B "sudo mv /home/ubuntu/core-hadoop/* $HADOOP_CONF_DIR"
clush -a -B ls -l $HADOOP_CONF_DIR
```

Set appropriate permissions:

```
clush -a -B sudo chown -R $HDFS_USER:$HADOOP_GROUP $HADOOP_CONF_DIR/../
clush -a -B sudo chmod -R 755 $HADOOP_CONF_DIR/../
clush -a -B ls -ld $HADOOP_CONF_DIR/../
clush -a -B ls -l $HADOOP_CONF_DIR/../
```

Validating the Core Hadoop Installation

Follow these steps to install HDP:

Use the following instructions to start core Hadoop and perform the smoke tests:

- Format and Start HDFS
- Smoke Test HDFS
- Start YARN
- Start MapReduce JobHistory Server
- Smoke Test MapReduce

Format and Start HDFS

1. Execute these commands on the Namenode host machine:

```
$sudo -u $HDFS_USER /usr/lib/hadoop/bin/hadoop namenode –format
$sudo -u $HDFS_USER /usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR
start namenode
```

```
ubuntu@ubuntu20-namenode:~$ sudo -u $HDFS_USER /usr/lib/hadoop/bin/hadoop namenode -format
DEPRECATED: Use of this script to execute hdfs command is deprecated.
Instead use the hdfs command for it.

14/04/24 18:25:24 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = ubuntu20-namenode.novalocal/192.168.10.15
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 2.2.0.2.0.11.0-1
ubuntu@ubuntu20-namenode:~$ sudo /usr/lib/hadoop/sbin/hadoop-daemon.sh --config
$HADOOP_CONF_DIR start namenode
starting namenode, logging to /var/log/hadoop/root/hadoop-root-namenode-ubuntu20
-namenode.out
```

Make sure the namenode is up and running by typing the command:

```
$sudo jps
```

2. Execute these commands on the secondarynamenode:

```
$sudo -u $HDFS_USER /usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR
start secondarynamenode
```

```
ubuntu@ubuntu19-resourcemgr:~$ sudo -u $HDFS_USER /usr/lib/hadoop/sbin/hadoop-da
emon.sh --config $HADOOP_CONF_DIR start secondarynamenode
starting secondarynamenode, logging to /var/log/hadoop/hdfs/hadoop-hdfs-secondar
ynamenode-ubuntu19-resourcemgr.out
ubuntu@ubuntu19-resourcemgr:~$ sudo jps
960 SecondaryNameNode
1031 Jps
```

Execute these commands on all datanodes:

