

---

# **Documentation Imported from HPE HOS**

---

# Table of Contents

Planning .....	xii
I. Planning .....	1
1. Registering SUSE Linux .....	3
Registering SUSE Linux during the Installation .....	3
Registering SUSE Linux from the Installed System .....	3
Registering from the Installed System .....	3
Registering SUSE Linux during Automated Deployment .....	4
2. Hardware and Software Support Matrix .....	5
OpenStack Version Information .....	5
Supported Hardware .....	5
Supported Hardware Configurations .....	5
Cloud Scaling .....	5
Supported Software .....	6
Notes about Performance .....	6
Disk Calculator .....	6
Disk Calculator for Compute-Centric Deployments .....	6
Enter Input Parameters .....	7
Audit Logging Adjustment .....	9
Select the Deployment Model .....	9
Match to a Disk Model .....	10
Entry Scale Disk Models .....	10
MML Disk Models .....	11
Notes about disk sizing for Cinder bootable volumes .....	12
KVM Guest OS Support .....	12
ESX Guest OS Support .....	13
Ironic Guest OS Support .....	13
3. Recommended Hardware Minimums for the Example Configurations .....	14
Recommended Hardware Minimums for an Entry-scale KVM with VSA Model ....	14
Recommended Hardware Minimums for an Entry-scale KVM with Ceph Model ...	15
Recommended Hardware Minimums for an Entry-scale ESX, KVM with VSA Model .....	16
Recommended Hardware Minimums for an Entry-scale ESX, KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging .....	18
Recommended Hardware Minimums for an Ironic Flat Network Model .....	20
Recommended Hardware Minimums for an Entry-scale Swift Model .....	20
4. High Availability .....	24
High Availability Concepts Overview .....	24
Highly Available Cloud Infrastructure .....	24
High Availability of Controllers .....	24
High Availability Routing - Centralized .....	25
High Availability Routing - Distributed .....	26
Availability Zones .....	27
Compute with KVM .....	27
Nova Availability Zones .....	28
Compute with ESX Hypervisor .....	28
Block Storage with StoreVirtual VSA .....	28
Deploy VSA cluster across Availability Zones/Racks .....	29
Cinder Availability Zones .....	30
Object Storage with Swift .....	30
Highly Available Cloud Applications and Workloads .....	31
What is not Highly Available? .....	31

More Information .....	32
5. Third Party Integrations .....	33
II. Helion Lifecycle Manager Overview .....	34
6. Input Model .....	38
Introduction to the Input Model .....	38
New in HPE Helion OpenStack 5.0 .....	38
Concepts .....	38
Cloud .....	40
Control Planes .....	40
Services .....	42
Server Roles .....	42
Disk Model .....	42
Memory Model .....	43
CPU Model .....	44
Servers .....	44
Server Groups .....	45
Networking .....	46
Configuration Data .....	49
7. Configuration Objects .....	50
Cloud Configuration .....	50
Control Plane .....	51
Clusters .....	53
Resources .....	54
Multiple Control Planes .....	55
Load Balancer Definitions in Control Planes .....	56
Load Balancers .....	56
Regions .....	57
Servers .....	59
Server Groups .....	61
Server Roles .....	62
Disk Models .....	63
Volume Groups .....	64
Device Groups .....	66
Disk Sizing for Virtual Machine Servers .....	66
Memory Models .....	67
Huge Pages .....	68
Memory Sizing for Virtual Machine Servers .....	68
CPU Models .....	68
CPU Assignments .....	69
CPU Usage .....	69
Components and Roles in the CPU Model .....	70
CPU sizing for virtual machine servers .....	70
Interface Models .....	70
network-interfaces .....	72
Bonding .....	73
fcoe-interfaces .....	75
dpdk-devices .....	76
NIC Mappings .....	77
NIC Mappings for Virtual Machine Servers .....	78
Network Groups .....	79
Load Balancer Definitions in Network Groups .....	82
Network Tags .....	82
MTU (Maximum Transmission Unit) .....	85
Networks .....	85

Firewall Rules .....	87
Rule .....	88
Configuration Data .....	88
Neutron network-tags .....	90
Neutron Configuration Data .....	90
Octavia Configuration Data .....	92
Ironic Configuration Data .....	92
Swift Configuration Data .....	93
Pass Through .....	94
8. Other Topics .....	95
Services and Service Components .....	95
Name Generation .....	97
Persisted Data .....	98
Persisted Server Allocations .....	99
Persisted Address Allocations .....	100
Server Allocation .....	101
Server Network Selection .....	101
Network Route Validation .....	101
Configuring Neutron Provider VLANs .....	105
Standalone Lifecycle Manager .....	106
9. Configuration Processor Information Files .....	108
address_info.yml .....	109
firewall_info.yml .....	110
net_info.yml .....	111
route_info.yml .....	111
server_info.yml .....	112
service_info.yml .....	113
control_plane_topology.yml .....	113
network_topology.yml .....	115
region_topology.yml .....	116
service_topology.yml .....	117
private_data_metadata.yml .....	117
password_change.yml .....	119
explain.txt .....	119
CloudDiagram.txt .....	120
HTML Representation .....	124
10. Example Configurations .....	125
HPE Helion OpenStack Example Configurations .....	125
Modifying the Entry-scale KVM with VSA Model for Your Environment .....	126
Alternative Configurations .....	126
KVM Examples .....	126
Entry-scale KVM with VSA Model .....	126
Entry-scale KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging .....	129
Entry-scale KVM with Ceph Model .....	130
Mid-scale KVM with VSA Model .....	147
ESX Examples .....	149
Entry-scale ESX, KVM with VSA Model .....	149
Entry-scale ESX, KVM with VSA Model with Dedicated Cluster for Me- tering, Monitoring, and Logging .....	151
Swift Examples .....	154
Entry-scale Swift Model .....	154
Ironic Examples .....	159
Entry-scale Cloud with Ironic Flat Network .....	159

Entry-scale Cloud with Ironic Multi-Tenancy .....	161
11. Modifying the Entry-scale KVM with VSA Model for Your Environment .....	163
Localizing the Input Model .....	163
Customizing the Input Model .....	170
disks_controller.yml .....	170
File Systems Storage .....	170
Swift Storage .....	171
disks_vsa.yml .....	171
disks_compute.yml .....	172
VSA with or without Adaptive Optimization (AO) .....	172
Creating Multiple VSA Clusters .....	173
Cloud Configuration Changes to Create More Than One Cluster .....	173
Cloud Configuration Changes to Create Two Cluster with Different Set of Disks .....	176
Configuring a Separate iSCSI Network to use with VSA .....	177
12. Modifying Example Configurations for Object Storage using Swift .....	180
Object Storage using Swift Overview .....	180
What is the Object Storage (Swift) Service? .....	180
Object Storage (Swift) Services .....	180
Allocating Proxy, Account, and Container (PAC) Servers for Object Storage .....	181
To allocate Swift PAC servers .....	181
Allocating Object Servers .....	181
To Allocate a Swift Object Server .....	182
Creating Roles for Swift Nodes .....	182
Allocating Disk Drives for Object Storage .....	183
Making Changes to a Swift Disk Model .....	183
Swift Requirements for Device Group Drives .....	186
Creating a Swift Proxy, Account, and Container (PAC) Cluster .....	186
Steps to Create a Swift Proxy, Account, and Container (PAC) Cluster .....	186
Service Components .....	187
Creating Object Server Resource Nodes .....	187
Understanding Swift Network and Service Requirements .....	188
Understanding Swift Ring Specifications .....	189
Ring Specifications in the Input Model .....	189
Replication Ring Parameters .....	190
Erasure Coded Rings .....	191
Selecting a Partition Power .....	192
Designing Storage Policies .....	194
Specifying Storage Policies .....	195
Designing Swift Zones .....	196
Using Server Groups to Specify Swift Zones .....	197
Specifying Swift Zones at Ring Level .....	198
Customizing Swift Service Configuration Files .....	199
Configuring Swift Container Rate Limit .....	199
Configuring Swift Account Server Logging Level .....	200
13. Alternative Configurations .....	202
SLES Compute Nodes .....	202
Entry-scale KVM with Ceph Model .....	204
Entry-scale KVM with Ceph Model with Two Networks .....	204
Using a Dedicated Lifecycle Manager Node .....	208
Specifying a dedicated lifecycle manager in your input model .....	209
Configuring HPE Helion OpenStack without DVR .....	212
Configuring HPE Helion OpenStack with Provider VLANs and Physical Routers Only .....	213

Considerations When Installing Two Systems on One Subnet .....	214
<b>Installation Guide .....</b>	<b>ccxvi</b>
<b>Installation Overview .....</b>	<b>ccxxvii</b>
For More Information .....	ccxxviii
<b>I. Pre-Installation .....</b>	<b>1</b>
1. Overview .....	3
2. Pre-Installation Checklist .....	4
BIOS and iLO Settings .....	4
Network Setup and Configuration .....	4
Lifecycle Manager .....	7
Information for the <code>nic_mappings.yml</code> Input File .....	7
Control Plane .....	8
Compute Hosts .....	9
VSA Hosts .....	9
Additional Comments .....	10
SLES Pre-Installation Checks .....	10
Introduction .....	10
Sample iptables rules must be removed on SLES .....	10
3. Using Git for Configuration Management .....	12
Initialization on a new deployment .....	12
Updating any configuration, including the default configuration .....	13
4. Boot from SAN and Multipath Configuration .....	14
Introduction .....	14
Install Phase Configuration .....	14
QLogic FCoE restrictions and additional configurations .....	15
Red Hat Compute Host for FCoE .....	17
Installing the HPE Helion OpenStack 5.0 iso for nodes that support Boot from SAN .....	19
5. Installing the L2 Gateway Agent for the Networking Service .....	25
Introduction .....	25
Sample network topology (for illustration purposes) .....	25
Networks .....	26
HPE 5930 switch configuration .....	27
Configuring the provider data path network .....	28
Enabling and configuring the L2 gateway agent .....	31
Routing between software and hardware .....	33
Connecting a baremetal server to the HPE 5930 switch .....	34
Configuration on a baremetal server .....	35
NIC bonding and IRF configuration .....	35
Scale numbers tested .....	35
L2 gateway commands .....	35
<b>II. Cloud Installation .....</b>	<b>37</b>
6. Overview .....	43
7. Installing via the GUI .....	45
Before you begin .....	46
Creating a CSV file for import .....	46
Run the GUI to install your cloud .....	47
Use the installer securely .....	47
Final Steps .....	48
8. DNS Service Installation Overview .....	49
Getting Started .....	49
Install the DNS Service with PowerDNS .....	49
Install DNS Service with PowerDNS .....	49
Configure the Backend .....	49

Install the DNS Service with BIND .....	51
Install DNS Service with BIND .....	51
Configure the Backend .....	51
Install the DNS Service with InfoBlox .....	51
Prerequisites .....	52
Configure the Backend .....	52
Configure DNS Domain and NS Records .....	55
9. Magnum Overview .....	57
Magnum Architecture .....	57
Install the Magnum Service .....	65
Installing Magnum as part of new HPE Helion OpenStack 5.0 environment .....	66
Adding Magnum to existing HPE Helion OpenStack 5.0 environment .....	66
Integrate Magnum with the DNS Service .....	67
10. Installing Mid-scale and Entry-scale KVM .....	70
Important Notes .....	70
Before You Start .....	71
Setting Up the Lifecycle Manager .....	71
Configuring Your Environment .....	73
Provisioning Your Baremetal Nodes .....	74
Running the Configuration Processor .....	76
Configuring TLS .....	77
Deploying the Cloud .....	77
Configuring a Block Storage Backend (Optional) .....	78
Post-Installation Verification and Administration .....	78
11. Installation for Helion Entry Scale ESX (with OVSVApp, KVM with VSA Model) .....	79
Important Notes .....	79
Before You Start .....	80
Prerequisites .....	80
Deploy ESX Cloud with OVSVApp .....	81
Procedure to Deploy ESX Cloud with OVSVApp .....	81
Setting Up the Lifecycle Manager .....	81
Prepare and Deploy Cloud Controllers .....	83
Provisioning Your Baremetal Nodes .....	84
Running the Configuration Processor .....	86
Deploying the Cloud .....	86
Prepare and Deploy ESX Computes and OVSVAPPs .....	87
Validate the block storage .....	92
Validate the compute .....	92
Validate the neutron .....	93
12. Sample activationtemplate.json File for ESX Compute .....	94
13. Installing Baremetal (Ironic) .....	107
Installation for HPE Helion Entry-scale Cloud with Ironic Flat Network .....	107
Before You Start .....	107
Setting Up the Lifecycle Manager .....	107
Configure Your Environment .....	109
Provisioning Your Baremetal Nodes .....	110
Running the Configuration Processor .....	112
Deploying the Cloud .....	112
Ironic configuration .....	113
Node Configuration .....	114
TLS Certificates with Ironic Python Agent (IPA) Images .....	114
Ironic in Multiple Control Plane .....	116

Networking for Baremetal in Multiple Control Plane .....	117
Handling Optional Swift Service .....	117
Instance Provisioning .....	117
Provisioning Baremetal Nodes with Flat Network Model .....	118
Supplied Images .....	119
Provisioning a node .....	119
Creating a node using <code>agent_ilo</code> .....	119
Creating a node using <code>agent_ipmi</code> .....	121
Create Flavor .....	122
Create network port .....	123
Create Glance Image .....	123
Generate Key Pair .....	124
Determine the neutron network ID .....	124
Boot the node .....	124
Create Glance Images using <code>diskimage-builder</code> (DIB) .....	125
Creating BIOS images for RHEL .....	126
Creating BIOS images for Ubuntu .....	126
Creating UEFI images .....	126
Provisioning Baremetal Nodes with Multi-Tenancy .....	127
View Ironic System Details .....	132
View details about the server using <code>nova show &lt;nova-node-id&gt;</code> .....	132
View detailed information about a node using <code>ironic node-show &lt;ironic-node-id&gt;</code> .....	133
View detailed information about a port using <code>ironic port-show &lt;ironic-port-id&gt;</code> .....	134
View detailed information about a hypervisor using <code>nova hypervisor-list</code> and <code>nova hypervisor-show</code> .....	134
View a list of all running services using <code>nova service-list</code> .....	135
Troubleshooting Ironic Installation .....	135
No valid host was found. There are not enough hosts available. ....	136
Deployment to a node fails and in "ironic node-list" command, the power_state column for the node is shown as "None" .....	138
Error Downloading Image .....	139
Using node-inspection can cause temporary claim of IP addresses ....	139
Node permanently stuck in deploying state .....	139
The NICs in the baremetal node should come first in boot order .....	139
Increase in the number of nodes can cause power commands to fail .....	140
DHCP succeeds with PXE but times out with iPXE .....	140
Node Cleaning .....	141
Setup .....	142
In use .....	142
Troubleshooting .....	143
Disabling Node Cleaning .....	143
Ironic and OneView .....	143
Enabling Ironic OneView driver in HPE Helion OpenStack .....	143
Adding OneView Appliance Credentials .....	144
Encrypting the OneView Password .....	144
Decrypting the OneView Password .....	144
Registering Baremetal Node for OneView Driver .....	144
Updating Node Properties .....	145
Creating Port for Driver .....	145
Creating a Node .....	145
Getting Data using REST API .....	145
Ironic OneView CLI .....	145

RAID Configuration for Ironic .....	146
Audit Support for Ironic .....	150
API Audit Logging .....	150
Enabling API Audit Logging .....	151
Sample Audit Event .....	151
14. Installation for HPE Helion Entry-scale Cloud with Swift Only .....	153
Important Notes .....	153
Before You Start .....	153
Setting Up the Lifecycle Manager .....	153
Configure Your Environment .....	156
Running the Configuration Processor .....	157
Provisioning Your Baremetal Nodes .....	158
Deploying the Cloud .....	159
Post-Installation Verification and Administration .....	160
15. Installing SLES Compute .....	161
SLES Compute Node Installation Overview .....	161
SLES Support .....	161
Using the Lifecycle Manager to Deploy SLES Compute Nodes .....	161
Deploying legacy BIOS SLES compute nodes .....	162
Provisioning SLES Yourself .....	162
Introduction .....	162
Configure Lifecycle Manager to Enable SLES .....	162
Install SLES 12 SP2 .....	163
Assign a static IP .....	163
Add <code>stack</code> user and home directory .....	164
Allow user <code>stack</code> to <code>sudo</code> without password .....	164
Add <code>zypper</code> repository .....	164
Add Required Packages .....	164
Set up passwordless SSH access .....	164
Using SLES as a Ceph Client .....	165
16. HLM-Hypervisor instructions .....	166
Introduction .....	166
Model changes .....	166
passthrough-network-groups .....	166
hlm-hypervisor .....	167
Bootstrap Instructions .....	173
Customizing for VCP .....	173
Installing a fully input model managed Virtual Control Plane based HPE	
Helion OpenStack Cloud .....	173
Provisioning just the Virtual Control Plane VMs .....	174
Manually running each phase of the Virtual Control Plane provisioning .....	174
NIC Mapping for HLM Virtual-Controllers .....	175
Recommended Mappings .....	176
Notification of actions to be taken post upgrade .....	176
Reboot of a HLM-Hypervisor node .....	176
Staged install on HLM-Hypervisor individual play books .....	177
Virtual Controller Replacement .....	177
Ansible host-specific variables for network-group access .....	178
Example 3rd-Party Ansible for Network Configuration Access .....	181
Ansible host-specific variables for gluster .....	182
"Empty" HLM Hypervisor Node .....	182
Monasca Monitoring of HLM Hypervisors .....	182
Monitored Metrics .....	182
Configured Alarms .....	183

HLM Hypervisor Monitoring Configuration .....	183
17. Integrations .....	185
Ceph Overview .....	185
Overview .....	185
Deployment Architecture .....	186
Ceph Networking .....	186
Placement of service component .....	187
Ceph Deployment Architecture .....	188
Alternative supported architecture .....	189
Core networking .....	189
Placement of service components .....	190
Hardware recommendations .....	190
Ceph Deployment and Configurations .....	190
Alternative Supported Choices .....	201
Usage of Ceph Storage .....	214
Managing Ceph Clusters After Deployment .....	227
Configuring for VSA Block Storage Backend .....	227
Prerequisites .....	228
Notes .....	228
Configure HPE StoreVirtual VSA .....	228
Using Ansible .....	229
Using the CMC Utility .....	230
Configure VSA as the Backend .....	241
Post-Installation Tasks .....	244
Configuring for 3PAR Block Storage Backend .....	244
Prerequisites .....	244
Notes .....	245
Multipath Support .....	245
Configure 3PAR FC as a Cinder Backend .....	246
Configure 3PAR iSCSI as Cinder backend .....	247
Post-Installation Tasks .....	249
Ironic OneView Integration .....	249
Prerequisites .....	249
Integrating with OneView .....	249
Registering Node in OneView .....	250
Provisioning Ironic Node .....	251
18. Troubleshooting the Installation .....	257
Issues during Lifecycle-manager Setup .....	257
Issues while Provisioning your Baremetal Nodes .....	257
Issues while Updating Configuration Files .....	262
Issues while Deploying the Cloud .....	263
19. Troubleshooting the Block Storage Backend Configuration .....	268
20. Troubleshooting the ESX .....	270
Issue: If hlm-ux-services.service is not running, the EON resource-activate and resource-deactivate commands fails .....	270
Issue: ESX onboard Compute .....	270
Issue: Migrate eon-conductor .....	270
Issue: ESX Cluster shows UNKNOWN in OpsConsole .....	273
Issue: Unable to view the VM console in Horizon UI .....	273
III. Post-Installation .....	275
21. Overview .....	278
22. Cloud Verification .....	279
API Verification .....	279
Prerequisites .....	279

Tempest Integration Tests .....	279
Running the Tests .....	280
Viewing Test Results .....	280
Customizing the Test Run .....	281
Run Tests for Specific Services and Exclude Specific Features .....	281
Run Tests Matching a Series of White and Blacklists .....	281
23. UI Verification .....	283
Verifying Your Block Storage Backend .....	283
Create a Volume .....	283
Attach Volume to an Instance .....	284
Detach Volume from Instance .....	284
Delete Volume .....	285
Verifying Your Object Storage (Swift) .....	285
Verify the Object Storage (Swift) Operations .....	286
Uploading an Image for Use .....	287
Running the Playbook .....	287
How to Curate Your Own Images .....	287
Using the GlanceClient CLI to Create Images .....	287
Creating an External Network .....	287
Notes .....	288
Using the Ansible Playbook .....	288
Using the NeutronClient CLI .....	288
Next Steps .....	289
24. Installing OpenStack Clients .....	290
25. Configuring Transport Layer Security (TLS) .....	293
Configuring TLS in the input model .....	294
User-provided certificates and trust chains .....	295
Edit the input model to include your certificate files .....	296
Generate a self-signed CA .....	297
Generate a certificate signing request .....	299
Generate a server certificate .....	299
Upload to the lifecycle manager .....	302
Configuring the cipher suite .....	302
Testing .....	303
Verifying that the trust chain is correctly deployed .....	303
Turning TLS on or off .....	303
26. Configuring Availability Zones .....	305
27. Configuring Load Balancer as a Service .....	306
Prerequisites .....	307
Octavia Load Balancing Provider .....	307
Setup of prerequisites .....	307
Create Load Balancers .....	317
Create Floating IPs for Load Balancer .....	320
Testing the Octavia Load Balancer .....	321
28. Other Common Post-Installation Tasks .....	323
Determining Your User Credentials .....	323
Configure your lifecycle manager to use the command-line tools .....	323
Protect home directory .....	323
Back up Your SSH Keys .....	324
Retrieving Service Endpoints .....	324
Other Common Post-Installation Tasks .....	324

# **Planning**

---

## **Planning**

---

# Table of Contents

I. Planning .....	1
1. Registering SUSE Linux .....	3
Registering SUSE Linux during the Installation .....	3
Registering SUSE Linux from the Installed System .....	3
Registering from the Installed System .....	3
Registering SUSE Linux during Automated Deployment .....	4
2. Hardware and Software Support Matrix .....	5
OpenStack Version Information .....	5
Supported Hardware .....	5
Supported Hardware Configurations .....	5
Cloud Scaling .....	5
Supported Software .....	6
Notes about Performance .....	6
Disk Calculator .....	6
Disk Calculator for Compute-Centric Deployments .....	6
Enter Input Parameters .....	7
Audit Logging Adjustment .....	9
Select the Deployment Model .....	9
Match to a Disk Model .....	10
Entry Scale Disk Models .....	10
MML Disk Models .....	11
Notes about disk sizing for Cinder bootable volumes .....	12
KVM Guest OS Support .....	12
ESX Guest OS Support .....	13
Ironic Guest OS Support .....	13
3. Recommended Hardware Minimums for the Example Configurations .....	14
Recommended Hardware Minimums for an Entry-scale KVM with VSA Model .....	14
Recommended Hardware Minimums for an Entry-scale KVM with Ceph Model .....	15
Recommended Hardware Minimums for an Entry-scale ESX, KVM with VSA Model ....	16
Recommended Hardware Minimums for an Entry-scale ESX, KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging .....	18
Recommended Hardware Minimums for an Ironic Flat Network Model .....	20
Recommended Hardware Minimums for an Entry-scale Swift Model .....	20
4. High Availability .....	24
High Availability Concepts Overview .....	24
Highly Available Cloud Infrastructure .....	24
High Availability of Controllers .....	24
High Availability Routing - Centralized .....	25
High Availability Routing - Distributed .....	26
Availability Zones .....	27
Compute with KVM .....	27
Nova Availability Zones .....	28
Compute with ESX Hypervisor .....	28
Block Storage with StoreVirtual VSA .....	28
Deploy VSA cluster across Availability Zones/Racks .....	29
Cinder Availability Zones .....	30
Object Storage with Swift .....	30
Highly Available Cloud Applications and Workloads .....	31
What is not Highly Available? .....	31
More Information .....	32
5. Third Party Integrations .....	33

II. Helion Lifecycle Manager Overview .....	34
6. Input Model .....	38
Introduction to the Input Model .....	38
New in HPE Helion OpenStack 5.0 .....	38
Concepts .....	38
Cloud .....	40
Control Planes .....	40
Services .....	42
Server Roles .....	42
Disk Model .....	42
Memory Model .....	43
CPU Model .....	44
Servers .....	44
Server Groups .....	45
Networking .....	46
Configuration Data .....	49
7. Configuration Objects .....	50
Cloud Configuration .....	50
Control Plane .....	51
Clusters .....	53
Resources .....	54
Multiple Control Planes .....	55
Load Balancer Definitions in Control Planes .....	56
Load Balancers .....	56
Regions .....	57
Servers .....	59
Server Groups .....	61
Server Roles .....	62
Disk Models .....	63
Volume Groups .....	64
Device Groups .....	66
Disk Sizing for Virtual Machine Servers .....	66
Memory Models .....	67
Huge Pages .....	68
Memory Sizing for Virtual Machine Servers .....	68
CPU Models .....	68
CPU Assignments .....	69
CPU Usage .....	69
Components and Roles in the CPU Model .....	70
CPU sizing for virtual machine servers .....	70
Interface Models .....	70
network-interfaces .....	72
Bonding .....	73
fcoe-interfaces .....	75
dpdk-devices .....	76
NIC Mappings .....	77
NIC Mappings for Virtual Machine Servers .....	78
Network Groups .....	79
Load Balancer Definitions in Network Groups .....	82
Network Tags .....	82
MTU (Maximum Transmission Unit) .....	85
Networks .....	85
Firewall Rules .....	87
Rule .....	88

Configuration Data .....	88
Neutron network-tags .....	90
Neutron Configuration Data .....	90
Octavia Configuration Data .....	92
Ironic Configuration Data .....	92
Swift Configuration Data .....	93
Pass Through .....	94
8. Other Topics .....	95
Services and Service Components .....	95
Name Generation .....	97
Persisted Data .....	98
Persisted Server Allocations .....	99
Persisted Address Allocations .....	100
Server Allocation .....	101
Server Network Selection .....	101
Network Route Validation .....	101
Configuring Neutron Provider VLANs .....	105
Standalone Lifecycle Manager .....	106
9. Configuration Processor Information Files .....	108
address_info.yml .....	109
firewall_info.yml .....	110
net_info.yml .....	111
route_info.yml .....	111
server_info.yml .....	112
service_info.yml .....	113
control_plane_topology.yml .....	113
network_topology.yml .....	115
region_topology.yml .....	116
service_topology.yml .....	117
private_data_metadata.yml .....	117
password_change.yml .....	119
explain.txt .....	119
CloudDiagram.txt .....	120
HTML Representation .....	124
10. Example Configurations .....	125
HPE Helion OpenStack Example Configurations .....	125
Modifying the Entry-scale KVM with VSA Model for Your Environment .....	126
Alternative Configurations .....	126
KVM Examples .....	126
Entry-scale KVM with VSA Model .....	126
Entry-scale KVM with VSA model with Dedicated Cluster for Metering, Moni- toring, and Logging .....	129
Entry-scale KVM with Ceph Model .....	130
Mid-scale KVM with VSA Model .....	147
ESX Examples .....	149
Entry-scale ESX, KVM with VSA Model .....	149
Entry-scale ESX, KVM with VSA Model with Dedicated Cluster for Metering, Monitoring, and Logging .....	151
Swift Examples .....	154
Entry-scale Swift Model .....	154
Ironic Examples .....	159
Entry-scale Cloud with Ironic Flat Network .....	159
Entry-scale Cloud with Ironic Multi-Tenancy .....	161
11. Modifying the Entry-scale KVM with VSA Model for Your Environment .....	163

Localizing the Input Model .....	163
Customizing the Input Model .....	170
disks_controller.yml .....	170
File Systems Storage .....	170
Swift Storage .....	171
disks_vsa.yml .....	171
disks_compute.yml .....	172
VSA with or without Adaptive Optimization (AO) .....	172
Creating Multiple VSA Clusters .....	173
Cloud Configuration Changes to Create More Than One Cluster .....	173
Cloud Configuration Changes to Create Two Cluster with Different Set of Disks .....	176
Configuring a Separate iSCSI Network to use with VSA .....	177
12. Modifying Example Configurations for Object Storage using Swift .....	180
Object Storage using Swift Overview .....	180
What is the Object Storage (Swift) Service? .....	180
Object Storage (Swift) Services .....	180
Allocating Proxy, Account, and Container (PAC) Servers for Object Storage .....	181
To allocate Swift PAC servers .....	181
Allocating Object Servers .....	181
To Allocate a Swift Object Server .....	182
Creating Roles for Swift Nodes .....	182
Allocating Disk Drives for Object Storage .....	183
Making Changes to a Swift Disk Model .....	183
Swift Requirements for Device Group Drives .....	186
Creating a Swift Proxy, Account, and Container (PAC) Cluster .....	186
Steps to Create a Swift Proxy, Account, and Container (PAC) Cluster .....	186
Service Components .....	187
Creating Object Server Resource Nodes .....	187
Understanding Swift Network and Service Requirements .....	188
Understanding Swift Ring Specifications .....	189
Ring Specifications in the Input Model .....	189
Replication Ring Parameters .....	190
Erasure Coded Rings .....	191
Selecting a Partition Power .....	192
Designing Storage Policies .....	194
Specifying Storage Policies .....	195
Designing Swift Zones .....	196
Using Server Groups to Specify Swift Zones .....	197
Specifying Swift Zones at Ring Level .....	198
Customizing Swift Service Configuration Files .....	199
Configuring Swift Container Rate Limit .....	199
Configuring Swift Account Server Logging Level .....	200
13. Alternative Configurations .....	202
SLES Compute Nodes .....	202
Entry-scale KVM with Ceph Model .....	204
Entry-scale KVM with Ceph Model with Two Networks .....	204
Using a Dedicated Lifecycle Manager Node .....	208
Specifying a dedicated lifecycle manager in your input model .....	209
Configuring HPE Helion OpenStack without DVR .....	212
Configuring HPE Helion OpenStack with Provider VLANs and Physical Routers Only .....	213
Considerations When Installing Two Systems on One Subnet .....	214

---

## **Part I. Planning**

---

# Table of Contents

1. Registering SUSE Linux .....	3
Registering SUSE Linux during the Installation .....	3
Registering SUSE Linux from the Installed System .....	3
Registering from the Installed System .....	3
Registering SUSE Linux during Automated Deployment .....	4
2. Hardware and Software Support Matrix .....	5
OpenStack Version Information .....	5
Supported Hardware .....	5
Supported Hardware Configurations .....	5
Cloud Scaling .....	5
Supported Software .....	6
Notes about Performance .....	6
Disk Calculator .....	6
Disk Calculator for Compute-Centric Deployments .....	6
Enter Input Parameters .....	7
Audit Logging Adjustment .....	9
Select the Deployment Model .....	9
Match to a Disk Model .....	10
Entry Scale Disk Models .....	10
MML Disk Models .....	11
Notes about disk sizing for Cinder bootable volumes .....	12
KVM Guest OS Support .....	12
ESX Guest OS Support .....	13
Ironic Guest OS Support .....	13
3. Recommended Hardware Minimums for the Example Configurations .....	14
Recommended Hardware Minimums for an Entry-scale KVM with VSA Model .....	14
Recommended Hardware Minimums for an Entry-scale KVM with Ceph Model .....	15
Recommended Hardware Minimums for an Entry-scale ESX, KVM with VSA Model .....	16
Recommended Hardware Minimums for an Entry-scale ESX, KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging .....	18
Recommended Hardware Minimums for an Ironic Flat Network Model .....	20
Recommended Hardware Minimums for an Entry-scale Swift Model .....	20
4. High Availability .....	24
High Availability Concepts Overview .....	24
Highly Available Cloud Infrastructure .....	24
High Availability of Controllers .....	24
High Availability Routing - Centralized .....	25
High Availability Routing - Distributed .....	26
Availability Zones .....	27
Compute with KVM .....	27
Nova Availability Zones .....	28
Compute with ESX Hypervisor .....	28
Block Storage with StoreVirtual VSA .....	28
Deploy VSA cluster across Availability Zones/Racks .....	29
Cinder Availability Zones .....	30
Object Storage with Swift .....	30
Highly Available Cloud Applications and Workloads .....	31
What is not Highly Available? .....	31
More Information .....	32
5. Third Party Integrations .....	33

---

# Chapter 1. Registering SUSE Linux

To get technical support and product updates, you need to register and activate your SUSE product with the SUSE Customer Center. It is recommended to register during the installation, since this will enable you to install the system with the latest updates and patches available. However, if you are offline or want to skip the registration step, you can register at any time later from the installed system.

## Note

In case your organization does not provide a local registration server, registering SUSE Linux requires a SUSE account. In case you do not have a SUSE account yet, go to the SUSE Customer Center home page (<https://scc.suse.com/>) to create one.

## Registering SUSE Linux during the Installation

To register your system, provide the E-mail address associated with the SUSE account you or your organization uses to manage subscriptions. In case you do not have a SUSE account yet, go to the SUSE Customer Center home page (<https://scc.suse.com/>) to create one.

Enter the Registration Code you received with your copy of SUSE Linux Enterprise Server. Proceed with Next to start the registration process.

By default the system is registered with the SUSE Customer Center. However, if your organization provides local registration servers you can either choose one from the list of auto-detected servers or provide the URL at "Register System via local SMT Server". Proceed with Next.

During the registration, the online update repositories will be added to your installation setup. When finished, you can choose whether to install the latest available package versions from the update repositories. This ensures that SUSE Linux Enterprise Server is installed with the latest security updates available. If you choose No, all packages will be installed from the installation media. Proceed with Next.

If the system was successfully registered during installation, YaST will disable repositories from local installation media such as CD/DVD or flash disks when the installation has been completed. This prevents problems if the installation source is no longer available and ensures that you always get the latest updates from the online repositories.

## Registering SUSE Linux from the Installed System

### Registering from the Installed System

If you have skipped the registration during the installation or want to re-register your system, you can register the system at any time using the YaST module Product Registration or the command line tool SUSEConnect.

#### Registering with YaST

To register the system start YaST+Software+Product Registration. Provide the E-mail address associated with the SUSE account you or your organization uses to manage subscriptions. In case you do not have a SUSE account yet, go to the SUSE Customer Center home page (<https://scc.suse.com/>) to create one.

Enter the Registration Code you received with your copy of SUSE Linux Enterprise Server. Proceed with Next to start the registration process.

By default the system is registered with the SUSE Customer Center. However, if your organization provides local registration servers you can either choose one from the list of auto-detected servers or provide the URL at Register System via local SMT Server. Proceed with Next.

### **Registering with SUSEConnect**

To register from the command line, use the command

```
sudo SUSEConnect -r <REGISTRATION_CODE> -e <EMAIL_ADDRESS>
```

Replace <REGISTRATION\_CODE> with the Registration Code you received with your copy of SUSE Linux Enterprise Server. Replace <EMAIL\_ADDRESS> with the E-mail address associated with the SUSE account you or your organization uses to manage subscriptions. To register with a local registration server, also provide the URL to the server:

```
sudo SUSEConnect -r <REGISTRATION_CODE> -e <EMAIL_ADDRESS> --url "https://suse_reg
```

## **Registering SUSE Linux during Automated Deployment**

If you deploy your instances automatically using AutoYaST, you can register the system during the installation by providing the respective information in the AutoYaST control file. Refer to [https://www.suse.com/documentation/sles-12/book\\_autoyast/data/createprofile\\_register.html](https://www.suse.com/documentation/sles-12/book_autoyast/data/createprofile_register.html) for details.

---

# Chapter 2. Hardware and Software Support Matrix

This document lists the details about the supported hardware and software for HPE Helion OpenStack 5.0

## Firmware Requirements

Before performing any installation or upgrade of a HPE Helion OpenStack release on HPE (ProLiant) servers, the Service Pack for ProLiant (SPP) should be applied to be compatible with latest releases in firmware. The Service Pack for ProLiant (SPP) can be downloaded from <http://www.hpe.com/info/spp>

## OpenStack Version Information

HPE Helion OpenStack 5.0 services have been updated to the OpenStack Newton [<http://www.openstack.org/software/mitaka>] release.

## Supported Hardware

For information about hardware supported in HPE Helion OpenStack 5.0, see HPE Helion Ready Solution Catalog [<http://docs.hpcloud.com/#hrc/helionReady.html>].

## Supported Hardware Configurations

HPE Helion OpenStack 5.0 supports the following hardware configurations for a deployment.

### Storage Interconnects/Protocols

- 10Gb Ethernet.
- Software iSCSI
- FibreChannel (FC)

### Multipath

HPE Helion OpenStack 5.0 supports Fibre Channel and FCoE boot from SAN in multipath environments. The following list outlines the current limitations based on testing:

- Emulex based LPE1605 Native Fibre Channel - Up to 1024 paths during boot
- Qlogic based SN100Q Native Fibre Channel - Up to 1024 paths during boot
- Emulex Flex Fabric 650 series - Up to 1024 paths during boot
- Emulex Flex Fabric 554FLB - Up to 1024 paths during boot
- Qlogic Flex Fabric 536 and 630 series - Up to 1024 paths during boot

## Cloud Scaling

In HPE Helion OpenStack 5.0 a total of 200 total compute nodes in a single region across any of the following hypervisors is supported:

- VMware ESX
- Linux for HPE Helion/KVM
- Red Hat Enterprise Linux/KVM

You can distribute the compute nodes in any number of deployments as long as the total is no more than 200. Example: 100 ESX + 100 RHEL/KVM or 75 ESX + 25 HPE Linux/KVM + 100 RHEL/KVM.

HPE Helion OpenStack 5.0 supports a total of 8000 virtual machines across a total of 200 compute nodes.

HPE Helion OpenStack 5.0 supports 100 baremetal ironic nodes in a single region.

## Supported Software

### Supported ESXi versions

HPE Helion OpenStack 5.0 currently supports the following ESXi versions:

- ESXi version 5.5 (Update 3)
- ESXi version 6.0
- ESXi version 6.0 (Update 1b)

The following are the requirements for your vCenter server:

- Software
  - vCenter 5.5 Update 3 and above (It is recommended to run the same server version as the ESXi hosts)
- License Requirements
  - vSphere Enterprise Plus license

## Notes about Performance

We have the following recommendations to ensure good performance of your cloud environment:

- On the control plane nodes, you will want good I/O performance. Your array controllers must have cache controllers and we advise against the use of RAID-5.
- On compute nodes, the I/O performance will influence the virtual machine start-up performance. We also recommend the use of cache controllers in your storage arrays.
- If you are using dedicated object storage (Swift) nodes, in particular the account, container, and object servers, we recommend that your storage arrays have cache controllers.
- For best performance, set the the servers power management setting in the iLO to OS Control Mode. This power mode setting is only available on servers that include the HP Power Regulator.

## Disk Calculator

### Disk Calculator for Compute-Centric Deployments

This topic provides guidance on how to estimate the amount of disk space required for a compute-centric HPE Helion OpenStack deployment. To accurately estimate the disk space needed, it is important to

understand how Helion utilizes resources. Although there are a variety of factors, including the number of compute nodes, a large portion of the utilization is driven by operational tools, such as monitoring, metering, and logging.

## **Important**

The disk calculator does not accurately estimate a Swift-centric deployment at this time. For more information on Swift, see the section called “Recommended Hardware Minimums for an Entry-scale Swift Model”.

The usage of disk space by operational tools can be estimated from the following parameters:

- **Number of compute nodes + Number of VM's running on each compute node**
- **Number of services being monitored or metered + Amount of logs created**
- **Retention periods for operational data** (for Elastic Search, Vertica/InfluxDB, and Kafka)

## **Important**

If you also enable auditing, follow the steps in the the section called “ Audit Logging Adjustment ” section to enter additional input parameters.

### **Disk Estimation Process**

HPE Helion OpenStack provides entry scale and scale-out models for deployment. This disk estimation tool, currently in a spreadsheet form, helps you decide which disk model to start from as well as what customizations you need to meet your deployment requirements. The disk estimation process also provides default settings and minimum values for the parameters that drive disk size.

## **Important**

Kafka is the queuing system used to process metering monitoring and logging (MML) data. Kafka stores the queued data on disk, so the disk space available will have a large impact on the amount of data the MML systems can process. Providing less than the minimum disk space for Kafka will result in loss of MML data and can affect other components on the control plane. The default for Kafka is 1 hour which is 17 GB.

### **To estimate the disk sizes required for your deployment:**

1. the section called “ Enter Input Parameters ”
2. If you also enable auditing, follow the steps in the the section called “ Audit Logging Adjustment ” section to enter additional input parameters.
3. the section called “Select the Deployment Model”
4. the section called “Match to a Disk Model”Match the selected deployment to a disk model example.

## **Enter Input Parameters**

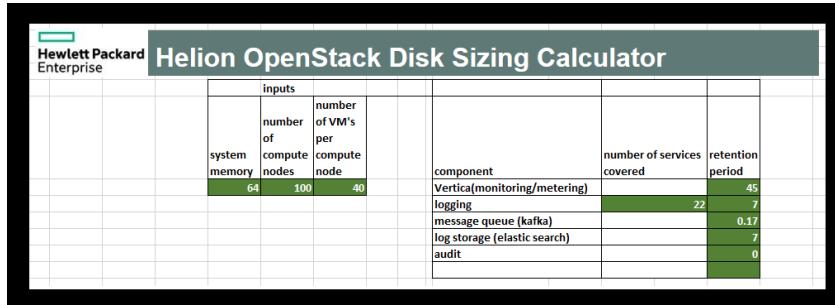
The Disk Calculator spreadsheet automatically displays the minimum requirements for the components that define disk size. You can replace the default values with either the number you have to work with or the number that you want to support.

## **Important**

If you want to enable audit logging, follow the steps in the the section called “ Audit Logging Adjustment ” section to enter additional input parameters.

<b>Input Parameter</b>	<b>Default</b>	<b>Minimum</b>
System Memory	64 GB	64 GB
Compute Nodes	100	100
VMs per Compute Node	40	40
Component: Vertica	45 days retention period	30 days
Component: Logging	22 services covered 7 days retention period	7 days retention period
Component: Kafka (message queue)	0.17 of an hour retention period	0.042 of an hour retention period
Component: Elastic Search (log storage)	7 days retention period	7 days retention period
Component: Audit	0 days retention period	0 days retention period

The following diagram shows the input parameters in the spreadsheet.



To provide the parameters required to estimate disk size:

1. Open the disk calculator spreadsheet.
2. At the bottom of the spreadsheet, click on the **Draft Sizing Tool4** tab.
3. To set the server RAM size, replace the default value in the **System Memory** field.
4. To set the number of compute nodes, replace the default in the **Compute Nodes** field.
5. To set the average number of virtual machines per compute node, replace the default in the **VM's per Compute Node** field.
6. To set the number of days you want the metering and logging files retained, replace the default in the **Vertica Retention Period** field.
7. To set logging values, replace the default in **Number of Services Covered** and **Retention Period**.

## **Important**

If you enable additional logging of services than those set by default, then you must increase the number in the **Logging Number of Services Field**.

8. To set a value for Kafka messages to be retained, replace the default in the **Kafka Retention Period** field.
9. To set a value for Elastic Search log file retention, replace the default in the **Elastic Search Retention Period** field.
10. To set a value for Audit logging file retention, replace the default in the **Audit Retention Period** field.

## Audit Logging Adjustment

If you want to enable audit logging, you must enter additional input parameters to ensure there is enough room to retain the audit logs. The following diagram shows the parameters you need to specify in the Disk Calculator spreadsheet.

			API/Core Services	Networking	Swift - Images	MMLB	MySQL/RabbitMQ
	number of services on cluster		13	10	5	9	6
	number Audit Enabled services on cluster		9	1	1	2	
Filesystem	used by	subcomponents					
/					60		
/var/crash					64		
/var/log	logging		175	134	67	121	81
/var/lib/mysql	monitoring, core services		0	0	0	0	60
/var/lib/rabbitmq	logging, core services		0	0	0	0	26
/var/vertica	MM		0	0	0	362	0
/var/kafka	MML		0	0	0	141	0
/var/lib/elasticsearch	LB		0	0	0	246	0
	logging BURA						
/var/lib/zookeeper	monitoring, logging, metering		0	0	0	1	0
/var/audit	logging		7	0	3	1	
/var/lib/glance/work_dir	glance		0	0	0	0	0

To add audit logging to disk size calculations:

1. Determine which services you have enabled to collect audit logging information. This is part of HLM configuration.
2. Enter the number of Audit Enabled services on cluster. Auditing is disabled by default, so these values will initially be 0. If audit logging is enabled, initial suggested values would be 9 for API/Core Services, 1 for Networking, 1 for Swift, and 2 for MMLB.

### Important

If you enable logging for services beyond the defaults, you must change the **Number of Services on a Cluster** field in the spreadsheet. It is recommended that you increase the total services covered as well as increment the number on the appropriate cluster. For example, if you enable Apache logs on the core services, then the total would increase to 23 and the api/core services entry would change from 13 to 14.

3. To include Glance image space in your estimation, determine the size of the images that will be cached.
4. Enter the total size needed to store Glance images in the **/var/lib/glance/work\_dir** field.

## Select the Deployment Model

To decide which architecture will meet all of your requirements, use the values given in the Disk Calculator spreadsheet. Keeping in mind the rough scale you expect to target as well as any need to separate services, choose an Entry Scale, Entry Scale MML, or Mid Scale deployment. Once you have chosen a deployment you can match it to the sample disk models in the the section called “Match to a Disk Model” section. The

following diagram shows the deployment options that are recommended if you use the default values in the Disk Calculator spreadsheet.

<b>Entry Scale</b>	<b>750</b>			
<b>Entry Scale MML</b>	216 573 252			
<b>Mid Scale</b>	216	195	195	573 252
	API/Core Services	Networking	Swift	MMLB MySQL/RabbitMQ

For example, in the above diagram, if you wanted to choose an Entry Scale MML deployment, the calculator recommends the following disk sizes:

- 216GB for API/Core Service
- 216GB for Neutron (networking)
- 216GB for Swift (storage)
- 573GB for MMLB
- 252GB for MySQL/RabbitMQ

## Match to a Disk Model

For each of the entry-scale and scale-out cloud models, there is a set of associated disk models that can be used as the basis for your deployment. These models provide examples of potential parameters for operational tools and are expected to be used as the starting point for actual deployments. Since each deployment can vary greatly, the disk calculator spreadsheet provides a way to create the basic disk model and customize it to fit the specific parameters your deployment. Once you have estimated disk sizes and chosen a deployment architecture, you can choose which example disk partitioning file to use from the tables below. Keep in mind if you are enabling more options than are listed in the Disk Calculator, or if you want to plan for growth, you will need to manually adjust parameters as necessary.

Disk models are provided for each deployment option based on the expected size of the disk available to the control plane nodes. The available space is then partitioned by percentage to be allocated to each of the required volumes on the control plane. Each of the disk models is targeted at a specific set of parameters which can be found in the following tables:

- Entry Scale: 600 GB, 1 TB
- Mid Scale/ Entry Scale MML Servers: 600 GB, 2 TB, 4.5 TB

## Entry Scale Disk Models

These models include a single cluster of control plane nodes and all services.

### 600GB Entry Scale

Component	Parameters
compute nodes	100  This model provides lower than recommended retention and should only be used for POC deployments.

### 1TB Entry Scale

<b>Component</b>	<b>Parameters</b>
compute nodes	100
local logging var/log	7 day retention
metering/monitoring /var/vertica	45 day retention
centralized logging /var/lib/elasticsearch	7 day retention
Kafka Message Queue /var/kafka	4 hour retention

## MML Disk Models

These mid-scale and entry-scale MML models include separate control plane nodes for core services, metering/monitoring/logging, and MySQL/RabbitMQ. Optionally you can also separate out Swift (storage) and Neutron (networking). MML servers are the ones that will need modification based on the scale and operational parameters.

### 600GB MML Server

<b>Component</b>	<b>Parameters</b>
compute nodes	100
local logging var/log	7 day retention
metering/monitoring /var/vertica	30 day retention
centralized logging /var/lib/elasticsearch	<p style="text-align: center;"><b>Caution</b></p> <p style="text-align: center;">45 days is the default minimum.</p>
Kafka Message Queue /var/kafka	7 day retention

### 2TB MML Server

<b>Component</b>	<b>Parameters</b>
compute nodes	200
local logging var/log	7 day retention
metering/monitoring	45 day retention

Component	Parameters
/var/vertica	
centralized logging	7 day retention
/var/lib/elasticsearch	
Kafka Message Queue	12 hour retention
/var/kafka	

#### 4.5TB MML Server

Component	Parameters
compute nodes	200
local logging	7 day retention
var/log	
metering/monitoring	45 day retention
/var/vertica	
centralized logging	45 day retention
/var/lib/elasticsearch	
Kafka Message Queue	12 hour retention
/var/kafka	

## Notes about disk sizing for Cinder bootable volumes

When creating your disk model for nodes that will have the cinder volume role make sure that there is sufficient disk space allocated for a temporary space for image conversion if you will be creating bootable volumes.

By default, Cinder uses `/var/lib/cinder` for image conversion and this will be on the root filesystem unless it is explicitly separated. You can ensure there is enough space by ensuring that the root file system is sufficiently large, or by creating a logical volume mounted at `/var/lib/cinder` in the disk model when installing the system.

If you have post-installation issues with creating bootable volumes, see the *FIXME: broken external xref* documentation for steps to resolve these issues.

## KVM Guest OS Support

A **Verified** Guest OS has been tested by HPE and appears to function properly as a Nova compute virtual machine on HPE Helion OpenStack 5.0.

A **Certified** Guest OS has been officially tested by the operating system vendor, or by HPE under the vendor's authorized program, and will be supported by the operating system vendor as a Nova compute virtual machine on HPE Helion OpenStack 5.0.

KVM Guest Operating System	Verified	Certified
Windows Server 2008		Yes

KVM Guest Operating System	Verified	Certified
Windows Server 2008 R2		Yes
Windows Server 2012		Yes
Windows Server 2012 R2		Yes
CentOS 6.7	Yes	
CentOS 7.1	Yes	
CoreOS - Stable	Yes	
Debian 7.9	Yes	
Debian 8.2	Yes	
RHEL 6.7	Yes	
RHEL 7.1	Yes	
RHEL Atomic	Yes	
SLES 11 SP4	Yes	
SLES 12	Yes	
Ubuntu 14.04	Yes	

## ESX Guest OS Support

For ESX, refer to the VMware Compatibility Guide [<http://www.vmware.com/resources/compatibility/search.php?>  
action=search&deviceCategory=software&advancedORbasic=advanced&maxDisplay-  
Rows=50&key=&productId=4&gos\_vmw\_product\_release%5B  
%5D=90&datePosted=-1&partnerId%5B%5D=-1&os\_bits=-1&os\_use%5B%5D=-1&os\_family%5B  
%5D=-1&os\_type%5B%5D=-1&rorre=0].

## Ironic Guest OS Support

A **Verified** Guest OS has been tested by HPE and appears to function properly as a bare metal instance on HPE Helion OpenStack 5.0.

A **Certified** Guest OS has been officially tested by the operating system vendor, or by HPE under the vendor's authorized program, and will be supported by the operating system vendor as a bare metal instance on HPE Helion OpenStack 5.0.

Ironic Guest Operating System	Verified	Certified
RHEL 6.7	Yes	
RHEL 7.1	Yes	
Ubuntu 14.04	Yes	

---

# Chapter 3. Recommended Hardware Minimums for the Example Configurations

## Firmware Requirements

Before performing any installation or upgrade of a HPE Helion OpenStack release on HPE (ProLiant) servers, the Service Pack for ProLiant (SPP) should be applied to be compatible with latest releases in firmware. The Service Pack for ProLiant (SPP) can be downloaded from <http://www.hpe.com/info/spp>

## Recommended Hardware Minimums for an Entry-scale KVM with VSA Model

These recommended minimums are based on the included Chapter 10, *Example Configurations* included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

### Note

The disk requirements detailed below can be met with logical drives, logical volumes, or external storage such as a 3PAR array.

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"><li>• 1 x 600 GB (minimum) - operating system drive</li><li>• 2 x 600 GB (minimum) - Data drive</li></ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
Compute	Compute	1-3	2 X 600 GB (minimum)	32 GB (memory must be sized based on the virtual machine in-	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64) with hard-

Recommended Hardware Minimums  
for the Example Configurations

---

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
				stances hosted on the Compute node)		ware virtualization support. The CPU cores must be sized based on the VM instances hosted by the Compute node.
Block Storage (Optional) (Ceph)	VSA or OSD	0 or 3 (which will provide the recommended redundancy)	3 X 600 GB (minimum) See Chapter 2, <i>Pre-Installation Checklist</i> for more details.	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

For more details about the supported network requirements, see Chapter 10, *Example Configurations*.

## Recommended Hardware Minimums for an Entry-scale KVM with Ceph Model

The table below lists out the key characteristics needed per server role for this configuration.

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"> <li>1 x 600 GB (minimum) - operating system drive</li> <li>2 x 600 GB (minimum) - Data drive</li> </ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
Compute (KVM hypervisor)	Compute	1-3	2 X 600 GB (minimum)	32 GB (memory must be sized based on the VM instances hosted by the Compute node)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

Recommended Hardware Minimums  
for the Example Configurations

---

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			on the virtual machine instances hosted on the Compute node)			tel x86_64) with hardware virtualization support. The CPU cores must be sized based on the VM instances hosted by the Compute node.
CEPH-OSD	ceph-osd	0 or 3 (which will provide the recommended redundancy)	3 X 600 GB (minimum)	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
RADOS Gateway	radosgw	2	2 x 600 GB (minimum)	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

## Recommended Hardware Minimums for an Entry-scale ESX, KVM with VSA Model

These recommended minimums are based on the included included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

HPE Helion OpenStack 5.0 currently supports the following ESXi versions:

- ESXi version 5.5 (Update 3)
- ESXi version 6.0
- ESXi version 6.0 (Update 1b)

The following are the requirements for your vCenter server:

- Software
  - vCenter 5.5 Update 3 and above (It is recommended to run the same server version as the ESXi hosts)
- License Requirements
  - vSphere Enterprise Plus license

Recommended Hardware Minimums  
for the Example Configurations

---

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"> <li>1 x 600 GB (minimum) - operating system drive</li> <li>2 x 600 GB (minimum) - Data drive</li> </ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
Compute (ESXi hypervisor)		2	2 X 1 TB (minimum, shared across all nodes)	128 GB (minimum)	2 x 10 Gbit/s +1 NIC (for DC access)	16 CPU (64-bit) cores total (Intel x86_64)
Compute (KVM hypervisor)	kvm-compute	1-3	2 X 600 GB (minimum)	32 GB (memory must be sized based on the virtual machine instances hosted on the Compute node)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64) with hardware virtualization support. The CPU cores must be sized based on the VM instances hosted by the Compute node.
Block Storage (Optional)	VSA	0 or 3 (which will provide the recommended redundancy)	3 X 600 GB (minimum) See Chapter 2, <i>Pre-Installation Checklist</i> for more details.	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

# Recommended Hardware Minimums for an Entry-scale ESX, KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging

These recommended minimums are based on the included included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

HPE Helion OpenStack 5.0 currently supports the following ESXi versions:

- ESXi version 5.5 (Update 3)
- ESXi version 6.0
- ESXi version 6.0 (Update 1b)

The following are the requirements for your vCenter server:

- Software
  - vCenter 5.5 Update 3 and above (It is recommended to run the same server version as the ESXi hosts)
- License Requirements
  - vSphere Enterprise Plus license

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Core-API Controller	2	<ul style="list-style-type: none"> <li>• 1 x 600 GB (minimum) - operating system drive</li> <li>• 2 x 300 GB (minimum) - Swift drive</li> </ul>	128 GB	2 x 10 Gbit/s with PXE Support	24 CPU (64-bit) cores total (Intel x86_64)
	DBMQ Cluster	3	<ul style="list-style-type: none"> <li>• 1 x 600 GB (minimum) - operating system drive</li> </ul>	96 GB	2 x 10 Gbit/s with PXE Support	24 CPU (64-bit) cores total (Intel x86_64)

Recommended Hardware Minimums  
for the Example Configurations

---

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			<ul style="list-style-type: none"> <li>• 1 x 300 GB (minimum) - MySQL drive</li> </ul>			
	Metering Mon/Log Cluster	3	<ul style="list-style-type: none"> <li>• 1 x 600 GB (minimum) - operating system drive</li> </ul>	128 GB	2 x 10 Gbit/s with one PXE enabled port	24 CPU (64-bit) cores total (Intel x86_64)
Compute (ESXi hypervisor)		2 (minimum)	2 X 1 TB (minimum, shared across all nodes)	64 GB (memory must be sized based on the virtual machine instances hosted on the Compute node)	2 x 10 Gbit/s +1 NIC (for Data Center access)	16 CPU (64-bit) cores total (Intel x86_64)
Compute (KVM hypervisor)	kvm-compute	1-3	2 X 600 GB (minimum)	32 GB (memory must be sized based on the virtual machine instances hosted on the Compute node)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64) with hardware virtualization support. The CPU cores must be sized based on the VM instances hosted by the Compute node.
Block Storage (Optional)	VSA	0 or 3 (which will provide the recommended redundancy)	3 X 600 GB (minimum) See Chapter 2, <i>Pre-Installation Checklist</i> for more details.	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

# Recommended Hardware Minimums for an Ironic Flat Network Model

When using the `agent_i1o` driver, you should ensure that the most recent iLO controller firmware is installed. A recommended minimum for the iLO4 controller is version 2.30.

The recommended minimum hardware requirements are based on the Chapter 10, *Example Configurations* included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"><li>• 1 x 600 GB (minimum) - operating system drive</li><li>• 2 x 600 GB (minimum) - Data drive</li></ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
Compute	Compute	1	1 X 600 GB (minimum)	16 GB	2 x 10 Gbit/s with one PXE enabled port	16 CPU (64-bit) cores total (Intel x86_64)

For more details about the supported network requirements, see Chapter 10, *Example Configurations*.

# Recommended Hardware Minimums for an Entry-scale Swift Model

These recommended minimums are based on the included Chapter 10, *Example Configurations* included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

The `entry-scale-swift` example runs the Swift proxy, account and container services on the three controller servers. However, it is possible to extend the model to include the Swift proxy, account and container services on dedicated servers (typically referred to as the Swift proxy servers). If you are using this model, we have included the recommended Swift proxy servers specs in the table below.

Recommended Hardware Minimums  
for the Example Configurations

---

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"> <li>1 x 600 GB (minimum) - operating system drive</li> <li>2 x 600 GB (minimum) - Swift account/container data drive</li> </ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
Swift Object	swobj	3	<p>If using x3 replication only:</p> <ul style="list-style-type: none"> <li>1 x 600 GB (minimum, see considerations at bottom of page for more details)</li> </ul> <p>If using Erasure Codes only or a mix of x3 replication and Erasure Codes:</p> <ul style="list-style-type: none"> <li>6 x 600 GB (minimum, see considerations at bottom of page for more details)</li> </ul>	32 GB (see considerations at bottom of page for more details)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

Recommended Hardware Minimums  
for the Example Configurations

---

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			<p><b>Note</b></p> <p>The disk speeds (RPM) chosen should be consistent within the same ring or storage policy. It's best to not use disks with mixed disk speeds within the same Swift ring.</p>			
Swift Proxy, Account, and Container	swpac	3	2 x 600 GB (minimum, see considerations at bottom of page for more details)	64 GB (see considerations at bottom of page for more details)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

### **Considerations for your Swift object and proxy, account, container servers RAM and disk capacity needs**

Swift can have a diverse number of hardware configurations. For example, a Swift object server may have just a few disks (minimum of 6 for erasure codes) or up to 70 and beyond. The memory requirement needs to be increased as more disks are added. The general rule of thumb for memory needed is 0.5 GB per TB of storage. For example, a system with 24 hard drives at 8TB each, giving a total capacity of 192TB, should use 96GB of RAM. However, this does not work well for a system with a small number of small hard drives or a very large number of very large drives. So, if after calculating the memory given this guideline, if the answer is less than 32GB then go with 32GB of memory minimum and if the answer is over 256GB then use 256GB maximum, no need to use more memory than that.

When considering the capacity needs for the Swift proxy, account, and container (PAC) servers, you should calculate 2% of the total raw storage size of your object servers to specify the storage required for the PAC servers. So, for example, if you were using the example we provided earlier and you had an object server setup of 24 hard drives with 8TB each for a total of 192TB and you had a total of 6 object servers, that would give a raw total of 1152TB. So you would take 2% of that, which is 23TB, and ensure that much storage capacity was available on your Swift proxy, account, and container (PAC) server cluster. If you had a cluster of three Swift PAC servers, that would be ~8TB each.

Another general rule of thumb is that if you are expecting to have more than a million objects in a container then you should consider using SSDs on the Swift PAC servers rather than HDDs.

---

# Chapter 4. High Availability

## High Availability Concepts Overview

A highly available (HA) cloud ensures that a minimum level of cloud resources are always available on request, which results in uninterrupted operations for users.

In order to achieve this high availability of infrastructure and workloads, we define the scope of HA to be limited to protecting these only against single points of failure (SPOF). Single points of failure include:

- **Hardware SPOFs:** Hardware failures can take the form of server failures, memory going bad, power failures, hypervisors crashing, hard disks dying, NIC cards breaking, switch ports failing, network cables loosening, and so forth.
- **Software SPOFs:** Server processes can crash due to software defects, out-of-memory conditions, operating system kernel panic, and so forth.

By design, HPE Helion OpenStack strives to create a system architecture resilient to SPOFs, and does not attempt to automatically protect the system against multiple cascading levels of failures; such cascading failures will result in an unpredictable state. Hence, the cloud operator is encouraged to recover and restore any failed component, as soon as the first level of failure occurs.

## Highly Available Cloud Infrastructure

The highly available cloud infrastructure consists of the following:

- High Availability of Controllers
- Availability Zones
- Compute with KVM
- Nova Availability Zones
- Compute with ESX
- Block Storage with StoreVirtual VSA
- Object Storage with Swift

## High Availability of Controllers

The HPE Helion OpenStack installer deploys highly available configurations of OpenStack cloud services, resilient against single points of failure.

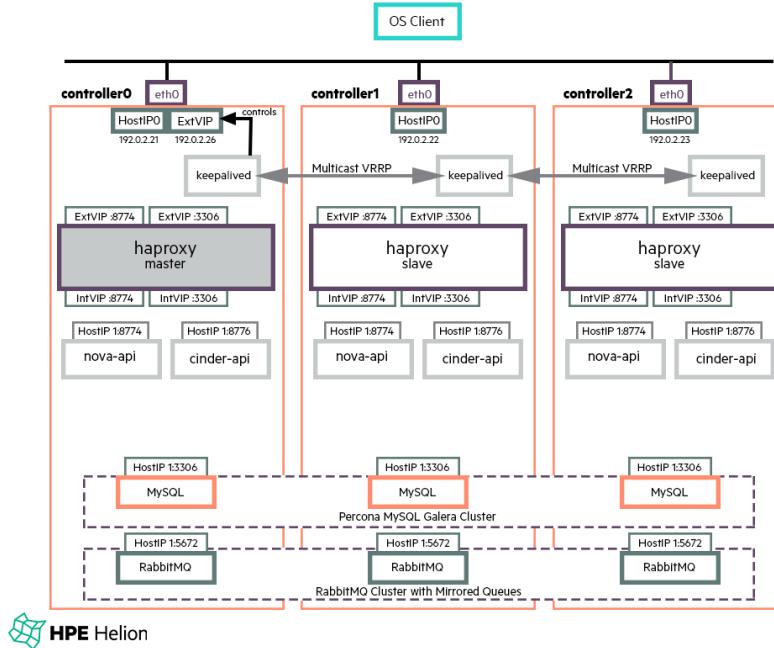
The high availability of the controller components comes in two main forms.

- Many services are stateless and multiple instances are run across the control plane in active-active mode. The API services (nova-api, cinder-api, etc.) are accessed through the HA proxy load balancer whereas the internal services (nova-scheduler, cinder-scheduler, etc.), are accessed through the message broker. These services use the database cluster to persist any data.

## Note

The HA proxy load balancer is also run in active-active mode and keepalived (used for Virtual IP (VIP) Management) is run in active-active mode, with only one keepalived instance holding the VIP at any one point in time.

- The high availability of the message queue service and the database service is achieved by running these in a clustered mode across the three nodes of the control plane: RabbitMQ cluster with Mirrored Queues and Percona MySQL Galera cluster.



The above diagram illustrates the HA architecture with the focus on VIP management and load balancing. It only shows a subset of active-active API instances and does not show examples of other services such as nova-scheduler, cinder-scheduler, etc.

In the above diagram, requests from an OpenStack client to the API services are sent to VIP and port combination; for example, 192.0.2.26:8774 for a Nova request. The load balancer listens for requests on that VIP and port. When it receives a request, it selects one of the controller nodes configured for handling Nova requests, in this particular case, and then forwards the request to the IP of the selected controller node on the same port.

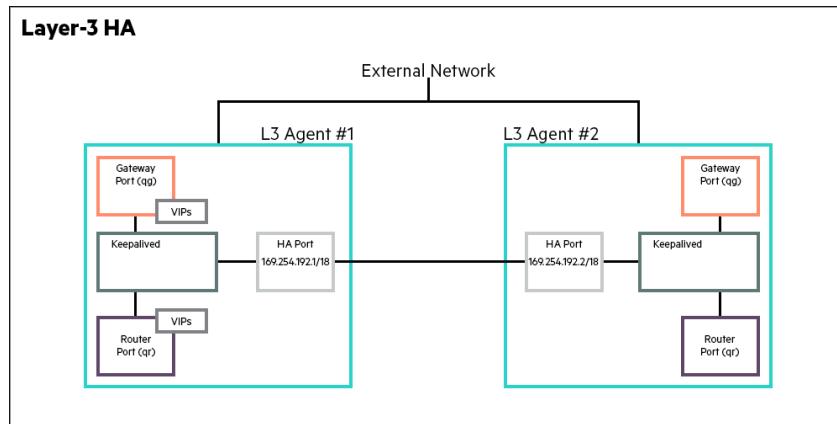
The nova-api service, which is listening for requests on the IP of its host machine, then receives the request and deals with it accordingly. The database service is also accessed through the load balancer. RabbitMQ, on the other hand, is not currently accessed through VIP/HA proxy as the clients are configured with the set of nodes in the RabbitMQ cluster and failover between cluster nodes is automatically handled by the clients.

## High Availability Routing - Centralized

Incorporating High Availability into a system involves implementing redundancies in the component that is being made highly available. In Centralized Virtual Router (CVR), that element is the Layer 3 agent

a.k.a L3 agent. By making L3 agent highly available, upon failure all HA routers are migrated from the primary L3 agent to a secondary L3 agent. The implementation efficiency of an HA subsystem is measured by the number of packets that are lost when the secondary L3 agent is made the master.

In HPE Helion OpenStack, the primary and secondary L3 agents run continuously, and failover involves a rapid switchover of mastership to the secondary agent (IEFT RFC 5798). The failover essentially involves a switchover from a already running master to already running slave. This substantially reduces the latency of the HA. The mechanism used by the master and the slave to implement a failover is implemented using Linux's pacemaker HA resource manager. This CRM (Cluster resource manager) uses VRRP (Virtual Router Redundancy Protocol) to implement the HA mechanism. VRRP is a industry standard protocol and defined in RFC 5798.



L3 HA uses of VRRP comes with several benefits.

The primary benefit is the failover mechanism does not involve interprocess communication overhead (order of 10s of seconds). By not using an RPC mechanism to invoke the secondary agent to assume the primary agents role enables VRRP to achieve failover within 1-2 seconds.

In VRRP, the primary and secondary routers are all active. As the routers are running, it is a matter of making the router aware of its primary/master status. This switchover takes less than 2 seconds instead of 60+ seconds it would have taken to start a backup router and failover.

The failover depends upon a heartbeat link between the primary and secondary. That link in HPE Helion OpenStack 3.0 uses keepalived package of the pacemaker resource manager. The heartbeats are sent at a 2 second intervals between the primary and secondary. As per the VRRP protocol, if the secondary does not hear from the master after 3 intervals, it assumes the function of the primary.

Further, all the routable IP addresses i.e. the VIPs (virtual IPs) are assigned to the primary agent.

For information on more creating HA routers, see:

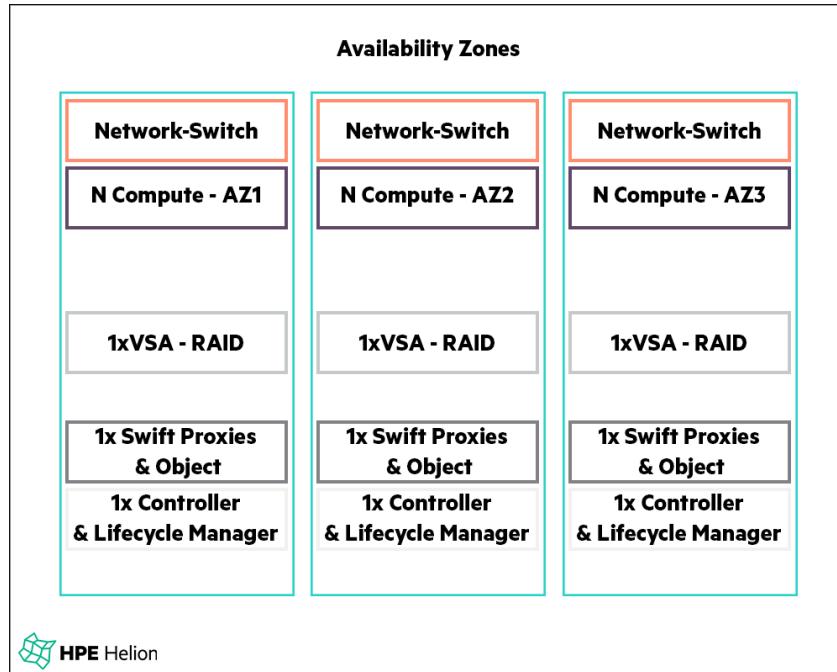
## High Availability Routing - Distributed

The OpenStack Distributed Virtual Router (DVR) function delivers HA through its distributed architecture. The one centralized function remaining is source network address translation (SNAT), where high availability is provided by DVR SNAT HA.

DVR SNAT HA is enabled on a per router basis and requires that two or more L3 agents capable of providing SNAT services be running on the system. If a minimum number of L3 agents is configured to 1 or lower, the neutron server will fail to start and a log message will be created. The L3 Agents must be running on a control-plane node, L3 agents running on a compute node do not provide SNAT services.

For more information on creating HA routers, see: .

## Availability Zones



While planning your OpenStack deployment, you should decide on how to zone various types of nodes - such as compute, block storage, and object storage. For example, you may decide to place all servers in the same rack in the same zone. For larger deployments, you may plan more elaborate redundancy schemes for redundant power, network ISP connection, and even physical firewalling between zones (*this aspect is outside the scope of this document*).

HPE Helion OpenStack offers APIs, CLIs and Horizon UIs for the administrator to define and user to consume, availability zones for Nova, Cinder and Swift services. This section outlines the process to deploy specific types of nodes to specific physical servers, and makes a statement of available support for these types of availability zones in the current release.

### Note

By default, HPE Helion OpenStack is deployed in a single availability zone upon installation. Multiple availability zones can be configured by an administrator post-install, if required. Refer to the Chapter 5: Scaling [<http://docs.openstack.org/openstack-ops/content/scaling.html>] (in the OpenStack Operations Guide for more information).

## Compute with KVM

You can deploy your KVM nova-compute nodes either during initial installation, or by adding compute nodes post initial installation.

While adding compute nodes post initial installation, you can specify the target physical servers for deploying the compute nodes.

Learn more about .

## Nova Availability Zones

Nova host aggregates and Nova availability zones can be used to segregate Nova compute nodes across different failure zones.

## Compute with ESX Hypervisor

Compute nodes deployed on ESX Hypervisor can be made highly available using the HA feature of VMware ESX Clusters. For more information on VMware HA, please refer to your VMware ESX documentation.

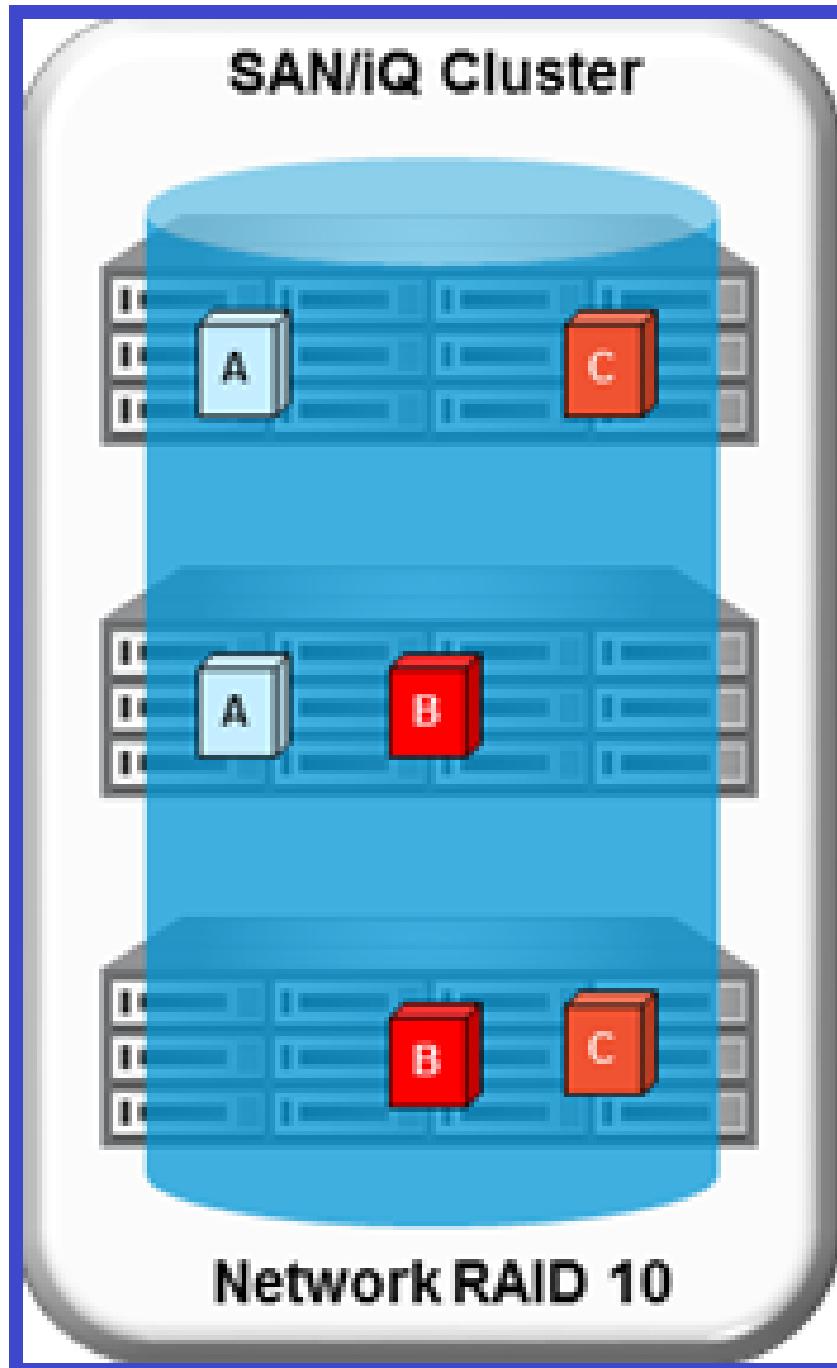
## Block Storage with StoreVirtual VSA

Highly available Cinder block storage volumes are provided by the network RAID 10 implementation in the HPE StoreVirtual VSA software. You can deploy the VSA nodes in three node cluster and specify Network RAID 10 protection for Cinder volumes.

The underlying SAN/iQ operating system of the StoreVirtual VSA ensures that the two-way replication maintains two mirrored copies of data for each volume.

This Network RAID 10 capability ensures that failure of any single server does not cause data loss, and maintains data access to the clients.

Furthermore, each of the VSA nodes of the cluster can be strategically deployed in different zones of your data center for maximum redundancy and resiliency. For more information on how to deploy VSA nodes on desired target servers, refer to the the section called “Configuring for VSA Block Storage Backend” document.



## Deploy VSA cluster across Availability Zones/ Racks

In the Availability Zone image above, the input model example has 3 VSA servers in three different server-groups (Racks) (server-groups are logical separations). You can configure these server-groups in different physical Racks to provide the required hardware isolation. See input model examples for the section called “Entry-scale KVM with VSA Model”, the section called “Entry-scale ESX, KVM with VSA

Model with Dedicated Cluster for Metering, Monitoring, and Logging”, and FIXME: the section called “Mid-scale KVM with VSA Model”

The recommended configuration for a VSA Cluster is to use RAID 10 with 3 mirrors to guarantee that data is replicated across 3 VSA nodes spreading across AZs/Racks. Using one of the two options below, you can expand storage capacity by adding 3 more nodes.

1. Add new three VSA nodes to the existing cluster and ensure that each new VSA node is on different AZ/Rack.

### Note

There is a 1500 volumes limit per VSA cluster.

2. Create a new VSA cluster with the 3 new nodes.

## Cinder Availability Zones

Cinder availability zones are not supported for general consumption in the current release.

## Object Storage with Swift

High availability in Swift is achieved at two levels.

### Control Plane

The Swift API is served by multiple Swift proxy nodes. Client requests are directed to all Swift proxy nodes by the HA Proxy load balancer in round-robin fashion. The HA Proxy load balancer regularly checks the node is responding, so that if it fails, traffic is directed to the remaining nodes. The Swift service will continue to operate and respond to client requests as long as at least one Swift proxy server is running.

If a Swift proxy node fails in the middle of a transaction, the transaction fails. However it is standard practice for Swift clients to retry operations. This is transparent to applications that use the python-swift-client library.

The entry-scale example cloud models contain three Swift proxy nodes. However, it is possible to add additional clusters with additional Swift proxy nodes to handle a larger workload or to provide additional resiliency.

### Data

Multiple replicas of all data is stored. This happens for account, container and object data. The example cloud models recommend a replica count of three. However, you may change this to a higher value if needed.

When Swift stores different replicas of the same item on disk, it ensures that as far as possible, each replica is stored in a different zone, server or drive. This means that if a single server of disk drives fails, there should be two copies of the item on other servers or disk drives.

If a disk drive is failed, Swift will continue to store three replicas. The replicas that would normally be stored on the failed drive are “handed off” to another drive on the system. When the failed drive is replaced, the data on that drive is reconstructed by the replication process. The replication process re-creates the “missing” replicas by copying them to the drive using one of the other remaining replicas. While this is happening, Swift can continue to store and retrieve data.

# Highly Available Cloud Applications and Workloads

Projects writing applications to be deployed in the cloud must be aware of the cloud architecture and potential points of failure and architect their applications accordingly for high availability.

Some guidelines for consideration:

1. Assume intermittent failures and plan for retries
  - **OpenStack Service APIs:** invocations can fail - you should carefully evaluate the response of each invocation, and retry in case of failures.
  - **Compute:** VMs can die - monitor and restart them
  - **Network:** Network calls can fail - retry should be successful
  - **Storage:** Storage connection can hiccup - retry should be successful
2. Build redundancy into your application tiers
  - • Replicate VMs containing stateless services such as Web application tier or Web service API tier and put them behind load balancers (you must implement your own HA Proxy type load balancer in your application VMs until HPE Helion OpenStack delivers the LBaaS service).
  - Boot the replicated VMs into different Nova availability zones.
  - If your VM stores state information on its local disk (Ephemeral Storage), and you cannot afford to lose it, then boot the VM off a Cinder volume.
  - Take periodic snapshots of the VM which will back it up to Swift through Glance.
  - Your data on ephemeral may get corrupted (but not your backup data in Swift and not your data on Cinder volumes).
  - Take regular snapshots of Cinder volumes and also back up Cinder volumes or your data exports into Swift.
3. Instead of rolling your own highly available stateful services, use readily available HPE Helion OpenStack platform services such as Designate, the DNS service.

## What is not Highly Available?

Lifecycle Manager

The lifecycle manager in HPE Helion OpenStack is not highly-available. The lifecycle manager state/data are all maintained in a filesystem and are backed up by the Freezer service. In case of lifecycle manager failure, the state/data can be recovered from the backup.

Control Plane

High availability is not supported for Network Services (LBaaS, VPNaaS, FWaaS)

Nova-consoleauth

Nova-consoleauth is a singleton service, it can only run on a single node at a time. While nova-consoleauth is not high availability,

Cinder Volume and Backup Services

some work has been done to provide the ability to switch nova-consoleauth to another controller node in case of a failure. More Information on troubleshooting Nova-consoleauth can be found in the .

Keystone Cron Jobs

Cinder Volume and Backup Services are not high availability and started on one controller node at a time. More information on Cinder Volume and Backup Services can be found in

The Keystone cron job is a singleton service, which can only run on a single node at a time. A manual setup process for this job will be required in case of a node failure. More information on enabling the cron job for Keystone on the other nodes can be found in .

## More Information

- OpenStack High-availability Guide [<http://docs.openstack.org/high-availability-guide/content/ch-intro.html>]
- 12-Factor Apps [<http://12factor.net/>]

---

# Chapter 5. Third Party Integrations

We have the following documentation showing how to integrate HPE Helion OpenStack 5.0 with third party solutions.

## **General Integrations**

- HPE Helion OpenStack 5.0 supports the integration of 3rd-party components with a HPE Helion OpenStack platform deployment, whether that is a completely separate service or a plugin driver to an existing service in the HPE Helion OpenStack stack. The 3rd-party mechanism supports the integration of a range of different types of content.

## **Logging Service**

- This documentation demonstrates the possible integration between the HPE Helion OpenStack 5.0 centralized logging solution and Splunk including the steps to setup and forward logs.