```
$clush –b –w 192.168.10.[24-167] “sudo -u $HDFS_USER
/usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start datanode”
```

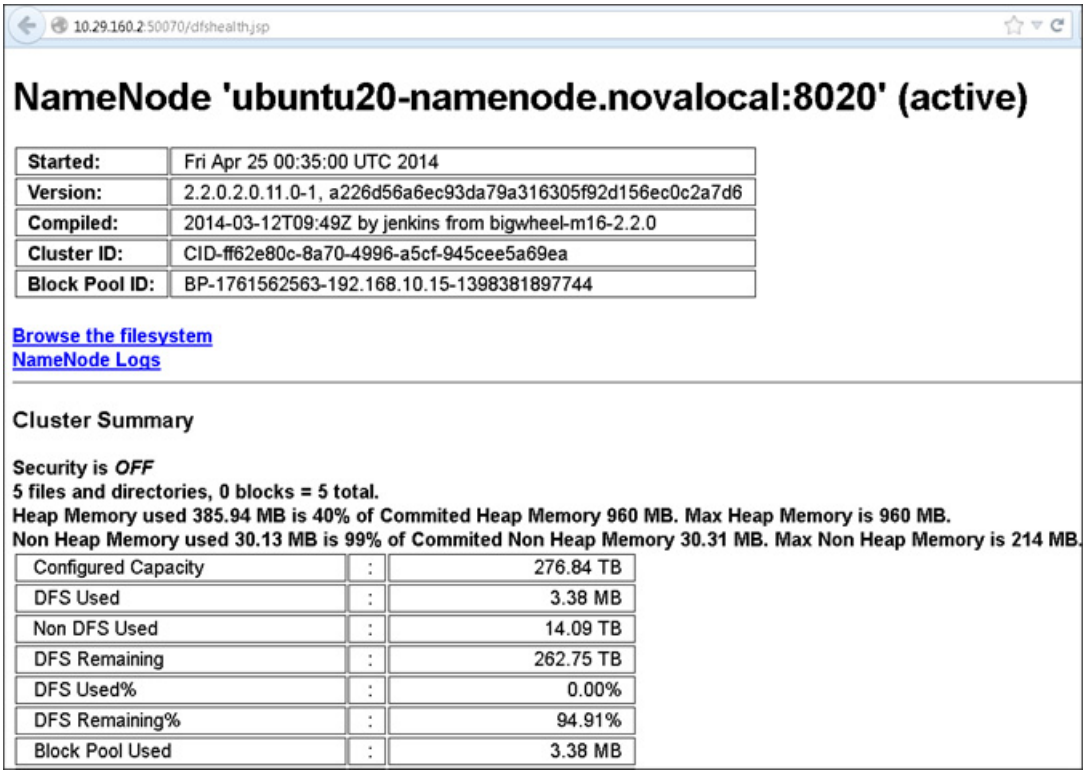
```
ubuntu@ubuntu20-namenode:~$ clush -b -w 192.168.10.[24-167] "sudo -u $HDFS_USER
/usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start datanode"
```

Smoke Test HDFS

1. See if you can reach the NameNode server with your browser:

   http://<namenode-vm-ip>:50070

Figure 93 NameNode Server



2. Create hdfs user directory in HDFS:

```
sudo -u $HDFS_USER hadoop fs -mkdir -p /user/hdfs
```

3. Try copying a file into HDFS and listing that file:

```
sudo -u $HDFS_USER hadoop fs -copyFromLocal /etc/passwd passwd hadoop fs -ls
```

```
ubuntu@ubuntu20-namenode:~$ sudo -u $HDFS_USER hadoop fs -mkdir -p /user/hdfs
ubuntu@ubuntu20-namenode:~$ sudo -u $HDFS_USER hadoop fs -copyFromLocal /etc/passwd
 passwd
ubuntu@ubuntu20-namenode:~$ hadoop fs -ls /user/hdfs
Found 2 items
drwx------   - hdfs hdfs          0 2014-04-25 00:55 /user/hdfs/.Trash
-rw-r--r--   3 hdfs hdfs       1436 2014-04-25 00:55 /user/hdfs/passwd
```

4. Test browsing HDFS:

```
http://<datanode-vm-ip>:50075/browseDirectory.jsp?namenodeInfoPort=50070&dir=/&nnaddr<namenode-vm-ip>:8020
```

Start YARN

1. Execute the following commands from the ResourceManager server.

   Update **/usr/lib/hadoop-yarn/sbin/yarn-daemon.sh** in ResourceManager by setting the following:

```
HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
```

```
HADOOP_LIBEXEC_DIR=${HADOOP_LIBEXEC_DIR:-$DEFAULT_LIBEXEC_DIR}
HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
. $HADOOP_LIBEXEC_DIR/yarn-config.sh
```

Copy this file to all nodes

```
clush -a -b -c /usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --dest=/home/ubuntu
clush -a -b sudo cp /home/ubuntu/yarn-daemon.sh /usr/lib/hadoop-yarn/sbin/

sudo -u $YARN_USER /usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config
$HADOOP_CONF_DIR start resourcemanager
```

2. Execute these commands from all the nodemanager nodes:

```
clush –B –w 192.168.10.[24-167] sudo –u $YARN_USER
/usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR start
nodemanager
```

Start MapReduce JobHistory Server

1. Change permissions on the container-executor file.

```
sudo chown -R root:hadoop /usr/lib/hadoop-yarn/bin/container-executor
sudo chmod -R 6050 /usr/lib/hadoop-yarn/bin/container-executor
```

2. Execute these commands from the JobHistory server to set up directories on HDFS.

Ubuntu requires a passwd for all the users. Create a password for **$HDFS_USER/hdfs** and login as hdfs.

```
sudo passwd hdfs
<enter Password>
```

Run the following on Resource-manager VM as $HDFS_USER

```
su $HDFS_USER

hadoop fs -mkdir -p /mr-history/tmp
hadoop fs -chmod -R 1777 /mr-history/tmp
hadoop fs -mkdir -p /mr-history/done
hadoop fs -chmod -R 1777 /mr-history/done
hadoop fs -chown -R mapred:hdfs /mr-history

hadoop fs -mkdir -p /app-logs
hadoop fs -chmod -R 1777 /app-logs
hadoop fs -chown yarn /app-logs

hadoop fs –ls /
```

```
ubuntu@ubuntu19-resourcemgr:~$ sudo jps
12288 Jps
6514 SecondaryNameNode
12209 JobHistoryServer
ubuntu@ubuntu19-resourcemgr:~$ su $HDFS_USER
Password:
$ hadoop fs -mkdir -p /app-logs
$ hadoop fs -chmod -R 1777 /app-logs
$ hadoop fs -chown yarn /app-logs
$ hadoop fs -chown -R mapred:hdfs /mr-history
$ hadoop fs -ls /
Found 2 items
drwxrwxrwt   - yarn   hdfs          0 2014-04-24 23:14 /app-logs
drwxr-xr-x   - mapred hdfs          0 2014-04-24 22:57 /mr-history
```

3. Execute these commands from the JobHistory server.

   Update the file **/usr/lib/hadoop-mapreduce/sbin/mr-jobhistory-daemon.sh** only on ResourceManager VM by adding the line HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec/ as follows:

```
DEFAULT_LIBEXEC_DIR="$bin"/../libexec
HADOOP_LIBEXEC_DIR=${HADOOP_LIBEXEC_DIR:-$DEFAULT_LIBEXEC_DIR}
HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
if [ -e ${HADOOP_LIBEXEC_DIR}/mapred-config.sh ]; then
  . $HADOOP_LIBEXEC_DIR/mapred-config.sh
fi

sudo -u $MAPRED_USER /usr/lib/hadoop-mapreduce/sbin/mr-jobhistory-daemon.sh
--config $HADOOP_CONF_DIR start historyserver
```

Smoke Test MapReduce

1. Open the ResourceManager: http://$resourcemanager.full.hostname:8088/
2. Smoke test using Terasort and sort 10GB of data.

```
su $HDFS_USER
sudo -u $HDFS_USER /usr/lib/hadoop/bin/hadoop jar
/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2*.jar teragen 100000000
/test/10gsort/input
sudo -u $HDFS_USER /usr/lib/hadoop/bin/hadoop jar
/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples-2*.jar terasort
/test/10gsort/input /test/10gsort/output
```

Installing HBase and Zookeeper

This section describes installing and testing Apache HBase, a distributed, column-oriented database that provides the ability to access and manipulate data randomly in the context of the large blocks that make up HDFS. It also describes installing and testing Apache ZooKeeper, a centralized tool for providing services to highly distributed systems.