---

## **Part II. Helion Lifecycle Manager Overview**

---

# Table of Contents

6. Input Model .....	38
Introduction to the Input Model .....	38
New in HPE Helion OpenStack 5.0 .....	38
Concepts .....	38
Cloud .....	40
Control Planes .....	40
Services .....	42
Server Roles .....	42
Disk Model .....	42
Memory Model .....	43
CPU Model .....	44
Servers .....	44
Server Groups .....	45
Networking .....	46
Configuration Data .....	49
7. Configuration Objects .....	50
Cloud Configuration .....	50
Control Plane .....	51
Clusters .....	53
Resources .....	54
Multiple Control Planes .....	55
Load Balancer Definitions in Control Planes .....	56
Load Balancers .....	56
Regions .....	57
Servers .....	59
Server Groups .....	61
Server Roles .....	62
Disk Models .....	63
Volume Groups .....	64
Device Groups .....	66
Disk Sizing for Virtual Machine Servers .....	66
Memory Models .....	67
Huge Pages .....	68
Memory Sizing for Virtual Machine Servers .....	68
CPU Models .....	68
CPU Assignments .....	69
CPU Usage .....	69
Components and Roles in the CPU Model .....	70
CPU sizing for virtual machine servers .....	70
Interface Models .....	70
network-interfaces .....	72
Bonding .....	73
fcoe-interfaces .....	75
dpdk-devices .....	76
NIC Mappings .....	77
NIC Mappings for Virtual Machine Servers .....	78
Network Groups .....	79
Load Balancer Definitions in Network Groups .....	82
Network Tags .....	82
MTU (Maximum Transmission Unit) .....	85
Networks .....	85

Firewall Rules .....	87
Rule .....	88
Configuration Data .....	88
Neutron network-tags .....	90
Neutron Configuration Data .....	90
Octavia Configuration Data .....	92
Ironic Configuration Data .....	92
Swift Configuration Data .....	93
Pass Through .....	94
8. Other Topics .....	95
Services and Service Components .....	95
Name Generation .....	97
Persisted Data .....	98
Persisted Server Allocations .....	99
Persisted Address Allocations .....	100
Server Allocation .....	101
Server Network Selection .....	101
Network Route Validation .....	101
Configuring Neutron Provider VLANs .....	105
Standalone Lifecycle Manager .....	106
9. Configuration Processor Information Files .....	108
address_info.yml .....	109
firewall_info.yml .....	110
net_info.yml .....	111
route_info.yml .....	111
server_info.yml .....	112
service_info.yml .....	113
control_plane_topology.yml .....	113
network_topology.yml .....	115
region_topology.yml .....	116
service_topology.yml .....	117
private_data_metadata.yml .....	117
password_change.yml .....	119
explain.txt .....	119
CloudDiagram.txt .....	120
HTML Representation .....	124
10. Example Configurations .....	125
HPE Helion OpenStack Example Configurations .....	125
Modifying the Entry-scale KVM with VSA Model for Your Environment .....	126
Alternative Configurations .....	126
KVM Examples .....	126
Entry-scale KVM with VSA Model .....	126
Entry-scale KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging .....	129
Entry-scale KVM with Ceph Model .....	130
Mid-scale KVM with VSA Model .....	147
ESX Examples .....	149
Entry-scale ESX, KVM with VSA Model .....	149
Entry-scale ESX, KVM with VSA Model with Dedicated Cluster for Metering, Monitoring, and Logging .....	151
Swift Examples .....	154
Entry-scale Swift Model .....	154
Ironic Examples .....	159
Entry-scale Cloud with Ironic Flat Network .....	159

Entry-scale Cloud with Ironic Multi-Tenancy .....	161
11. Modifying the Entry-scale KVM with VSA Model for Your Environment .....	163
Localizing the Input Model .....	163
Customizing the Input Model .....	170
disks_controller.yml .....	170
File Systems Storage .....	170
Swift Storage .....	171
disks_vsa.yml .....	171
disks_compute.yml .....	172
VSA with or without Adaptive Optimization (AO) .....	172
Creating Multiple VSA Clusters .....	173
Cloud Configuration Changes to Create More Than One Cluster .....	173
Cloud Configuration Changes to Create Two Cluster with Different Set of Disks .....	176
Configuring a Separate iSCSI Network to use with VSA .....	177
12. Modifying Example Configurations for Object Storage using Swift .....	180
Object Storage using Swift Overview .....	180
What is the Object Storage (Swift) Service? .....	180
Object Storage (Swift) Services .....	180
Allocating Proxy, Account, and Container (PAC) Servers for Object Storage .....	181
To allocate Swift PAC servers .....	181
Allocating Object Servers .....	181
To Allocate a Swift Object Server .....	182
Creating Roles for Swift Nodes .....	182
Allocating Disk Drives for Object Storage .....	183
Making Changes to a Swift Disk Model .....	183
Swift Requirements for Device Group Drives .....	186
Creating a Swift Proxy, Account, and Container (PAC) Cluster .....	186
Steps to Create a Swift Proxy, Account, and Container (PAC) Cluster .....	186
Service Components .....	187
Creating Object Server Resource Nodes .....	187
Understanding Swift Network and Service Requirements .....	188
Understanding Swift Ring Specifications .....	189
Ring Specifications in the Input Model .....	189
Replication Ring Parameters .....	190
Erasure Coded Rings .....	191
Selecting a Partition Power .....	192
Designing Storage Policies .....	194
Specifying Storage Policies .....	195
Designing Swift Zones .....	196
Using Server Groups to Specify Swift Zones .....	197
Specifying Swift Zones at Ring Level .....	198
Customizing Swift Service Configuration Files .....	199
Configuring Swift Container Rate Limit .....	199
Configuring Swift Account Server Logging Level .....	200
13. Alternative Configurations .....	202
SLES Compute Nodes .....	202
Entry-scale KVM with Ceph Model .....	204
Entry-scale KVM with Ceph Model with Two Networks .....	204
Using a Dedicated Lifecycle Manager Node .....	208
Specifying a dedicated lifecycle manager in your input model .....	209
Configuring HPE Helion OpenStack without DVR .....	212
Configuring HPE Helion OpenStack with Provider VLANs and Physical Routers Only .....	213
Considerations When Installing Two Systems on One Subnet .....	214

---

# Chapter 6. Input Model

## Introduction to the Input Model

This document describes how the HPE Helion OpenStack input model can be used to define and configure the cloud.

HPE Helion OpenStack ships with a set of example input models that can be used as starting points for defining a custom cloud.

The input model allows you, the cloud administrator, to describe the cloud configuration in terms of:

- Which OpenStack services run on which server nodes
- How individual servers are configured in terms of disk and network adapters
- The overall network configuration of the cloud
- Network traffic separation
- CIDR and VLAN assignments

The input model is consumed by the configuration processor which parses and validates the input model and outputs the effective configuration that will be deployed to each server that makes up your cloud.

The document is structured as follows:

- Concepts - This explains the ideas behind the declarative model approach used in HPE Helion OpenStack 5.0 and the core concepts used in describing that model
- Input Model - This section provides a description of each of the configuration entities in the input model
- Core Examples - In this section we provide samples and definitions of some of the more important configuration entities

## New in HPE Helion OpenStack 5.0

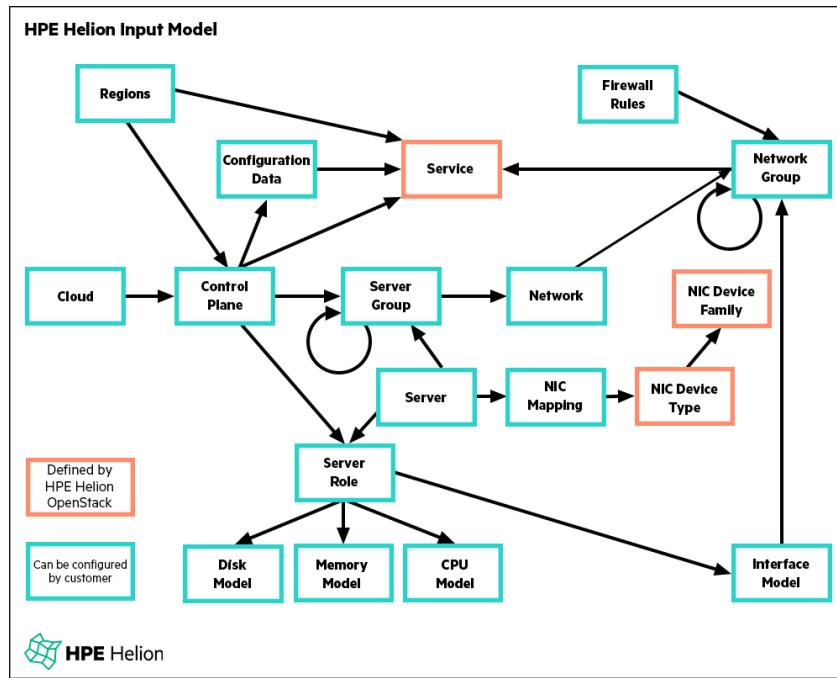
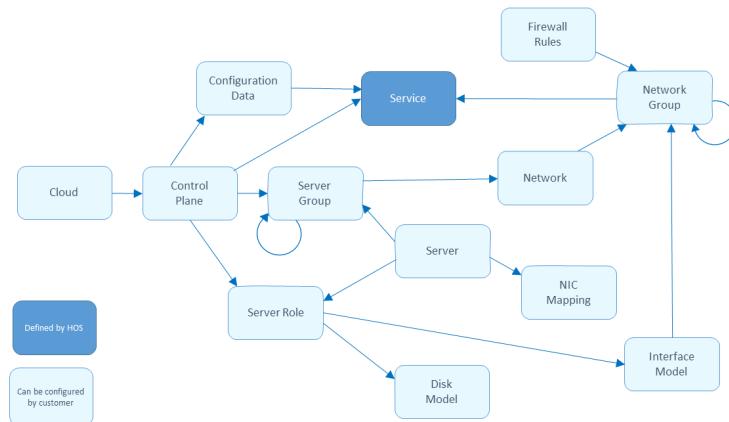
HPE Helion OpenStack 5.0 introduces the following additions to the cloud model:

- Container as a Service (Magnum) Support has been included.
- neutron-ml2-qos service addition.

## Concepts

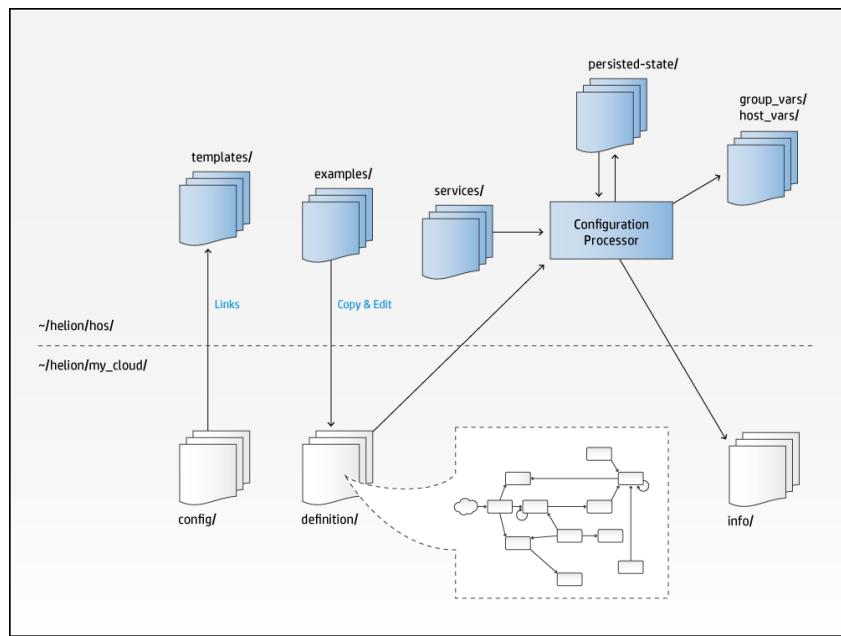
An HPE Helion OpenStack 5.0 cloud is defined by a declarative model that is described in a series of configuration objects. These configuration objects are represented in YAML files which together constitute the various example configurations provided as templates with this release. These examples can be used nearly unchanged, with the exception of necessary changes to IP addresses and other site and hardware-specific identifiers. Alternatively, the examples may be customized to meet site requirements.

The following diagram shows the set of configuration objects and their relationships. All objects have a name that you may set to be something meaningful for your context. In the examples these names are provided in capital letters as a convention. These names have no significance to HPE Helion OpenStack, rather it is the relationships between them that define the configuration.



The configuration processor reads and validates the input model described in the YAML files discussed above, combines it with the service definitions provided by HPE Helion OpenStack and any persisted state information about the current deployment to produce a set of Ansible variables that can be used to deploy the cloud. It also produces a set of information files that provide details about the configuration.

The relationship between the file systems on the HPE Helion OpenStack deployment server and the configuration processor is shown in the following diagram. Below the line are the directories that you, the cloud administrator, interact with. Above the line are the directories that are maintained by HPE Helion OpenStack.



Download a high-res version [[..../..../media/inputmodel/hphelionopenstack\\_directories\\_lg.png](#)]

The input model is read from the `~/helion/my_cloud/definition` directory. Although the supplied examples use separate files for each type of object in the model, the names and layout of the files have no significance to the configuration processor, it simply reads all of the .yml files in this directory. Cloud administrators are therefore free to use whatever structure is best for their context. For example, you may decide to maintain separate files or sub-directories for each physical rack of servers.

As mentioned, the examples use the conventional upper casing for object names, but these strings are used only to define the relationship between objects. They have no specific significance to the configuration processor.

## Cloud

The Cloud definition includes a few top-level configuration values such as the name of the cloud, the host prefix, details of external services (NTP, DNS, SMTP) and the firewall settings.

The location of the cloud configuration file also tells the configuration processor where to look for the files that define all of the other objects in the input model.

## Control Planes

*A control-plane runs one or more services distributed across clusters and resource groups.*

*A control-plane uses servers with a particular server-role.*

A control-plane provides the operating environment for a set of services; normally consisting of a set of shared services (MySQL, RabbitMQ, HA Proxy, Apache, etc), OpenStack control services (API, schedulers, etc) and the resources they are managing (compute, storage, etc).

A simple cloud may have a single control-plane which runs all of the services. A more complex cloud may have multiple control-planes to allow for more than one instance of some services. (Note that support for multiple control-planes is a non-core feature in HPE Helion OpenStack 5.0 and not covered by the examples). Services that need to consume (use) another service (such as Neutron consuming MySQL,

Nova consuming Neutron) always use the service within the same control-plane. In addition a control-plane can describe which services can be consumed from other control-planes. It is one of the functions of the configuration processor to resolve these relationships and make sure that each consumer/service is provided with the configuration details to connect to the appropriate provider/service.

Each control-plane is structured as clusters and resources. The clusters are typically used to host the OpenStack services that manage the cloud such as API servers, database servers, Neutron agents, and Swift proxies, while the resources are used to host the scale-out OpenStack services such as Nova-Compute or Swift-Object services. This is a representation convenience rather than a strict rule, for example it is possible to run the Swift-Object service in the management cluster in a smaller-scale cloud that is not designed for scale-out object serving.

A cluster can contain one or more servers and you can have one or more clusters depending on the capacity and scalability needs of the cloud that you are building. Spreading services across multiple clusters provides greater scalability, but it requires a greater number of physical servers. A common pattern for a large cloud is to run high data volume services such as monitoring and logging in a separate cluster. A cloud with a high object storage requirement will typically also run the Swift service in its own cluster.

Clusters in this context are a mechanism for grouping service components in physical servers, but all instances of a component in a control-plane work collectively. For example, if HA Proxy is configured to run on multiple clusters within the same control-plane then all of those instances will work as a single instance of the ha-proxy service.

Both clusters and resources define the type (via a list of server-roles) and number of servers (min and max or count) they require.

The control-plane can also define a list of failure-zones (server-groups) from which to allocate servers.

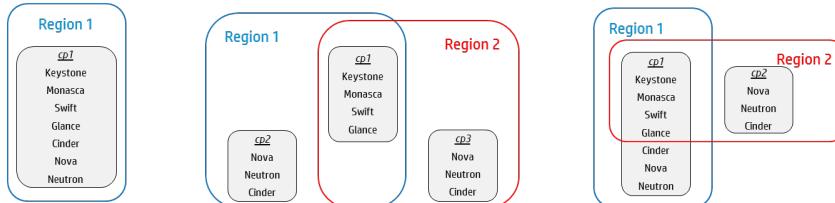
## Control Planes and Regions

A region in OpenStack terms is a collection of URLs that together provide a consistent set of services (Nova, Neutron, Swift, etc). Regions are represented in the Keystone identity service catalog and clients can decide which region they want to use.

For the owner of a cloud, regions provide a way of segmenting resources for scale, resilience, and isolation.

Regions don't have to be disjointed; for example, you can have a Swift service shared across more than one region, in which case the Swift URL for both regions will be the same and any region-specific services will use the same Swift instance. However, not all services can be shared in this way. For example, a Neutron service cannot be used by more than one Nova and so these will generally be deployed as region specific services, provided from separate control-planes. Equally in HPE Helion OpenStack, Mysql and RabbitMQ cannot be shared by more than one instance of the same service (for example a single mysql cluster cannot be used by two different instances of Nova, and so these are also deployed on a per-control basis).

In the input model, each region is defined as a set of services drawn from one or more control-planes. All of the following are valid mappings of control-planes to regions:



In a simple single control-plane cloud, there is no need for a separate region definition and the control-plane itself can define the region name.

## Services

*A control-plane runs one or more services.*

A service is the collection of service-components that provide a particular feature; for example, Nova provides the compute service and consists of the following service-components: nova-api, nova-scheduler, nova-conductor, nova-novncproxy, and nova-compute. Some services, like the authentication/identity service Keystone, only consist of a single service-component.

To define your cloud, all you need to know about a service are the names of the service-components. The details of the services themselves and how they interact with each other is captured in service definition files provided by HPE Helion OpenStack.

When specifying your HPE Helion OpenStack cloud you have to decide where components will run and how they connect to the networks. For example, should they all run in one control-plane sharing common services or be distributed across multiple control-planes to provide separate instances of some services? The HPE Helion OpenStack supplied examples provide solutions for some typical configurations.

Where services run is defined in the control-plane. How they connect to networks is defined in the network-groups.

## Server Roles

*Clusters and resources use servers with a particular set of server-roles.*

You're going to be running the services on physical servers, and you're going to need a way to specify which type of servers you want to use where. This is defined via the server-role. Each server-role describes how to configure the physical aspects of a server to fulfill the needs of a particular role. You'll generally use a different role whenever the servers are physically different (have different disks or network interfaces) or if you want to use some specific servers in a particular role (for example to choose which of a set of identical servers are to be used in the control plane).

Each server-role has a relationship to four other entities - the disk-model, the interface-model, the memory-model and the cpu-model:

- The **disk-model** specifies how to configure and use a server's local storage and it specifies disk sizing information for virtual machine servers. The disk model is described in the next section.
- The **interface-model** describes how a server's network interfaces are to be configured and used. This is covered in more details in the networking section.
- An optional **memory-model** specifies how to configure and use huge pages. The memory-model specifies memory sizing information for virtual machine servers.
- An optional **cpu-model** specifies how the CPUs will be used by Nova and by DPDK. The cpu-model specifies CPU sizing information for virtual machine servers.

## Disk Model

*Each physical disk device is associated with a device-group or a volume-group.*

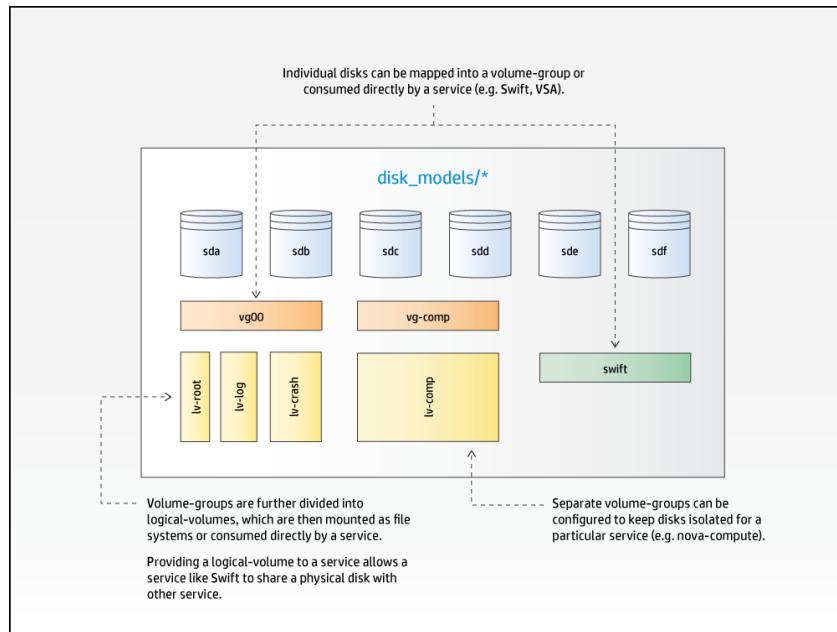
*Device-groups are consumed by services.*

*Volume-groups are divided into logical-volumes.*

*Logical-volumes are mounted as file systems or consumed by services.*

Disk-models define how local storage is to be configured and presented to services. Disk-models are identified by a name, which you will specify. The HPE Helion OpenStack examples provide some typical configurations. As this is an area that varies with respect to the services that are hosted on a server and the number of disks available, it is impossible to cover all possible permutations you may need to express via modifications to the examples.

Within a disk-model, disk devices are assigned to either a device-group or a volume-group.



Download a high-res version [[..../..../media/inputmodel/hphelionopenstack\\_diskmodels\\_lg.png](#)]

A device-group is a set of one or more disks that are to be consumed directly by a service. For example, a set of disks to be used by Swift. The device-group identifies the list of disk devices, the service, and a few service-specific attributes that tell the service about the intended use (for example, in the case of Swift this is the ring names). When a device is assigned to a device-group, the associated service is responsible for the management of the disks. This management includes the creation and mounting of file systems. (Swift can provide additional data integrity when it has full control over the file systems and mount points.)

A volume-group is used to present disk devices in a LVM volume group. It also contains details of the logical volumes to be created including the file system type and mount point. Logical volume sizes are expressed as a percentage of the total capacity of the volume group. A logical-volume can also be consumed by a service in the same way as a device-group. This allows services to manage their own devices on configurations that have limited numbers of disk drives.

Disk models also provide disk sizing information for virtual machine servers.

## Memory Model

Memory models define how the memory of a server should be configured to meet the needs of a particular role. It allows a number of HugePages to be defined at both the server and numa-node level.

Memory models also provide memory sizing information for virtual machine servers.

Memory models are optional - it is valid to have a server role without a memory model.

## CPU Model

CPU models define how CPUs of a server will be used. The model allows CPUs to be assigned for use by components such as Nova (for VMs) and Open vSwitch (for DPDK). It also allows those CPUs to be isolated from the general kernel SMP balancing and scheduling algorithms.

CPU models also provide CPU sizing information for virtual machine servers.

CPU models are optional - it is valid to have a server role without a cpu model.

## Servers

*Servers have a server-role which determines how they will be used in the cloud.*

Servers (in the input model) enumerate the resources available for your cloud. In addition, in this definition file you can either provide HPE Helion OpenStack with all of the details it needs to PXE boot and install an operating system onto the server, or, if you prefer to use your own operating system installation tooling you can simply provide the details needed to be able to SSH into the servers and start the deployment.

The address specified for the server will be the one used by HPE Helion OpenStack for lifecycle management and must be part of a network which is in the input model. If you are using HPE Helion OpenStack to install the operating system this network must be an untagged VLAN. The first server must be installed manually from the HPE Helion OpenStack ISO and this server must be included in the input model as well.

In addition to the network details used to install or connect to the server, each server defines what its server-role is and to which server-group it belongs.

## Virtual Machines as Servers

Starting in HPE Helion OpenStack 5.0, servers can be configured as hypervisors which host virtual machines that are can be defined, instantiated and configured as servers for hosting HPE Helion OpenStack services. Both the hypervisors and the VMs are treated as “servers” within the input model with additional attributes to define the relationship.

To define a hypervisor in the input model:

- Set the attribute `hlm-hypervisor: true` for the server.
- Define how network groups that are needed by the VMs will be mapped to the hypervisor's interfaces by adding `passthrough-network-groups` to the interface model of the hypervisor.

To define a virtual machine server in the input model:

- Create a server object that includes the `hypervisor-id` attribute. This indicates that the server is a virtual machine, and also identifies the physical server that will host the VM.
- Specify virtual machine CPU sizing information in the server's CPU model.
- Specify virtual machine memory sizing information in the server's memory model.
- Specify virtual machine disk sizing information in the server's disk model.
- Specify the virtual machine's network devices in the server's nic-mappings object.

Given the above information, HPE Helion OpenStack will configure the hypervisors and create thea virtual machines running the HPE Linux operating system. Subsequent configuration of the virtual machine's

operating system, and installation and configuration of HPE Helion OpenStack services on that virtual machine server, use the same lifecycle management mechanisms as physical servers.

Note that a hypervisor can still be used to run HPE Helion OpenStack services, so for example a server can be both running VMs (that in turn are running API services, MySQL, etc) and also be running services that are better not virtualized such as Swift, VSA, etc.

## Server Groups

*A server is associated with a server-group.*

*A control-plane can use server-groups as failure zones for server allocation.*

*A server-group may be associated with a list of networks.*

*A server-group can contain other server-groups.*

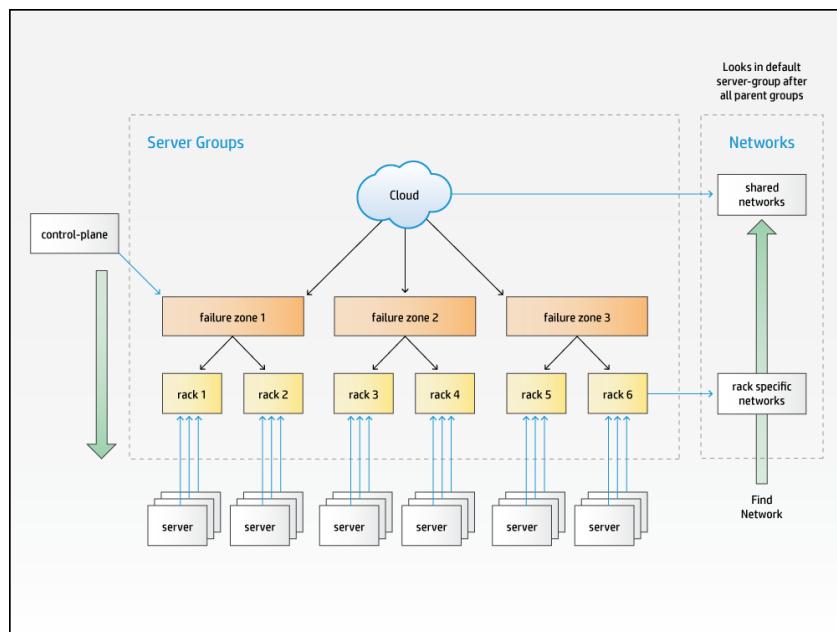
The practice of locating physical servers in a number of racks or enclosures in a data center is common. Such racks generally provide a degree of physical isolation that allows for separate power and/or network connectivity.

In the HPE Helion OpenStack model we support this configuration by allowing you to define a hierarchy of server-groups. Each server is associated with one server-group, normally at the bottom of the hierarchy.

Server-groups are an optional part of the input model - if you don't define any then all servers and networks will be allocated as if they are part of the same server-group.

## Server Groups and Failure Zones

A control-plane defines a list of server-groups as the failure zones from which it wants to use servers. All servers in a server-group listed as a failure zone in the control-plane and any server-groups they contain are considered part of that failure zone for allocation purposes. The following example shows how three levels of server-groups can be used to model a failure zone consisting of multiple racks, each of which in turn contains a number of servers.



Download a high-res version [[..../..../media/inputmodel/hphelionopenstack\\_servergroups\\_lg.png](#)]

When allocating servers, the configuration processor will traverse down the hierarchy of server-groups listed as failure zones until it can find an available server with the required server-role. If the allocation policy is defined to be strict, it will allocate servers equally across each of the failure zones. A cluster or resource-group can also independently specify the failure zones it wants to use if needed.

## Server Groups and Networks

Each L3 network in a cloud must be associated with all or some of the servers, typically following a physical pattern (such as having separate networks for each rack or set of racks). This is also represented in the HPE Helion OpenStack model via server-groups, each group lists zero or more networks to which servers associated with server-groups at or below this point in the hierarchy are connected.

When the configuration processor needs to resolve the specific network a server should be configured to use, it traverses up the hierarchy of server-groups, starting with the group the server is directly associated with, until it finds a server-group that lists a network in the required network group.

The level in the server-group hierarchy at which a network is associated will depend on the span of connectivity it must provide. In the above example there might be networks in some network-groups which are per rack (i.e. Rack 1 and Rack 2 list different networks from the same network-group) and networks in a different network-group that span failure zones (the network used to provide floating IP addresses to virtual machines for example).

## Networking

In addition to the mapping of services to specific clusters and resources we must also be able to define how the services connect to one or more networks.

In a simple cloud there may be a single L3 network but more typically there are functional and physical layers of network separation that need to be expressed.

Functional network separation provides different networks for different types of traffic; for example, it is common practice in even small clouds to separate the External APIs that users will use to access the cloud and the external IP addresses that users will use to access their virtual machines. In more complex clouds it's common to also separate out virtual networking between virtual machines, block storage traffic, and volume traffic onto their own sets of networks. In the input model, this level of separation is represented by network-groups.

Physical separation is required when there are separate L3 network segments providing the same type of traffic; for example, where each rack uses a different subnet. This level of separation is represented in the input model by the networks within each network-group.

## Network Groups

*Service endpoints attach to networks in a specific network-group.*

*Network-groups can define routes to other networks.*

*Network-groups encapsulate the configuration for services via network-tags*

A network-group defines the traffic separation model and all of the properties that are common to the set of L3 networks that carry each type of traffic. They define where services are attached to the network model and the routing within that model.

In terms of service connectivity, all that has to be captured in the network-groups definition is the same service-component names that are used when defining control-planes. HPE Helion OpenStack also allows a default attachment to be used to specify "all service-components" that aren't explicitly connected to an-

other network-group. So, for example, to isolate Swift traffic, the swift-account, swift-container, and swift-object service components are attached to an "Object" network-group and all other services are connected to "Management" network-group via the default relationship.

The details of how each service connects, such as what port it uses, if it should be behind a load balancer, if and how it should be registered in Keystone, and so forth, are defined in the service definition files provided by HPE Helion OpenStack.

In any configuration with multiple networks, controlling the routing is a major consideration. In HPE Helion OpenStack, routing is controlled at the network-group level. First, all networks are configured to provide the route to any other networks in the same network-group. In addition, a network-group may be configured to provide the route any other networks in the same network-group; for example, if the internal APIs are in a dedicated network-group (a common configuration in a complex network because a network group with load balancers cannot be segmented) then other network-groups may need to include a route to the internal API network-group so that services can access the internal API endpoints. Routes may also be required to define how to access an external storage network or to define a general default route.

As part of the HPE Helion OpenStack deployment, networks are configured to act as the default route for all traffic that was received via that network (so that response packets always return via the network the request came from).

Note that HPE Helion OpenStack will configure the routing rules on the servers it deploys and will validate that the routes between services exist in the model, but ensuring that gateways can provide the required routes is the responsibility of your network configuration. The configuration processor provides information about the routes it is expecting to be configured.

For a detailed description of how the configuration processor validates routes, refer to the section called "Network Route Validation".

## Load Balancers

Load-balancers provide a specific type of routing and are defined as a relationship between the virtual IP address (VIP) on a network in one network group and a set of service endpoints (which may be on networks in the same or a different network-group).

As each load-balancer is defined providing a virtual IP on a network-group, it follows that those network-groups can each only have one network associated to them.

The load-balancer definition includes a list of service-components and endpoint roles it will provide a virtual IP for. This model allows service-specific load-balancers to be defined on different network-groups. A "default" value is used to express "all service-components" which require a virtual IP address and are not explicitly configured in another load-balancer configuration. The details of how the load-balancer should be configured for each service, such as which ports to use, how to check for service liveness, etc, are provided in the HPE Helion OpenStack supplied service definition files.

Where there are multiple instances of a service (i.e in a cloud with multiple control-planes), each control-plane needs its own set of virtual IP address and different values for some properties such as the external name and security certificate. To accommodate this in HPE Helion OpenStack 5.0, load-balancers are defined as part of the control-plane, with the network groups defining just which load-balancers are attached to them.

Load balancers are always implemented by an ha-proxy service in the same control-plane as the services.

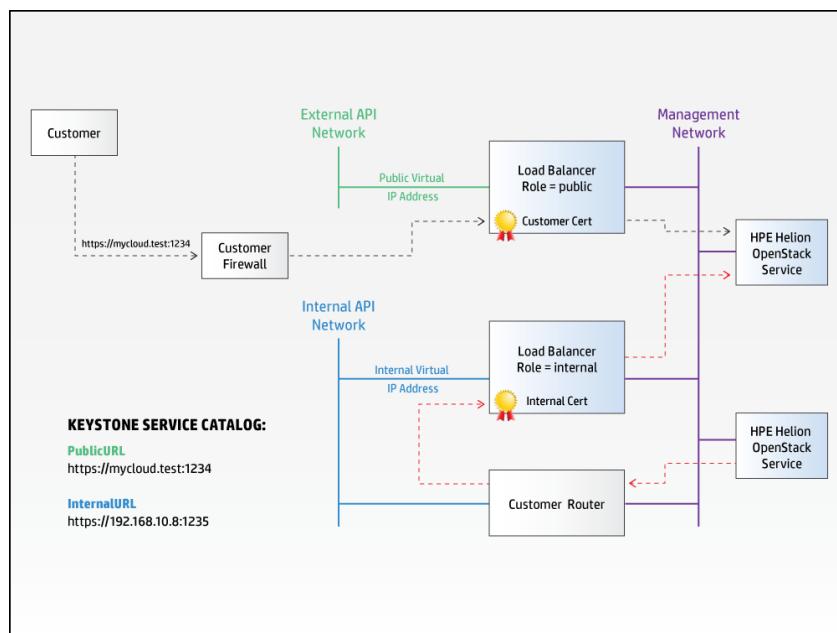
## Separation of Public, Admin, and Internal Endpoints

The list of endpoint roles for a load-balancer make it possible to configure separate load-balancers for public and internal access to services, and the configuration processor uses this information to both ensure

the correct registrations in Keystone and to make sure the internal traffic is routed to the correct endpoint. HPE Helion OpenStack services are configured to only connect to other services via internal virtual IP addresses and endpoints, allowing the name and security certificate of public endpoints to be controlled by the customer and set to values that may not be resolvable/accessible from the servers making up the cloud.

Note that each load-balancer defined in the input model will be allocated a separate virtual IP address even when the load-balancers are part of the same network-group. Because of the need to be able to separate both public and internal access, HPE Helion OpenStack will not allow a single load-balancer to provide both public and internal access. Load-balancers in this context are logical entities (sets of rules to transfer traffic from a virtual IP address to one or more endpoints).

The following diagram shows a possible configuration in which the hostname associated with the public URL has been configured to resolve to a firewall controlling external access to the cloud. Within the cloud, HPE Helion OpenStack services are configured to use the internal URL to access a separate virtual IP address.



Download a high-res version [../../../../media/inputmodel/hphelionopenstack\_loadbalancers\_lg.png]

## Network Tags

Network tags are defined by some HPE Helion OpenStack service-components and are used to convey information between the network model and the service, allowing the dependent aspects of the service to be automatically configured.

Network tags also convey requirements a service may have for aspects of the server network configuration, for example, that a bridge is required on the corresponding network device on a server where that service-component is installed.

See the section called “Network Tags” for more information on specific tags and their usage.

## Networks

*A network is part of a network-group.*

Networks are fairly simple definitions. Each network defines the details of its VLAN, optional address details (CIDR, start and end address, gateway address), and which network-group it is a member of.

## Interface Model

A *server-role* identifies an *interface-model* that describes how its network interfaces are to be configured and used.

Network groups are mapped onto specific network interfaces via an interface-model, which describes the network devices that need to be created (bonds, ovs-bridges, etc) and their properties.

An interface-model acts like a template; it can define how some or all of the network-groups are to be mapped for a particular combination of physical NICs. However, it is the service-components on each server that determine which network-groups are required and hence which interfaces and networks will be configured. This means that interface-models can be shared between different server-roles. For example, an API role and a database role may share an interface model even though they may have different disk models and they will require a different subset of the network-groups.

Within an interface-model, physical ports are identified by a device name, which in turn is resolved to a physical port on a server basis via a nic-mapping. To allow different physical servers to share an interface-model, the nic-mapping is defined as a property of each server.

The *interface-model* can also be used to describe how network devices are to be configured for use with DPDK, SR-IOV, and PCI Passthrough.

## NIC Mapping

When a server has more than a single physical network port, a nic-mapping is required to unambiguously identify each port. Standard Linux mapping of ports to interface names at the time of initial discovery (e.g. eth0, eth1, eth2, ...) is not uniformly consistent from server to server, so a mapping of PCI bus address to interface name is instead.

NIC mappings are also used to specify the device type for interfaces that are to be used for SR-IOV or PCI Passthrough. Each HPE Helion OpenStack release includes the data for the supported device types.

## Firewall Configuration

The configuration processor uses the details it has about which networks and ports service-components use to create a set of firewall rules for each server. The model allows additional user-defined rules on a per network-group basis.

## Configuration Data

Configuration Data is used to provide settings which have to be applied in a specific context, or where the data needs to be verified against or merged with other values in the input model.

For example, when defining a Neutron provider network to be used by Octavia, the network needs to be included in the routing configuration generated by the Configuration Processor.

---

# Chapter 7. Configuration Objects

## Cloud Configuration

The top-level cloud configuration file, `cloudConfig.yml`, defines some global values for the HPE Helion OpenStack Cloud, as described in the table below.

The snippet below shows the start of the control plane definition file.

```
---
product:
  version: 2

cloud:
  name: entry-scale-kvm-vsa

  hostname-data:
    host-prefix: helion
    member-prefix: -m

  ntp-servers:
    - "ntp-server1"

  # dns resolving configuration for your site
  dns-settings:
    nameservers:
      - name-server1

  firewall-settings:
    enable: true
    # log dropped packets
    logging: true

  audit-settings:
    audit-dir: /var/audit
    default: disabled
    enabled-services:
      - keystone
```

Key	Value Description
name	An administrator-defined name for the cloud
hostname-data (optional)	Provides control over some parts of the generated names (see ) Consists of two values: <ul style="list-style-type: none"><li>• host-prefix - default is to use the cloud name (above)</li><li>• member-prefix - default is "-m"</li></ul>
ntp-servers (optional)	A list of external NTP servers your cloud has access to. If specified by name then the names need to be resolvable via the external DNS nameservers you specify in

Key	Value Description
	the next section. All servers running the "ntp-server" component will be configured to use these external NTP servers.
dns-settings (optional)	DNS configuration data that will be applied to all servers. See example configuration for a full list of values.
smtp-settings (optional)	SMTP client configuration data that will be applied to all servers. See example configurations for a full list of values.
firewall-settings (optional)	Used to enable/disable the firewall feature and to enable/disable logging of dropped packets.  The default is to have the firewall enabled.
audit-settings (optional)	Used to enable/disable the production of audit data from services.  The default is to have audit disabled for all services.

## Control Plane

The snippet below shows the start of the control plane definition file.

```
---  
product:  
  version: 2  
  
control-planes:  
  - name: control-plane-1  
    control-plane-prefix: cp1  
    region-name: region1  
    failure-zones:  
      - AZ1  
      - AZ2  
      - AZ3  
    configuration-data:  
      - NEUTRON-CONFIG-CP1  
      - OCTAVIA-CONFIG-CP1  
    common-service-components:  
      - logging-producer  
      - monasca-agent  
      - freezer-agent  
      - stunnel  
      - lifecycle-manager-target  
    clusters:  
      - name: cluster1  
        cluster-prefix: cl  
        server-role: CONTROLLER-ROLE  
        member-count: 3  
        allocation-policy: strict  
        service-components:  
          - lifecycle-manager  
          - ntp-server  
          - swift-ring-builder  
          - mysql  
          - ip-cluster
```

```
...
resources:
  - name: compute
    resource-prefix: comp
    server-role: COMPUTE-ROLE
    allocation-policy: any
    min-count: 0
    service-components:
      - ntp-client
      - nova-compute
      - nova-compute-kvm
      - neutron-l3-agent
...

```

Key	Value Description
name	This name identifies the control plane. This value is used to persist server allocations (see ) and cannot be changed once servers have been allocated.
control-plane-prefix (optional)	The control-plane-prefix is used as part of the host-name (see ). If not specified, the control plane name is used.
region-name	This name identifies the Keystone region within which services in the control plane will be registered.  For clouds consisting of multiple control planes, this attribute should be omitted and the regions object should be used to set the region name (see Regions).
uses (optional)	Identifies the services this control will consume from other control planes (see Multiple control planes).
load-balancers (optional)	A list of load balancer definitions for this control plane (see Load balancer definitions in control planes).  For a multi control-plane cloud load balancers must be defined in each control-plane. For a single control-plane cloud they may be defined either in the control plane or as part of a network group (compatibility with HPE Helion OpenStack 3.0)
common-service-components (optional)	This lists a set of service components that run on all servers in the control plane (clusters and resource pools)
failure-zones (optional)	A list of server-group names that servers for this control plane will be allocated from. If no failure-zones are specified, only servers not associated with a server-group will be used. (see the section called “Server Groups and Failure Zones” for a description of server-groups as failure zones.)

<b>Key</b>	<b>Value Description</b>
configuration-data (optional)	A list of configuration data settings to be used for services in this control plane (see the section called “Configuration Data”)
clusters	A list of clusters for this control plane (see the section called “Clusters”).
resources	A list of resource groups for this control plane (see the section called “Resources”).

## Clusters

<b>Key</b>	<b>Value Description</b>
name	Cluster and resource names must be unique within a control plane. This value is used to persist server allocations (see the section called “Persisted Data”) and cannot be changed once servers have been allocated.
cluster-prefix (optional)	The cluster prefix is used in the hostname (see the section called “Name Generation”). If not supplied then the cluster name is used.
server-role	This can either be a string (for a single role) or a list of roles. Only servers matching one of the specified server-roles will be allocated to this cluster. (see the section called “Server Roles” for a description of server roles)
service-components	The list of service-components to be deployed on the servers allocated for the cluster. (The common-service-components for the control plane are also deployed.)
member-count min-count max-count (all optional)	<p>Defines the number of servers to add to the cluster.</p> <p>The number of servers that can be supported in a cluster depends on the services it is running. For example MySQL and RabbitMQ can only be deployed on clusters on 1 (non-HA) or 3 (HA) servers. Other services may support different sizes of cluster.</p> <p>If min-count is specified, then at least that number of servers will be allocated to the cluster. If min-count is not specified it defaults to a value of 1.</p> <p>If max-count is specified, then the cluster will be limited to that number of servers. If max-count is not specified then all servers matching the required role and failure-zones will be allocated to the cluster.</p> <p>Specifying member-count is equivalent to specifying min-count and max-count with the same value.</p>
failure-zones (optional)	A list of server-groups that servers will be allocated from. If specified, it overrides the list of values

Key	Value Description
	specified for the control-plane. If not specified, the control-plane value is used. (see the section called “Server Groups and Failure Zones” for a description of server groups as failure zones).
allocation-policy (optional)	<p>Defines how failure zones will be used when allocating servers.</p> <p><b>strict:</b> Server allocations will be distributed across all specified failure zones. (if max-count is not a whole number, an exact multiple of the number of zones, then some zones may provide one more server than other zones)</p> <p><b>any:</b> Server allocations will be made from any combination of failure zones.</p> <p>The default allocation-policy for a cluster is <i>strict</i>.</p>
configuration-data (optional)	A list of configuration-data settings that will be applied to the services in this cluster. The values for each service will be combined with any values defined as part of the configuration-data list for the control-plane. If a value is specified by settings in both lists, the value defined here takes precedence.

## Resources

Key	Value Description
name	<p>The name of this group of resources. Cluster names and resource-node names must be unique within a control plane. Additionally, clusters and resources cannot share names within a control-plane.</p> <p>This value is used to persist server allocations (see the section called “Persisted Data”) and cannot be changed once servers have been allocated.</p>
resource-prefix	The resource-prefix is used in the name generation. (see the section called “Name Generation”)
server-role	This can either be a string (for a single role) or a list of roles. Only servers matching one of the specified server-roles will be allocated to this resource group. (see the section called “Server Roles” for a description of server roles).
service-components	The list of service-components to be deployed on the servers in this resource group. (The common-service-components for the control plane are also deployed.)
member-count	Defines the number of servers to add to the cluster.
min-count	The number of servers that can be supported in a cluster depends on the services it is running. For ex-

Key	Value Description
max-count (all optional)	<p>ample MySQL and RabbitMQ can only be deployed on clusters on 1 (non-HA) or 3 (HA) servers. Other services may support different sizes of cluster.</p> <p>If min-count is specified, then at least that number of servers will be allocated to the cluster. If min-count is not specified it defaults to a value of 1.</p> <p>If max-count is specified, then the cluster will be limited to that number of servers. If max-count is not specified then all servers matching the required role and failure-zones will be allocated to the cluster.</p> <p>Specifying member-count is equivalent to specifying min-count and max-count with the same value.</p>
failure-zones (optional)	<p>A list of server-groups that servers will be allocated from. If specified, it overrides the list of values specified for the control-plane. If not specified, the control-plane value is used. (see the section called “Server Groups and Failure Zones” for a description of server groups as failure zones).</p>
allocation-policy (optional)	<p>Defines how failure zones will be used when allocating servers.</p> <p><b>strict:</b> Server allocations will be distributed across all specified failure zones. (if max-count is not a whole number, an exact multiple of the number of zones, then some zones may provide one more server than other zones)</p> <p><b>any:</b> Server allocations will be made from any combination of failure zones.</p> <p>The default allocation-policy for resources is <i>any</i>.</p>
configuration-data (optional)	<p>A list of configuration-data settings that will be applied to the services in this cluster. The values for each service will be combined with any values defined as part of the configuration-data list for the control-plane. If a value is specified by settings in both lists, the value defined here takes precedence.</p>

## Multiple Control Planes

The dependencies between service components (e.g. Nova needs MySql and Keystone API) is defined as part of the service definitions provided by HPE Helion OpenStack, the control-planes define how those dependencies will be met. For clouds consisting of multiple control-planes, the relationship between services in different control planes is defined by a `uses` attribute in its control-plane object. Services will always use other services in the same control-plane before looking to see if the required service can be provided from another control-plane. For example, a service component in control-plane `cp-2` (e.g. `nova-api`) might use service components from control-plane `cp-shared` (e.g. `keystone-api`).

```
control-planes:  
  - name: cp-2  
    uses:  
      - from: cp-shared  
        service-components:  
          - any
```

Key	Value Description
from	The name of the control-plane providing services which may be consumed by this control-plane.
service-components	A list of service components from the specified control-plane which may be consumed by services in this control-plane. The reserved keyword any indicates that any service component from the specified control-plane may be consumed by services in this control-plane.

## Load Balancer Definitions in Control Planes

Starting in HPE Helion OpenStack 5.0, a load-balancer may be defined within a control-plane object, and referenced by name from a network-groups object. The following example shows load balancer extlb defined in control-plane cp1 and referenced from the EXTERNAL-API network group. See section Load balancers for a complete description of load balance attributes.

```
network-groups:  
  - name: EXTERNAL-API  
    load-balancers:  
      - extlb  
  
control-planes:  
  - name: cp1  
    load-balancers:  
      - provider: ip-cluster  
        name: extlb  
        external-name:  
        tls-components:  
          - default  
        roles:  
          - public  
        cert-file: cp1-extlb-cert
```

## Load Balancers

Load balancers may be defined as part of a network-group object, or as part of a control-plane object. When a load-balancer is defined in a control-plane, it must be referenced by name only from the associated network-group object.

For clouds consisting of multiple control planes, load balancers must be defined as part of a control-plane object. This allows different load balancer configurations for each control plane.

In either case, a load-balancer definition has the following attributes:

```
load-balancers:  
  - provider: ip-cluster  
    name: extlb  
    external-name:  
  
    tls-components:  
      - default  
    roles:  
      - public  
    cert-file: cp1-extlb-cert
```

Key	Value Description
name	An administrator defined name for the load balancer. This name is used to make the association from a network-group.
provider	The service component that implements the load balancer. Currently only ip-cluster (ha-proxy) is supported. Future releases will provide support for external load balancers.
roles	The list of endpoint roles that this load balancer provides (see below). Valid roles are public, internal, and admin. To ensure separation of concerns, the role public cannot be combined with any other role. See Load Balancers for an example of how the role provides endpoint separation.
components (optional)	The list of service-components for which the load balancer provides a non-encrypted virtual IP address for.
tls-components (optional)	The list of service-components for which the load balancer provides TLS-terminated virtual IP addresses for.
external-name (optional)	The name to be registered in Keystone for the public-CURL. If not specified, the virtual IP address will be registered. Note that this value cannot be changed after the initial deployment.
cert-file (optional)	The name of the certificate file to be used for tls endpoints. If not specified, a file name will be constructed using the format <cp-name>-<lb-name>-cert, where cp-name is the control-plane name and lb-name is the load-balancer name.

## Regions

The regions configuration object is used to define how a set of services from one or more control-planes are mapped into Openstack regions (entries within the Keystone catalog).

Within each region a given service is provided by one control plane, but the set of services in the region may be provided by multiple control planes.

A service in a given control plane may be included in more than one region.

```
---
product:
  version: 2

regions:
  - name: region1
    includes:
      - control-plane: control-plane-1
        services:
          - all

  - name: region2
    includes:
      - control-plane: control-plane-1
        services:
          - keystone
          - glance
          - swift
          - designate
          - ceilometer
          - barbican
          - horizon
          - monasca
          - freezer
          - logging
          - operations
      - control-plane: control-plane-2
        services:
          - cinder
          - nova
          - neutron
          - octavia
          - heat
```

Key	Value Description
name	The name of the region in the Keystone service catalog.
includes	A list of services to include in this region, broken down by the control planes providing the services.

Key	Value Description
control-plane	A control-plane name.
services	A list of service names. This list specifies the services from this control-plane to be included in this region. The reserved keyword <code>all</code> may be used when all services from the control-plane are to be included.

# Servers

The servers configuration object is used to list the available servers for deploying the cloud.

Optionally, it can be used as an input file to the operating system installation process, in which case some additional fields (identified below) will be necessary.

```
---  
product:  
    version: 2  
  
baremetal:  
    subnet: 192.168.10.0  
    netmask: 255.255.255.0  
  
servers:  
    - id: controller1  
        ip-addr: 192.168.10.3  
        role: CONTROLLER-ROLE  
        server-group: RACK1  
        nic-mapping: HP-DL360-4PORT  
        mac-addr: b2:72:8d:ac:7c:6f  
        ilo-ip: 192.168.9.3  
        ilo-password: password  
        ilo-user: admin  
  
    - id: controller2  
        ip-addr: 192.168.10.4  
        role: CONTROLLER-ROLE  
        server-group: RACK2  
        nic-mapping: HP-DL360-4PORT  
        mac-addr: 8a:8e:64:55:43:76  
        ilo-ip: 192.168.9.4  
        ilo-password: password  
        ilo-user: admin
```

Key	Value Description
id	An administrator-defined identifier for the server. IDs must be unique and are used to track server allocations. (see the section called “Persisted Data”).
ip-addr	<p>The IP address is used by the configuration processor to install and configure the service components on this server.</p> <p>This IP address must be within the range of a network defined in this model.</p> <p>When the servers file is being used for operating system installation, this IP address will be assigned to the node by the installation process, and the associated network must be an untagged VLAN.</p>
hostname (optional)	The value to use for the hostname of the server. If specified this will be used to set the hostname value

<b>Key</b>	<b>Value Description</b>
	of the server which will in turn be reflected in systems such as Nova, Monasca, etc. If not specified the hostname will be derived based on where the server is used and the network defined to provide hostnames.
role	Identifies the server-role of the server. (see for a description of server roles)
nic-mapping	Name of the nic-mappings entry to apply to this server. (See the section called “NIC Mappings”.)
server-group (optional)	Identifies the server-groups entry that this server belongs to. (see the section called “Server Groups”)
boot-from-san (optional)	Must be set to true if the server needs to be configured to boot from SAN storage. Default is False
fcoe-interfaces (optional)	A list of network devices that will be used for accessing FCoE storage. This is only needed for devices that present as native FCoE, not devices such as Emulex which present as a FC device.
ansible-options (optional)	A string of additional variables to be set when defining the server as a host in Ansible. For example, <code>ansible_ssh_port=5986</code>
mac-addr (optional)	Needed when the servers file is being used for operating system installation. This identifies the MAC address on the server that will be used to network install the operating system.
distro-id (optional)	<p>The name of the cobbler server profile to be used when the servers file is used for operating system installation. Supported values are:</p> <ul style="list-style-type: none"> <li>• <code>hlinux-x86_64</code> (default)</li> <li>• <code>rhel72-x86_64</code></li> <li>• <code>rhel72-x86_64-multipath</code></li> </ul> <p><b>Important</b></p> <p>RHEL is only supported for KVM compute hosts. Note that you need to add a <code>-multipath</code> suffix to the distro-id value when using multipath with RHEL.</p>
kopt-extras (optional)	Provides additional command line arguments to be passed to the booting network kernel. For example, <code>vga=769</code> sets the video mode for the install to low resolution which can be useful for remote console users.
ilo-ip (optional)	Needed when the servers file is being used for operating system installation. This provides the IP address of the power management (e.g. IPMI, iLO) subsystem.

Key	Value Description
ilo-user (optional)	Needed when the servers file is being used for operating system installation. This provides the user name of the power management (e.g. ipmi-ip, iLO) subsystem.
ilo-password (optional)	Needed when the servers file is being used for operating system installation. This provides the user password of the power management (e.g. ipmi-ip, iLO) subsystem.
ilo-extras (optional)	Needed when the servers file is being used for operating system installation. Additional options to pass to ipmitool. For example, this may be required if the servers require additional IPMI addressing parameters.
moonshot (optional)	Provides the node identifier for HPE Moonshot servers, e.g. c4n1 where c4 is the cartridge and n1 is node 1.
hypervisor-id (optional)	This attribute serves two purposes: it indicates that this server is a virtual machine (VM), and it specifies the server id of the HLM hypervisor that will host the VM.
hlm-hypervisor (optional)	When set to True, this attribute identifies a server as a HLM hypervisor. A HLM hypervisor is a server that may be used to host other servers that are themselves virtual machines. Default value is False.

## Server Groups

The server-groups configuration object provides a mechanism for organizing servers and networks into a hierarchy that can be used for allocation and network resolution (see ).

```
---
product:
  version: 2

  - name: CLOUD
    server-groups:
      - AZ1
      - AZ2
      - AZ3
    networks:
      - EXTERNAL-API-NET
      - EXTERNAL-VM-NET
      - GUEST-NET
      - MANAGEMENT-NET

    #
    # Create a group for each failure zone
    #
    - name: AZ1
      server-groups:
```

```
- RACK1

- name: AZ2
  server-groups:
    - RACK2

- name: AZ3
  server-groups:
    - RACK3

#
# Create a group for each rack
#
- name: RACK1
- name: RACK2
- name: RACK3
```

Key	Value Description
name	An administrator-defined name for the server group. The name is used to link server-groups together and to identify server-groups to be used as failure zones in a control-plane. (see the section called “Control Plane”)
server-groups (optional)	A list of server-group names that are nested below this group in the hierarchy. Each server group can only be listed in one other server group (i.e. in a strict tree topology).
networks (optional)	A list of network names (see the section called “Networks”). See the section called “Server Groups and Networks” for a description of how networks are matched to servers via server groups.

## Server Roles

The server-roles configuration object is a list of the various server roles that you can use in your cloud. Each server role is linked to other configuration objects:

- Disk model (see )
- Interface model (see )
- Memory model (see )
- CPU model (see )

Server roles are referenced in the servers (see ) configuration object above.

```
---
product:
  version: 2

server-roles:

  - name: CONTROLLER-ROLE
```

```
interface-model: CONTROLLER-INTERFACES
disk-model: CONTROLLER-DISKS

- name: COMPUTE-ROLE
  interface-model: COMPUTE-INTERFACES
  disk-model: COMPUTE-DISKS
  memory-model: COMPUTE-MEMORY
  cpu-model: COMPUTE-CPU

- name: VSA-ROLE
  interface-model: VSA-INTERFACES
  disk-model: VSA-DISKS
```

Key	Value Description
name	An administrator-defined name for the role.
interface-model	The name of the interface-model to be used for this server-role.  Different server-roles can use the same interface-model.
disk-model	The name of the disk-model to use for this server-role.  Different server-roles can use the same disk-model.
memory-model (optional)	The name of the memory-model to use for this server-role.  Different server-roles can use the same memory-model.
cpu-model (optional)	The name of the cpu-model to use for this server-role.  Different server-roles can use the same cpu-model.

## Disk Models

The disk-models configuration object is used to specify how the directly attached disks on the server should be configured. It can also identify which service or service component consumes the disk, e.g. Swift object server, and provide service-specific information associated with the disk. It is also used to specify disk sizing information for virtual machine servers.

Disk models can be used as raw devices or as logical volumes and the disk model provides a configuration item for each.

If the operating system has been installed by the HPE Helion OpenStack installation process then the root disk will already have been set up as a volume-group with a single logical-volume. This logical-volume will have been created on a partition identified, symbolically, in the configuration files as /dev/sda\_root. This is due to the fact that different BIOS systems (UEFI, Legacy) will result in different partition numbers on the root disk.

---

product:

```
version: 2

disk-models:
- name: VSA-DISKS

volume-groups:
- ...
device-groups:
- ...
vm-size:
...
```

Key	Value Description
name	The name of the disk-model that is referenced from one or more server-roles.
volume-groups	A list of volume-groups to be configured (see below). There must be at least one volume-group describing the root file system.
device-groups (optional)	A list of device-groups (see below)
vm-size (optional)	Disk sizing information for virtual machine servers (see below).

## Volume Groups

The volume-groups configuration object is used to define volume groups and their constituent logical volumes.

Note that volume-groups are not exact analogs of device-groups. A volume-group specifies a set of physical volumes used to make up a volume-group that is then subdivided into multiple logical volumes.

The HPE Helion OpenStack operating system installation automatically creates a volume-group name "hlm-vg" on the first drive in the system. It creates a "root" logical volume there. The volume-group can be expanded by adding more physical-volumes (see examples). In addition, it is possible to create more logical-volumes on this volume-group to provide dedicated capacity for different services or file system mounts.

```
volume-groups:
- name: hlm-vg
  physical-volumes:
    - /dev/sda_root

  logical-volumes:
    - name: root
      size: 35%
      fstype: ext4
      mount: /

    - name: log
      size: 50%
      mount: /var/log
      fstype: ext4
```

```

mkfs-opt: -O large_file

- ...

- name: vg-comp
  physical-volumes:
    - /dev/sdb
  logical-volumes:
    - name: compute
      size: 95%
      mount: /var/lib/nova
      fstype: ext4
      mkfs-opt: -O large_file

```

Key	Value Descriptions
name	The name that will be assigned to the volume-group
physical-volumes	<p>A list of physical disks that make up the volume group.</p> <p>As installed by the HPE Helion OpenStack operating system install process, the volume group "hlm-vg" will use a large partition (sda_root) on the first disk. This can be expanded by adding additional disk(s).</p>
	<p><b>Important</b></p> <p>Multipath storage should be listed as the corresponding /dev/mapper/mpath&lt;x&gt;</p>
logical-volumes	A list of logical volume devices to create from the above named volume group.
name	The name to assign to the logical volume.
size	The size, expressed as a percentage of the entire volume group capacity, to assign to the logical volume.
fstype (optional)	The file system type to create on the logical volume. If nonE specified, the volume is not formatted.
mkfs-opt (optional)	Options, e.g. -O large_file to pass to the mkfs command.
mode (optional)	The mode changes the root file system mode bits, which can be either a symbolic representation or an octal number representing the bit pattern for the new mode bits.
mount (optional)	Mount point for the file system.
consumer attributes (optional, consumer dependent)	<p>These will vary according to the service consuming the device group. The examples section provides sample content for the different services.</p> <p>Note, not all services support the use of logical volumes. VSA requires raw devices.</p>

## Device Groups

The device-groups configuration object provides the mechanism to make the whole of a physical disk available to a service.

```
device-groups:
  - name: vsa-data
    consumer:
      name: vsa
      usage: data
    devices:
      - name: /dev/sdc
  - name: vsa-cache
    consumer:
      name: vsa
      usage: adaptive-optimization
    devices:
      - name: /dev/sdb
```

Key	Value Descriptions
name	An administrator-defined name for the device group.
devices	A list of named devices to be assigned to this group. There must be at least one device in the group.  Multipath storage should be listed as the corresponding /dev/mapper/mpath<x>
consumer	Identifies the name of one of the storage services (e.g. one of the following: Swift, Cinder, Ceph, VSA, etc) that will consume the disks in this device group.
consumer attributes	These will vary according to the service consuming the device group. The examples section provides sample content for the different services.

## Disk Sizing for Virtual Machine Servers

The vm-size configuration object specifies the names and sizes of disks to be created for a virtual machine server.

```
vm-size:
  disks:
    - name: /dev/vda_root
      size: 1T
    - name: /dev/vdb
      size: 1T
    - name: /dev/vdc
      size: 1T
```

Key	Value Descriptions
disks	A list of disk names and sizes
name	Disk device name
size	The disk size in kilobytes, megabytes, gigabytes or terabytes specified as nX where:  n is an integer greater than zero  X is one of "K", "M" or "G"

## Memory Models

The memory-models configuration object describes details of the optional configuration of Huge Pages. It also describes the amount of memory to be allocated for virtual machine servers.

The memory-model allows the number of pages of a particular size to be configured at the server level or at the numa-node level.

The following example would configure :

- five 2MB pages in each of numa nodes 0 and 1
- three 1GB pages (distributed across all numa nodes)
- six 2MB pages (distributed across all numa nodes)

```
memory-models:
  - name: COMPUTE-MEMORY-NUMA
    default-huge-page-size: 2M
    huge-pages:
      - size: 2M
        count: 5
        numa-node: 0
      - size: 2M
        count: 5
        numa-node: 1
      - size: 1G
        count: 3
      - size: 2M
        count: 6
  - name: VIRTUAL-CONTROLLER-MEMORY
    vm-size:
      ram: 6G
```

Key	Value Description
name	The name of the memory-model that is referenced from one or more server-roles.

Key	Value Description
default-huge-page-size (optional)	The default page size that will be used is specified when allocating huge pages.  If not specified, the default is set by the operating system.
huge-pages	A list of huge page definitions (see below).
vm-size (optional)	Memory sizing information for virtual machine servers.

## Huge Pages

Key	Value Description
size	The page size in kilobytes, megabytes, or gigabytes specified as $nX$ where:  $n$ is an integer greater than zero $X$ is one of "K", "M" or "G"
count	The number of pages of this size to create (must be greater than zero).
numa-node (optional)	If specified the pages will be created in the memory associated with this numa node.  If not specified the pages are distributed across numa nodes by the operating system.

## Memory Sizing for Virtual Machine Servers

Key	Value Description
ram	The amount of memory to be allocated for a virtual machine server in kilobytes, megabytes, or gigabytes specified as $nX$ where:  $n$ is an integer greater than zero $X$ is one of "K", "M" or "G"

## CPU Models

The `cpu-models` configuration object describes how CPUs are assigned for use by service components such as Nova (for VMs) and Open vSwitch (for DPDK), and whether or not those CPUs are isolated from

the general kernel SMP balancing and scheduling algorithms. It also describes the number of vCPUs for virtual machine servers.

```
---  
product:  
    version: 2  
  
cpu-models:  
    - name: COMPUTE-CPU  
        assignments:  
            - components:  
                - nova-compute-kvm  
                    cpu:  
                        - processor-ids: 0-1,3,5-7  
                            role: vm  
                        - components:  
                            - openvswitch  
                                cpu:  
                                    - processor-ids: 4,12  
                                        isolate: False  
                                        role: eal  
                                    - processor-ids: 2,10  
                                        role: pmd  
                - name: VIRTUAL-CONTROLLER-CPU  
                    vm-size:  
                        vcpus: 4
```

### cpu-models

Key	Value Description
name	An administrator-defined name for the cpu model.
assignments	A list of CPU assignments (see ).
vm-size (optional)	CPU sizing information for virtual machine servers.

## CPU Assignments

### assignments

Key	Value Description
components	A list of components to which the CPUs will be assigned.
cpu	A list of CPU usage objects (see the section called “CPU Usage” below).

## CPU Usage

### cpu

Key	Value Description
processor-ids	A list of CPU IDs as seen by the operating system.
isolate (optional)	A boolean value which indicates if the CPUs are to be isolated from the general kernel SMP balancing and scheduling algorithms. The specified processor IDs will be configured in the Linux kernel isolcpus parameter.  The default value is True.
role	A role within the component for which the CPUs will be used.

## Components and Roles in the CPU Model

Component	Role	Description
nova-compute-kvm	vm	The specified processor IDs will be configured in the Nova vcpu_pin_set option.
openvswitch	eal	The specified processor IDs will be configured in the Open vSwitch DPDK EAL -c (coremask) option. Refer to the DPDK documentation for details.
	pmd	The specified processor IDs will be configured in the Open vSwitch pmd-cpu-mask option. Refer to the Open vSwitch documentation and the ovs-vswitchd.conf.db man page for details.

## CPU sizing for virtual machine servers

Key	Value Description
vcpus	The number of vCPUs for a virtual machine server.

## Interface Models

The interface-models configuration object describes how network interfaces are bonded and the mapping of network groups onto interfaces. Interface devices are identified by name and mapped to a particular physical port by the nic-mapping (see ).

```
---
product:
  version: 2

interface-models:
  - name: INTERFACE_SET_CONTROLLER
    network-interfaces:
      - name: BONDED_INTERFACE
        device:
          name: bond0
        bond-data:
          provider: linux
          devices:
            - name: hed3
            - name: hed4
        options:
          mode: active-backup
```

```
        miimon: 200
        primary: hed3
    network-groups:
        - EXTERNAL_API
        - EXTERNAL_VM
        - GUEST

    - name: UNBONDED_INTERFACE
        device:
            name: hed0
    network-groups:
        - MGMT

fcoe-interfaces:
    - name: FCOE_DEVICES
        devices:
            - eth7
            - eth8

    - name: INTERFACE_SET_DPDK
        network-interfaces:
            - name: BONDED_DPDK_INTERFACE
                device:
                    name: bond0
                bond-data:
                    provider: openvswitch
                    devices:
                        - name: dpdk0
                        - name: dpdk1
                    options:
                        mode: active-backup
                network-groups:
                    - GUEST
            - name: UNBONDED_DPDK_INTERFACE
                device:
                    name: dpdk2
                network-groups:
                    - PHYSNET2
        dpdk-devices:
            - devices:
                - name: dpdk0
                - name: dpdk1
                - name: dpdk2
                    driver: igb_uio
            components:
                - openvswitch
        eal-options:
            - name: socket-mem
                value: 1024,0
            - name: n
                value: 2
    component-options:
```

```
- name: n-dpdk-rxqs
  value: 64
```

Key	Value Description
name	An administrator-defined name for the interface model.
network-interfaces	A list of network interface definitions.
fcoe-interfaces (optional)	A list of network interfaces that will be used for Fibre Channel over Ethernet (FCoE). This is only needed for devices that present as a native FCoE device, not cards such as Emulex which present FCoE as a FC device.
<b>Important</b>	
	The devices must be “raw” device names, not names controlled via a nic-mapping.
dpdk-devices (optional)	A list of DPDK device definitions.

## network-interfaces

The network-interfaces configuration object has the following attributes:

Key	Value Description
name	An administrator-defined name for the interface
device	A dictionary containing the network device name (as seen on the associated server) and associated properties (see the section called “network-interfaces device”F for details).
bond-data (optional)	See the section called “Bonding” for details.
network-groups (optional if forced-network-groups is defined)	A list of one or more network-groups (see the section called “Network Groups”) containing networks (see the section called “Networks”) that can be accessed via this interface. Networks in these groups will only be configured if there is at least one service-component on the server which matches the list of component-endpoints defined in the network-group.
forced-network-groups (optional if network-groups is defined)	A list of one or more network-groups (see the section called “Network Groups”) containing networks (see the section called “Networks”) that can be accessed via this interface. Networks in these groups are always configured on the server.
passthrough-network-groups (optional)	A list of one or more network-groups (see the section called “Network Groups”) containing networks (see the section called “Networks”) that can be accessed by servers running as virtual machines

Key	Value Description
	on an HLM hypervisor server. Networks in these groups are not configured on the HLM hypervisor server unless they also are specified in the <code>network-groups</code> or <code>forced-network-groups</code> attributes.

## network-interfaces device

### network-interfaces device

The network-interfaces device configuration object has the following attributes:

Key	Value Description
name	When configuring a bond, this is used as the bond device name - the names of the devices to be bonded are specified in the bond-data section.  If the interface is not bonded, this must be the name of the device specified by the nic-mapping (see NIC Mapping).
vf-count (optional)	Indicates that the interface is to be used for SR-IOV. The value is the number of virtual functions to be created. The associated device specified by the nic-mapping must have a valid nice-device-type.  vf-count cannot be specified on bonded interfaces  Interfaces used for SR-IOV must be associated with a network with <code>tagged-vlan: false</code> .
sriov-only (optional)	Only valid when vf-count is specified. If set to true then the interface is to be used for virtual functions only and the physical function will not be used.  The default value is False.
pci-pt (optional)	If set to true then the interface is used for PCI passthrough.  The default value is False.

## Bonding

A bond-data definition is used to configure a bond device, and consists of the following attributes:

Key	Value Descriptions
provider	Identifies the software used to instantiate the bond device. The supported values are <ul style="list-style-type: none"> <li>• <b>linux</b> to use the Linux bonding driver.</li> <li>• <b>windows</b> (for Windows hyperV servers)</li> <li>• <b>openvswitch</b> to use Open vSwitch bonding.</li> </ul>

Key	Value Descriptions
devices	A dictionary containing network device names used to form the bond. The device names must be the logical-name specified by the nic-mapping (see the section called “NIC Mapping”).
options	A dictionary containing bond configuration options. The <i>linux</i> provider options are described in the the section called “Bond configuration options for the “linux” provider” section. The <i>openvswitch</i> provider options are described in the the section called “Bond Data Options for the “openvswitch” Provider” section.

## Bond configuration options for the "linux" provider

The Linux bonding driver supports a large number of parameters that control the operation of the bond, as described in the Linux Ethernet Bonding Driver HOWTO [<https://www.kernel.org/doc/Documentation/networking/bonding.txt>] document. The parameter names and values may be specified as key-value pairs in the options section of bond-data.

Options used in the HPE Helion OpenStack examples are:

Key	Value Descriptions
mode	Specifies the bonding policy. Possible values are: <ul style="list-style-type: none"> <li>• balance-rr - Transmit packets in sequential order from the first available slave through the last.</li> <li>• active-backup - Only one slave in the bond is active. A different slave becomes active if, and only if, the active slave fails.</li> <li>• balance-xor - Transmit based on the selected transmit hash policy.</li> <li>• broadcast - Transmits everything on all slave interfaces.</li> <li>• 802.3ad - IEEE 802.3ad Dynamic link aggregation.</li> <li>• balance-tlb - Adaptive transmit load balancing: channel bonding that does not require any special switch support.</li> <li>• balance-alb - Adaptive load balancing: includes balance-tlb plus receive load balancing (rlb) for IPV4 traffic and does not require any special switch support.</li> </ul>
miimon	Specifies the MII link monitoring frequency in milliseconds. This determines how often the link state of each slave is inspected for link failures. Accepts values in milliseconds.

Key	Value Descriptions
primary	<p>The device to use as the primary when the mode is one of the possible values below:</p> <ul style="list-style-type: none"> <li>• active-backup</li> <li>• balance-tlb</li> <li>• balance-alb</li> </ul>

## Bond Data Options for the "openvswitch" Provider

The bond configuration options for Open vSwitch bonds are:

Key	Value Descriptions
mode	<p>Specifies the bonding mode. Possible values include:</p> <ul style="list-style-type: none"> <li>• active-backup</li> <li>• balance#tcp</li> <li>• balance#slb</li> </ul> <p>Refer to the Open vSwitch ovs-vswitchd.conf.db man page for details.</p>

## Bond configuration options for the "windows" provider

Bond configuration options for windows bonds are:

Key	Value Descriptions
mode	<p>Specifies the bonding mode. Possible values are:</p> <ul style="list-style-type: none"> <li>• SwitchIndependent</li> <li>• Static</li> <li>• LACP</li> </ul> <p>Refer to the Windows HyperV documentation for details.</p>

## fcoe-interfaces

The fcoe-interfaces configuration object has the following attributes:

Key	Value Description
name	An administrator-defined name for the group of FCOE interfaces
devices	A list of network devices that will be configured for FCOE

Key	Value Description
	Entries in this must be the name of a device specified by the nic-mapping (see the section called “NIC Mappings”).

## dpdk-devices

The dpdk-devices configuration object has the following attributes:

Key	Value Descriptions
devices	A list of network devices to be configured for DPDK. See the section called “dpdk-devices devices”.
eal-options	A list of key-value pairs that may be used to set DPDK Environmental Abstraction Layer (EAL) options. Refer to the DPDK documentation for details.  Note that the cpu-model should be used to specify the processor IDs to be used by EAL for this component. The EAL coremask (-c) option will be set automatically based on the information in the cpu-model, and so should not be specified here. See the section called “CPU Models”.
component-options	A list of key-value pairs that may be used to set component-specific configuration options.

## dpdk-devices devices

The devices configuration object within dpdk-devices has the following attributes:

Key	Value Descriptions
name	The name of a network device to be used with DPDK. The device names must be the logical-name specified by the nic-mapping (see the section called “NIC Mappings”).
driver (optional)	Defines the userspace I/O driver to be used for network devices where the native device driver does not provide userspace I/O capabilities.  The default value is <code>igb_uio</code> .

## DPDK component-options for the openvswitch component

The following options are supported for use with the openvswitch component:

Name	Value Descriptions
n-dpdk-rxqs	Number of rx queues for each DPDK interface. Refer to the Open vSwitch documentation and the <code>ovs-vswitchd.conf.db</code> man page for details.

Note that the cpu-model should be used to define the cpu affinity of the Open vSwitch PMD (Poll Mode Driver) threads. The Open vSwitch pmd-cpu-mask option will be set automatically based on the information in the cpu-model. See the section called “CPU Models”

## NIC Mappings

The nic-mappings configuration object is used to ensure that the network device name used by the operating system always maps to the same physical device. A nic-mapping is associated to a server in the server definition file. (see ). Devices should be named hedN to avoid name clashes with any other devices configured during the operating system install as well as any interfaces that are not being managed by HPE Helion OpenStack, ensuring that all devices on a baremetal machine are specified in the file. An excerpt from `nic_mappings.yml` illustrates:

```
---
product:
  version: 2

nic-mappings:
  - name: HP-DL360-4PORT
    physical-ports:
      - logical-name: hed1
        type: simple-port
        bus-address: "0000:07:00.0"

      - logical-name: hed2
        type: simple-port
        bus-address: "0000:08:00.0"
        nic-device-type: '8086:10fb'

      - logical-name: hed3
        type: multi-port
        bus-address: "0000:09:00.0"
        port-attributes:
          port-num: 0

      - logical-name: hed4
        type: multi-port
        bus-address: "0000:09:00.0"
        port-attributes:
          port-num: 1
```

Each entry in the nic-mappings list has the following attributes:

Key	Value Description
name	An administrator-defined name for the mapping. This name may be used in a server definition (see the section called “Servers”) to apply the mapping to that server.
physical-ports	A list containing device name to address mapping information.

Each entry in the physical-ports list has the following attributes:

Key	Value Description
logical-name	The network device name that will be associated with the device at the specified <i>bus-address</i> . The logical-name specified here can be used as a device name in network interface model definitions. (see the section called “Interface Models”)
type	<p>The type of port. HPE Helion OpenStack 5.0 supports "simple-port" and "multi-port". Use "simple-port" if your device has a unique bus-address. Use "multi-port" if your hardware requires a "port-num" attribute to identify a single port on a multi-port device. An examples of such a device is:</p> <ul style="list-style-type: none"> <li>Mellanox Technologies MT26438 [ConnectX VPI PCIe 2.0 5GT/s - IB QDR / 10GigE Virtualization+]</li> </ul>
bus-address	PCI bus address of the port. Enclose the bus address in quotation marks so yaml does not misinterpret the embedded colon (:) characters. See Chapter 2, <i>Pre-Installation Checklist</i> for details on how to determine this value.
port-attributes (required if type is multi-port)	Provides a list of attributes for the physical port. The current implementation supports only one attribute, "port-num". Multi-port devices share a bus-address. Use the "port-num" attribute to identify which physical port on the multi-port device to map. See Chapter 2, <i>Pre-Installation Checklist</i> for details on how to determine this value.
nic-device-type (optional)	Specifies the PCI vendor ID and device ID of the port in the format of <vendor_id>:<device_id>, for example, 8086:10fb.

## NIC Mappings for Virtual Machine Servers

Virtual machine servers use the standard nic-mappings format described above, subject to the following constraints:

- logical name with unit number 0 (e.g. `hed0`) is not supported
- port type must be `simple-port`
- bus addresses must use the following sequence of values: `0000:01:01.0`, `0000:01:02.0`, `0000:01:03.0`, etc.
- `port-attributes` is not supported
- `nic-device-type` is not supported
- in the interface model for the virtual machine server, the device at bus address `0000:01:01.0` (e.g. `hed1`) must host the network group associated with `ip-addr` attribute defined in the virtual machine server’s server object

Here are example nic-mappings for virtual machine servers with one vNIC and four vNICs:

```
- name: VIRTUAL-1PORT
  physical-ports:
    - logical-name: hed1
      type: simple-port
      bus-address: "0000:01:01.0"

- name: VIRTUAL-4PORT
  physical-ports:
    - logical-name: hed1
      type: simple-port
      bus-address: "0000:01:01.0"
    physical-ports:
      - logical-name: hed2
        type: simple-port
        bus-address: "0000:01:02.0"
    physical-ports:
      - logical-name: hed3
        type: simple-port
        bus-address: "0000:01:03.0"
    physical-ports:
      - logical-name: hed4
        type: simple-port
        bus-address: "0000:01:04.0"
```

## Network Groups

Network-groups define the overall network topology, including where service-components connect, what load balancers are to be deployed, which connections use TLS, and network routing. They also provide the data needed to map Neutron's network configuration to the physical networking.

```
---
product:
  version: 2

network-groups:

- name: EXTERNAL-API
  hostname-suffix: extapi

  load-balancers:
    - provider: ip-cluster
      name: extlb
      external-name:

      tls-components:
        - default
      roles:
        - public
      cert-file: my-public-helion-cert

- name: EXTERNAL-VM
  tags:
```

```

    - neutron.13_agent.external_network_bridge

    - name: GUEST
      hostname-suffix: guest
      tags:
        - neutron.networks.vxlan

    - name: MANAGEMENT
      hostname-suffix: mgmt
      hostname: true

      component-endpoints:
        - default

      routes:
        - default

      load-balancers:
        - provider: ip-cluster
          name: lb
          components:
            - default
          roles:
            - internal
            - admin

      tags:
        - neutron.networks.vlan:
          provider-physical-network: physnet1

```

Key	Value Description
name	An administrator-defined name for the network group. The name is used to make references from other parts of the input model.
component-endpoints (optional)	The list of service-components that will bind to or need direct access to networks in this network-group.
hostname (optional)	If set to true, the name of the address associated with a network in this group will be used to set the hostname of the server.
	<p style="text-align: center;"><b>Important</b></p> <p style="text-align: center;">hostname<b>must</b> be set to <b>true</b> for one, and only one, of your network groups.</p>
hostname-suffix (optional)	If supplied, this string will be used in the name generation (see the section called “Name Generation”). If not specified, the name of the network-group will be used.
load-balancers (optional)	A list of load balancers to be configured on networks in this network-group. Because load balances need a virtual IP address, any network group that contains a

<b>Key</b>	<b>Value Description</b>
	<p>load balancer can only have one network associated with it.</p> <p>For clouds consisting of a single control plane, a load balancer may be fully defined within a <code>network-group</code> object. See Load balancer definitions in network groups.</p> <p>Starting in HPE Helion OpenStack 5.0, a load balancer may be defined within a <code>control-plane</code> object and referenced by name from a <code>network-group</code> object. See the section called “Load Balancer Definitions in Network Groups” in control planes.</p>
routes (optional)	<p>A list of network-groups that networks in this group provide access to via their gateway. This can include the value <code>default</code> to define the default route.</p> <p>A network group with no services attached to it can be used to define routes to external networks.</p> <p>The name of a Neutron provider network defined via configuration-data (see the section called “neutron-provider-networks”) can also be included in this list.</p>
tags (optional)	<p>A list of network tags. Tags provide the linkage between the physical network configuration and the Neutron network configuration.</p> <p>Starting in HPE Helion OpenStack 5.0, network tags may be defined as part of a Neutron <code>configuration-data</code> object rather than as part of a <code>network-group</code> object (see the section called “Neutron Configuration Data”).</p>
mtu (optional)	<p>Specifies the MTU value required for networks in this network group. If not specified a default value of 1500 is used.</p> <p>See the section called “MTU (Maximum Transmission Unit)” on how MTU settings are applied to interfaces when there are multiple tagged networks on the same interface.</p>

A load balancer definition has the following attributes:

<b>Key</b>	<b>Value Description</b>
name	An administrator-defined name for the load balancer.
provider	The service component that implements the load balancer. Currently only "ip-cluster" (ha-proxy) is supported. Future releases will provide support for external load balancers.

Key	Value Description
roles	The list of endpoint roles that this load balancer provides (see below). Valid roles are "public", "internal", and "admin". To ensure separation of concerns, the role "public" cannot be combined with any other role. See the section called "Load Balancers" for an example of how the role provides endpoint separation.
components (optional)	The list of service-components for which the load balancer provides a non-encrypted virtual IP address for.
tls-components (optional)	The list of service-components for which the load balancer provides TLS-terminated virtual IP addresses for. In HPE Helion OpenStack 3.0, TLS is now supported for internal as well as public endpoints.
external-name (optional)	The name to be registered in Keystone for the public-CURL. If not specified, the virtual IP address will be registered. Note that this value cannot be changed after the initial deployment.
cert-file (optional)	The name of the certificate file to be used for TLS endpoints.

## Load Balancer Definitions in Network Groups

In a cloud consisting of a single control-plane, a `load-balancer` may be fully defined within a `network-groups` object as shown in the examples above. See section the section called "Load Balancers" for a complete description of load balancer attributes.

Starting in HPE Helion OpenStack 5.0, a `load-balancer` may be defined within a `control-plane` object in which case the network-group provides just a list of load balancer names as shown below. See section the section called "Load Balancers" definitions in control planes.

`network-groups`:

```

- name: EXTERNAL-API
  hostname-suffix: extapi

  load-balancers:
    - lb-cp1
    - lb-cp2
  
```

The same load balancer name can be used in multiple control-planes to make the above list simpler.

## Network Tags

HPE Helion OpenStack supports a small number of network tags which may be used to convey information between the input model and the service components (currently only Neutron uses network tags). A network tag consists minimally of a tag name; but some network tags have additional attributes.

**neutron.networks.vxlan**

Tag	Value Description
neutron.networks.vxlan	This tag causes Neutron to be configured to use VxLAN as the underlay for tenant networks. The associated network group will carry the VxLAN traffic.
tenant-vxlan-id-range (optional)	Used to specify the VxLAN identifier range in the format “<min-id>:<max-id>”. The default range is "1001:65535". Enclose the range in quotation marks. Multiple ranges can be specified as a comma-separated list.