```
clush -a -b sudo apt-get -y install zookeeper hbase
```

Create the Hbase Log and PID Directories

To create the Hbase log and PID directories, run these commands:

```
clush -a -b sudo mkdir -p $HBASE_LOG_DIR;
clush -a -b sudo chown -R $HBASE_USER:$HADOOP_GROUP $HBASE_LOG_DIR;
clush -a -b sudo chmod -R 755 $HBASE_LOG_DIR;
clush -a -b sudo mkdir -p $HBASE_PID_DIR;
clush -a -b sudo chown -R $HBASE_USER:$HADOOP_GROUP  $HBASE_PID_DIR;
clush -a -b sudo chmod -R 755  $HBASE_PID_DIR;
```

Create the Zookeeper Data and Pid Directories

To create the Zookeeper data and PID directories, run these commands:

```
clush -B -w 192.168.10.[15,23,24] sudo mkdir -p $ZOOKEEPER_LOG_DIR;
clush -B -w 192.168.10.[15,23,24] sudo  chown -R $ZOOKEEPER_USER:$HADOOP_GROUP
$ZOOKEEPER_LOG_DIR;
clush -B -w 192.168.10.[15,23,24] sudo  chmod -R 755 $ZOOKEEPER_LOG_DIR;

clush -B -w 192.168.10.[15,23,24] sudo  mkdir -p $ZOOKEEPER_PID_DIR;
clush -B -w 192.168.10.[15,23,24] sudo  chown -R $ZOOKEEPER_USER:$HADOOP_GROUP
$ZOOKEEPER_PID_DIR;
clush -B -w 192.168.10.[15,23,24] sudo  chmod -R 755 $ZOOKEEPER_PID_DIR;

clush -B -w 192.168.10.[15,23,24] sudo  mkdir -p $ZOOKEEPER_DATA_DIR;
clush -B -w 192.168.10.[15,23,24] sudo  chmod -R 755 $ZOOKEEPER_DATA_DIR;
clush -B -w 192.168.10.[15,23,24] sudo  chown -R $ZOOKEEPER_USER:$HADOOP_GROUP
$ZOOKEEPER_DATA_DIR
```

Initialize the zookeeper data directories with the 'myid' file. Create one file per Zookeeper server (namenode, resourcemanager and ubuntu1-vm1 servers), and put the number of that server in each file.

```
sudo -u $ZOOKEEPER_USER vi $ZOOKEEPER_DATA_DIR/myid
```

In the myid file on the namenode server, enter the corresponding number:

```
1
```

In the myid file on the resourcemanager server, enter the corresponding number:

```
2
```

In the myid file on the third server, enter the corresponding number:

```
3
```

Set Up the Configuration Files for Zookeeper and Hbase

There are several configuration files that need to be set up for HBase and ZooKeeper.

- Extract the HBase and ZooKeeper configuration files to separate temporary directories. The files are located in the **configuration_files/hbase and configuration_files/zookeeper** directories of the companion files.
- Modify the configuration files. In the respective temporary directories, locate the following files and modify the properties based on your environment. Search for TODO in the files for the properties to replace.

1. Edit zoo.cfg and modify the following properties:

   ```
   zoo.cfg
   ```

   Modify zoo.cfg in the namenode in the companion files and change TODO section as follows adding three servers

```
# the directory where the snapshot is stored.
dataDir=/mnt/hadoop/zookeeper/data
# the port at which the clients will connect
clientPort=2181
server.1=ubuntu20-namenode:2888:3888
server.2=ubuntu19-resourcemgr:2888:3888
server.3=ubuntu1-vm1:2888:3888

clush -b -w 192.168.10.[15,23,24] -c zoo.cfg --dest=/home/ubuntu/
clush -b -w 192.168.10.[15,23,24] sudo cp /home/ubuntu/zoo.cfg /etc/zookeeper/
```

2. Edit the **hbase-site.xml** and modify the following properties:

```
<name>hbase.rootdir</name>
<value>hdfs://ubuntu20-namenode:8020/apps/hbase/data</value>

<name>hbase.zookeeper.quorum</name>
<value>ubuntu20-namenode,ubuntu19-resourcemgr,ubuntu1-vm1</value>

<property>
 <name>hbase.master.info.bindAddress</name>
 <value>ubuntu19-resourcemgr</value>
<description>Enter the HBase Master server hostname </description>
```

3. Update **hbase-env.sh** and modify the following properties:

   This increases the HeapSize to 4G for both Hbase Master and Region Server.

```
export HBASE_MASTER_OPTS="-Xmx4096m"
export HBASE_REGIONSERVER_OPTS="-Xmx4096m"
```

1. Edit the **regionservers** file and list all the RegionServers hostnames (separated by newline character) in your environment.

```
ubuntu1-vm1
ubuntu1-vm2
ubuntu1-vm3
…
ubuntu20-vm8
```

2. Copy the configuration files

   On all hosts create the config directory:

```
clush –a –b sudo rm -r $HBASE_CONF_DIR ;
clush -a -b sudo mkdir -p $HBASE_CONF_DIR ;
clush –a –b sudo rm -r $ZOOKEEPER_CONF_DIR ;
clush –a –b sudo mkdir -p $ZOOKEEPER_CONF_DIR ;
```

3. Copy all the HBase configuration files to **$HBASE_CONF_DIR** and the ZooKeeper configuration files to **$ZOOKEEPER_CONF_DIR** directory of all relevant nodes.

   For Zookeeper it is Namenode, Resourcemanager and one more VM and for Hbase, to HBase Master (at resource manager vm and all regionservers). cd to <companion files>/configuration_files/zookeeper on Namenode

```
clush -b -w 192.168.10.[15,23,24] sudo mkdir /home/ubuntu/zookeeper
clush -b -w 192.168.10.[15,23,24] sudo chmod 777 /home/ubuntu/zookeeper
clush -b -w 192.168.10.[15,23,24] -c * --dest=/home/ubuntu/zookeeper
clush -b -w 192.168.10.[15,23,24] sudo cp /home/ubuntu/zookeeper/*
$ZOOKEEPER_CONF_DIR
```

   Similarly for HBase

   cd to <companion files>/configuration_files/hbase on Namenode

```
mv /etc/hbase/conf/log4j.properties /etc/hbase/conf/log4j.properties.orig
```

```
clush -b -w 192.168.10.[23-167] sudo mkdir /home/ubuntu/hbase
clush -b -w 192.168.10.[23-167] sudo chmod 777 /home/ubuntu/hbase
clush -b -w 192.168.10.[23-167] -c * --dest=/home/ubuntu/hbase
clush -b -w 192.168.10.[23-167] sudo cp /home/ubuntu/hbase/* $HBASE_CONF_DIR
```

4. Set appropriate permissions:

```
clush -b -w 192.168.10.[23-167] sudo chmod a+x $HBASE_CONF_DIR/;
clush -b -w 192.168.10.[23-167] sudo chown -R $HBASE_USER:$HADOOP_GROUP
$HBASE_CONF_DIR/../  ;
clush -b -w 192.168.10.[23-167] sudo chmod -R 755 $HBASE_CONF_DIR/../

clush -b -w 192.168.10.[15,23,24] sudo chmod a+x $ZOOKEEPER_CONF_DIR/;
clush -b -w 192.168.10.[15,23,24] sudo chown -R $ZOOKEEPER_USER:$HADOOP_GROUP
$ZOOKEEPER_CONF_DIR/../  ;
clush -b -w 192.168.10.[15,23,24] sudo chmod -R 755 $ZOOKEEPER_CONF_DIR/../
```

Start Zookeeper

Execute this command from the Namenode node:

```
clush -b -w 192.168.10.[15,23,24]  sudo -u $ZOOKEEPER_USER
/usr/lib/zookeeper/bin/zkServer.sh start $ZOOKEEPER_CONF_DIR/zoo.cfg

clush -b -w 192.168.10.[15,23,24]  sudo jps
```

This above jps command should show process “QuorumPeerMain”

Start HBase

Execute this command from the HBase master node (same asResourceManager VM):