Example using the default ID range:

```
tags:
- neutron.networks.vxlan
```

Example using a user-defined ID range:

```
tags:
- neutron.networks.vxlan:
  tenant-vxlan-id-range: "1:20000"
```

Example using multiple user-defined ID range:

```
tags:
- neutron.networks.vxlan:
  tenant-vxlan-id-range: "1:2000,3000:4000,5000:6000"
```

### **neutron.networks.vlan**

Tag	Value Description
neutron.networks.vlan	This tag causes Neutron to be configured for provider VLAN networks, and optionally to use VLAN as the underlay for tenant networks. The associated network group will carry the VLAN traffic. This tag can be specified on multiple network groups.  NOTE: this tag does not cause any Neutron networks to be created, that must be done in Neutron after the cloud is deployed.
provider-physical-network	The provider network name. This is the name to be used in the Neutron API for the <i>provider:physical_network</i> parameter of network objects.
tenant-vlan-id-range (optional)	This attribute causes Neutron to use VLAN for tenant networks; omit this attribute if you are using provider VLANs only. It specifies the VLAN ID range for tenant networks, in the format “<min-id>:<max-id>”. Enclose the range in quotation

Tag	Value Description
	marks. Multiple ranges can be specified as a comma-separated list.

Example using a provider vlan only (may be used with tenant VxLAN):

```
tags:  
  - neutron.networks.vlan:  
    provider-physical-network: physnet1
```

Example using a tenant and provider VLAN:

```
tags:  
  - neutron.networks.vlan:  
    provider-physical-network: physnet1  
    tenant-vlan-id-range: "30:50,100:200"
```

#### **neutron.networks.flat**

Tag	Value Description
neutron.networks.flat	This tag causes Neutron to be configured for provider flat networks. The associated network group will carry the traffic. This tag can be specified on multiple network groups.  NOTE: this tag does not cause any Neutron networks to be created, that must be done in Neutron after the cloud is deployed.
provider-physical-network	The provider network name. This is the name to be used in the Neutron API for the <i>provider:physical_network</i> parameter of network objects. When specified on multiple network groups, the name must be unique for each network group.

Example using a provider flat network:

```
tags:  
  - neutron.networks.flat:  
    provider-physical-network: flatnet1
```

#### **neutron.l3\_agent.external\_network\_bridge**

Tag	Value Description
neutron.l3_agent.external_network_bridge	This tag causes the Neutron L3 Agent to be configured to use the associated network group as the Neutron external network for floating IP addresses. A CIDR <b>should not</b> be defined for the associated physical network, as that will cause addresses from that network to be configured in the hypervisor. When this tag is used, provider networks cannot be used as external networks.

Tag	Value Description
	NOTE: this tag does not cause a Neutron external networks to be created, that must be done in Neutron after the cloud is deployed.

Example using neutron.l3\_agent.external\_network\_bridge:

```
tags:
  - neutron.l3_agent.external_network_bridge
```

## MTU (Maximum Transmission Unit)

A network group may optionally specify an MTU for its networks to use. Because a network-interface in the interface-model may have a mix of one untagged-vlan network group and one or more tagged-vlan network groups, there are some special requirements when specifying an MTU on a network group.

If the network group consists of untagged-vlan network(s) then its specified MTU must be greater than or equal to the MTU of any tagged-vlan network groups which are co-located on the same network-interface.

For example consider a network group with untagged VLANs, NET-GROUP-1, which is going to share (via a Network Interface definition) a device (eth0) with two network groups with tagged VLANs: NET-GROUP-2 (ID=201, MTU=1550) and NET-GROUP-3 (ID=301, MTU=9000).

The device (eth0) must have an MTU which is large enough to accommodate the VLAN in NET-GROUP-3. Since NET-GROUP-1 has untagged VLANs it will also be using this device and so it must also have an MTU of 9000, which results in the following configuration.

```
+eth0 (9000)      ----- this MTU comes from NET-GROUP-1
| |
| +---+ vlan201@eth0 (1550)
\-----+ vlan301@eth0 (9000)
```

Where an interface is used only by network groups with tagged VLANs the MTU of the device or bond will be set to the highest MTU value in those groups.

For example if bond0 is configured to be used by three network groups: NET-GROUP-1 (ID=101, MTU=3000) , NET-GROUP-2 (ID=201, MTU=1550) and NET-GROUP-3 (ID=301, MTU=9000).

Then the resulting configuration would be:

```
+bond0 (9000)      ----- because of NET-GROUP-3
| | |
| +---+vlan101@bond0 (3000)
| +---+vlan201@bond0 (1550)
\-----+vlan301@bond0 (9000)
```

## Networks

A network definition represents a physical L3 network used by the cloud infrastructure. Note that these are different from the network definitions that are created/configured in Neutron, although some of the networks may be used by Neutron.

```
---  
product:  
    version: 2  
  
networks:  
    - name: NET_EXTERNAL_VM  
        vlanid: 102  
        tagged-vlan: true  
        network-group: EXTERNAL_VM  
  
    - name: NET_GUEST  
        vlanid: 103  
        tagged-vlan: true  
        cidr: 10.1.1.0/24  
        gateway-ip: 10.1.1.1  
        network-group: GUEST  
  
    - name: NET_MGMT  
        vlanid: 100  
        tagged-vlan: false  
        cidr: 10.2.1.0/24  
        addresses:  
            - 10.2.1.10-10.2.1.20  
            - 10.2.1.24  
            - 10.2.1.30-10.2.1.36  
        gateway-ip: 10.2.1.1  
        network-group: MGMT
```

Key	Value Description
name	The name of this network. The network <i>name</i> may be used in a server-group definition (see the section called “Server Groups”) to specify a particular network from within a network-group to be associated with a set of servers.
network-group	The name of the associated network group.
vlanid (optional)	The IEEE 802.1Q VLAN Identifier, a value in the range 1 through 4094. A <i>vlanid</i> must be specified when <i>tagged-vlan</i> is true.
tagged-vlan (optional)	May be set to <code>true</code> or <code>false</code> . If true, packets for this network carry the <i>vlanid</i> in the packet header; such packets are referred to as VLAN-tagged frames in IEEE 802.1Q.
cidr (optional)	The IP subnet associated with this network.
addresses (optional)	A list of IP addresses or IP address ranges (specified as <start addr>-<end addr> from which server addresses may be allocated. The default value is the first host address within the CIDR (e.g. the .1 address).  The <i>addresses</i> parameter provides more flexibility than the <i>start-address</i> and <i>end-ad-</i>

Key	Value Description
	dress parameters and so is the preferred means of specifying this data.
start-address (optional) (deprecated)	An IP address within the <i>CIDR</i> which will be used as the start of the range of IP addresses from which server addresses may be allocated. The default value is the first host address within the <i>CIDR</i> (e.g. the .1 address).
	<p style="text-align: center;"><b>Important</b></p> <p>This parameter is deprecated in favor of the new <code>addresses</code> parameter. This parameter may be removed in a future release.</p>
end-address (optional) (deprecated)	An IP address within the <i>CIDR</i> which will be used as the end of the range of IP addresses from which server addresses may be allocated. The default value is the last host address within the <i>CIDR</i> (e.g. the .254 address of a /24).
	<p style="text-align: center;"><b>Important</b></p> <p>This parameter is deprecated in favor of the new <code>addresses</code> parameter. This parameter may be removed in a future release.</p>
gateway-ip (optional)	The IP address of the gateway for this network. Gateway addresses must be specified if the associated network-group provides routes.

## Firewall Rules

The configuration processor will automatically generate "allow" firewall rules for each server based on the services deployed and block all other ports. The firewall rules in the input model allow the customer to define additional rules for each network group.

Administrator-defined rules are applied after all rules generated by the Configuration Processor.

```
---
product:
  version: 2

firewall-rules:

  - name: PING
    network-groups:
      - MANAGEMENT
      - GUEST
      - EXTERNAL-API
    rules:
      # open ICMP echo request (ping)
```

```
- type: allow
  remote-ip-prefix: 0.0.0.0/0
  # icmp type
  port-range-min: 8
  # icmp code
  port-range-max: 0
  protocol: icmp
```

Key	Value Description
name	An administrator-defined name for the group of rules.
network-groups	A list of network-group names that the rules apply to. A value of "all" matches all network-groups.
rules	A list of rules. Rules are applied in the order in which they appear in the list, apart from the control provided by the "final" option (see above). The order between sets of rules is indeterminate.

## Rule

Each rule in the list takes the following parameters (which match the parameters of a Neutron security group rule):

Key	Value Description
type	Must allow
remote-ip-prefix	Range of remote addresses in CIDR format that this rule applies to.
port-range-min	Defines the range of ports covered by the rule. Note that if the protocol is icmp then port-range-min is the ICMP type and port-range-max is the ICMP code.
port-range-max	
protocol	Must be one of tcp, udp, or icmp.

## Configuration Data

Configuration data allows values to be passed into the model to be used in the context of a specific control plane or cluster. The content and format of the data is service specific.

```
---
product:
  version: 2

configuration-data:
  - name: NEUTRON-CONFIG-CP1
    services:
      - neutron
  data:
    neutron_provider_networks:
      - name: OCTAVIA-MGMT-NET
```

```
provider:
  - network_type: vlan
    physical_network: physnet1
    segmentation_id: 106
  cidr: 172.30.1.0/24
  no_gateway: True
  enable_dhcp: True
  allocation_pools:
    - start: 172.30.1.10
      end: 172.30.1.250
  host_routes:
    # route to MANAGEMENT-NET-1
    - destination: 192.168.245.0/24
      nexthop: 172.30.1.1

  neutron_external_networks:
    - name: ext-net
      cidr: 172.31.0.0/24
      gateway: 172.31.0.1
      provider:
        - network_type: vlan
          physical_network: physnet1
          segmentation_id: 107
      allocation_pools:
        - start: 172.31.0.2
          end: 172.31.0.254

  network-tags:
    - network-group: MANAGEMENT
      tags:
        - neutron.networks.vxlan
        - neutron.networks.vlan:
          provider-physical-network: physnet1
    - network-group: EXTERNAL-VM
      tags:
        - neutron.l3_agent.external_network_bridge
```

Key	Value Description
name	An administrator-defined name for the set of configuration data.
services	A list of services that the data applies to. Note that these are service names (e.g. neutron, octavia, etc) not service-component names (neutron-server, octavia-api, etc).
data	A service specific data structure (see below).
network-tags (optional, Neutron-only)	A list of network tags. Tags provide the linkage between the physical network configuration and the Neutron network configuration.  Starting in HPE Helion OpenStack 5.0, network tags may be defined as part of a Neutron configura-

Key	Value Description
	tion-data object rather than as part of a network-group object.

## Neutron network-tags

Key	Value Description
network-group	The name of the network-group with which the tags are associated.
tags	A list of network tags. Tags provide the linkage between the physical network configuration and the Neutron network configuration. See section Network Tags.

## Neutron Configuration Data

Key	Value Description
neutron-provider-networks	A list of provider networks that will be created in Neutron.
neutron-external-networks	A list of external networks that will be created in Neutron. These networks will have the “router:external” attribute set to True.

### neutron-provider-networks

Key	Value Description
name	The name for this network in Neutron.  This name must be distinct from the names of any Network Groups in the model to enable it to be included in the “routes” value of a network group.
provider	Details of network to be created <ul style="list-style-type: none"> <li>• network_type</li> <li>• physical_network</li> <li>• segmentation_id</li> </ul> These values are passed as --provider: options to the Neutron net-create command
cidr	The CIDR to use for the network. This is passed to the Neutron subnet-create command.
shared (optional)	A boolean value that specifies if the network can be shared.  This value is passed to the Neutron net-create command.

Key	Value Description
allocation_pools (optional)	A list of start and end address pairs that limit the set of IP addresses that can be allocated for this network.  These values are passed to the Neutron subnet-create command.
host_routes (optional)	A list of routes to be defined for the network. Each route consists of a destination in cidr format and a nexthop address.  These values are passed to the Neutron subnet-create command.
gateway_ip (optional)	A gateway address for the network.  This value is passed to the Neutron subnet-create command.
no_gateway (optional)	A Boolean value indicating that the gateway should not be distributed on this network.  This is translated into the no-gateway option to the Neutron subnet-create command
enable_dhcp (optional)	A Boolean value indicating that DHCP should be enabled. The default if not specified is to not enable DHCP.  This value is passed to the Neutron subnet-create command.

## neutron-external-networks

Key	Value Description
name	The name for this network in Neutron.  This name must be distinct from the names of any Network Groups in the model to enable it to be included in the “routes” value of a network group.
provider (optional)	The provider attributes are specified when using Neutron provider networks as external networks. Provider attributes should not be specified when the external network is configured with the neutron.l3_agent.external_network_bridge.  Standard provider network attributes may be specified: <ul style="list-style-type: none"><li>• network_type</li><li>• physical_network</li><li>• segmentation_id</li></ul>

Key	Value Description
	These values are passed as --provider: options to the Neutron net-create command
cidr	The CIDR to use for the network. This is passed to the Neutron subnet-create command.
allocation_pools (optional)	A list of start and end address pairs that limit the set of IP addresses that can be allocated for this network.  These values are passed to the Neutron subnet-create command.
gateway (optional)	A gateway address for the network.  This value is passed to the Neutron subnet-create command.

## Octavia Configuration Data

```
---
product:
  version: 2

configuration-data:
  - name: OCTAVIA-CONFIG-CP1
    services:
      - octavia
    data:
      amp_network_name: OCTAVIA-MGMT-NET
```

Key	Value Description
amp_network_name	The name of the Neutron provider network that Octavia will use for management access to load balancers.

## Ironic Configuration Data

```
---
product:
  version: 2

configuration-data:
  - name: IRONIC-CONFIG-CP1
    services:
      - ironic
    data:
      cleaning_network: guest-network
      enable_node_cleaning: true
      enable_oneview: false
```

```
oneview_manager_url:  
oneview_username:  
oneview_encrypted_password:  
oneview_allow_insecure_connections:  
tls_cacert_file:  
enable_agent_drivers: true
```

Refer to the documentation on configuring Ironic for details of the above attributes.

## Swift Configuration Data

```
---  
product:  
  version: 2  
  
configuration-data:  
- name: SWIFT-CONFIG-CP1  
  services:  
    - swift  
  data:  
    control_plane_rings:  
      swift-zones:  
        - id: 1  
          server-groups:  
            - AZ1  
        - id: 2  
          server-groups:  
            - AZ2  
        - id: 3  
          server-groups:  
            - AZ3  
      rings:  
        - name: account  
          display-name: Account Ring  
          min-part-hours: 16  
          partition-power: 12  
          replication-policy:  
            replica-count: 3  
  
        - name: container  
          display-name: Container Ring  
          min-part-hours: 16  
          partition-power: 12  
          replication-policy:  
            replica-count: 3  
  
        - name: object-0  
          display-name: General  
          default: yes  
          min-part-hours: 16  
          partition-power: 12  
          replication-policy:
```

```
replica-count: 3
```

Refer to the documentation on the section called “Understanding Swift Ring Specifications” for details of the above attributes.

## Pass Through

Through pass\_through definitions, certain configuration values can be assigned and used.

```
product:
  version: 2

pass-through:
  global:
    esx_cloud: true
  servers:

    data:
      vmware:
        cert_check: false
        vcenter_cluster: Cluster1
        vcenter_id: BC9DED4E-1639-481D-B190-2B54A2BF5674
        vcenter_ip: 10.1.200.41
        vcenter_port: 443
        vcenter_username: administrator@vsphere.local
        id: 7d8c415b541ca9ecf9608b35b32261e6c0bf275a
```

Key	Value Description
global	These values will be used at the cloud level.
servers	These values will be assigned to a specific server(s) using the server-id.

# Chapter 8. Other Topics

## Services and Service Components

		Service	Service Components
Compute	Virtual Machine Provisioning	nova	nova-api nova-compute nova-compute-hyperv nova-compute-ironic nova-compute-kvm nova-conductor nova-console-auth nova-esx-compute-proxy nova-metadata nova-novncproxy nova-scheduler nova-scheduler-ironic
	Bare Metal Provisioning	ironic	ironic-api ironic-conductor
	ESX Integration	eon	eon-api eon-conductor
Networking	Networking	neutron	infoblox-ipam-agent neutron-dhcp-agent neutron-l2gateway-agent neutron-l3-agent neutron-lbaas-agent neutron-lbaasv2-agent neutron-metadata-agent neutron-ml2-plugin neutron-openvswitch-agent neutron-ovsvapp-agent neutron-server neutron-sriov-nic-agent neutron-vpn-agent
	Network Load Balancer	octavia	octavia-api octavia-health-manager
	Domain Name Service (DNS)	designate	designate-api designate-central designate-mdns designate-mdns-external designate-pool-manager designate-zone-manager
Storage	Ceph Storage	ceph	ceph-monitor ceph-osd ceph-osd-internal ceph-radosgw
	Block Storage	cinder	cinder-api cinder-backup cinder-scheduler

		<b>Service</b>	<b>Service Components</b>
			cinder-volume
	Object Storage	swift	swift-account swift-common swift-container swift-object swift-proxy swift-ring-builder swift-rsync
	HPE Virtual Storage Array	vsa-storage	cmc-service vsa
Image	Image Management	glance	glance-api glance-registry
Security	Key Management	barbican	barbican-api barbican-worker
	Identity and Authentication	keystone	keystone-api
Orchestration	Orchestration	heat	heat-api heat-api-cfn heat-api-cloudwatch heat-engine
Operations	Telemetry	ceilometer	ceilometer-agent-notification ceilometer-api ceilometer-common ceilometer-polling
	Backup and Recovery	freezer	freezer-agent freezer-api
	Helion Lifecycle Management	hlm	hlm-ux-services lifecycle-manager lifecycle-manager-target
	Dashboard	horizon	horizon
	Centralized Logging	logging	logging-api logging-producer logging-rotate logging-server
	Monitoring	monasca	monasca-agent monasca-api monasca-dashboard monasca-liveness-check monasca-notifier monasca-persistor monasca-threshold monasca-transform
	Operations Console	operations	ops-console-web
	Openstack Functional Test Suite	tempest	tempest
	Foundation	clients	barbican-client

		<b>Service</b>	<b>Service Components</b>
			ceilometer-client cinder-client designate-client eon-client glance-client heat-client ironic-client keystone-client monasca-client neutron-client nova-client openstack-client swift-client
	Supporting Services	foundation	apache2 bind bind-ext helion-ca influxdb ip-cluster kafka memcached mysql ntp-client ntp-server openvswitch powerdns powerdns-ext rabbitmq spark storm vertica zookeeper

## Name Generation

Names are generated by the configuration processor for all allocated IP addresses. A server connected to multiple networks will have multiple names associated with it. One of these may be assigned as the host-name for a server via the network-group configuration (see the section called “NIC Mappings”). Names are generated from data taken from various parts of the input model as described in the following sections.

### Clusters

Names generated for servers in a cluster have the following form:

```
<cloud>-<control-plane>-<cluster><member-prefix><member_id>-<network>
```

Example: helion-cpl1-core-m1-mgmt

<b>Name</b>	<b>Description</b>
<cloud>	Comes from the hostname-data section of the cloud object (see the section called “Cloud Configuration”)

Name	Description
<control-plane>	is the control-plane prefix or name (see the section called “Control Plane”)
<cluster>	is the cluster-prefix name (see the section called “Clusters”)
<member-prefix>	comes from the hostname-data section of the cloud object (see the section called “Cloud Configuration”)
<member_id>	is the ordinal within the cluster, generated by the configuration processor as servers are allocated to the cluster
<network>	comes from the hostname-suffix of the network group to which the network belongs (see the section called “NIC Mappings”).

## Resource Nodes

Names generated for servers in a resource group have the following form:

```
<cloud>-<control-plane>-<resource-prefix><member_id>-<network>
```

Example: helion-cp1-comp0001-mgmt

Name	Description
<cloud>	Comes from the hostname-data section of the cloud object (see the section called “Cloud Configuration”).
<control-plane>	is the control-plane prefix or name (see the section called “Control Plane”).
<resource-prefix>	is the resource-prefix value name (see the section called “Resources”).
<member_id>	is the ordinal within the cluster, generated by the configuration processor as servers are allocated to the cluster, padded with leading zeroes to four digits.
<network>	comes from the hostname-suffix of the network group to which the network belongs to (see the section called “NIC Mappings”)

## Persisted Data

The configuration processor makes allocation decisions on servers and IP addresses which it needs to remember between successive runs so that if new servers are added to the input model they don't disrupt the previously deployed allocations.

To allow users to make multiple iterations of the input model before deployment HPE Helion OpenStack will only persist data when the administrator confirms that they are about to deploy the results via the "ready-deployment" operation. To understand this better, consider the following example:

Imagine you have completed your HPE Helion OpenStack deployment with servers A, B, and C and you want to add two new compute nodes by adding servers D and E to the input model.

When you add these to the input model and re-run the configuration processor it will read the persisted data for A, B, and C and allocate D and E as new servers. The configuration processor now has allocation data for A, B, C, D, and E -- which it keeps in a staging area (actually a special branch in git) until we get confirmation that the configuration processor has done what you intended and you are ready to deploy the revised configuration.

If you notice that the role of E is wrong and it became a Swift node instead of a Nova node you need to be able to change the input model and re-run the configuration processor. This is fine because the allocations of D and E have not been confirmed, and so the configuration processor will re-read the data about A, B, C and re-allocate D and E now to the correct clusters, updating the persisted data in the staging area.

You can loop through this as many times as needed. Each time, the configuration processor is processing the deltas to what is deployed, not the results of the previous run. When you are ready to use the results of the configuration processor, you run `ready-deployment.yaml` which commits the data in the staging area into the persisted data. The next run of the configuration processor will then start from the persisted data for A, B, C, D, and E.

## Persisted Server Allocations

Server allocations are persisted by the administrator-defined server ID (see the section called “Servers”), and include the control plane, cluster/resource name, and ordinal within the cluster or resource group.

To guard against data loss, the configuration processor persists server allocations even when the server ID no longer exists in the input model -- for example, if a server was removed accidentally and the configuration processor allocated a new server to the same ordinal, then it would be very difficult to recover from that situation.

The following example illustrates the behavior:

A cloud is deployed with four servers with IDs of A, B, C, and D that can all be used in a resource group with `min-size=0` and `max-size=3`. At the end of this deployment they persisted state is as follows:

ID	Control Plane	Resource Group	Ordinal	State	Deployed As
A	ccp	compute	1	Allocated	mycloud-ccp-comp0001
B	ccp	compute	2	Allocated	mycloud-ccp-comp0002
C	ccp	compute	3	Allocated	mycloud-ccp-comp0003
D				Available	

(In this example server D has not been allocated because the group is at its max size, and there are no other groups that required this server)

If server B is removed from the input model and the configuration processor is re-run, the state is changed to:

ID	Control Plane	Resource Group	Ordinal	State	Deployed As
A	ccp	compute	1	Allocated	mycloud-ccp-comp0001
B	ccp	compute	2	Deleted	

ID	Control Plane	Resource Group	Ordinal	State	Deployed As
C	ccp	compute	3	Allocated	mycloud-ccp-comp0003
D	ccp	compute	4	Allocated	mycloud-ccp-comp0004

The details associated with server B are still retained, but the configuration processor will not generate any deployment data for this server. Server D has been added to the group to meet the minimum size requirement but has been given a different ordinal and hence will get different names and IP addresses than were given to server B.

If server B is added back into the input model the resulting state will be:

ID	Control Plane	Resource Group	Ordinal	State	Deployed As
A	ccp	compute	1	Allocated	mycloud-ccp-comp0001
B	ccp	compute	2	Deleted	
C	ccp	compute	3	Allocated	mycloud-ccp-comp0003
D	ccp	compute	4	Allocated	mycloud-ccp-comp0004

The configuration processor will issue a warning that server B cannot be returned to the compute group because it would exceed the max-size constraint. However, because the configuration processor knows that server B is associated with this group it won't allocate it to any other group that could use it, since that might lead to data loss on that server.

If the max-size value of the group was increased, then server B would be allocated back to the group, with its previous name and addresses (mycloud-ccp-comp0002).

Note that the configuration processor relies on the server ID to identify a physical server. If the ID value of a server is changed the configuration processor will treat it as a new server. Conversely, if a different physical server is added with the same ID as a deleted server the configuration processor will assume that it is the original server being returned to the model.

You can force the removal of persisted data for servers that are no longer in the input model by running the configuration processor with the `remove_deleted_servers` option, like below:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml -e remove_deleted_servers
```

## Persisted Address Allocations

The configuration processor persists IP address allocations by the generated name (see the section called “Name Generation” for how names are generated). As with servers, once an address has been allocated that address will remain allocated until the configuration processor is explicitly told that it is no longer required. The configuration processor will generate warnings for addresses that are persisted but no longer used.

You can remove persisted address allocations that are no longer used in the input model by running the configuration processor with the `free_unused_addresses` option, like below:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml -e free_unused_addresses
```

## Server Allocation

The configuration processor allocates servers to a cluster or resource group in the following sequence:

1. Any servers that are persisted with a state of "allocated" are first returned to the cluster or resource group. Such servers are always allocated even if this contradicts the cluster size, failure-zones, or list of server roles since it is assumed that these servers are actively deployed.
2. If the cluster or resource group is still below its minimum size, then any servers that are persisted with a state of "deleted", but where the server is now listed in the input model (i.e. the server was removed but is now back), are added to the group providing they meet the failure-zone and server-role criteria. If they do not meet the criteria then a warning is given and the server remains in a deleted state (i.e. it is still not allocated to any other cluster or group). These servers are not part of the current deployment, and so you must resolve any conflicts before they can be redeployed.
3. If the cluster or resource group is still below its minimum size, the configuration processor will allocate additional servers that meet the failure-zone and server-role criteria. If the allocation policy is set to "strict" then the failure zones of servers already in the cluster or resource group are not considered until an equal number of servers has been allocated from each zone.

## Server Network Selection

Once the configuration processor has allocated a server to a cluster or resource group it uses the information in the associated interface-model to determine which networks need to be configured. It does this by:

1. Looking at the service-components that are to run on the server (from the control-plane definition)
2. Looking to see which network-group each of those components is attached to (from the network-groups definition)
3. Looking to see if there are any network-tags related to a service-component running on this server, and if so, adding those network-groups to the list (also from the network-groups definition)
4. Looking to see if there are any network-groups that the interface-model says should be forced onto the server
5. It then searches the server-group hierarchy (as described in the section called “Server Groups and Networks”) to find a network in each of the network-groups it needs to attach to

If there is no network available to a server, either because the interface-model doesn't include the required network-group, or there is no network from that group in the appropriate part of the server-groups hierarchy, then the configuration processor will generate an error.

The configuration processor will also generate an error if the server address does not match any of the networks it will be connected to.

## Network Route Validation

Once the configuration processor has allocated all of the required servers and matched them to the appropriate networks, it validates that all service-components have the required network routes to other service-components.

It does this by using the data in the services section of the input model which provides details of which service-components need to connect to each other. This data is not configurable by the administrator; however, it is provided as part of the HPE Helion OpenStack release.

For each server, the configuration processor looks at the list of service-components it runs and determines the network addresses of every other service-component it needs to connect to (depending on the service, this might be a virtual IP address on a load balancer or a set of addresses for the service).

If the target address is on a network that this server is connected to, then there is no routing required. If the target address is on a different network, then the Configuration Processor looks at each network the server is connected to and looks at the routes defined in the corresponding network-group. If the network-group provides a route to the network-group of the target address, then that route is considered valid.

Networks within the same network-group are always considered as routed to each other; networks from different network-groups must have an explicit entry in the `routes` stanza of the network-group definition. Routes to a named network-group are always considered before a "default" route.

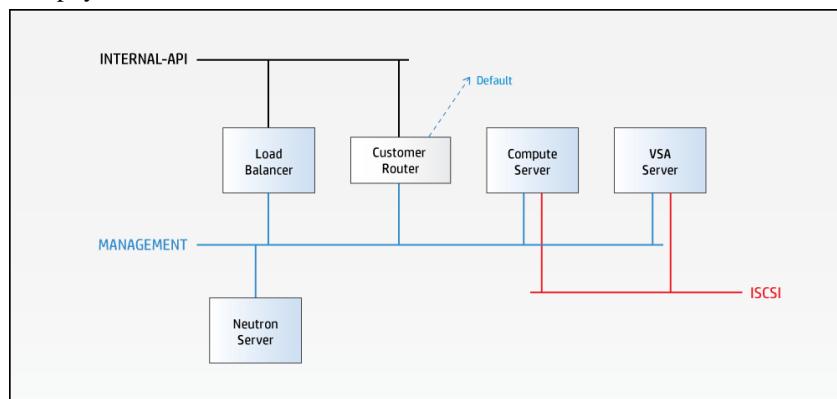
A warning is given for any routes which are using the "default" route since it is possible that the user did not intend to route this traffic. Such warning can be removed by adding the appropriate network-group to the list of routes.

The configuration processor provides details of all routes between networks that it is expecting to be configured in the `info/route_info.yaml` file.

To illustrate how network routing is defined in the input model, consider the following example:

A compute server is configured to run nova-compute which requires access to the Neutron API servers and the VSA block storage service. The Neutron API servers have a virtual IP address provided by a load balancer in the INTERNAL-API network-group and the VSA service is connected to the ISCSI network-group. Nova-compute itself is part of the set of components attached by default to the MANAGEMENT network-group. The intention is to have virtual machines on the compute server connect to the VSA storage via the ISCSI network.

The physical network is shown below:



Download a high-res version [[..../..../media/inputmodel/hphelionopenstack\\_networkroutevalidation\\_1g.png](#)]

The corresponding entries in the network-groups are:

- name: INTERNAL-API  
hostname-suffix: intapi

```
load-balancers:
  - provider: ip-cluster
    name: lb
    components:
      - default
    roles:
      - internal
      - admin

  - name: MANAGEMENT
    hostname-suffix: mgmt
    hostname: true

    component-endpoints:
      - default

  routes:
    - INTERNAL-API
    - default

  - name: ISCSI
    hostname-suffix: iscsi

    component-endpoints:
      - vsa
```

And the interface-model for the compute server looks like this:

```
- name: INTERFACE_SET_COMPUTE
network-interfaces:
  - name: BOND0
    device:
      name: bond0
    bond-data:
      options:
        mode: active-backup
        miimon: 200
        primary: hed5
      provider: linux
    devices:
      - name: hed4
      - name: hed5
  network-groups:
    - MANAGEMENT
    - ISCSI
```

When validating the route from nova-compute to the Neutron API, the configuration processor will detect that the target address is on a network in the INTERNAL-API network group, and that the MANAGEMENT network (which is connected to the compute server) provides a route to this network, and thus considers this route valid.

When validating the route from nova-compute to VSA, the configuration processor will detect that the target address is on a network in the ISCSI network group. However, because there is no service compo-

ment on the compute server connected to the ISCSI network (according to the network-group definition) the ISCSI network will not have been configured on the compute server (see the section called “Server Network Selection”. The configuration processor will detect that the MANAGEMENT network-group provides a “default” route and thus considers the route as valid (it is, of course, valid to route ISCSI traffic); however, because this is using the default route, a warning will be issued:

```
# route-generator-2.0           WRN: Default routing used between networks
The following networks are using a 'default' route rule. To remove this warning
either add an explicit route in the source network group or force the network to
attach in the interface model used by the servers.
  MANAGEMENT-NET-RACK1 to ISCSI-NET
    helion-ccp-comp0001
  MANAGEMENT-NET-RACK 2 to ISCSI-NET
    helion-ccp-comp0002
  MANAGEMENT-NET-RACK 3 to SCSI-NET
    helion-ccp-comp0003
```

To remove this warning, you can either add ISCSI to the list of routes in the MANAGEMENT network group (routed ISCSI traffic is still a valid configuration) or force the compute server to attach to the ISCSI network-group by adding it as a forced-network-group in the interface-model, like this:

```
- name: INTERFACE_SET_COMPUTE
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed5
      provider: linux
      devices:
        - name: hed4
        - name: hed5
  network-groups:
    - MANAGEMENT
  forced-network-groups:
    - ISCSI
```

With the attachment to the ISCSI network group forced, the configuration processor will attach the compute server to a network in that group and validate the route as either being direct or between networks in the same network-group.

The generated `route_info.yml` file will include entries such as the following, showing the routes that are still expected to be configured between networks in the MANAGEMENT network group and the INTERNAL-API network group.

```
MANAGEMENT-NET-RACK1:
  INTERNAL-API-NET:
    default: false
    used_by:
      nova-compute:
```

```
neutron-server:  
  - helion-ccp-comp0001  
MANAGEMENT-NET-RACK2:  
INTERNAL-API-NET:  
  default: false  
  used_by:  
    nova-compute:  
      neutron-server:  
        - helion-ccp-comp0003
```

## Configuring Neutron Provider VLANs

Neutron provider VLANs are networks that map directly to an 802.1Q VLAN in the cloud provider's physical network infrastructure. There are four aspects to a provider VLAN configuration:

- Network infrastructure configuration (e.g. the top-of-rack switch)
- Server networking configuration (for compute nodes and Neutron network nodes)
- Neutron configuration file settings
- Creation of the corresponding network objects in Neutron

The physical network infrastructure must be configured to convey the provider VLAN traffic as tagged VLANs to the cloud compute nodes and Neutron network nodes. Configuration of the physical network infrastructure is outside the scope of the HPE Helion OpenStack 5.0 software.

HPE Helion OpenStack 5.0 automates the server networking configuration and the Neutron configuration based on information in the cloud definition. To configure the system for provider VLANs, specify the `neutron.networks.vlan` tag with a `provider-physical-network` attribute on one or more network-groups as described in the section called "Network Tags". For example (some attributes omitted for brevity):

```
network-groups:  
  - name: NET_GROUP_A  
    tags:  
      - neutron.networks.vlan:  
          provider-physical-network: physnet1  
  
  - name: NET_GROUP_B  
    tags:  
      - neutron.networks.vlan:  
          provider-physical-network: physnet2
```

A network-group is associated with a server network interface via an interface-model as described in the section called "Interface Models". For example (some attributes omitted for brevity):

```
interface-models:  
  - name: INTERFACE_SET_X  
    network-interfaces:  
      - device:  
          name: bond0
```

```
network-groups:  
  - NET_GROUP_A  
- device:  
  name: hed3  
network-groups:  
  - NET_GROUP_B
```

A network-group used for provider VLANs may contain only a single HPE Helion OpenStack network, because that VLAN must span all compute nodes and any Neutron network nodes/controllers (i.e. it is a single L2 segment). The HPE Helion OpenStack network must be defined with `tagged-vlan: false`, otherwise a linux VLAN network interface will be created. For example:

```
networks:  
  - name: NET_A  
    tagged-vlan: false  
    network-group: NET_GROUP_A  
  - name: NET_B  
    tagged-vlan: false  
    network-group: NET_GROUP_B
```

When the cloud is deployed, HPE Helion OpenStack 5.0 will create the appropriate bridges on the servers, and set the appropriate attributes in the Neutron configuration files (e.g. `bridge_mappings`).

After the cloud has been deployed, create Neutron network objects for each provider VLAN using the Neutron CLI:

```
neutron net-create --provider:network_type vlan --provider:physical_network physnet1  
neutron net-create --provider:network_type vlan --provider:physical_network physnet2
```

## Standalone Lifecycle Manager

All the examples use a “deployer-in-the-cloud” scenario where the first controller is also the deployer/life-cycle manager. If you want to use a standalone lifecycle manager, you will need to add the relevant details in `control_plane.yml`, `servers.yml` and related configuration files.

### **control\_plane.yml**

```
control-planes:  
- clusters:  
  - allocation-policy: strict  
    cluster-prefix: c0  
    member-count: 1  
    name: c0  
    server-role: DEPLOYER-ROLE  
    service-components:  
      - lifecycle-manager  
      - ntp-server
```

### **servers.yml**

```
servers:
```

```
- id: deployer
 ilo-ip: 10.1.8.73
 ilo-password: mul3d33r
 ilo-user: hosqaadm
  ip-addr: 10.240.20.21
  is-deployer: true
  mac-addr: 8c:dc:d4:b5:ce:18
  nic-mapping: HP-DL360-4PORT
role: DEPLOYER-ROLE
  server-group: RACK1
```

**server\_roles.yml**

```
server-roles:
- disk-model: DEPLOYER-600GB-DISKS
  interface-model: DEPLOYER-INTERFACES
  name: DEPLOYER-ROLE
```

**disks\_deployer\_600GB.yml:**

```
disk-models:
- device-groups:
  - consumer:
    attrs:
      rings:
        - account
        - container
        - object-0
      name: swift
    devices:
      - name: /dev/sdc
      - name: /dev/sdd
      - name: /dev/sde
      name: swiftobj
name: DEPLOYER-600GB-DISKS
volume-groups:
- consumer:
  name: os
logical-volumes:
- fstype: ext4
  mount: /
  name: root
  size: 6%
...
...
```

---

# Chapter 9. Configuration Processor Information Files

In addition to producing all of the data needed to deploy and configure the cloud, the configuration processor also creates a number of information files that provide details of the resulting configuration.

These files can be found in `~/helion/my_cloud/info` after the first configuration processor run. This directory is also rebuilt each time the Configuration Processor is run.

Most of the files are in YAML format, allowing them to be used in further automation tasks if required.

File	Provides details of
<code>address_info.yml</code>	IP address assignments on each network. See the section called “ <code>address_info.yml</code> ”
<code>firewall_info.yml</code>	All ports that are open on each network by the firewall configuration. Can be used if you want to configure an additional firewall in front of the API network, for example. See the section called “ <code>firewall_info.yml</code> ”
<code>net_info.yml</code>	IP addresses assigned to services. For example, this provides the data needed to complete the configuration of VSA clusters. See the section called “ <code>net_info.yml</code> ”
<code>route_info.yml</code>	Routes that need to be configured between networks. See the section called “ <code>route_info.yml</code> ”
<code>server_info.yml</code>	How servers have been allocated, including their network configuration. Allows details of a server to be found from its ID. See the section called “ <code>server_info.yml</code> ”
<code>service_info.yml</code>	Details of where components of each service are deployed. See the section called “ <code>service_info.yml</code> ”
<code>control_plane_topology.yml</code>	Details the structure of the cloud from the perspective of each control-plane. See the section called “ <code>control_plane_topology.yml</code> ”
<code>network_topology.yml</code>	Details the structure of the cloud from the perspective of each control-plane. See the section called “ <code>network_topology.yml</code> ”
<code>region_topology.yml</code>	Details the structure of the cloud from the perspective of each region. See the section called “ <code>region_topology.yml</code> ”
<code>service_topology.yml</code>	Details the structure of the cloud from the perspective of each service. See the section called “ <code>service_topology.yml</code> ”
<code>private_data_metadata.yml</code>	Details the secrets that are generated by the configuration processor – the names of the secrets, along with the service(s) that use each secret and a list of the clusters on which the service that consumes

<b>File</b>	<b>Provides details of</b>
	the secret is deployed. See the section called “private_data_metadata.yml”
password_change.yml	Details the secrets that have been changed by the configuration processor – information for each secret is the same as for private_data_metadata.yml. See the section called “password_change.yml”
explain.txt	An explanation of the decisions the configuration processor has made when allocating servers and networks. See the section called “explain.txt”
CloudDiagram.txt	A pictorial representation of the cloud. See the section called “CloudDiagram.txt”

The examples are taken from the entry-scale-kvm-vsa example configuration.

## address\_info.yml

This file provides details of all the IP addresses allocated by the Configuration Processor:

```
<Network Groups>
  <List of Networks>
    <IP Address>
      <List of Aliases>
```

Example:

```
EXTERNAL-API:
EXTERNAL-API-NET:
  10.0.1.2:
    - helion-cp1-c1-m1-extapi
  10.0.1.3:
    - helion-cp1-c1-m2-extapi
  10.0.1.4:
    - helion-cp1-c1-m3-extapi
  10.0.1.5:
    - helion-cp1-vip-public-SWF-PRX-extapi
    - helion-cp1-vip-public-FRE-API-extapi
    - helion-cp1-vip-public-GLA-API-extapi
    - helion-cp1-vip-public-HEA-ACW-extapi
    - helion-cp1-vip-public-HEA-ACF-extapi
    - helion-cp1-vip-public-NEU-SVR-extapi
    - helion-cp1-vip-public-KEY-API-extapi
    - helion-cp1-vip-public-MON-API-extapi
    - helion-cp1-vip-public-HEA-API-extapi
    - helion-cp1-vip-public-NOV-API-extapi
    - helion-cp1-vip-public-CND-API-extapi
    - helion-cp1-vip-public-CEI-API-extapi
    - helion-cp1-vip-public-SHP-API-extapi
    - helion-cp1-vip-public-OPS-WEB-extapi
```

```
- helion-cpl-vip-public-HZN-WEB-extapi
- helion-cpl-vip-public-NOV-VNC-extapi
EXTERNAL-VM:
  EXTERNAL-VM-NET: {}
GUEST:
  GUEST-NET:
    10.1.1.2:
      - helion-cpl-c1-m1-guest
    10.1.1.3:
      - helion-cpl-c1-m2-guest
    10.1.1.4:
      - helion-cpl-c1-m3-guest
    10.1.1.5:
      - helion-cpl-comp0001-guest
MANAGEMENT:
  ...
  ...
```

## firewall\_info.yml

This file provides details of all the network ports that will be opened on the deployed cloud. Data is ordered by network. If you want to configure an external firewall in front of the External API network, then you would need to open the ports listed in that section.

```
<Network Name>
  List of:
    <Port>
    <Protocol>
    <List of IP Addresses>
    <List of Components>
```

Example:

```
EXTERNAL-API:
- addresses:
  - 10.0.1.5
  components:
  - horizon
  port: '443'
  protocol: tcp
- addresses:
  - 10.0.1.5
  components:
  - keystone-api
  port: '5000'
  protocol: tcp
```

*Port 443 (tcp) is open on network EXTERNAL-API for address 10.0.1.5 because it is used by Horizon*

*Port 5000 (tcp) is open on network EXTERNAL-API for address 10.0.1.5 because it is used by Keystone API*

## net\_info.yml

This file provides details of IP addresses that have been allocated for a service. This data is typically used for service configuration after the initial deployment.

```
service_ips:  
  <Service-Name>  
    control_plane:  <Control Plane Name>  
    cluster: <Cluster or Resource Name>  
    network: <Network Name>  
    cluster_ip:  
      hostname: <Hostname alias of address allocated for the cluster>  
      ip_address: <IP address allocated for the cluster>  
    hosts: (list)  
      hostname: <Hostname of server in the cluster>  
      ip_address: <IP address of server in the cluster>
```

Example:

```
service_ips:  
  vsa:  
    -   cluster: vsa  
        cluster_ip:  
          hostname: helion-cpl-vsa-VSA-BLK-mgmt  
          ip_address: 192.168.10.7  
        control_plane: control-plane-1  
        hosts:  
          -   hostname: helion-cpl-vsa0001-VSA-BLK-mgmt  
              ip_address: 192.168.10.2  
          -   hostname: helion-cpl-vsa0002-VSA-BLK-mgmt  
              ip_address: 192.168.10.8  
          -   hostname: helion-cpl-vsa0003-VSA-BLK-mgmt  
              ip_address: 192.168.10.12  
        network: MANAGEMENT-NET
```

*Resource group "vsa" in "control-plane-1" has been allocated 192.168.10.7 on network MANAGEMENT-NET as a cluster address and consists of 3 servers with addresses 192.168.10.2, 192.168.10.8, and 192.168.10.12.*

## route\_info.yml

This file provides details of routes between networks that need to be configured. Available routes are defined in the input model as part of the network-groups data; this file shows which routes will actually be used. HPE Helion OpenStack will reconfigure routing rules on the servers, you must configure the corresponding routes within your physical network. Routes must be configured to be symmetrical -- only the direction in which a connection is initiated is captured in this file.

Note that simple models may not require any routes, with all servers being attached to common L3 networks. The following example is taken from the tech-preview/mid-scale-kvm-vsa example.

```
<Source-Network-Name>
```

```
<Target-Network-Name>
  default: <true if this is this the result of a "default" route rule>
  used_by:
    <source-service>
      <target-service>
        <list of hosts using this route>
```

Example:

```
MANAGEMENT-NET-RACK1:
  INTERNAL-API-NET:
    default: false
    used_by:
      ceilometer-client:
      ceilometer-api:
        - helion-cp1-mtrmon-m1
      keystone-api:
        - helion-cp1-mtrmon-m1
MANAGEMENT-NET-RACK2:
  default: false
  used_by:
    cinder-backup:
    rabbitmq:
      - helion-cp1-core-m1
```

A route is required from network **MANAGEMENT-NET-RACK1** to network **INTERNAL-API-NET** so that **ceilometer-client** can connect to **ceilometer-api** from server **helion-cp1-mtrmon-m1** and to **keystone-api** from the same server.

A route is required from network **MANAGEMENT-NET-RACK1** to network **MANAGEMENT-NET-RACK2** so that **cinder-backup** can connect to **rabbitmq** from server **helion-cp1-core-m1**

## **server\_info.yml**

This file provides details of how servers have been allocated by the Configuration Processor. This provides the easiest way to find where a specific physical server (identified by **server-id**) is being used.

```
<Server-id>
  failure-zone: <failure zone that the server was allocated from>
  hostname: <hostname of the server>
  net_data: <network configuration>
  state: < "allocated" | "available" >
```

Example:

```
controller1:
  failure-zone: AZ1
  hostname: helion-cp1-c1-m1-mgmt
  net_data:
    BOND0:
      EXTERNAL-API-NET:
        addr: 10.0.1.2
```

```
        tagged-vlan: true
        vlan-id: 101
    EXTERNAL-VM-NET:
        addr: null
        tagged-vlan: true
        vlan-id: 102
    GUEST-NET:
        addr: 10.1.1.2
        tagged-vlan: true
        vlan-id: 103
    MANAGEMENT-NET:
        addr: 192.168.10.3
        tagged-vlan: false
        vlan-id: 100
    state: allocated
```

## service\_info.yml

This file provides details of how services are distributed across the cloud.

```
#60;control-plane>
<service>
    <service component>
        <list of hosts>
```

Example:

```
control-plane-1:
    neutron:
        neutron-client:
            - helion-cpl-c1-m1-mgmt
            - helion-cpl-c1-m2-mgmt
            - helion-cpl-c1-m3-mgmt
        neutron-dhcp-agent:
            - helion-cpl-c1-m1-mgmt
            - helion-cpl-c1-m2-mgmt
            - helion-cpl-c1-m3-mgmt
        neutron-l3-agent:
            - helion-cpl-comp0001-mgmt
        neutron-lbaasv2-agent:
            - helion-cpl-comp0001-mgmt
    ...
    ...
```

## control\_plane\_topology.yml

This file provides details of the topology of the cloud from the perspective of each control plane:

```
control_planes:
    <control-plane-name>
        load-balancers:
            <load-balancer-name>:
```

```
address: <IP address of VIP>
cert-file: <name of cert file>
external-name: <name to used for endpoints>
network: <name of the network this LB is connected to>
network_group: <name of the network group this LB is connect to
provider: <service component providing the LB>
roles: <list of roles of this LB>
services:
    <service-name>:
        <component-name>:
            aliases:
                <role>: <Name in /etc/hosts>
            host-tls: <Boolean, true if connection from LB uses TLS>
            hosts: <List of hosts for this service>
            port: <port used for this component>
            vip-tls: <Boolean, true if the VIP terminates TLS>
clusters:
    <cluster-name>
        failure-zones:
            <failure-zone-name>:
                <list of hosts>
        services:
            <service name>:
                components:
                    <list of service components>
            regions:
                <list of region names>
resources:
    <resource-name>:
        <as for clusters above>
```

**Example:**

```
control_planes:
control-plane-1:
    clusters:
        cluster1:
            failure_zones:
                AZ1:
                    - helion-cpl-c1-m1-mgmt
                AZ2:
                    - helion-cpl-c1-m2-mgmt
                AZ3:
                    - helion-cpl-c1-m3-mgmt
            services:
                barbican:
                    components:
                        - barbican-api
                        - barbican-worker
                    regions:
                        - region1
```

```
load-balancers:  
    extlb:  
        address: 10.0.1.5  
        cert-file: my-public-entry-scale-kvm-vsa-cert  
        external-name: ''  
        network: EXTERNAL-API-NET  
        network-group: EXTERNAL-API  
        provider: ip-cluster  
        roles:  
        - public  
        services:  
            barbican:  
                barbican-api:  
                    aliases:  
                        public: helion-cpl-vip-public-KEYMGR-API-extapi  
                    host-tls: true  
                    hosts:  
                    - helion-cpl-c1-m1-mgmt  
                    - helion-cpl-c1-m2-mgmt  
                    - helion-cpl-c1-m3-mgmt  
                    port: '9311'  
                    vip-tls: true
```

## network\_topology.yml

This file provides details of the topology of the cloud from the perspective of each network\_group:

```
network-groups:  
    <network-group-name>:  
        <network-name>:  
            control-planes:  
                <control-plane-name>:  
                    clusters:  
                        <cluster-name>:  
                            servers:  
                                <hlm-server-name>: <ip address>  
                            vips:  
                                <ip address>: <load balancer name>  
                    resources:  
                        <resource-group-name>:  
                            servers:  
                                <hlm-server-name>: <ip address>
```

### Example:

```
network_groups:  
    EXTERNAL-API:  
        EXTERNAL-API-NET:  
            control_planes:  
                control-plane-1:
```

```
clusters:
  cluster1:
    servers:
      helion-cpl-c1-m1: 10.0.1.2
      helion-cpl-c1-m2: 10.0.1.3
      helion-cpl-c1-m3: 10.0.1.4
    vips:
      10.0.1.5: extlb

EXTERNAL-VM:
EXTERNAL-VM-NET:
  control_planes:
    control-plane-1:
      clusters:
        cluster1:
          servers:
            helion-cpl-c1-m1: null
            helion-cpl-c1-m2: null
            helion-cpl-c1-m3: null
      resources:
        compute:
          servers:
            helion-cpl-comp0001: null
```

## region\_topology.yml

This file provides details of the topology of the cloud from the perspective of each region:

```
regions:
<region-name>:
  control_planes:
    <control-plane-name>:
      services:
        <service-name>:
          <list of service components>
```

**Example:**

```
regions:
  region1:
    control_planes:
      control-plane-1:
        services:
          barbican:
            - barbican-api
            - barbican-worker
          ceilometer:
            - ceilometer-common
            - ceilometer-agent-notification
            - ceilometer-api
            - ceilometer-polling
          cinder:
            - cinder-api
```

- cinder-volume
- cinder-scheduler
- cinder-backup

## service\_topology.yml

This file provides details of the topology of the cloud from the perspective of each service:

```
services:  
  <service-name>:  
    components:  
      <component-name>:  
        control-planes:  
          <control-plane-name>:  
            clusters:  
              <cluster-name>:  
                <list of servers>  
            resources:  
              <resource-group-name>:  
                <list of servers>  
            regions:  
              <list of regions>
```

**Example:**

```
services:  
  freezer:  
    components:  
      freezer-agent:  
        control_planes:  
          control-plane-1:  
            clusters:  
              cluster1:  
                - helion-cp1-c1-m1-mgmt  
                - helion-cp1-c1-m2-mgmt  
                - helion-cp1-c1-m3-mgmt  
            regions:  
              - region1  
            resources:  
              compute:  
                - helion-cp1-comp0001-mgmt  
              vsa:  
                - helion-cp1-vsa0001-mgmt  
                - helion-cp1-vsa0002-mgmt  
                - helion-cp1-vsa0003-mgmt  
            regions:  
              - region1
```

## private\_data\_metadata.yml

This file provide details of the secrets that are generated by the configuration processor. The details include:

- The names of each secret
- Metadata about each secret. This is a list where each element contains details about each component service that uses the secret.
  - The component service that uses the secret, and if applicable the service that this component "consumes" when using the secret
  - The list of clusters on which the component service is deployed
  - The control plane cp on which the services are deployed
- A version number (the model version number)

```
<secret>
  <metadata>
    <list of metadata>
      <clusters>
        <list of clusters>
      <component>
      <consumes>
      <control-plane>
    <version>
```

For example:

```
barbican_admin_password:
  metadata:
    - clusters:
      - cluster1
    component: barbican-api
    cp: ccp
  version: '2.0'
keystone_swift_password:
  metadata:
    - clusters:
      - cluster1
    component: swift-proxy
    consumes: keystone-api
    cp: ccp
  version: '2.0'
metadata_proxy_shared_secret:
  metadata:
    - clusters:
      - cluster1
    component: nova-metadata
    cp: ccp
    - clusters:
      - cluster1
      - compute
    component: neutron-metadata-agent
    cp: ccp
  version: '2.0'
```

...

## **password\_change.yml**

This file provides details equivalent to those in private\_data\_metadata.yml for passwords which have been changed from their original values, using the procedure outlined in the HPE Helion OpenStack documentation

## **explain.txt**

This file provides details of the server allocation and network configuration decisions the configuration processor has made. The sequence of information recorded is:

- Any service components that are automatically added
- Allocation of servers to clusters and resource groups
- Resolution of the network configuration for each server
- Resolution of the network configuration of each load balancer

Example:

```
Add required services to control plane control-plane-1
=====
control-plane-1: Added nova-metadata required by nova-api
control-plane-1: Added swift-common required by swift-proxy
control-plane-1: Added swift-rsync required by swift-account

Allocate Servers for control plane control-plane-1
=====

cluster: cluster1
-----
Persisted allocation for server 'controller1' (AZ1)
Persisted allocation for server 'controller2' (AZ2)
Searching for server with role ['CONTROLLER-ROLE'] in zones: set(['AZ3'])
Allocated server 'controller3' (AZ3)

resource: vsa
-----
Persisted allocation for server 'vsal' (AZ1)
Persisted allocation for server 'vsal2' (AZ2)
Persisted allocation for server 'vsal3' (AZ3)
Searching for server with role ['VSA-ROLE'] in zones: set(['AZ1', 'AZ2'])

resource: compute
-----
Persisted allocation for server 'computel' (AZ1)
Searching for server with role ['COMPUTE-ROLE'] in zones: set(['AZ1', 'AZ2'])

Resolve Networks for Servers
=====
```

```
server: helion-cpl-c1-m1
-----
    add EXTERNAL-API for component ip-cluster
    add MANAGEMENT for component ip-cluster
    add MANAGEMENT for lifecycle-manager (default)
    add MANAGEMENT for ntp-server (default)
    ...
    add MANAGEMENT for swift-rsync (default)
    add GUEST for tag neutron.networks.vxlan (neutron-openvswitch-agent)
    add EXTERNAL-VM for tag neutron.13_agent.external_network_bridge (neutron)
Using persisted address 10.0.1.2 for server helion-cpl-c1-m1 on network
Using address 192.168.10.3 for server helion-cpl-c1-m1 on network MANAGEMENT
Using persisted address 10.1.1.2 for server helion-cpl-c1-m1 on network

...
Define load balancers
=====

Load balancer: extlb
-----
    Using persisted address 10.0.1.5 for vip extlb helion-cpl-vip-extlb-exta
    Add nova-api for roles ['public'] due to 'default'
    Add glance-api for roles ['public'] due to 'default'
    ...

Map load balancers to providers
=====

Network EXTERNAL-API-NET
-----
10.0.1.5: ip-cluster nova-api roles: ['public'] vip-port: 8774 host-port:
10.0.1.5: ip-cluster glance-api roles: ['public'] vip-port: 9292 host-port:
10.0.1.5: ip-cluster keystone-api roles: ['public'] vip-port: 5000 host-port:
10.0.1.5: ip-cluster swift-proxy roles: ['public'] vip-port: 8080 host-port:
10.0.1.5: ip-cluster monasca-api roles: ['public'] vip-port: 8070 host-port:
10.0.1.5: ip-cluster heat-api-cfn roles: ['public'] vip-port: 8000 host-port:
10.0.1.5: ip-cluster ops-console-web roles: ['public'] vip-port: 9095 host-port:
10.0.1.5: ip-cluster heat-api roles: ['public'] vip-port: 8004 host-port:
10.0.1.5: ip-cluster nova-novncproxy roles: ['public'] vip-port: 6080 host-port:
10.0.1.5: ip-cluster neutron-server roles: ['public'] vip-port: 9696 host-port:
10.0.1.5: ip-cluster heat-api-cloudwatch roles: ['public'] vip-port: 8000 host-port:
10.0.1.5: ip-cluster ceilometer-api roles: ['public'] vip-port: 8777 host-port:
10.0.1.5: ip-cluster freezer-api roles: ['public'] vip-port: 9090 host-port:
10.0.1.5: ip-cluster horizon roles: ['public'] vip-port: 443 host-port:
10.0.1.5: ip-cluster cinder-api roles: ['public'] vip-port: 8776 host-port:
```

## CloudDiagram.txt

This file provides a pictorial representation of the cloud. Although this file is still produced, it is superseded by the HTML output described in the following section.

Example:

## Configuration Processor Information Files

---

```
+-ControlPlane: region1 (control-plane-1)
|   +-Cluster cluster1 ()
|       +-helion-cp1-cl-m1 (192.168.10.3)-----+      +-helion-cp1-cl-m2
|           ceilometer
|           ceilometer-agent-central
|           ceilometer-agent-notification
|           ceilometer-api
|           ceilometer-client
|           ceilometer-collector
|           ceilometer-common
|           ceilometer-expirer
|           cinder
|           cinder-api
|           cinder-backup
|           cinder-client
|           cinder-scheduler
|           cinder-volume
|           foundation
|           apache2
|           ip-cluster
|           kafka
|           memcached
|           mysql
|           ntp-server
|           openstack-client
|           rabbitmq
|           storm
|           stunnel
|           swift-common
|           swift-rsync
|           vertica
|           zookeeper
|           freezer
|           freezer-agent
|           freezer-api
|           glance
|           glance-api
|           glance-client
|           glance-registry
|           heat
|           heat-api
|           heat-api-cfn
|           heat-api-cloudwatch
|           heat-client
|           heat-engine
|           horizon
|           horizon
|           keystone
|           keystone-api
|           keystone-client
|           logging
|
|           ceilometer
|           ceilometer-agent
|           ceilometer-agent
|           ceilometer-api
|           ceilometer-api
|           ceilometer-client
|           ceilometer-client
|           ceilometer-common
|           ceilometer-common
|           ceilometer-expirer
|           cinder
|           cinder-api
|           cinder-backup
|           cinder-client
|           cinder-scheduler
|           cinder-volume
|           foundation
|           apache2
|           ip-cluster
|           kafka
|           memcached
|           mysql
|           ntp-server
|           openstack-client
|           rabbitmq
|           storm
|           stunnel
|           swift-common
|           swift-rsync
|           vertica
|           zookeeper
|           freezer
|           freezer-agent
|           freezer-api
|           glance
|           glance-api
|           glance-client
|           glance-registry
|           heat
|           heat-api
|           heat-api-cfn
|           heat-api-cloud
|           heat-client
|           heat-engine
|           horizon
|           horizon
|           keystone
|           keystone-api
|           keystone-client
|           logging
```

Configuration Processor Information Files

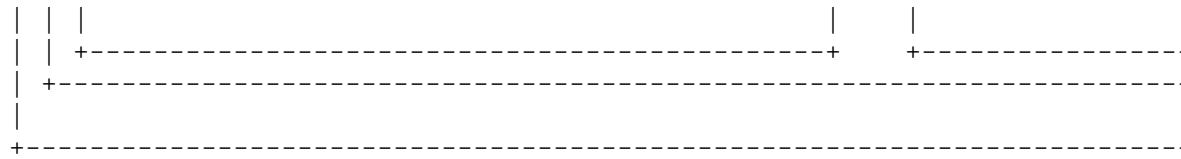
---

<pre>             logging-producer             logging-server         monasca             monasca-agent             monasca-api             monasca-client             monasca-notifier             monasca-persistor             monasca-threshold         neutron             neutron-client             neutron-dhcp-agent             neutron-metadata-agent             neutron-ml2-plugin             neutron-openvswitch-agent             neutron-server             neutron-vpn-agent         nova             nova-api             nova-client             nova-conductor             nova-console-auth             nova-metadata             nova-novncproxy             nova-scheduler         operations             lifecycle-manager             lifecycle-manager-target             ops-console-monitor             ops-console-web         swift             swift-account             swift-client             swift-container             swift-object             swift-proxy             swift-ring-builder         vsa-storage             cmc-service           +-----+     bond0 (hed3, hed4)     EXTERNAL-API-NET (10.0.1.2)     EXTERNAL-VM-NET     GUEST-NET (10.1.1.2)     MANAGEMENT-NET (192.168.10.3)           +-----+     +-compute-----+       </pre>	<pre>             logging-producer             logging-server         monasca             monasca-agent             monasca-api             monasca-client             monasca-notifi             monasca-persis             monasca-thresh         neutron             neutron-client             neutron-dhcp-a             neutron-metada             neutron-ml2-pl             neutron-openvs             neutron-server             neutron-vpn-ag         nova             nova-api             nova-client             nova-conducto             nova-console-a             nova-metadata             nova-novncprox             nova-scheduler         operations             lifecycle-mana             lifecycle-mana             ops-console-mo             ops-console-we         swift             swift-account             swift-client             swift-containe             swift-object             swift-proxy             swift-ring-bui         vsa-storage             cmc-service           +-----+     bond0 (hed3, hed     EXTERNAL-API-N     EXTERNAL-VM-NE     GUEST-NET (10.     MANAGEMENT-NET </pre>
--	--

## Configuration Processor Information Files

---

```
+--COMPUTE-ROLE (AZ1) (1 servers)-----+
| foundation
|   ntp-client
|   stunnel
|   freezer
|     freezer-agent
|   logging
|     logging-producer
|   monasca
|     monasca-agent
|   neutron
|     neutron-l3-agent
|     neutron-lbaasv2-agent
|     neutron-metadata-agent
|     neutron-openvswitch-agent
|   nova
|     nova-compute
|     nova-compute-kvm
|   operations
|     lifecycle-manager-target
|
+-----+
| bond0 (hed3, hed4)
|   EXTERNAL-VM-NET
|   GUEST-NET (10.1.1.0/24)
|   MANAGEMENT-NET (192.168.10.0/24)
+-----+
+--vsa-----+
++VSA-ROLE (AZ1) (1 servers)-----+    ++VSA-ROLE (AZ2) (1 servers)-----+
| foundation
|   ntp-client
|   stunnel
|   freezer
|     freezer-agent
|   logging
|     logging-producer
|   monasca
|     monasca-agent
|   operations
|     lifecycle-manager-target
|   vsa-storage
|   vsa
|
+-----+
| bond0 (hed3, hed4)
|   MANAGEMENT-NET (192.168.10.0/24)
+-----+
| bond0 (hed3, hed4)
|   MANAGEMENT-NET (192.168.10.0/24)
```



# HTML Representation

An HTML representation of the cloud can be found in `~/helion/my_cloud/html` after the first Configuration Processor run. This directory is also rebuilt each time the Configuration Processor is run. These files combine the data in the input model with allocation decisions made by the Configuration processor to allow the configured cloud to be viewed from a number of different perspectives.

Most of the entries on the HTML pages provide either links to other parts of the HTML output or additional details via hover text.

## Cloud: entry-scale-kvm-vsa

[Control Plane View](#)   [Region View](#)   [Service View](#)   [Network View](#)   [Server View](#)   [Server Groups View](#)

### control-plane-1

Clusters	Resources	Load Balancers
cluster1	vsa	extlb
barbican ceilometer cinder designate freezer glance heat horizon keystone logging monasca neutron nova octavia operations swift tempest vsa-storage  foundation clients hlm	compute  freezer  logging monasca neutron nova  vsa-storage  foundation hlm	barbican ceilometer cinder designate freezer glance heat horizon keystone logging monasca neutron nova  operations swift  foundation hlm
AZ1	<a href="#">helion-cpl-c1-m1-mgmt</a>	<a href="#">helion-cpl-vsa0001-mgmt</a>
AZ2	<a href="#">helion-cpl-c1-m2-mgmt</a>	<a href="#">helion-cpl-vsa0002-mgmt</a>
AZ3	<a href="#">helion-cpl-c1-m3-mgmt</a>	<a href="#">helion-cpl-vsa0003-mgmt</a>
		<a href="#">10.0.1.5</a>
		<a href="#">192.168.10.13</a>

## Cloud: entry-scale-kvm-vsa

[Control Plane View](#)   [Region View](#)   [Service View](#)   [Network View](#)   [Server View](#)   [Server Groups View](#)

### Network Topology

control-plane-1				
Clusters	Resources			
cluster1	vsa	compute	lb	extlb
EXTERNAL-API	EXTERNAL-API-NET			
MANAGEMENT	MANAGEMENT-NET	MANAGEMENT-NET	MANAGEMENT-NET	
GUEST	GUEST-NET		GUEST-NET	
EXTERNAL-VM	EXTERNAL-VM-NET		EXTERNAL-VM-NET	
OCTAVIA-MGMT-NET				

### Network Groups

EXTERNAL-API

Network Group	Networks	Address	Server	Interface Model
Components: powerdns-ext Load Balancers: extlb control-plane-1	EXTERNAL-API-NET vlan id: 10 (tagged) cidr: 10.0.10.24 gateway-ip: 10.0.1.1 mtu: 1500	10.0.1.4 10.0.1.3 10.0.1.2 10.0.1.5	<a href="#">helion-cpl-c1-m3</a> <a href="#">helion-cpl-c1-m2</a> <a href="#">helion-cpl-c1-m1</a> <a href="#">extlb</a>	<a href="#">CONTROLLER-INTERFACES</a>

---

# Chapter 10. Example Configurations

The HPE Helion OpenStack 5.0 system ships with a collection of pre-qualified example configurations. These are designed to help you to get up and running quickly with a minimum number of configuration changes.

The HPE Helion OpenStack input model allows a wide variety of configuration parameters that may, at first glance, appear daunting. The example configurations are designed to simplify this process by providing pre-built and pre-qualified examples that need only a minimum number of modifications to get started.

## HPE Helion OpenStack Example Configurations

This section briefly describes the various example configurations and their capabilities. It also describes in detail, for the entry-scale-kvm-vsa example, how you can adapt the input model to work in your environment.

The following pre-qualified examples are shipped with HPE Helion OpenStack 5.0:

Name	Location
the section called “Entry-scale KVM with VSA Model”	<code>~/helion/examples/entry-scale-kvm-vsa</code>
the section called “Entry-scale KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging”	<code>~/helion/examples/entry-scale-kvm-vsa-mml</code>
the section called “Entry-scale KVM with Ceph Model”	<code>~/helion/examples/entry-scale-kvm-ceph</code>
the section called “Mid-scale KVM with VSA Model”	<code>~/helion/examples/mid-scale-kvm-vsa</code>
the section called “Entry-scale ESX, KVM with VSA Model”	<code>~/helion/examples/entry-scale-esx-kvm-vsa</code>
the section called “Entry-scale ESX, KVM with VSA Model with Dedicated Cluster for Metering, Monitoring, and Logging”	<code>~/helion/examples/entry-scale-esx-kvm-vsa-mml</code>
the section called “Entry-scale Swift Model”	<code>~/helion/examples/entry-scale-swift</code>
the section called “Entry-scale Cloud with Ironic Flat Network”	<code>~/helion/examples/entry-scale-ironic-flat-network</code>
the section called “Entry-scale Cloud with Ironic Multi-Tenancy”	<code>~/helion/examples/entry-scale-ironic-multi-tenancy</code>

The entry-scale systems are designed to provide an entry-level solution that can be scaled from a small number of nodes to a moderately high node count (approximately 100 compute nodes, for example).

In the mid-scale model, the cloud control plane is subdivided into a number of dedicated service clusters to provide more processing power for individual control plane elements. This enables a greater number of

resources to be supported (compute nodes, Swift object servers). This model also shows how a segmented network can be expressed in the HPE Helion OpenStack model.

## Modifying the Entry-scale KVM with VSA Model for Your Environment

This section covers the changes that need to be made to the input model to deploy and run this cloud model in your environment.

- the section called “Localizing the Input Model”
- the section called “Customizing the Input Model”

## Alternative Configurations

In HPE Helion OpenStack 5.0 there are alternative configurations that we recommend for specific purposes and this section we will outline them.

- the section called “Entry-scale KVM with Ceph Model”
- ???
- the section called “Using a Dedicated Lifecycle Manager Node”
- the section called “Configuring HPE Helion OpenStack without DVR”
- the section called “Configuring HPE Helion OpenStack with Provider VLANs and Physical Routers Only”
- the section called “Considerations When Installing Two Systems on One Subnet”

## KVM Examples

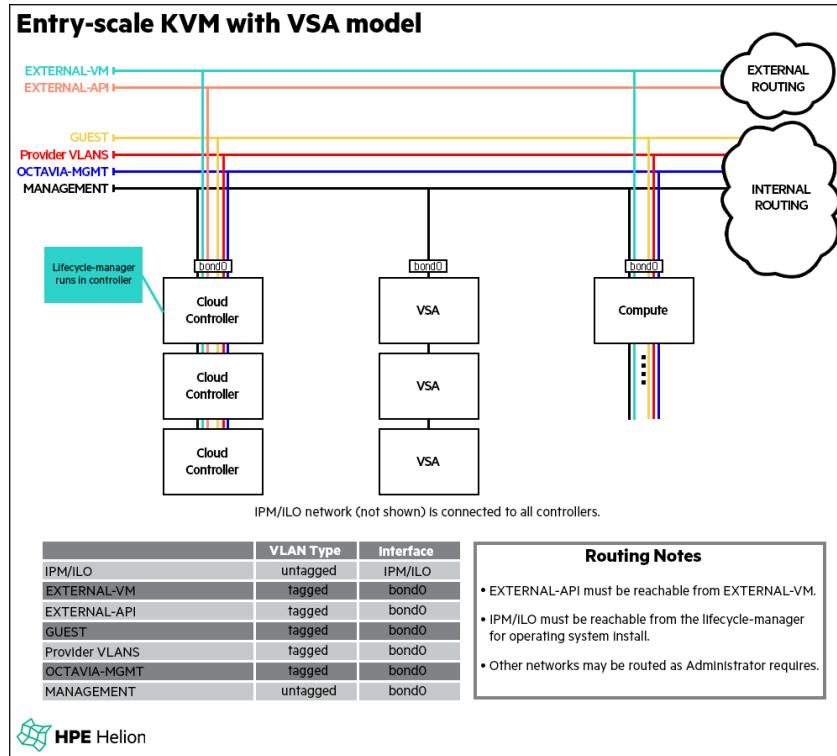
### Entry-scale KVM with VSA Model

This model provides a KVM-based cloud with VSA for volume storage, and has been tested to a scale of 100 compute nodes.

The example is focused on the minimum server count to support a highly-available (HA) compute cloud deployment. The first (manually installed) server, often referred to as the deployer or lifecycle manager, is also used as one of the controller nodes. This model consists of a minimum server count of seven, with three controllers, three VSA storage servers, and one compute server. Swift storage in this example is contained on the controllers.

Note that the VSA storage requires a minimum of three servers for a HA configuration, although the deployment will work with as little as one VSA node.

This model can also be deployed without the VSA servers and configured to use an external storage device, such as a 3PAR array, which would reduce the minimum server count to four.



Download [../../../../media/entryScale/Entry-ScaleAllNetworks.png]

Download Editable Visio Network Diagram Template [../../../../media/templates/HOS\_Network\_Diagram\_Template.zip]

The example requires the following networks:

- **External API** - This is the network that users will use to make requests to the cloud.
- **External VM** - This is the network that will be used to provide access to virtual machines (via floating IP addresses).
- **Guest/VxLAN** - This is the network that will carry traffic between virtual machines on private networks within the cloud.
- **Octavia Management** - This is the network that will be used for the Octavia load balancing service.
- **Management** - This is the network that will be used for all internal traffic between the cloud services, including node provisioning. This network must be on an untagged VLAN.

All of these networks are configured to be presented via a pair of bonded NICs. The example also enables additional provider VLANs to be configured in Neutron on this interface.

In the diagram, "External Routing" refers to whatever routing you want to provide to allow users to access the External API and External VM networks. Note that the EXTERNAL\_API network must be reachable from the EXTERNAL\_VM network if you want virtual machines to be able to make API calls to the cloud. "Internal Routing" refers to whatever routing you want to provide to allow administrators to access the Management network.

If you are using HPE Helion OpenStack to install the operating system, then an IPMI/iLO network connected to the IPMI/iLO ports of all servers and routable from the lifecycle manager server is also required for BIOS and power management of the nodes during the operating system installation process.

The example uses the following disk configurations:

- **Controllers** - One operating system disk and two disks for Swift storage.
- **VSA** - One operating system disk and two disks for VSA storage.
- **Compute** - One operating system disk and one disk for virtual machine ephemeral storage.

For details about how to modify this example to match your environment, see Chapter 11, *Modifying the Entry-scale KVM with VSA Model for Your Environment*.

These recommended minimums are based on the included included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

## Note

The disk requirements detailed below can be met with logical drives, logical volumes, or external storage such as a 3PAR array.

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"><li>• 1 x 600 GB (minimum) - operating system drive</li><li>• 2 x 600 GB (minimum) - Data drive</li></ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
Compute	Compute	1-3	2 X 600 GB (minimum)	32 GB (memory must be sized based on the virtual machine instances hosted on the Compute node)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64) with hardware virtualization support. The CPU cores must be sized based on the VM instances hosted by the Compute node.

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations				
			Disk	Memory	Network	CPU	
Block Storage (Optional)	VSA or OSD (Ceph)	0 or 3 (which will provide the recommended redundancy)	3 X 600 GB (minimum) See Chapter 2, <i>Pre-Installation Checklist</i> for more details.	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)	

For more details about the supported network requirements, see .

## Entry-scale KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging

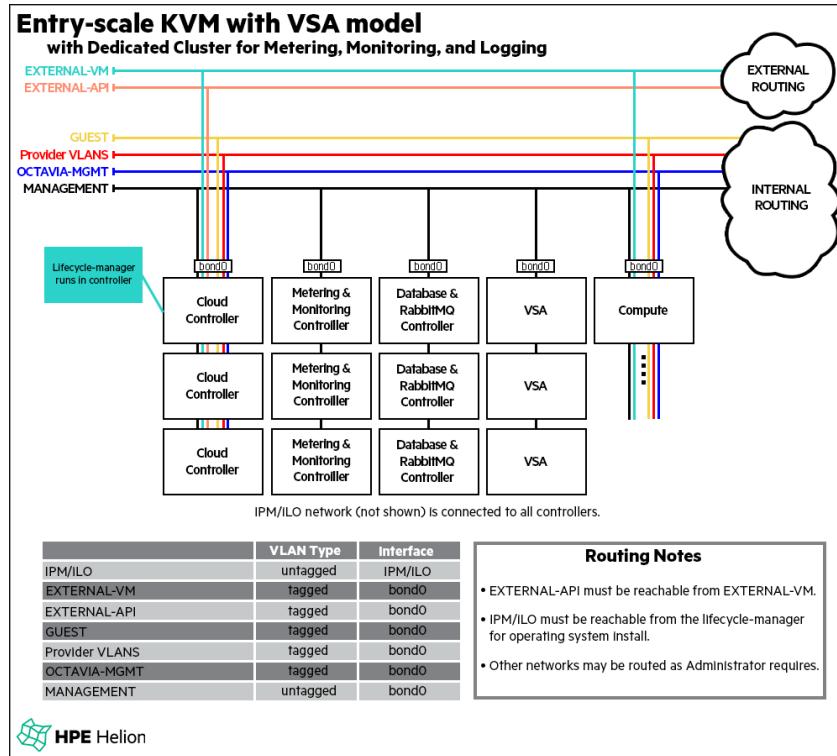
This model is a variant of the previous Entry-scale KVM with VSA model. It is designed to support greater levels of metering, monitoring, and logging.

- Metering - All meters required to support charge-back/show-back for core Infrastructure as a Service (IaaS) elements.
- Logging - Run all services at INFO level with the ability to change the settings to DEBUG in order to triage specific error conditions. Minimum retention for logs is 30 days to satisfy audit and compliance requirements.
- Monitoring - Full performance metrics and health checks for all services.

In order to provide increased processing power for these services, the following configuration changes are made to the control plane in this model:

- All services associated with metering, monitoring, and logging run on a dedicated three-node cluster. Three nodes are required for high availability with quorum.
- A dedicated three node cluster is used for RabbitMQ message queue and database services. This cluster is also used to provide additional processing for the message queue and database load associated with the additional metering, monitoring, and logging load. Three nodes are required for high availability with quorum.
- The main API cluster is reduced to two nodes. These services are stateless and do not require a quorum node for high availability.

This diagram below illustrates the physical networking used in this configuration.



Download the full image [[..../media/entryScaleDed/Entry-ScaleDedicatedAllNetworks.png](#)]

Download Editable Visio Network Diagram Template [[..../media/templates/HOS\\_Network\\_Diagram\\_Template.zip](#)]

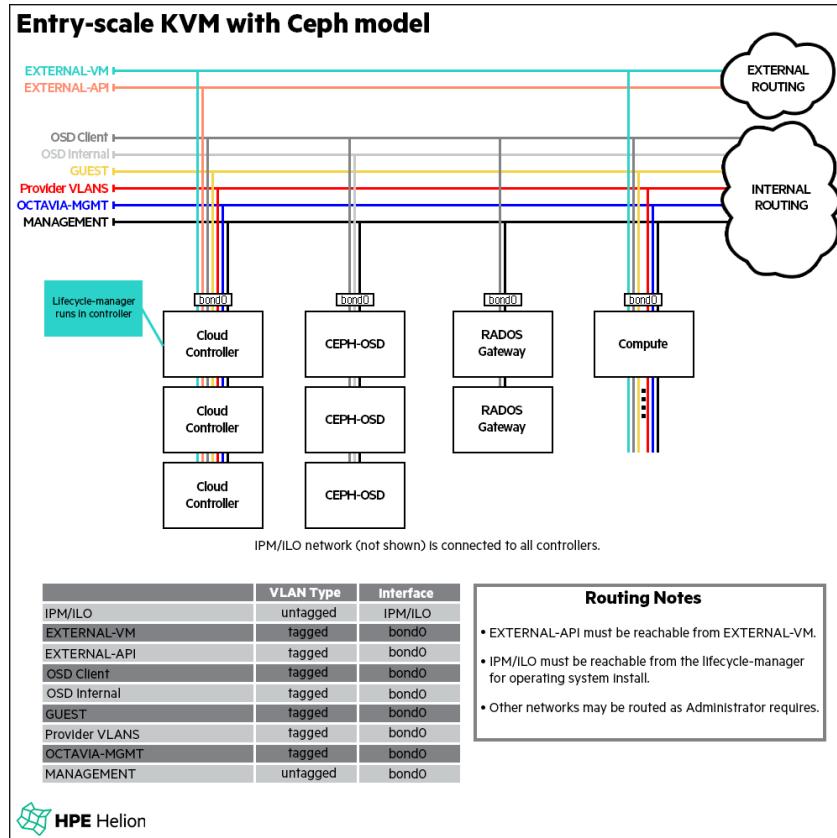
## Entry-scale KVM with Ceph Model

This example provides a KVM-based cloud using Ceph for both block and object storage.

The network traffic is segregated into the following VLANs:

- **Cloud Management** - This is the network that will be used for all internal traffic between the cloud services.
- **OSD Internal** - This is the network that will be used for internal traffic of cluster among Ceph OSD servers. Only Ceph OSD servers will need connectivity to this network.
- **OSD Client** - This is the network that Ceph clients will use to talk to Ceph Monitor and OSDs. Cloud controllers, Nova Compute, Ceph Monitor, OSD and Rados Gateway servers will need connectivity to this network.

This diagram below illustrates the physical networking used in this configuration. Click any network name in the diagram to see that network isolated.



Download full image [[..../..../media/entryScaleCeph/Entry-Scale-Ceph-AllNetworks.png](#)]

Download Editable Visio Network Diagram Template [[..../..../media/templates/HOS\\_Network\\_Diagram\\_Template.zip](#)]

This configuration is based on the entry-scale-kvm-ceph cloud input model which is included with the HPE Helion OpenStack distro. You will need to make the changes outlined below prior to the deployment of your Ceph cluster.

The table below lists out the key characteristics needed per server role for this configuration.

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"> <li>• 1 x 600 GB (minimum) - operating system drive</li> <li>• 2 x 600 GB (min-</li> </ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			imum) - Data drive			
Compute (KVM hypervisor)	Compute	1-3	2 X 600 GB (minimum)	32 GB (memory must be sized based on the virtual machine instances hosted on the Compute node)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64) with hardware virtualization support. The CPU cores must be sized based on the VM instances hosted by the Compute node.
CEPH-OSD	ceph-osd	0 or 3 (which will provide the recommended redundancy)	3 X 600 GB (minimum)	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
RADOS Gateway	radosgw	2	2 x 600 GB (minimum)	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

## nic\_mappings.yml

Ensure that your baremetal server NIC interfaces are correctly specified in the `~/helion/my_cloud/definition/data/nic_mappings.yml` file and that they meet the server requirements.

Here is an example with notes in-line:

`nic-mappings:`

```
## NIC specification for controller nodes. A bonded interface is used for the management network while a separate interface is used to connect to the Ceph nodes.
- name: CONTROLLER-NIC-MAPPING
  physical-ports:
    - logical-name: hed1
      type: simple-port
      bus-address: "0000:07:00.0"

    - logical-name: hed2
      type: simple-port
      bus-address: "0000:08:00.0"

    - logical-name: hed3
```

```
        type: simple-port
        bus-address: "0000:09:00.0"

        - logical-name: hed4
          type: simple-port
          bus-address: "0000:0a:00.0"

## NIC specification for compute nodes. One interface is used for the management
## network while the second interface is used to connect to the Ceph nodes.
- name: COMPUTE-NIC-MAPPING
  physical-ports:
    - logical-name: hed3
      type: simple-port
      bus-address: "0000:04:00.0"

    - logical-name: hed4
      type: simple-port
      bus-address: "0000:04:00.1"

## NIC specification for OSD nodes. The first interface is used for management net
## traffic. The second interface is used for client or public traffic. The third
## interface is used for internal OSD traffic.
- name: OSD-NIC-MAPPING
  physical-ports:
    - logical-name: hed1
      type: simple-port
      bus-address: "0000:06:00.0"

    - logical-name: hed2
      type: simple-port
      bus-address: "0000:06:00.1"

    - logical-name: hed3
      type: simple-port
      bus-address: "0000:06:00.2"

## NIC specification for RADOS Gateway nodes. The first interface is used for mana
## traffic. The second interface is used for client or public traffic.
- name: RGW-NIC-MAPPING
  physical-ports:
    - logical-name: hed1
      type: simple-port
      bus-address: "0000:07:00.0"

    - logical-name: hed2
      type: simple-port
      bus-address: "0000:07:00.1"
```

## servers.yml

Ensure that your servers in the `~/helion/my_cloud/definition/data/servers.yml` file are mapped to the correct NIC interface.

An example with the bolded line for `nic-mapping` illustrating this:

```
# Controller Nodes
- id: controller1
  ip-addr: 10.13.111.138
  server-group: RACK1
  role: CONTROLLER-ROLE
nic-mapping: CONTROLLER-NIC-MAPPING
  mac-addr: "f0:92:1c:05:69:10"
  ilo-ip: 10.12.8.214
  ilo-password: password
  ilo-user: admin

# Compute Nodes
- id: compute1
  ip-addr: 10.13.111.139
  server-group: RACK1
  role: COMPUTE-ROLE
nic-mapping: COMPUTE-NIC-MAPPING
  mac-addr: "83:92:1c:55:69:b0"
  ilo-ip: 10.12.8.215
  ilo-password: password
  ilo-user: admin

# OSD Nodes
- id: osd1
  ip-addr: 10.13.111.140
  server-group: RACK1
  role: OSD-ROLE
nic-mapping: OSD-NIC-MAPPING
  mac-addr: "d9:92:1c:25:69:e0"
  ilo-ip: 10.12.8.216
  ilo-password: password
  ilo-user: admin

# Ceph RGW Nodes
- id: rgw1
  ip-addr: 192.168.10.12
  role: RGW-ROLE
  server-group: RACK1
nic-mapping: RGW-NIC-MAPPING
  mac-addr: "8b:f6:9e:ca:3b:62"
  ilo-ip: 192.168.9.12
  ilo-password: password
  ilo-user: admin

- id: rgw2
  ip-addr: 192.168.10.13
  role: RGW-ROLE
  server-group: RACK2
nic-mapping: RGW-NIC-MAPPING
  mac-addr: "8b:f6:9e:ca:3b:63"
  ilo-ip: 192.168.9.13
  ilo-password: password
  ilo-user: admin
```

## net\_interfaces.yml

Define a new interface set for your OSD interfaces in the `~/helion/my_cloud/definition/data/net_interfaces.yml` file.

Here is an example with notes in-line:

```
- name: CONTROLLER-INTERFACES
  network-interfaces:
    ## This bonded interface is used by the controller
    ## nodes for cloud management traffic.
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed1
        provider: linux
        devices:
          - name: hed1
          - name: hed2
      network-groups:
        - EXTERNAL-API
        - EXTERNAL-VM
        - GUEST
        - MANAGEMENT
    ## This interface is used to connect the controller
    ## node to the Ceph nodes so that any Ceph client
    ## like cinder-volume can route data directly to
    ## Ceph over this interface.
    - name: ETH2
      device:
        name: hed3
      network-groups:
        - OSD-CLIENT

- name: COMPUTE-INTERFACES
  network-interfaces:
    - name: HETH3
      device:
        name: hed3
      forced-network-groups:
        - EXTERNAL-VM
        - GUEST
        - MANAGEMENT
    ## This interface is used to connect the compute node
    ## to the Ceph cluster so that a workload VM can route
    ## data traffic to the Ceph cluster over this interface.
    - name: HETH4
      device:
        name: hed4
```

```
forced-network-groups:
  - OSD-CLIENT

  - name: OSD-INTERFACES
    network-interfaces:
      ## This defines the interface used for management
      ## traffic like logging, monitoring, etc.
      - name: HETH1
        device:
          name: hed1
        network-groups:
          - MANAGEMENT
      ## This defines the interface used for client
      ## or data traffic.
      - name: HETH2
        device:
          name: hed2
        network-groups:
          - OSD-CLIENT
      ## This defines the interface used for internal
      ## cluster communication among OSD nodes.
      - name: HETH3
        device:
          name: hed3
        network-groups:
          - OSD-INTERNAL

  - name: RGW-INTERFACES
    network-interfaces:
      - name: BOND0
        device:
          name: bond0
        bond-data:
          options:
            mode: active-backup
            miimon: 200
            primary: hed3
          provider: linux
          devices:
            - name: hed3
            - name: hed4
    network-groups:
      - MANAGEMENT
      - OSD-CLIENT
```

## network\_groups.yml

Define the OSD network group in the `~/helion/my_cloud/definition/data/network_groups.yml` file:

```
#  
# OSD client  
#  
# This is the network group that will be used for
```

```
# internal traffic of cluster among OSDs.  
#  
- name: OSD-CLIENT  
  hostname-suffix: osdc  
  
  component-endpoints  
    - ceph-monitor  
    - ceph-osd  
    - ceph-radosgw  
  
#  
# OSD internal  
#  
# This is the network group that will be used for  
# internal traffic of cluster among OSDs.  
#  
- name: OSD-INTERNAL  
  hostname-suffix: osdi  
  
  component-endpoints:  
    - ceph-osd-internal
```

## networks.yml

Define the OSD VLAN in the `~/helion/my_cloud/definition/data/networks.yml` file.

The example below defines two separate network VLANs:

```
- name: OSD-CLIENT-NET  
  vlanid: 112  
  tagged-vlan: true  
  cidr: 192.168.187.0/24  
  gateway-ip: 192.168.187.1  
  network-group: OSD-CLIENT  
  
- name: OSD-INTERNAL-NET  
  vlanid: 116  
  tagged-vlan: true  
  cidr: 192.168.200.0/24  
  gateway-ip: 192.168.200.1  
  network-group: OSD-INTERNAL
```

## server\_groups.yml

Add the OSD network to the server groups in the `~/helion/my_cloud/definition/data/server_groups.yml` file, indicated by the bold portion below:

```
- name: CLOUD  
  server-groups:  
    - AZ1  
    - AZ2  
    - AZ3  
  networks:
```

- EXTERNAL-API-NET
- EXTERNAL-VM-NET
- GUEST-NET
- MANAGEMENT-NET
- **OSD-CLIENT-NET**
- **OSD-INTERNAL-NET**

## firewall\_rules.yml

Modify the firewall rules in the `~/helion/my_cloud/definition/data/firewall_rules.yml` file to allow OSD nodes to be pingable via the OSD network, indicated by the bold portion below:

### Note

Enabling ping for OSD-CLIENT and OSD-INTERNAL is optional. Enabling ping on these networks might make debugging connectivity issues on these networks easier.

```
- name: PING
network-groups:
- MANAGEMENT
- GUEST
- EXTERNAL-API
- OSD-CLIENT
- OSD-INTERNAL
rules:
# open ICMP echo request (ping)
- type: allow
  remote-ip-prefix: 0.0.0.0/0
  # icmp type
  port-range-min: 8
  # icmp code
  port-range-max: 0
  protocol: icmp
```

## Edit the README.html and README.md Files

You can edit the `~/helion/my_cloud/definition/README.html` and `~/helion/my_cloud/definition/README.md` files to reflect the OSD network group information if you wish. This change does not have any semantic implication and only assists with the readability of your model.

## Deploying Ceph Monitor Services on Dedicated Resource Nodes

In the HPE Helion OpenStack 5.0 example configurations, the Ceph monitor service is installed on the controller nodes by default. If you wish to break these out into their own cluster then you can do so by modifying the input model to form a separate cluster.

### Important

If you want to deploy the monitor service as a dedicated resource node, then you must decide prior to the deployment of Ceph. HPE Helion OpenStack 5.0 does not support deployment transition. Once Ceph is deployed, you cannot migrate the monitor service from controller to dedicated resource nodes.

## Prerequisite

The lifecycle manager must be set up before starting Ceph deployment. For more details on the installation of the lifecycle manager, see .

## To Install the Ceph Monitor Service on Dedicated Resource Nodes

Perform the following procedure to install the Ceph monitor on dedicated nodes. Note that Ceph requires at least 3 monitoring servers to form a cluster in case of node failure.

1. Log in to the lifecycle manager.
2. You will use the `entry-scale-kvm-ceph` example configuration as the base for these steps. Copy the example configuration files into the required setup directory before beginning the edit process:

```
cp -r ~/helion/examples/entry-scale-kvm-ceph/* ~/helion/my_cloud/definition/
```

3. Make the following edits to the `~/helion/my_cloud/definition/data/control_plane.yml` file:
  - a. Remove the reference to `- ceph-monitor` under the `service-components` section for your control plane cluster.
  - b. Add the details for your Ceph monitoring cluster. It is shown as the bolded portion in the example below, we added the rest to show the proper positioning:

```
clusters:  
  - name: cluster1  
    cluster-prefix: c1  
    server-role: CONTROLLER-ROLE  
    member-count: 3  
    allocation-policy: strict  
    service-components:  
      - lifecycle-manager  
      - ntp-server  
      ...  
  
  - name: ceph-mon  
    cluster-prefix: ceph-mon  
    server-role: CEP-MON-ROLE  
    min-count: 3  
    allocation-policy: strict  
    service-components:  
      - ntp-client  
      - ceph-monitor  
  
  - name: rgw  
    cluster-prefix: rgw  
    server-role: RGW-ROLE  
    ...
```

## Note

The indentation in the file is important to review the file to ensure it matches before continuing on.

4. Edit the `~/helion/my_cloud/definition/data/servers.yml` file to define all of the Ceph monitor nodes in the cluster. Here is an example, you will want to edit the values to match your environment:

```
# Ceph Monitor Nodes
- id: ceph-mon1
  ip-addr: 10.13.111.141
  server-group: RACK1
  role: CEP-MON-ROLE
  nic-mapping: MY-4PORT-SERVER
  mac-addr: "f0:92:1c:05:69:10"
  ilo-ip: 10.12.8.217
  ilo-password: password
  ilo-user: admin

- id: ceph-mon2
  ip-addr: 10.13.111.142
  server-group: RACK2
  role: CEP-MON-ROLE
  nic-mapping: MY-4PORT-SERVER
  mac-addr: "83:92:1c:55:69:b0"
  ilo-ip: 10.12.8.218
  ilo-password: password
  ilo-user: admin

- id: ceph-mon3
  ip-addr: 10.13.111.143
  server-group: RACK3
  role: CEP-MON-ROLE
  nic-mapping: MY-4PORT-SERVER
  mac-addr: "d9:92:1c:25:69:e0"
  ilo-ip: 10.12.8.219
  ilo-password: password
  ilo-user: admin

# Ceph RGW Nodes
- id: rgw1
  ...
  ...
```

5. Edit the `~/helion/my_cloud/definition/data/net_interfaces.yml` file to define a new network interface set for your Ceph monitors. You can copy the RGW-INTERFACES model as a base and then edit it to match your environment:

#### Three-network Ceph example:

```
interface-models:
  # Edit the device names and bond options
  # to match your environment
  #
  - name: CONTROLLER-INTERFACES
    network-interfaces:
      ## This bonded interface is used by the controller
      ## nodes for cloud management traffic.
      - name: BOND0
```

```
device:
  name: bond0
bond-data:
  options:
    mode: active-backup
    miimon: 200
    primary: hed1
  provider: linux
  devices:
    - name: hed1
    - name: hed2
network-groups:
  - EXTERNAL-API
  - EXTERNAL-VM
  - GUEST
  - MANAGEMENT
## This interface is used to connect the controller
## node to the Ceph nodes so that any Ceph client
## like cinder-volume can route data directly to
## Ceph over thisinterface.
  - name: HETH3
  device:
    name: hed3
  forced-network-groups:
    - OSD-CLIENT

  - name: COMPUTE-INTERFACES
  network-interfaces:
    - name: HETH3
    device:
      name: hed3
  network-groups:
    - EXTERNAL-VM
    - GUEST
    - MANAGEMENT
## This interface is used to connect the compute node
## to the Ceph cluster so that a workload VM can route
## data traffic to the Ceph cluster over thisinterface.
  - name: HETH4
  device:
    name: hed4
  forced-network-groups:
    - OSD-CLIENT

  - name: CEP-MON-INTERFACES
  network-interfaces:
    ## This defines the interface used for management
    ## traffic like logging, monitoring, etc.
  - name: BOND0
  device:
    name: bond0
bond-data:
  options:
    mode: active-backup
```

```
        miimon: 200
        primary: hed1
    provider: linux
    devices:
        - name: hed1
        - name: hed2
    network-groups:
        - MANAGEMENT
    ## This interface is used to connect the client
    ## node to the Ceph nodes so that any Ceph client
    ## like cinder-volume can route data directly to
    ## Ceph over this interface.
    - name: HETH3
    device:
        name: hed3
    forced-network-groups:
        - OSD-CLIENT

    - name: OSD-INTERFACES
    network-interfaces:
    ## This defines the interface used for management
    ## traffic like logging, monitoring, etc.
    - name: BOND0
    device:
        name: bond0
    bond-data:
        options:
            mode: active-backup
            miimon: 200
            primary: hed1
        provider: linux
        devices:
            - name: hed1
            - name: hed2
    network-groups:
        - MANAGEMENT
    ## This defines the interface used for client
    ## or data traffic.
    - name: HETH3
    device:
        name: hed3
    network-groups:
        - OSD-CLIENT
    ## This defines the interface used for internal
    ## cluster communication among OSD nodes.
    - name: HETH4
    device:
        name: hed4
    network-groups:
        - OSD-INTERNAL
```

**Two-network Ceph example:**

```
interface-models:
```

```
# Edit the device names and bond options
# to match your environment
#
- name: CONTROLLER-INTERFACES
  network-interfaces:
    ## This bonded interface is used by the controller
    ## nodes for cloud management traffic.
    ## The same interface is also used to connect the client
    ## node to the Ceph nodes so that any Ceph client
    ## like cinder-volume can route data directly to
    ## Ceph over thisinterface.
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed1
          provider: linux
        devices:
          - name: hed1
          - name: hed2
      network-groups:
        - EXTERNAL-API
        - EXTERNAL-VM
        - GUEST
        - MANAGEMENT

- name: COMPUTE-INTERFACES
  network-interfaces:
    ## The same interface is also used to connect the compute node
    ## to the Ceph cluster so that a workload VM can route
    ## data traffic to the Ceph cluster over thisinterface.
    - name: HETH3
      device:
        name: hed3
      network-groups:
        - EXTERNAL-VM
        - GUEST
        - MANAGEMENT

- name: CEP-MON-INTERFACES
  network-interfaces:
    ## This defines the interface used for management
    ## traffic like logging, monitoring, etc.
    ## The same interface is also used to connect the client
    ## node to the Ceph nodes so that any Ceph client
    ## like cinder-volume can route data directly to
    ## Ceph over thisinterface.
    - name: BOND0
      device:
        name: bond0
      bond-data:
```

```
options:
    mode: active-backup
    miimon: 200
    primary: hed1
provider: linux
devices:
    - name: hed1
    - name: hed2
network-groups:
    - MANAGEMENT

- name: OSD-INTERFACES
network-interfaces:
## This defines the interface used for management
## traffic like logging, monitoring, etc.
## The same interface is also used for client
## or data traffic.
    - name: BOND0
device:
    name: bond0
bond-data:
options:
    mode: active-backup
    miimon: 200
    primary: hed1
provider: linux
devices:
    - name: hed1
    - name: hed2
network-groups:
    - MANAGEMENT
## This defines the interface used for internal
## cluster communication among OSD nodes.
    - name: HETH4
device:
    name: hed4
network-groups:
    - OSD-INTERNAL
```

**Single-network Ceph example:**

```
interface-models:
    # Edit the device names and bond options
    # to match your environment
    #
- name: CONTROLLER-INTERFACES
network-interfaces:
## This bonded interface is used by the controller
## nodes for cloud management traffic.
## The same interface is also used to connect the client
## node to the Ceph nodes so that any Ceph client
## like cinder-volume can route data directly to
## Ceph over this interface.
    - name: BOND0
```

```
device:
  name: bond0
bond-data:
  options:
    mode: active-backup
    miimon: 200
    primary: hed1
  provider: linux
  devices:
    - name: hed1
    - name: hed2
network-groups:
  - EXTERNAL-API
  - EXTERNAL-VM
  - GUEST
  - MANAGEMENT

- name: COMPUTE-INTERFACES
## This interface is also used to connect the compute node
## to the Ceph cluster so that a workload VM can route
## data traffic to the Ceph cluster over thisinterface.
  network-interfaces:
    - name: HETH3
    device:
      name: hed3
  network-groups:
    - EXTERNAL-VM
    - GUEST
    - MANAGEMENT

- name: CEP-MON-INTERFACES
  network-interfaces:
    ## This defines the interface used for management
    ## traffic like logging, monitoring, etc.
    ## The same interface is also used to connect the client
    ## node to the Ceph nodes so that any Ceph client
    ## like cinder-volume can route data directly to
    ## Ceph over thisinterface.
    - name: BOND0
    device:
      name: bond0
  bond-data:
    options:
      mode: active-backup
      miimon: 200
      primary: hed1
    provider: linux
    devices:
      - name: hed1
      - name: hed2
  network-groups:
    - MANAGEMENT

- name: OSD-INTERFACES
```

```
network-interfaces:  
    ## This defines the interface used for management  
    ## traffic like logging, monitoring, etc.  
    ## The same interface is also used for internal cluster  
    ## communication among the OSD nodes.  
    ## The same interface is also used for internal  
    ## cluster communication among OSD nodes.  
    - name: BOND0  
      device:  
        name: bond0  
      bond-data:  
        options:  
          mode: active-backup  
          miimon: 200  
          primary: hed1  
        provider: linux  
        devices:  
          - name: hed1  
          - name: hed2  
      network-groups:  
        - MANAGEMENT
```

6. Create a new file named `disks_ceph_monitor.yml` in the `~/helion/my_cloud/definition/data/` directory which will define the disk model for your Ceph monitors. You can use the `disks_rgw.yml` file as a base and then edit to match your environment:

```
disk-models:  
  - name: CEP-MON-DISKS  
    # Disk model to be used for Ceph monitor nodes  
    # /dev/sda_root is used as a volume group for /, /var/log and /var/crash  
    # sda_root is a templated value to align with whatever partition is really used  
    # This value is checked in os config and replaced by the partition actually used  
    # on sda e.g. sdal or sda5  
  
    volume-groups:  
      - name: hlm-vg  
        physical-volumes:  
          - /dev/sda_root  
  
        logical-volumes:  
          # The policy is not to consume 100% of the space of each volume group.  
          # 5% should be left free for snapshots and to allow for some flexibility.  
          - name: root  
            size: 30%  
            fstype: ext4  
            mount: /  
          - name: log  
            size: 45%  
            mount: /var/log  
            fstype: ext4  
            mkfs-opt: -O large_file  
          - name: crash  
            size: 20%  
            mount: /var/crash
```

```
fstype: ext4
mkfs-opt: -O large_file
consumer:
  name: os
```

7. Edit the `~/helion/my_cloud/definition/data/server_roles.yml` file to define a new server role for your Ceph monitors:

```
- name: CEP-MON-ROLE
  interface-model: CEP-MON-INTERFACES
  disk-model: CEP-MON-DISKS
```

8. Commit your configuration:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "adding dedicated Ceph monitor cluster"
```

9. Run the following playbook to add your nodes into Cobbler:

```
cd ~/helion/hos/ansible/
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

10. To reimage all the nodes using PXE, run the following playbook:

```
cd ~/helion/hos/ansible/
ansible-playbook -i hosts/localhost bm-reimage.yml
```

11. Run the configuration processor:

```
cd ~/helion/hos/ansible/
ansible-playbook -i hosts/localhost config-processor-run.yml
```

12. Update your deployment directory with this playbook:

```
cd ~/helion/hos/ansible/
ansible-playbook -i hosts/localhost ready-deployment.yml
```

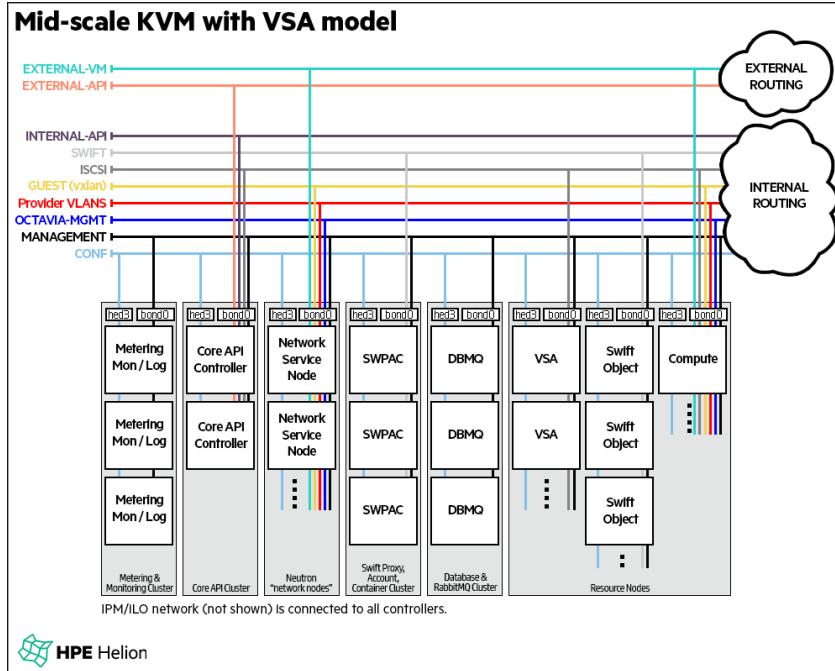
13. Deploy these changes:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts site.yml
```

## Mid-scale KVM with VSA Model

The mid-scale model illustrates two important aspects of configuring HPE Helion OpenStack for increased scale. The controller services are distributed across a greater number of controllers and a number of the networks are configured as multiple L3 segments (implementing per-rack networking).

## Example Configurations



Network Group	VLAN type	Interface	Multiple networks per group?
IPMI/ILO	untagged	IPMI/ILO	Possible
CONF	untagged	hed3	No *
MANAGEMENT	untagged	bond0	Possible
OCTAVIA-MGMT	tagged	bond0	Possible
Provider VLANs	tagged	bond0	n/a
GUEST	tagged	bond0	Possible
iSCSI	tagged	bond0	No *
SWIFT	tagged	bond0	Possible
INTERNAL-API	tagged	bond0	No *
EXTERNAL-API	tagged	bond0	No *
EXTERNAL-VM	tagged	bond0	No *

### Routing Notes:

- EXTERNAL-API must be reachable from EXTERNAL-VM so in-cloud VMs can use the OpenStack APIs via their publicURL.
  - INTERNAL-API must be reachable from MANAGEMENT so services on the MANAGEMENT network can use the OpenStack APIs via their InternalURL or AdminURL.
  - When there are multiple networks in a network-group, each network in the group must be reachable from other networks in that group.
  - IPMI/ILO must be reachable from CONF for os-install.
  - Other networks may be routed as Administrator requires.
- \* Regarding multiple networks per group, some groups contain only a single network due to application constraints:
- VSA nodes share a cluster virtual IP addresses on the iSCSI network, the virtual IP addresses may be hosted by any VSA node.
  - Core API nodes share a cluster virtual IP addresses on both the INTERNAL-API and EXTERNAL-API networks; the virtual IP addresses may be hosted by any core API node.
  - Neutron expects the EXTERNAL-VM network to span all compute nodes and network service nodes for floating IPs and router default SNAT IP addresses.
  - The lifecycle-manager provides PXE boot services on CONF.

Download the full network image [../../../../media/networkImages/Mid-Scale-AllNetworks.png]

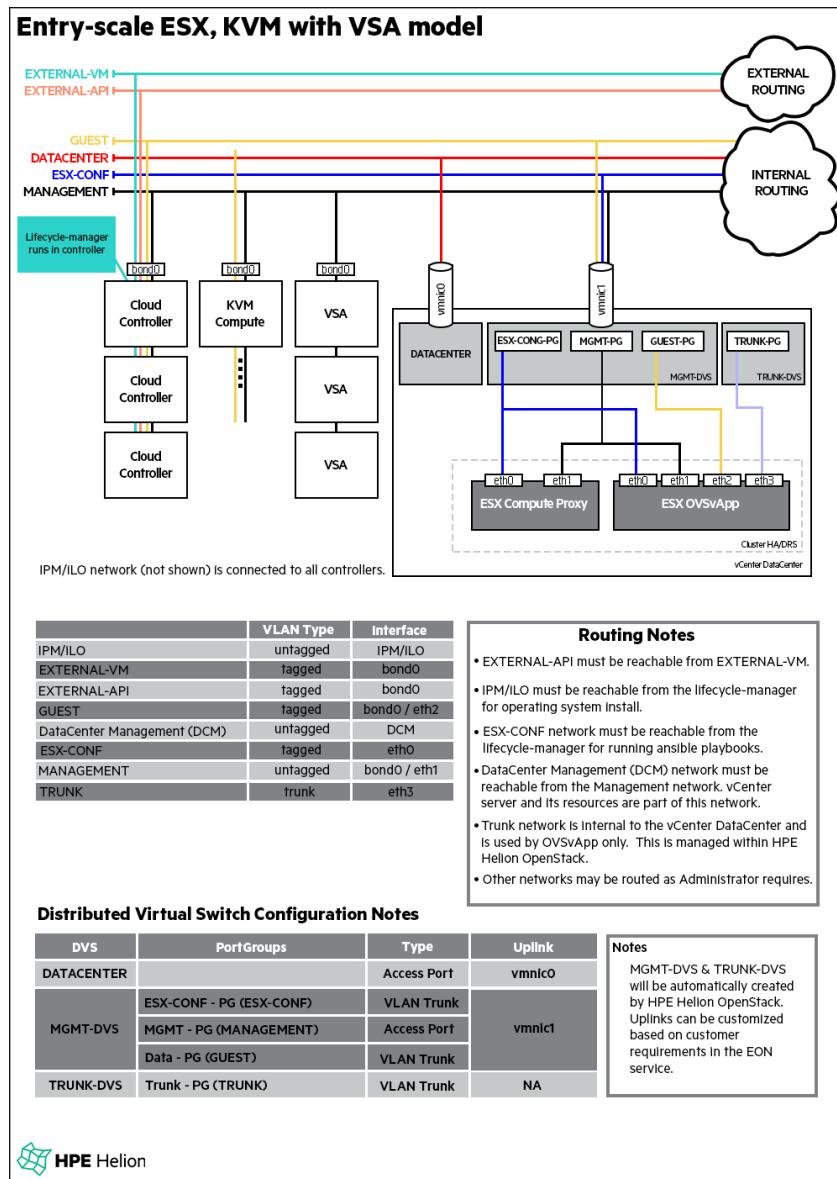
Download Editable Visio Network Diagram Template [../../../../media/templates/HOS\_Network\_Diagram\_Template.zip]

The distribution of services across controllers is only one possible configuration, and other combinations can also be expressed.

# ESX Examples

## Entry-scale ESX, KVM with VSA Model

This example shows how to integrate HPE Helion OpenStack with ESX, KVM with VSA in the same Cloud. The controller configuration is essentially the same as in the Entry-scale KVM with VSA Model example, but the resource nodes supported are ESX (provided by vCenter), KVM and VSA. In addition, a number of controller virtual machines are created for each vCenter cluster: one ESX Compute virtual machine (which provides the nova-compute proxy for vCenter) and one OVSvApp virtual machine per cluster member (which provides network access). These virtual machines are created automatically by HPE Helion OpenStack as part of activating the vCenter cluster, and are therefore not defined in the example.



Download [../../../../media/hos.docs/exampleconfigs/Entry-ScaleESX-KVMwithVSAmodel.png]

Download Editable Visio Network Diagram Template [../../../../media/templates/HOS\_Network\_Diagram\_Template.zip]

The physical networking configuration is also largely the same as the KVM example, with the default GUEST network VxLAN as the Neutron networking model.

A separate configuration network (CONF) is required for configuration access from the lifecycle manager. This network must be reachable from the Management network.

These recommended minimums are based on the included included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

HPE Helion OpenStack 5.0 currently supports the following ESXi versions:

- ESXi version 5.5 (Update 3)
- ESXi version 6.0
- ESXi version 6.0 (Update 1b)

The following are the requirements for your vCenter server:

- Software
  - vCenter 5.5 Update 3 and above (It is recommended to run the same server version as the ESXi hosts)
- License Requirements
  - vSphere Enterprise Plus license

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"> <li>• 1 x 600 GB (minimum) - operating system drive</li> <li>• 2 x 600 GB (minimum) - Data drive</li> </ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
Compute (ESXi hypervisor)		2	2 X 1 TB (minimum, shared across all nodes)	128 GB (minimum)	2 x 10 Gbit/s +1 NIC (for DC access)	16 CPU (64-bit) cores total (Intel x86_64)
Compute (KVM hypervisor)	kvm-compute	1-3	2 X 600 GB (minimum)	32 GB (memory must be sized based)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (In-

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			on the virtual machine instances hosted on the Compute node)			tel x86_64) with hardware virtualization support. The CPU cores must be sized based on the VM instances hosted by the Compute node.
Block Storage (Optional)	VSA	0 or 3 (which will provide the recommended redundancy)	3 X 600 GB (minimum) See Chapter 2, <i>Pre-Installation Checklist</i> for more details.	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

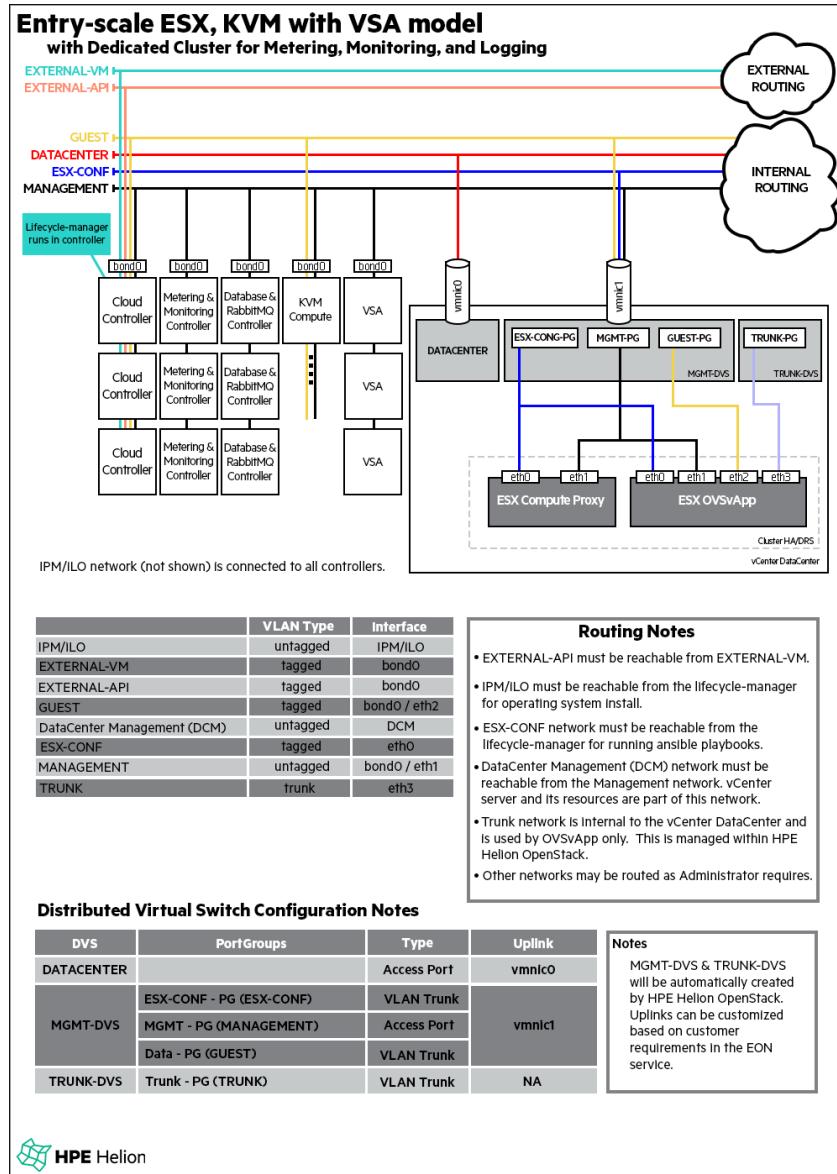
## Entry-scale ESX, KVM with VSA Model with Dedicated Cluster for Metering, Monitoring, and Logging

This model is a variant of the Entry-scale ESX KVM with VSA model. It is designed to support greater levels of metering, monitoring, and logging.

- Metering - All meters required to support charge-back/show-back for core Infrastructure as a Service (IaaS) elements.
- Logging - Run all services at INFO level with the ability to change the settings to DEBUG in order to triage specific error conditions. Minimum retention for logs is 30 days to satisfy audit and compliance requirements.
- Monitoring - Full performance metrics and health checks for all services.

In order to provide increased processing power for these services, the following configuration changes are made to the control plane in this model:

- All services associated with metering, monitoring, and logging run on a dedicated three-node cluster. Three nodes are required for high availability with quorum.
- A dedicated three node cluster is used for RabbitMQ message queue and database services. This cluster is also used to provide additional processing for the message queue and database load associated with the additional metering, monitoring, and logging load. Three nodes are required for high availability with quorum.
- The main API cluster is reduced to two nodes. These services are stateless and do not require a quorum node for high availability.



Download [../../../../media/hos.docs/exampleconfigs/Entry-ScaleESX-KVMwithVSADEDicatedMML.png]

Download Editable Visio Network Diagram Template [../../../../media/templates/HOS\_Network\_Diagram\_Template.zip]

The physical networking configuration is also largely the same as the KVM example, with the default GUEST network VxLAN as the Neutron networking model.

A separate configuration network (CONF) is required for configuration access from the lifecycle manager. This network must be reachable from the Management network.

These recommended minimums are based on the included included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

HPE Helion OpenStack 5.0 currently supports the following ESXi versions:

- ESXi version 5.5 (Update 3)

- ESXi version 6.0

- ESXi version 6.0 (Update 1b)

The following are the requirements for your vCenter server:

- Software
  - vCenter 5.5 Update 3 and above (It is recommended to run the same server version as the ESXi hosts)
- License Requirements
  - vSphere Enterprise Plus license

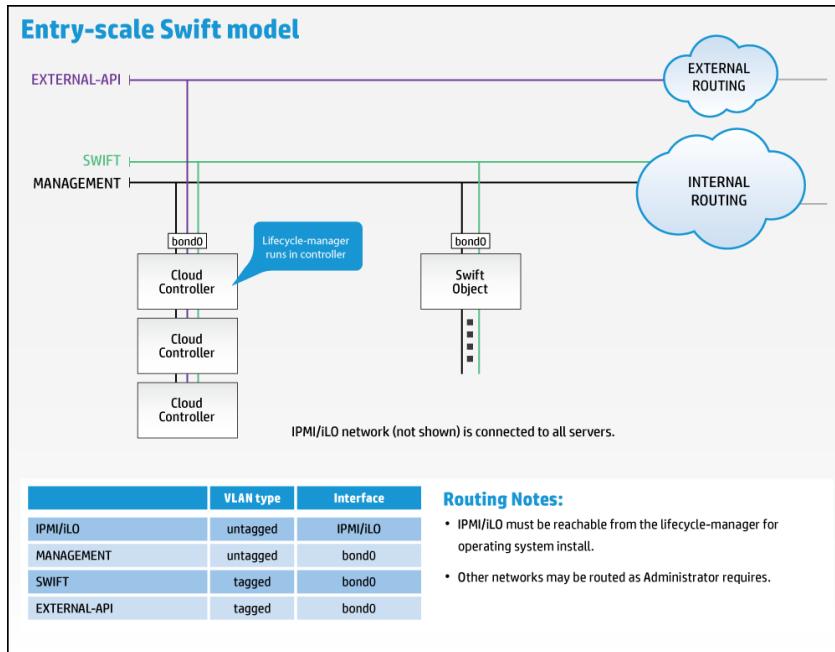
Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Core-API Controller	2	<ul style="list-style-type: none"> <li>• 1 x 600 GB (minimum) - operating system drive</li> <li>• 2 x 300 GB (minimum) - Swift drive</li> </ul>	128 GB	2 x 10 Gbit/s with PXE Support	24 CPU (64-bit) cores total (Intel x86_64)
	DBMQ Cluster	3	<ul style="list-style-type: none"> <li>• 1 x 600 GB (minimum) - operating system drive</li> <li>• 1 x 300 GB (minimum) - MySQL drive</li> </ul>	96 GB	2 x 10 Gbit/s with PXE Support	24 CPU (64-bit) cores total (Intel x86_64)
	Metering Mon/Log Cluster	3	<ul style="list-style-type: none"> <li>• 1 x 600 GB (minimum) - operating system drive</li> </ul>	128 GB	2 x 10 Gbit/s with one PXE enabled port	24 CPU (64-bit) cores total (Intel x86_64)
Compute (ESXi hypervisor)		2 (minimum)	2 X 1 TB (minimum, shared)	64 GB (memory must be sized based)	2 x 10 Gbit/s +1 NIC (for bit)	16 CPU (64-bit) cores

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			across all nodes)	on the virtual machine instances hosted on the Compute node)	Data Center access)	total (Intel x86_64)
Compute (KVM hypervisor)	kvm-compute	1-3	2 X 600 GB (minimum)	32 GB (memory must be sized based on the virtual machine instances hosted on the Compute node)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64) with hardware virtualization support. The CPU cores must be sized based on the VM instances hosted by the Compute node.
Block Storage (Optional)	VSA	0 or 3 (which will provide the recommended redundancy)	3 X 600 GB (minimum) See Chapter 2, <i>Pre-Installation Checklist</i> for more details.	32 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

## Swift Examples

### Entry-scale Swift Model

This example shows how HPE Helion OpenStack can be configured to provide a Swift-only configuration, consisting of three controllers and one or more Swift object servers.



Download a high-resolution version [[..../media/examples/entry\\_scale\\_swift\\_lg.png](#)]

Download Editable Visio Network Diagram Template [[..../media/templates/HOS\\_Network\\_Diagram\\_Template.zip](#)]

The example requires the following networks:

- **External API** - This is the network that users will use to make requests to the cloud.
- **Swift** - This is the network that will be used for all data traffic between the Swift services.
- **Management** - This is the network that will be used for all internal traffic between the cloud services, including node provisioning. This network must be on an untagged VLAN.

All of these networks are configured to be presented via a pair of bonded NICs. The example also enables provider VLANs to be configured in Neutron on this interface.

In the diagram "External Routing" refers to whatever routing you want to provide to allow users to access the External API. "Internal Routing" refers to whatever routing you want to provide to allow administrators to access the Management network.

If you are using HPE Helion OpenStack to install the operating system, then an IPMI/iLO network connected to the IPMI/iLO ports of all servers and routable from the lifecycle manager is also required for BIOS and power management of the node during the operating system installation process.

In the example the controllers use one disk for the operating system and two disks for Swift proxy and account storage. The Swift object servers use one disk for the operating system and four disks for Swift storage. These values can be modified to suit your environment.

These recommended minimums are based on the included included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

The entry-scale-swift example runs the Swift proxy, account and container services on the three controller servers. However, it is possible to extend the model to include the Swift proxy, account and

## Example Configurations

---

container services on dedicated servers (typically referred to as the Swift proxy servers). If you are using this model, we have included the recommended Swift proxy servers specs in the table below.

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	<ul style="list-style-type: none"> <li>1 x 600 GB (minimum) - operating system drive</li> <li>2 x 600 GB (minimum) - Swift account/container data drive</li> </ul>	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)
Swift Object	swobj	3	<p>If using x3 replication only:</p> <ul style="list-style-type: none"> <li>1 x 600 GB (minimum, see considerations at bottom of page for more details)</li> </ul> <p>If using Erasure Codes only or a mix of x3 replication and Erasure Codes:</p> <ul style="list-style-type: none"> <li>6 x 600 GB (minimum, see considerations at bottom of page for</li> </ul>	32 GB (see considerations at bottom of page for more details)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			more details)			
Swift Proxy, Account, and Container	swpac	3	2 x 600 GB (minimum, see considerations at bottom of page)	64 GB (see considerations at bottom of page for more details)	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores total (Intel x86_64)

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			for more details)			

**Considerations for your Swift object and proxy, account, container servers RAM and disk capacity needs**

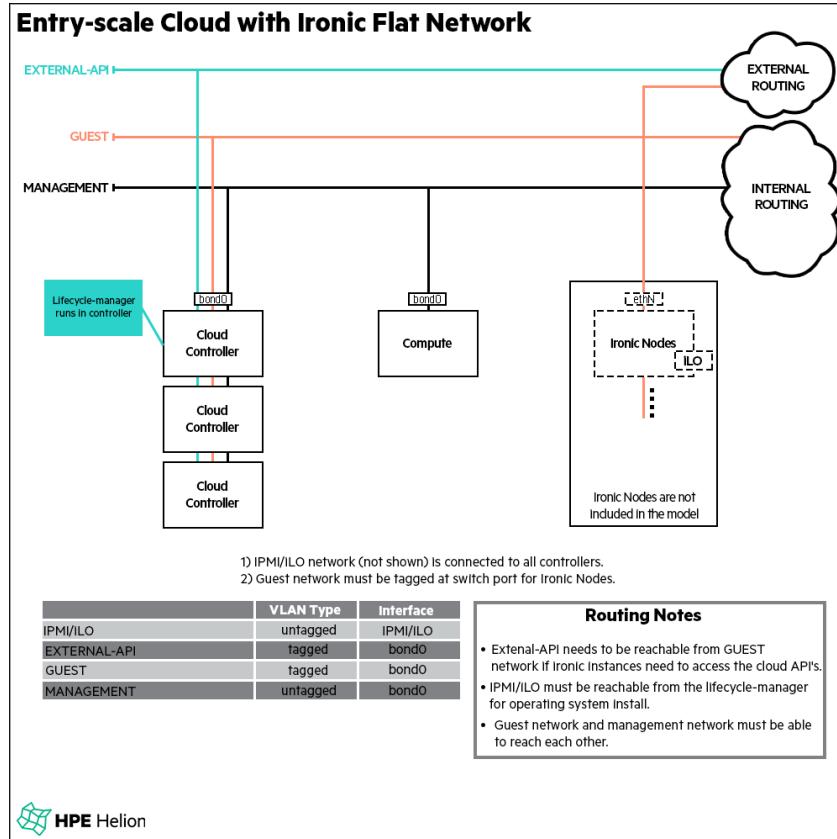
Swift can have a diverse number of hardware configurations. For example, a Swift object server may have just a few disks (minimum of 6 for erasure codes) or up to 70 and beyond. The memory requirement needs to be increased as more disks are added. The general rule of thumb for memory needed is 0.5 GB per TB of storage. For example, a system with 24 hard drives at 8TB each, giving a total capacity of 192TB, should use 96GB of RAM. However, this does not work well for a system with a small number of small hard drives or a very large number of very large drives. So, if after calculating the memory given this guideline, if the answer is less than 32GB then go with 32GB of memory minimum and if the answer is over 256GB then use 256GB maximum, no need to use more memory than that.

When considering the capacity needs for the Swift proxy, account, and container (PAC) servers, you should calculate 2% of the total raw storage size of your object servers to specify the storage required for the PAC servers. So, for example, if you were using the example we provided earlier and you had an object server setup of 24 hard drives with 8TB each for a total of 192TB and you had a total of 6 object servers, that would give a raw total of 1152TB. So you would take 2% of that, which is 23TB, and ensure that much storage capacity was available on your Swift proxy, account, and container (PAC) server cluster. If you had a cluster of three Swift PAC servers, that would be ~8TB each.

Another general rule of thumb is that if you are expecting to have more than a million objects in a container then you should consider using SSDs on the Swift PAC servers rather than HDDs.

# Ironic Examples

## Entry-scale Cloud with Ironic Flat Network



Download full image [[..../..../media/hos.docs/exampleconfigs/entry\\_scale\\_ironic.png](#)]

Download Editable Visio Network Diagram Template [[..../..../media/templates/HOS\\_Network\\_Diagram\\_Template.zip](#)]

When using the `agent_i1o` driver, you should ensure that the most recent iLO controller firmware is installed. A recommended minimum for the iLO4 controller is version 2.30.

The recommended minimum hardware requirements are based on the included with the base installation and are suitable only for demo environments. For production systems you will want to consider your capacity and performance requirements when making decisions about your hardware.

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
Dedicated lifecycle manager (optional)	Lifecycle-manager	1	300 GB	8 GB	1 x 10 Gbit/s with PXE Support	8 CPU (64-bit) cores total (Intel x86_64)
Control Plane	Controller	3	• 1 x 600 GB (minimum) - op-	64 GB	2 x 10 Gbit/s with one PXE enabled port	8 CPU (64-bit) cores

Node Type	Role Name	Required Number	Server Hardware - Minimum Requirements and Recommendations			
			Disk	Memory	Network	CPU
			<ul style="list-style-type: none"> <li>• Primary operating system drive</li> <li>• 2 x 600 GB (minimum) - Data drive</li> </ul>			total (Intel x86_64)
Compute	Compute	1	1 X 600 GB (minimum)	16 GB	2 x 10 Gbit/s with one PXE enabled port	16 CPU (64-bit) cores total (Intel x86_64)

For more details about the supported network requirements, see .

## Configuration Files

This section contains examples of the configuration files for the Entry-scale Cloud with Ironic Flat Network.

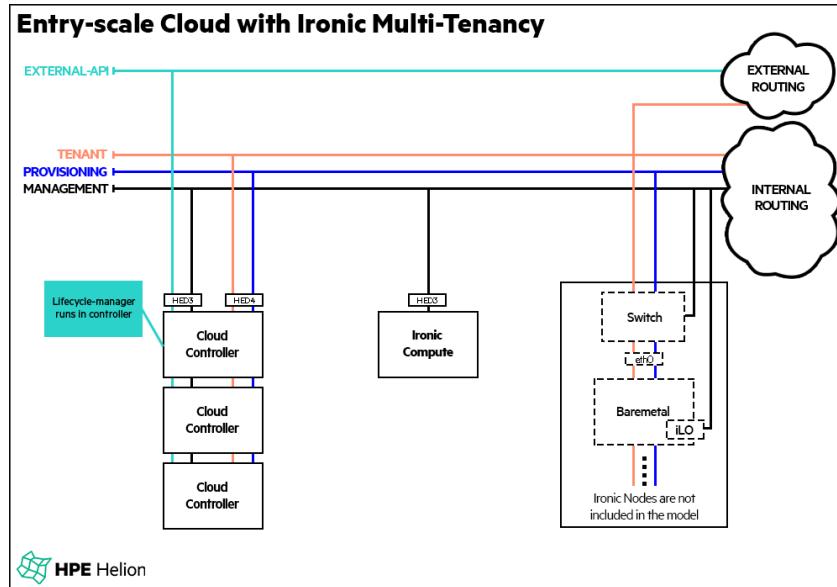
```
--- product: version: 2 control-planes: - name: control-plane-1 control-plane-prefix: cp1 region-name: region1 failure-zones: - AZ1 - AZ2 - AZ3 common-service-components: - logging-producer - monasca-agent - freezer-agent - stunnel - lifecycle-manager-target clusters: - name: cluster1 cluster-prefix: c1 server-role: CONTROLLER-ROLE member-count: 3 allocation-policy: strict service-components: - lifecycle-manager - ntp-server - swift-ring-builder - mysql - ip-cluster - apache2 - keystone-api - keystone-client - rabbitmq - glance-api - glance-registry - glance-client - nova-api - nova-scheduler-ironic - nova-scheduler - nova-conductor - nova-console-auth - nova-novncproxy - nova-client - neutron-server - neutron-ml2-plugin - neutron-dhcp-agent - neutron-metadata-agent - neutron-openvswitch-agent - neutron-client - horizon - swift-proxy - memcached - swift-account - swift-container - swift-object - swift-client - heat-api - heat-api-cfn - heat-api-cloudwatch - heat-engine - heat-client - ironic-api - ironic-conductor - ironic-client - openstack-client - ceilometer-api - ceilometer-polling - ceilometer-agent-notification - ceilometer-common - ceilometer-client - zookeeper - kafka - vertica - storm - monasca-api - monasca-persister - monasca-notifier - monasca-threshold - monasca-client - logging-server - ops-console-web - ops-console-monitor - freezer-api - barbican-api - barbican-client - barbican-worker resources: - name: ironic-compute resource-prefix: ir-compute server-role: IRONIC-COMPUTE-ROLE allocation-policy: any service-components: - neutron-openvswitch-agent - nova-compute-ironic - nova-compute - ntp-client --- product: version: 2 networks: ## This example uses the following networks ## Network CIDR VLAN # ----- # External API 10.0.1.0/24 101 (tagged) # Guest 102 (tagged) # Management 192.168.10.0/24 100 (untagged) # # Notes: # 1. Defined as part of Neutron configuration # # Modify these values to match your environment # - name: EXTERNAL-API-NET tagged-vlan: true vlanid: 101 cidr: 10.0.1.0/24 gateway-ip: 10.0.1.1 network-group: EXTERNAL-API #start-address: 10.0.1.10 #end-address: 10.0.1.250 - name: GUEST-NET tagged-vlan: true vlanid: 102 network-group: GUEST - name: MANAGEMENT-NET tagged-vlan: false vlanid: 100 cidr: 192.168.10.0/24 gateway-ip: 192.168.10.1 network-group: MANAGEMENT #start-address: 192.168.10.10 #end-address: 192.168.10.250 --- product: version: 2 network-groups: # # External API # # This is the network group that users will use to # access the public API endpoints of your cloud # - name: EXTERNAL-API hostname-suffix: extapi load-balancers: - provider: ip-cluster name: extlb # If external-name is set then public urls in keystone # will use this name instead of the IP address. # You must either set this to a name that can be resolved in your network # or comment out this line to use IP addresses external-name: tls-components: - default roles: - public cert-file: my-public-entriescale-ironic-cert # This is the name of the certificate that will be used on load balancer. # Replace this with name
```

```

of file in "~helion/my_cloud/config/tls/certs/". # This is the certificate that matches your setting for external-name # # Note that it is also possible to have per service certificates: # # cert-file: # default: my-public-entryscale-ironic-cert # horizon: my-horizon-cert # nova-api: my-nova-cert # # # GUEST # # This is the network group that will be used to provide # private networks to Baremetals # - name: GUEST hostname-suffix: guest tags: - neutron.networks.flat: provider-physical-network: physnet1 # Management # # This is the network group that will be used to for # management traffic within the cloud. # # The interface used by this group will be presented # to Neutron as physnet1, and used by provider VLANS # - name: MANAGEMENT hostname-suffix: mgmt hostname: true component-endpoints: - default routes: - default load-balancers: - provider: ip-cluster name: lb components: - default roles: - internal - admin --- product: version: 2 interface-models: # These examples uses hed3 and hed4 as a bonded # pair for all networks on all three server roles # # Edit the device names and bond options # to match your environment # - name: CONTROLLER-INTERFACES network-interfaces: - name: BOND0 device: name: bond0 bond-data: options: mode: active-backup miimon: 200 primary: hed3 provider: linux devices: - name: hed3 - name: hed4 network-groups: - EXTERNAL-API - GUEST - MANAGEMENT - name: COMPUTE-IRONIC-INTERFACES network-interfaces: - name: BOND0 device: name: bond0 bond-data: options: mode: active-backup miimon: 200 primary: hed3 provider: linux devices: - name: hed3 - name: hed4 network-groups: - MANAGEMENT - GUEST

```

## Entry-scale Cloud with Ironic Multi-Tenancy



	VLAN switch configuration	VLAN type in HOS	Interface
EXTERNAL-API	Tagged for controllers, needs subnet with IP address range	tagged	<ul style="list-style-type: none"> <li>• hed3 on controllers</li> </ul>
MANAGEMENT	Untagged for controllers and compute, needs subnet with IP address range	untagged	<ul style="list-style-type: none"> <li>• hed3 on controllers and compute</li> </ul>
PROVISIONING	Tagged for controllers, needs subnet with IP address range. For ironic baremetal nodes, switch	neutron provider VLAN (untagged)	<ul style="list-style-type: none"> <li>• hed4 on controllers</li> <li>• eth0 on baremetal nodes</li> </ul>

	<b>VLAN switch configuration</b>	<b>VLAN type in HOS</b>	<b>Interface</b>
	config will be set dynamically by Neutron.		
TENANT	Tagged range of VLANs. Number of VLANs in range may be up to number of baremetal nodes (for each node have it's own network). For ironic baremetal nodes, switch config will be set dynamically by Neutron.	neutron provider VLAN (untagged)	<ul style="list-style-type: none"> <li>• hed4 on controllers</li> <li>• eth0 on baremetal nodes</li> </ul>

**Routing notes**

1. EXTERNAL-API needs to be reachable from TENANT VLANs if ironic instances need to access the cloud APIs
2. Controller and compute nodes IPMI/iLO must be reachable from lifecycle-manager for operating system install
3. Baremetal node IPMI/iLO must be reachable from controllers via MANAGEMENT network for operating system install
4. Switch management IPs must be reachable from controllers MANAGEMENT network for VLAN configuration setting
5. TENANT VLANs should be configured to allow inbound/outbound external access, if external access is needed for Ironic instances

---

# Chapter 11. Modifying the Entry-scale KVM with VSA Model for Your Environment

This section covers the changes that need to be made to the input model to deploy and run this cloud model in your environment.

This section is written from the perspective of the `entry-scale-kvm-vsa` example, although the same principles apply to all of the examples.

There are two categories of modifications that we will look at:

1. Localizations - These are the minimum set of changes that you need to make to adapt the examples to run in your environment. These are mostly concerned with networking. See the section called “Localizing the Input Model”.
2. Customizations - These describe more general changes that you can make to your model, e.g. changing disk storage layouts. See the section called “Customizing the Input Model”

Note that, as a convention, the examples use upper case for the object names, but these strings are only used to define the relationships between objects and have no specific significance to the configuration processor. You can change the names to values that are relevant to your context providing you do so consistently across the input model.

## Localizing the Input Model

This section covers the minimum set of changes needed to localize the cloud for your environment. This assumes you are using other features of the example unchanged:

- Update `networks.yml` to specify the network addresses (VLAN IDs and CIDR values) for your cloud.
- Update `nic_mappings.yml` to specify the PCI bus information for your servers' Ethernet devices.
- Update `net_interfaces.yml` to provide network interface configurations, such as bond settings and bond devices.
- Update `network_groups.yml` to provide the public URL for your cloud and to provide security certificates.
- Update `servers.yml` to provide information about your servers.

### **networks.yml**

You will need to allocate site specific CIDRs and VLANs for these networks and update these values in the `networks.yml` file. The example models define the following networks:

Network	CIDR	VLAN ID	Tagged / Untagged
External API	10.0.1.0/24	101	Tagged

<b>Network</b>	<b>CIDR</b>	<b>VLAN ID</b>	<b>Tagged / Untagged</b>
External VM	Addresses configured by Neutron, leave blank in the file.	102	Tagged
Guest	10.1.1.0/24	103	Tagged
Management	192.168.10.0/24	100	Untagged

You will need to edit this file to provide your local values for these networks.

The CIDR for the External VM network is configured separately using the Neutron API. (For instructions, see the section called “Creating an External Network”.) You will only specify its VLAN ID during the installation process.

The Management network is shown as untagged. This is required if you are using this network to PXE install the operating system on the cloud nodes.

The example networks.yml file is shown below. Modify the bolded fields to reflect your site values.

```

networks:
  #
  # This example uses the following networks
  #
  # Network      CIDR          VLAN
  # -----      ----          -----
  # External API 10.0.1.0/24   101 (tagged)
  # External VM   see note 1    102 (tagged)
  # Guest         10.1.1.0/24   103 (tagged)
  # Management    192.168.10.0/24 100 (untagged)
  #
  # Notes:
  # 1. Defined as part of Neutron configuration
  #
  # Modify these values to match your environment
  #
  - name: EXTERNAL-API-NET
    vlanid: 101
    tagged-vlan: true
    cidr: 10.0.1.0/24
    gateway-ip: 10.0.1.1
    network-group: EXTERNAL-API

  - name: EXTERNAL-VM-NET
    vlanid: 102
    tagged-vlan: true
    network-group: EXTERNAL-VM

  - name: GUEST-NET
    vlanid: 103
    tagged-vlan: true
    cidr: 10.1.1.0/24
    gateway-ip: 10.1.1.1
    network-group: GUEST

```

```
- name: MANAGEMENT-NET
  vlanid: 100
  tagged-vlan: false
  cidr: 192.168.10.0/24
  gateway-ip: 192.168.10.1
  network-group: MANAGEMENT
```

## nic\_mappings.yml

This file maps Ethernet port names to specific bus slots. Due to inherent race conditions associated with multiple PCI device discovery there is no guarantee that Ethernet devices will be named as expected by the operating system, and it is possible that different port naming will exist on different servers with the same physical configuration.

To provide a deterministic naming pattern, the input model supports an explicit mapping from PCI bus address to a user specified name. HPE Helion OpenStack uses the prefix **hed** (Helion Ethernet Device) to name such devices to avoid any name clashes with the **eth** names assigned by the operating system.

The example `nic_mappings.yml` file is shown below.

`nic-mappings:`

```
- name: HP-DL360-4PORT
  physical-ports:
    - logical-name: hed1
      type: simple-port
      bus-address: "0000:07:00.0"

    - logical-name: hed2
      type: simple-port
      bus-address: "0000:08:00.0"

    - logical-name: hed3
      type: simple-port
      bus-address: "0000:09:00.0"

    - logical-name: hed4
      type: simple-port
      bus-address: "0000:0a:00.0"

- name: MY-2PORT-SERVER
  physical-ports:
    - logical-name: hed3
      type: simple-port
      bus-address: "0000:04:00.0"

    - logical-name: hed4
      type: simple-port
      bus-address: "0000:04:00.1"
```

This defines two sets of NIC mappings, representing two different physical server types. The name of each mapping is used as a value in the `servers.yml` file to associate each server with its required mapping. This enables the use of different server models or servers with different network hardware.

Each mapping lists a set of ports with the following information:

- **Logical name** - HPE Helion OpenStack uses the form hedN.
- **Type** - Only simple-port types are supported in HPE Helion OpenStack 5.0.
- **Bus-address** - The PIC bus address of the port.

The PCI bus address can be found using the `lspci` command on one of the servers. This command can produce a lot of output, so you can use the following command which will limit the output to list Ethernet class devices only:

```
sudo lspci -D |grep -i net
```

Here is an example output:

```
$ sudo lspci -D |grep -i net
0000:02:00.0 Ethernet controller: Broadcom Corporation NetXtreme BCM5719 Gigabit E
0000:02:00.1 Ethernet controller: Broadcom Corporation NetXtreme BCM5719 Gigabit E
0000:02:00.2 Ethernet controller: Broadcom Corporation NetXtreme BCM5719 Gigabit E
0000:02:00.3 Ethernet controller: Broadcom Corporation NetXtreme BCM5719 Gigabit E
0000:04:00.0 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Ne
0000:04:00.1 Ethernet controller: Intel Corporation 82599ES 10-Gigabit SFI/SFP+ Ne
```

To localize this file, replace the mapping names with the names of your choice and enumerate the ports as required.

## net\_interfaces.yml

This file is used to define how the network interfaces are to be configured. The example reflects the slightly different configuration of controller, compute nodes, and VSA nodes.

If network bonding is to be used, this file specifies how bonding is to be set up. It also specifies which networks are to be associated with each interface.

The example uses a bond of interfaces hed3 and hed4. You only need to modify this file if you have mapped your physical ports to different names, or if you need to modify the bond options.

The section of configuration file is shown below, which will create a bonded interface using the named hed3 and hed4 NIC mappings described in the previous section.

```
- name: CONTROLLER-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed3
      provider: linux
      devices:
        - name: hed3
        - name: hed4
  network-groups:
```

- EXTERNAL-API
- EXTERNAL-VM
- GUEST
- MANAGEMENT

If your system cannot support bonding, then you can modify this specification to specify a non-bonded interface, for example using device `hed3`:

```
- name: CONTROLLER-INTERFACES
  network-interfaces:
    - name: hed3
      device:
        name: hed3
  network-groups:
    - EXTERNAL-API
    - EXTERNAL-VM
    - GUEST
    - MANAGEMENT
```

## **network\_groups.yml**

This file defines the networks groups used in your cloud. A network-group defines the traffic separation model, and all of the properties that are common to the set of L3 networks that carry each type of traffic. They define where services and load balancers are attached to the network model and the routing within that model.

In this example, the following network groups are defined:

- **EXTERNAL-API** - This network group is used for external IP traffic to the cloud. In addition, it defines:
  - The characteristics of the load balancer to be used for the external API.
  - The Transport Layer Security (TLS) attributes.
- **EXTERNAL-VM** - Floating IPs for virtual machines are created on this network group. This is identified by the tag value `neutron.l3_agent.external_network_bridge`.
- **GUEST** - Tenant VxLAN traffic is carried on this network group. This is identified by the tag value `neutron.networks.vxlan`.
- **MANAGEMENT** - This is the default network group for traffic between service components in the cloud. In addition, it defines:
  - An internal load balancer is defined on this network group for managing internal and administrative API requests.

Most of the values in this file should be left unmodified if you are using the network model defined by the example. More complex modifications are supported but are outside the scope of this document.

However, the values related to the external API network are site-specific and need to be modified:

- Provide an external URL for the cloud.
- Provide the name of the security certificate to use.

The example `network_groups.yml` file is shown below, modify the bolded fields to reflect your site values.

```
# External API
#
# This is the network group that users will use to
# access the public API endpoints of your cloud
#
- name: EXTERNAL-API
  hostname-suffix: extapi

load-balancers:
  - provider: ip-cluster
    name: extlb
    # If external-name is set then public urls in keystone
    # will use this name instead of the IP address
    # You must either set this to a name that can be resolved
    # in your network
    # or comment out this line to use IP addresses
external-name:

tls-components:
  - default
roles:
  - public
cert-file: my-public-kvm-vsa-cert
```

The above bolded sections as follows:

**external-name** - The external name defines how the public URLs will be registered in Keystone. Users of your cloud will need to be able to resolve this URL to access the cloud APIs, and if you are using the TLS, the name must match the certificate used.

Because this value is difficult to change after initial deployment, this value is left blank in the supplied example which prevents the configuration processor from running until a value has been supplied. If you want to register the public URLs as IP addresses instead of a name, then you can comment out this line.

**cert-file** - Provide the name of the file located in `~/helion/my_cloud/config/tls/certs/` that will be used for your cloud endpoints. As shown above, this can be either a single certificate for all endpoints or a default certificate file and a set of service-specific certificate files.

**tls-components** - If you do not want to use a TLS for the public URLs then change the entry that says `tls-components` to `components`.

## servers.yml

This file is where you provide the details of the physical servers that make up your cloud. There are two sections to this file: `baremetal` and `servers`:

```
baremetal:
  # NOTE: These values need to be changed to match your environment.
  # Define the network range that contains the ip-addr values for
```

```
# the individual servers listed below.  
subnet: 192.168.10.0  
netmask: 255.255.255.0
```

The two values in this section are used to configure cobbler for operating system installation and must match the network values for the addresses given for the servers.

The servers section below provides the details of each individual server. For example, here are the details for the first controller:

```
servers:  
  
# Controllers  
- id: controller1  
  ip-addr: 192.168.10.3  
  role: CONTROLLER-ROLE  
  server-group: RACK1  
  nic-mapping: HP-DL360-4PORT  
  mac-addr: b2:72:8d:ac:7c:6f  
  ilo-ip: 192.168.9.3  
  ilo-password: password  
  ilo-user: admin
```

Here is a description of each of the above bolded sections:

**id** - A name you provide to uniquely identify a server. This can be any string which makes sense in your context, such as an asset tag, descriptive name, etc. The system will use this value to remember how the server has been allocated.

**ip-addr** - The IP address that the system will use for SSH connections to the server for deployment and configuration changes. This address must be in the IP range of one of the networks in the model. In the example, the servers are provided with addresses from the MANAGEMENT network.

**role** - A string that refers to an entry in `server_roles.yml` that tells the system how to configure the disks and network interfaces for this server. Roles are also used to define which servers can be used for specific purposes. Adding and changing roles is beyond the scope of this walkthrough - for more information, see the section called “Ring Specifications in the Input Model”.

**server-group** - Tells the system how this server is physically related to networks and other servers. Server groups are used to ensure that servers in a cluster are selected from different physical groups. The example provides a set of server groups that divide the servers into three sets called **RACK1**, **RACK2**, and **RACK3**. Modifying the server group structure is beyond the scope of this walkthrough - for more information, see the section called “NIC Mappings”.

**nic-mapping** - The name of a network port mapping definition (for more information, see ). You need to set this to the mapping that corresponds to this server.

**mac-addr** - The MAC address of the interface associated with this server that will be used for PXE boot.

**ilo-ip** - The IP address of the iLO or IPMI port for this server.

**ilo-user and ilo-password** - The login details used to access the iLO or IPMI port of this server. The iLO password value can be provided as an OpenSSL encrypted string. (For instructions on how to generate encrypted passwords, see Chapter 10, *Installing Mid-scale and Entry-scale KVM*.

# Customizing the Input Model

This section covers additional changes that you can make to further adapt the example to your environment:

- Update `disks_controller.yml` to add additional disk capacity to your controllers.
- Update `disks_vsa.yml` to add additional disk capacity to your VSA servers.
- Update `disks_compute.yml` to add additional disk capacity to your compute servers.

## `disks_controller.yml`

The disk configuration of the controllers consists of two sections: a definition of a volume group that provides a number of file-systems for various subsystems, and device-group that provides disk capacity for Swift.

## File Systems Storage

The root volume group (`hlm-vg`) is divided into a number of logical volumes that provide separate file systems for the various services that are co-hosted on the controllers in the entry-scale examples. The capacity of each file system is expressed as a percentage of the overall volume group capacity. Because not all file system usage scales linearly, two different disk configurations are provided:

- **CONTROLLER-DISKS** - Based on a 512 GB root volume group.
- **CONTROLLER-1TB-DISKS** - Provides a higher percentage of space for the logging service.

As supplied, the example uses the smaller disk model. To use the larger disk model you need to modify the `disk-models` parameter in the `server_roles.yml` file, as shown below:

```
server-roles:  
  
  - name: CONTROLLER-ROLE  
    interface-model: CONTROLLER-INTERFACES  
    disk-model: CONTROLLER-1TB-DISKS
```

To add additional disks to the root volume group, you need to modify the volume group definition in whichever disk model you are using. The following example shows adding an additional disk, `/dev/sdd` to the `disks_controller.yml` file:

```
disk-models:  
  - name: CONTROLLER-DISKS  
  
    volume-groups:  
      - name: hlm-vg  
        physical-volumes:  
  
          # NOTE: 'sda_root' is a templated value. This value is checked in  
          # os-config and replaced by the partition actually used on sda  
          #e.g. sdal or sda5  
          - /dev/sda_root
```

- /dev/sdd

## Swift Storage

Swift storage is configured as a device-group and has a syntax that allows disks to be allocated to specific rings. In the example, two disks are allocated to Swift to be shared by the account, container, and object-0 rings.

```
device-groups:
  - name: swiftobj
    devices:
      - name: /dev/sdb
      - name: /dev/sdc
      # Add any additional disks for swift here
      # -name: /dev/sdd
      # -name: /dev/sde
    consumer:
      name: swift
      attrs:
        rings:
          - account
          - container
          - object-0
```

For instruction to configure additional Swift storage, see the section called “Allocating Disk Drives for Object Storage”.

## disks\_vsa.yml

VSA storage is configured as a device-group and has a syntax that allows disks to be allocated for data storage or for adaptive optimization (caching). As a best practice, you should use solid state drives for adaptive optimization. The example disk configuration for VSA nodes has two disks, one for data and one of adaptive optimization (AO”).

```
device-groups:
  - name: vsa-data
    consumer:
      name: vsa
      usage: data
    devices:
      - name: /dev/sdc
  - name: vsa-cache
    consumer:
      name: vsa
      usage: adaptive-optimization
    devices:
      - name: /dev/sdb
```

Additional capacity can be added by adding more disks to the vsa-data device group. Similarly, caching capacity can be increased by adding more high speed storage devices to the vsa-cache device group.

## disks\_compute.yml

The example disk configuration for compute nodes consists of two volume groups: one for the operating system and one for the ephemeral storage for virtual machines, with one disk allocated to each.

Additional virtual machine ephemeral storage capacity can be configured by adding additional disks to the vg-comp volume group. The following example shows the addition of two more disks, /dev/sdc and /dev/sdd, to the `disks_compute.yml` file:

```
- name: vg-comp
  physical-volumes:
    - /dev/sdb
    - /dev/sdc
    - /dev/sdd
  logical-volumes:
    - name: compute
      size: 95%
      mount: /var/lib/nova
      fstype: ext4
      mkfs-opt: -O large_file
```

## VSA with or without Adaptive Optimization (AO)

VSA may be deployed with adaptive optimization (AO) or without AO. AO allows built-in storage tiering for VSA. While deploying VSA with or without AO you must ensure to use the appropriate disk input model.

If you are using VSA with AO, you will have an extra device group section where the usage is identified as adaptive-optimization as described in the following example:

```
Additional disks can be added if available
device_groups:
  - name: vsa-data
    consumer:
      name: vsa
      usage: data
    devices:
      - name: /dev/sdc
  - name: /dev/sdd
  - name: /dev/sde
  - name: /dev/sdf

  - name: vsa-cache
    consumer:
      name: vsa
      usage: adaptive-optimization
    devices:
      - name: /dev/sdb
```

VSA without AO consists of only data disks as described in the following example:

```
Additional disks can be added if available
device_groups:
  - name: vsa-data
    consumer:
      name: vsa
      usage: data
    devices:
      - name: /dev/sdc
      - name: /dev/sdd
      - name: /dev/sde
      - name: /dev/sdf
```

It is recommended to use SSD disk for AO.

### Note

A single VSA node can have a maximum of seven raw disks (excluding the operating system disks) attached to it, which is defined in the disk input model for your VSA nodes. It is expected that no more than seven disks are specified (including Adaptive Optimization disks) per VSA node. For example, if you want to deploy VSA with two disks for Adaptive Optimization then your disk input model should not specify more than five raw disks for data and two raw disks for Adaptive Optimization. Exceeding the disk limit causes VSA deployment failure.

## Creating Multiple VSA Clusters

The HPE Helion OpenStack 5.0 input model comes with one cluster and three VSA nodes. This is the default configuration available in the input model, but the input model allows you to create multiple VSA clusters of same or different types by modifying the YAML file.

### Tip

The term **add** and **update** in the document means editing the respective YAML files to add or update the configurations/values.

### Prerequisites

You must have a minimum of three nodes per VSA cluster.

## Cloud Configuration Changes to Create More Than One Cluster

In this cloud configuration, you can modify the YAML files for a same set of disks:

1. Add new nodes in the `servers.yml` file with a unique name and `node_id` for each cluster.

Example: In the following example we are adding one more cluster. Similarly, you can keep adding clusters based on your requirements.

The following `servers.yml` file lists six nodes for two clusters:

```
- id: vsa1
  ip-addr: 192.168.61.15
  role: VSA-ROLE
```

```
server-group: RACK1
nic-mapping: HP-BL460c-4PORT
ilo-ip: 10.1.192.232
ilo-password: gone2far
ilo-user: Administrator
mac-addr: 5C:B9:01:78:8C:B0

- id: vsa2
  ip-addr: 192.168.61.16
  role: VSA-ROLE
  server-group: RACK2
  nic-mapping: HP-BL460c-4PORT
  ilo-ip: 10.1.192.233
  ilo-password: gone2far
  ilo-user: Administrator
  mac-addr: 5C:B9:01:78:0E:30

- id: vsa3
  ip-addr: 192.168.61.17
  role: VSA-ROLE
  server-group: RACK3
  nic-mapping: HP-BL460c-4PORT
  ilo-ip: 10.1.192.234
  ilo-password: gone2far
  ilo-user: Administrator
  mac-addr: 5C:B9:01:78:2D:00

- id: vsa4
  ip-addr: 192.168.62.18
  role: VSA-ROLE-1
  server-group: RACK1
  nic-mapping: HP-BL460c-4PORT
  ilo-ip: 10.1.193.232
  ilo-password: gone2far
  ilo-user: Administrator
  mac-addr: 5C:B9:01:78:8C:B0

- id: vsa5
  ip-addr: 192.168.63.19
  role: VSA-ROLE-1
  server-group: RACK2
  nic-mapping: HP-BL460c-4PORT
  ilo-ip: 10.1.194.233
  ilo-password: gone2far
  ilo-user: Administrator
  mac-addr: 5C:B9:01:78:0E:30

- id: vsa6
  ip-addr: 192.168.64.20
  role: VSA-ROLE-1
  server-group: RACK3
  nic-mapping: HP-BL460c-4PORT
  ilo-ip: 10.1.195.234
  ilo-password: gone2far
```

```
ilo-user: Administrator
mac-addr: 5C:B9:01:78:2D:00
```

2. Add a new resource in the `control_plane.yml` file with the name, resource-prefix, and server-role.

Example: The following `control_plane.yml` file contains the information of the newly added resource nodes:

```
resources:
  - name: vsa
    resource-prefix: vsa
    server-role: ROLE-VSA
    allocation-policy: strict
    min-count: 0
    service-components:
      - ntp-client
      - vsa

  - name: vsa1
    resource-prefix: vsa1
    server-role: ROLE-VSA-1
    allocation-policy: strict
    min-count: 0
    service-components:
      - ntp-client
      - vsa
```

The control plane has the following fields:

<code>name</code>	The name assigned for the cluster. In the above example <b>vsa</b> and <b>vsa1</b> .
<code>resource-prefix</code>	The prefix of that resource cluster.
<code>server-role</code>	The role must be unique for each cluster.

3. Update `server_roles.yml` with new VSA nodes.

Example: In the following `server_roles.yml` file, new VSA nodes are added/updated:

```
server-roles:
  - name: ROLE-VSA
    interface-model: INTERFACE_SET_VSA
    disk-model: DISK_SET_VSA

  - name: ROLE-VSA-1
    interface-model: INTERFACE_SET_VSA
    disk-model: DISK_SET_VSA
```

The server roles have the following fields:

<code>name</code>	The name assigned to the cluster. In the above example <b>vsa</b> and <b>vsa1</b> .
<code>disk-model</code>	The type of disk available for the clusters. It can be the same set of disks or a different set of disks.

In the above example, only one set of disk models  
is shown (for example:**DISK\_SET\_VSA**).

## Cloud Configuration Changes to Create Two Cluster with Different Set of Disks

You must edit the YAML files to create more than one cluster. In this cloud configuration, you can add or update the YAML files for a different set of disks, i.e., one with adoptive optimization (AO) enabled and another without adoptive optimization.

1. Add new nodes in `servers.yml` with a unique name.
2. Add a new resource node under `resources` in `control_plane.yml` with a unique name, resource-prefix, and server-role.
3. Modify the `disk_set` for AO and without AO. Ensure that you use different files and different `disk_set`.

### Note

In the HPE Helion OpenStack 5.0 input model you will find only a `disks_vsa.yml` file, which contains the information for AO and without AO. You must create a separate YAML file and enter the configuration information of AO in that file. For creating non AO disk, refer to the section called “VSA with or without Adaptive Optimization (AO)”.

4. Update `server_roles.yml` with new VSA nodes and appropriate `disk_set` used for that node.

Example: In the following `servers_roles.yml` file you can see both AO and without AO assigned for the node:

```
---
product:
  version: 2

server-roles:

  - name: ROLE-CONTROLLER
    interface-model: INTERFACE_SET_CONTROLLER
    disk-model: DISK_SET_CONTROLLER

  - name: ROLE-COMPUTE
    interface-model: INTERFACE_SET_COMPUTE
    disk-model: DISK_SET_COMPUTE

  - name: ROLE-VSA
    interface-model: INTERFACE_SET_VSA
    disk-model: DISK_SET_VSA

  - name: ROLE-VSA-1
    interface-model: INTERFACE_SET_VSA
    disk-model: DISK_SET_VSA_AO
```

## Note

If you have configured your cloud to have more than one cluster or n-clusters, remember to note down all the cluster IPs.

# Configuring a Separate iSCSI Network to use with VSA

This page describes the procedure to assign a separate iSCSI network to use with VSA nodes. You must configure controller and compute nodes along with VSA to use a separate iSCSI network.

Perform the following procedure to assign a separate iSCSI network:

1. Log in to the lifecycle manager.
2. Edit the following files at `~/helion/my_cloud/definition/data` to assign a separate iSCSI network to controller nodes, compute nodes, and VSA nodes:

## Note

The input YAML files need to be changed during the cloud deployment.

- a. `networks.yml`: Enter the name of the network-group as shown in the example below. In the following example, the name of the network-group is "ISCSI" and this name should remain consistent in other files too.

```
- name: NET_ISCSI
  vlanid: 3287
  tagged-vlan: true
  cidr: 172.16.13.0/24
  gateway-ip: 172.16.13.1
  network-group: ISCSI
```

- b. `net_interfaces.yml`: A new field (forced-network-groups) is added in this file, as shown in the sample below.

```
Interface-models
  - name: INTERFACE_SET_CONTROLLER
    network-interfaces:
      - name: BOND0
        device:
          name: bond0
        bond-data:
          options:
            mode: "802.3ad"
            miimon: 200
          provider: linux
          devices:
            - name: Port0_10G1
            - name: Port1_10G1
    network-groups:
      - MGMT
      - TENANT
```

```
forced-network-groups:  
  - ISCSI  
  
  - name: INTERFACE_SET_COMPUTE  
    network-interfaces:  
      - name: BOND0  
        device:  
          name: bond0  
        bond-data:  
          options:  
            mode: "802.3ad"  
            miimon: 200  
          provider: linux  
          devices:  
            - name: Port0_10G1  
            - name: Port1_10G1  
    network-groups:  
      - MGMT  
      - TENANT  
forced-network-groups:  
  - ISCSI  
  
  - name: INTERFACE_SET_VSA  
    network-interfaces:  
      - name: BOND0  
        device:  
          name: bond0  
        bond-data:  
          options:  
            mode: "802.3ad"  
            miimon: 200  
          provider: linux  
          devices:  
            - name: Port0_10G1  
            - name: Port1_10G1  
    network-groups:  
      - MGMT  
      - TENANT  
forced-network-groups:  
  - ISCSI
```

c. firewall\_rules.yml

```
firewall-rules  
  - name: PING  
    network-groups:  
      - MGMT  
      - TENANT  
      - EXTERNAL_API  
      - ISCSI  
      - SWIFT
```

d. network\_groups.yml

```
network-groups:  
  - name: ISCSI  
    hostname-suffix: iscsi  
    component-endpoints:  
      - vsa  
  
e. server_groups.yml  
  
server_groups.yml  
  networks:  
    #Define the Global networks shared across all the Racks  
    - NET_EXTERNAL_API  
    - NET_EXTERNAL_VM  
    - NET_TENANT  
    - NET_MGMT  
    - NET_SWIFT  
    - NET_ISCSI
```

3. Commit your changes.

```
git add -A  
git commit -m "Add Node <name>"
```

4. Run the configuration processor:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Run the following command to create a deployment directory.

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the site.yml playbook using the command below.

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts site.yml
```

## Note

If the iSCSI network is not explicitly configured on the controller nodes then boot from cinder volumes would fail.

---

# Chapter 12. Modifying Example Configurations for Object Storage using Swift

This section contains detailed descriptions about the Swift-specific parts of the input model. For example input models, see Chapter 10, *Example Configurations*. For general descriptions of the input model, see the section called “Networks”. In addition, the Swift ring specifications are available in the `~/helion/my_cloud/definition/data/swift/rings.yml` file.

Usually, the example models provide most of the data that is required to create a valid input model. However, before you start to deploy, you must do the following:

- Check the disk model used by your nodes and that all disk drives are correctly named and used as described in the section called “Swift Requirements for Device Group Drives”.
- Select an appropriate partition power for your rings. For more information, see the section called “Understanding Swift Ring Specifications”.

For further information, read these related pages:

## Object Storage using Swift Overview

### What is the Object Storage (Swift) Service?

The HPE Helion OpenStack Object Storage using Swift service leverages Swift which uses software-defined storage (SDS) layered on top of industry-standard servers using native storage devices. Swift presents an object paradigm, using an underlying set of disk drives. The disk drives are managed by a data structure called a "ring" and you can store, retrieve, and delete objects in containers using RESTful APIs.

HPE Helion OpenStack Object Storage using Swift provides a highly-available, resilient, and scalable storage pool for unstructured data. It has a highly-durable architecture, with no single point of failure. In addition, HPE Helion OpenStack includes the concept of cloud models, where the user can modify the cloud input model to provide the configuration required for their environment.

### Object Storage (Swift) Services

A Swift system consists of a number of services:

- Swift-proxy provides the API for all requests to the Swift system.
- Account and container services provide storage management of the accounts and containers.
- Object services provide storage management for object storage.

These services can be co-located in a number of ways. The following general pattern exists in the example cloud models distributed in HPE Helion OpenStack:

- The swift-proxy, account, container, and object services run on the same (PACO) node type in the control plane. This is used for smaller clouds or where Swift is a minor element in a larger cloud. This is the model seen in most of the entry-scale models.

- The swift-proxy, account, and container services run on one (PAC) node type in a cluster in a control plane and the object services run on another (OBJ) node type in a resource pool. This deployment model, known as the Entry-Scale Swift model, is used in larger clouds or where a larger Swift system is in use or planned. See the section called “Entry-scale Swift Model” for more details.

The Swift storage service can be scaled both vertically (nodes with larger or more disks) and horizontally (more Swift storage nodes) to handle an increased number of simultaneous user connections and provide larger storage space.

Swift is configured through a number of YAML files in the HPE Helion implementation of the OpenStack Object Storage (Swift) service. For more details on the configuration of the YAML files, see Chapter 12, *Modifying Example Configurations for Object Storage using Swift*.

## Allocating Proxy, Account, and Container (PAC) Servers for Object Storage

A Swift proxy, account, and container (PAC) server is a node that runs the swift-proxy, swift-account and swift-container services. It is used to respond to API requests and to store account and container data. The PAC node does not store object data.

This section describes the procedure to allocate PAC servers during the **initial** deployment of the system.

### To allocate Swift PAC servers

Perform the following steps to allocate PAC servers:

- Verify if the example input model already contains a suitable server role. The server roles are usually described in the `data/server_roles.yml` file. If the server role is not described, you must add a suitable server role and allocate drives to store object data. For instructions, see the section called “Creating Roles for Swift Nodes” and the section called “Allocating Disk Drives for Object Storage”.
- Verify if the example input model has assigned a cluster to Swift proxy, account, container servers. It is usually mentioned in the `data/control_plane.yml` file. If the cluster is not assigned, then add a suitable cluster. For instructions, see **FIXME**: the section called “Creating a Swift Proxy, Account, and Container (PAC) Cluster”.
- Identify the physical servers and their IP address and other detailed information.
  - You add these details to the servers list (usually in the `data/servers.yml` file).
  - As with all servers, you must also verify and/or modify the server-groups information (usually in `data/server_groups.yml`)

The only part of this process that is unique to Swift is the allocation of disk drives for use by the account and container rings. For instructions, see the section called “Allocating Disk Drives for Object Storage”.

## Allocating Object Servers

A Swift object server is a node that runs the swift-object service (**only**) and is used to store object data. It does not run the swift-proxy, swift-account, or swift-container services.

This section describes the procedure to allocate a Swift object server during the **initial** deployment of the system.

## To Allocate a Swift Object Server

Perform the following steps to allocate one or more Swift object servers:

- Verify if the example input model already contains a suitable server role. The server roles are usually described in the `data/server_roles.yml` file. If the server role is not described, you must add a suitable server role. For instructions, see the section called “Creating Roles for Swift Nodes”. While adding a server role for the Swift object server, you will also allocate drives to store object data. For instructions, see the section called “Allocating Disk Drives for Object Storage”.
- Verify if the example input model has a resource node assigned to Swift object servers. The resource nodes are usually assigned in the `data/control_plane.yml` file. If it is not assigned, you must add a suitable resource node. For instructions, see the section called “Creating Object Server Resource Nodes”.
- Identify the physical servers and their IP address and other detailed information. Add the details for the servers in either of the following YAML files and verify the server-groups information:
  - Add details in the servers list (usually in the `data/servers.yml` file).
  - As with all servers, you must also verify and/or modify the server-groups information (usually in the `data/server_groups.yml` file).