```
<login as $HDFS_USER as “su $HDFS_USER”>
/usr/lib/hadoop/bin/hadoop fs -mkdir /apps
/usr/lib/hadoop/bin/hadoop fs -mkdir /apps/hbase /usr/lib/hadoop/bin/hadoop fs
-chown -R hbase /apps/hbase
exit


sudo -u $HBASE_USER /usr/lib/hbase/bin/hbase-daemon.sh --config $HBASE_CONF_DIR
start master
```

Execute this command from each HBase regionserver node:

```
clush -b -w 192.168.10.[24-167] sudo -u $HBASE_USER
/usr/lib/hbase/bin/hbase-daemon.sh --config $HBASE_CONF_DIR start regionserver
```

Installing Mahout

Install Mahout on all the VMs other than the namenode and the resourcemanager as follows:

```
clush -B -w 192.168.10.[24-167] sudo apt-get -y install mahout
```

Installing Apache Pig

This section describes installing and testing Apache Pig on all VMs other than namenode and resourcemanager. Apache Pig is a platform for creating higher level data flow programs that can be compiled into sequences of MapReduce programs, using Pig, the platform’s native language.

```
clush -b -w 192.168.10.[24-167] sudo apt-get -y install pig
```

```
clush -a -b echo $PIG_CONF_DIR
clush -a -b sudo rm -r $PIG_CONF_DIR
clush -a -b sudo mkdir -p $PIG_CONF_DIR
clush -a -b sudo mkdir -p /home/ubuntu/pig
clush -b -w 192.168.10.[24-167] sudo chmod 777 /home/ubuntu/pig/
clush -b -w 192.168.10.[24-167] -c * --dest=/home/ubuntu/pig
clush -b -w 192.168.10.[24-167] sudo cp /home/ubuntu/pig/* /etc/pig/conf/
```

Installing Apache Hive/HCatalog

Install Apache Hive on all VM. Hive will be configured and run mostly from Resource Manager.

Create Hive Log Directories

```
clush –a –b sudo mkdir -p $HIVE_LOG_DIR;
clush –a –b sudo chown -R $HIVE_USER:$HADOOP_GROUP $HIVE_LOG_DIR;
clush –a –b sudo chmod -R 755 $HIVE_LOG_DIR;
```

Install Hive and Hcatalog

```
clush –a –b  sudo apt-get -y install hive hcatalog
clush –a –b sudo apt-get -y install mysql-connector-java*

clush –a –b sudo cp /usr/share/java mysql-connector-java-5.0.8.jar
mysql-connector-java.jar /usr/lib/hive/lib/

sudo cp /usr/share/java/mysql-connector-java* /usr/lib/hive/lib/
 clush -a -b sudo cp /usr/share/java/mysql-connector-java* /usr/lib/hive/lib/
```

Hive/HCatalog Configuration

We have created a sample user “cisco” for this CVD as mentioned in above pre-requisite section under MySQL, update the hive configuration file **`/etc/hive/hive-site.xml`**.

```
<name>javax.jdo.option.ConnectionUserName</name>
<value>cisco</value>

<name>javax.jdo.option.ConnectionPassword</name>
<value>cisco</value>

For the following
<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://$mysql.full.hostname:3306/$database.name?createDatabaseIfNotExist=
true</value>
 <description>Enter your JDBC connection string. </description>

Update as follows
<name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:mysql://ubuntu19-resourcemgr:3306/hive?createDatabaseIfNotExist=true</valu
e>
 <description>Enter your JDBC connection string. </description>
```

This will create database hive

```
<name>hive.metastore.uris</name>
 <value>thrift://$metastore.server.full.hostname:9083</value>
as
<name>hive.metastore.uris</name>
```

```
<value>thrift://ubuntu19-resourcemgr:9083</value>
```

Create and Copy Hive Configuration files to all data/task VMs

```
clush -b -w 192.168.10.[24-167] sudo rm -r $HIVE_CONF_DIR ;
clush -b -w 192.168.10.[24-167] sudo mkdir -p $HIVE_CONF_DIR ;

clush -b -w 192.168.10.[24-167] sudo mkdir /home/ubuntu/hive
clush -b -w 192.168.10.[24-167] sudo chmod 777 /home/ubuntu/hive
```

Copy the above update hive configuration and all other Hive files in HDP companion files to all the data/task VMs.