The only part of this process that is unique to Swift is the allocation of disk drives for use by the object ring. For instructions, see the section called “Allocating Disk Drives for Object Storage”.

## Creating Roles for Swift Nodes

To create roles for Swift nodes, you must edit the `data/server_roles.yml` file and add an entry to the server-roles list using the following syntax:

```
server-roles:  
.  
. .  
- name: <pick-a-name>  
  interface-model: <specify-a-name>  
  disk-model: <specify-a-name>
```

The fields for server roles are defined as follows:

<b>name</b>	Specifies a name assigned for the role. In the following example, <b>SWOBJ-ROLE</b> is the role name.
<b>interface-model</b>	You can either select an existing interface model or create one specifically for Swift object servers. In the following example <b>SWOBJ-INTERFACES</b> is used. For more information, see the section called “Understanding Swift Network and Service Requirements”.
<b>disk-model</b>	You can either select an existing model or create one specifically for Swift object servers. In the following example <b>SWOBJ-DISKS</b> is used. For more in-

formation, see the section called “Allocating Disk Drives for Object Storage”.

```
server-roles:  
.  
. .  
- name: SWOBJ-ROLE  
  interface-model: SWOBJ-INTERFACES  
  disk-model: SWOBJ-DISKS
```

## Allocating Disk Drives for Object Storage

The disk model describes the configuration of disk drives and their usage. The examples include several disk models. You must always review the disk devices before making any changes to the existing the disk model.

### Making Changes to a Swift Disk Model

There are several reasons for changing the disk model:

- If you have additional drives available, you can add them to the devices list.
- If the disk devices listed in the example disk model have different names on your servers. This may be due to different hardware drives. Edit the disk model and change the device names to the correct names.
- If you prefer a different disk drive than the one listed in the model. For example, if /dev/sdb and /dev/sdc are slow hard drives and you have SSD drives available in /dev/sdd and /dev/sde. In this case, delete /dev/sdb and /dev/sdc and replace them with /dev/sdd and /dev/sde.

#### Note

Disk drives must not contain labels or file systems from a prior usage. For more information, see the section called “Swift Requirements for Device Group Drives”.

#### Tip

The terms **add** and **delete** in the document means editing the respective YAML files to add or delete the configurations/values.

### Swift Consumer Syntax

The consumer field determines the usage of a disk drive or logical volume by Swift. The syntax of the consumer field is as follows:

```
consumer:  
  name: swift  
  attrs:  
    rings:  
      - name: <ring-name>  
      - name: <ring-name>  
      - etc...
```

The fields for consumer are defined as follows:

<b>name</b>	Specifies the service that uses the device group. A name field containing <b>swift</b> indicates that the drives or logical volumes are used by Swift.
<b>attrs</b>	Lists the rings that the devices are allocated to. It must contain a <b>rings</b> item.
<b>rings</b>	Contains a list of ring names. In the <b>rings</b> list, the name field is optional.

The following are the different configurations (patterns) of the proxy, account, container, and object services:

- Proxy, account, container, and object (PACO) run on same node type.
- Proxy, account, and container run on a node type (PAC) and the object services run on a dedicated object server (OBJ).

## Note

The proxy service does not have any rings associated with it.

Example: **PACO** - proxy, account, container, and object run on the same node type.

```
consumer:
  name: swift
  attrs:
    rings:
      - name: account
      - name: container
      - name: object-0
```

Example: **PAC** - proxy, account, and container run on the same node type.

```
consumer:
  name: swift
  attrs:
    rings:
      - name: account
      - name: container
```

Example: **OBJ** - Dedicated object server. The following example shows two Storage Policies (object-0 and object-1). For more information, see the section called “Designing Storage Policies”.

```
consumer:
  name: swift
  attrs:
    rings:
      - name: object-0
      - name: object-1
```

## Swift Device Groups

You may have several device groups if you have several different uses for different sets of drives.

The following example shows a configuration where one drive is used for account and container rings and the other drives are used by the object-0 ring:

```
device-groups:  
  
- name: swiftpac  
  devices:  
    - name: /dev/sdb  
  consumer:  
    name: swift  
    attrs:  
      - name: account  
      - name: container  
- name: swiftobj  
  devices:  
    - name: /dev/sdc  
    - name: /dev/sde  
    - name: /dev/sdf  
  consumer:  
    name: swift  
    attrs:  
      rings:  
        - name: object-0
```

## Swift Logical Volumes

### Caution

Be careful while using logical volumes to store Swift data. The data remains intact during an upgrade, but will be lost if the server is reimaged. If you use logical volumes you must ensure that you only reimage one server at a time. This is to allow the data from the other replicas to be replicated back to the logical volume once the reimage is complete.

Swift can use a logical volume. To do this, ensure you meet the requirements listed in the table below:

• <i>mount</i>	Do not specify these attributes.
• <i>mkfs-opt</i>	
• <i>fstype</i>	
• <i>name</i>	Specify both of these attributes.
• <i>size</i>	
• <i>consumer</i>	This attribute must have a <i>name</i> field set to <b>swift</b> .

Following is an example of Swift logical volumes:

```
...  
...  
- name: swift  
  size: 50%  
  consumer:  
    name: swift  
    attrs:
```

```
rings:  
- name: object-0  
- name: object-1
```

## Swift Requirements for Device Group Drives

To install and deploy, Swift requires that the disk drives listed in the devices list of the device-groups item in a disk model meet the following criteria (if not, the deployment will fail):

- The disk device must exist on the server. For example, if you add `/dev/sdX` to a server with only three devices, then the deploy process will fail.
- The disk device must be unpartitioned or have a single partition that uses the whole drive.
- The partition must not be labeled. For instructions, see .
- The XFS file system must not contain a file system label. For instructions, see .
- If the disk drive is already labeled as described above, the `swiftlm-drive-provision` process will assume that the drive has valuable data and will not use or modify the drive.

## Creating a Swift Proxy, Account, and Container (PAC) Cluster

If you already have a cluster with the server-role SWPAC-ROLE there is no need to proceed through these steps.

## Steps to Create a Swift Proxy, Account, and Container (PAC) Cluster

To create a cluster for Swift proxy, account, and container (PAC) servers, you must identify the control plane and node type/role:

1. In the `~/helion/my_cloud/definition/data/control_plane.yml` file, identify the control plane that the PAC servers are associated with.
2. Next, identify the node type/role used by the Swift PAC servers. In the following example, `server-role` is set to **SWPAC-ROLE**.

Add an entry to the `clusters` item in the `control-plane` section.

Example:

```
control-planes:  
- name: control-plane-1  
  control-plane-prefix: cp1  
  
  . . .  
clusters:  
  . . .  
- name: swpac1
```

```
cluster-prefix: c2
server-role: SWPAC-ROLE
member-count: 3
allocation-policy: strict
service-components:
  - ntp-client
  - swift-ring-builder
  - swift-proxy
  - swift-account
  - swift-container
  - swift-client
```

## Important

Please do not change the name of the cluster `swpac` as it may conflict with an existing cluster.  
A name such as `swpac1`, `swpac2` or `swpac3` would be advisable.

3. If you have more than three servers available that have the `SWPAC-ROLE` assigned to them, you must change `member-count` to match the number of servers.

For example, if you have four servers with a role of `SWPAC-ROLE`, then the `member-count` should be 4.

## Service Components

A Swift PAC server requires the following service components:

- `ntp-client`
- `swift-proxy`
- `swift-account`
- `swift-container`
- `swift-ring-builder`
- `swift-client`

## Creating Object Server Resource Nodes

To create a resource node for Swift object servers, you must identify the control plane and node type/role:

- In the `data/control_plane.yml` file, identify the control plane that the object servers are associated with.
- Next, identify the node type/role used by the Swift object servers. In the following example, `server-role` is set to **SWOBJ-ROLE**:

Add an entry to the `resources` item in the **control-plane**:

```
control-planes:
  - name: control-plane-1
    control-plane-prefix: cpl
```

```
region-name: region1
. . .
resources:
. . .
- name: swobj
  resource-prefix: swobj
  server-role: SWOBJ-ROLE
  allocation-policy: strict
  min-count: 0
  service-components:
    - ntp-client
    - swift-object
```

### Service Components

A Swift object server requires the following service components:

- `ntp-client`
- `swift-object`
- `swift-client` is optional; installs the `python-swiftclient` package on the server.

Resource nodes do not have a member count attribute. So the number of servers allocated with the **SWOBJ-ROLE** is the number of servers in the `data/servers.yaml` file with a server role of **SWOBJ-ROLE**.

## Understanding Swift Network and Service Requirements

This topic describes Swift's requirements for which service components must exist in the input model and how these relate to the network model. This information is useful if you are creating a cluster or resource node, or when defining the networks used by Swift. The network model allows many options and configurations. For smooth Swift operation, the following must be **true**:

- The following services must have a **direct** connection to the same network:
  - `swift-proxy`
  - `swift-account`
  - `swift-container`
  - `swift-object`
  - `swift-ring-builder`
- The `swift-proxy` service must have a **direct** connection to the same network as the `cluster-ip` service.
- The memcached service must be configured on a cluster of the control plane. In small deployments, it is convenient to run it on the same cluster as the horizon service. For larger deployments, with many nodes running the `swift-proxy` service, it is better to **co-locate** the `swift-proxy` and memcached

services. The `swift-proxy` and `swift-container` services must have a **direct** connection to the same network as the `memcached` service.

- The `swift-proxy` and `swift-ring-builder` service must be **co-located** in the same cluster of the control plane.
- The `ntp-client` service must be **present** on all Swift nodes.

## Understanding Swift Ring Specifications

In Swift, the ring is responsible for mapping data on particular disks. There is a separate ring for account databases, container databases, and each object storage policy, but each ring works similarly. The `swift-ring-builder` utility is used to build and manage rings. This utility uses a builder file to contain ring information and additional data required to build future rings. In HPE Helion OpenStack 5.0, you will use the cloud model to specify how the rings are configured and used. This model is used to automatically invoke the `swift-ring-builder` utility as part of the deploy process. (Normally, you will not run the `swift-ring-builder` utility directly.)

The rings are specified in the input model using the **configuration-data** key. The `configuration-data` in the `control-planes` definition is given a name that you will then use in the `swift_config.yml` file. If you have several control planes hosting Swift services, the ring specifications can use a shared `configuration-data` object, however it is considered best practice to give each Swift instance its own `configuration-data` object.

### Ring Specifications in HPE Helion OpenStack 2.x and 3.x

In HPE Helion OpenStack 2.x and 3.x, ring specifications were mentioned in the `~/helion/my_cloud/definition/data/swift/rings.yml` file. HPE Helion OpenStack 4.x continues to support ring specifications in that file. If you upgrade to HPE Helion OpenStack 4.x, you do not need to make any changes.

## Ring Specifications in the Input Model

In most models, the ring-specification is mentioned in the `~/helion/my_cloud/definition/data/swift/swift_config.yml` file. For example:

```
configuration-data:  
  - name: SWIFT-CONFIG-CP1  
    services:  
      - swift  
    data:  
      control_plane_rings:  
        swift-zones:  
          - id: 1  
            server-groups:  
              - AZ1  
          - id: 2  
            server-groups:  
              - AZ2  
          - id: 3  
            server-groups:  
              - AZ3  
      rings:  
        - name: account
```

```

display-name: Account Ring
min-part-hours: 16
partition-power: 12
replication-policy:
    replica-count: 3

- name: container
  display-name: Container Ring
  min-part-hours: 16
  partition-power: 12
  replication-policy:
    replica-count: 3

- name: object-0
  display-name: General
  default: yes
  min-part-hours: 16
  partition-power: 12
  replication-policy:
    replica-count: 3

```

The above sample file shows that the rings are specified using the configuration-data object **SWIFT-CONFIG-CPI** and has three rings as follows:

- **Account ring:** You must always specify a ring called **account**. The account ring is used by Swift to store metadata about the projects in your system. In Swift, a Keystone project maps to a Swift account. The `display-name` is informational and not used.
- **Container ring:** You must always specify a ring called **container**. The `display-name` is informational and not used.
- **Object ring:** This ring is also known as a storage policy. You must always specify a ring called **object-0**. It is possible to have multiple object rings, which is known as *storage policies*. The `display-name` is the name of the storage policy and can be used by users of the Swift system when they create containers. It allows them to specify the storage policy that the container uses. In the example, the storage policy is called **General**. There are also two aliases for the storage policy name: `GeneralPolicy` and `AnotherAliasForGeneral`. In this example, you can use `General`, `GeneralPolicy`, or `AnotherAliasForGeneral` to refer to this storage policy. The `aliases` item is optional. The `display-name` is required.
- **Min-part-hours, partition-power, replication-policy** and **replica-count** are described in the following section.

## Replication Ring Parameters

The ring parameters for traditional replication rings are defined as follows:

Parameter	Description
<code>replica-count</code>	Defines the number of copies of object created.  Use this to control the degree of resiliency or availability. The <code>replica-count</code> is normally set to <b>3</b> (i.e., Swift will keep three copies of accounts, containers, or objects). As a best practice, you should

Parameter	Description
	not decrease the value to lower than 3. And, if you want a higher resiliency, you can increase the value.
min-part-hours	<p>Changes the value used to decide when a given partition can be moved. This is the number of hours that the <code>swift-ring-builder</code> tool will enforce between ring rebuilds. On a small system, this can be as low as <b>1</b> (one hour). The value can be different for each ring.</p> <p>In the example above, the <code>swift-ring-builder</code> will enforce a minimum of 16 hours between ring rebuilds. However, this time is system-dependent so you will be unable to determine the appropriate value for <code>min-part-hours</code> until you have more experience with your system.</p> <p>A value of 0 (zero) is not allowed.</p> <p>In prior releases, this parameter was called <code>min-part-time</code>. The older name is still supported, however do not specify both <code>min-part-hours</code> and <code>min-part-time</code> in the same files.</p>
partition-power	The optimal value for this parameter is related to the number of disk drives that you allocate to Swift storage. As a best practice, you should use the same drives for both the account and container rings. In this case, the <code>partition-power</code> value should be the same. For more information, see the section called “Selecting a Partition Power”.
replication-policy	Specifies that a ring uses replicated storage. The duplicate copies of the object are created and stored on different disk drives. All replicas are identical. If one is lost or corrupted, the system automatically copies one of the remaining replicas to restore the missing replica.
default	The default value in the above sample file of ring-specification is set to <b>yes</b> , which means that the storage policy is enabled to store objects. For more information, see the section called “Designing Storage Policies”.

## Erasure Coded Rings

In the cloud model, a `ring-specification` is mentioned in the `~/helion/my_cloud/definition/data/swift/rings.yml` file. A typical erasure coded ring in this file looks like this:

```
- name: object-1
  display-name: EC_ring
  default: no
  min-part-hours: 16
  partition-power: 12
```

```

erasure-coding-policy:
  ec-type: jerasure_rs_vand
  ec-num-data-frags: 10
  ec-num-parity-frags: 4
  ec-object-segment-size: 1048576

```

The additional parameters are defined as follows:

Parameter	Description
ec-type	This is the particular erasure policy scheme that is being used. The supported ec_types in HPE Helion OpenStack 5.0 are: <ul style="list-style-type: none"> <li>• jerasure_rs_vand =&gt; Vandermonde Reed-Solomon encoding, based on Jerasure</li> </ul>
erasure-coding-policy	This line indicates that the object ring will be of type "erasure coding"
ec-num-data-frags	This indicated the number of data fragments for an object in the ring.
ec-num-parity-frags	This indicated the number of parity fragments for an object in the ring.
ec-object-segment-size	The amount of data that will be buffered up before feeding a segment into the encoder/decoder. The default value is 1048576.

When using an erasure coded ring, the number of devices in the ring must be greater than or equal to the total number of fragments of an object. For example, if you define an erasure coded ring with 10 data fragments and 4 parity fragments, there must be at least 14 (10+4) devices added to the ring.

When using erasure codes, for a PUT object to be successful it must store `ec_ndata + 1` fragment to achieve quorum. Where the number of data fragments (`ec_ndata`) is 10 then at least 11 fragments must be saved for the object PUT to be successful. The 11 fragments must be saved to different drives. To tolerate a single object server going down, say in a system with 3 object servers, each object server must have at least 6 drives assigned to the erasure coded storage policy. So with a single object server down, 12 drives are available between the remaining object servers. This allows an object PUT to save 12 fragments, one more than the minimum to achieve quorum.

Unlike replication rings, none of the erasure coded parameters may be edited after the initial creation. Otherwise there is potential for permanent loss of access to the data.

On the face of it, you would expect that an erasure coded configuration that uses a data to parity ratio of 10:4, that the data consumed storing the object is 1.4 times the size of the object just like the x3 replication takes x3 times the size of the data when storing the object. However, for erasure coding, this 10:4 ratio is not correct. The efficiency (ie. how much storage is needed to store the object) is very poor for small objects and improves as the object size grows. However, the improvement is not linear. If all of your files are less than 32K in size, erasure coding will take more space to store than the x3 replication.

## Selecting a Partition Power

When storing an object, the object storage system hashes the name. This hash results in a hit on a partition (so a number of different object names result in the same partition number). Generally, the partition is mapped to available disk drives. With a replica count of 3, each partition is mapped to three different disk

drives. The hashing algorithm used hashes over a fixed number of partitions. The partition-power attribute determines the number of partitions you have.

Partition power is used to distribute the data uniformly across drives in a Swift nodes. It also defines the storage cluster capacity. You must set the partition power value based on the total amount of storage you expect your entire ring to use.

You should select a partition power for a given ring that is appropriate to the number of disk drives you allocate to the ring for the following reasons:

- If you use a high partition power and have a few disk drives, each disk drive will have thousands of partitions. With too many partitions, audit and other processes in the Object Storage system cannot walk the partitions in a reasonable time and updates will not occur in a timely manner.
- If you use a low partition power and have many disk drives, you will have tens (or maybe only one) partition on a drive. The Object Storage system does not use size when hashing to a partition - it hashes the name.

With many partitions on a drive, a large partition is cancelled out by a smaller partition so the overall drive usage is similar. However, with very small numbers of partitions, the uneven distribution of sizes can be reflected in uneven disk drive usage (so one drive becomes full while a neighboring drive is empty).

An ideal number of partitions per drive is 100. If you know the number of drives, select a partition power that will give you approximately 100 partitions per drive. Usually, you install a system with a specific number of drives and add drives as needed. However, you cannot change the value of the partition power. Hence you must select a value that is a compromise between current and planned capacity.

## **Important**

If you are installing a small capacity system and you need to grow to a very large capacity but you cannot fit within any of the ranges in the table, please seek help from Professional Services to plan your system.

There are additional factors that can help mitigate the fixed nature of the partition power:

- Account and container storage represents a small fraction (typically 1 percent) of your object storage needs. Hence, you can select a smaller partition power (relative to object ring partition power) for the account and container rings.
- For object storage, you can add additional storage policies (i.e., another object ring). When you have reached capacity in an existing storage policy, you can add a new storage policy with a higher partition power (because you now have more disk drives in your system). This means that you can install your system using a small partition power appropriate to a small number of initial disk drives. Later, when you have many disk drives, the new storage policy can have a higher value appropriate to the larger number of drives.

However, when you continue to add storage capacity, existing containers will continue to use their original storage policy. Hence, the additional objects must be added to new containers to take advantage of the new storage policy.

Use the following table to select an appropriate partition power for each ring. The partition power of a ring cannot be changed, so it is important to select an appropriate value. This table is based on a replica count of 3. If your replica count is different, or you are unable to find your system in the table, then see for information of selecting a partition power.

The table assumes that when you first deploy Swift, you have a small number of drives (the minimum column in the table), and later you add drives.

## Note

- Use the total number of drives. For example, if you have three servers, each with two drives, the total number of drives is six.
- The lookup should be done separately for each of the account, container and object rings. Since account and containers represent approximately 1 to 2 percent of object storage, you will probably use fewer drives for the account and container rings (i.e., you will have fewer proxy, account, and container (PAC) servers) so that your object rings may have a higher partition power.
- The largest anticipated number of drives imposes a limit in the minimum drives you can have. (For more information, see .) This means that, if you anticipate significant growth, your initial system can be small, but under a certain limit. For example, if you determine that the maximum number of drives the system will grow to is 40,000, then use a partition power of 17 as listed in the table below. In addition, a minimum of 36 drives is required to build the smallest system with this partition power.
- The table assumes that disk drives are the same size. The actual size of a drive is not significant.

# Designing Storage Policies

Storage policies enable you to differentiate the way objects are stored.

Reasons to use storage policies include the following:

- Different types or classes of disk drive

You can use different drives to store various type of data. For example, you can use 7.5K RPM high-capacity drives for one type of data and fast SSD drives for another type of data.

- Different redundancy or availability needs

You can define the redundancy and availability based on your requirement. You can use a replica count of 3 for "normal" data and a replica count of 4 for "critical" data.

- Growing of cluster capacity

If the storage cluster capacity grows beyond the recommended partition power as described in the section called "Understanding Swift Ring Specifications".

- Erasure-coded storage and replicated storage

If you use erasure-coded storage for some objects and replicated storage for other objects.

Storage policies are implemented on a per-container basis. If you want a non-default storage policy to be used for a new container, you can explicitly specify the storage policy to use when you create the container. You can change which storage policy is the default. However, this does not affect existing containers. Once the storage policy of a container is set, the policy for that container cannot be changed.

The disk drives used by storage policies can overlap or be distinct. If the storage policies overlap (i.e., have disks in common between two storage policies), it is recommended to use the same set of disk drives

for both policies. But in the case where there is a partial overlap in disk drives, because one storage policy receives many objects, the drives that are common to both policies must store more objects than drives that are only allocated to one storage policy. This can be appropriate for a situation where the overlapped disk drives are larger than the non-overlapped drives.

## Specifying Storage Policies

There are two places where storage policies are specified in the input model:

- The attribute of the storage policy is specified in ring-specification in the `data/swift/rings.yml` file for a given region.
- When associating disk drives with specific rings in a disk model. This specifies which drives and nodes use the storage policy. In other word words, where data associated with a storage policy is stored.

A storage policy is specified similar to other rings. However, the following features are unique to storage policies:

- Storage policies are applicable to object rings only. The account or container rings cannot have storage policies.
- There is a format for the ring name: `object-<index>`, where index is a number in the range 0 to 9 (in this release). For example: `object-0`.
- The object-0 ring must always be specified.
- Once a storage policy is deployed, it should never be deleted. You can remove all disk drives for the storage policy, however the ring specification itself cannot be deleted.
- You can use the `display-name` attribute when creating a container to indicate which storage policy you want to use for that container.
- One of the storage policies can be the default policy. If you do not specify the storage policy then the object created in new container uses the default storage policy.
- If you change the default, only containers created later will have that changed default policy.

The following example shows three storage policies in use. Note that the third storage policy example is an erasure coded ring.

```
rings:  
  . . .  
  - name: object-0  
    display-name: General  
    default: no  
    min-part-hours: 16  
    partition-power: 12  
    replication-policy:  
      replica-count: 3  
  - name: object-1  
    display-name: Data  
    default: yes  
    min-part-hours: 16  
    partition-power: 20  
    replication-policy:  
      replica-count: 3
```

```

- name: object-2
  display-name: Archive
  default: no
  min-part-hours: 16
  partition-power: 20
  erasure-coded-policy:
    ec-type: jerasure_rs_vand
    ec-num-data-fragments: 10
    ec-num-parity-fragments: 4
    ec-object-segment-size: 1048576

```

## Designing Swift Zones

The concept of Swift zones allows you to control the placement of replicas on different groups of servers. When constructing rings and allocating replicas to specific disk drives, Swift will, where possible, allocate replicas using the following hierarchy so that the greatest amount of resiliency is achieved by avoiding single points of failure:

- Swift will place each replica on a different disk drive within the same server.
- Swift will place each replica on a different server.
- Swift will place each replica in a different Swift zone.

If you have three servers and a replica count of three, it is easy for Swift to place each replica on a different server. If you only have two servers though, Swift will place two replicas on one server (different drives on the server) and one copy on the other server.

With only three servers there is no need to use the Swift zone concept. However, if you have more servers than your replica count, the Swift zone concept can be used to control the degree of resiliency. The following table shows how data is placed and explains what happens under various failure scenarios. In all cases, a replica count of three is assumed and that there are a total of six servers.

Number of Swift Zones	Replica Placement	Failure Scenarios	Details
One (all servers in the same zone)	Replicas are placed on different servers. For any given object, you have no control over which servers the replicas are placed on.	One server fails	You are guaranteed that there are two other replicas.
		Two servers fail	You are guaranteed that there is one remaining replica.
		Three servers fail	1/3 of the objects cannot be accessed. 2/3 of the objects have three replicas.
Two (three servers in each Swift zone)	Half the objects have two replicas in Swift zone 1 with one replica in Swift zone 2. The other objects are reversed, with one replica in Swift zone 1 and two replicas in Swift zone 2.	One Swift zone fails	You are guaranteed to have at least one replica. Half the objects have two remaining replicas and the other half have a single replica.

<b>Number of Swift Zones</b>	<b>Replica Placement</b>	<b>Failure Scenarios</b>	<b>Details</b>
Three (two servers in each Swift zone)	Each zone contains a replica. For any given object, there is a replica in each Swift zone.	One Swift zone fails	You are guaranteed to have two replicas of every object.
		Two Swift zones fail	You are guaranteed to have one replica of every object.

The following sections show examples of how to specify the Swift zones in your input model.

## Using Server Groups to Specify Swift Zones

Swift zones are specified in the ring specifications using the server group concept. To define a Swift zone, you specify:

- An id - this is the Swift zone number
- A list of associated server groups

Server groups are defined in your input model. The example input models typically define a number of server groups. You can use these pre-defined server groups or create your own.

For example, the following three models use the example server groups CLOUD, AZ1, AZ2 and AZ3. Each of these examples achieves the same effect – creating a single Swift zone.

```

ring-specifications:
    - region: region1
      swift-zones:
        - id: 1
          server-groups:
            - CLOUD
          rings:
            ...
      ...
ring-specifications:
    - region: region1
      swift-zones:
        - id: 1
          server-groups:
            - AZ1
            - AZ2
            - AZ3
          rings:
            ...
server-groups:
    - name: ZONE_ONE
      server-groups:
        - AZ1
        - AZ2
        - AZ3
      ring-specifications:
        - region: region1
          swift-zones:

```

```
- id: 1
server-groups:
- ZONE_ONE
rings:
..
```

Alternatively, if you omit the `swift-zones` specification, a single Swift zone is used by default for all servers.

In the following example, three Swift zones are specified and mapped to the same availability zones that Nova uses (assuming you are using one of the example input models):

```
ring-specifications:
- region: region1
swift-zones:
- id: 1
server-groups:
- AZ1
- id: 2
server-groups:
- AZ2
- id: 3
server-groups:
- AZ3
```

In this example, it shows a datacenter with four availability zones which are mapped to two Swift zones. This type of setup may be used if you had two buildings where each building has a duplicated network infrastructure:

```
ring-specifications:
- region: region1
swift-zones:
- id: 1
server-groups:
- AZ1
- AZ2
- id: 2
server-groups:
- AZ3
- AZ4
```

## Specifying Swift Zones at Ring Level

Usually, you would use the same Swift zone layout for all rings in your system. However, it is possible to specify a different layout for a given ring. The following example shows that the account, container and object-0 rings have two zones, but the object-1 ring has a single zone.

```
ring-specifications:
- region: region1
swift-zones:
- id: 1
server-groups:
- AZ1
- id: 2
server-groups:
```

```
- AZ2
  rings
    - name: account
    ...
    - name: container
    ...
    - name: object-0
    ...
    - name: object-1
  swift-zones:
    - id: 1
  server-groups:
    - CLOUD
    ...

```

## Customizing Swift Service Configuration Files

HPE Helion OpenStack 5.0 enables you to modify various Swift service configuration files. The following Swift service configuration files are located on the lifecycle manager in the `~/helion/my_cloud/config/swift/` directory:

- `account-server.conf.j2`
- `container-reconciler.conf.j2`
- `container-server.conf.j2`
- `container-sync-realms.conf.j2`
- `object-expirer.conf.j2`
- `object-server.conf.j2`
- `proxy-server.conf.j2`
- `rsyncd.conf.j2`
- `swift.conf.j2`
- `swift-recon.j2`

There are many configuration options that can be set or changed, including **container rate limit** and **logging level**:

## Configuring Swift Container Rate Limit

The Swift container rate limit allows you to limit the number of PUT and DELETE requests of an object based on the number of objects in a container. For example, suppose the `container_ratelimit_x = r`. It means that for containers of size x, limit requests per second to r.

To enable container rate limiting:

1. Log in to the lifecycle manager.
2. Edit the DEFAULT section of `~/helion/my_cloud/config/swift/proxy-server.conf.j2`:

```
container_ratelimit_0
                      = 100 container_ratelimit_1000000 = 100
                      container_ratelimit_5000000 = 50
```

This will set the PUT and DELETE object rate limit to 100 requests per second for containers with up to 1,000,000 objects. Also, the PUT and DELETE rate for containers with between 1,000,000 and 5,000,000 objects will vary linearly from between 100 and 50 requests per second as the container object count increases.

3. Commit your changes to git:

```
cd ~/helion/hos/ansible git add -A
                               git commit -m "<commit message>"
```

4. Run the configuration processor:

```
cd ~/helion/hos/ansible
                               ansible-playbook -i hosts/localhost
                               config-processor-run.yml
```

5. Create a deployment directory:

```
cd ~/helion/hos/ansible
                               ansible-playbook -i hosts/localhost
                               ready-deployment.yml
```

6. Run the `swift-reconfigure.yml` playbook to reconfigure the Swift servers:

```
cd ~/scratch/ansible/next/hos/ansible
                               ansible-playbook -i hosts/verb_hosts
                               swift-reconfigure.yml
```

## Configuring Swift Account Server Logging Level

By default the Swift logging level is set to INFO. As a best practice, do not set the log level to DEBUG for a long period of time. Use it for troubleshooting issues and then change it back to INFO.

Perform the following steps to set the logging level of the `account-server` to DEBUG:

1. Log in to the lifecycle manager.

2. Edit the `DEFAULT` section of `~/helion/my_cloud/config/swift/account-server.conf.j2`:

```
[DEFAULT] . . log_level = DEBUG
```

3. Commit your changes to git:

```
cd ~/helion/hos/ansible git add -A
                               git commit -m "<commit message>"
```

4. Run the configuration processor:

```
cd ~/helion/hos/ansible
                               ansible-playbook -i hosts/localhost
                               config-processor-run.yml
```

5. Create a deployment directory:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost  
ready-deployment.yml
```

6. Run the `swift-reconfigure.yml` playbook to reconfigure the Swift servers:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts  
swift-reconfigure.yml
```

For more information, see .

---

# Chapter 13. Alternative Configurations

## Abstract

In HPE Helion OpenStack 5.0 there are alternative configurations that we recommend for specific purposes

## SLES Compute Nodes

net\_interfaces.yml

```
- name: SLES-COMPUTE-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          mimon: 200
          primary: hed1
        provider: linux
        devices:
          - name: hed1
          - name: hed2
  network-groups:
    - EXTERNAL-VM
    - GUEST
    - MANAGEMENT
```

servers.yml

```
- id: compute1
  ip-addr: 10.13.111.15
  role: SLES-COMPUTE-ROLE
  server-group: RACK1
  nic-mapping: DL360p_G8_2Port
  mac-addr: ec:b1:d7:77:d0:b0
  ilo-ip: 10.12.13.14
  ilo-password: *****
  ilo-user: Administrator
  distro-id: sles12sp2-x86_64
```

server\_roles.yml

```
- name: SLES-COMPUTE-ROLE
  interface-model: SLES-COMPUTE-INTERFACES
  disk-model: SLES-COMPUTE-DISKS
```

disk\_compute.yml

```
- name: SLES-COMPUTE-DISKS
  volume-groups:
```

```
- name: hlm-vg
  physical-volumes:
    - /dev/sda_root

  logical-volumes:
    # The policy is not to consume 100% of the space of each volume group.
    # 5% should be left free for snapshots and to allow for some flexibility.
    - name: root
      size: 35%
      fstype: ext4
      mount: /
    - name: log
      size: 50%
      mount: /var/log
      fstype: ext4
      mkfs-opt: -O large_file
    - name: crash
      size: 10%
      mount: /var/crash
      fstype: ext4
      mkfs-opt: -O large_file

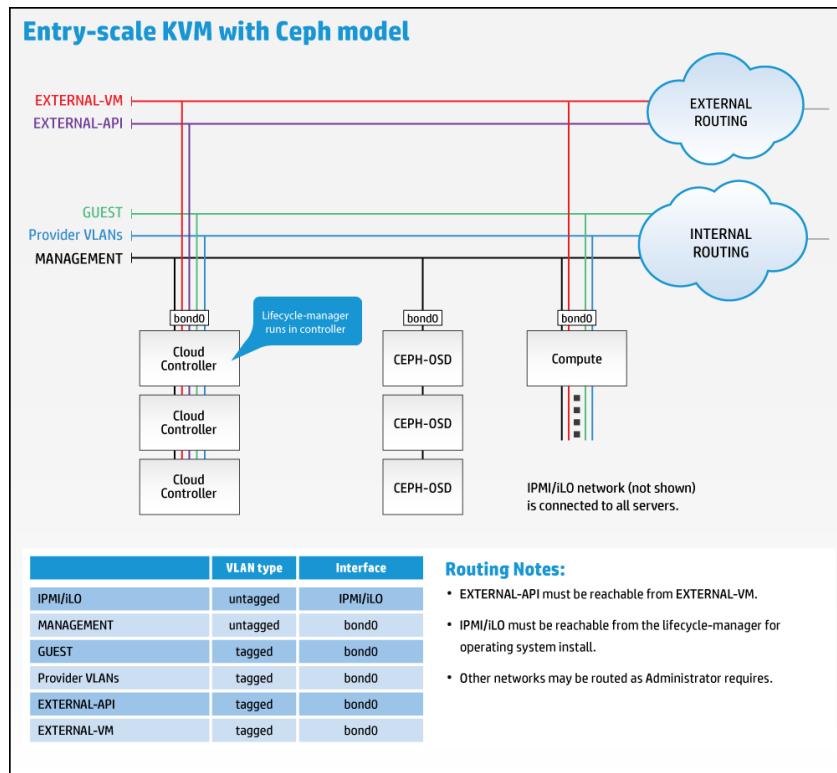
- name: vg-comp
  # this VG is dedicated to Nova Compute to keep VM IOPS off the OS disk
  physical-volumes:
    - /dev/sdb
  logical-volumes:
    - name: compute
      size: 95%
      mount: /var/lib/nova
      fstype: ext4
      mkfs-opt: -O large_file

control_plane.yml

control-planes:
- name: control-plane-1
  control-plane-prefix: cp1
  region-name: region1
.....
resources:
- name: sles-compute
  resource-prefix: sles-comp
  server-role: SLES-COMPUTE-ROLE
  allocation-policy: any
  min-count: 1
  service-components:
    - ntp-client
    - nova-compute
    - nova-compute-kvm
    - neutron-l3-agent
    - neutron-metadata-agent
    - neutron-openvswitch-agent
    - neutron-lbaasv2-agent
```

# Entry-scale KVM with Ceph Model

For smaller environments, Ceph can be altered to use a single-network model:



Download a high-resolution version [../../../../media/examples/entry\_scale\_kvm\_ceph\_lg.png]

# Entry-scale KVM with Ceph Model with Two Networks

HPE Helion OpenStack Ceph is a unified storage system for various storage use cases for an OpenStack-based cloud. It is highly reliable, easy to manage, and horizontally scalable as demand grows.

Ceph clients directly talk to OSD daemons for storage operations instead of client routing the request to a specific gateway as is commonly found in other storage solutions. OSD daemons perform data replication and participate in recovery activities. In general, a pool is configured with a replica count of three, causing daemons to transact three times the amount of client data over the cluster network. So, every 4 MB of write data is likely to result in 12 MB of data movement across Ceph clusters. Considering this network traffic, it is important to segregate Ceph data traffic, which can be primarily categorized into three segments:

- **Management traffic** - primarily includes all admin related operations such as pool creation, crush map modification, user creation, etc.
- **Client (data) traffic** - primarily includes client requests sent to OSD daemons.
- **Cluster (replication) traffic** - primarily includes replication and recovery data traffic among OSD daemons.

For a high performing cluster, the network configuration is important. Segregating the data traffic using multiple networks allows for this. For medium-size production environments we recommend to have a

cluster with at least two networks: a client data network (front-side) and a cluster (back-side) network. For larger production environments we recommend that you segregate all three network traffic types by utilizing three networks. This particular document shows you how to setup two networks but you can use the same principles to create three networks.

Also, segregating networks provides additional security as well because your cluster network does not need to be connected to the internet directly. This helps in preventing spoof attacks and allows the OSD daemons to keep communicating without intervention so that placement groups can be brought to active + clean state whenever required.

This model is a variant of the Entry-scale KVM with Ceph model. It is designed with two VLANs: a public (front-side) network and a cluster (back-side) network. This enables more options in regards to scaling.

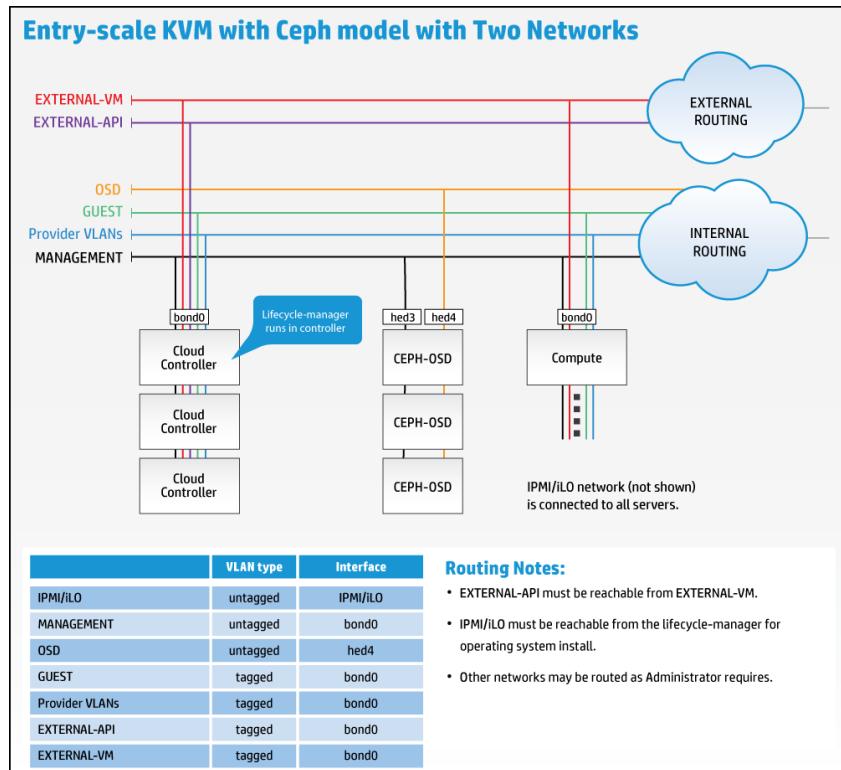
This model uses the following components:

- Three controller nodes, one KVM compute node, and three Ceph OSD nodes.
- The Ceph monitor component of the Ceph cluster is deployed on the controller nodes along with other OpenStack service components. This limits your cloud to three monitor nodes which should be suitable for most production environments.
- Allows two VLANs (i.e. management VLAN and OSD VLAN) which segregates Ceph client traffic from Ceph cluster traffic. The management network will be used to carry cloud management data, such as RabbitMQ, HOPS, and database traffic, Ceph management data, such as pool creation, as well as client data traffic, such as cinder-volume writing blocks to Ceph storage pools. The Ceph cluster network will be dedicated for OSD daemons and will be used to carry replication traffic.
- A single compute node is initially provided with this example configuration. If additional compute capacity is required then further compute nodes can be added to the configuration by adding more nodes to the compute resource plane. The same applies to OSD nodes as well. Three OSD nodes are initially provided with this example configuration. If additional OSD capacity is required then further OSD nodes can be added to the configuration by adding more nodes to the OSD resource plane.

The table below lists out the key characteristics needed per server role for this configuration.

Server role	Quantity	Compute Requirement	Network Requirement
Controller	3	2x 10 core 2.66 GHz 96 - 128 GB RAM	2x 10Gb Dual Port NIC
Compute (KVM hypervisor)	1 (minimum)	2x 12 core 2.66 GHz (ES-2690v3) Intel Xeon 256 GB RAM	1x 10Gb Dual Port NIC
OSD	3 (minimum)	RAM is dependent upon the number of disks. 1 GB per TB of disk capacity is recommended.	1x 10Gb Dual Port NIC

This diagram below illustrates the physical networking used in this configuration.



Download a high-resolution version [[..../media/hos.docs/entry\\_scale\\_kvm\\_ceph\\_two\\_network\\_lg.png](#)]

This configuration is based on the entry-scale-kvm-ceph cloud input model which is included with the HPE Helion OpenStack distro. You will need to make the changes outlined below prior to the deployment of your Ceph cluster with two networks. Note that if you already have a Ceph cluster deployed with a single network these steps cannot be used to migrate to a dual-network setup. The recommendation in these cases will be that you make a clean installation which will result in the loss of your existing data unless you make arrangements to have it backed up beforehand.

`Nic_mappings.yml` Ensure that your baremetal server NIC interfaces are correctly specified in the `~/helion/my_cloud/definition/data/nic_mappings.yml` file and that they meet the server requirements.

Here is an example with notes in-line:

`nic-mappings:`

```
## NIC specification for controller nodes. A bonded interface is
## used for the management network.
- name: DL360p_4PORT
  physical-ports:
    - logical-name: hed1
      type: simple-port
      bus-address: "0000:07:00.0"
    - logical-name: hed2
      type: simple-port
      bus-address: "0000:08:00.0"
    - logical-name: hed3
```

```
        type: simple-port
        bus-address: "0000:09:00.0"

    - logical-name: hed4
      type: simple-port
      bus-address: "0000:0a:00.0"

## NIC specification for compute and OSD nodes should be
## of this type.
- name: MY-2PORT-SERVER
  physical-ports:
    - logical-name: hed3
      type: simple-port
      bus-address: "0000:04:00.0"

    - logical-name: hed4
      type: simple-port
      bus-address: "0000:04:00.1"
```

`Net_interfaces.yml` Define a new interface set for your OSD interfaces in the `~/helion/my_cloud/definition/data/net_interfaces.yml` file.

Ensure that the appropriate NIC is configured to both the Management and OSD network groups, indicated below:

```
- name: OSD-INTERFACES
  network-interfaces:
    - name: ETH3
      device:
        name: hed3
      network-groups:
        - MANAGEMENT
    - name: ETH4
      device:
        name: hed4
      network-groups:
        - OSD
```

`Network_groups.yml` Define the OSD network group in the `~/helion/my_cloud/definition/data/network_groups.yml` file:

```
#
# OSD
#
# This is the network group that will be used for
# internal traffic of cluster among OSDs.
#
- name: OSD
  hostname-suffix: osd

  component-endpoints:
    - ceph-osd-internal
```

`Networks.yml` Define the OSD VLAN in the `~/helion/my_cloud/definition/data/networks.yml` file:

```
- name: OSD-NET
  vlanid: 112
  tagged-vlan: false
  cidr: 10.0.1.0/24
  gateway-ip: 10.0.1.1
  network-group: OSD
```

`Server_groups.yml` Add the OSD network to the server groups in the `~/helion/my_cloud/definition/data/server_groups.yml` file, indicated by the bold portion below:

```
- name: CLOUD
  server-groups:
    - AZ1
    - AZ2
    - AZ3
  networks:
    - EXTERNAL-API-NET
    - EXTERNAL-VM-NET
    - GUEST-NET
    - MANAGEMENT-NET
    - OSD-NET
```

`Firewall_rules.yml` Modify the firewall rules in the `~/helion/my_cloud/definition/data/firewall_rules.yml` file to allow OSD nodes to be pingable via the OSD network, indicated by the bold portion below:

```
- name: PING
  network-groups:
    - MANAGEMENT
    - GUEST
    - EXTERNAL-API
    - OSD
  rules:
    # open ICMP echo request (ping)
    - type: allow
      remote-ip-prefix: 0.0.0.0/0
      # icmp type
      port-range-min: 8
      # icmp code
      port-range-max: 0
      protocol: icmp
```

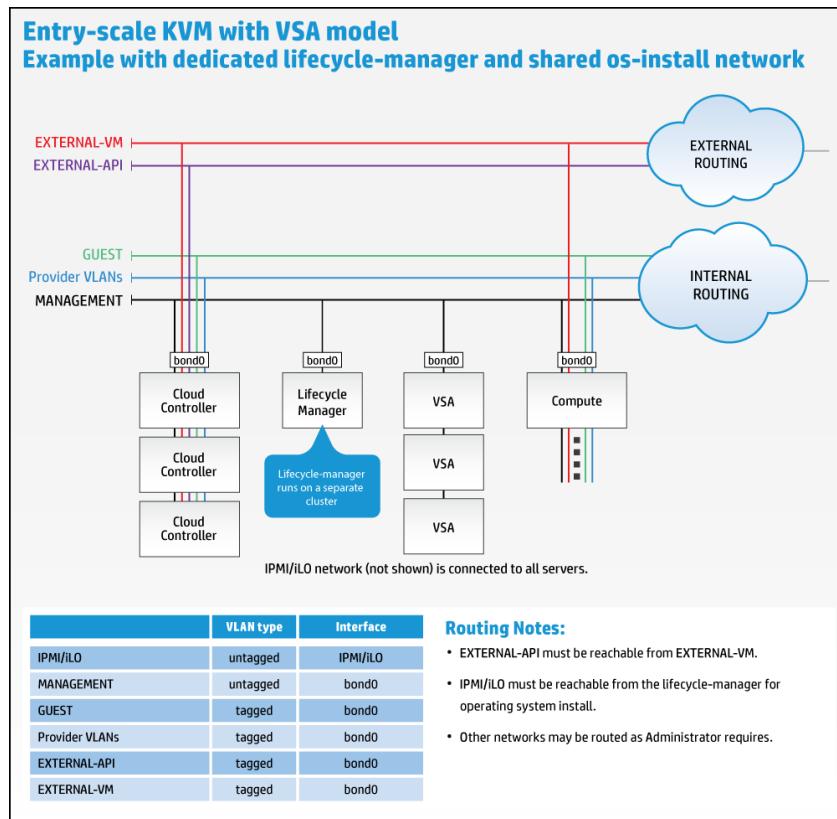
Edit the `README.html` and `README.md` files. You can edit the `~/helion/my_cloud/definition/README.html` and `~/helion/my_cloud/definition/README.md` files to reflect the OSD network group information if you wish. This change does not have any semantic implication and only assists with the readability of your model.

## Using a Dedicated Lifecycle Manager Node

All of the example configurations included host the lifecycle manager on the first controller nodes. It is also possible to deploy this service on a dedicated node. A typical use case for wanting to run the dedicated lifecycle manager is to be able to test the deployment of different configurations without having to re-

install the first server. Some administrators might also prefer the additional security of keeping all of the configuration data on a separate server from those that users of the cloud connect to (although all of the data can be encrypted and SSH keys can be password protected).

Here is a graphical representation of what this setup would look like:



Download a high-resolution version [../../../../media/examples/entry\_scale\_kvm\_vsa\_shared\_lg.png]

## Specifying a dedicated lifecycle manager in your input model

In order to specify a dedicated lifecycle manager in your input model, make the following edits to your configuration files.

### Important

The indentation of each of the input files is important and will cause errors if not done correctly. Use the existing content in each of these files as a reference when adding additional content for your lifecycle manager.

- Update `control_plane.yml` to add the lifecycle manager.
- Update `server_roles.yml` to add the lifecycle manager role.
- Update `net_interfaces.yml` to add the interface definition for the lifecycle manager.
- Create a `disks_lifecycle_manager.yml` file to define the disk layout for the lifecycle manager.

- Update `servers.yml` to add the dedicated lifecycle manager node.

`Control_plane.yml` The snippet below shows the addition of a single node cluster into the control plane to host the lifecycle manager service. Note that, in addition to adding the new cluster, you also have to remove the lifecycle manager component from the `cluster1` in the examples:

```
clusters:
  - name: cluster0
    cluster-prefix: c0
    server-role: LIFECYCLE-MANAGER-ROLE
    member-count: 1
    allocation-policy: strict
    service-components:
      - lifecycle-manager
      - ntp-client
  - name: cluster1
    cluster-prefix: c1
    server-role: CONTROLLER-ROLE
    member-count: 3
    allocation-policy: strict
    service-components:
      - ntp-server
```

This specifies a single node of role `LIFECYCLE-MANAGER-ROLE` hosting the lifecycle manager.

`Server_roles.yml` The snippet below shows the insertion of the new server roles definition:

```
server-roles:
  - name: LIFECYCLE-MANAGER-ROLE
    interface-model: LIFECYCLE-MANAGER-INTERFACES
    disk-model: LIFECYCLE-MANAGER-DISKS
  - name: CONTROLLER-ROLE
```

This defines a new server role which references a new interface-model and disk-model to be used when configuring the server.

`Net-interfaces.yml` The snippet below shows the insertion of the network-interface info:

```
- name: LIFECYCLE-MANAGER-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed3
      provider: linux
      devices:
        - name: hed3
        - name: hed4
  network-groups:
```

#### - MANAGEMENT

This assumes that the server uses the same physical networking layout as the other servers in the example. For details on how to modify this to match your configuration, see .

`Disks_lifecycle_manager.yml` In the examples, disk-models are provided as separate files (this is just a convention, not a limitation) so the following should be added as a new file named `disks_life-cycle_manager.yml`:

```
---
```

```
product:
    version: 2

disk-models:
- name: LIFECYCLE-MANAGER-DISKS
  # Disk model to be used for Lifecycle Managers nodes
  # /dev/sda_root is used as a volume group for /, /var/log and /var/crash
  # sda_root is a templated value to align with whatever partition is really used
  # This value is checked in os config and replaced by the partition actually used
  # on sda e.g. sdal or sda5

volume-groups:
- name: hlm-vg
  physical-volumes:
    - /dev/sda_root

logical-volumes:
# The policy is not to consume 100% of the space of each volume group.
# 5% should be left free for snapshots and to allow for some flexibility.
- name: root
  size: 80%
  fstype: ext4
  mount: /
- name: crash
  size: 15%
  mount: /var/crash
  fstype: ext4
  mkfs-opt: -O large_file
consumer:
    name: os
```

`Servers.yml` The snippet below shows the insertion of an additional server used for hosting the lifecycle manager. Provide the address information here for the server you are running on, i.e., the node where you have installed the HPE Helion OpenStack ISO.

```
servers:
# NOTE: Addresses of servers need to be changed to match your environment.
#
#       Add additional servers as required

#Lifecycle-manager
- id: lifecycle-manager
  ip-addr: <your IP address here>
  role: LIFECYCLE-MANAGER-ROLE
  server-group: RACK1
```

```
nic-mapping: HP-SL230-4PORT
mac-addr: 8c:dc:d4:b5:c9:e0
# ipmi information is not needed

# Controllers
- id: controller1
  ip-addr: 192.168.10.3
  role: CONTROLLER-ROLE
```

## Configuring HPE Helion OpenStack without DVR

By default in the KVM model, the Neutron service utilizes distributed routing (DVR). This is the recommended setup because it allows for high availability. However, if you would like to disable this feature, here are the steps to achieve this.

On your lifecycle manager, make the following changes:

1. In the `~/helion/my_cloud/config/neutron/neutron.conf.j2` file, change the line below from:

```
router_distributed = {{ router_distributed }}
```

to:

```
router_distributed = False
```

2. In the `~/helion/my_cloud/config/neutron/ml2_conf.ini.j2` file, change the line below from:

```
enable_distributed_routing = True
```

to:

```
enable_distributed_routing = False
```

3. In the `~/helion/my_cloud/config/neutron/l3_agent.ini.j2` file, change the line below from:

```
agent_mode = {{ neutron_l3_agent_mode }}
```

to:

```
agent_mode = legacy
```

4. In the `~/helion/my_cloud/definition/data/control_plane.yml` file, remove the following values from the Compute resource service-components list:

```
- neutron-l3-agent
- neutron-metadata-agent
```

5. Commit your changes to your local git repository:

```
cd ~/helion/hos/ansible
```

```
git add -A  
git commit -m "My config or other commit message"
```

6. Run the configuration processor:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

7. Run the ready deployment playbook:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. Continue installation. More information on cloud deployments are available in the Chapter 6, *Overview*

## Configuring HPE Helion OpenStack with Provider VLANs and Physical Routers Only

Another option for configuring Neutron is to use provider VLANs and physical routers only, here are the steps to achieve this.

On your lifecycle manager, make the following changes:

1. In the `~/helion/my_cloud/config/neutron/neutron.conf.j2` file, change the line below from:

```
router_distributed = {{ router_distributed }}
```

to:

```
router_distributed = False
```

2. In the `~/helion/my_cloud/config/neutron/ml2_conf.ini.j2` file, change the line below from:

```
enable_distributed_routing = True
```

to:

```
enable_distributed_routing = False
```

3. In the `~/helion/my_cloud/config/neutron/dhcp_agent.ini.j2` file, change the line below from:

```
enable_isolated_metadata = {{ neutron_enable_isolated_metadata }}
```

to:

```
enable_isolated_metadata = True
```

4. In the `~/helion/my_cloud/definition/data/control_plane.yml` file, remove the following values from the Compute resource service-components list:

```
- neutron-l3-agent  
- neutron-metadata-agent
```

# Considerations When Installing Two Systems on One Subnet

If you wish to install two separate HPE Helion OpenStack 5.0 systems using a single subnet, you will need to consider the following notes.

The `ip_cluster` service includes the `keepalived` daemon which maintains virtual IPs (VIPs) on cluster nodes. In order to maintain VIPs, it communicates between cluster nodes over the VRRP protocol.

A VRRP virtual routerid identifies a particular VRRP cluster and must be unique for a subnet. If you have two VRRP clusters with the same virtual routerid, causing a clash of VRRP traffic, the VIPs are unlikely to be up or pingable and you are likely to get the following signature in your `/etc/keepalived/keepalived.log`:

```
Dec 16 15:43:43 helion-cp1-c1-m1-mgmt Keepalived_vrrp[2218]: ip address associated
Dec 16 15:43:43 helion-cp1-c1-m1-mgmt Keepalived_vrrp[2218]: one or more VIP associa
Dec 16 15:43:43 helion-cp1-c1-m1-mgmt Keepalived_vrrp[2218]: bogus VRRP packet rec
Dec 16 15:43:43 helion-cp1-c1-m1-mgmt Keepalived_vrrp[2218]: VRRP_Instance(VI_2) i
```

To resolve this issue, our recommendation is to install your separate HPE Helion OpenStack 5.0 systems with VRRP traffic on different subnets.

If this is not possible, you may also assign a unique routerid to your separate HPE Helion OpenStack 5.0 system by changing the `keepalived_vrrp_offset` service configurable. The routerid is currently derived using the `keepalived_vrrp_index` which comes from a configuration processor variable and the `keepalived_vrrp_offset`.

For example,

1. Log in to your lifecycle manager.
2. Edit your `~/helion/my_cloud/config/keepalived/defaults.yml` file and change the value of the following line:

```
keepalived_vrrp_offset: 0
```

Change the off value to a number that uniquely identifies a separate vrrp cluster. For example:

```
keepalived_vrrp_offset: 0 for the 1st vrrp cluster on this subnet.
```

```
keepalived_vrrp_offset: 1 for the 2nd vrrp cluster on this subnet.
```

```
keepalived_vrrp_offset: 2 for the 3rd vrrp cluster on this subnet.
```

## Important

You should be aware that the files in the `~/helion/my_cloud/config/` directory are symlinks to the `~/helion/hos/ansible/` directory. For example, `~/helion/my_cloud/config/keepalived/defaults.yml` is a symbolic link to `~/helion/hos/ansible/roles/keepalived/defaults/main.yml`

```
ls -al ~/helion/my_cloud/config/keepalived/defaults.yml
lrwxrwxrwx 1 stack stack 55 May 24 20:38 /home/stack/helion/my_cloud/config/ke
```

If you are using a tool like `sed` to make edits to files in this directory, you might break the symbolic link and create a new copy of the file. To maintain the link, you will need to force `sed` to follow the link:

```
sed -i --follow-symlinks 's$keepalived_vrrp_offset: 0$keepalived_vrrp_offset:'
```

Alternatively, you could directly edit the target of the link `~/helion/hos/ansible/roles/keepalived/default/main.yml`.

3. Commit your configuration to the *FIXME: broken external xref*, as follows:

```
cd ~/helion/hos/ansible  
git add -A  
git commit -m "changing Admin password"
```

4. Run the configuration processor with this command:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Use the playbook below to create a deployment directory:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. If you are making this change after your initial install, run the following reconfigure playbook to make this change in your environment:

```
cd ~/scratch/ansible/next/hos/ansible/  
ansible-playbook -i hosts/verb_hosts FND-CLU-reconfigure.yml
```

# Installation Guide

---

## **Installation Guide**

---

# Table of Contents

Installation Overview .....	ccxxvii
For More Information .....	ccxxviii
I. Pre-Installation .....	1
1. Overview .....	3
2. Pre-Installation Checklist .....	4
BIOS and iLO Settings .....	4
Network Setup and Configuration .....	4
Lifecycle Manager .....	7
Information for the <code>nic_mappings.yml</code> Input File .....	7
Control Plane .....	8
Compute Hosts .....	9
VSA Hosts .....	9
Additional Comments .....	10
SLES Pre-Installation Checks .....	10
Introduction .....	10
Sample iptables rules must be removed on SLES .....	10
3. Using Git for Configuration Management .....	12
Initialization on a new deployment .....	12
Updating any configuration, including the default configuration .....	13
4. Boot from SAN and Multipath Configuration .....	14
Introduction .....	14
Install Phase Configuration .....	14
QLogic FCoE restrictions and additional configurations .....	15
Red Hat Compute Host for FCoE .....	17
Installing the HPE Helion OpenStack 5.0 iso for nodes that support Boot from SAN .....	19
5. Installing the L2 Gateway Agent for the Networking Service .....	25
Introduction .....	25
Sample network topology (for illustration purposes) .....	25
Networks .....	26
HPE 5930 switch configuration .....	27
Configuring the provider data path network .....	28
Enabling and configuring the L2 gateway agent .....	31
Routing between software and hardware .....	33
Connecting a baremetal server to the HPE 5930 switch .....	34
Configuration on a baremetal server .....	35
NIC bonding and IRF configuration .....	35
Scale numbers tested .....	35
L2 gateway commands .....	35
II. Cloud Installation .....	37
6. Overview .....	43
7. Installing via the GUI .....	45
Before you begin .....	46
Creating a CSV file for import .....	46
Run the GUI to install your cloud .....	47
Use the installer securely .....	47
Final Steps .....	48
8. DNS Service Installation Overview .....	49
Getting Started .....	49
Install the DNS Service with PowerDNS .....	49
Install DNS Service with PowerDNS .....	49

Configure the Backend .....	49
Install the DNS Service with BIND .....	51
Install DNS Service with BIND .....	51
Configure the Backend .....	51
Install the DNS Service with InfoBlox .....	51
Prerequisites .....	52
Configure the Backend .....	52
Configure DNS Domain and NS Records .....	55
9. Magnum Overview .....	57
Magnum Architecture .....	57
Install the Magnum Service .....	65
Installing Magnum as part of new HPE Helion OpenStack 5.0 environment .....	66
Adding Magnum to existing HPE Helion OpenStack 5.0 environment .....	66
Integrate Magnum with the DNS Service .....	67
10. Installing Mid-scale and Entry-scale KVM .....	70
Important Notes .....	70
Before You Start .....	71
Setting Up the Lifecycle Manager .....	71
Configuring Your Environment .....	73
Provisioning Your Baremetal Nodes .....	74
Running the Configuration Processor .....	76
Configuring TLS .....	77
Deploying the Cloud .....	77
Configuring a Block Storage Backend (Optional) .....	78
Post-Installation Verification and Administration .....	78
11. Installation for Helion Entry Scale ESX (with OVSVApp, KVM with VSA Model) .....	79
Important Notes .....	79
Before You Start .....	80
Prerequisites .....	80
Deploy ESX Cloud with OVSVApp .....	81
Procedure to Deploy ESX Cloud with OVSVApp .....	81
Setting Up the Lifecycle Manager .....	81
Prepare and Deploy Cloud Controllers .....	83
Provisioning Your Baremetal Nodes .....	84
Running the Configuration Processor .....	86
Deploying the Cloud .....	86
Prepare and Deploy ESX Computes and OVSVAPPs .....	87
Validate the block storage .....	92
Validate the compute .....	92
Validate the neutron .....	93
12. Sample activationtemplate.json File for ESX Compute .....	94
13. Installing Baremetal (Ironic) .....	107
Installation for HPE Helion Entry-scale Cloud with Ironic Flat Network .....	107
Before You Start .....	107
Setting Up the Lifecycle Manager .....	107
Configure Your Environment .....	109
Provisioning Your Baremetal Nodes .....	110
Running the Configuration Processor .....	112
Deploying the Cloud .....	112
Ironic configuration .....	113
Node Configuration .....	114
TLS Certificates with Ironic Python Agent (IPA) Images .....	114
Ironic in Multiple Control Plane .....	116
Networking for Baremetal in Multiple Control Plane .....	117

Handling Optional Swift Service .....	117
Instance Provisioning .....	117
Provisioning Baremetal Nodes with Flat Network Model .....	118
Supplied Images .....	119
Provisioning a node .....	119
Creating a node using <code>agent_ilo</code> .....	119
Creating a node using <code>agent_ipmi</code> .....	121
Create Flavor .....	122
Create network port .....	123
Create Glance Image .....	123
Generate Key Pair .....	124
Determine the neutron network ID .....	124
Boot the node .....	124
Create Glance Images using <code>diskimage-builder</code> (DIB) .....	125
Creating BIOS images for RHEL .....	126
Creating BIOS images for Ubuntu .....	126
Creating UEFI images .....	126
Provisioning Baremetal Nodes with Multi-Tenancy .....	127
View Ironic System Details .....	132
View details about the server using <code>nova show &lt;nova-node-id&gt;</code> .....	132
View detailed information about a node using <code>ironic node-show &lt;ironic-node-id&gt;</code> .....	133
View detailed information about a port using <code>ironic port-show &lt;ironic-port-id&gt;</code> .....	134
View detailed information about a hypervisor using <code>nova hypervisor-list</code> and <code>nova hypervisor-show</code> .....	134
View a list of all running services using <code>nova service-list</code> .....	135
Troubleshooting Ironic Installation .....	135
No valid host was found. There are not enough hosts available. ....	136
Deployment to a node fails and in "ironic node-list" command, the <code>power_state</code> column for the node is shown as "None" .....	138
Error Downloading Image .....	139
Using <code>node-inspection</code> can cause temporary claim of IP addresses .....	139
Node permanently stuck in deploying state .....	139
The NICs in the baremetal node should come first in boot order .....	139
Increase in the number of nodes can cause power commands to fail .....	140
DHCP succeeds with PXE but times out with iPXE .....	140
Node Cleaning .....	141
Setup .....	142
In use .....	142
Troubleshooting .....	143
Disabling Node Cleaning .....	143
Ironic and OneView .....	143
Enabling Ironic OneView driver in HPE Helion OpenStack .....	143
Adding OneView Appliance Credentials .....	144
Encrypting the OneView Password .....	144
Decrypting the OneView Password .....	144
Registering Baremetal Node for OneView Driver .....	144
Updating Node Properties .....	145
Creating Port for Driver .....	145
Creating a Node .....	145
Getting Data using REST API .....	145
Ironic OneView CLI .....	145
RAID Configuration for Ironic .....	146

Audit Support for Ironic .....	150
API Audit Logging .....	150
Enabling API Audit Logging .....	151
Sample Audit Event .....	151
14. Installation for HPE Helion Entry-scale Cloud with Swift Only .....	153
Important Notes .....	153
Before You Start .....	153
Setting Up the Lifecycle Manager .....	153
Configure Your Environment .....	156
Running the Configuration Processor .....	157
Provisioning Your Baremetal Nodes .....	158
Deploying the Cloud .....	159
Post-Installation Verification and Administration .....	160
15. Installing SLES Compute .....	161
SLES Compute Node Installation Overview .....	161
SLES Support .....	161
Using the Lifecycle Manager to Deploy SLES Compute Nodes .....	161
Deploying legacy BIOS SLES compute nodes .....	162
Provisioning SLES Yourself .....	162
Introduction .....	162
Configure Lifecycle Manager to Enable SLES .....	162
Install SLES 12 SP2 .....	163
Assign a static IP .....	163
Add stack user and home directory .....	164
Allow user stack to sudo without password .....	164
Add zypper repository .....	164
Add Required Packages .....	164
Set up passwordless SSH access .....	164
Using SLES as a Ceph Client .....	165
16. HLM-Hypervisor instructions .....	166
Introduction .....	166
Model changes .....	166
passthrough-network-groups .....	166
hlm-hypervisor .....	167
Bootstrap Instructions .....	173
Customizing for VCP .....	173
Installing a fully input model managed Virtual Control Plane based HPE Helion .....	173
OpenStack Cloud .....	173
Provisioning just the Virtual Control Plane VMs .....	174
Manually running each phase of the Virtual Control Plane provisioning .....	174
NIC Mapping for HLM Virtual-Controllers .....	175
Recommended Mappings .....	176
Notification of actions to be taken post upgrade .....	176
Reboot of a HLM-Hypervisor node .....	176
Staged install on HLM-Hypervisor individual play books. ....	177
Virtual Controller Replacement .....	177
Ansible host-specific variables for network-group access .....	178
Example 3rd-Party Ansible for Network Configuration Access .....	181
Ansible host-specific variables for gluster .....	182
"Empty" HLM Hypervisor Node .....	182
Monasca Monitoring of HLM Hypervisors .....	182
Monitored Metrics .....	182
Configured Alarms .....	183
HLM Hypervisor Monitoring Configuration .....	183

17. Integrations .....	185
Ceph Overview .....	185
Overview .....	185
Deployment Architecture .....	186
Ceph Networking .....	186
Placement of service component .....	187
Ceph Deployment Architecture .....	188
Alternative supported architecture .....	189
Core networking .....	189
Placement of service components .....	190
Hardware recommendations .....	190
Ceph Deployment and Configurations .....	190
Alternative Supported Choices .....	201
Usage of Ceph Storage .....	214
Managing Ceph Clusters After Deployment .....	227
Configuring for VSA Block Storage Backend .....	227
Prerequisites .....	228
Notes .....	228
Configure HPE StoreVirtual VSA .....	228
Using Ansible .....	229
Using the CMC Utility .....	230
Configure VSA as the Backend .....	241
Post-Installation Tasks .....	244
Configuring for 3PAR Block Storage Backend .....	244
Prerequisites .....	244
Notes .....	245
Multipath Support .....	245
Configure 3PAR FC as a Cinder Backend .....	246
Configure 3PAR iSCSI as Cinder backend .....	247
Post-Installation Tasks .....	249
Ironic OneView Integration .....	249
Prerequisites .....	249
Integrating with OneView .....	249
Registering Node in OneView .....	250
Provisioning Ironic Node .....	251
18. Troubleshooting the Installation .....	257
Issues during Lifecycle-manager Setup .....	257
Issues while Provisioning your Baremetal Nodes .....	257
Issues while Updating Configuration Files .....	262
Issues while Deploying the Cloud .....	263
19. Troubleshooting the Block Storage Backend Configuration .....	268
20. Troubleshooting the ESX .....	270
Issue: If hlm-ux-services.service is not running, the EON resource-activate and re-source-deactivate commands fails .....	270
Issue: ESX onboard Compute .....	270
Issue: Migrate eon-conductor .....	270
Issue: ESX Cluster shows UNKNOWN in OpsConsole .....	273
Issue: Unable to view the VM console in Horizon UI .....	273
III. Post-Installation .....	275
21. Overview .....	278
22. Cloud Verification .....	279
API Verification .....	279
Prerequisites .....	279
Tempest Integration Tests .....	279

Running the Tests .....	280
Viewing Test Results .....	280
Customizing the Test Run .....	281
Run Tests for Specific Services and Exclude Specific Features .....	281
Run Tests Matching a Series of White and Blacklists .....	281
23. UI Verification .....	283
Verifying Your Block Storage Backend .....	283
Create a Volume .....	283
Attach Volume to an Instance .....	284
Detach Volume from Instance .....	284
Delete Volume .....	285
Verifying Your Object Storage (Swift) .....	285
Verify the Object Storage (Swift) Operations .....	286
Uploading an Image for Use .....	287
Running the Playbook .....	287
How to Curate Your Own Images .....	287
Using the GlanceClient CLI to Create Images .....	287
Creating an External Network .....	287
Notes .....	288
Using the Ansible Playbook .....	288
Using the NeutronClient CLI .....	288
Next Steps .....	289
24. Installing OpenStack Clients .....	290
25. Configuring Transport Layer Security (TLS) .....	293
Configuring TLS in the input model .....	294
User-provided certificates and trust chains .....	295
Edit the input model to include your certificate files .....	296
Generate a self-signed CA .....	297
Generate a certificate signing request .....	299
Generate a server certificate .....	299
Upload to the lifecycle manager .....	302
Configuring the cipher suite .....	302
Testing .....	303
Verifying that the trust chain is correctly deployed .....	303
Turning TLS on or off .....	303
26. Configuring Availability Zones .....	305
27. Configuring Load Balancer as a Service .....	306
Prerequisites .....	307
Octavia Load Balancing Provider .....	307
Setup of prerequisites .....	307
Create Load Balancers .....	317
Create Floating IPs for Load Balancer .....	320
Testing the Octavia Load Balancer .....	321
28. Other Common Post-Installation Tasks .....	323
Determining Your User Credentials .....	323
Configure your lifecycle manager to use the command-line tools .....	323
Protect home directory .....	323
Back up Your SSH Keys .....	324
Retrieving Service Endpoints .....	324
Other Common Post-Installation Tasks .....	324

---

## **List of Figures**

5.1. Figure 1 .....	26
5.2. Figure 2 .....	26
5.3. Figure 3 .....	27
13.1. Architecture of Multiple Control Plane with Ironic .....	117

---

## **List of Tables**

8.1. ....	49
9.1. ....	58

---

## **List of Examples**

25.1. Certificate request file .....	297
--------------------------------------	-----

---

# Installation Overview

Before jumping into your installation, we recommend taking the time to read through our documentation to get an overview of the sample configurations HPE Helion OpenStack 5.0 offers. We have highly tuned example configurations for each of these Cloud models:

Name	Location
the section called “Entry-scale KVM with VSA Model”	<code>~/helion/examples/entry-scale-kvm-vsa</code>
the section called “Entry-scale KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging”	<code>~/helion/examples/entry-scale-kvm-vsa-mml</code>
the section called “Entry-scale KVM with Ceph Model”	<code>~/helion/examples/entry-scale-kvm-ceph</code>
the section called “Mid-scale KVM with VSA Model”	<code>~/helion/examples/mid-scale-kvm-vsa</code>
the section called “Entry-scale ESX, KVM with VSA Model”	<code>~/helion/examples/entry-scale-esx-kvm-vsa</code>
the section called “Entry-scale ESX, KVM with VSA Model with Dedicated Cluster for Metering, Monitoring, and Logging”	<code>~/helion/examples/entry-scale-esx-kvm-vsa-mml</code>
the section called “Entry-scale Swift Model”	<code>~/helion/examples/entry-scale-swift</code>
the section called “Entry-scale Cloud with Ironic Flat Network”	<code>~/helion/examples/entry-scale-ironic-flat-network</code>
the section called “Entry-scale Cloud with Ironic Multi-Tenancy”	<code>~/helion/examples/entry-scale-ironic-multi-tenancy</code>

- the section called “Entry-scale KVM with VSA Model”
- 
- the section called “Entry-scale Swift Model”
- 
- 

## Using the Command-line

You should use the command-line if:

- You are installing a more complex or large-scale cloud.
- You have more than 15 nodes.
- You need to use availability zones or the server groups functionality of the cloud model. See the for more information.
- You want to customize the cloud configuration beyond the tuned defaults that HPE provides out of the box.

- You need deeper customizations than are possible to express using the GUI.

Instructions for installing via the command-line are here:

- Chapter 10, *Installing Mid-scale and Entry-scale KVM*
- Chapter 11, *Installation for Helion Entry Scale ESX (with OVSvApp, KVM with VSA Model)*

## For More Information

Other useful documents that will help you understand, plan, configure, and install your cloud are listed below.

- Chapter 3, *Using Git for Configuration Management*
- Chapter 18, *Troubleshooting the Installation*
- Chapter 22, *Cloud Verification*
- Chapter 28, *Other Common Post-Installation Tasks*

---

## **Part I. Pre-Installation**

---

# Table of Contents

1. Overview .....	3
2. Pre-Installation Checklist .....	4
BIOS and iLO Settings .....	4
Network Setup and Configuration .....	4
Lifecycle Manager .....	7
Information for the <code>nic_mappings.yml</code> Input File .....	7
Control Plane .....	8
Compute Hosts .....	9
VSA Hosts .....	9
Additional Comments .....	10
SLES Pre-Installation Checks .....	10
Introduction .....	10
Sample iptables rules must be removed on SLES .....	10
3. Using Git for Configuration Management .....	12
Initialization on a new deployment .....	12
Updating any configuration, including the default configuration .....	13
4. Boot from SAN and Multipath Configuration .....	14
Introduction .....	14
Install Phase Configuration .....	14
QLogic FCoE restrictions and additional configurations .....	15
Red Hat Compute Host for FCoE .....	17
Installing the HPE Helion OpenStack 5.0 iso for nodes that support Boot from SAN .....	19
5. Installing the L2 Gateway Agent for the Networking Service .....	25
Introduction .....	25
Sample network topology (for illustration purposes) .....	25
Networks .....	26
HPE 5930 switch configuration .....	27
Configuring the provider data path network .....	28
Enabling and configuring the L2 gateway agent .....	31
Routing between software and hardware .....	33
Connecting a baremetal server to the HPE 5930 switch .....	34
Configuration on a baremetal server .....	35
NIC bonding and IRF configuration .....	35
Scale numbers tested .....	35
L2 gateway commands .....	35

---

# Chapter 1. Overview

We have a Chapter 2, *Pre-Installation Checklist* we recommend going through to ensure your environment meets the requirements of the Cloud model you choose.

Once you have an idea of the configuration you want to choose for your cloud and you have gone through the pre-installation steps, you have two options for installation. You can use a GUI that runs in your web browser or you can install via the command-line that exposes the full power and flexibility of HPE Helion OpenStack 5.0.

## Using the GUI

You should use the GUI if:

- You are not planning to deploy availability zones or use L3 segmentation in your initial deployment.
- You are happy with the tuned HPE default OpenStack configurations.

Instructions for installing via the GUI are here.

- Chapter 7, *Installing via the GUI*

Note that reconfiguration of your cloud, at the moment, can only be done via the command-line. The GUI installer is for initial installation only.

---

# Chapter 2. Pre-Installation Checklist

## Important

The formatting of this page facilitates printing it out and using it to record details of your setup.

This checklist is focused on the Entry-scale KVM with VSA model but you can alter it to fit the example configuration you chose for your cloud.

## BIOS and iLO Settings

Ensure that the following BIOS and iLO settings are applied to each baremetal server:

#	Item
	Setup the iLO Advanced license in the iLO configuration
	Choose either UEFI or Legacy BIOS in the BIOS settings
	Verify the Date and Time settings in the BIOS.  <b>Verify Time and Date</b>  It is important that all of your systems have the correct date/time because the HPE Store-Virtual VSA license has a start date. If the start date is later than the system time then the installation will fail. Note also that Helion installs and runs with UTC, not local time.
	Ensure that Wake-on-LAN is disabled in the BIOS
	Ensure that the NIC port to be used for PXE installation has PXE enabled in the BIOS
	Ensure that all other NIC ports have PXE disabled in the BIOS
	Ensure all hardware in the server not directly used by Helion is disabled

## Network Setup and Configuration

Before installing HPE Helion OpenStack, the following networks must be provisioned and tested. The networks are not installed or managed by the Cloud. You must install and manage the networks as documented in the .

Note that if you want a pluggable IPAM driver, it must be specified at install time. Only with a clean install of HPE Helion OpenStack 5.0 can you specify a different IPAM driver. If upgrading, you must use the default driver. More information can be found in .

Use these checklists to confirm and record your network configuration information.

### Router

The IP router used with HPE Helion OpenStack must support the updated of its ARP table through gratuitous ARP packets.

### PXE Installation Network

When provisioning the IP range, allocate sufficient IP addresses to cover both the current number of servers and any planned expansion. Use the following table to help calculate the requirements:

Instance	Description	IPs
Deployer O/S		1
Controller server O/S (x3)		3
Compute servers (2nd thru 100th)	single IP per server	
VSA host servers	single IP per server	

#	Item	Value
	Network is untagged	
	No DHCP servers other than Helion are on the network	
	Switch PVID used to map any "internal" VLANs to untagged	
	Routable to the IPMI network	
	IP CIDR	
	IP Range (Usable IPs)	begin: end:
	Default IP Gateway	

### Management Network

The management network is the backbone used for the majority of HPE Helion OpenStack management communications. Control messages are exchanged between the Controllers, Compute hosts, and Cinder backends through this network. In addition to the control flows, the management network is also used to transport Swift and iSCSI based Cinder block storage traffic between servers.

When provisioning the IP Range, allocate sufficient IP addresses to cover both the current number of servers and any planned expansion. Use the following table to help calculate the requirements:

Instance	Description	IPs
Controller server O/S (x3)		3
Controller VIP		1
Compute servers (2nd thru 100th)	single IP per server	
VSA VM servers	single IP per server	
VSA VIP per cluster		

#	Item	Value
	Network is untagged	
	No DHCP servers other than Helion are on the network	
	Switch PVID used to map any "internal" VLANs to untagged	
	IP CIDR	
	IP Range (Usable IPs)	begin: end:
	Default IP Gateway	
	VLAN ID	

### IPMI Network

The IPMI network is used to connect the IPMI interfaces on the servers that are assigned for use with implementing the cloud. For HPE ProLiant servers, the IPMI network connects to the HPE iLO management device port for the server. This network is used by Cobbler to control the state of the servers during baremetal deployments.

#	Item	Value
	Network is untagged	
	Routable to the Management Network	
	IP Subnet	
	Default IP Gateway	

### External API Network

The External network is used to connect OpenStack endpoints to an external public network such as a company's intranet or the public internet in the case of a public cloud provider.

When provisioning the IP Range, allocate sufficient IP addresses to cover both the current number of servers and any planned expansion. Use the following table to help calculate the requirements.

Instance	Description	IPs
Controller server O/S (x3)		3
Controller VIP		1

#	Item	Value
	VLAN Tag assigned:	
	IP CIDR	
	IP Range (Usable IPs)	begin: end:
	Default IP Gateway	
	VLAN ID	

### External VM Network

The External VM network is used to connect cloud instances to an external public network such as a company's intranet or the public internet in the case of a public cloud provider. The external network has a predefined range of Floating IPs which are assigned to individual instances to enable communications to and from the instance to the assigned corporate intranet/internet. There should be a route between the External VM and External API networks so that instances provisioned in the cloud, may access the Cloud API endpoints, using the instance floating IPs.

#	Item	Value
	VLAN Tag assigned:	
	IP CIDR	
	IP Range (Usable IPs)	begin: end:
	Default IP Gateway	
	VLAN ID	

# Lifecycle Manager

This server contains the HPE Helion OpenStack installer, which is based on Git, Ansible, and Cobbler.

#	Item	Value
	Disk Requirement: Single 8GB disk needed per the Chapter 2, <i>Hardware and Software Support Matrix</i>	
	Chapter 10, <i>Installing Mid-scale and Entry-scale KVM</i>	
	Ensure your local DNS nameserver is placed into your /etc/resolv.conf file	
	Install and configure NTP for your environment	
	Ensure your NTP server(s) is placed into your /etc/ntp.conf file	
	NTP time source:	

## Information for the nic\_mappings.yml Input File

Log onto each type of physical server you have and issue platform-appropriate commands to identify the bus-address and port-num values that may be required. For example, run the following command:

```
sudo lspci -D | grep -i net
```

and use this information for the bus-address value in your nic\_mappings.yml file.

### NIC Adapter PCI Bus Address Output

To find the port-num use:

```
cat /sys/class/net/<device name>/dev_port
```

where the 'device-name' is the name of the device **currently mapped** to this address, not necessarily the name of the device **to be mapped**.

### Network Device Port Number Output

Network Device Port Number Output

## Control Plane

The Control Plane consists of three servers, in a highly available cluster, that host the core HPE Helion OpenStack services including Nova, Keystone, Glance, Cinder, Heat, Neutron, Swift, Ceilometer, and Horizon. Additional services include mysql, ip-cluster, apache2, rabbitmq, memcached, zookeeper, kafka, influxDB, storm, monasca, logging, and cmc.

### Note

To mitigate the "split-brain" situation described in it is recommended that you have HA network configuration with Multi-Chassis Link Aggregation (MLAG) and NIC bonding configured for all the controllers to deliver system-level redundancy as well network-level resiliency. Also reducing the ARP timeout on the TOR switches will help.

#### Control Plane 1

#	Item	Value
	Disk Requirement: 3x 512 GB disks (or enough space to create three logical drives with that amount of space)	
	Ensure the disks are wiped	
	MAC address of first NIC	
	A second NIC, or a set of bonded NICs are required	
	iLO IP address	
	iLO Username/Password	

#### Control Plane 2

#	Item	Value
	Disk Requirement: 3x 512 GB disks (or enough space to create three logical drives with that amount of space)	
	Ensure the disks are wiped	
	MAC address of first NIC	
	A second NIC, or a set of bonded NICs are required	
	iLO IP address	
	iLO Username/Password	

#### Control Plane 3

#	Item	Value
	Disk Requirement: 3x 512 GB disks (or enough space to create three logical drives with that amount of space)	
	Ensure the disks are wiped	
	MAC address of first NIC	
	A second NIC, or a set of bonded NICs are required	
	iLO IP address	
	iLO Username/Password	

## Compute Hosts

One or more KVM Compute servers will be used as the compute host targets for instances.

#	Item
	Disk Requirement: 2x 512 GB disks (or enough space to create three logical drives with that amount of space)
	A NIC for PXE boot and a second NIC, or a NIC for PXE and a set of bonded NICs are required
	Ensure the disks are wiped

Table to record your Compute host details:

ID	NIC	MAC	Ad- dress	iLO	User- name/Password	iLO IP Address	CPU/Mem/Disk

## VSA Hosts

Three or more servers with local disk volumes running HPE StoreVirtual VSA to provide Cinder block storage resources.

### Note

The cluster created from VSA nodes should be quorum. In other words, the cluster must be formed with either 3,5,7 and so on.

#	Item
	Disk Requirement: 3x 512 GB disks (or enough space to create three logical drives with that amount of space) <ul style="list-style-type: none"> <li>- The VSA appliance deployed on a host is expected to consume ~40 GB of disk space from the host root disk for ephemeral storage to run the VSA virtual machine.</li> </ul>
	A NIC for PXE boot and a second NIC, or a NIC for PXE and a set of bonded NICs are required
	Ensure the disks are wiped

Table to record your VSA host details:

ID	NIC MAC Address	iLO User- name/Pass- word	iLO IP Ad- dress	CPU/Mem/ Disk	VSA Data Vol- ume

## Additional Comments

This section is so you can add any additional information that you deem necessary.

## SLES Pre-Installation Checks

### Introduction

**This document needs review for SLES content.**

As part of the preparation to provision a SLES compute node you need to be aware of the issues raised in this article and take any remedial action required.

### Sample iptables rules must be removed on SLES

HPE Helion OpenStack 5.0 uses iptables to secure access to lifecycle manager network interfaces and on Red Hat this requires the package `iptables-services` to be installed. The package will provide sample iptables configurations for IPv4 and IPv6 if none existed before. This sample configuration is inappropriate for HPE Helion OpenStack operation and the node will not be able to run HOS with these rules installed.

The files installed are:

- `/etc/sysconfig/iptables`
- `/etc/sysconfig/ip6tables`

If these files **do not exist** on the candidate Red Hat systems the rest of this note may be skipped as the HPE Helion OpenStack 5.0 install will prevent the installation of the sample files. However, if these files do exist, there are a number of steps that you must follow before you install HPE Helion OpenStack 5.0.

The contents of these two files are displayed here for reference:

```
/etc/sysconfig/iptables
```

```
# sample configuration for iptables service
# you can edit this manually or use system-config-firewall
# please do not ask us to add additional ports/services to this default configuration
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

```
/etc/sysconfig/ip6tables
```

```
# sample configuration for ip6tables service
# you can edit this manually or use system-config-firewall
# please do not ask us to add additional ports/services to this default configuration
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p ipv6-icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -p tcp -m state --state NEW -m tcp --dport 22 -j ACCEPT
-A INPUT -d fe80::/64 -p udp -m udp --dport 546 -m state --state NEW -j ACCEPT
-A INPUT -j REJECT --reject-with icmp6-adm-prohibited
-A FORWARD -j REJECT --reject-with icmp6-adm-prohibited
COMMIT
```

These rules are applied to network traffic on *any interface* on the system whereas HPE Helion OpenStack components and OpenStack components manage *interface specific* rules. With reference to the security policies that you may require for your environment, it is essential that you either:

- Delete the files /etc/sysconfig/iptables and /etc/sysconfig/ip6tables as none of the contained rules are required.
- Ensure that any remaining rules are limited to interfaces not used by HPE Helion OpenStack.

## Important

If these files existed on your system, and did contain content, the corresponding rules are currently installed and active on the system. Once you delete these two files (or edit them to limit the rules to interfaces not used by HPE Helion OpenStack), you will need to reboot the system to activate the new settings.

---

# Chapter 3. Using Git for Configuration Management

In HPE Helion OpenStack 5.0, a local git repository is used to track configuration changes and the configuration processor (CP) uses this repository. The introduction of a git workflow also means that your configuration history is maintained, making rollbacks easier as well as keeping a record of previous configuration settings.

The git repository is installed by the lifecycle manager on the lifecycle manager node.

The git repository provides a way for you to merge changes that you pull down as “upstream” (that is, from HP) updates, and allows you to manage your own configuration changes.

## Initialization on a new deployment

On a system new to HPE Helion OpenStack 5.0, the lifecycle manager will prepare a git repository under `~/helion`. The lifecycle manager provisioning runs the `hlm-init-deployer` script automatically - this calls `ansible-playbook -i hosts/localhost git-00-initialise.yml`.

As a result, the `~/helion` directory is initialized as a git repo (if it is empty). It is initialized with four empty branches:

hos	This holds the upstream source code corresponding to the contents of the <code>~/helion</code> directory on a pristine fresh installation. Every source code release that is downloaded from HPE is applied as a fresh commit to this branch. This branch contains no customization by the end user.
site	This branch begins life as a copy of the first 'hos' drop. It is onto this branch that you commit your configuration changes. It's the branch most visible to the end user.
ansible	This branch contains the variable definitions generated by the CP that our main ansible playbooks need. This includes the <code>verb_hosts</code> file that describes to ansible what servers are playing what roles. The <code>ready-deployment</code> playbook takes this output and assembles a <code>~/scratch</code> directory containing the ansible playbooks together with the variable definitions in this branch. The result is a working ansible directory <code>~/scratch/ansible/next/hos/ansible</code> from which the main deployment playbooks may be successfully run.
cp-persistent	This branch contains the persistent state that the CP needs to maintain. That state is mostly the assignment of IP addresses and roles to particular servers. Some operational procedures may involve editing the contents of this branch: for example, retiring a machine from service or repurposing it.

Two temporary branches are created and populated at run time:

staging-ansible	This branch hosts the most recent commit that will be appended to the ansible branch.
staging-cp-persistent	This branch hosts the most recent commit that will be appended to the cp-persistent branch.

## Note

The information above provides insight into the workings of the configuration processor and the git repository. However, in practice you can simply follow the steps below to make configuration changes.

# Updating any configuration, including the default configuration

When you need to make updates to a configuration you must:

1. Check out the `site` branch. You may already be on that branch. If so, git will tell you that and the command will leave you there.

```
git checkout site
```

2. Edit the YAML file or files that contain the configuration you want to change.
3. Commit the changes to the `site` branch.

```
git add -A  
git commit -m "your commit message goes here in quotes"
```

If you want to add a single file to your git repository, you can use the command below, as opposed to using `git add -A`:

```
git add PATH_TO_FILE
```

For example, if you made a change to your `servers.yml` file and wanted to only commit that change, you would use this command:

```
git add ~/hestepoN/my_cloud/definition/data/servers.yml
```

4. To produce the required configuration processor output from those changes. Review the output files manually if required, run the configuration processor:

```
cd ~/hestepoN/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Ready the deployment area

```
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the deployment playbooks from the resulting scratch directory.

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts site.yml
```

---

# Chapter 4. Boot from SAN and Multipath Configuration

## Introduction

For information about supported hardware for multipathing, see the section called “Supported Hardware Configurations”.

### Important

When exporting a LUN to a node for boot from san, you should ensure that "LUN 0" is assigned to the LUN and configure any setup dialog that is necessary in the firmware to consume this LUN0 for OS boot.

### Important

Any hosts that are connected to 3PAR storage must have a host\_persona of 2-generationalua set on the 3PAR. Refer to the 3PAR documentation for the steps necessary to check this and change if necessary.

iSCSI boot from SAN is not supported. For more information on the use of Cinder with multipath, see the section called “Multipath Support”.

In order to allow HPE Helion OpenStack 5.0 use volumes from a SAN, a number of configuration options need to be specified for both the install phases and for the osconfig phases. In all cases the devices that are utilized are devices for which multipath is configured.

## Install Phase Configuration

For FC connected nodes and for FCoE nodes where the network processor used is from the Emulex family such as for the 650FLB, the following changes need to be made.

1. In each stanza of the servers.yaml insert a line stating `boot-from-san: true`

```
- id: controller2
  ip-addr: 192.168.10.4
  role: CONTROLLER-ROLE
  server-group: RACK2
  nic-mapping: HP-DL360-4PORT
  mac-addr: "8a:8e:64:55:43:76"
  ilo-ip: 192.168.9.4
  ilo-password: password
  ilo-user: admin
  boot-from-san: true
```

This uses the disk `/dev/mapper/mpatha` as the default device on which to install the OS.

2. In the disk input models you also need to specify the devices that will be used via their multipath names (which will be of the form `/dev/mapper/mpatha`, `/dev/mapper/mpathb` etc).

```
volume-groups:
  - name: hlm-vg
    physical-volumes:

      # NOTE: 'sda_root' is a templated value. This value is checked in
      # os-config and replaced by the partition actually used on sda
      #e.g. sdal or sda5
      - /dev/mapper/mpatha_root

      ...
      - name: vg-comp
        physical-volumes:
          - /dev/mapper/mpathb
```

## QLogic FCoE restrictions and additional configurations

If you are using network cards such as Qlogic Flex Fabric 536 and 630 series then there are additional OS configuration steps to support the importation of LUNs as well as some restrictions on supported configurations.

The restrictions are:

1. Only one network card can be enabled in the system.
2. The FCoE interfaces on this card are dedicated to FCoE traffic - these cannot have ip addresses associated with them.
3. Nic mapping cannot be used.

In addition to the configuration options above, you also need to specify the FCoE interfaces for install and for os configuration. There are 3 places where you need to add additional configuration options for fcoe-support:

- In `servers.yml`, which is used for configuration of the system during OS install, FCoE interfaces need to be specified for each server. In particular, the mac addresses of the FCoE interfaces need to be given, **not** the symbolic name (e.g. eth2).

```
- id: compute1
  ip-addr: 10.245.224.201
  role: COMPUTE-ROLE
  server-group: RACK2
  mac-addr: 6c:c2:17:33:4c:a0
  ilo-ip: 10.1.66.26
  ilo-user: hlinux
  ilo-password: hlinux123
  boot-from-san: True
  fcoe-interfaces:
    - 6c:c2:17:33:4c:a1
    - 6c:c2:17:33:4c:a9
```

## Important

Nic mapping can not be used.

- For the osconfig phase, you will need to specify the `fcoe-interfaces` as a peer of `network-interfaces` in the `net_interfaces.yml` file:

```
- name: CONTROLLER-INTERFACES
  fcoe-interfaces:
    - name: fcoe
      devices:
        - eth2
        - eth3
  network-interfaces:
    - name: eth0
      device:
        name: eth0
  network-groups:
    - EXTERNAL-API
    - EXTERNAL-VM
    - GUEST
    - MANAGEMENT
```

## Important

The mac addresses specified in the `fcoe-interfaces` stanza in `servers.yml` must correspond to the symbolic names used in the `fcoe-interfaces` stanza in `net_interfaces.yml`.

Also, to satisfy the restriction outlined in `#multipath_boot_from_san/restriction2` above, there can be no overlap between the devices in `fcoe-interfaces` and those in `network-interfaces` in the `net_interfaces.yml` file. In the example, `eth2` and `eth3` are `fcoe-interfaces` while `eth0` is in `network-interfaces`.

- As part of the initial install from an iso, additional parameters need to be supplied on the kernel command line:

```
multipath=true partman-fcoe/interfaces=<mac address1>,<mac address2> disk-detect
```

See the sections of installing from an ISO using UEFI (the section called “UEFI Boot Mode with QLogic FCoE”) and BIOS (the section called “Legacy BIOS Boot Mode with QLogic FCoE”).

Since nic mapping is not used to guarantee order of the networks across the system the installer will remap the network interfaces in a deterministic fashion as part of the install. As part of the installer dialogue, if DHCP is not configured for the interface, it is necessary to confirm that the appropriate interface is assigned the ip address. The network interfaces may not be at the names expected when installing via an iso. When you are asked to apply an ip address to an interface, enter Alt-F2 and in the console window issue an `ip` a command to examine the interfaces and their associated mac addresses. Take a note of the interface name with the expected mac address and use this in the subsequent dialogue. Enter Alt-F1 to return to the installation screen. You should note that the names of the interfaces may have changed after the installation completes. These names are used consistently in any subsequent operations.

Therefore, even if FCoE is not used for boot from SAN (e.g. for cinder), then it is recommended that `fcoe-interfaces` be specified as part of install (without the multipath or disk detect options). Alternatively, you need to run `osconfig-fcoe-reorder.yml` before `site.yml` or `osconfig-run.yml` is invoked to reorder the networks in a similar manner to the installer. In this case, the nodes will need to be manually rebooted for the network reorder to take effect. You will run `osconfig-fcoe-reorder.yml` in the following scenarios:

- If you have used a third-party installer to provision your baremetal nodes
- If you are booting from a local disk (i.e. one that is not presented from the SAN) but you want to use FCoE later, for example, for cinder.

To run the command:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts osconfig-fcoe-reorder.yml
```

If you do not run `osconfig-fcoe-reorder.yml`, you will encounter a failure in `osconfig-run.yml`.

If you are booting from a local disk, the LUNs that will be imported over FCoE will not be visible before `site.yml` or `osconfig-run.yml` has been run. However, if you need to import the LUNs before this, for instance, in scenarios where you need to run `wipe_disks.yml`, then you can run the `fcoe-enable` playbook across the nodes in question. This will configure FCoE and import the LUNs presented to the nodes.

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/verb_hosts fcoe-enable.yml
```

## Red Hat Compute Host for FCoE

When installing a Red Hat compute host, the names of the network interfaces will have names like `ens1f2` rather than `eth2` therefore a separate role and associated `network-interfaces` and `fcoe-interfaces` descriptions will have to be provided in the input model for the Red Hat compute hosts. Here are some excerpts highlighting the changes required:

### `net_interfaces.yml`

```
- name: RHEL-COMPUTE-INTERFACES  
  fcoe-interfaces:  
    - name: fcoe  
      devices:  
        - ens1f2  
        - ens1f3  
    network-interfaces:  
      - name: ens1f0  
        device:  
          name: ens1f0  
    network-groups:  
      - EXTERNAL-VM  
      - GUEST
```

- MANAGEMENT

**control\_plane.yml**

```
- name: rhel-compute
  resource-prefix: rhcomp
  server-role: RHEL-COMPUTE-ROLE
  allocation-policy: any
  min-count: 0
  service-components:
    - ntp-client
    - nova-compute
    - nova-compute-kvm
    - neutron-l3-agent
    - neutron-metadata-agent
    - neutron-openvswitch-agent
    - neutron-lbaasv2-agent
```

**server\_roles.yml**

```
- name: RHEL-COMPUTE-ROLE
  interface-model: RHEL-COMPUTE-INTERFACES
  disk-model: COMPUTE-DISKS
```

**servers.yml**

```
- id: QlogicFCoE-Cmp2
  ip-addr: 10.245.224.204
  role: RHEL-COMPUTE-ROLE
  server-group: RACK2
  mac-addr: "6c:c2:17:33:4c:a0"
  ilo-ip: 10.1.66.26
  ilo-password: hlinux123
  ilo-user: hlinux
  boot-from-san: True
  distro-id: rhel72-x86_64-multipath
  fcoe-interfaces:
    - 6c:c2:17:33:4c:a1
    - 6c:c2:17:33:4c:a9
```

## Important

Note that you need to add a `-multipath` suffix to the `distro-id` value when using multipath with RHEL.

If you are installing a Red Hat compute node with QLogic FCoE boot from SAN, you will need to edit the `/var/lib/cobbler/kickstarts/rhel72-anaconda-ks-multipath.cfg` file and uncomment the **two** sections that are prefaced by:

#Qlogic-FCOE: Uncomment the below lines if using qlogic fcoe boot from san

The values will be overwritten on running the cobbler-deploy play.

### Note

It is highly recommended as part of a Red Hat install that only one disk is presented to the node during the install phase. **You should ensure that any additional LUNs that are required are attached and visible on the compute hosts before running the site.yml playbook.**

## Installing the HPE Helion OpenStack 5.0 iso for nodes that support Boot from SAN

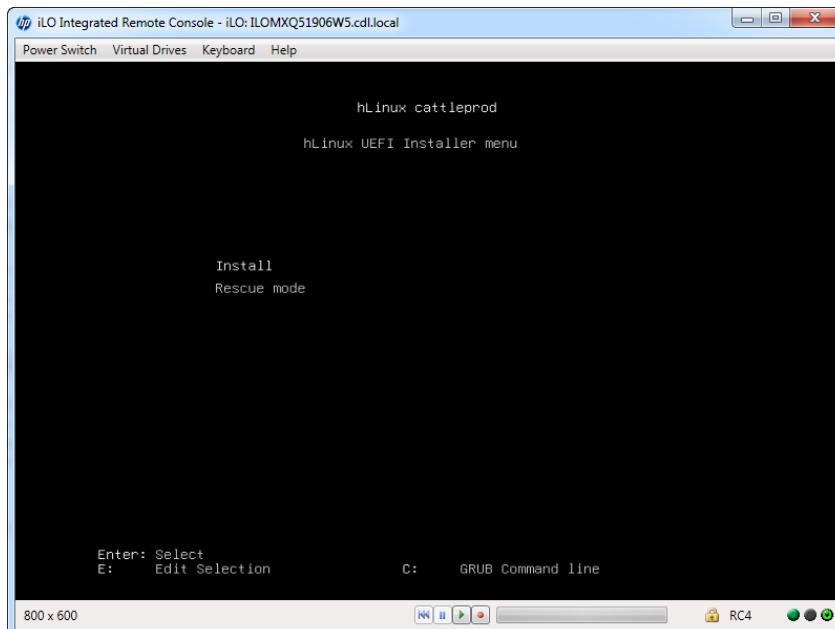
This section describes how to install the iso on the Lifecycle Manager to support the following configurations:

- the section called “UEFI Boot Mode”
- the section called “UEFI Boot Mode with QLogic FCoE”
- the section called “Legacy BIOS Boot Mode”
- the section called “Legacy BIOS Boot Mode with QLogic FCoE”

The lifecycle manager will then automatically handle the installation on nodes that supports Boot from SAN based on the configuration information in servers.yml and the disk models, as described in the preceding section.

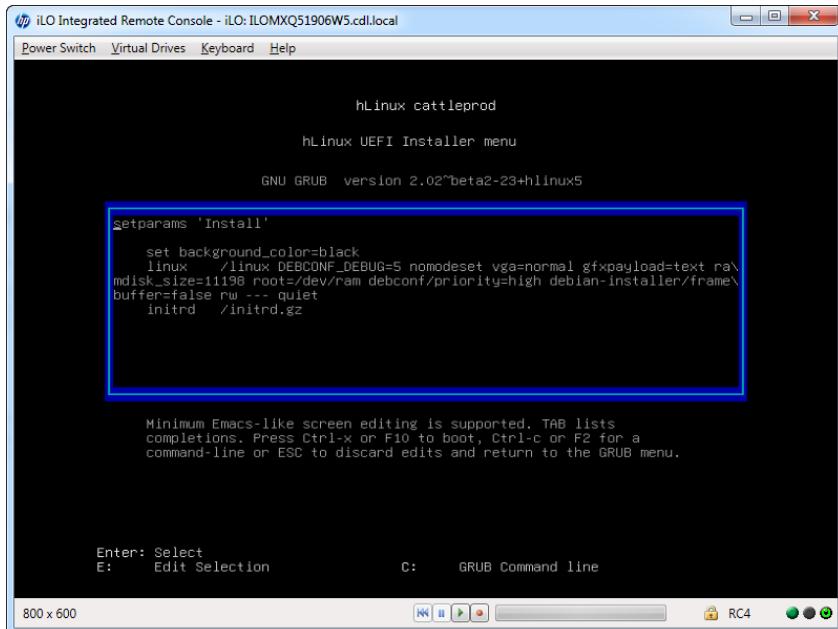
### UEFI Boot Mode

On the installer boot menu, press **E** to enter the **Edit Selection** screen.

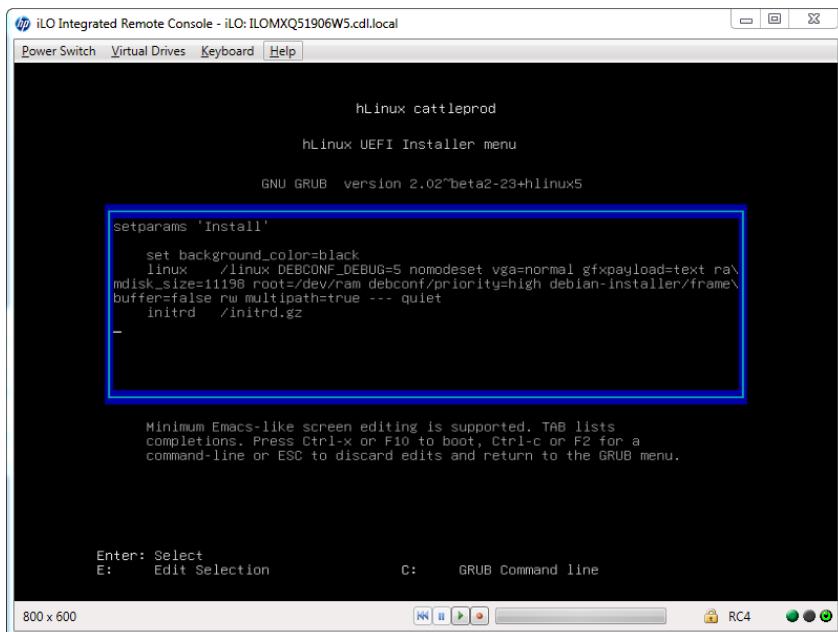


## Boot from SAN and Multipath Configuration

This brings up the `Edit Selection` screen.



Enter the text `multipath=true` before `--- quiet`:



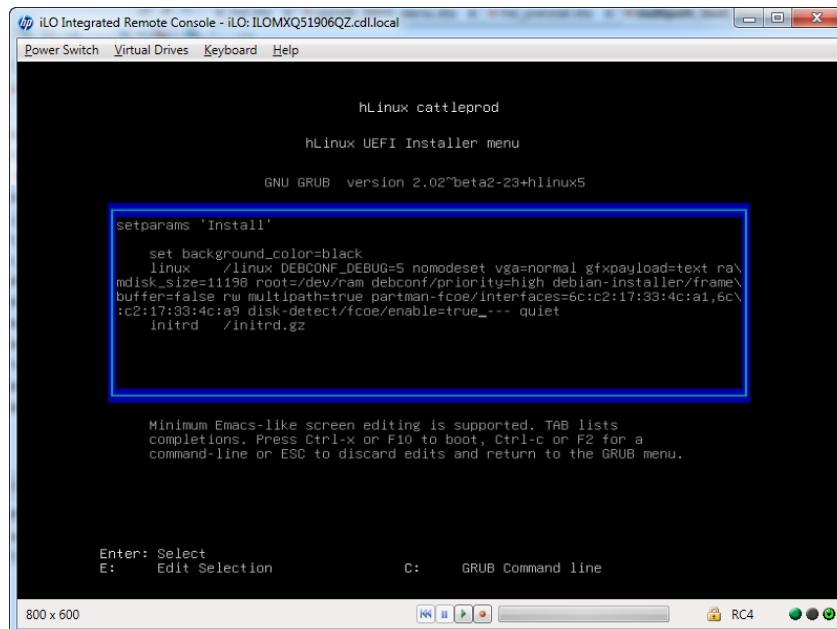
Press **Ctrl+X** or **F10** to proceed with the install.

## UEFI Boot Mode with QLogic FCoE

At the installer boot menu, press **E** to enter the `Edit Selection` screen as described in the preceding section. In addition to inserting `multipath=true`, it is necessary to supply details of the FCoE network interfaces. In the example below, the interfaces are specified as:

## Boot from SAN and Multipath Configuration

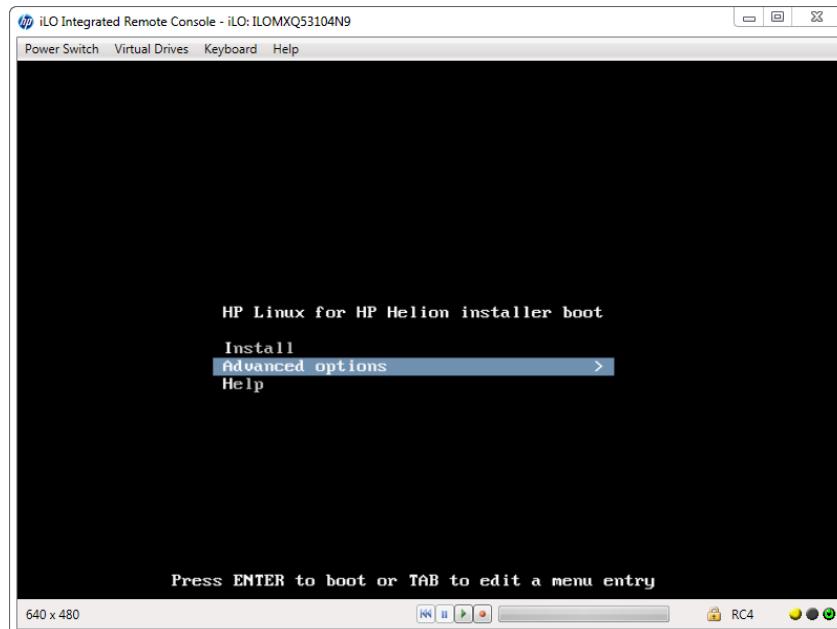
partman-fcoe/interfaces=6c:c2:17:33:4c:a1,6c:c2:17:33:4c:a9 disk-detect/fcoe/enab



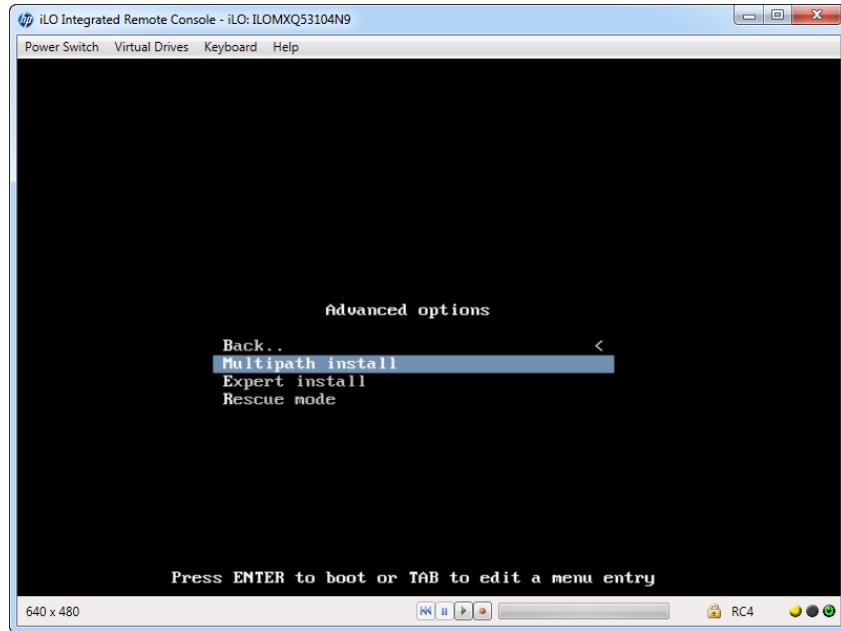
Press **Ctrl+X** or **F10** to proceed with the install.

## Legacy BIOS Boot Mode

On the installer boot menu, navigate (using **↓**) to the **Advanced options** entry and then press **Enter**.



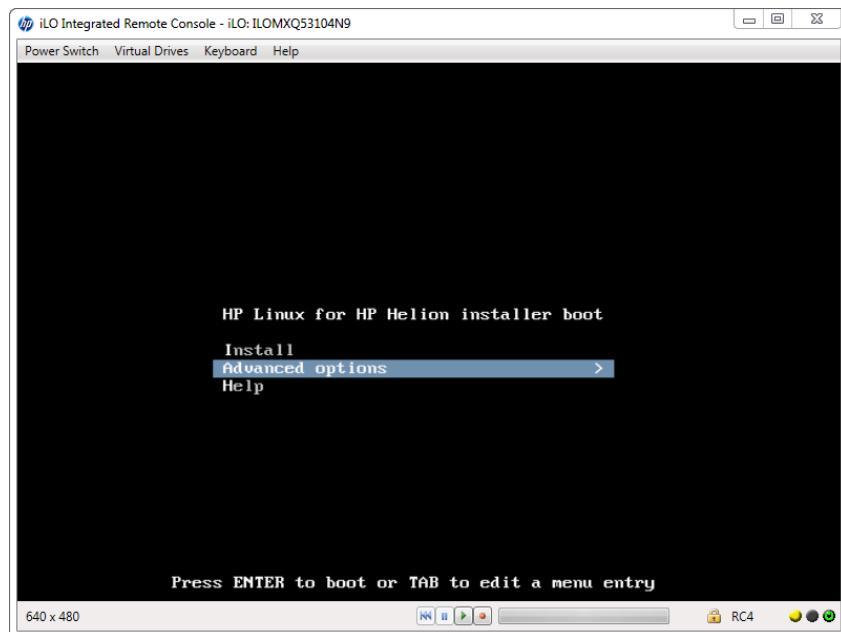
This will bring up the Advanced options menu:



Navigate to the Multipath install entry and press **Enter** to start the installation.

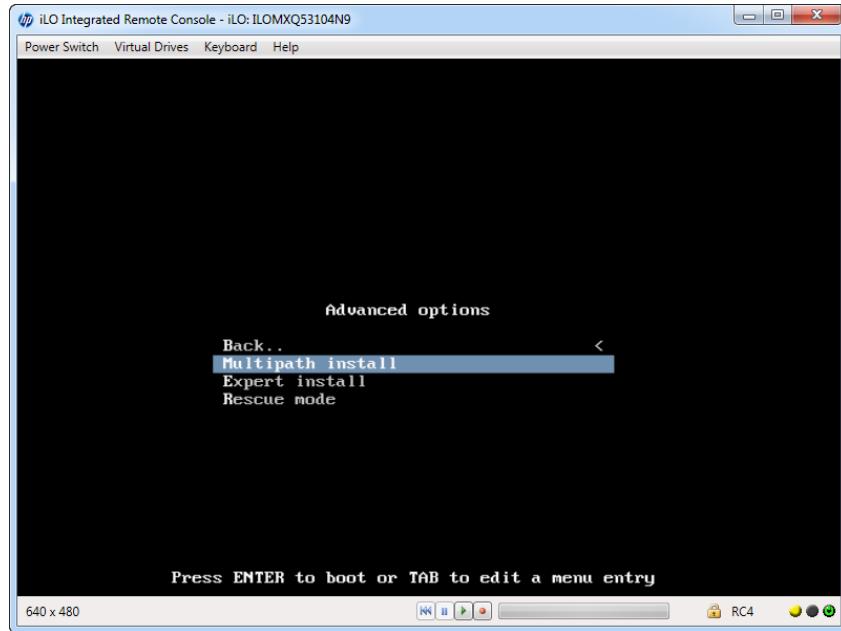
## Legacy BIOS Boot Mode with QLogic FCoE

On the installer boot menu, navigate (using ↓) to the Advanced options entry and then press **Enter**.

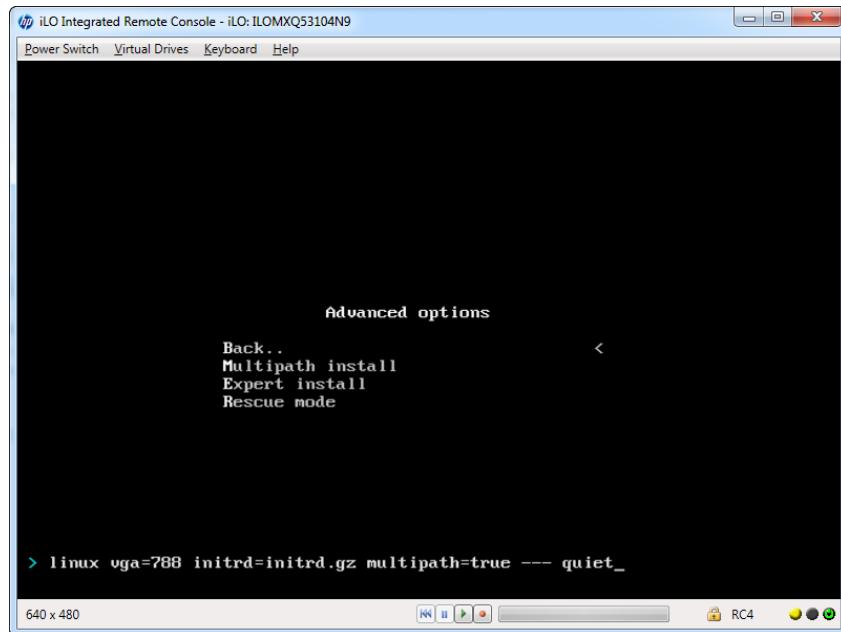


This will bring up the Advanced options menu:

## Boot from SAN and Multipath Configuration



Navigate to the Multipath install entry and press **→** to edit the entry details. Notice that the kernel command line is now displayed at the bottom of the screen and that multipath=true is already specified.

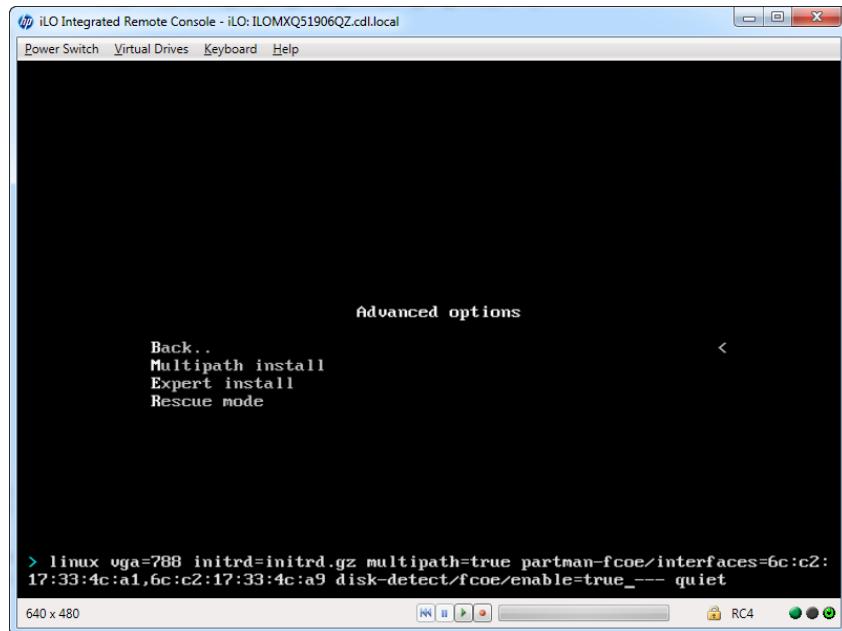


Edit the kernel command line to add the FCoE network interfaces before the **---**. In the example below, the interfaces are specified as:

```
partman-fcoe/interfaces=6c:c2:17:33:4c:a1,6c:c2:17:33:4c:a9 disk-detect/fcoe/enab
```

## Boot from SAN and Multipath Configuration

---



Now press **Enter** to start the installation.

---

# **Chapter 5. Installing the L2 Gateway Agent for the Networking Service**

## **Introduction**

The L2 gateway is a service plug-in to the Neutron networking service that allows two L2 networks to be seamlessly connected to create a single L2 broadcast domain. The initial implementation provides for the ability to connect a virtual Neutron VxLAN network to a physical VLAN using a VTEP-capable HPE 5930 switch. The L2 gateway is to be enabled only for VxLAN deployments.

To begin L2 gateway agent setup, you need to configure your switch. These instructions use an HPE FlexFabric 5930 Switch Series [<http://www8.hp.com/us/en/products/networking-switches/product-detail.html?oid=6604154#!tab=models>] switch.

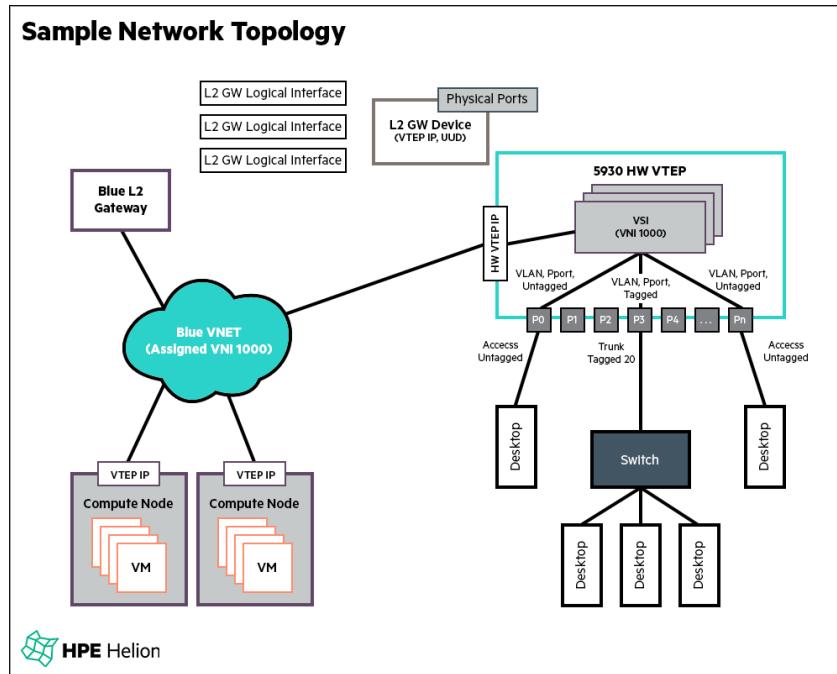
- Use case
- Sample network topology
- Networks
- HPE 5930 Switch Configuration
- Enabling and configuring L2 gateway agent
- Routing between software and hardware VTEP networks
- Connecting baremetal server to HPE 5930 switch
- Configuration on baremetal server to receive tagged packets from the switch
- NIC bonding and IRF configuration
- Scale numbers tested
- L2 gateway commands

## **Sample network topology (for illustration purposes)**

When viewing the following network diagram, assume that the blue VNET has been created by the tenant and has been assigned a segmentation ID of 1000 (VNI 1000). The Cloud Admin is now connecting physical servers to this VNET.

Assume also that the blue L2 Gateway is created and points to the HW VTEPS, the physical ports, the VLAN, and if it is an access or trunk port (tagged or untagged)

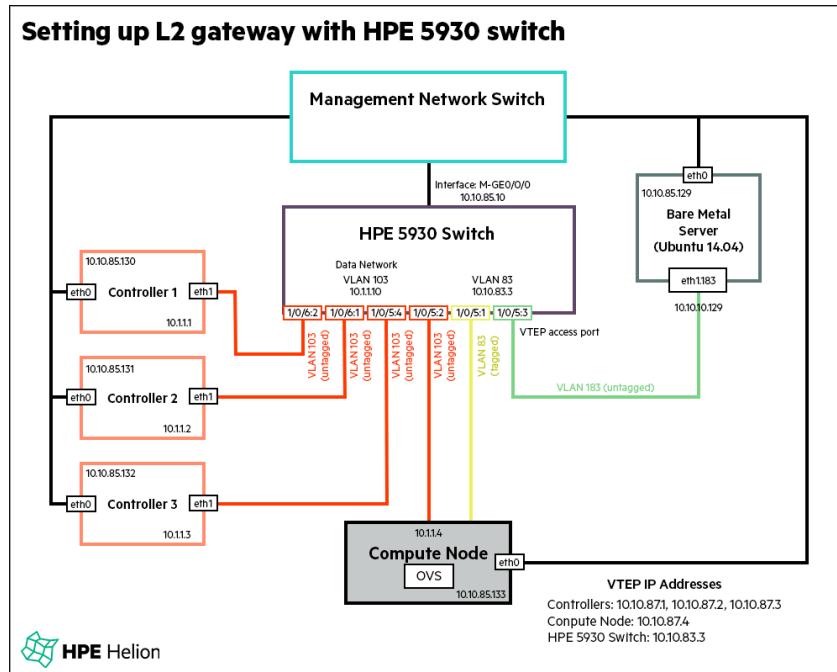
**Figure 5.1. Figure 1**



## Networks

The following diagram illustrates an example network configuration.

**Figure 5.2. Figure 2**



An L2 gateway is useful in extending virtual networks (VxLAN) in a cloud onto physical VLAN networks. The L2 gateway switch converts VxLAN packets into VLAN packets and back again, as shown in the following diagram. This topic assumes a VxLAN deployment.

**Figure 5.3. Figure 3**



- Management Network: 10.10.85.0/24
- Data Network: 10.1.1.0/24
- Tenant VM Network: 10.10.10.0/24

Note: These IP ranges are used in the topology shown in the diagram for illustration only.

## HPE 5930 switch configuration

1. Telnet to the 5930 switch and provide your username and password.

2. Go into system view:

```
system-view
```

3. Create the required VLANs and VLAN ranges:

```
vlan 103
vlan 83
vlan 183
vlan 1261 to 1270
```

4. Assign an IP address to VLAN 103. This is used as a data path network for VxLAN traffic.

```
interface vlan 103
ip address 10.1.1.10 255.255.255.0
```

5. Assign an IP address to VLAN 83. This is used as a hardware VTEP network.

```
interface vlan 83
ip address 10.10.83.3 255.255.255.0
```

The 5930 switch has a fortygigE1/0/5 interface to which a splitter cable is connected that splits the network it into four tengigEthernet (tengigEthernet1/0/5:1 to tengigEthernet1/0/5:4) interfaces:

- engigEthernet1/0/5:1 and tengigEthernet1/0/5:2 are connected to the compute node. This is required just to bring the interface up. In other words, in order to have the HPE 5930 switch work as a router, there should be at least one interface of that particular VLAN up. Alternatively, the interface can be connected to any host or network element.
- tengigEthernet1/0/5:3 is connected to a baremetal server.

- tengigEthernet1/0/5:4 is connected to controller 3, as shown in Figure 2.

The switch's fortygigE1/0/6 interface to which the splitter cable is connected splits it into four tengigEthernet (tengigEthernet1/0/6:1 to tengigEthernet1/0/6:4) interfaces:

- tengigEthernet1/0/6:1 is connected to controller 2
- tengigEthernet1/0/6:2 is connected to controller 1

Note: 6:3 and 6:4 are not used although they are available.

6. Split the fortygigE 1/0/5 interface into tengig interfaces:

```
interface fortygigE 1/0/5
using tengig
The interface FortyGigE1/0/5 will be deleted. Continue? [Y/N]: y
```

7. Configure the Ten-GigabitEthernet1/0/5:1 interface:

```
interface Ten-GigabitEthernet1/0/5:1
port link-type trunk
port trunk permit vlan 83
```

## Configuring the provider data path network

1. Configure the Ten-GigabitEthernet1/0/5:2 interface:

```
interface Ten-GigabitEthernet1/0/5:2
port link-type trunk
port trunk permit vlan 103
port trunk permit vlan 1261 to 1270
port trunk pvid vlan 103
```

2. Configure the Ten-GigabitEthernet1/0/5:4 interface:

```
interface Ten-GigabitEthernet1/0/5:4
port link-type trunk
port trunk permit vlan 103
port trunk permit vlan 1261 to 1270
port trunk pvid vlan 103
```

3. Configure the Ten-GigabitEthernet1/0/5:3 interface:

```
interface Ten-GigabitEthernet1/0/5:3
port link-type trunk
port trunk permit vlan 183
vtep access port
```

4. Split the fortygigE 1/0/6 interface into tengig interfaces:

```
interface fortygigE 1/0/6
using tengig
The interface FortyGigE1/0/6 will be deleted. Continue? [Y/N]: y
```

5. Configure the Ten-GigabitEthernet1/0/6:1 interface:

```
interface Ten-GigabitEthernet1/0/6:1
port link-type trunk
port trunk permit vlan 103
port trunk permit vlan 1261 to 1270
port trunk pvid vlan 103
```

6. Configure the Ten-GigabitEthernet1/0/6:2 interface:

```
interface Ten-GigabitEthernet1/0/6:2
port link-type trunk
port trunk permit vlan 103
port trunk permit vlan 1261 to 1270
port trunk pvid vlan 103
```

7. Enable l2vpn:

```
l2vpn enable
```

8. Configure a passive TCP connection for OVSDB on port 6632:

```
ovsdb server ptcp port 6632
```

9. Enable OVSDB server:

```
ovsdb server enable
```

10. Enable a VTEP process:

```
vtep enable
```

11. Configure 10.10.83.3 as the VTEP source IP. This acts as a hardware VTEP IP.

```
tunnel global source-address 10.10.83.3
```

12. Configure the VTEP access port:

```
interface Ten-GigabitEthernet1/0/5:3
vtep access port
```

13. Disable VxLAN tunnel mac-learning:

```
vxlan tunnel mac-learning disable
```

14. Display the current configuration of the 5930 switch and verify the configuration:

```
display current-configuration
```

After switch configuration is complete, you can dump OVSDB to see the entries.

1. Run the ovsdb-client from any Linux machine reachable by the switch:

```
ovsdb-client dump --pretty tcp:10.10.85.10:6632
```

```
sdn@small-hLinux:~$ ovsdb-client dump --pretty tcp:10.10.85.10:6632
Arp_Sources_Local table
 _uuid locator src_mac
----- ----- -----
```

## Installing the L2 Gateway Agent for the Networking Service

```
Arp_Sources_Remote table
_uuid locator src_mac
-----
Global table
_uuid                                         managers switches
-----
2c891edc-439b-4144-84d9-fa4cd88092bf [ ]           [f5f4b43b-40bc-4640-b580-d4b12011f
Logical_Binding_Stats table
_uuid bytes_from_local bytes_to_local packets_from_local packets_to_local
-----
Logical_Router table
_uuid description name static_routes switch_binding
-----
Logical_Switch table
_uuid description name tunnel_key
-----
Manager table
_uuid inactivity_probe is_connected max_backoff other_config status target
-----
Mcast_Macs_Local table
MAC _uuid ipaddr locator_set logical_switch
-----
Mcast_Macs_Remote table
MAC _uuid ipaddr locator_set logical_switch
-----
Physical_Locator table
_uuid dst_ip encapsulation_type
-----
Physical_Locator_Set table
_uuid locators
-----
Physical_Port table
_uuid                                         description name
po
fda90870-656c-479b-91ee-852b70e2007e " "           "Ten-GigabitEthernet1/0/5:3" [U
Physical_Switch table
_uuid                                         description management_ips name      ports
f5f4b43b-40bc-4640-b580-d4b12011f688 " "           [ ]           "L2GTWY02" [fda9
Tunnel table
_uuid bfd_config_local bfd_config_remote bfd_params bfd_status local remote
-----
```

```
Ucast_Macs_Local table
MAC _uuid ipaddr locator logical_switch
--- ----- ----- -----
```

```
Ucast_Macs_Remote table
MAC _uuid ipaddr locator logical_switch
--- ----- ----- -----
```

## Enabling and configuring the L2 gateway agent

1. Update the input model (in `control_plane.yml`) to specify where you want to run the neutron-l2gateway-agent. For example, see the line in bold in the following yml:

```
---
product:
version: 2
control-planes:
- name: cp
  region-name: region1
failure-zones:
- AZ1
  common-service-components:
    - logging-producer
    - monasca-agent
    - freezer-agent
    - stunnel
    - lifecycle-manager-target
clusters:
- name: cluster1
  cluster-prefix: c1
  server-role: ROLE-CONTROLLER
  member-count: 2
  allocation-policy: strict
service-components:

...
- neutron-l2gateway-agent
... )
```

2. Update `l2gateway_agent.ini.j2`. For example, here the IP address (10.10.85.10) must be the management IP address of your 5930 switch. Open the file in vi:

```
$ vi /home/stack/my_cloud/config/neutron/l2gateway_agent.ini.j2
```

3. Then make the changes:

```
[ovsdb]
# (StrOpt) OVSDB server tuples in the format
# <ovsdb_name>:<ip address>:<port>[,<ovsdb_name>:<ip address>:<port>]
# - ovsdb_name: a symbolic name that helps identifies keys and certificate files
# - ip address: the address or dns name for the ovsdb server
# - port: the port (ssl is supported)
ovsdb_hosts = hardware_vtep:10.10.85.10:6632
```

4. By default, the L2 gateway agent initiates a connection to OVSDB servers running on the L2 gateway switches. Set the attribute `enable_manager` to `True` if you want to change this behavior (to make L2 gateway switches initiate a connection to the L2 gateway agent). In this case, it is assumed that the Manager table in the OVSDB hardware\_vtep schema on the L2 gateway switch has been populated with the management IP address of the L2 gateway agent and the port.

```
#enable_manager = False
#connection can be initiated by the ovsdb server.
#By default 'enable_manager' value is False, turn on the variable to True
#to initiate the connection from ovsdb server to l2gw agent.
```

5. If the port that is configured with `enable_manager = True` is any port other than 6632, update the `2.0/services/neutron/l2gateway-agent.yml` input model file with that port number:

```
endpoints:
  - port: '6632'
    roles:
      - ovsdb-server
```

6. Note: The following command can be used to set the Manager table on the switch from a remote system:

```
sudo vtep-ctl --db=tcp:10.10.85.10:6632 set-manager tcp:10.10.85.130:6632
```

7. For SSL communication, the command is:

```
sudo vtep-ctl --db=tcp:10.10.85.10:6632 set-manager ssl:10.10.85.130:6632
```

where **10.10.85.10** is the management IP address of the L2 gateway switch and **10.10.85.130** is the management IP of the host on which the L2 gateway agent runs.

Therefore, in the above topology, this command has to be repeated for **10.10.85.131** and **10.10.85.132**.

8. If you are not using SSL, comment out the following:

```
#l2_gw_agent_priv_key_base_path={{ neutron_l2gateway_agent_creds_dir }}/keys
#l2_gw_agent_cert_base_path={{ neutron_l2gateway_agent_creds_dir }}/certs
#l2_gw_agent_ca_cert_base_path={{ neutron_l2gateway_agent_creds_dir }}/ca_certs
```

9. If you are using SSL, then rather than commenting out the attributes, specify the directory path of the private key, the certificate, and the CA cert that the agent should use to communicate with the L2 gateway switch which has the OVSDB server enabled for SSL communication.

Make sure that the directory path of the files is given permissions 755, and the files' owner is root and the group is root with 644 permissions.

**Private key:** The name should be the same as the symbolic name used above in `ovsdb_hosts` attribute. The extension of the file should be ".key". With respect to the above example, the filename will be `hardware_vtep.key`

**Certificate** The name should be the same as the symbolic name used above in `ovsdb_hosts` attribute. The extension of the file should be ".cert". With respect to the above example, the filename will be `hardware_vtep.cert`

**CA certificate** The name should be the same as the symbolic name used above in `ovsdb_hosts` attribute. The extension of the file should be ".ca\_cert". With respect to the above example, the filename will be `hardware_vtep.ca_cert`

10.To enable the HPE 5930 switch for SSL communication, execute the following commands:

```
undo ovsdb server ptcp
undo ovsdb server enable
ovsdb server ca-certificate flash:/cacert.pem bootstrap
ovsdb server certificate flash:/sc-cert.pem
ovsdb server private-key flash:/sc-privkey.pem
ovsdb server pssl port 6632
ovsdb server enable
```

11.Data from the OVSDB sever with SSL can be viewed using the following command:

```
ovsdb-client -C <ca-cert.pem> -p <client-private-key.pem> -c <client-cert.pem> d
```

## Routing between software and hardware

### VTEP networks

In order to allow L2 gateway switches to send VxLAN packets over the correct tunnels destined for the compute node and controller node VTEPs, you must ensure that the cloud VTEP (compute and controller) IP addresses are in different network/subnet from that of the L2 gateway switches. You must also create a route between these two networks. This is explained below.

1. In the following example of the input model file `networks.yml`, the GUEST-NET represents the cloud data VxLAN network. REMOTE-NET is the network that represents the hardware VTEP network.

```
# networks.yml
networks:
  - name: GUEST-NET
    vlanid: 103
    tagged-vlan: false
    cidr: 10.1.1.0/24
    gateway-ip: 10.1.1.10
    network-group: GUEST

  - name: REMOTE-NET
    vlanid: 183
    tagged-vlan: false
    cidr: 10.10.83.0/24
    gateway-ip: 10.10.83.3
    network-group: REMOTE
```

2. The route must be configured between the two networks in the `network-groups.yml` input model file:

```
# network_groups.yml
network-groups:
  - name: REMOTE
    routes:
      - GUEST

  - name: GUEST
    hostname-suffix: guest
    tags:
```

```
- neutron.networks.vxlan:  
    tenant-vxlan-id-range: "1:5000"  
routes:  
    - REMOTE
```

**Note that the IP route is configured on the compute node. Per this route, the HPE 5930 acts as a gateway that routes between the two networks.**

3. On the compute node, it looks like this:

```
stack@padawan-cpl-comp0001-mgmt:~$ sudo ip route  
10.10.83.0/24 via 10.1.1.10 dev eth4
```

4. Run the following Ansible playbooks to apply the changes.

config-processor-run.yml:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

ready-deployment.yml:

```
ansible-playbook -i hosts/localhost ready-deployment.yml
```

hlm-reconfigure.yml:

```
cd ~/scratch/ansible/next/hos/ansible/  
ansible-playbook -i hosts/verb_hosts hlm-reconfigure.yml
```

**Notes:**

1. Make sure that the controller cluster is able to reach the management IP address of the L2 gateway switch. Otherwise, the L2 gateway agents running on the controllers will not be able to reach the gateway switches.
2. Make sure that the interface on the baremetal server connected to the 5930 switch is tagged (this is explained shortly).

## Connecting a baremetal server to the HPE 5930 switch

As Ubuntu (baremetal server) is not aware of tagged packets, it is not possible for a virtual machine to communicate with a baremetal box.

You (the administrator) must manually perform configuration changes to the interface in the HPE 5930 switch to which the baremetal server is connected so that the switch can send untagged packets. Either one of the following command sets can be used to do so:

```
Interface <interface number>  
service-instance <service-instance id>  
encapsulation untagged  
xconnect vsi <vsi-name>
```

Or:

```
Interface <interface number>
service-instance <service-instance id>
encapsulation s-vid <vlan-id>
xconnect vsi <vsi-name> access-mode ethernet
```

There are two ways of configuring the baremetal server to communicate with virtual machines. If the switch sends tagged traffic, then the baremetal server should be able to receive the tagged traffic.

## Configuration on a baremetal server

### To receive tagged packets from the switch

If the configuration changes mentioned previously are not made on the switch to send untagged traffic to the baremetal server, then the following changes are required on the baremetal server so that it can receive tagged traffic from the switch.

Bare metal Management ip: 10.10.85.129 on interface em1 Switch 5930 is connected to baremetal on eth1  
Need to set the IP address into tagged interface of eth1

1. create tagged (VLAN 183) interface

```
vconfig add eth1 183
```

2. Assign the IP address (10.10.10.129) to eth1.183 tagged interface (IP from the subnet 10.10.10.0/24 as VM (10.10.10.4) spawned in Compute node belongs to this subnet).

```
ifconfig eth1.183 10.10.10.129/24
```

## NIC bonding and IRF configuration

With L2 gateway in actiondeployment, NIC bonding can be enabled on compute nodes. For more details on NIC bonding, please refer to the section called “Interface Models”. In order to achieve high availability, HPE 5930 switches can be configured to form a cluster using Intelligent Resilient Framework (IRF). Please refer to the HPE FlexFabric 5930 Switch Series configuraton guide [<http://h20564.www2.hpe.com/hpsc/doc/public/display?docId=c04567141>] for details.

## Scale numbers tested

- Number of neutron port MACs tested on a single switch: 4000
- Number of HPE 5930 switches tested: 2
- Number of baremetal connected to a single HPE 5930 switch: 100
- Number of L2 gateway connections to different networks: 800

## L2 gateway commands

These commands are not part of the L2 gateway deployment. They are to be executed after L2 gateway is deployed.

1. Create Network

```
neutron net-create net1
```

2. Create Subnet

```
neutron subnet-create net1 10.10.10.0/24
```

3. Boot a tenant VM ( nova boot --image <Image-id> --flavor 2 --nic net-id=<net\_id> <VM name>)

```
nova boot --image 1f3cd49d-9239-49cf-8736-76bac5360489 --flavor 2 --nic net-id=4
```

Assume the VM was assigned the IP address 10.10.10.4

4. Create the L2 gateway filling in your information here:

```
neutron l2-gateway-create --device name="<switch name>" ,interface_names="<interf
```

For this example:

```
neutron l2-gateway-create --device name="L2GTWY02" ,interface_names="Ten-GigabitE
```

Ping from the VM (10.10.10.4) to baremetal server and from baremetal server (10.10.10.129) to the VM Ping should not work as there is no gateway connection created yet.

5. Create l2 gateway Connection

```
neutron l2-gateway-connection-create gw1 net1 --segmentation-id 183
```

6. Ping from VM (10.10.10.4) to baremetal and from baremetal (10.10.10.129) to VM Ping should work.

7. Delete l2 gateway Connection

```
neutron l2-gateway-connection-delete <gateway id/gateway_name>
```

8. Ping from the VM (10.10.10.4) to baremetal and from baremetal (10.10.10.129) to the VM. Ping should not work as l2 gateway connection was deleted.

---

## **Part II. Cloud Installation**

---

# Table of Contents

6. Overview .....	43
7. Installing via the GUI .....	45
Before you begin .....	46
Creating a CSV file for import .....	46
Run the GUI to install your cloud .....	47
Use the installer securely .....	47
Final Steps .....	48
8. DNS Service Installation Overview .....	49
Getting Started .....	49
Install the DNS Service with PowerDNS .....	49
Install DNS Service with PowerDNS .....	49
Configure the Backend .....	49
Install the DNS Service with BIND .....	51
Install DNS Service with BIND .....	51
Configure the Backend .....	51
Install the DNS Service with InfoBlox .....	51
Prerequisites .....	52
Configure the Backend .....	52
Configure DNS Domain and NS Records .....	55
9. Magnum Overview .....	57
Magnum Architecture .....	57
Install the Magnum Service .....	65
Installing Magnum as part of new HPE Helion OpenStack 5.0 environment .....	66
Adding Magnum to existing HPE Helion OpenStack 5.0 environment .....	66
Integrate Magnum with the DNS Service .....	67
10. Installing Mid-scale and Entry-scale KVM .....	70
Important Notes .....	70
Before You Start .....	71
Setting Up the Lifecycle Manager .....	71
Configuring Your Environment .....	73
Provisioning Your Baremetal Nodes .....	74
Running the Configuration Processor .....	76
Configuring TLS .....	77
Deploying the Cloud .....	77
Configuring a Block Storage Backend (Optional) .....	78
Post-Installation Verification and Administration .....	78
11. Installation for Helion Entry Scale ESX (with OVSvApp, KVM with VSA Model) .....	79
Important Notes .....	79
Before You Start .....	80
Prerequisites .....	80
Deploy ESX Cloud with OVSvApp .....	81
Procedure to Deploy ESX Cloud with OVSvApp .....	81
Setting Up the Lifecycle Manager .....	81
Prepare and Deploy Cloud Controllers .....	83
Provisioning Your Baremetal Nodes .....	84
Running the Configuration Processor .....	86
Deploying the Cloud .....	86
Prepare and Deploy ESX Computes and OVSvAPPs .....	87
Validate the block storage .....	92
Validate the compute .....	92
Validate the neutron .....	93

12. Sample activationtemplate.json File for ESX Compute .....	94
13. Installing Baremetal (Ironic) .....	107
Installation for HPE Helion Entry-scale Cloud with Ironic Flat Network .....	107
Before You Start .....	107
Setting Up the Lifecycle Manager .....	107
Configure Your Environment .....	109
Provisioning Your Baremetal Nodes .....	110
Running the Configuration Processor .....	112
Deploying the Cloud .....	112
Ironic configuration .....	113
Node Configuration .....	114
TLS Certificates with Ironic Python Agent (IPA) Images .....	114
Ironic in Multiple Control Plane .....	116
Networking for Baremetal in Multiple Control Plane .....	117
Handling Optional Swift Service .....	117
Instance Provisioning .....	117
Provisioning Baremetal Nodes with Flat Network Model .....	118
Supplied Images .....	119
Provisioning a node .....	119
Creating a node using agent_ilc .....	119
Creating a node using agent_ipmi .....	121
Create Flavor .....	122
Create network port .....	123
Create Glance Image .....	123
Generate Key Pair .....	124
Determine the neutron network ID .....	124
Boot the node .....	124
Create Glance Images using diskimage-builder (DIB) .....	125
Creating BIOS images for RHEL .....	126
Creating BIOS images for Ubuntu .....	126
Creating UEFI images .....	126
Provisioning Baremetal Nodes with Multi-Tenancy .....	127
View Ironic System Details .....	132
View details about the server using <b>nova show &lt;nova-node-id&gt;</b> .....	132
View detailed information about a node using <b>ironic node-show &lt;ironic-node-id&gt;</b> ..	133
View detailed information about a port using <b>ironic port-show &lt;ironic-port-id&gt;</b> .....	134
View detailed information about a hypervisor using <b>nova hypervisor-list</b> and <b>nova hypervisor-show</b> .....	134
View a list of all running services using <b>nova service-list</b> .....	135
Troubleshooting Ironic Installation .....	135
No valid host was found. There are not enough hosts available. ....	136
Deployment to a node fails and in "ironic node-list" command, the power_state column for the node is shown as "None" .....	138
Error Downloading Image .....	139
Using node-inspection can cause temporary claim of IP addresses .....	139
Node permanently stuck in deploying state .....	139
The NICs in the baremetal node should come first in boot order .....	139
Increase in the number of nodes can cause power commands to fail .....	140
DHCP succeeds with PXE but times out with iPXE .....	140
Node Cleaning .....	141
Setup .....	142
In use .....	142
Troubleshooting .....	143
Disabling Node Cleaning .....	143

Ironic and OneView .....	143
Enabling Ironic OneView driver in HPE Helion OpenStack .....	143
Adding OneView Appliance Credentials .....	144
Encrypting the OneView Password .....	144
Decrypting the OneView Password .....	144
Registering Baremetal Node for OneView Driver .....	144
Updating Node Properties .....	145
Creating Port for Driver .....	145
Creating a Node .....	145
Getting Data using REST API .....	145
Ironic OneView CLI .....	145
RAID Configuration for Ironic .....	146
Audit Support for Ironic .....	150
API Audit Logging .....	150
Enabling API Audit Logging .....	151
Sample Audit Event .....	151
14. Installation for HPE Helion Entry-scale Cloud with Swift Only .....	153
Important Notes .....	153
Before You Start .....	153
Setting Up the Lifecycle Manager .....	153
Configure Your Environment .....	156
Running the Configuration Processor .....	157
Provisioning Your Baremetal Nodes .....	158
Deploying the Cloud .....	159
Post-Installation Verification and Administration .....	160
15. Installing SLES Compute .....	161
SLES Compute Node Installation Overview .....	161
SLES Support .....	161
Using the Lifecycle Manager to Deploy SLES Compute Nodes .....	161
Deploying legacy BIOS SLES compute nodes .....	162
Provisioning SLES Yourself .....	162
Introduction .....	162
Configure Lifecycle Manager to Enable SLES .....	162
Install SLES 12 SP2 .....	163
Assign a static IP .....	163
Add stack user and home directory .....	164
Allow user stack to sudo without password .....	164
Add zypper repository .....	164
Add Required Packages .....	164
Set up passwordless SSH access .....	164
Using SLES as a Ceph Client .....	165
16. HLM-Hypervisor instructions .....	166
Introduction .....	166
Model changes .....	166
passthrough-network-groups .....	166
hlm-hypervisor .....	167
Bootstrap Instructions .....	173
Customizing for VCP .....	173
Installing a fully input model managed Virtual Control Plane based HPE Helion	
OpenStack Cloud .....	173
Provisioning just the Virtual Control Plane VMs .....	174
Manually running each phase of the Virtual Control Plane provisioning .....	174
NIC Mapping for HLM Virtual-Controllers .....	175
Recommended Mappings .....	176

Notification of actions to be taken post upgrade .....	176
Reboot of a HLM-Hypervisor node .....	176
Staged install on HLM-Hypervisor individual play books .....	177
Virtual Controller Replacement .....	177
Ansible host-specific variables for network-group access .....	178
Example 3rd-Party Ansible for Network Configuration Access .....	181
Ansible host-specific variables for gluster .....	182
"Empty" HLM Hypervisor Node .....	182
Monasca Monitoring of HLM Hypervisors .....	182
Monitored Metrics .....	182
Configured Alarms .....	183
HLM Hypervisor Monitoring Configuration .....	183
17. Integrations .....	185
Ceph Overview .....	185
Overview .....	185
Deployment Architecture .....	186
Ceph Networking .....	186
Placement of service component .....	187
Ceph Deployment Architecture .....	188
Alternative supported architecture .....	189
Core networking .....	189
Placement of service components .....	190
Hardware recommendations .....	190
Ceph Deployment and Configurations .....	190
Alternative Supported Choices .....	201
Usage of Ceph Storage .....	214
Managing Ceph Clusters After Deployment .....	227
Configuring for VSA Block Storage Backend .....	227
Prerequisites .....	228
Notes .....	228
Configure HPE StoreVirtual VSA .....	228
Using Ansible .....	229
Using the CMC Utility .....	230
Configure VSA as the Backend .....	241
Post-Installation Tasks .....	244
Configuring for 3PAR Block Storage Backend .....	244
Prerequisites .....	244
Notes .....	245
Multipath Support .....	245
Configure 3PAR FC as a Cinder Backend .....	246
Configure 3PAR iSCSI as Cinder backend .....	247
Post-Installation Tasks .....	249
Ironic OneView Integration .....	249
Prerequisites .....	249
Integrating with OneView .....	249
Registering Node in OneView .....	250
Provisioning Ironic Node .....	251
18. Troubleshooting the Installation .....	257
Issues during Lifecycle-manager Setup .....	257
Issues while Provisioning your Baremetal Nodes .....	257
Issues while Updating Configuration Files .....	262
Issues while Deploying the Cloud .....	263
19. Troubleshooting the Block Storage Backend Configuration .....	268
20. Troubleshooting the ESX .....	270

Issue: If hlm-ux-services.service is not running, the EON resource-activate and resource-de- activate commands fails .....	270
Issue: ESX onboard Compute .....	270
Issue: Migrate eon-conductor .....	270
Issue: ESX Cluster shows UNKNOWN in OpsConsole .....	273
Issue: Unable to view the VM console in Horizon UI .....	273

---

# Chapter 6. Overview

Before jumping into your installation, we recommend taking the time to read through our Chapter 10, *Example Configurations* documentation to get an overview of the sample configurations HPE Helion OpenStack 5.0 offers. We have highly tuned example configurations for each of these Cloud models:

- the section called “Entry-scale KVM with VSA Model”
- 
- the section called “Entry-scale Swift Model”
- 
- the section called “Mid-scale KVM with VSA Model”

Name	Location
the section called “Entry-scale KVM with VSA Model”	<code>~/helion/examples/entry-scale-kvm-vsa</code>
the section called “Entry-scale KVM with VSA model with Dedicated Cluster for Metering, Monitoring, and Logging”	<code>~/helion/examples/entry-scale-kvm-vsa-mml</code>
the section called “Entry-scale KVM with Ceph Model”	<code>~/helion/examples/entry-scale-kvm-ceph</code>
the section called “Mid-scale KVM with VSA Model”	<code>~/helion/examples/mid-scale-kvm-vsa</code>
the section called “Entry-scale ESX, KVM with VSA Model”	<code>~/helion/examples/entry-scale-esx-kvm-vsa</code>
the section called “Entry-scale ESX, KVM with VSA Model with Dedicated Cluster for Metering, Monitoring, and Logging”	<code>~/helion/examples/entry-scale-esx-kvm-vsa-mml</code>
the section called “Entry-scale Swift Model”	<code>~/helion/examples/entry-scale-swift</code>
the section called “Entry-scale Cloud with Ironic Flat Network”	<code>~/helion/examples/entry-scale-ironic-flat-network</code>
the section called “Entry-scale Cloud with Ironic Multi-Tenancy”	<code>~/helion/examples/entry-scale-ironic-multi-tenancy</code>

## Using the Command-line

You should use the command-line if:

- You are installing a more complex or large-scale cloud.
- You have more than 15 nodes.
- You need to use availability zones or the server groups functionality of the cloud model. See the for more information.
- You want to customize the cloud configuration beyond the tuned defaults that HPE provides out of the box.

- You need deeper customizations than are possible to express using the GUI.

Instructions for installing via the command-line are here:

- Chapter 10, *Installing Mid-scale and Entry-scale KVM*
- Chapter 11, *Installation for Helion Entry Scale ESX (with OVSvApp, KVM with VSA Model)*

---

# Chapter 7. Installing via the GUI

HPE Helion OpenStack 5.0 comes with a UI installer for installing your cloud the first time. Rather than searching through and editing a number of YAML files to input your specific configuration data, finding the correct location, and maintaining the format of the files, the UI presents fields in which you may enter your information. In addition, this latest version allows you to populate server information via a .csv file import.

Using the installer, the series Cobbler and Ansible scripts that initiate the installation of HPE Helion OpenStack are run for you behind the scenes.

The UI app populates the configuration files that the configuration processor uses as input to its processes. Therefore you can inspect the files whenever you like, or before reconfiguring your cloud.

Note, the GUI installer will periodically save the configuration files during the process and when you click the Deploy button at the end of the process. This also means that if you were to try to change anything in the files themselves mid process, the values would get overwritten by values you have entered in the GUI.

These YAML configuration files are stored in a local git repository as well, as explained in Chapter 3, *Using Git for Configuration Management*.

When installing via the GUI, you begin with a model (or a cloud configuration template) and edit the contents to reflect your environment.

However, the full range of customizations is not possible via the GUI as some configuration items are not exposed. Below is the list of what can and cannot be changed:

**Changes to the following items may be made:**

- servers (including Linux for HPE Helion installation configuration)
- networks
- disk models
- interface models
- NIC mappings
- NTP servers
- Name servers
- tags in network groups

**Changes to the following items cannot be made:**

- server groups
- server roles
- network groups
- firewall rules
- cloudConfig.yml (i.e. DNS, SMTP, firewall settings)
- control planes

# Before you begin

Before you run the GUI installer to install your cloud, there are a number of things you need to do:

1. Prepare your servers
2. Gather your information
  - Server names
  - IP addresses
  - Server Roles
  - PXE MAC addresses
  - PXE IP addresses
  - PXE interfaces
  - IPMI/iLO IP address, username, password
3. Choose your Chapter 10, *Example Configurations* template. No action other than an understanding of your needs is necessary at this point. You will indicate which model you wish to deploy in a GUI screen. Note, however, that the only VSA options for the *Entry-scale KVM with VSA* model is to have 0 or 3 VSA nodes. Note also that you cannot run the lifecycle manager on a dedicated node using the GUI installer; it can only run in the control plane using the models exposed in the GUI.
4. Before you use the GUI to install your cloud, you may install the operating system, Linux for HPE Helion, on your nodes (servers) if you prefer. Otherwise, the installer will install it for you.

If you are installing the operating system on all your nodes yourself, note that you must do so using the Linux for HPE Helion image that is included in the HPE Helion OpenStack 5.0 package. You may find the instructions on the following pages helpful even when using your own tools to install the operating systems:

- Chapter 10, *Installing Mid-scale and Entry-scale KVM*
- Chapter 11, *Installation for Helion Entry Scale ESX (with OVSvApp, KVM with VSA Model)*

## Note

When following the guidance for operating system installation on those pages, stop before the sections for **Running the configuration processor**. The GUI installer will run those steps, and in the case where you have the GUI also install the operating systems, it will run all the previous steps (playbooks) as well.

# Creating a CSV file for import

You may create a CSV file for import into the GUI, indicating your server information rather than entering it directly into the GUI. The following table shows the fields you need and those that are optional.

Field	Optional	OS Install Only?	Aliases
Server ID	NO	NO	server_id, id

IP Address	NO	NO	ip, ip_address, address
MAC Address	NO	YES	mac, mac_address, mac_addr
IPMI IP Address	NO	YES	ipmi_ip, ilo_ip
IPMI User	NO	YES	ipmi_user, ilo_user, user
IPMI Password	NO	YES	Ipmi_password, ilo_password, password
Server Role	NO	NO	server_role, role
Server Group	YES	NO	server_group, group
NIC Mapping	NO	NO	server_nic_map, nic_map, nic_mapping

Here “OS Install Only?” indicates that this field is ONLY needed if you have indicated that you want the installer to install HPE Linux on the servers.

The aliases are all the valid names that can be used in the CSV file for the column header for a given field. Note field names are not case sensitive and that you can use either ‘ ‘ (space) or ‘-‘ (hyphen) in place of underscore for a field name.

So, an example header line in the CSV file could be:

```
Server ID, ip address, mac-address, IPMI_ip, password, user, server-role, server-G
```

The order of the fields does NOT have to match the order in the table above.

## Run the GUI to install your cloud

The GUI is found on the lifecycle manager node at: [http://<lifecycle manager\\_IP>:79/dayzero](http://<lifecycle manager_IP>:79/dayzero)

To create a secure tunnel/port forwarding into Apache from the lifecycle manager to be able to use the GUI in a browser, SSH tunnel into the lifecycle manager with the username/password set up when installing the lifecycle manager. The username and password should be what was set in "Set up the lifecycle manager" in the installation instructions. For example:

```
ssh -v -N -L 8080:<lifecycle manager_IP>:79 <Username>@<lifecycle manager_IP>
```

Then on your local machine, the one you are tunneling from, point your web browser to: <http://localhost:8080/dayzero>

As you proceed through the GUI, you can either enter your server names and IP addresses manually or import them from the CSV file using the CSV fields described above.

In the Assign network groups: External VM network fields, **do not** add entries for CIDR, Gateway, or IP Range, only VLAN ID. Neutron will assign these.

When the installation is complete, the Ansible log will be displayed on the final page.

## Use the installer securely

The GUI installer is a web-based application that runs on the lifecycle manager node and is implemented in node.js. Currently this application runs behind Apache 2.0 where requests are proxied to the node.js Express Web Server. The node.js Express Server runs on port 3000 and Apache runs on port 79.

You should access the installer through Apache and not directly from the Express server. An Example URL looks like this: `http://<lifecycle manager-host-ip>:79/dayzero` When you finish installing, the Installer will be disabled via Ansible. It is also disabled once deployment is completed following the CLI instructions..

The GUI is used only once: during the initial installation of your cloud.

For VSA deployments, once the GUI completes its installation, you must follow the steps outlined in the section called “Configuring for VSA Block Storage Backend”.

After deployment, you may also continue to Chapter 22, *Cloud Verification* and Chapter 28, *Other Common Post-Installation Tasks*.

To understand a cloud configuration more thoroughly and to learn how to make any changes later, visit these documentation pages:

- the section called “Ring Specifications in the Input Model”
- Chapter 3, *Using Git for Configuration Management*

### Note

Because the GUI installer is disabled after deployment is complete, whether you installed your cloud via the GUI or via the command line, if you want to re-enable the GUI later but are not re-installing from scratch, run these commands:

```
sudo a2ensite dayzero-apache.conf  
sudo systemctl start dayzero.service
```

## Final Steps

- The public Horizon Dashboard connection is over TLS, thus please use `HTTPS://` to navigate to Horizon with the IP address the installer presents once install is complete.
- Visit the for Ceph post-install steps.

---

# Chapter 8. DNS Service Installation Overview

The HPE Helion OpenStack DNS Service supports several different backends for domain name service. The choice of backend must be included in the deployment model before the HPE Helion OpenStack install is completed.

The backends that are available within the DNS Service are separated into two categories, self-contained and external.

**Table 8.1.**

Category	Backend	Description	Recommended For
Self-contained	PowerDNS 3.4.1, BIND 9.9.5	All components necessary will be installed and configured as part of the HPE Helion OpenStack install.	POCs and customers who wish to keep cloud and traditional DNS separated.
External	InfoBlox	The authoritative DNS server itself is external to HPE Helion OpenStack. Management and configuration is out of scope for the lifecycle manager but remains the responsibility of the customer.	Customers who wish to integrate with their existing DNS infrastructure.

## Getting Started

## Install the DNS Service with PowerDNS

### Install DNS Service with PowerDNS

HPE Helion OpenStack DNS Service defaults to the PowerDNS Backend if another backend is not configured for domain name service. PowerDNS will be deployed to one or more control planes clusters. The following configuration example denotes where the PowerDNS service is installed.

### Configure the Backend

To configure the backend for PowerDNS, follow these steps.

1. Ensure the DNS Service components, and the PowerDNS component have been placed on a cluster. PowerDNS may be placed on a separate cluster to the other DNS Service components.

```
control-planes:  
    - name: control-plane-1  
      region-name: region1  
  
    clusters:  
        - name: cluster1
```

```
service-components:
- lifecycle-manager
- mysql
- ip-cluster
- apache2
- ...
- designate-api
- designate-central
- designate-pool-manager
- designate-zone-manager
- designate-mdns
- designate-client
- powerdns
```

2. Edit the `~/helion/my_cloud/definitions/data/network_groups.yml` file to include the powerdns-ext.

```
- name: EXTERNAL-API
hostname-suffix: extapi
component-endpoints:
- powerdns-ext
load-balancers:
- provider: ip-cluster
```

3. Edit the `~/helion/my_cloud/definitions/data/firewall_rules.yml` to allow UDP/TCP access.

```
- name: DNSUdp
# network-groups is a list of all the network group names that the rules apply to
network-groups:
- EXTERNAL-API
rules:
- type: allow
# range of remote addresses in CIDR format that this rule applies to
remote-ip-prefix: 0.0.0.0/0
port-range-min: 53
port-range-max: 53
protocol: udp

- name: DNStcp
# network-groups is a list of all the network group names that the rules apply to
network-groups:
- EXTERNAL-API
rules:
- type: allow
# range of remote addresses in CIDR format that this rule applies to
remote-ip-prefix: 0.0.0.0/0
port-range-min: 53
port-range-max: 53
protocol: tcp
```

Please see for post installation DNS Service configuration.

# Install the DNS Service with BIND

## Install DNS Service with BIND

HPE Helion OpenStack DNS Service and BIND can be installed together instead of the default **PowerDNS** backend. BIND will be deployed to one or more control planes clusters. The following configuration example denotes where the BIND service is installed.

## Configure the Backend

Ensure the DNS Service components, and the BIND component have been placed on a cluster. BIND may be placed on a separate cluster to the other DNS Service components. Additionally, ensure the default **powerdns** component has been removed.

```
control-planes:
  - name: control-plane-1
    region-name: region1

    clusters:
      - name: cluster1
        service-components:
          - lifecycle-manager
          - mysql
          - ip-cluster
          - apache2
          - ...
          - designate-api
          - designate-central
          - designate-pool-manager
          - designate-zone-manager
          - designate-mdns
          - designate-client
          - bind
```

### Update Input Model

Once the backend is configured, you will need to add `bind-ext` to the `network_groups.yml` file.

1. Edit `helion/my_cloud/definition/data/network_groups.yml`, add `bind-ext` to component-endpoints.

```
name: EXTERNAL-API
hostname-suffix: extapi
component-endpoints:
  - bind-ext
```

2. Save file.

# Install the DNS Service with InfoBlox

HPE Helion OpenStack DNS Service can be installed with the **InfoBlox** backend instead of the default **PowerDNS** backend. In order to use InfoBlox as the backend, all prerequisites must be satisfied and the configuration of InfoBlox complete.

## Note

No DNS server will be deployed onto the HPE Helion OpenStack nodes. Instead, zones will be hosted on the **InfoBlox** servers.

## Important

The InfoBlox integration was enabled and validated in HPE Helion OpenStack 3.x. To use InfoBlox with version [4.x / 5.x] we recommend that you review the InfoBlox Deployment guide which can be found here: <https://www.infoblox.com/wp-content/uploads/infoblox-deployment-guide-infoblox-openstack-driver.pdf> and address any questions with your InfoBlox support contact.

## Prerequisites

1. An existing InfoBlox system deployed within your organization.
2. IPs and other credentials required to access the InfoBlox WS API.
3. Network connectivity to and from the cluster containing designate and the InfoBlox WS API.

This information is available from your InfoBlox system administrator.

## Configure the Backend

If not already present, DNS Service components must be placed on a cluster. Additionally, ensure the default **powerdns** component has been removed, and **bind** has not been added. Replace the **designate-mdns** component with the **designate-mdns-external** component.

```
control-planes:  
  - name: control-plane-1  
    region-name: region1  
    - lifecycle-manager-target  
  
clusters:  
  - name: cluster1  
    service-components:  
      - lifecycle-manager  
      - mysql  
      - ip-cluster  
      - apache2  
      - ...  
      - designate-api  
      - designate-central  
      - designate-pool-manager  
      - designate-zone-manager  
      - designate-mdns-external  
      - designate-client
```

You will need to provide DNS Service information details on your InfoBlox deployment. Open the designate pool-manager configuration template:

```
$ cd ~/helion/my_cloud
```

```
$ nano config/designate/pool-manager.conf.j2
```

In the config/designate/pool-manager.conf.j2, find the following code block:

```
nameservers:  
  - host: <infoblox-server-ip>  
    port: <infoblox-port-number>  
  ...  
  also_notifies:  
  - host: <infoblox-server-ip>  
    port: <infoblox-port-number>
```

Make the following changes:

1. Uncomment the block for infoblox and also\_notifies. In Jinja2, this means replacing the {# and #} markers with {{ and }}.
2. Fill in the API URL, username, password, and all remaining fields.
3. Save the file.

Once complete, the block should look like this:

```
- type: infoblox  
  description: infoblox Cluster  
  
  masters:  
  {% if DES_PMG.consumes_DES_MDN_EXT is defined %}  
  {% for mdn_member in DES_PMG.consumes_DES_MDN_EXT.members.private %}  
    - host: {{ mdn_member.ip_address }}  
      port: {{ mdn_member.port }}  
  {% endfor %}  
  {% endif %}  
  
  options:  
    wapi_url: https://<infoblox-server-ip>/wapi/v2.2.2/  
    username: hos-designate  
    password: MySecretPassword  
    ns_group: designate  
    sslverify: False  
    dns_view: default  
    network_view: default
```

You will need to inspect and commit the changes before proceeding with the deployment:

```
$diff --git a/ansible/roles/designate-pool-manager/templates/pools.yaml.j2 b/h  
index 291c6c9..b7fb39c 100644  
--- a/ansible/roles/designate-pool-manager/templates/pools.yaml.j2  
+++ b/ansible/roles/designate-pool-manager/templates/pools.yaml.j2  
@@ -28,6 +28,8 @@  
    priority: 2  
  
    nameservers:  
+    - host: <infoblox-server-ip>  
+      port: <infoblox-port-number>  
  {% if DES_PMG.consumes_FND_PDN is defined %}
```

```
{% for pdn_member in DES_PMG.consumes_FND_PDN.members.private %}
    - host: {{ pdn_member.ip_address }}
@@ -40,7 +42,9 @@
    port: {{ bnd_member.port }}
{% endfor %}
{% endif %}
-# also_notifies:
+ also_notifies:
+   - host: <infoblox-server-ip>
+     port: <infoblox-port-number>
     targets:
     {% if DES_PMG.consumes_FND_PDN is defined %}
         - type: powerdns
@@ -89,27 +93,27 @@
     {% endfor %}
     {% endif %}

-#
-#   - type: infoblox
-#     description: infoblox Cluster
-#
-#   masters:
+   - type: infoblox
+     description: infoblox Cluster
+
+   masters:
     {% if DES_PMG.consumes_DES_MDN_EXT is defined %}
     {% for mdn_member in DES_PMG.consumes_DES_MDN_EXT.members.private %}
-#         - host: {{ mdn_member.ip_address }}
-#           port: {{ mdn_member.port }}
+         - host: {{ mdn_member.ip_address }}
+           port: {{ mdn_member.port }}
     {% endfor %}
     {% endif %}
-#
-#   options:
-#     wapi_url: https://127.0.0.1/wapi/v2.2.2/
-#     username: admin
-#     password: infoblox
-#     ns_group: designate
-#     sslverify: False
-#     dns_view: default
-#     network_view: default
-#
+   options:
+     wapi_url: https://127.0.0.1/wapi/v2.2.2/
+     username: admin
+     password: infoblox
+     ns_group: designate
+     sslverify: False
+     dns_view: default
```

## SSL and CA Certificate

To enable SSL Verify, edit the following file:

```
$ cd ~/helion/my_cloud  
$ nano config/designate/pools.yaml.j2
```

In the infoblox section, set sslverify to true:

```
sslverify: True
```

To generate a CA certificate for InfoBlox, follow these steps:

1. Login to the InfoBlox User Interface
2. Generate a self signed certificate by selecting: System->Certificate->HTTP Cert->Generate Self Signed Certificate
3. Provide the hostname as the InfoBlox server-ip
4. Reload the InfoBlox User Interface. The certificate will be loaded and can be verified through the browser.
5. If you want to download the certificate, select: System->Certificate->HTTP Cert->Download the Certificate
6. Copy Certificate file to ~/helion/my\_cloud/config/tls/cacerts/  
Commit changes.

```
$ git commit -a -m "Configure Designate InfoBlox Backend"
```

## Configure DNS Domain and NS Records

To configure the default DNS domain and Name Server records for the default pool, follow these steps.