```
clush -b -w 192.168.10.[24-167] -c * --dest=/home/ubuntu/hive
clush -b -w 192.168.10.[24-167] sudo cp /home/ubuntu/hive/* $HIVE_CONF_DIR

clush -b -w 192.168.10.[24-167] sudo chown -R $HIVE_USER:$HADOOP_GROUP
$HIVE_CONF_DIR/../ ;
clush -b -w 192.168.10.[24-167] sudo chmod -R 755 $HIVE_CONF_DIR/../ ;
```

Create Hive directories on Hadoop

```
sudo -u $HDFS_USER hadoop fs -mkdir -p /user/$HIVE_USER
sudo -u $HDFS_USER hadoop fs -chown $HIVE_USER:$HDFS_USER /user/$HIVE_USER

sudo -u $HDFS_USER hadoop fs -mkdir -p /apps/hive/warehouse

sudo -u $HDFS_USER hadoop fs -chown -R $HIVE_USER:$HDFS_USER /apps/hive
sudo -u $HDFS_USER hadoop fs -chmod -R 775 /apps/hive
sudo -u $HDFS_USER hadoop fs -mkdir -p /tmp/scratch
sudo -u $HDFS_USER hadoop fs -chown -R $HIVE_USER:$HDFS_USER /tmp/scratch