1. Ensure that `designate_config.yml` file is present in the `~/helion/my_cloud/definition/data/designate` folder. If the file or folder is not present, create the folder and copy `designate_config.yml` file from one of the example input models (for example, `~/helion/examples/entry-scale-esx-kvm-vsa/data/designate/designate_config.yml`).
2. Modify the **dns\_domain** and/or **ns\_records** entries in the `designate_config.yml` file.

```
data:  
  dns_domain: example.org.  
  ns_records:  
    hostname: ns1.example.org.  
    priority: 1  
    hostname: ns2.example.org.  
    priority: 2
```

3. Edit your input model's `control_plane.yml` file to include **DESIGNATE-CONFIG-CP1** in **configuration-data** section.

```
control-planes:  
  - name: control-plane-1
```

```
region-name: region1
lifecycle-manager-target
configuration-data:
  - DESIGNATE-CONFIG-CP1
  - NEUTRON-CONFIG-CP1
```

4. Continue your cloud deployment by reviewing and committing your changes.

```
$ git add ~/helion/my_cloud/definition/data/designate/designate_config.yml
$ git commit -m "Adding DNS Domain and NS Records"
```

### Note

In an , you will have 3 ns\_records since the DNS service runs on all three control planes. In a or the above example would be correct since there are only two controller nodes.

---

# Chapter 9. Magnum Overview

The HPE Helion OpenStack Magnum Service provides container orchestration engines such as Docker Swarm, Kubernetes, and Apache Mesos available as first class resources. HPE Helion OpenStack Magnum uses Heat to orchestrate an OS image which contains Docker and Kubernetes and runs that image in either virtual machines or bare metal in a cluster configuration.

## Magnum Architecture

As an OpenStack API service, Magnum provides Container as a Service (CaaS) functionality. Magnum is capable of working with container orchestration engines (COE) such as Kubernetes, Docker Swarm, and Apache Mesos.

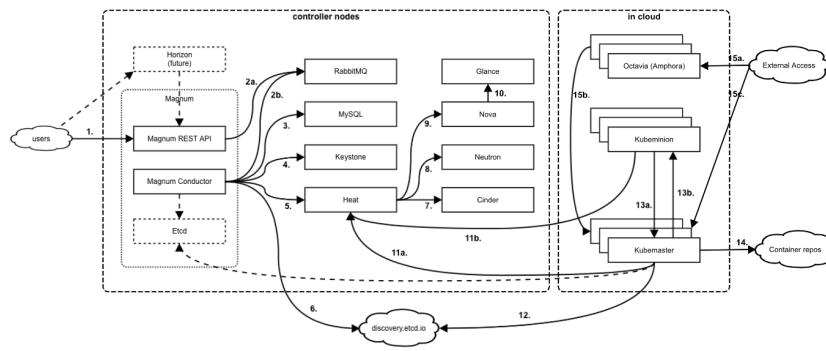
### Components

- **Magnum API:** RESTful API for cluster and cluster template operations.
- **Magnum Conductor:** Performs operations on clusters requested by Magnum API in an asynchronous manner.
- **Magnum CLI:** Command-line interface to the Magnum API.
- **Etcfd (planned, currently using public service):** Remote key/value storage for distributed cluster bootstrap and discovery.
- **Kubemaster (in case of Kubernetes COE):** One or more VM(s) or baremetal server(s), representing a control plane for Kubernetes cluster.
- **Kubeminion (in case of Kubernetes COE):** One or more VM(s) or baremetal server(s), representing a workload node for Kubernetes cluster.
- **Octavia VM aka Amphora (in case of Kubernetes COE with enabled load balancer functionality):** One or more VM(s), created by HOS LBaaS v2, performing request load balancing for Kubemasters.

### Data

Data Name	Confiden-tiality	Integri-ty	Availabili-ty	Backup?	Description
Session Tokens	Confidential	High	Medium	No	Session tokens not stored.
System Request	Confidential	High	Medium	No	Data in motion or in MQ not stored.
MySQL Data-base "Magnum"	Confidential	High	High	Yes	Contains user preferences. Backed up to Swift daily.
etcd data	Confidential	High	Low	No	Kubemaster IPs and cluster info. Only used during cluster bootstrap.

### Service Architecture Diagram



## Interfaces

**Table 9.1.**

Interface	Net-work Name	Net-work Protocol	Re-questor	Re-quest	Re-quest Cred-entials	Re-quest Auth-ori-zation	Listen-er	Re-sponse	Re-sponse Cred-entials	De-scrip-tion of Opera-tion
1	Exter-nal-API	HTTPS	User	Manage clusters	Key-stone token	Manage objects which belong to current project	Mag-num API	Opera-tion status with or without data	TLS Certifi-cate	CRUD opera-tions on cluster tem-plates and clusters
2a	Inter-nal-API	AMQP over HTTPS	Mag-num API	En-queue mes-sages	Rabbit-MQ user-name, pass-word	Rabbit-MQ queue read/ write opera-tions	Rabbit-MQ	Opera-tion status	TLS Certifi-cate	Notifi-cations issued when cluster CRUD opera-tions re-quested
2b	Inter-nal-API	AMQP over HTTPS	Mag-num Con-ductor	Read queued mes-sages	Rabbit-MQ user-name, pass-word	Rabbit-MQ queue read/ write opera-tions	Rabbit-MQ	Opera-tion status	TLS Certifi-cate	Notifi-cations issued when cluster CRUD opera-tions re-quested
3	Inter-nal-API	MySQL over HTTPS	Mag-num Con-ductor	Persist data in MySQL	MySQL user-name,	Mag-num data-base	MySQL	Opera-tion status with	TLS Certifi-cate	Persist clus-ter/clus-ter tem-

Interface	Net-work Name	Net-work Protocol	Re-questor	Re-quest	Re-quest Credentials	Re-quest Authorization	Listener	Re-sponse	Re-sponse Credentials	Description of Operation
					pass-word			or without data		plate data, read persisted data
4	Internal-API	HTTPS	Magnum Conductor	Create per-cluster user in dedicated domain, no role assignments initially	Trustee domain admin user-name, pass-word	Manage users in dedicated magnum domain	Key-stone	Operation status with or without data	TLS Certificate	Magnum generates user record in a dedicated Key-stone domain for each cluster
5	Internal-API	HTTPS	Magnum Conductor	Create per-cluster stack	Key-stone token	Limited to scope of authorized user	Heat	Operation status with or without data	TLS Certificate	Magnum creates Heat stack for each cluster
6	External Network	HTTPS	Magnum Conductor	Bootstrap a cluster in public discovery https://discovery.etcd.io	Unguessable URL over HTTPS. URL is only available to software processes needing it.	Read and update	Public discovery service	Cluster discovery URL	TLS Certificate	Create key/ value registry of specified size in public storage. This is used to stand up a cluster of kubernetes master nodes (refer to interface call #12)

Interface	Net-work Name	Net-work Protocol	Re-questor	Re-quest	Re-quest Credentials	Re-quest Authorization	Listener	Re-sponse	Re-sponse Credentials	Description of Operation
7	Internal-API	HTTPS	Heat Engine	Create Cinder volumes	Key-stone token	Limited to scope of authorized user	Cinder API	Operation status with or without data	TLS Certificate	Heat creates Cinder volumes as part of stack
8	Internal-API	HTTPS	Heat Engine	Create networks, routers, load balancers	Key-stone token	Limited to scope of authorized user	Neutron API	Operation status with or without data	TLS Certificate	Heat creates networks, routers, load balancers as part of the stack
9	Internal-API	HTTPS	Heat Engine	Create Nova VNs, attach volumes	Key-stone token	Limited to scope of authorized user	Nova API	Operation status with or without data	TLS Certificate	Heat creates Nova VMs as part of the stack
10	Internal-API	HTTPS	Nova	Read pre-configured Glance image	Key-stone token	Limited to scope of authorized user	Glance API	Operation status with or without data	TLS Certificate	Nova is using pre-configured image in Glance to bootstrap VMs
11a	External-API	HTTPS	Cluster member (VM or Ironic node)	Heat notification	Key-stone token	Limited to scope of authorized user	Heat API	Operation status with or without data	TLS Certificate	Heat is using OS::Heat::WaitCondition resource. VM is expected to call Heat

Interface	Net-work Name	Net-work Protocol	Re-questor	Re-quest	Re-quest Cred-entials	Re-quest Autho-ri-zation	Listener	Re-sponse	Re-sponse Cred-entials	De-scrip-tion of Opera-tion
										notifi-cation URL upon comple-tion of certain boot-strap opera-tion.
11b	Exter-nal-API	HTTPS	Cluster member (VM or ironic node)	Heat notifi-cation	Key-stone token	Limited to scope of au-thorized user	Heat API	Opera-tion sta-tus with or with-out data	TLS Certifi-cate	Heat is using OS::Heat::Wait-Condition re-source. VM is expect-ed to call Heat notifi-cation URL upon comple-tion of certain boot-strap opera-tion.
12	Exter-nal-API	HTTPS	Cluster member (VM or Ironic node)	Update cluster member state in a pub-lic reg-istry at https://disco-ver.y.etcd.oware proce-ses need-ing it.	Unguess-able URL over HTTPS only avail-able to soft-ware process-es need-ing it.	Read and up-date	Public disco-ver.y ser-vicve	Opera-tion sta-tus	TLS Certifi-cate	Update key/ value pair in a reg-istry created by inter-face call #6

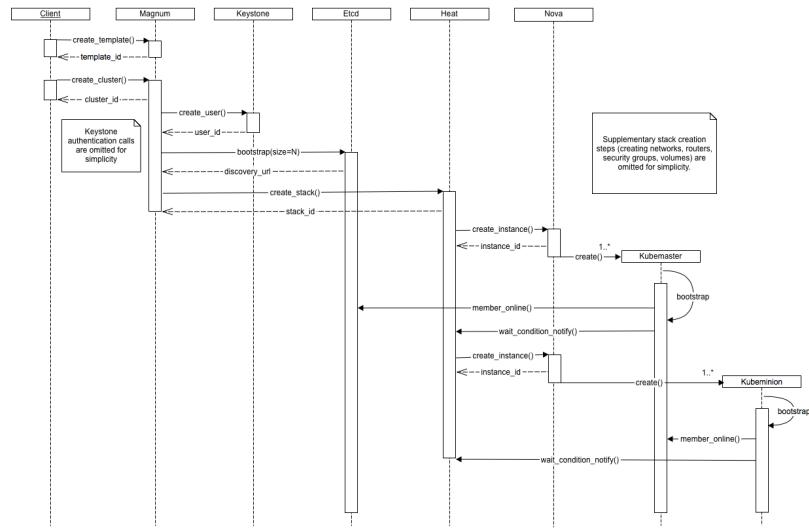
Interface	Net-work Name	Net-work Protocol	Re-questor	Re-quest	Re-quest Credentials	Re-quest Authorization	Listener	Re-sponse	Re-sponse Credentials	De-scrip-tion of Opera-tion
13a	A VxLAN encapsulated private network on the Guest network	HTTPS	Cluster member (VM or Ironic node)	Various communications inside Kubernetes cluster	Tenant specific	Tenant specific	Cluster member (VM or Ironic node)	Tenant specific	TLS Certificate	Various calls performed to build Kubernetes clusters, deploy applications and put workload
13b	A VxLAN encapsulated private network on the Guest network	HTTPS	Cluster member (VM or Ironic node)	Various communications inside Kubernetes cluster	Tenant specific	Tenant specific	Cluster member (VM or Ironic node)	Tenant specific	TLS Certificate	Various calls performed to build Kubernetes clusters, deploy applications and put workload
14	Guest/External	HTTPS	Cluster member (VM or Ironic node)	Download container images	None	None	External	Container image data	TLS Certificate	Kubernetes is making calls to external repositories to download pre-packed container images
15a	External/EXT_VM (Floating IP)	HTTPS	Tenant specific	Tenant specific	Tenant specific	Tenant specific	Octavia load balancer	Tenant specific	Tenant specific	External workload handled

Interface	Net-work Name	Net-work Protocol	Re-questor	Re-quest	Re-quest Credentials	Re-quest Authorization	Listener	Re-sponse	Re-sponse Credentials	Description of Operation
										by container applications
15b	Guest	HTTPS	Tenant specific	Tenant specific	Tenant specific	Tenant specific	Cluster member (VM or Ironic node)	Tenant specific	Tenant specific	External workload handled by container applications
15c	External/EXT_VM (Floating IP)	HTTPS	Tenant specific	Tenant specific	Tenant specific	Tenant specific	Cluster member (VM or Ironic node)	Tenant specific	Tenant specific	External workload handled by container applications

## Dependencies

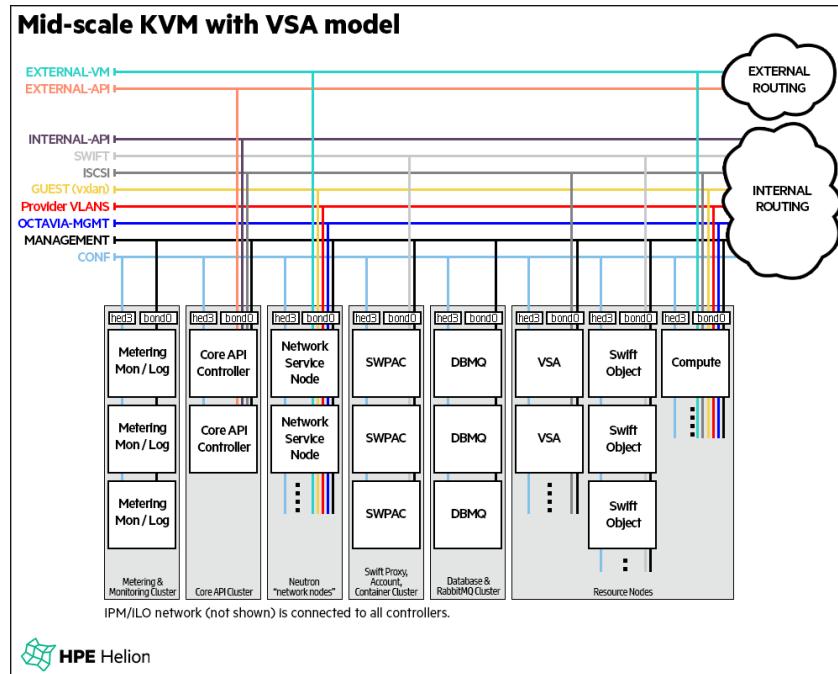
- Keystone
- RabbitMQ
- MySQL
- Heat
- Glance
- Nova
- Cinder
- Neutron
- Barbican
- Swift

## Sequence Diagram



### Implementation

Magnum API and Magnum Conductor are run on the HPE Helion OpenStack controllers (or core nodes in case of mid-scale deployments).



### Security Groups

Source CIDR/Security Group	Port/Range	Protocol	Notes
Any IP	22	SSH	Tenant Admin access
Any IP/Kubernetes Security Group	2379-2380	HTTPS	Etcd Traffic
Any IP/Kubernetes Security Group	6443	HTTPS	kube-apiserver

Source CIDR/Security Group	Port/Range	Protocol	Notes
Any IP/Kubernetes Security Group	7080	HTTPS	kube-apiserver
Any IP/Kubernetes Security Group	8080	HTTPS	kube-apiserver
Any IP/Kubernetes Security Group	30000-32767	HTTPS	kube-apiserver
Any IP/Kubernetes Security Group	any	tenant app specific	tenant app specific

## Network Ports

Port/Range	Protocol	Notes
22	SSH	Admin Access
9511	HTTPS	Magnum API Access
2379-2380	HTTPS	Etcdb (planned)

## Summary of Controls

Summary of controls spanning multiple components and interfaces:

- **Audit:** Magnum performs logging. Logs are collected by the centralized logging service.
- **Authentication:** Authentication via Keystone tokens at APIs. Password authentication to MQ and DB using specific users with randomly-generated passwords.
- **Authorization:** OpenStack provides admin and non-admin roles that are indicated in session tokens. Processes run at minimum privilege. Processes run as unique user/group definitions (magnum/magnum). Appropriate filesystem controls prevent other processes from accessing service's files. Magnum config file is mode 600. Logs written using group adm, user magnum, mode 640. IPtables ensure that no unneeded ports are open. Security Groups provide authorization controls between in-cloud components.
- **Availability:** Redundant hosts, clustered DB, and fail-over provide high availability.
- **Confidentiality:** Network connections over TLS. Network separation via VLANs. Data and config files protected via filesystem controls. Unencrypted local traffic is bound to localhost. Separation of customer traffic on the TUL network via Open Flow (VxLANs).
- **Integrity:** Network connections over TLS. Network separation via VLANs. DB API integrity protected by SQL Alchemy. Data and config files are protected by filesystem controls. Unencrypted traffic is bound to localhost.

# Install the Magnum Service

Installing the Magnum Service can be performed as part of a new HPE Helion OpenStack 5.0 environment or can be added to an existing HPE Helion OpenStack 5.0 environment. Both installations require container management services, running in Magnum cluster VMs with access to specific Openstack API endpoints. The following TCP ports need to be open in your firewall to allow access from VMs to external (public) HPE Helion OpenStack endpoints.

TCP Port	Service
5000	Identity
8004	Heat
9511	Magnum

Additionally, Magnum is dependent on the following OpenStack services.

- Keystone
- Heat
- Nova KVM
- Neutron
- Glance
- Cinder
- Swift
- Barbican
- LBaaS v2 (Octavia) - *optional*

## Installing Magnum as part of new HPE Helion OpenStack 5.0 environment

Magnum components are already included in example HPE Helion OpenStack models based on Nova KVM, such as **entry-scale-kvm-vsa**, **entry-scale-kvm-vsa-mml** and **mid-scale**. These models contain the Magnum dependencies (see above). You can follow generic installation instruction for Mid-Scale and Entry-Scale KM model by using this guide:

### Note

1. If you modify the cloud model to utilize a dedicated lifecycle manager, add `magnum-client` item to the list of service components for the lifecycle manager cluster.
2. Magnum needs a properly configured external endpoint. While preparing the cloud model, ensure that `external-name` setting in `data/network_groups.yml` is set to valid host-name, which can be resolved on DNS server, and a valid TLS certificate is installed for your external endpoint. For non-production test installations, you can omit `external-name`. In test installations, the HPE Helion OpenStack installer will use an IP address as a public endpoint hostname, and automatically generate a new certificate, signed by the internal CA. Please refer to Chapter 25, *Configuring Transport Layer Security (TLS)* for more details.
3. To use LBaaS v2 (Octavia) for container management and container applications, follow the additional steps to configure LBaaS v2 in the guide .

## Adding Magnum to existing HPE Helion OpenStack 5.0 environment

Adding Magnum to an already deployed HPE Helion OpenStack 5.0 installation or during and upgrade can be achieved by performing the following steps.

1. Add items listed below to the list of service components in `/home/stack/helion/my_cloud/definition/data/control_plane.yml`. Add them to clusters which have `server-role` set to `CONTROLLER-ROLE` (entry-scale models) or `CORE_ROLE` (mid-scale model).
  - `magnum-api`
  - `magnum-conductor`
2. If your environment utilizes a dedicated lifecycle manager, add `magnum-client` to the list of service components for the lifecycle manager.
3. Commit your changes to the local git repository. Run the following playbooks as described in Chapter 3, *Using Git for Configuration Management* for your Installation.
  - `config-processor-run.yml`
  - `ready-deployment.yml`
  - `site.yml`
4. Ensure that your external endpoint is configured correctly. The current public endpoint configuration can be verified by running the following commands on the lifecycle manager.

```
$ source service.osrc
$ openstack endpoint list --interface=public --service=identity
+-----+-----+-----+
| ID | Region | Service Name | Service Type | Ena
+-----+-----+-----+
| d83e87ef474a497f8d8373357cce4aa3 | region1 | keystone | identity | Tru
+-----+-----+-----+
```

Ensure that the endpoint URL is using either an IP address, or a valid hostname, which can be resolved on the DNS server. If the URL is using an invalid hostname (for example, 'myhelion.test'), follow the steps described in Chapter 25, *Configuring Transport Layer Security (TLS)* to configure a valid external endpoint. You will need to update the `external-name` setting in the `data/network_group-s.yml` to a valid hostname, which can be resolved on DNS server, and provide a valid TLS certificate for the external endpoint. For non-production test installations, you can omit the `external-name`. The HPE Helion OpenStack installer will use an IP address as public endpoint hostname, and automatically generate a new certificate, signed by the internal CA. Please refer to Chapter 25, *Configuring Transport Layer Security (TLS)* for more information.

5. Ensure that LBaaS v2 (Octavia) is correctly configured. For more information, see Chapter 27, *Configuring Load Balancer as a Service*.

## Integrate Magnum with the DNS Service

Integration with DNSaaS may be needed if:

1. External endpoint is configured to use 'myhelion.test' as hostname and HPE Helion OpenStack frontend certificate is issued for this hostname.
2. Minions are registered using Nova VM names as hostnames Kubernetes API server. Most `kubectl` command won't work if the VM name (e.g. 'cl-mu3eevqizh-1-b3vifun6qtuh-kube-minion-ff4cqjgsuzhy') is not getting resolved at provided DNS server/

Follow these steps to integrate the Magnum Service with the DNS Service.

1. Allow connections from VMs to EXT-API

```
sudo modprobe 8021q
sudo ip link add link virbr5 name vlan108 type vlan id 108
sudo ip link set dev vlan108 up
sudo ip addr add 192.168.14.200/24 dev vlan108
sudo iptables -t nat -A POSTROUTING -o vlan108 -j MASQUERADE
```

2. Workaround for ????

```
$ grep enable_host_header hos/ansible/roles/designate-api/templates/api.conf.j2
enable_host_header = False
```

3. Run the designate reconfigure playbook.

```
$ cd ~/scratch/ansible/next/hos/ansible/
$ ansible-playbook -i hosts/verb_hosts designate-reconfigure.yml
```

4. Set up Designate to resolve myhelion.test correctly.

```
$ openstack zone create --email hostmaster@myhelion.test myhelion.test.
# wait for status to become active
$ EXTERNAL_VIP=$(grep HZN-WEB-extapi /etc/hosts | awk '{ print $1 }')
$ openstack recordset create --records $EXTERNAL_VIP --type A myhelion.test. myhelion.test.
# wait for status to become active
$ LOCAL_MGMT_IP=$(grep `hostname` /etc/hosts | awk '{ print $1 }')
$ nslookup myhelion.test $LOCAL_MGMT_IP
Server:      192.168.14.2
Address:      192.168.14.2#53
Name:         myhelion.test
Address:      192.168.14.5
```

5. If you need to add/override a top level domain record:

```
$ openstack tld create --name net
$ openstack zone create --email hostmaster@proxy.houston.hpecorp.net proxy.houston.hpecorp.net.
$ openstack recordset create --records 16.85.88.10 --type A proxy.houston.hpecorp.net.
$ nslookup proxy.houston.hpecorp.net 192.168.14.2
Server:      192.168.14.2
Address:      192.168.14.2#53
Name:         proxy.houston.hpecorp.net
Address:      16.85.88.10
```

6. Enable propagation of dns\_assignment and dns\_name attributes to neutron ports, as per <https://docs.openstack.org/draft/networking-guide/config-dns-int.html>

```
# optionally add 'dns_domain = <some domain name>.' to [DEFAULT] section of hos-
stack@ksperf2-cp1-c1-m1-mgmt:~/helion$ cat <<-EOF >>hos/services/designate/api.y
```

```
provides-data:
  - to:
    - name: neutron-ml2-plugin
  data:
    - option: extension_drivers
      values:
        - dns
```

```
EOF
$ git commit -a -m "Enable DNS support for neutron ports"
$ cd hos/ansible
$ ansible-playbook -i hosts/localhost config-processor-run.yml
$ ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Enable DNSaaS registration of created VMs by editing the `~/helion/hos/ansible/roles/neutron-common/templates/neutron.conf.j2` file. You will need to add `external_dns_driver = designate` to the **[DEFAULT]** section and create a new **[designate]** section for the Designate specific configurations.

```
...
advertise_mtu = False
dns_domain = ksperf.
external_dns_driver = designate
{{ neutron_api_extensions_path|trim }}
{{ neutron_vlan_transparent|trim }}

# Add additional options here

[designate]
url = https://10.240.48.45:9001
admin_auth_url = https://10.240.48.45:35357/v3
admin_username = designate
admin_password = P8lZ9FdHuoW
admin_tenant_name = services
allow_reverse_dns_lookup = True
ipv4_ptr_zone_prefix_size = 24
ipv6_ptr_zone_prefix_size = 116
ca_cert = /etc/ssl/certs/ca-certificates.crt
```

8. Commit your changes.

```
$ git commit -a -m "Enable DNSaaS registration of Nova VMs"
[site f4755c0] Enable DNSaaS registration of Nova VMs
1 file changed, 11 insertions(+)
```

---

# Chapter 10. Installing Mid-scale and Entry-scale KVM

- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 
- 

## Important Notes

- If you are looking for information about when to use the GUI installer and when to use the CLI, see the Installation Overview.
- Review the Chapter 2, *Hardware and Software Support Matrix* that we have listed.
- Review the release notes to make yourself aware of any known issues and limitations.
- The installation process can occur in different phases. For example, you can install the control plane only and then add Compute nodes afterwards if you would like.
- If you run into issues during installation, we have put together a list of Chapter 18, *Troubleshooting the Installation* you can reference.
- Make sure all disks on the system(s) are wiped before you begin the install. (For Swift, refer to the section called “Swift Requirements for Device Group Drives”)
- There is no requirement to have a dedicated network for OS-install and system deployment, this can be shared with the management network. More information can be found in Chapter 10, *Example Configurations*.
- You may see the terms deployer and lifecycle manager used interchangeably. These are referring to the same nodes in your environment.
- When running the Ansible playbook in this installation guide, if a runbook fails you will see in the error response to use the `--limit` switch when retrying a playbook. This should be avoided. You can simply re-run any playbook without this switch.
- DVR is not supported with ESX compute.
- When you attach a Cinder volume to the VM running on the ESXi host, the volume will not get detected automatically. Make sure to set the image metadata `vmware_adaptertype=lsiLogicsas` for image before launching the instance. This will help to discover the volume change appropriately.

# Before You Start

We have put together a Chapter 2, *Pre-Installation Checklist* that should help with the recommended pre-installation tasks.

# Setting Up the Lifecycle Manager

## Installing the Lifecycle Manager

The lifecycle manager will contain the installation scripts and configuration files to deploy your cloud. You can set up the lifecycle manager on a dedicated node or you do so on your first controller node. The default choice is to use the first controller node as the lifecycle manager.

1. Sign in and download the product and signature files at the link below:
  - a. Software Entitlement Portal [<http://www.hpe.com/software/entitlements>]
2. You can verify the download was complete via the signature verification process outlined here [<https://h20392.www2.hpe.com/portal/swdepot/displayProductInfo.do?productNumber=HPLin-uxCodeSigning>].
3. Boot your lifecycle manager from the ISO contained in the download.
4. Enter "install" to start installation.

## Note

"install" is all lower case

5. Select the language. Note that only the English language selection is currently supported.
6. Select the location.
7. Select the keyboard layout.
8. Select the primary network interface, if prompted:
  - a. Assign IP address, subnet mask, and default gateway
9. Create new account:
  - a. Enter a username.
  - b. Enter a password.
  - c. Enter time zone.

Once the initial installation is finished, complete the lifecycle manager setup with these steps:

1. Ensure your lifecycle manager has a valid DNS nameserver specified in `/etc/resolv.conf`.
2. Set the environment variable `LC_ALL`:

```
export LC_ALL=C
```

## Note

This can be added to `~stack/.bashrc` or `/etc/bash.bashrc`.

At the end of this section you should have a node set up with Linux for HPE Helion on it.

### Configure and Run the Lifecycle Manager

#### Important

It is critical that you don't run any of the commands below as the `root` user or use `sudo`, unless it is stated explicitly in the steps. Run them as the user you just created (or `stack` if you left the default of "stack").

1. Log into your lifecycle manager as the user you created and mount the install media at `/media/cdrom`. It may be necessary to use `wget` or another file transfer method to transfer the install media to the lifecycle manager before completing this step. Here is the command to mount the media:

Example for HPE Helion OpenStack® 5.0:

```
sudo mount HelionOpenStack-5.0.iso /media/cdrom
```

2. Unpack the tarball that is in the `/media/cdrom/hos/` directory:

Example for HPE Helion OpenStack 5.0

```
tar xvf /media/cdrom/hos/hos-5.0.0-<timestamp>.tar
```

3. Run the `hos-init.bash` script which is included in the build:

Example for HPE Helion OpenStack® 5.0

```
~/hos-5.0.0/hos-init.bash
```

You will be prompted to enter an optional SSH passphrase when running `hos-init.bash`. This passphrase is used to protect the key used by Ansible when connecting to its client nodes. If you do not want to use a passphrase then just press **Enter** at the prompt.

For automated installation (e.g. CI) it is possible to disable SSH passphrase prompting by setting the `HOS_INIT_AUTO` environment variable before running `hos-init.bash`, like this:

```
export HOS_INIT_AUTO=y
```

If you have protected the SSH key with a passphrase then execute the following commands to avoid having to enter the passphrase on every attempt by Ansible to connect to its client nodes:

```
eval $(ssh-agent)  
ssh-add ~/.ssh/id_rsa
```

At the end of this section you should have a local directory structure, as described below:

<code>~/helion/</code>	Top level directory
<code>~/helion/examples/</code>	Directory contains the config input files of the
<code>~/helion/my_cloud/definition/</code>	Directory contains the config input files
<code>~/helion/my_cloud/config/</code>	Directory contains .j2 files which are symlinks to
<code>~/helion/hos/</code>	Directory contains files used by the installer

## Warning

It is important that you do not add any extra files in the `~/helion` directory on your lifecycle manager. This includes temporary save files. Doing so will cause errors during the installation.

## Important

Be aware that the files in the `~/helion/my_cloud/config/` directory are symbolic links to the `~/helion/hos/ansible/` directory. For example, `~/helion/my_cloud/config/keepalived/defaults.yml` is a symbolic link to `~/helion/hos/ansible/roles/keepalived/defaults/main.yml`.

```
ls -al ~/helion/my_cloud/config/keepalived/defaults.yml
lrwxrwxrwx 1 stack stack 55 May 24 20:38 /home/stack/helion/my_cloud/config/kee
```

If you are using a tool like `sed` to make edits to files in this directory, you might break the symbolic link and create a new copy of the file. To maintain the link, you will need to force `sed` to follow the link:

```
sed -i --follow-symlinks \
's$keepalived_vrrp_offset: 0$keepalived_vrrp_offset: 2$' \
~/helion/my_cloud/config/keepalived/defaults.yml
```

Alternatively, directly edit the target of the link `~/helion/hos/ansible/roles/keepalived/defaults/main.yml`.

For any troubleshooting information regarding these steps, see the section called “Issues during Lifecycle-manager Setup”.

# Configuring Your Environment

During the configuration phase of the installation you will be making modifications to the example configuration input files to match your cloud environment. You should use the Chapter 10, *Example Configurations* documentation for detailed information on how to do this. There is also a `README.md` file included in each of the example directories on the lifecycle manager that has useful information about the models.

In the steps below we show how to set up the directory structure with the example input files as well as use the optional encryption methods for your sensitive data.

1. Setup your configuration files, as follows:

- Copy the example configuration files into the required setup directory and edit them to contain the details of your environment.

For example, if you want to use the Helion Mid-scale KVM with VSA model, you can use this command to copy the files to your cloud definition directory:

```
cp -r ~/helion/examples/mid-scale-kvm-vsa/* ~/helion/my_cloud/definition/
```

If you want to use the Helion Entry-scale KVM with VSA model, you can use this command to copy the files to your cloud definition directory:

```
cp -r ~/helion/examples/entry-scale-kvm-vsa/* ~/helion/my_cloud/definition/
```

- b. Begin inputting your environment information into the configuration files in the ~/helion/my\_cloud/definition directory.

If you are using the Mid-scale or Entry-scale KVM with VSA model, see Chapter 11, *Modifying the Entry-scale KVM with VSA Model for Your Environment* for details.

If you are using the Entry-scale KVM with Ceph model, see for details.

2. [OPTIONAL] - You can use the `hosencrypt.py` script to encrypt your iLo passwords. This script uses OpenSSL.

- a. Change to the Ansible directory:

```
cd ~/helion/hos/ansible
```

- b. Put the encryption key into the following environment variable:

```
export HOS_USER_PASSWORD_ENCRYPT_KEY=<encryption key>
```

- c. Run the python script below and follow the instructions. Enter a password that you want to encrypt.

```
./hosencrypt.py
```

- d. Take the string generated and place it in the "ilo-password" field in your ~/helion/my\_cloud/definition/data/servers.yml file, remembering to enclose it in quotes.

- e. Repeat the above for each server.

## Note

Before you run any playbooks, remember that you need to export the encryption key in the following environment variable: `export HOS_USER_PASSWORD_ENCRYPT_KEY=<encryption key>`

3. Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*), as follows:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

## Important

This step needs to be repeated any time you make changes to your configuration files before you move onto the following steps. See Chapter 3, *Using Git for Configuration Management* for more information.

# Provisioning Your Baremetal Nodes

To provision the baremetal nodes in your cloud deployment you can either use the automated operating system installation process provided by HPE Helion OpenStack or you can use the 3rd party installation tooling of your choice. We will outline both methods below:

## Using 3rd Party Baremetal Installers

If you do not wish to use the automated operating system installation tooling included with 5.0 then the requirements that have to be met using the installation tooling of your choice are:

- The operating system must be installed via the HPE Linux for HPE Helion OpenStack ISO provided on the Software Entitlement Portal [<http://www.hpe.com/software/entitlements>].
- Each node must have SSH keys in place that allows the same user from the lifecycle manager node who will be doing the deployment to SSH to each node without a password.
- Passwordless sudo needs to be enabled for the user.
- There should be a LVM logical volume as /root on each node.
- If the LVM volume group name for the volume group holding the "root" LVM logical volume is hlm-vg then it will align with the disk input models in the examples.
- Ensure that openssh-server, python, python-apt, and rsync are installed.

If you chose this method for installing your baremetal hardware, skip forward to the Chapter 10, *Installing Mid-scale and Entry-scale KVM* step.

If you would like to use the automated operating system installation tools provided by 5.0 then complete all of the steps below.

### **Using the Automated Operating System Installation Provided by HPE Helion OpenStack**

#### **Part One: Deploy Cobbler**

This phase of the install process takes the baremetal information that was provided in servers.yml and installs the Cobbler provisioning tool and loads this information into Cobbler. This sets each node to netboot-enabled: true in Cobbler. Each node will be automatically marked as netboot-enabled: false when it completes its operating system install successfully. Even if the node tries to PXE boot subsequently, Cobbler will not serve it. This is deliberate so that you can't reimagine a live node by accident.

The cobbler-deploy.yml playbook prompts for a password - this is the password that will be encrypted and stored in Cobbler, which is associated with the user running the command on the lifecycle manager, that you will use to log in to the nodes via their consoles after install. The username is the same as the user set up in the initial dialogue when installing the lifecycle manager from the iso, and is the same user that is running the cobbler-deploy play.

1. Run the following playbook which confirms that there is iLo connectivity for each of your nodes so that they are accessible to be re-imaged in a later step:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost bm-power-status.yml
```

2. Run the following playbook to deploy Cobbler:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

#### **Part Two: Image the Nodes**

This phase of the install process goes through a number of distinct steps:

1. Powers down the nodes to be installed

2. Sets the nodes hardware boot order so that the first option is a network boot.
3. Powers on the nodes. (The nodes will then boot from the network and be installed using infrastructure set up in the previous phase)
4. Waits for the nodes to power themselves down (this indicates a success install). This can take some time.
5. Sets the boot order to hard disk and powers on the nodes.
6. Waits for the nodes to be ssh-able and verifies that they have the signature expected.

The reimage command is:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost bm-reimage.yml [-e nodelist=node1,node2,node3]
```

If a nodelist is not specified then the set of nodes in cobbler with netboot-enabled: True is selected. The playbook pauses at the start to give you a chance to review the set of nodes that it is targeting and to confirm that it's correct.

You can use the command below which will list all of your nodes with the netboot-enabled: True flag set:

```
sudo cobbler system find --netboot-enabled=1
```

For any troubleshooting information regarding these steps, see the section called “Issues while Provisioning your Baremetal Nodes”.

## Running the Configuration Processor

Once you have your configuration files setup you need to run the configuration processor to complete your configuration.

When you run the configuration processor you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the ansible group\_vars and host\_vars that it produces for subsequent deploy runs. You will need this password for subsequent ansible deploy and configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise just press **Enter** to bypass this.

Run the configuration processor with this command:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

For automated installation (e.g. CI) you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor

```
ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e reke
```

If you receive an error during this step then there is probably an issue with one or more of your configuration files. We recommend that you verify that all of the information in each of your configuration files is correct for your environment and then commit those changes to git using the instructions in the previous section before re-running the configuration processor again.

For any troubleshooting information regarding these steps, see the section called “Issues while Updating Configuration Files”.

# Configuring TLS

## Important

This section is optional, but recommended, for a HPE Helion OpenStack installation.

After you run the configuration processor the first time, the IP addresses for your environment will be generated and populated in the `~/helion/my_cloud/info/address_info.yml` file. It's at this point that you will want to take into consideration whether or not you want to configure TLS and setup a SSL certificate for your environment. Please read Chapter 25, *Configuring Transport Layer Security (TLS)* before proceeding for how to achieve this.

# Deploying the Cloud

1. Use the playbook below to create a deployment directory:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

2. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your partitions on your nodes are completely wiped before continuing with the installation. If you are using fresh machines this step may not be necessary.

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts wipe_disks.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts wipe_disks.yml --ask-vault-pass
```

3. Run the `site.yml` playbook below:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts site.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts site.yml --ask-vault-pass
```

## Note

The step above runs `osconfig` to configure the cloud and `hlm-deploy` to deploy the cloud. Therefore, this step may run for a while, perhaps 45 minutes or more, depending on the number of nodes in your environment.

4. Verify that the network is working correctly. Ping each IP in the `/etc/hosts` file from one of the controller nodes.

For any troubleshooting information regarding these steps, see the section called “Issues while Deploying the Cloud”.

## Configuring a Block Storage Backend (Optional)

HPE Helion OpenStack supports VSA, 3PAR, and Ceph as block storage backend options. You can utilize one or more of these as setting up multiple block storage backends and multiple volume types is supported.

Regardless of whether you have a single or multiple block storage backends defined in your `cinder.conf.j2` file then you can create one or more volume types using the specific attributes associated with the backend. You can find details on how to do that for each of the supported backend types here:

- the section called “Configuring for VSA Block Storage Backend”
- the section called “Ceph Overview”
- the section called “Configuring for 3PAR Block Storage Backend”

## Post-Installation Verification and Administration

We recommend verifying the installation using the instructions in Chapter 22, *Cloud Verification*.

There are also a list of other common post-installation administrative tasks listed in the Chapter 28, *Other Common Post-Installation Tasks* list.

---

# Chapter 11. Installation for Helion Entry Scale ESX (with OVSvApp, KVM with VSA Model)

This document describes the procedure for the deployment of an ESX cloud using input model and preparing and deploying ESX Computes and OVSvAPPs.

It contains the following topics:

- 
- 
- 
- 

## Important

Before you start your ESX cloud deployment make sure to you read the following instructions carefully.

## Important Notes

- If you are looking for information about when to use the GUI installer and when to use the CLI, see the Installation Overview.
- Review the Chapter 2, *Hardware and Software Support Matrix* that we have listed.
- Review the release notes to make yourself aware of any known issues and limitations.
- The installation process can occur in different phases. For example, you can install the control plane only and then add Compute nodes afterwards if you would like.
- If you run into issues during installation, we have put together a list of Chapter 18, *Troubleshooting the Installation* you can reference.
- Make sure all disks on the system(s) are wiped before you begin the install. (For Swift, refer to the section called “Swift Requirements for Device Group Drives”)
- There is no requirement to have a dedicated network for OS-install and system deployment, this can be shared with the management network. More information can be found in Chapter 10, *Example Configurations*.
- You may see the terms deployer and lifecycle manager used interchangeably. These are referring to the same nodes in your environment.
- When running the Ansible playbook in this installation guide, if a runbook fails you will see in the error response to use the `--limit` switch when retrying a playbook. This should be avoided. You can simply re-run any playbook without this switch.

- DVR is not supported with ESX compute.
- When you attach a Cinder volume to the VM running on the ESXi host, the volume will not get detected automatically. Make sure to set the image metadata **vmware\_adaptertype=lsiLogicsas** for image before launching the instance. This will help to discover the volume change appropriately.

## Before You Start

We have put together a Chapter 2, *Pre-Installation Checklist* that should help with the recommended pre-installation tasks.

## Prerequisites

ESX/vCenter integration is not fully automatic, vCenter administrators are advised of the following responsibilities to ensure secure operation:

1. The VMware administrator is responsible for administration of the vCenter servers and the ESX nodes using the VMware administration tools. These responsibilities include:
  - a. Installing and configuring vCenter server
  - b. Installing and configuring ESX server and ESX cluster
  - c. Installing and configuring shared datastores
  - d. Establishing network connectivity between the ESX network and the HPE Helion OpenStack® management network
2. The VMware administration staff is responsible for the review of vCenter logs. These logs are not automatically included in HPE Helion OpenStack® centralized logging.
3. The VMware administrator is responsible for administration of the vCenter servers and the ESX nodes using the VMware administration tools.
4. Logging levels for vCenter should be set appropriately to prevent logging of the password for the HPE Helion OpenStack® message queue.
5. The vCenter cluster and ESX Compute nodes must be appropriately backed up.
6. Backup procedures for vCenter should ensure that the file containing the HPE Helion OpenStack® configuration as part of Nova and Cinder volume services is backed up and the backups are protected appropriately.
7. Since the file containing the HPE Helion OpenStack® message queue password could appear in the swap area of a vCenter server, appropriate controls should be applied to the vCenter cluster to prevent discovery of the password via snooping of the swap area or memory dumps
8. It is recommended to have a common shared storage for all the ESXi hosts in a particular cluster.
9. Ensure that you have enabled HA (High Availability) and DRS (Distributed Resource Scheduler) settings in a cluster configuration before running the installation. DRS/HA is disabled only for OVSVApp. This is done so that it does not move to a different host. If you do not enable DRS/HA prior to installation then you will not be able to disable it only for OVSVApp. As a result DRS/HA can migrate OVSVApp to different host, which will create a network loop.

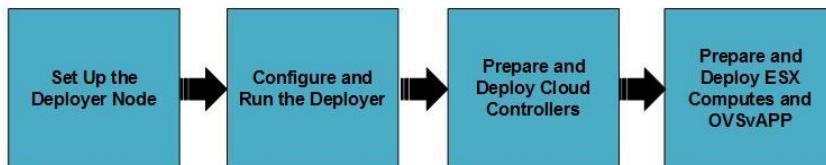
## Note

No two clusters should have the same name across datacenters in a given vCenter.

10.L3 HA VRRP is enabled by default.

# Deploy ESX Cloud with OVSVApp

The following diagram depicts the procedure to configure and deploy ESX cloud.



# Procedure to Deploy ESX Cloud with OVSVApp

## Setting Up the Lifecycle Manager

### Installing the Lifecycle Manager

The lifecycle manager will contain the installation scripts and configuration files to deploy your cloud. You can set up the lifecycle manager on a dedicated node or you do so on your first controller node. The default choice is to use the first controller node as the lifecycle manager.

1. Sign in and download the product and signature files at the link below:
  - a. Software Entitlement Portal [<http://www.hpe.com/software/entitlements>]
2. You can verify the download was complete via the signature verification process outlined here [<https://h20392.www2.hpe.com/portal/swdepot/displayProductInfo.do?productNumber=HPLinuxCodeSigning>].
3. Boot your lifecycle manager from the ISO contained in the download.
4. Enter "install" to start installation.

## Note

"install" is all lower case

5. Select the language. Note that only the English language selection is currently supported.
6. Select the location.
7. Select the keyboard layout.
8. Select the primary network interface, if prompted:
  - a. Assign IP address, subnet mask, and default gateway
9. Create new account:

- a. Enter a username.
- b. Enter a password.
- c. Enter time zone.

Once the initial installation is finished, complete the lifecycle manager setup with these steps:

1. Ensure your lifecycle manager has a valid DNS nameserver specified in /etc/resolv.conf.
2. Set the environment variable LC\_ALL:

```
export LC_ALL=C
```

## Note

This can be added to ~stack/.bashrc or /etc/bash.bashrc.

At the end of this section you should have a node set up with Linux for HPE Helion on it.

## Configure and Run the Lifecycle Manager

### Important

It is critical that you don't run any of the commands below as the root user or use sudo, unless it is stated explicitly in the steps. Run them as the user you just created (or stack if you left the default of "stack").

1. Log into your lifecycle manager as the user you created and mount the install media at /media/cdrom. It may be necessary to use wget or another file transfer method to transfer the install media to the lifecycle manager before completing this step. Here is the command to mount the media:

Example for HPE Helion OpenStack® 5.0:

```
sudo mount HelionOpenStack-5.0.iso /media/cdrom
```

2. Unpack the tarball that is in the /media/cdrom/hos/ directory:

Example for HPE Helion OpenStack 5.0

```
tar xvf /media/cdrom/hos/hos-5.0.0-<timestamp>.tar
```

3. Run the hos-init.bash script which is included in the build:

Example for HPE Helion OpenStack® 5.0

```
~/hos-5.0.0/hos-init.bash
```

You will be prompted to enter an optional SSH passphrase when running hos-init.bash. This passphrase is used to protect the key used by Ansible when connecting to its client nodes. If you do not want to use a passphrase then just press **Enter** at the prompt.

For automated installation (e.g. CI) it is possible to disable SSH passphrase prompting by setting the HOS\_INIT\_AUTO environment variable before running hos-init.bash, like this:

```
export HOS_INIT_AUTO=y
```

---

If you have protected the SSH key with a passphrase then execute the following commands to avoid having to enter the passphrase on every attempt by Ansible to connect to its client nodes:

```
eval $(ssh-agent)  
ssh-add ~/.ssh/id_rsa
```

At the end of this section you should have a local directory structure, as described below:

~/helion/	Top level directory
~/helion/examples/	Directory contains the config input files of the
~/helion/my_cloud/definition/	Directory contains the config input files
~/helion/my_cloud/config/	Directory contains .j2 files which are symlinks to
~/helion/hos/	Directory contains files used by the installer

## Warning

It is important that you do not add any extra files in the ~/helion directory on your lifecycle manager. This includes temporary save files. Doing so will cause errors during the installation.

## Important

Be aware that the files in the ~/helion/my\_cloud/config/ directory are symlinks to the ~/helion/hos/ansible/ directory. For example, ~/helion/my\_cloud/config/keepalived/defaults.yml is a symbolic link to ~/helion/hos/ansible/roles/keepalived/defaults/main.yml.

```
ls -al ~/helion/my_cloud/config/keepalived/defaults.yml  
lrwxrwxrwx 1 stack stack 55 May 24 20:38 /home/stack/helion/my_cloud/config/kee
```

If you are using a tool like **sed** to make edits to files in this directory, you might break the symbolic link and create a new copy of the file. To maintain the link, you will need to force **sed** to follow the link:

```
sed -i --follow-symlinks \  
'$keepalived_vrrp_offset: 0$keepalived_vrrp_offset: 2$' \  
~/helion/my_cloud/config/keepalived/defaults.yml
```

Alternatively, directly edit the target of the link ~/helion/hos/ansible/roles/keepalived/defaults/main.yml.

For any troubleshooting information regarding these steps, see the section called “Issues during Lifecycle-manager Setup”.

# Prepare and Deploy Cloud Controllers

1. Setup your configuration files, as follows:

a. Copy the example configuration files into the required setup directory and edit them as required:

```
cp -r ~/helion/examples/entry-scale-esx-kvm-vsa/* ~/helion/my_cloud/definition
```

See a sample set of configuration files in the ~/helion/examples/entry-scale-esx-kvm-vsa directory. The accompanying README.md file explains the contents of each of the configuration files.

- b. Begin inputting your environment information into the configuration files in the `~/helion/my_cloud/definition` directory.

## Note

If you want to use a dedicated deployer node in your ESX deployment, add **eon-client** service-component, to manage vCenter via EON operation from the deployer node, in the `control_plane.yml` file as shown in the following example.

```
clusters:
  - name: cluster0
    cluster-prefix: c0
    server-role: HLM-ROLE
    service-components:
      - lifecycle-manager
      - eon-client
    ...
  ...
```

2. Commit your cloud deploy configuration to the Chapter 3, *Using Git for Configuration Management*, as follows:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

## Note

This step needs to be repeated any time you make changes to your configuration files before you move onto the following steps. See Chapter 3, *Using Git for Configuration Management* for more information.

# Provisioning Your Baremetal Nodes

To provision the baremetal nodes in your cloud deployment you can either use the automated operating system installation process provided by HPE Helion OpenStack or you can use the 3rd party installation tooling of your choice. We will outline both methods below:

### Using 3rd Party Baremetal Installers

If you do not wish to use the automated operating system installation tooling included with 5.0 then the requirements that have to be met using the installation tooling of your choice are:

- The operating system must be installed via the HPE Linux for HPE Helion OpenStack ISO provided on the Software Entitlement Portal [<http://www.hpe.com/software/entitlements>].
- Each node must have SSH keys in place that allows the same user from the lifecycle manager node who will be doing the deployment to SSH to each node without a password.
- Passwordless sudo needs to be enabled for the user.
- There should be a LVM logical volume as `/root` on each node.
- If the LVM volume group name for the volume group holding the "root" LVM logical volume is `hlm-vg` then it will align with the disk input models in the examples.

- Ensure that `openssh-server`, `python`, `python-apt`, and `rsync` are installed.

If you chose this method for installing your baremetal hardware, skip forward to the Chapter 10, *Installing Mid-scale and Entry-scale KVM* step.

If you would like to use the automated operating system installation tools provided by 5.0 then complete all of the steps below.

## Using the Automated Operating System Installation Provided by HPE Helion OpenStack

### Part One: Deploy Cobbler

This phase of the install process takes the baremetal information that was provided in `servers.yml` and installs the Cobbler provisioning tool and loads this information into Cobbler. This sets each node to `netboot-enabled: true` in Cobbler. Each node will be automatically marked as `netboot-enabled: false` when it completes its operating system install successfully. Even if the node tries to PXE boot subsequently, Cobbler will not serve it. This is deliberate so that you can't reimagine a live node by accident.

The `cobbler-deploy.yml` playbook prompts for a password - this is the password that will be encrypted and stored in Cobbler, which is associated with the user running the command on the lifecycle manager, that you will use to log in to the nodes via their consoles after install. The username is the same as the user set up in the initial dialogue when installing the lifecycle manager from the iso, and is the same user that is running the `cobbler-deploy` play.

1. Run the following playbook which confirms that there is iLo connectivity for each of your nodes so that they are accessible to be re-imaged in a later step:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost bm-power-status.yml
```

2. Run the following playbook to deploy Cobbler:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

### Part Two: Image the Nodes

This phase of the install process goes through a number of distinct steps:

1. Powers down the nodes to be installed
2. Sets the nodes hardware boot order so that the first option is a network boot.
3. Powers on the nodes. (The nodes will then boot from the network and be installed using infrastructure set up in the previous phase)
4. Waits for the nodes to power themselves down (this indicates a success install). This can take some time.
5. Sets the boot order to hard disk and powers on the nodes.
6. Waits for the nodes to be ssh-able and verifies that they have the signature expected.

The reimage command is:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost bm-reimage.yml [-e nodelist=node1,node2,node3]
```

If a nodelist is not specified then the set of nodes in cobbler with `netboot-enabled: True` is selected. The playbook pauses at the start to give you a chance to review the set of nodes that it is targeting and to confirm that it's correct.

You can use the command below which will list all of your nodes with the `netboot-enabled: True` flag set:

```
sudo cobbler system find --netboot-enabled=1
```

For any troubleshooting information regarding these steps, see the section called “Issues while Provisioning your Baremetal Nodes”.

## Running the Configuration Processor

Once you have your configuration files setup you need to run the configuration processor to complete your configuration.

When you run the configuration processor you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the ansible `group_vars` and `host_vars` that it produces for subsequent deploy runs. You will need this password for subsequent ansible deploy and configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise just press **Enter** to bypass this.

Run the configuration processor with this command:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

For automated installation (e.g. CI) you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor

```
ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e rekey=""
```

If you receive an error during this step then there is probably an issue with one or more of your configuration files. We recommend that you verify that all of the information in each of your configuration files is correct for your environment and then commit those changes to git using the instructions in the previous section before re-running the configuration processor again.

For any troubleshooting information regarding these steps, see the section called “Issues while Updating Configuration Files”.

## Deploying the Cloud

1. Use the playbook below to create a deployment directory:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

2. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your partitions on your nodes are completely wiped before continuing with the installation. If you are using fresh machines this step may not be necessary.

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts wipe_disks.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts wipe_disks.yml --ask-vault-pass
```

3. Run the site.yml playbook below:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts site.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts site.yml --ask-vault-pass
```

### Note

The step above runs osconfig to configure the cloud and hlm-deploy to deploy the cloud. Therefore, this step may run for a while, perhaps 45 minutes or more, depending on the number of nodes in your environment.

4. Verify that the network is working correctly. Ping each IP in the /etc/hosts file from one of the controller nodes.

For any troubleshooting information regarding these steps, see the section called “Issues while Deploying the Cloud”.

## Prepare and Deploy ESX Computes and OVS-vAPPs

The following sections describe the procedure to install and configure ESX compute and OVSVAPPs on vCenter.

### Preparation for ESX Cloud Deployment

This section describes the procedures to prepare and deploy the ESX computes and OVSVAPPs for deployment.

1. Login to the lifecycle manager.
2. Source service.osrc.
- 3.
4.
  - a.
  - b.
  - c.

### Manage vCenters and Clusters

The following section describes the detailed procedure on managing the vCenters and clusters.

#### Register a vCenter Server

## Note

- If ESX related roles are added in the input model after the controller cluster is up, you must run `hlm-reconfigure.yml` before adding the eon resource-manager.
- Make sure to provide single quotes (' ') or backslash (\) for the special characters (& ! ; " ' ( ) | \ >) that are used when setting a password for vCenter registration . For example: '2the!Moon' or 2the\!Moon. You can use either of the format to set the vCenter password.

vCenter provides centralized management of virtual host and virtual machines from a single console.

1. Add a vCenter using EON python client.

```
# eon resource-manager-add [ --name <RESOURCE_MANAGER_NAME> ] --ip-address <RESOUR
```

where:

- Resource Manager Name - the identical name of the vCenter server.
- Resource Manager IP address - the IP address of the vCenter server.
- Resource Manager Username - the admin privilege username for the vCenter.
- Resource Manager Password - the password for the above username.
- Resource Manager Port - the vCenter server port. By default it is 443.
- Resource Manager Type - specify vcenter.

## Important

Please do not change the vCenter Port unless you are certain it is required to do so.

### Sample Output:

```
# eon resource-manager-add --name vc01 --ip-address 10.1.200.38 --username admin
+-----+-----+
| Property | Value |
+-----+-----+
| ID       | BC9DED4E-1639-481D-B190-2B54A2BF5674 |
| IPv4Address | 10.1.200.38 |
| Name     | vc01 |
| Password | <SANITIZED> |
| Port      | 443 |
| State     | registered |
| Type      | vcenter |
| Username  | administrator@vsphere.local |
+-----+-----+
```

### Show vCenter

1. Show vCenter using EON python client.

```
# eon resource-manager-show <RESOURCE_MANAGER_ID>
```

### Sample Output:

## Installation for Helion En- try Scale ESX (with OVS- vApp, KVM with VSA Model)

```
eon resource-manager-show BC9DED4E-1639-481D-B190-2B54A2BF5674
+-----+-----+
| Property | Value |
+-----+-----+
| Clusters | Cluster1, virtClust, Cluster2 |
| Datacenters | DC1, DC2 |
| ID | BC9DED4E-1639-481D-B190-2B54A2BF5674 |
| IPv4Address | 10.1.200.38 |
| Name | vc01 |
| Password | <SANITIZED> |
| Port | 443 |
| State | registered |
| Type | vcenter |
| Username | administrator@vsphere.local |
+-----+-----+
```

### Create and edit an activation template

This involves getting a sample network information template and modifying the details of the template. You will use the template to register the cloud network configuration for the vCenter.

1. Execute the following command to generate a sample activation template:

```
eon get-activation-template [--filename <ACTIVATION_JSON_NAME>] --type <RESOURCE_TYPE>
```

#### Sample Output:

```
eon get-activation-template --filename activationtemplate.json --type esxcluster
-----
Saved the sample network file in /home/user/activationtemplate.json
-----
```

2. Change to the /home/user/ directory:

```
cd /home/user/
```

3. Modify the template (json file) as per your environment. See the sample file in Chapter 12, *Sample activationtemplate.json File for ESX Compute*.

### Activate Clusters

This involves using the activation template to register the cloud network configuration for the vCenter.

This process spawns one compute proxy VM per cluster and one OVSVApp VM per host and configures the networking as defined in the JSON template. This process also updates the input model with the service VM details, executes the ansible playbooks on the new nodes, and performs a post activation check on the service VMs.

1. Activate the cluster for the selected vCenter.

```
# eon resource-activate <RESOURCE_ID> --config-json /home/user/<ACTIVATION_JSON_NAME>
```

### Note

To retrieve the resource ID, execute `eon resource-list`.

## Note

Activating the first cluster in a datacenter requires you to provide an activation template JSON. You can provide the same activation template or a new activation template to activate the subsequent clusters.

## Note

Minimum disk space required for a cluster activation is (number of hosts in that cluster +1 )\* 45 GB.

2. Execute the following command to view the status of the cluster:

```
eon resource-list
+-----+-----+-----+-----+
| ID           | Name      | Moid      | Resource Mana
+-----+-----+-----+-----+
| 1228fce5-df5d-445c-834e-ae633ac7e426 | Cluster2   | domain-c184 | BC9DED4E-1639
| 469710f6-e9f2-48a4-aace-1f00cbd60487 | virtClust  | domain-c943 | BC9DED4E-1639
| a3003a32-6e3a-4d89-a072-ec64a4247fb0 | Cluster1   | domain-c21  | BC9DED4E-1639
+-----+-----+-----+-----+
```

When the state is activated, the input model is updated with the service VM details, a git commit has been performed, the required playbooks and the post activation checks are completed successfully.

## Note

In multi-region setups, monitoring services will be deployed/running in a shared control-plane, unlike the normal deployment, where all services will be in one single control-plane. During ESX cluster activation in the multi-region scenarios, `generate_hosts_file` monitoring playbooks will be skipped due to absence of monitoring services in the ESX control-plane. To overcome this, you must run the playbook manually without limit.

```
cd ~/scratch/ansible/next/hos/ansible/
ansible-playbook -i hosts/verb_hosts site.yml --tag "generate_hosts_file"
```

## Note

Whenever an ESX compute is activated or deactivated, you must run the following playbook to reflect the changes in opsconsole.

```
cd scratch/ansible/next/hos/ansible/
ansible-playbook -i hosts/verb_hosts ops-console-reconfigure.yml
```

## Modify the Volume Configuration File

Once the cluster is activated you must configure the volume.

Perform the following steps to modify the volume configuration files:

1. Change the directory. The `cinder.conf.j2` is present in following directories :

```
cd /home/stack/helion/hos/ansible/roles/_CND-CMN/templates
```

OR

```
cd /home/stack/helion/my_cloud/config/cinder
```

It is recommended to modify the `cinder.conf.j2` available in `/home/stack/he-  
lion/my_cloud/config/cinder`

2. Modify the `cinder.conf.j2` as follows:

```
# Configure the enabled backends  
enabled_backends=<unique-section-name>  
  
# Start of section for VMDK block storage  
#  
# If you have configured VMDK Block storage for cinder you must  
# uncomment this section, and replace all strings in angle brackets  
# with the correct values for vCenter you have configured. You  
# must also add the section name to the list of values in the  
# 'enabled_backends' variable above. You must provide unique section  
# each time you configure a new backend.  
  
#[<unique-section-name>]  
#vmware_api_retry_count = 10  
#vmware_tmp_dir = /tmp  
#vmware_image_transfer_timeout_secs = 7200  
#vmware_task_poll_interval = 0.5  
#vmware_max_objects_retrieval = 100  
#vmware_volume_folder = cinder-volumes  
#volume_driver = cinder.volume.drivers.vmware.vmdk.VMwareVcVmdkDriver  
#vmware_host_ip = <ip_address_of_vcenter>  
#vmware_host_username = <vcenter_username>  
#vmware_host_password = <password>  
#  
#volume_backend_name = <vmdk-backend-name>  
#  
# End of section for VMDK block storage
```

### **Commit your Cloud Definition**

1. Add the cloud deployment definition to git :

```
cd /home/stack/helion/hos/ansible;  
git add -A;  
git commit -m 'Adding ESX Configurations or other commit message';
```

2. Prepare your environment for deployment:

```
ansible-playbook -i hosts/localhost config-processor-run.yml;  
ansible-playbook -i hosts/localhost ready-deployment.yml;  
cd /home/stack/scratch/ansible/next/hos/ansible;
```

### **Configuring VMDK block storage**

Execute the following command to configure VMDK block storage:

```
ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
```

---

If there are more backends (like VSA) defined, you must create and use specific volume type to ensure that volume is created in ESX, as shown below:

```
# cinder type-create "ESX VMDK Storage"
...
# cinder type-key "ESX VMDK Storage" set volume_backend_name=<name of VMDK backend>
...
# cinder create --volume-type "ESX VMDK Storage" 1
...
```

## Validate the block storage

You can validate that the VMDK block storage is added to the cloud successfully using the following command:

```
# cinder service-list
```

Binary	Host	Zone	Status	State	Up
cinder-backup	ha-volume-manager	nova	enabled	up	2016-06-14
cinder-scheduler	ha-volume-manager	nova	enabled	down	2016-06-13
cinder-scheduler	ha-volume-manager	nova	enabled	up	2016-06-14
cinder-volume	ha-volume-manager	nova	enabled	down	2016-06-13
cinder-volume	ha-volume-manager@vmdk1	nova	enabled	up	2016-06-14
cinder-volume	ha-volume-manager@vsal1	nova	enabled	up	2016-06-14

## Validate the compute

You can validate that the ESX compute cluster is added to the cloud successfully using the following command:

```
# nova service-list
```

ID	Binary	Host	Zone	Status	Sta
3	nova-conductor	esxhos-joh-core-m1-mgmt	internal	enabled	up
63	nova-scheduler	esxhos-joh-core-m1-mgmt	internal	enabled	up
66	nova-conductor	esxhos-joh-core-m2-mgmt	internal	enabled	up
111	nova-conductor	esxhos-joh-core-m3-mgmt	internal	enabled	up
129	nova-scheduler	esxhos-joh-core-m3-mgmt	internal	enabled	up
132	nova-consoleauth	esxhos-joh-core-m1-mgmt	internal	enabled	up
135	nova-scheduler	esxhos-joh-core-m2-mgmt	internal	enabled	up
138	nova-compute	esxhos-joh-esx-comp0001-mgmt	nova	enabled	up

You can validate the hypervisor list using the following command:

```
# nova hypervisor-list
```

ID	Hypervisor hostname	State	Status

Installation for Helion En-  
try Scale ESX (with OVS-  
vApp, KVM with VSA Model)

	9	domain-c40.9FDCFA66-6677-42A1-83FF-16DC32448021	up	enabled	
--	---	---	----	---------	--

## Validate the neutron

You can validate that the networking of the ESX cluster using the following command:

```
# neutron agent-list
```

+-----+   id   +-----+	+-----+   agent_type   +-----+	+-----+   host   +-----+
097bdbbc3-108c-41ca-8b52-9d249f65077f   Metadata agent   esxhos-joh-core-m3-m	DHCP agent   esxhos-joh-core-m3-m	
2b255256-9505-489e-93bf-0d37f7ff83e4   OVSVApp Agent   esxhos-joh-esx-ovsva		
43843bcb-f929-4a42-a1e1-207ff97dc09e   L3 agent   esxhos-joh-core-m3-m		
4f0be657-fe9e-4bab-be8-d8e688b6573e   OVSVApp Agent   esxhos-joh-esx-ovsva		
6db7d585-604e-4205-b29c-b9684bfb0cb2   DHCP agent   esxhos-joh-core-m1-m		
700f1a18-d237-4605-bf54-145951bd12db   DHCP agent   esxhos-joh-core-m2-m		
a723fb23-2c87-4f69-b272-643df870f0b6   L3 agent   esxhos-joh-core-m1-m		
bd512ed5-9f39-4ad6-a505-148f58cf2e64   Open vSwitch agent   esxhos-joh-core-m2-m		
e3d4b8be-077e-4503-bed2-031871f3829e   Open vSwitch agent   esxhos-joh-core-m1-m		
e5d75917-1d26-4443-99f4-45d8c574e3f0   Metadata agent   esxhos-joh-core-m1-m		
e6a40540-adad-4ee7-b194-db18ac7287bd   Metadata agent   esxhos-joh-core-m2-m		
f019c268-a6bd-4f66-8a46-6762c720a5bd   L3 agent   esxhos-joh-core-m1-m		
fc83c8dd-ea4-4ad5-aed3-95ab41e567bc   Open vSwitch agent   esxhos-joh-core-m3-m		

---

# Chapter 12. Sample activationtemplate.json File for ESX Compute

The process for installing the Helion Entry Scale ESX, KVM with VSA Model requires creating a sample network information template and modifying the details of the template. You will use the template to register the cloud network configuration for the vCenter. The template is in the JSON format.

Execute the following command to get the activation template.

```
eon get-activation-template [--filename <Activate JSON>] --type <Type of the resource>
```

For example:

```
eon get-activation-template --filename activationtemplate.json --type esxcluster
-----
Saved the sample network file in /home/user/activationtemplate.json
-----
```

EON supports two network driver for ESX deployment. Based on your deployment of ESX cloud, the template (activationtemplate.json) is automatically pre-configured specific to the driver

1. OVSvApp driver: For ESX with OVSvApp driver (default)
2. NoOp driver: For ESX with DCN driver

A sample of activationtemplate.json file for OVSvApp driver (default) and NoOp are shown as follows.

- **OVSvApp template**

A Sample json file for OVSvApp driver where networking solution is provided by OVSvApp appliances (default).

```
{
  "input_model": {
    "server_group": "RACK1"
  },
  "network_properties": {
    "switches": [
      {
        "type": "dvSwitch",
        "name": "MGMT-DVS",
        "physical_nics": "vmnic1",
        "mtu": "1500"
      },
      {
        "type": "dvSwitch",
        "name": "TRUNK-DVS",
        "physical_nics": "",
        "mtu": "1500"
      }
    ]
  }
}
```

Sample activationTemplate.json File for ESX Compute

---

```
        ] ,
      "vm_config": [
        {
          "nics": [
            {
              "device": "eth0",
              "type": "vmxnet3",
              "pci_id": "",
              "portGroup": "ESX-CONF"
            },
            {
              "device": "eth1",
              "type": "vmxnet3",
              "pci_id": "",
              "portGroup": "MGMT"
            }
          ],
          "server_role": "ESX-COMPUTE-ROLE",
          "memory_in_mb": "4096",
          "cpu": "4"
        },
        {
          "nics": [
            {
              "device": "eth0",
              "type": "vmxnet3",
              "pci_id": "",
              "portGroup": "ESX-CONF"
            },
            {
              "device": "eth1",
              "type": "vmxnet3",
              "pci_id": "",
              "portGroup": "MGMT"
            },
            {
              "device": "eth2",
              "type": "vmxnet3",
              "pci_id": "",
              "portGroup": "GUEST"
            },
            {
              "device": "eth3",
              "type": "vmxnet3",
              "pci_id": "",
              "portGroup": "TRUNK"
            }
          ],
          "server_role": "OVSAPP-ROLE",
          "memory_in_mb": "4096",
          "cpu": "4"
        }
      ],
      "portGroups": [
```

```
{  
  "nic_teaming": {  
    "network_failover_detection": "1",  
    "notify_switches": "yes",  
    "load_balancing": "1",  
    "active_nics": "vmnic1"  
  },  
  "vlan_type": "trunk",  
  "vlan": "33",  
  "name": "ESX-CONF",  
  "switchName": "MGMT-DVS"  
},  
{  
  "nic_teaming": {  
    "network_failover_detection": "1",  
    "notify_switches": "yes",  
    "load_balancing": "1",  
    "active_nics": "vmnic1"  
  },  
  "vlan_type": "none",  
  "vlan": "",  
  "name": "MGMT",  
  "switchName": "MGMT-DVS"  
},  
{  
  "vlan": "34",  
  "name": "GUEST",  
  "vlan_type": "trunk",  
  "switchName": "MGMT-DVS",  
  "nic_teaming": {  
    "network_failover_detection": "1",  
    "notify_switches": "yes",  
    "load_balancing": "1",  
    "active_nics": "vmnic1"  
  },  
  "cloud_network_type": "vxlan"  
},  
{  
  "vlan_type": "trunk",  
  "vlan": "1-4094",  
  "name": "TRUNK",  
  "switchName": "TRUNK-DVS"  
}  
],  
"template_info": {  
  "upload_to_cluster": false  
},  
"esx_conf_net": {  
  "start_ip": "",  
  "cidr": "10.20.18.0/23",  
  "end_ip": "",  
  "gateway": "10.20.18.1",  
  "portGroup": "ESX-CONF"  
}
```

```
        }
    }
```

- **NoOp template**

A sample json file for NoOp driver where networking solution is provided by DCN.

```
{
  "input_model": {
    "server_group": "RACK1"
  },
  "network_properties": {
    "switches": [
      {
        "type": "dvSwitch",
        "name": "MGMT-DVS",
        "physical_nics": "vmnic1",
        "mtu": "1500"
      }
    ],
    "vm_config": [
      {
        "nics": [
          {
            "device": "eth0",
            "type": "vmxnet3",
            "pci_id": "",
            "portGroup": "ESX-CONF"
          },
          {
            "device": "eth1",
            "type": "vmxnet3",
            "pci_id": "",
            "portGroup": "MGMT"
          }
        ],
        "server_role": "ESX-COMPUTE-ROLE",
        "memory_in_mb": "4096",
        "cpu": "4"
      }
    ],
    "portGroups": [
      {
        "nic_teaming": {
          "network_failover_detection": "1",
          "notify_switches": "yes",
          "load_balancing": "1",
          "active_nics": "vmnic1"
        },
        "vlan_type": "trunk",
        "vlan": "33",
        "name": "ESX-CONF",
        "switchName": "MGMT-DVS"
      }
    ]
}
```

```

    "nic_teaming": {
        "network_failover_detection": "1",
        "notify_switches": "yes",
        "load_balancing": "1",
        "active_nics": "vmnic1"
    },
    "vlan_type": "none",
    "vlan": "",
    "name": "MGMT",
    "switchName": "MGMT-DVS"
}
],
"template_info": {
    "upload_to_cluster": false
},
"esx_conf_net": {
    "start_ip": "",
    "cidr": "10.20.18.0/23",
    "end_ip": "",
    "gateway": "10.20.18.1",
    "portGroup": "ESX-CONF"
}
}
}

```

The above template consists of the following sections:

- Input model
  - Server group
  - Network properties
    - Port Groups to be created or reused
    - Distributed Virtual Switches to be created or reused
    - Template information
    - Virtual machine configuration (RAM, CPU and network interfaces)
    - Ansible network

The main purpose of the activation template is to create an environment to host `nova-compute-proxy` (in case of OVSvApp driver) or only `nova-compute-proxy` (in case of NoOp driver). The environment here refers to the required virtual networking (Distributed Virtual Switch (DVS) and port groups), VMs (RAM, CPU, NIC and associated server role) going to host the Neutron and Nova services.

- **Input model**

It describes the configuration of each entities in the input model.

In the following example we have specified the value of `server_group` as RACK1.

```
{
    "input_model": {

```

```
    "server_group": "RACK1"
},
```

- **Server group**

This key takes values specified in `server_groups.yml` file defined in the input model. Default server groups defined in the input model are "RACK1", "RACK2", and "RACK3" based on different network definitions. This implies that all the member hosts in the cluster are part of the same RACK.

- **Network properties**

You must define and configure the following network properties.

- **Port Groups**

EON creates the portGroups (PGs) if the specified PG is unavailable. The `portGroups` key contains a JSON list of one or more entries.

You can set the tenant network type as VLAN or VXLAN or both VLAN and VXLAN. The `cloud_network_type` must be set as per the portGroup.

## Important

Do not modify the VLAN range for trunk PG. The value should remain as 1-4094

In the following example there are three portGroups with a same tenant network type with an active NIC. Also, if there is any failover in the switches then it is notified as the value is set to `yes` with the failover detection set to 1.

```
"portGroups": [
  {
    "nic_teaming": {
      "network_failover_detection": "1",
      "notify_switches": "yes",
      "load_balancing": "1",
      "active_nics": "vmnic1"
    },
    "vlan_type": "trunk",
    "vlan": "33",
    "name": "ESX-CONF",
    "switchName": "MGMT-DVS"
  },
  {
    "nic_teaming": {
      "network_failover_detection": "1",
      "notify_switches": "yes",
      "load_balancing": "1",
      "active_nics": "vmnic1"
    },
    "vlan_type": "trunk",
    "vlan": "3307",
    "name": "MGMT",
    "switchName": "MGMT-DVS"
  },
  {
    "
```

---

Sample activationTemplate.json File for ESX Compute

---

```
"vlan": "3285",
"name": "GUEST",
"vlan_type": "trunk",
"switchName": "MGMT-DVS",
"nic_teamming": {
  "network_failover_detection": "1",
  "notify_switches": "yes",
  "load_balancing": "1",
  "active_nics": "vmnic1"
},
"cloud_network_type": "vxlan"
},
```

In the following example a portGroup **GUEST** has an active NIC and **both** tenant network types. Also, if there is any failover in the switches then it is notified as the value is set to **yes** with the failover detection set to 1. If you want only tenant vlan, you can define this.

```
"portGroups": [
  {
    "name": "GUEST",
    "vlan_type": "trunk",
    "vlan": "3285,3286-3290",
    "switchName": "MGMT-DVS",
    "nic_teamming": {
      "network_failover_detection": "1",
      "notify_switches": "yes",
      "load_balancing": "1",
      "active_nics": "vmnic1"
    }
    "cloud_network_type": "vxlan, vlan"
  }
]
```

In the following example a portGroup **GUEST** has an active NIC with vlan as a tenant network type. Also, if there is any failover in the switches then it is notified as the value is set to **yes** with the failover detection set to 1.

```
"portGroups": [
  {
    "name": "GUEST",
    "vlan_type": "trunk",
    "vlan": "3285-3290",
    "switchName": "MGMT-DVS",
    "nic_teamming": {
      "network_failover_detection": "1",
      "notify_switches": "yes",
      "load_balancing": "1",
      "active_nics": "vmnic1"
    }
    "cloud_network_type": "vlan"
  }
]
```

In the following example a portGroup **GUEST** has an active NIC with vxlan as a tenant network types. Also, if there is any failover in the switches then it is notified as the value is set to **yes** with the failover detection set to 1.

```
"portGroups": [
    {
        "name": "GUEST",
        "vlan_type": "trunk",
        "vlan": "3285",
        "switchName": "MGMT-DVS",
        "nic_teaming": {
            "network_failover_detection": "1",
            "notify_switches": "yes",
            "load_balancing": "1",
            "active_nics": "vmnic1"
        }
        "cloud_network_type": "vxlan"
    }
]
```

The following table provides the list of parameters that must be configured.

Parameter	Description
name (String)	Unique name for the portGroup to be created or used.
vlan_type (String)	The following are acceptable VLAN types: <ul style="list-style-type: none"> <li>• none = Untagged port</li> <li>• vlan = VLAN port and only the VLAN ID mentioned in "VLAN" key is allowed</li> <li>• trunk = Port allows multiple VLAN networks as mentioned by "VLAN" Key</li> </ul>
vlan	Uplinks for the switch (comma separated string). The vlan is configured on the NIC. For example, "vlan" : "33" or "vlan" : "0,33-35".  Example: Two uplinks -- "vmnic0, vmnic1". Single uplink – 'vmnic0'. No uplinks – ".
switchName (String)	Name of the Distributed Virtual Switch (DVS) under which this port-group should be created.
load_balancing (Int)	Specify how to choose an uplink. Acceptable values are 1 to 5. <ol style="list-style-type: none"> <li>1. Route based on the originating virtual port</li> <li>2. Route based on IP hash</li> <li>3. Route based on source MAC hash</li> <li>4. Route based on physical NIC load</li> </ol>

Sample activationtem-  
plate.json File for ESX Compute

---

Parameter	Description
	5. Use explicit failover order
notify_switches (Boolean)	Select whether to notify switches in the case of failover.
network_failover_detection (Int)	Specify the method to use for failover detection. <ul style="list-style-type: none"> <li>• Link Status only</li> <li>• Beacon Probing</li> </ul>
active_nics	(Comma separated string) Specify the uplinks to be used.
cloud_network_type (String)	(Optional) Specify the tenant network type ("vlan" or "vxlan", or both ("vlan", "vxlan"). In case of VLAN, the security policy for the port group is modified to enable 'Promiscuous Mode' and 'Forged transmits'.

Configuration of PG carrying Data in the single NIC VXLAN model.

```
+-----+-----+
| Input model | Net_conf.json Vxlan |
+-----+-----+
| tagged-vlan: true | "vlan": "33",
| vlanid: 33 | "vlan_type": "trunk",
+-----+-----+
| tagged-vlan: false | "vlan": "",
| vlanid: | "vlan_type": "none",
+-----+-----+
| tagged-vlan: false | If the physical uplink
| vlanid: 33 | switch is tagged to
| | particular vlan
| | "vlan": "33",
| | "vlan_type": "vlan",
+-----+-----+
```

Configuration of PG carrying Data in the single NIC VLAN model.

```
+-----+-----+
| Input model | Net_conf.json Vlan |
+-----+-----+
| tagged-vlan: false | "vlan": "0, 1231-1331",
| vlanid: |
| tenant-vlan-id:1231-1331 | "vlan_type": "trunk",
+-----+-----+
```

• **Distributed Virtual Switches**

EON creates the virtual switch if the specified switch does not exists in the vCenter. User needs to provide the same kind of input required in creating one from the vSphere web-client.

---

Sample activation.json File for ESX Compute

---

When 2 networks ( PXE & CLM) are using a single dvSwitch(DVS-MGMT) and their MTU setting are different, the MTU setting of the network which have higher value should be assigned to the dvSwitch(DVS-MGMT) in the ESX activation.json.

**OVSvApp:** A minimum of two switch entries are required. One for OVSvApp trunk DVS with no physical\_nics and other for management switch with physical\_nics. In the following example we have created two DVS with a name MGMT-DVS (with vmnic1 as the uplink) and TRUNK-DVS (with no uplink)

In the following example we have created two DVS with a name MGMT-DVS and TRUNK-DVS and used vmnic1 as the uplink for the switch.

```
"switches": [
  {
    "type": "dvSwitch",
    "name": "MGMT-DVS",
    "physical_nics": "vmnic1",
    "mtu": "1500"
  },
  {
    "type": "dvSwitch",
    "name": "TRUNK-DVS",
    "physical_nics": "",
    "mtu": "1500"
  }
]
```

**NoOp:** One management switch entry is required with physical\_nics

In the following example we have created a switch with a name MGMT-DVS and used vmnic1 as the uplink for the switch.

```
"switches": [
  {
    "type": "dvSwitch",
    "name": "MGMT-DVS",
    "physical_nics": "vmnic1",
    "mtu": "1500"
  },
]
```

The following table provides the list of parameters that must be configured.

Parameter	Description
type	The category of switch. It can be Virtual Standard Switch (VSS) or Distributed Virtual Switch (DVS). The supported type of switch is DVS.
name (String)	Unique name for the switch.
physical_nics	Uplinks for the switch (comma separated string). Example: Two uplinks -- "vmnic0, vmnic1". Single uplink – 'vmnic0'. No uplinks – " ".

Parameter	Description
MTU (Optional)	Maximum Transmission Unit (MTU) packet size. Provide "mtu" value within the range of 1500 to 9000.

- **Template information**

This key takes a dedicated boolean value (true or false) for the parameter `upload_to_cluster`. If the value is set to true, every activation of a cluster will upload the OVA template to the cluster. By default, it is set to false. In this case, the OVA is uploaded to the first cluster which is being activated. The format of the template name is **hlm-shell-vm-<hlm\_version>-<dc-name>**, user given "template\_name" will not be considered.

```
"template_info": { "upload_to_cluster": false}
```

- **Virtual machine configuration**

EON creates one `nova-compute-proxy` virtual machine and 'N' number of OVSvApp virtual machines where N represents the number of ESXi hosts participating in the cluster. In other words, a OVSvApp VM will be deployed per host.

In the following example we have mentioned the server role as **ESX-Compute-Role** and **OVS-VAPP-ROLE**. These roles are defined in the `server_role.yml` file of the input model. We require a template to create a virtual machine. 4 vCPU is configured along with 4096 MB memory. The name of the network interfaces are eth0, eth1, eth2, and eth3 and interface model is vmxnet3.