sudo -u $HDFS_USER hadoop fs -chmod -R 777 /tmp/scratch
```

This completes the Manual installation of HDP 2.0. For more information, see: http://docs.hortonworks.com/HDPDocuments/HDP2/HDP-2.0.9.1/bk_installing_manually_book/content/rpm-chap1.html

Conclusion

Hadoop has evolved into a leading data management platform across all verticals. The Cisco CPA v2 for Big Data for HDP 2.0 with OpenStack offers a dependable multi-tenant Hadoop deployment model for Hadoop as a Service.

The configuration detailed in the document can be extended to clusters of various sizes depending on what application demands. Up to 160 servers (10 racks) can be supported with no additional switching in a single UCS domain. Each additional rack requires two Cisco Nexus 2232PP 10GigE Fabric Extenders and 16 Cisco UCS C240 M3 Rack-Mount Servers. Scaling beyond 10 racks (160 servers) can be implemented by interconnecting multiple UCS domains using Nexus 6000/7000 Series switches, scalable to thousands of servers and to hundreds of petabytes storage, and managed from a single pane using UCS Central.

Bill of Material

This section gives the BOM for the 64 node Capacity Optimized with Flash memory Cluster. See Table 13 for BOM for the master rack, Table 14 for expansion racks (rack 2 to 4), Table 17 and 18 for software components.

Table 10 Bill of Materials for Base Rack

Part Number	Description	Quantity
UCS-SL-CPA2-CF	Capacity Optimized with Flash memory Cluster	1
UCSC-C240-M3L	UCS C240 M3 LFF w/oCPU mem HD PCIe PSU w/ rail kit expdr	16
UCSC-NYTRO-200GB=	Cisco Nytro MegaRAID 200GB Controller	16
UCSC-PCIE-CSC-02	Cisco VIC 1225 Dual Port 10Gb SFP+ CNA	16
CAB-N5K6A-NA	Power Cord 200/240V 6A North America	32
UCSC-PSU2-1200	1200W 2u Power Supply For UCS	32
UCSC-RAIL-2U	2U Rail Kit for UCS C-Series servers	16
UCSC-HS-C240M3	Heat Sink for UCS C240 M3 Rack Server	32
UCSC-PCIF-01F	Full height PCIe filler for C-Series	48
UCS-CPU-E52660B	2.20 GHz E5-2660 v2/95W 10C/25MB Cache/DDR3 1866MHz	32
UCS-MR-1X162RZ-A	16GB DDR3-1866-MHz RDIMM/PC3-14900/dual rank/x4/1.5v	256
UCS-HD4T7KS3-E	4TB SAS 7.2K RPM 3.5 inch HDD/hot plug/drive sled mounted	192
UCS-SL-BD-FI96	Cisco UCS 6296 FI w/ 18p LIC, Cables Bundle	2
N2K-UCS2232PF	Cisco Nexus 2232PP with 16 FET (2 AC PS, 1 FAN (Std Airflow)	2
SFP-H10GB-CU3M=	10GBASE-CU SFP+ Cable 3 Meter	28
RACK-UCS2	Cisco R42610 standard rack w/side panels	1
RP208-30-1P-U-2=	Cisco RP208-30-U-2 Single Phase PDU 20x C13 4x C19 (Country Specific)	2
CON-UCW3-RPDUX	UC PLUS 24X7X4 Cisco RP208-30-U-X Single Phase PDU 2x (Country Specific)	6

Table 11 Bill of Materials for Expansion Rack

Part Number	Description	Quantity
UCSC-C240-M3L	UCS C240 M3 LFF w/oCPU mem HD PCIe PSU w/ rail kit expdr	48
UCSC-NYTRO-200GB=	Cisco Nytro MegaRAID 200GB Controller	48
UCSC-PCIE-CSC-02	Cisco VIC 1225 Dual Port 10Gb SFP+ CNA	48

Table 11 Bill of Materials for Expansion Rack

Part Number	Description	Quantity
CAB-N5K6A-NA	Power Cord 200/240V 6A North America	96
UCSC-PSU2-1200	1200W 2u Power Supply For UCS	96
UCSC-RAIL-2U	2U Rail Kit for UCS C-Series servers	48
UCSC-HS-C240M3	Heat Sink for UCS C240 M3 Rack Server	96
UCSC-PCIF-01F	Full height PCIe filler for C-Series	144
UCS-CPU-E52660B	2.20 GHz E5-2660 v2/95W 10C/25MB Cache/DDR3 1866MHz	96
UCS-MR-1X162RZ-A	16GB DDR3-1866-MHz RDIMM/PC3-14900/dual rank/x4/1.5v	768
UCS-HD4T7KS3-E	4TB SAS 7.2K RPM 3.5 inch HDD/hot plug/drive sled mounted	576
N2K-UCS2232PF	4TB SAS 7.2K RPM 3.5 inch HDD/hot plug/drive sled mounted	6
CON-SNTP-UCS2232	SMARTNET 24X7X4 Cisco Nexus 2232PP	6
SFP-H10GB-CU3M=	10GBASE-CU SFP+ Cable 3 Meter	84
RACK-UCS2	Cisco R42610 standard rack w/side panels	3
RP208-30-1P-U-2=	Cisco RP208-30-U-2 Single Phase PDU 20x C13 4x C19 (Country Specific)	6
CON-UCW3-RPDUX	UC PLUS 24X7X4 Cisco RP208-30-U-X Single Phase PDU 2x (Country Specific)	18

Table 12 Canonical Ubuntu Linux License

Canonical Ubuntu 12.04.4 LTS	Description	Quantity
NA (Procured directly from Canonical)	12 month Ubuntu advantage support for Ubuntu Cloud Availability Zone (small).	1

Table 13 Hortonworks Software License

Hortonworks	Description	Quantity
NA (Procured directly from Hortonworks)	Hortonworks Data Platform 2.0	64