```
"vm_config": [
  {
    "nics": [
      {
        "device": "eth0",
        "type": "vmxnet3",
        "pci_id": "",
        "portGroup": "ESX-CONF"
      },
      {
        "device": "eth1",
        "type": "vmxnet3",
        "pci_id": "",
        "portGroup": "MGMT"
      }
    ],
    "server_role": "ESX-COMPUTE-ROLE",
    "memory_in_mb": "4096",
    "cpu": "4"
  },
  {
    "nics": [
      {
        "device": "eth0",
        "type": "vmxnet3",
        "pci_id": "",
        "portGroup": "ESX-CONF"
      },
      {
        "device": "eth1",
        "type": "vmxnet3",
        "pci_id": "",
        "portGroup": "MGMT"
      }
    ],
    "server_role": "OVS-VAPP-ROLE",
    "memory_in_mb": "4096",
    "cpu": "4"
  }
]
```

---

Sample activationtem-  
plate.json File for ESX Compute

---

```
{  
    "device": "eth1",  
    "type": "vmxnet3",  
    "pci_id": "",  
    "portGroup": "MGMT"  
},  
{  
    "device": "eth2",  
    "type": "vmxnet3",  
    "pci_id": "",  
    "portGroup": "GUEST"  
},  
{  
    "device": "eth3",  
    "type": "vmxnet3",  
    "pci_id": "",  
    "portGroup": "TRUNK"  
}  
],  
"server_role": "OVSVAPP-ROLE",  
"memory_in_mb": "4096",  
"cpu": "4"  
}  
],
```

EON creates only one nova-compute-proxy virtual machine for NoOp driver.

```
"vm_config": [  
    {  
        "nics": [  
            {  
                "device": "eth0",  
                "type": "vmxnet3",  
                "pci_id": "",  
                "portGroup": "ESX-CONF"  
            },  
            {  
                "device": "eth1",  
                "type": "vmxnet3",  
                "pci_id": "",  
                "portGroup": "MGMT"  
            }  
        ],  
        "server_role": "ESX-COMPUTE-ROLE",  
        "memory_in_mb": "4096",  
        "cpu": "4"  
    }  
]
```

The following table provides the list of parameters that must be configured.

Parameter	Description
server_role (String)	Specify the role associated to this appliance in the cloud input model.

Parameter	Description
CPU (Int)	Number of vCPUs to be configured.
memory_in_mb (Int)	Amount of memory to be configured in MB.
Device (String)	Network interface name. Generally in the format eth0, eth1... ethX.
portGroup (String)	Name of the virtual network (port-group) to be attached to the interface.
type (String)	Type of interface model. Supported models are 'vmxnet3' and 'e1000'.
pci_id	Device ID of the ESXi host when passthrough for a Network Device on the host is enabled.
nics	List of network interfaces to be connected to the virtual machine.

- **Ansible network**

In ESX cloud, you require a separate network for running ansible commands instead of sharing the management network. IPAM is managed by EON and assigns IP addresses to virtual machines from this network.

The following example provides a network configuration that is defined under `esx_conf_net`.

```
"esx_conf_net": {
    "start_ip": "",
    "cidr": "10.20.18.0/23",
    "end_ip": "",
    "gateway": "10.20.18.1",
    "portGroup": "ESX-CONF"
},
```

The following table provides the list of parameters that must be configured.

Parameter	Description
portGroup (String)	Name of the network interface for which IP address needs to be assigned.
CIDR (String)	Classless Inter-Domain Routing of ESX Configuration Network.
Start_ip (String)	First IPv4 allocation.
end_ip (String)	Last IPv4 allocation.
Gateway (String)	IPv4 address of the network gateway.

Return to installing .

---

# Chapter 13. Installing Baremetal (Ironic)

Bare Metal as a Service is enabled in this release for deployment of nova instances on bare metal nodes using flat networking. HPE DL and SL line of servers are supported.

## Installation for HPE Helion Entry-scale Cloud with Ironic Flat Network

This page describes the installation step requirements for the HPE Helion Entry-scale Cloud with Ironic Flat Network.

### Before You Start

We have put together a Chapter 2, *Pre-Installation Checklist* that should help with the recommended pre-installation tasks.

### Setting Up the Lifecycle Manager

#### Installing the Lifecycle Manager

The lifecycle manager will contain the installation scripts and configuration files to deploy your cloud. You can set up the lifecycle manager on a dedicated node or you do so on your first controller node. The default choice is to use the first controller node as the lifecycle manager.

1. Sign in and download the product and signature files at the link below:
  - a. Software Entitlement Portal [<http://www.hpe.com/software/entitlements>]
2. You can verify the download was complete via the signature verification process outlined here [<https://h20392.www2.hpe.com/portal/swdepot/displayProductInfo.do?productNumber=HPLinuxCodeSigning>].
3. Boot your lifecycle manager from the ISO contained in the download.
4. Enter "install" to start installation.

#### Note

"install" is all lower case

5. Select the language. Note that only the English language selection is currently supported.
6. Select the location.
7. Select the keyboard layout.
8. Select the primary network interface, if prompted:
  - a. Assign IP address, subnet mask, and default gateway

9. Create new account:
  - a. Enter a username.
  - b. Enter a password.
  - c. Enter time zone.

Once the initial installation is finished, complete the lifecycle manager setup with these steps:

1. Ensure your lifecycle manager has a valid DNS nameserver specified in /etc/resolv.conf.
2. Set the environment variable LC\_ALL:

```
export LC_ALL=C
```

### Note

This can be added to ~stack/.bashrc or /etc/bash.bashrc.

At the end of this section you should have a node set up with Linux for HPE Helion on it.

## Configure and Run the Lifecycle Manager

### Important

It is critical that you don't run any of the commands below as the root user or use sudo, unless it is stated explicitly in the steps. Run them as the user you just created (or stack if you left the default of "stack").

1. Log into your lifecycle manager as the user you created and mount the install media at /media/cdrom. It may be necessary to use wget or another file transfer method to transfer the install media to the lifecycle manager before completing this step. Here is the command to mount the media:

Example for HPE Helion OpenStack® 5.0:

```
sudo mount HelionOpenStack-5.0.iso /media/cdrom
```

2. Unpack the tarball that is in the /media/cdrom/hos/ directory:

Example for HPE Helion OpenStack 5.0

```
tar xvf /media/cdrom/hos/hos-5.0.0-<timestamp>.tar
```

3. Run the hos-init.bash script which is included in the build:

Example for HPE Helion OpenStack® 5.0

```
~/hos-5.0.0/hos-init.bash
```

You will be prompted to enter an optional SSH passphrase when running hos-init.bash. This passphrase is used to protect the key used by Ansible when connecting to its client nodes. If you do not want to use a passphrase then just press **Enter** at the prompt.

For automated installation (e.g. CI) it is possible to disable SSH passphrase prompting by setting the HOS\_INIT\_AUTO environment variable before running hos-init.bash, like this:

```
export HOS_INIT_AUTO=y
```

If you have protected the SSH key with a passphrase then execute the following commands to avoid having to enter the passphrase on every attempt by Ansible to connect to its client nodes:

```
eval $(ssh-agent)
ssh-add ~/.ssh/id_rsa
```

At the end of this section you should have a local directory structure, as described below:

~/helion/	Top level directory
~/helion/examples/	Directory contains the config input files of the
~/helion/my_cloud/definition/	Directory contains the config input files
~/helion/my_cloud/config/	Directory contains .j2 files which are symlinks to
~/helion/hos/	Directory contains files used by the installer

## Warning

It is important that you do not add any extra files in the ~/helion directory on your lifecycle manager. This includes temporary save files. Doing so will cause errors during the installation.

## Important

Be aware that the files in the ~/helion/my\_cloud/config/ directory are symlinks to the ~/helion/hos/ansible/ directory. For example, ~/helion/my\_cloud/config/keepalived/defaults.yml is a symbolic link to ~/helion/hos/ansible/roles/keepalived/defaults/main.yml.

```
ls -al ~/helion/my_cloud/config/keepalived/defaults.yml
lrwxrwxrwx 1 stack stack 55 May 24 20:38 /home/stack/helion/my_cloud/config/keepalived/defaults.yml --> /home/stack/helion/hos/ansible/roles/keepalived/defaults/main.yml
```

If you are using a tool like **sed** to make edits to files in this directory, you might break the symbolic link and create a new copy of the file. To maintain the link, you will need to force **sed** to follow the link:

```
sed -i --follow-symlinks \
's$keepalived_vrrp_offset: 0$keepalived_vrrp_offset: 2$' \
~/helion/my_cloud/config/keepalived/defaults.yml
```

Alternatively, directly edit the target of the link ~/helion/hos/ansible/roles/keepalived/defaults/main.yml.

For any troubleshooting information regarding these steps, see the section called “Issues during Lifecycle-manager Setup”.

# Configure Your Environment

Prior to deploying an operational environment with Ironic, operators need to be aware of the nature of TLS certificate authentication. As pre-built deployment agent ramdisks images are supplied, these ramdisk images will only authenticate known third-party TLS Certificate Authorities in the interest of end-to-end security. As such, uses of self-signed certificates and private certificate authorities will be unable to leverage ironic without modifying the supplied ramdisk images.

1. Setup your configuration files, as follows:
  - a. See the sample sets of configuration files in the ~/helion/examples/ directory. Each set will have an accompanying README.md file that explains the contents of each of the configuration files.

- b. Copy the example configuration files into the required setup directory and edit them to contain the details of your environment:

```
cp -r ~/helion/examples/entry-scale-ironic-flat-network/* ~/helion/my_cloud/de
```

- 2. [OPTIONAL] - You can use the `hosencrypt.py` script to encrypt your iLo passwords. This script uses OpenSSL.

- a. Change to the Ansible directory:

```
cd ~/helion/hos/ansible
```

- b. Put the encryption key into the following environment variable:

```
export HOS_USER_PASSWORD_ENCRYPT_KEY=<encryption key>
```

- c. Run the python script below and follow the instructions. Enter a password that you want to encrypt.

```
./hosencrypt.py
```

- d. Take the string generated and place it in the "ilo-password" field in your `~/helion/my_cloud/definition/data/servers.yml` file, remembering to enclose it in quotes.

- e. Repeat the above for each server.

## Note

Before you run any playbooks, remember that you need to export the encryption key in the following environment variable: `export HOS_USER_PASSWORD_ENCRYPT_KEY=<encryption key>`

- 3. Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*), as follows:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

## Important

This step needs to be repeated any time you make changes to your configuration files before you move onto the following steps. See Chapter 3, *Using Git for Configuration Management* for more information.

# Provisioning Your Baremetal Nodes

To provision the baremetal nodes in your cloud deployment you can either use the automated operating system installation process provided by HPE Helion OpenStack or you can use the 3rd party installation tooling of your choice. We will outline both methods below:

## Using 3rd Party Baremetal Installers

If you do not wish to use the automated operating system installation tooling included with 5.0 then the requirements that have to be met using the installation tooling of your choice are:

- The operating system must be installed via the HPE Linux for HPE Helion OpenStack ISO provided on the Software Entitlement Portal [<http://www.hpe.com/software/entitlements>].
- Each node must have SSH keys in place that allows the same user from the lifecycle manager node who will be doing the deployment to SSH to each node without a password.
- Passwordless sudo needs to be enabled for the user.
- There should be a LVM logical volume as /root on each node.
- If the LVM volume group name for the volume group holding the "root" LVM logical volume is hlm-vg then it will align with the disk input models in the examples.
- Ensure that `openssh-server`, `python`, `python-apt`, and `rsync` are installed.

If you chose this method for installing your baremetal hardware, skip forward to the Chapter 10, *Installing Mid-scale and Entry-scale KVM* step.

If you would like to use the automated operating system installation tools provided by 5.0 then complete all of the steps below.

## Using the Automated Operating System Installation Provided by HPE Helion OpenStack

### Part One: Deploy Cobbler

This phase of the install process takes the baremetal information that was provided in `servers.yml` and installs the Cobbler provisioning tool and loads this information into Cobbler. This sets each node to `netboot-enabled: true` in Cobbler. Each node will be automatically marked as `netboot-enabled: false` when it completes its operating system install successfully. Even if the node tries to PXE boot subsequently, Cobbler will not serve it. This is deliberate so that you can't reimagine a live node by accident.

The `cobbler-deploy.yml` playbook prompts for a password - this is the password that will be encrypted and stored in Cobbler, which is associated with the user running the command on the lifecycle manager, that you will use to log in to the nodes via their consoles after install. The username is the same as the user set up in the initial dialogue when installing the lifecycle manager from the iso, and is the same user that is running the cobbler-deploy play.

1. Run the following playbook which confirms that there is iLo connectivity for each of your nodes so that they are accessible to be re-imaged in a later step:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost bm-power-status.yml
```

2. Run the following playbook to deploy Cobbler:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

### Part Two: Image the Nodes

This phase of the install process goes through a number of distinct steps:

1. Powers down the nodes to be installed
2. Sets the nodes hardware boot order so that the first option is a network boot.
3. Powers on the nodes. (The nodes will then boot from the network and be installed using infrastructure set up in the previous phase)

4. Waits for the nodes to power themselves down (this indicates a success install). This can take some time.
5. Sets the boot order to hard disk and powers on the nodes.
6. Waits for the nodes to be ssh-able and verifies that they have the signature expected.

The reimage command is:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost bm-reimage.yml [-e nodelist=node1,node2,node3]
```

If a nodelist is not specified then the set of nodes in cobbler with netboot-enabled: True is selected. The playbook pauses at the start to give you a chance to review the set of nodes that it is targeting and to confirm that it's correct.

You can use the command below which will list all of your nodes with the netboot-enabled: True flag set:

```
sudo cobbler system find --netboot-enabled=1
```

For any troubleshooting information regarding these steps, see the section called “Issues while Provisioning your Baremetal Nodes”.

## Running the Configuration Processor

Once you have your configuration files setup you need to run the configuration processor to complete your configuration.

When you run the configuration processor you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the ansible group\_vars and host\_vars that it produces for subsequent deploy runs. You will need this password for subsequent ansible deploy and configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise just press **Enter** to bypass this.

Run the configuration processor with this command:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

For automated installation (e.g. CI) you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor

```
ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e reke
```

If you receive an error during this step then there is probably an issue with one or more of your configuration files. We recommend that you verify that all of the information in each of your configuration files is correct for your environment and then commit those changes to git using the instructions in the previous section before re-running the configuration processor again.

For any troubleshooting information regarding these steps, see the section called “Issues while Updating Configuration Files”.

## Deploying the Cloud

1. Use the playbook below to create a deployment directory:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

2. [OPTIONAL] - Run the `wipe_disks.yml` playbook to ensure all of your partitions on your nodes are completely wiped before continuing with the installation. If you are using fresh machines this step may not be necessary.

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts wipe_disks.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts wipe_disks.yml --ask-vault-pass
```

3. Run the `site.yml` playbook below:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts site.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts site.yml --ask-vault-pass
```

### Note

The step above runs `osconfig` to configure the cloud and `hlm-deploy` to deploy the cloud. Therefore, this step may run for a while, perhaps 45 minutes or more, depending on the number of nodes in your environment.

4. Verify that the network is working correctly. Ping each IP in the `/etc/hosts` file from one of the controller nodes.

For any troubleshooting information regarding these steps, see the section called “Issues while Deploying the Cloud”.

## Ironic configuration

Run the `ironic-cloud-configure.yml` playbook below:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts ironic-cloud-configure.yml
```

This step configures ironic flat network, uploads glance images and sets the ironic configuration.

To see the images uploaded to glance, run:

```
$ source ~/service.osrc
$ glance image-list
```

This will produce output like the following example, showing three images that have been added by Ironic:

ID	Name

d4e2a0ff-9575-4bed-ac5e-5130a1553d93	ir-deploy-iso-HOS3.0	
b759a1f0-3b33-4173-a6cb-be5706032124	ir-deploy-kernel-HOS3.0	
ce5f4037-e368-46f2-941f-c01e9072676c	ir-deploy-ramdisk-HOS3.0	

To see the network created by Ironic, run:

```
$ neutron net-list
```

This returns details of the "flat-net" generated by the Ironic install:

id	name	subnets	
f9474c06-6660-4a03-9644-f439e2a11010	flat-net	ca8f8df8-12c8-4e58-b1eb-76844	

## Node Configuration

### DHCP

Once booted, nodes obtain network configuration via DHCP. If multiple interfaces are to be utilized, a user may wish to pre-build images with settings to execute DHCP on all interfaces. An easy utility to build custom images is the diskimage-builder [<https://github.com/openstack/diskimage-builder>] utility which has a dhcp-all-interfaces [<https://github.com/openstack/diskimage-builder/tree/master/elements/dhcp-all-interfaces>] element that can be utilized to initiate DHCP on all interfaces.

The diskimage-builder tool does not support building UEFI images. The following patchset can be used to build UEFI images: <https://review.openstack.org/#/c/287784>.

## Configuration Drives

### Caution

Configuration Drives are stored unencrypted and should not include any sensitive data.

Users may wish to utilize Configuration Drives [[http://docs.openstack.org/user-guide/cli\\_config\\_drive.html](http://docs.openstack.org/user-guide/cli_config_drive.html)] to store metadata for initial boot setting customization. While Configuration Drives are extremely useful for initial machine configuration, as a general security practice, they should not include any sensitive data. Configuration Drives should only be trusted upon the initial boot of an instance. `cloud-init` utilizes a lock file for this purpose. Custom instance images should not rely upon the integrity of a Configuration Drive beyond the initial boot of a host as an administrative user with-in a deployed instance can potentially modify a configuration drive once written to disk and released for use.

## TLS Certificates with Ironic Python Agent (IPA) Images

As part of HPE Helion OpenStack 5.0, Ironic Python Agent, better known as IPA in the OpenStack community, images are supplied and loaded into glance. Two types of image exist. One is a traditional boot ramdisk which is used by the `agent_ipmitool`, `pxe_ipmitool`, and `pxe_ilo` drivers. The other is an ISO image that is supplied as virtual media to the host when using the `agent_ilo` driver.

As these images are built in advance, they are unaware of any private certificate authorities. Users attempting to utilize self-signed certificates or a private certificate authority will need to inject their signing certificate(s) into the image in order for IPA to be able to boot on a remote node, and ensure that the TLS

endpoints being connected to in HPE Helion OpenStack can be trusted. This is not an issue with publicly signed certificates.

As two different types of images exist, below are instructions for disassembling the image ramdisk file or the ISO image. Once this has been done, you will need to re-upload the files to glance, and update any impacted node's `driver_info`, for example, the `deploy_ramdisk` and `ilo_deploy_iso` settings that were set when the node was first defined. Respectively, this can be done with the

```
ironic node-update <node> replace driver_info/deploy_ramdisk=<glance_id>  
or  
ironic node-update <node> replace driver_info/ilo_deploy_iso=<glance_id>
```

## Adding a certificate into a ramdisk image

As root, from a folder where the ramdisk image is present, perform the following steps.

1. Download the ramdisk image to /tmp/ and name the file `ironic-deploy.initramfs`
2. Change your working directory to /tmp/

```
cd /tmp/
```

3. Create a temporary folder that will hold the temporarily extracted image contents.

```
mkdir new_deploy_ramdisk
```

4. Change your shell working directory to the temporary folder

```
cd /tmp/new_deploy_ramdisk
```

5. Extract the original deployment archive file.

```
zcat /tmp/ironic-deploy.initramfs | cpio --extract --make-directories
```

6. Append your CA certificate to the file located at /tmp/new\_deploy\_ramdisk/usr/local/lib/python2.7/dist-packages/requests/cacert.pem, example below:

```
cat your_ca_certificate.pem >> /tmp/new_deploy_ramdisk/usr/local/lib/python2.7/d
```

your\_ca\_certificate.pem is /etc/ssl/certs/ca-certificates.crt which can be obtained through

```
cat service.osrc | grep CACERT
```

7. Package a new ramdisk file. From within the /tmp/new\_deploy\_ramdisk folder, execute the following command:

```
find . | cpio --create --format='newc' | gzip -c -9 > /tmp/updated-ironic-deploy
```

Once completed, the new image can be found at /tmp/updated-ironic-deploy.initramfs, and will need to be uploaded to Glance. New Glance IDs will need to be recorded for any instances requiring this new image, as noted in the parent paragraph.

## Adding a certificate into an ISO image

As root, proceed with the following steps to disassemble the ISO image, and insert a new Certificate Authority or Signing Certificate into the provided image.

1. Download the deployment ISO image from glance and place it in /tmp/ as `ironic-deploy.iso`.

2. Make a temporary folder to mount the `ironic-deploy.iso` image to in order to extract its contents.

```
mkdir /mnt/deploy_iso
```

3. Mount the `ironic-deploy.iso` image file to the folder you just created.

```
mount -o loop /tmp/ironic-deploy.iso /mnt/deploy_iso/
```

4. Make a temporary location to stage the new image.

```
mkdir /tmp/new_deploy_iso
```

5. Copy the contents from the previous image into the new temporary staging location.

```
cp -ra /mnt/deploy_iso/. /tmp/new_deploy_iso/
```

6. Make a folder to house the ramdisk that will need to be extracted from the ISO image.

```
mkdir /tmp/new_deploy_ramdisk
```

7. Change your working directory to `/tmp/new_deploy_ramdisk`

```
cd /tmp/new_deploy_ramdisk
```

8. Unpack the ramdisk image.

```
zcat /tmp/new_deploy_iso/initrd | cpio --extract --make-directories
```

9. Append your CA certificate to the file located at `/tmp/new_deploy_ramdisk/usr/local/lib/python2.7/dist-packages/requests/cacert.pem`, example below:

```
cat your_ca_certificate.pem >> /tmp/new_deploy_ramdisk/usr/local/lib/python2.7/d
```

10. Replace the pre-existing initial ramdisk in the extracted disk image.

```
find . | cpio --create --format='newc' | gzip -c -9 > /tmp/new_deploy_iso/initrd
```

11. Change directory back to the new deployment ISO image folder.

```
cd /tmp/new_deploy_iso
```

12. Ensure that the program `xorriso` is available, you may need to execute `apt-get install xorriso` to install it.

13. Create the new ISO deployment image utilizing the following command.

```
xorriso -as mkisofs -b isolinux/isolinux.bin -c boot.cat -V VMEDIA_BOOT_ISO -r -
```

Once completed, the new image can be found at `/tmp/new_ironic_deploy.iso`, and will need to be uploaded to Glance. New Glance IDs will need to be recorded for any instances requiring this new image, as noted in the parent paragraph

## Ironic in Multiple Control Plane

HPE Helion OpenStack 5.0 introduces the concept of multiple control planes and multiple regions - see the Input Model documentation for the relevant section called “Control Planes and Regions” and the

section called “Multiple Control Planes”. This document covers the use of an Ironic region in a multiple control plane cloud model in HPE Helion OpenStack.

## Networking for Baremetal in Multiple Control Plane

**IRONIC-FLAT-NET** is the network configuration for baremetal control plane.

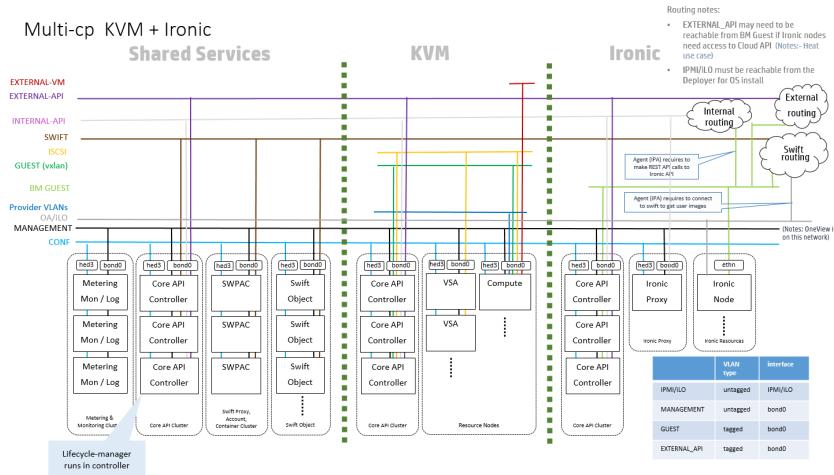
You need to set the environment variable **OS\_REGION\_NAME** to the ironic region in baremetal control plane. This will setup the ironic flat networking in neutron.

```
export OS_REGION_NAME=<ironic_region>
```

To see details of the IRONIC-FLAT-NETWORK created during configuration, use the following command:

```
neutron net-list
```

**Figure 13.1. Architecture of Multiple Control Plane with Ironic**



## Handling Optional Swift Service

Swift is very resource-intensive and as a result, it is now optional in the HPE Helion OpenStack control plane. A number of services depend on Swift, and if it is not present, they must provide a fallback strategy. For example, Glance can use the filesystem in place of Swift for its backend store.

In Ironic, agent-based drivers require Swift - if it is not present, it is necessary to disable access to this Ironic feature in the control plane. The `enable_agent_driver` flag has been added to the ironic configuration data and can have values of `true` or `false`. Setting this flag to `false` will disable swift configurations and the agent based drivers in the ironic control plane.

## Instance Provisioning

In a multiple control plane cloud setup, changes for glance container name in the swift namespace of `ironic-conductor.conf` introduces a conflict with the one in `glance-api.conf`. Provisioning with agent based drivers requires the container name to be the same in ironic and glance. Hence, on instance provisioning with agent based drivers (swift enabled), the agent is not able to fetch the images from glance store and fails at that point.

You can resolve this issue using the following steps:

1. Copy the value of `swift_store_container` from the file `/opt/stack/service/glance-api/etc/glance-api.conf`
2. Login to the lifecycle manager and use the value for `swift_container` in glance namespace of `~/scratch/ansible/next/hos/ansible/roles/ironic-common/templates/ironic-conductor.conf.j2`
3. Run the following playbook:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

## Provisioning Baremetal Nodes with Flat Network Model

### Important

The steps outlined in **must** be performed.

A number of drivers are available to provision and manage bare-metal machines. The drivers are named based on the deployment mode and the power management interface. HPE Helion OpenStack has been tested with the following drivers:

- `agent_ilo`
- `agent_ipmi`
- `pxe_ilo`
- `pxe_ipmi`

For HPE servers, `agent_ipmitool` and `agent_ilo` drivers are supported.

Before you start, you should be aware that:

1. Node Cleaning is enabled for all the drivers in HPE Helion OpenStack 5.0.
2. Node parameter settings must have matching flavors in terms of `cpus`, `local_gb`, and `memory_mb`,`boot_mode` and `cpu_arch`.
3. It is advisable that nodes enrolled for ipmitool drivers are pre-validated in terms of bios settings, in terms of boot mode, prior to setting capabilities.
4. Network cabling and interface layout should also be pre-validated in any given particular boot mode or configuration that is registered.
5. The use of `agent_` drivers is predicated upon Glance images being backed by a Swift image store, specifically the need for the temporary file access features. Use of Ceph or the filesystem as a Glance back-end image store means that the `agent_` drivers cannot be used.
6. Manual Cleaning (RAID) and Node inspection is supported by ilo drivers (`agent_ilo` and `pxe_ilo`)

## Supplied Images

As part of the Helion Entry-scale Ironic Cloud installation, Ironic Python Agent (IPA) images are supplied and loaded into Glance. To see the images that have been loaded, execute the following commands on the deployer node:

```
$ source ~/service.osrc
glance image-list
```

This will produce output like the following example, showing three images that have been added by Ironic:

ID	Name
b9499494-7db3-4448-b67f-233b86489clf	ir-deploy-iso-HOS4.0
8bee92b7-98ae-4242-b80e-1201a475361a	ir-deploy-kernel-HOS4.0
0c889803-469d-4aad-8cf6-3501e39c532c	ir-deploy-ramdisk-HOS4.0

The `ir-deploy-ramdisk-HOS4.0` image is a traditional boot ramdisk used by the `agent_ipmi-tool`, `pxe_ipmitool`, and `pxe_ililo` drivers while `ir-deploy-iso-HOS4.0` is an ISO image that is supplied as virtual media to the host when using the `agent_ilolo` driver.

## Provisioning a node

The information required to provision a node varies slightly depending on the driver used. In general the following details are required.

- Network access information and credentials to connect to the management interface of the node.
- Sufficient properties to allow for nova flavor matching.
- A deployment image to perform the actual deployment of the guest operating system to the baremetal node.

A combination of the `ironic node-create` and `ironic node-update` commands are used for registering a node's characteristics with the Ironic service. In particular, `ironic node-update <nodeid> add` and `ironic node-update <nodeid> replace` can be used to modify the properties of a node after it has been created while `ironic node-update <nodeid> remove` will remove a property.

## Creating a node using agent\_ilolo

If you want to use a boot mode of BIOS as opposed to UEFI, then you need to ensure that the boot mode has been set correctly on the iLO:

While the iLO driver can automatically set a node to boot in UEFI mode via the `boot_mode` defined capability, it cannot set BIOS boot mode once UEFI mode has been set.

Use the `ironic node-create` command to specify the `agent_ilolo` driver, network access and credential information for the iLO, properties of the node and the Glance ID of the supplied ISO IPA image. Note that memory size is specified in megabytes while disk size is gigabytes.

```
ironic node-create -d agent_ilolo -i ilo_address=10.1.196.117 -i ilo_username=Admini
```

```
-p cpus=2 -p cpu_arch=x86_64 -p memory_mb=64000 -p local_gb=99 \
-iilo_deploy_iso=b9499494-7db3-4448-b67f-233b86489clf
```

This will generate output similar to the following:

Property	Value
uuid	<b>ea7246fd-e1d6-4637-9699-0b7c59c22e67</b>
driver_info	{u'ilo_address': u'10.1.196.117', u'ilo_password': u'*****', u'ilo_deploy_iso': u'b9499494-7db3-4448-b67f-233b86489clf', u'ilo_username': u'Administrator'}
extra	{}
driver	agent_ilo
chassis_uuid	
properties	{u'memory_mb': 64000, u'local_gb': 99, u'cpus': 2, u'cpu_arch': u'x86_64'}
name	None

Now update the node with boot\_mode and boot\_option properties:

```
ironic node-update ea7246fd-e1d6-4637-9699-0b7c59c22e67 add properties/capabilitie
```

The `ironic node-update` command returns details for all the node's characteristics.

Property	Value
target_power_state	None
extra	{}
last_error	None
updated_at	None
maintenance_reason	None
provision_state	available
clean_step	{}
uuid	ea7246fd-e1d6-4637-9699-0b7c59c22e67
console_enabled	False
target_provision_state	None
provision_updated_at	None
maintenance	False
inspection_started_at	None
inspection_finished_at	None
power_state	None
driver	agent_ilo
reservation	None
properties	{u'memory_mb': 64000, u'cpu_arch': u'x86_64', u'local_gb': 99, u'capabilities': u'boot_mode:bios,boot_option:local'}
instance_uuid	None
name	None
driver_info	{u'ilo_address': u'10.1.196.117', u'ilo_password': u'*****'}

```

|                               | u'ilo_deploy_iso': u'b9499494-7db3-4448-b67f-233b86489c
|                               | u'ilo_username': u'Administrator'}
| created_at                  | 2016-03-11T10:17:10+00:00
| driver_internal_info        | {}
| chassis_uuid                |
| instance_info               | {}
+

```

## Creating a node using agent\_ipmi

Use the `ironic node-create` command to specify the `agent_ipmitool` driver, network access and credential information for the iLO, properties of the node and the Glance IDs of the supplied kernel and ramdisk images. Note that memory size is specified in megabytes while disk size is gigabytes.

```

ironic node-create -d agent_ipmitool \
-i ipmi_address=10.1.196.117 -i ipmi_username=Administrator -i ipmi_password=***** \
-p cpus=2 -p memory_mb=64000 -p local_gb=99 -p cpu_arch=x86_64 \
-i deploy_kernel=8bee92b7-98ae-4242-b80e-1201a475361a \
-i deploy_ramdisk=0c889803-469d-4aad-8cf6-3501e39c532c

```

This will generate output similar to the following:

```

+-----+-----+
| Property      | Value
+-----+-----+
| uuid          | cf25b19d-098d-406c-b7a1-e0c40cc84a12
| driver_info   | {u'deploy_kernel': u'8bee92b7-98ae-4242-b80e-1201a475361a',
|                 | u'ipmi_address': u'10.1.196.117', u'ipmi_username': u'Administrator',
|                 | u'ipmi_password': u'*****', u'deploy_ramdisk': u'0c889803-469d-4
|                 | 8cf6-3501e39c532c'}
| extra         | {}
| driver        | agent_ipmitool
| chassis_uuid  |
| properties    | {u'memory_mb': 64000, u'cpu_arch': u'x86_64', u'local_gb': 99, u'
|                 | 2}
| name          | None
+-----+-----+

```

Now update the node with `boot_mode` and `boot_option` properties:

```
ironic node-update cf25b19d-098d-406c-b7a1-e0c40cc84a12 add properties/capabiliti
```

The `ironic node-update` command returns details for all the node's characteristics.

```

+-----+-----+
| Property          | Value
+-----+-----+
| target_power_state | None
| extra              | {}
| last_error         | None
| updated_at         | None
| maintenance_reason | None
+-----+-----+

```

provision_state	available
clean_step	{}
uuid	cf25b19d-098d-406c-b7a1-e0c40cc84a12
console_enabled	False
target_provision_state	None
provision_updated_at	None
maintenance	False
inspection_started_at	None
inspection_finished_at	None
power_state	None
driver	agent_ipmitool
reservation	None
properties	{'memory_mb': 64000, 'cpu_arch': 'x86_64', 'local_gb': 2, 'capabilities': 'boot_mode:bios,boot_option:local'}
instance_uuid	None
name	None
driver_info	{'ipmi_password': '*****', 'ipmi_address': '10.1.1.2', 'ipmi_username': 'Administrator', 'deploy_kernel': '8bee92b7-98ae-4242-b80e-1201a475361a', 'deploy_ramdisk': '-469d-4aad-8cf6-3501e39c532c'}
created_at	2016-03-11T14:19:18+00:00
driver_internal_info	{}
chassis_uuid	{}
instance_info	{}

For more information on node enrollment, see the OpenStack documentation at <http://docs.openstack.org/developer/ironic/deploy/install-guide.html#enrollment>.

## Create Flavor

Nova uses flavors when fulfilling requests for bare metal nodes. The Nova scheduler attempts to match the requested flavor against the properties of the created Ironic nodes. So an administrator needs to set up flavors that correspond to the available bare metal nodes using the `nova flavor-create` command.

```
nova flavor-create bmtest auto 64000 99 2
```

ID	Name	Memory_MB	Disk	Ephemeral	Storage
645de08d-2bc6-43f1-8a5f-2315a75b1348	bmtest	64000	99	0	

To see a list of all the available flavors, run `nova flavor-list`

```
nova flavor-list
```

ID	Name	Memory_MB	Disk	Ephemeral	Storage
1	m1.tiny	512	1	0	
2	m1.small	2048	20	0	
3	m1.medium	4096	40	0	

4	m1.large	8192	80	0
5	m1.xlarge	16384	160	0
6	m1.baremetal	4096	80	0
645de08d-2bc6-43f1-8a5f-2315a75b1348	bmtest	64000	99	0

Now set the cpu architecture and boot mode and boot option capabilities:

```
nova flavor-key 645de08d-2bc6-43f1-8a5f-2315a75b1348 set cpu_arch=x86_64
nova flavor-key 645de08d-2bc6-43f1-8a5f-2315a75b1348 set capabilities:boot_option=
nova flavor-key 645de08d-2bc6-43f1-8a5f-2315a75b1348 set capabilities:boot_mode="b
```

For more information on flavor creation, see the OpenStack documentation at <http://docs.openstack.org/developer/ironic/deploy/install-guide.html#flavor-creation>.

## Create network port

Register the MAC addresses of all connected physical network interfaces intended for use with the baremetal node.

```
ironic port-create -a 5c:b9:01:88:f0:a4 -n ea7246fd-e1d6-4637-9699-0b7c59c22e67
```

## Create Glance Image

You can create a whole disk image with `diskimage-builder` [<https://github.com/openstack/diskimage-builder>] using a command similar to the following:

```
DIB_RELEASE=trusty disk-image-create -o ubuntu-trusty.qcow2 vm ubuntu
```

Then load the image into glance:

```
glance image-create --name='ubuntu' --disk-format=qcow2 --container-format=bare --
```

Property	Value
checksum	45a4a06997e64f7120795c68beeb0e3c
container_format	bare
created_at	2016-02-17T10:42:14Z
disk_format	qcow2
id	<b>17e4915a-ada0-4b95-bacf-ba67133f39a7</b>
min_disk	0
min_ram	0
name	ubuntu
owner	821b7bb8148f439191d108764301af64
protected	False
size	372047872
status	active
tags	[ ]
updated_at	2016-02-17T10:42:23Z
virtual_size	None
visibility	private

This image will be used subsequently to boot the baremetal node.

## Generate Key Pair

Create a keypair that you will use when you login to the newly booted node.

```
nova keypair-add ironic_kp > ironic_kp.pem
```

## Determine the neuron network ID

```
neutron net-list
```

id	name	subnets
<b>c010267c-9424-45be-8c05-99d68531ca8c</b>	flat-net	709ee2a1-4110-4b26-ba4d-deb745

## Boot the node

Before booting, it is advisable to power down the node:

```
ironic node-set-power-state ea7246fd-e1d6-4637-9699-0b7c59c22e67 off
```

You can now boot the baremetal node with the information complied in the preceding steps, using the neutron network id, the whole disk image id, the matching flavor and the key name.

```
nova boot --nic net-id=c010267c-9424-45be-8c05-99d68531ca8c \
--image 17e4915a-ada0-4b95-bacf-ba67133f39a7 \
--flavor 645de08d-2bc6-43f1-8a5f-2315a75b1348 \
--key-name ironic_kp ubuntu
```

This command returns information about the state of the node that is booting.

Property	Value
OS-EXT-AZ:availability_zone	-
OS-EXT-SRV-ATTR:host	-
OS-EXT-SRV-ATTR:hypervisor_hostname	-
OS-EXT-SRV-ATTR:instance_name	instance-00000001
OS-EXT-STS:power_state	0
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	-
OS-SRV-USG:terminated_at	-
accessIPv4	
accessIPv6	
adminPass	adpHw3KKTjHk
config_drive	

```

| created                         | 2016-03-11T11:00:28Z
| flavor                          | bmtest (645de08d-2bc6-43f1-8a5f-2315a75b1
| hostId                          |
| id                             | a90122ce-bba8-496f-92a0-8a7cb143007e
| image                           |
| key_name                        |
| metadata                         |
| name                            |
| os-extended-volumes:volumes_attached |
| progress                         |
| security_groups                  |
| status                           |
| tenant_id                        |
| updated                          |
| user_id                          |
+-----+

```

The boot procedure can typically take up to 10 minutes to complete. You can monitor the progress with the iLO console and by using the `nova list`, `nova show <nova_node_id>` and `ironic node-show <ironic_node_id>` commands.

```
nova list
```

ID	Name	Status	Task State	Power State
a90122ce-bba8-496f-92a0-8a7cb143007e	ubuntu	BUILD	spawning	NOSTATE

During the boot procedure, a login prompt will appear for Linux for HPE Helion:

Just ignore this login and wait for the login screen of your target Operating System to appear:

If you now run the `nova list` command, it should show the node in the running state:

```
nova list
```

ID	Name	Status	Task State	Power State
a90122ce-bba8-496f-92a0-8a7cb143007e	ubuntu	ACTIVE	-	Running

You can now log in to the booted node using the key you generated earlier. (You may be prompted to change the permissions on your private key files so that they are not accessible by others).

```
ssh ubuntu@192.3.15.14 -i ironic_kp.pem
```

## Create Glance Images using `diskimage-builder` (DIB)

The following sections show you how to create your own images using the OpenStack `diskimage-builder` tool - for information on how to install the tool see [http://docs.openstack.org/developer/diskimage-builder/user\\_guide/installation.html](http://docs.openstack.org/developer/diskimage-builder/user_guide/installation.html).

## Creating BIOS images for RHEL

1. Install dib-utils:

```
pip install dib-utils
```

2. Download the qcow2 guest image from RHEL portal and define DIB\_LOCAL\_IMAGE:

```
export DIB_LOCAL_IMAGE=/home/stack/rhel-guest-image-7.2-20151102.0.x86_64.qcow2
```

3. Create a local yum repo and define DIB\_YUM\_REPO\_CONF to be the path to the repo:

```
export DIB_YUM_REPO_CONF = <<path_to_repo_file>>
```

4. Use disk-image-create to create the image:

```
cd diskimage-builder/bin
./disk-image-create -o rhel.qcow2 vm dhcp-all-interfaces rhe17
```

## Creating BIOS images for Ubuntu

```
DIB_RELEASE=trusty ./disk-image-create -o ubuntu.qcow2 vm dhcp-all-interfaces ubu
```

## Creating UEFI images

To create UEFI images, you will need to setup sudo to not prompt for passwords and configure web-proxy. Once you have sudo and web-proxy configured, follow these steps:

1. Clone the diskimage-builder git repository.

```
git clone https://git.gozer.hpccloud.net/openstack/diskimage-builder.git -b feature/uefi
```

2. Apply the patch to your cloned diskimage-builder repository (<https://review.openstack.org/#/c/391850/6>) and then clone the diskimage-builder utilities (dib-utils) repository.

```
cd diskimage-builder/
git fetch git://git.openstack.org/openstack/diskimage-builder refs/changes/50/391850/6
cd ..
git clone https://git.gozer.hpccloud.net/openstack/dib-utils.git
```

3. Add execute permissions for disk-image-create.

```
chmod a+x /root/diskimage-builder/diskimage_builder/lib/disk-image-create
```

4. Export your path to include diskimage-builder utilities.

```
export PATH=/root/diskimage-builder/diskimage_builder/lib:/root/dib-utils/bin:$PATH
```

5. Verify that the diskimage-build utilities are in your cloned path by locating dib-run-parts.

```
which dib-run-parts
/root/dib-utils/bin/dib-run-parts
```

6. Verify that the diskimage-build is in your cloned path by locating disk-image-create.

```
which disk-image-create
/root/diskimage-builder/diskimage_builder/lib/disk-image-create
```

7. Install diskimage-builder.

```
cd diskimage-builder/
python setup.py install
```

8. Export the diskimage-builder lib files.

```
export _LIB=/root/diskimage-builder/diskimage_builder/lib
```

9. Create the UEFI Image.

- To create an Ubuntu UEFI image:

```
DIB_RELEASE=trusty DIB_UFBI_SUPPORT=true disk-image-create -o ubuntu-trusty-uefi.qcow2 vm ubuntu uefi
```

- To create a Fedora UEFI image:

```
DIB_UFBI_SUPPORT=true disk-image-create -o fedora-uefi.qcow2 vm fedora dhcp-all-in-one
```

10. For OS support, refer to supported\_distribution-.rst [https://github.com/openstack/diskimage-builder/blob/master/doc/source/user\_guide/supported\_distros.rst] in diskimage-builder user documentation.

## Provisioning Baremetal Nodes with Multi-Tenancy

1. Create a network and a subnet:

```
$ neutron net-create guest-net-1
Created a new network:
+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | True
| availability_zone_hints |
| availability_zones |
| created_at | 2017-06-10T02:49:56Z
| description |
| id | 256d55a6-9430-4f49-8a4c-cc5192f5321e
| ipv4_address_scope |
| ipv6_address_scope |
| mtu | 1500
| name | guest-net-1
| project_id | 57b792cdccc74d16a08fc7a396ee05b6
| provider:network_type | vlan
| provider:physical_network | physnet1
| provider:segmentation_id | 1152
| revision_number | 2
| router:external | False
```

```

| shared           | False
| status          | ACTIVE
| subnets         |
| tags            |
| tenant_id       | 57b792cdcd74d16a08fc7a396ee05b6
| updated_at      | 2017-06-10T02:49:57Z
+-----+-----+
$ neutron subnet-create guest-net-1 200.0.0.0/24
Created a new subnet:
+-----+-----+
| Field          | Value
+-----+-----+
| allocation_pools | {"start": "200.0.0.2", "end": "200.0.0.254"}
| cidr           | 200.0.0.0/24
| created_at     | 2017-06-10T02:53:08Z
| description     |
| dns_nameservers |
| enable_dhcp    | True
| gateway_ip     | 200.0.0.1
| host_routes    |
| id              | 53accf35-ae02-43ae-95d8-7b5efed18ae9
| ip_version      | 4
| ipv6_address_mode |
| ipv6_ra_mode    |
| name            |
| network_id      | 256d55a6-9430-4f49-8a4c-cc5192f5321e
| project_id      | 57b792cdcd74d16a08fc7a396ee05b6
| revision_number | 2
| service_types   |
| subnetpool_id   |
| tenant_id       | 57b792cdcd74d16a08fc7a396ee05b6
| updated_at      | 2017-06-10T02:53:08Z
+-----+-----+

```

## 2. Review glance image list

```

$ glance image-list
+-----+-----+
| ID          | Name
+-----+-----+
| 73205fb7-64f1-4243-8d0f-a0dd2e493c9d | cirros-0.3.4-x86_64
| 83eecf9c-d675-4bf9-a5d5-9cf1fe9ee9c2 | ir-deploy-iso-HOS5.0
| db3d131f-2fb0-4189-bb8d-424ee0886e4c | ir-deploy-kernel-HOS5.0
| 304cae15-3fe5-4f1c-8478-c65da5092a2c | ir-deploy-ramdisk-HOS5.0
+-----+-----+

```

## 3. Create Ironic node

```

$ ironic --ironic-api-version 1.22 node-create -d agent_ipmitool \
-n test-node-1 -i ipmi_address=192.168.9.69 -i ipmi_username=ilo_user \
-i ipmi_password=XXXXXXXX --network-interface neutron -p memory_mb=4096 \
-p cpu_arch=x86_64 -p local_gb=80 -p cpus=2 \
-p capabilities=boot_mode:bios,boot_option:local \

```

```

-p root_device='{"name": "/dev/sda"}' \
-i deploy_kernel=db3d131f-2fb0-4189-bb8d-424ee0886e4c \
-i deploy_ramdisk=304cae15-3fe5-4f1c-8478-c65da5092a2c

+-----+-----+
| Property | Value
+-----+-----+
| chassis_uuid | 
| driver | agent_ipmitool
| driver_info | {"deploy_kernel": "db3d131f-2fb0-4189-bb8d-424ee0886e4c", "ipmi_address": "192.168.9.69", "ipmi_username": "goze", "ipmi_password": "*****", "deploy_ramdisk": "304cae15-3fe5-4f1c-8478-c65da5092a2c"}
| extra | {}
| name | test-node-1
| network_interface | neutron
| properties | {"cpu_arch": "x86_64", "root_device": {"name": "/dev/sda"}, "cpus": 2, "capabilities": "boot_mode:bios,boot_option:uefi", "memory_mb": 4096, "local_gb": 80}
| resource_class | None
| uuid | cb4dda0d-f3b0-48b9-ac90-ba77b8c66162
+-----+-----+

```

ipmi\_address, ipmi\_username and ipmi\_password are iLo access parameters for baremetal Ironic node. Adjust memory\_mb, cpus, local\_gb to your node size requirements. They also need to be reflected in flavor setting (see below). Use capabilities boot\_mode:bios for baremetal nodes operating in Legacy BIOS mode. For UEFI baremetal nodes, use boot\_mode:uefi lookup deploy\_kernel and deploy\_ramdisk in glance image list output above.

## Important

Since we are using Ironic API version 1.22, node is created initial state **enroll**. It needs to be explicitly moved to **available** state. This behavior changed in API version 1.11

### 4. Create port

```

$ ironic --ironic-api-version 1.22 port-create --address f0:92:1c:05:6c:40 --node cb4dda0d-f3b0-48b9-ac90-ba77b8c66162
+-----+-----+
| Property | Value
+-----+-----+
| address | f0:92:1c:05:6c:40
| extra | {}
| local_link_connection | {"switch_info": "hp59srv1-a-11b", "port_id": "Ten-GigabitEthernet3/4", "switch_id": "e8:f7:24:bf:07:2e"}
| node_uuid | cb4dda0d-f3b0-48b9-ac90-ba77b8c66162
| pxe_enabled | True
| uuid | a49491f3-5595-413b-b4a7-bb6f9abec212
+-----+-----+

```

- for --address, use MAC of 1st NIC of ironic baremetal node, which will be used for PXE boot
- for --node, use ironic node uuid (see above)
- for -l switch\_id, use switch management interface MAC address. It can be retrieved by pinging switch management IP and looking up MAC address in 'arp -l -n' command output.

- for -l switch\_info, use switch\_id from data/ironic/ironic\_config.yml file. If you have several switch config definitions, use the right switch your baremetal node is connected to.
- for -l port\_id, use port ID on the switch

5. Move ironic node to manage and then available state

```
$ ironic node-set-provision-state test-node-1 manage
$ ironic node-set-provision-state test-node-1 provide
```

6. Once node is successfully moved to available state, it's resources should be included into Nova hypervisor statistics

```
$ nova hypervisor-stats
+-----+-----+
| Property | Value |
+-----+-----+
| count | 1 |
| current_workload | 0 |
| disk_available_least | 80 |
| free_disk_gb | 80 |
| free_ram_mb | 4096 |
| local_gb | 80 |
| local_gb_used | 0 |
| memory_mb | 4096 |
| memory_mb_used | 0 |
| running_vms | 0 |
| vcpus | 2 |
| vcpus_used | 0 |
+-----+-----+
```

7. Prepare a keypair, which will be used for logging into the node

```
$ nova keypair-add ironic_kp > ironic_kp.pem
```

8. Obtain user image and upload it to glance. Please refer to OpenStack documentation on user image creation: <https://docs.openstack.org/project-install-guide/baremetal/draft/configure-glance-images.html>.

## Note

Deployed images are already populated by HPE Helion OpenStack installer.

```
$ glance image-create --name='Ubuntu Trusty 14.04' --disk-format=qcow2 --container-format=bare
+-----+-----+
| Property | Value |
+-----+-----+
| checksum | d586d8d2107f328665760fee4c81caf0 |
| container_format | bare |
| created_at | 2017-06-13T22:38:45Z |
| disk_format | qcow2 |
| id | 9fdd54a3-ccf5-459c-a084-e50071d0aa39 |
| min_disk | 0 |
| min_ram | 0 |
| name | Ubuntu Trusty 16.04 |
| owner | 57b792cdcd74d16a08fc7a396ee05b6 |
```

```

| protected      | False
| size          | 371508736
| status         | active
| tags          | []
| updated_at    | 2017-06-13T22:38:55Z
| virtual_size   | None
| visibility     | private
+-----+

```

```

$ glance image-list
+-----+-----+
| ID           | Name
+-----+-----+
| 73205fb7-64f1-4243-8d0f-a0dd2e493c9d | cirros-0.3.4-x86_64
| 83ee9c-d675-4bf9-a5d5-9cf1fe9ee9c2 | ir-deploy-iso-HOS5.0
| db3d131f-2fb0-4189-bb8d-424ee0886e4c | ir-deploy-kernel-HOS5.0
| 304cae15-3fe5-4f1c-8478-c65da5092a2c | ir-deploy-ramdisk-HOS5.0
| 9fdd54a3-ccf5-459c-a084-e50071d0aa39 | Ubuntu Trusty 14.04
+-----+

```

9. Create a baremetal flavor and set flavor keys specifying requested node size, architecture and boot mode. A flavor can be re-used for several nodes having the same size, architecture and boot mode

```
$ nova flavor-create m1.ironic auto 4096 80 2
```

```
+-----+-----+-----+-----+
| ID           | Name       | Memory_MB | Disk | Ephemera
+-----+-----+-----+-----+
| ab69881b-0d5a-497d-93ae-6e28694c87bf | m1.ironic | 4096      | 80   | 0
+-----+-----+-----+-----+
```

```
$ nova flavor-key ab69881b-0d5a-497d-93ae-6e28694c87bf set cpu_arch=x86_64
$ nova flavor-key ab69881b-0d5a-497d-93ae-6e28694c87bf set capabilities:boot_opt
$ nova flavor-key ab69881b-0d5a-497d-93ae-6e28694c87bf set capabilities:boot_mod
```

Parameters must match parameters of ironic node above. Use `capabilities:boot_mod=e="bios"` for Legacy BIOS nodes. For UEFI nodes, use `capabilities:boot_mode="uefi"`

10. Boot the node

```
$ nova boot --flavor m1.ironic --image 9fdd54a3-ccf5-459c-a084-e50071d0aa39 --ke
+-----+-----+
| Property          | Value
+-----+-----+
| OS-DCF:diskConfig | MANUAL
| OS-EXT-AZ:availability_zone |
| OS-EXT-SRV-ATTR:host | -
| OS-EXT-SRV-ATTR:hypervisor_hostname | -
| OS-EXT-SRV-ATTR:instance_name |
| OS-EXT-STS:power_state | 0
| OS-EXT-STS:task_state | scheduling
| OS-EXT-STS:vm_state | building
| OS-SRV-USG:launched_at |
| OS-SRV-USG:terminated_at | -
| accessIPv4 |
| accessIPv6 |
+-----+
```

adminPass	XXXXXXXXXXXXXX
config_drive	
created	2017-06-14T21:25:18Z
flavor	m1.ironic (ab69881b-0d5a-497d-93ae-6e28
hostId	
id	f1a8c63e-da7b-4d9a-8648-b1baa6929682
image	Ubuntu Trusty 14.04 (9fdd54a3-ccf5-459c
key_name	ironic_kp
metadata	{}
name	test-node-1
os-extended-volumes:volumes_attached	[]
progress	0
security_groups	default
status	BUILD
tenant_id	57b792cdccc74d16a08fc7a396ee05b6
updated	2017-06-14T21:25:17Z
user_id	cc76d7469658401fdb4cf772278483d9

- for --image, use ID of user image created at previous step
- for --nic net-id, use id of tenant network created at the beginning

### Note

During the node provisioning, the following is happening in the background:

Neutron connects to switch management interfaces and assigns provisioning VLAN to baremetal node port on the switch. Ironic powers up the node using iLo interface. Node is booting IPA image via PXE. IPA image is writing provided user image onto specified root device (/dev/sda) and powers node down. Neutron connects to switch management interfaces and assigns tenant VLAN to baremetal node port on the switch. A VLAN ID is selected from provided range. Ironic powers up the node using iLo interface. Node is booting user image from disk.

11. Once provisioned, node will join the private tenant network. Access to private network from other networks is defined by switch configuration.

## View Ironic System Details

### View details about the server using nova show <no-va-node-id>

```
nova show a90122ce-bba8-496f-92a0-8a7cb143007e
```

Property	Value
OS-EXT-AZ:availability_zone	nova
OS-EXT-SRV-ATTR:host	helion-cpl-ir-compute0001-mgmt
OS-EXT-SRV-ATTR:hypervisor_hostname	ea7246fd-e1d6-4637-9699-0b7c59c22e67
OS-EXT-SRV-ATTR:instance_name	instance-0000000a

OS-EXT-STS:power_state	1
OS-EXT-STS:task_state	-
OS-EXT-STS:vm_state	active
OS-SRV-USG:launched_at	2016-03-11T12:26:25.000000
OS-SRV-USG:terminated_at	-
accessIPv4	
accessIPv6	
config_drive	
created	2016-03-11T12:17:54Z
flat-net network	192.3.15.14
flavor	bmtest (645de08d-2bc6-43f1-8a5f-2315a75b1
hostId	ecafa4f40eb5f72f7298de0ef51eb7769e3bad47c
id	a90122ce-bba8-496f-92a0-8a7cb143007e
image	ubuntu (17e4915a-ada0-4b95-bacf-ba67133f3
key_name	ironic_kp
metadata	{}
name	ubuntu
os-extended-volumes:volumes_attached	[ ]
progress	0
security_groups	default
status	ACTIVE
tenant_id	d53bcaf15afb4cb5aea3adaedbaa60dd
updated	2016-03-11T12:26:26Z
user_id	e580c645bfec4faeade7dbd24aaf990

## View detailed information about a node using ironic node-show <ironic-node-id>

```
ironic node-show ea7246fd-e1d6-4637-9699-0b7c59c22e67
```

Property	Value
target_power_state	None
extra	{}
last_error	None
updated_at	2016-03-11T12:26:25+00:00
maintenance_reason	None
provision_state	active
clean_step	{}
uuid	ea7246fd-e1d6-4637-9699-0b7c59c22e67
console_enabled	False
target_provision_state	None
provision_updated_at	2016-03-11T12:26:25+00:00
maintenance	False
inspection_started_at	None
inspection_finished_at	None
power_state	power on
driver	agent_ilo
reservation	None
properties	{u'memory_mb': 64000, u'cpu_arch': u'x86_64', u'local_g

```

| instance_uuid          | 2, u'capabilities': u'boot_mode:bios,boot_option:local'
| name                  | a90122ce-bba8-496f-92a0-8a7cb143007e
| driver_info            | None
|                       | {u'ilo_address': u'10.1.196.117', u'ilo_password': u'**'}
| created_at             | u'ilo_deploy_iso': u'b9499494-7db3-4448-b67f-233b86489c
| driver_internal_info   | u'ilo_username': u'Administrator'}
|                       | 2016-03-11T10:17:10+00:00
| chassis_uuid           | {u'agent_url': u'http://192.3.15.14:9999', u'is_whole_d
| instance_info           | u'agent_last_heartbeat': 1457699159}

| instance_uuid          | {u'root_gb': u'99', u'display_name': u'ubuntu', u'image'
| name                  | '17e4915a-ada0-4b95-bacf-ba67133f39a7', u'capabilities'
| driver_info            | "bios", "boot_option": "local"}', u'memory_mb': u'64000
|                       | u'2', u'image_url': u'http://192.168.12.2:8080/v1/AUTH_
| created_at             | 50cc4999a10d58d/glance/17e4915a-ada0-4b95-bacf-ba67133f
| driver_internal_info   | =ada691726337805981ac002c0fbfc905eb9783ea&temp_url_expi
| chassis_uuid           | u'image_container_format': u'bare', u'local_gb': u'99',
| instance_info           | u'image_disk_format': u'qcow2', u'image_checksum':
|                       | u'2d7bb1e78b26f32c50bd9da99102150b', u'swap_mb': u'0'}
+
+-----+

```

## View detailed information about a port using ironic port-show <ironic-port-id>

```
ironic port-show a17a4ef8-a711-40e2-aa27-2189c43f0b67
```

Property	Value
node_uuid	ea7246fd-e1d6-4637-9699-0b7c59c22e67
uuid	a17a4ef8-a711-40e2-aa27-2189c43f0b67
extra	{u'vif_port_id': u'82a5ab28-76a8-4c9d-bfb4-624aeb9721ea'}
created_at	2016-03-11T10:40:53+00:00
updated_at	2016-03-11T12:17:56+00:00
address	5c:b9:01:88:f0:a4

## View detailed information about a hypervisor using nova hypervisor-list and nova hypervisor-show

```
nova hypervisor-list
```

ID	Hypervisor hostname	State	Status
541	ea7246fd-e1d6-4637-9699-0b7c59c22e67	up	enabled

```
nova hypervisor-show ea7246fd-e1d6-4637-9699-0b7c59c22e67
```

Property	Value
cpu_info	
current_workload	0
disk_available_least	0
free_disk_gb	0
free_ram_mb	0
host_ip	192.168.12.6
hypervisor_hostname	ea7246fd-e1d6-4637-9699-0b7c59c22e67
hypervisor_type	ironic
hypervisor_version	1
id	541
local_gb	99
local_gb_used	99
memory_mb	64000
memory_mb_used	64000
running_vms	1
service_disabled_reason	None
service_host	helion-cpl-ir-compute0001-mgmt
service_id	25
state	up
status	enabled
vcpus	2
vcpus_used	2

## View a list of all running services using nova service-list

```
nova service-list
```

Id	Binary	Host	Zone	Status	St
1	nova-conductor	helion-cpl-c1-m1-mgmt	internal	enabled	up
7	nova-conductor	helion-cpl-c1-m2-mgmt	internal	enabled	up
10	nova-conductor	helion-cpl-c1-m3-mgmt	internal	enabled	up
13	nova-scheduler	helion-cpl-c1-m1-mgmt	internal	enabled	up
16	nova-scheduler	helion-cpl-c1-m3-mgmt	internal	enabled	up
19	nova-scheduler	helion-cpl-c1-m2-mgmt	internal	enabled	up
22	nova-consoleauth	helion-cpl-c1-m1-mgmt	internal	enabled	up
25	nova-compute	helion-cpl-ir-compute0001-mgmt	nova	enabled	up

## Troubleshooting Ironic Installation

Sometimes the `nova boot` command does not succeed and when you do a `nova list`, you will see output like the following:

```
nova list
```

ID	Name	Status	Task State	Power State
ee08f82e-8920-4360-be51-a3f995624e5f	ubuntu	ERROR	-	NOSTATE

You should execute the `nova show <nova-node-id>` and `ironic node-show <ironic-node-id>` commands to get more information about the error.

## No valid host was found. There are not enough hosts available.

This error is typically seen when performing the `nova boot` where there is a mismatch between the properties set on the node and the flavor used. For example, the output from a `nova show` command may look like this:

```
nova show ee08f82e-8920-4360-be51-a3f995624e5f
```

Property	Value
OS-EXT-AZ:availability_zone	-
OS-EXT-SRV-ATTR:host	-
OS-EXT-SRV-ATTR:hypervisor_hostname	-
OS-EXT-SRV-ATTR:instance_name	instance-00000001
OS-EXT-STS:power_state	0
OS-EXT-STS:task_state	-
OS-EXT-STS:vm_state	error
OS-SRV-USG:launched_at	-
OS-SRV-USG:terminated_at	-
accessIPv4	
accessIPv6	
config_drive	
created	2016-03-11T11:00:28Z
fault	{ "message": "No valid host was found. The request_spec, filter_properties) File \"/opt/stack/venv/nova-20160308T00 return func(*args, **kwargs) File \"/opt/stack/venv/nova-20160308T00 context, request_spec, filter_property File \"/opt/stack/venv/nova-20160308T00 return getattr(self.instance, __name) File \"/opt/stack/venv/nova-20160308T00 context, request_spec, filter_property File \"/opt/stack/venv/nova-20160308T00 request_spec=request_spec, filter_pro File \"/opt/stack/venv/nova-20160308T00 retry=self.retry) File \"/opt/stack/venv/nova-20160308T00 timeout=timeout, retry=retry) File \"/opt/stack/venv/nova-20160308T00

```

flavor
hostId
id
image
key_name
metadata
name
os-extended-volumes:volumes_attached
status
tenant_id
updated
user_id
+-----+
retry=retry)
File \"/opt/stack/venv/nova-20160308T00
raise result
", "created": "2016-03-11T11:00:29Z"}
bmtest (645de08d-2bc6-43f1-8a5f-2315a75b1
ee08f82e-8920-4360-be51-a3f995624e5f
ubuntu (17e4915a-ada0-4b95-bacf-ba67133f3
ironic_kp
{}
ubuntu
[]
ERROR
d53bcf15afb4cb5aea3adaedbaa60dd
2016-03-11T11:00:28Z
e580c645bfec4faeade7dbd24aaf990
+-----+

```

You can find more information about the error by inspecting the log file at `/var/log/nova/nova-scheduler.log` or alternatively by viewing the error location in the source files listed in the stack-trace (in bold above).

To find the mismatch, compare the properties of the ironic node:

Property	Value
target_power_state	None
extra	{}
last_error	None
updated_at	None
maintenance_reason	None
provision_state	available
clean_step	{}
uuid	ea7246fd-e1d6-4637-9699-0b7c59c22e67
console_enabled	False
target_provision_state	None
provision_updated_at	None
maintenance	False
inspection_started_at	None
inspection_finished_at	None
power_state	None
driver	agent_ilo
reservation	None
properties	{u'memory_mb': 64000, u'local_gb': 99, u'cpus': 2, u'ca u'boot_mode:bios,boot_option:local'}
instance_uuid	None
name	None
driver_info	{u'ilo_address': u'10.1.196.117', u'ilo_password': u'**' u'ilo_deploy_iso': u'b9499494-7db3-4448-b67f-233b86489c u'ilo_username': u'Administrator'}
created_at	2016-03-11T10:17:10+00:00
driver_internal_info	{}

```

| chassis_uuid
| instance_info
+-----+-----+
| { }

```

with the flavor characteristics:

```
nova flavor-show
```

```

+-----+-----+
| Property | Value
+-----+-----+
| OS-FLV-DISABLED:disabled | False
| OS-FLV-EXT-DATA:ephemeral | 0
| disk | 99
| extra_specs | {"capabilities:boot_option": "local", "cpu_arch": "x86_64", "ram": 64000}
| id | 645de08d-2bc6-43f1-8a5f-2315a75b1348
| name | bmtest
| os-flavor-access:is_public | True
| ram | 64000
| rxtx_factor | 1.0
| swap |
| vcpus | 2

```

In this instance, the problem is caused by the absence of the "cpu\_arch": "x86\_64" property on the ironic node. This can be resolved by updating the ironic node, adding the missing property:

```
ironic node-update ea7246fd-e1d6-4637-9699-0b7c59c22e67 add properties/cpu_arch=x86_64
```

and then re-running the nova boot command.

## Deployment to a node fails and in "ironic node-list" command, the power\_state column for the node is shown as "None"

**Possible cause:** The IPMI commands to the node take longer to change the power state of the server.

**Resolution:** Check if the node power state can be changed using the following command

```
ironic node-set-power-state $NODEUUID on
```

If the above command succeeds and the power\_state column is updated correctly, then the following steps are required to increase the power sync interval time.

On the first controller, reconfigure Ironic to increase the power sync interval time. In the example below, it is set to 120 seconds. This value may have to be tuned based on the setup.

1. Go to the ~/helion/my\_cloud/config/ironic/ directory and edit ironic-conductor.conf.j2 to set the sync\_power\_state\_interval value:

```
[conductor]
```

```
sync_power_state_interval = 120
```

2. Save the file and then run the following playbooks:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
ansible-playbook -i hosts/localhost ready-deployment.yml
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

## Error Downloading Image

If you encounter the error below during the deployment:

```
"u'message': u'Error downloading image: Download of image id 77700b53-9e15-406c-b2
u'code': 500,
u'type': u'ImageDownloadError',
u'details': u'Download of image id 77700b53-9e15-406c-b2d5-13e7d9b96551 failed: Im'
```

you should visit the Single Sign-On Settings in the Security page of iLO and change the Single Sign-On Trust Mode setting from the default of "Trust None (SSO disabled)" to "Trust by Certificate".

## Using node-inspection can cause temporary claim of IP addresses

**Possible cause:** Running node-inspection on a node discovers all the NIC ports including the NICs that don't have any connectivity. This causes a temporary consumption of the network IPs and increased usage of the allocated quota. As a result, other nodes are deprived of IP addresses and deployments can fail.

**Resolution:** You can add node properties manually added instead of using the inspection tool.

Note: Upgrade ipmitool to a version >= 1.8.15 or it may not return detailed information about the NIC interface for node-inspection.

## Node permanently stuck in deploying state

**Possible causes:**

- Ironic conductor service associated with the node could go down.
- There might be a properties mismatch. MAC address registered for the node could be incorrect.

**Resolution:** To recover from this condition, set the provision state of the node to Error and maintenance to True. Delete the node and re-register again.

## The NICs in the baremetal node should come first in boot order

**Possible causes:** By default, the boot order of baremetal node is set as NIC1, HDD and NIC2. If NIC1 fails, the nodes starts booting from HDD and the provisioning fails.

**Resolution:** Set boot order so that all the NICs appear before the hard disk of the baremetal as NIC1, NIC2..., HDD.

## Increase in the number of nodes can cause power commands to fail

**Possible causes:** Ironic periodically performs a power state sync with all the baremetal nodes. When the number of nodes increase, ironic does not get sufficient time to perform power operations.

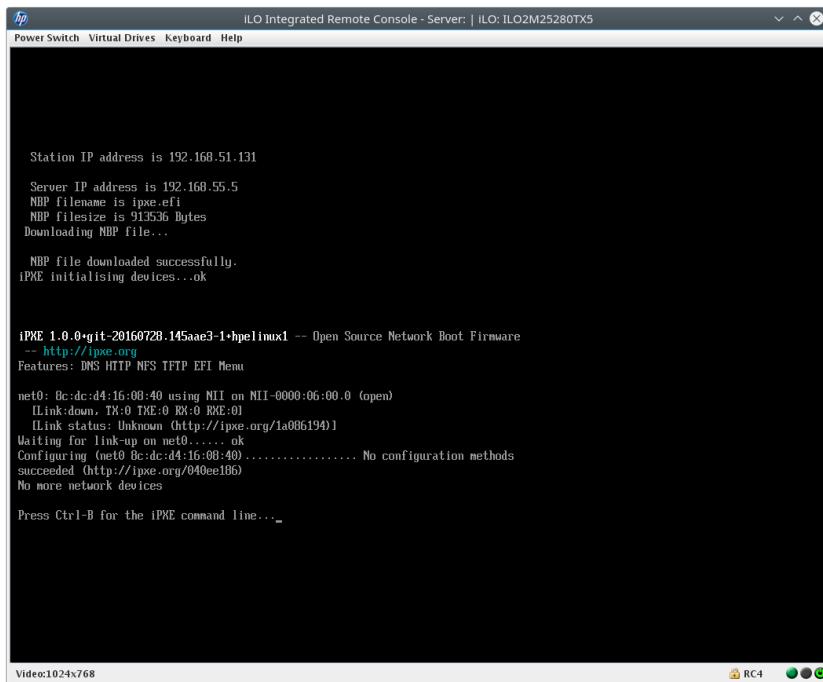
**Resolution:** The following procedure gives a way to increase sync\_power\_state\_interval:

- Edit the file `~/helion/my_cloud/config/ironic/ironic-conductor.conf.j2` and navigate to the section for `[conductor]`
- Increase the `sync_power_state_interval`. For example, for 100 nodes, set `sync_power_state_interval = 90` and save the file.
- Execute the following set of commands to reconfigure Ironic:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
ansible-playbook -i hosts/localhost ready-deployment.yml
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

## DHCP succeeds with PXE but times out with iPXE

If you see DHCP error "No configuration methods succeeded" in iPXE, right after successful DHCP performed by embedded NIC firmware (please refer to screenshot below), there may be an issue with Spanning Tree Protocol on the switch.



To avoid this error, Rapid Spanning Tree Protocol needs to be enabled on the switch. If this is not an option due to conservative loop detection strategies, use the steps outlined below to install the iPXE binary with increased DHCP timeouts.

1. Clone iPXE source code

```
$ git clone git://git.ipxe.org/ipxe.git
$ cd ipxe/src
```

2. Modify lines 22-25 in file config/dhcp.h, which declare reduced DHCP timeouts (1-10 secs). Comment out lines with reduced timeouts and uncomment normal PXE timeouts (4-32)

```
//#define DHCP_DISC_START_TIMEOUT_SEC      1
//#define DHCP_DISC_END_TIMEOUT_SEC        10
#define DHCP_DISC_START_TIMEOUT_SEC      4      /* as per PXE spec */
#define DHCP_DISC_END_TIMEOUT_SEC       32      /* as per PXE spec */
```

3. Make undionly.kpxe (BIOS) and ipxe.efi (UEFI) images

```
$ make bin/undionly.kpxe
$ make bin-x86_64-efi/ipxe.efi
```

4. Copy iPXE images to lifecycle manager

```
$ scp bin/undionly.kpxe bin-x86_64-efi/ipxe.efi stack@10.0.0.4:
stack@10.0.0.4's password:
undionly.kpxe
ipxe.efi
100%   6
100%  91
```

5. From deployer, distribute image files onto all 3 controllers

```
stack@helion-cp1-cl-m1-mgmt:~$ cd ~/scratch/ansible/next/hos/ansible/
stack@helion-cp1-cl-m1-mgmt:~/scratch/ansible/next/hos/ansible$ ansible -i hosts ...
stack@helion-cp1-cl-m1-mgmt:~/scratch/ansible/next/hos/ansible$ ansible -i hosts ...
...
```

There is no need to restart services. With next PXE boot attempt, iPXE binary with the increased timeout will be downloaded to the target node via TFTP.

## Node Cleaning

Cleaning is the process by which data is removed after a previous tenant has used the node. Cleaning requires use of ironic's agent\_drivers. It is extremely important to note that if the pxe\_drivers are utilized, no node cleaning operations will occur, and a previous tenant's data could be found on the node. The same risk of a previous tenant's data possibly can occur if cleaning is explicitly disabled as part of the installation.

By default, cleaning attempts to utilize ATA secure erase to wipe the contents of the disk. If secure erase is unavailable, the cleaning functionality built into the Ironic Python Agent falls back to an operation referred to as "shred" where random data is written over the contents of the disk, and then followed up by writing "0"s across the disk. This can be a time-consuming process.

An additional feature of cleaning is the ability to update firmware or potentially assert new hardware configuration, however, this is an advanced feature that must be built into the Ironic Python Agent image. Due to the complex nature of such operations, and the fact that no one size fits all, this requires a custom

Ironic Python Agent image to be constructed with an appropriate hardware manager. For more information on hardware managers, see <http://docs.openstack.org/developer/ironic-python-agent/#hardware-managers>

Ironic's upstream documentation for cleaning may be found here: <http://docs.openstack.org/developer/ironic/deploy/cleaning.html>

## Setup

Cleaning is enabled by default in ironic when installed via the Lifecycle Manager. You can verify this by examining the `ironic-conductor.conf` file. Look for:

```
[conductor]
clean_nodes=true
```

## In use

When enabled, cleaning will be run automatically when nodes go from active to available state or from manageable to available. To monitor what step of cleaning the node is in, run `ironic node-show`:

```
stack@helion-cp1-c1-m1-mgmt:~$ ironic node-show 4e6d4273-2535-4830-a826-7f67e71783
+-----+-----+
| Property          | Value
+-----+-----+
| target_power_state | None
| extra              | {}
| last_error         | None
| updated_at         | 2016-04-15T09:33:16+00:00
| maintenance_reason | None
| provision_state    | cleaning
| clean_step          | {}
| uuid                | 4e6d4273-2535-4830-a826-7f67e71783ed
| console_enabled     | False
| target_provision_state | available
| provision_updated_at | 2016-04-15T09:33:16+00:00
| maintenance          | False
| inspection_started_at | None
| inspection_finished_at | None
| power_state          | power off
| driver               | agent_ilo
| reservation          | helion-cp1-c1-m1-mgmt
| properties            | {u'memory_mb': 4096, u'cpu_arch': u'amd64', u'local_gb': 1, u'capabilities': u'boot_mode:uefi,boot_option:local'}
| instance_uuid         | None
| name                 | None
| driver_info           | {u'ilo_deploy_iso': u'249bf095-e741-441d-bc28-0f44a9b8c', u'ipmi_username': u'Administrator', u'deploy_kernel': u'3a78c0a9-3d8d-4764-9300-3e9c00e167a1', u'ilo_address': u'10.1.196.113', u'ipmi_address': u'10.1.196.113', u'debug_ip': u'd02c811c-e521-4926-9f26-0c88bbd2ee6d', u'ipmi_password': u'*****', u'ilo_password': u'*****', u'ilo_username': u'Administrator'}
| created_at            | 2016-04-14T08:30:08+00:00
| driver_internal_info | {u'clean_steps': None, u'hardware_manager_version': '1.0'}
```

```

|           | {u'generic_hardware_manager': u'1.0'}, u'is_whole_disk_|
|           | u'agent_erase_devices_iterations': 1, u'agent_url':|
| chassis_uuid | u'http://192.168.246.245:9999', u'agent_last_heartbeat'|
| instance_info | {}|

```

The status will be in the driver\_internal\_info field. You will also be able to see the clean\_steps list there.

## Troubleshooting

If an error occurs during the cleaning process, the node will enter the clean failed state so that it is not deployed. The node remains powered on for debugging purposes. The node can be moved to the manageable state to attempt a fix using the following command:

```
ironic node-set-provision-state <node id> manage
```

Once you have identified and fixed the issue, you can return the node to available state by executing the following commands:

```
ironic node-set-maintenance <node id> false
ironic node-set-provision-state <node id> provide
```

This will retry the cleaning steps and set the node to available state upon their successful completion.

## Disabling Node Cleaning

To disable node cleaning, you need to edit `helion/my_cloud/definition/data/ironic/ironic_config.yml` and set `enable_node_cleaning` to false.

Commit your changes:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "disable node cleaning"
```

Deploy these changes by re-running the configuration processor and reconfigure the ironic installation:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
ansible-playbook -i hosts/localhost ready-deployment.yml
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml
```

## Ironic and OneView

### Enabling Ironic OneView driver in HPE Helion OpenStack

Edit the file `~/helion/my_cloud/definition/data/ironic/ironicconfig.yml` and set the value

```
enable_oneview: true
```

This will enable the OneView driver for Ironic in HPE Helion OpenStack.

## Adding OneView Appliance Credentials

```
manage_url: https://<Onview appliance URL>  
  
oneview_username: "<Appliance username>"  
  
oneview_encrypted_password: "<Encrypted password>"  
  
oneview_allow_insecure_connections: <true/false>  
  
tls_cacert_file: <CA certificate for connection>
```

## Encrypting the OneView Password

Encryption can be applied using `hosencrypt.py` or using `openssl`. On the lifecyclemanager node, export the key used for encryption as environment variable:

```
export HOS_USER_PASSWORD_ENCRYPT_KEY="<key for encryption>"
```

And then execute the following commands:

```
cd ~/helion/hos/ansible  
python hosencrypt.py
```

Enter password to be encrypted when prompted. The script uses the key that was exported in the `HOS_USER_PASSWORD_ENCRYPT_KEY` to do the encryption.

For more information, see the documentation for .

## Decrypting the OneView Password

Before running the `site.yml` playbook, export the key used for encryption as environment variable:

```
export HOS_USER_PASSWORD_ENCRYPT_KEY="<key for encryption>"
```

The decryption of the password is then automatically handled in ironic-ansible playbooks.

## Registering Baremetal Node for OneView Driver

```
ironic node-create -d agent_pxe_oneview
```

Update node driver-info:

```
ironic node-update $NODE_UUID add driver_info/server_hardware_uri=$SH_URI
```

## Updating Node Properties

```
ironic node-update $NODE_UUID add \
properties/capabilities=server_hardware_type_uri:$SHT_URI,enclosure_group_uri:$E
```

## Creating Port for Driver

```
ironic port-create -n $NODE_UUID -a $MAC_ADDRESS
```

## Creating a Node

Create Node:

```
ironic node-create -n ovbay7 -d agent_pxe_oneview
```

Update driver info:

```
ironic node-update $ID add driver_info/server_hardware_uri="/rest/server-hardware/
```

Update node properties:

```
ironic node-update $ID add properties/local_gb=10
```

```
ironic node-update $ID add properties/cpus=24 properties/memory_mb=262144 properti
```

```
ironic node-update $ID add properties/capabilities=server_hardware_type_uri:'/rest/
```

## Getting Data using REST API

GET login session auth id:

```
curl -k https://<oneview_manger_url>/rest/login-sessions -H "content-type:application/json"
```

GET complete node details in json format:

```
curl -k "https://<manager_url>"/rest/server-hardware/30373237-3132-4753-4835-32325
```

## Ironic OneView CLI

ironic-oneview-cli is already installed in ironicclient venv with a symbolic link to it. To generate an rc file for the OneView CLI, follow these steps:

1. Run the following commands:

```
source ~/service.osrc
glance image-list
```

2. Note the `deploy-kernel` and `deploy-ramdisk` UUID and then run the following command to generate the `rc` file:

```
ironic-oneview genrc
```

You will be prompted to enter:

- OneView Manager URL
- Username
- `deploy-kernel`
- `deploy-ramdisk`
- `allow_insecure_connection`
- `cacert` file

The `ironic-oneview.rc` file will be generated in the current directory, by default. It is possible to specify a different location.

3. Source the generated file:

```
source ironic-oneview.rc
```

You can now use the CLI for node and flavor creation as follows:

```
ironic-oneview node-create
ironic-oneview flavor-create
```

## RAID Configuration for Ironic

1. Node Creation:

Check the raid capabilities of the driver:

```
ironic --ironic-api-version 1.15 driver-raid-logical-disk-properties pxe_ilo
```

This will generate output similar to the following:

Property	Description
<code>controller</code>	Controller to use for this logical disk. If not specified on the bare metal node. Optional.

Node State will be Available

```
ironic node-create -d pxe_ilo -i ilo_address=<ip_address> \
    -i ilo_username=<username> -i ilo_password=<password> \
    -i ilo_deploy_iso=<iso_id> -i deploy_kernel=<kernel_id> \
    -i deploy_ramdisk=<ramdisk_id> -p cpus=2 -p memory_mb=4096 \
    -p local_gb=80 -p cpu_arch=amd64 \
    -p capabilities="boot_option:local,boot_mode:bios"
```

```
ironic port-create -a <port> -n <node-uuid>
```

2. Apply the target raid configuration on the node:

See the OpenStack documentation for RAID configuration at <http://docs.openstack.org/developer/ironic/deploy/raid.html>.

Set the target RAID configuration by editing the file `raid_conf.json` and setting the appropriate values, for example:

```
{ "logical_disks": [ { "size_gb": 5, "raid_level": "0", "is_root_volume": true} ]}
```

and then run the following command:

```
ironic --ironic-api-version 1.15 node-set-target-raid-config <node-uuid> raid_config
```

The output produced should be similar to the following:

Property	Value
chassis_uuid	
clean_step	{}
console_enabled	False
created_at	2016-06-14T14:58:07+00:00

driver	pxe_ilo
driver_info	{u'ilo_deploy_iso': u'd43e589a-07db-4fce-a06e-98e2f38c', u'deploy_kernel': u'915c5c74-1ceb-4f78-bdb4-8547a90ac', u'ilo_address': u'10.1.196.116', u'deploy_ramdisk': u'154e7024-bf18-4ad2-95b0-726c09ce417a', u'ilo_password': u'Administrator'}
driver_internal_info	{u'agent_cached_clean_steps_refreshed': u'2016-06-15', u'agent_cached_clean_steps': {u'raid': [{u'interface': u'raid', u'priority': 0, u'step': u'delete_configuration'}, {u'interface': u'raid', u'priority': 0, u'step': u'create_configuration'}, {u'interface': u'deploy', u'priority': 10, u'step': u'reboot_abortable': True, u'erase_devices'}]}, u'hardware_manager_version': {u'generic_hardware_manager': u'agent_erase_devices_iterations': 1, u'agent_url': u'http://192.168.245.143:9999', u'agent_last_heartbeat': {}}, u'hardware_manager_version': {u'generic_hardware_manager': u'agent_erase_devices_iterations': 1, u'agent_url': u'http://192.168.245.143:9999', u'agent_last_heartbeat': {}}}
extra	{}
inspection_finished_at	None
inspection_started_at	None
instance_info	{u'deploy_key': u'XXN2ON0V9ER429MECETJMUG5YHTKOQOZ'}
instance_uuid	None
last_error	None
maintenance	False
maintenance_reason	None
name	None
power_state	power off
properties	{u'cpu_arch': u'amd64', u'root_device': {u'wwn': u'0x0000000000000000', u'cpus': 2, u'capabilities': u'boot_mode:bios,raid_level:6,boot_option:local', u'memory_gb': 8, u'local_gb': 4}}
provision_state	available
provision_updated_at	2016-06-15T07:16:27+00:00
reservation	padawan-ironic-cpl-c1-m2-mgmt
target_power_state	power off
target_provision_state	None
<b>target_raid_config</b>	{u'logical_disks': [{u'size_gb': 5, u'raid_level': u'5', u'is_root_volume': True}]}
updated_at	2016-06-15T07:44:22+00:00
uuid	22ab9f85-71a1-4748-8d6b-f6411558127e

Now set the state of the node to **manageable**:

```
ironic --ironic-api-version latest node-set-provision-state <node-uuid> manage
```

### 3. Manual cleaning steps:

Manual cleaning is enabled by default in production - the following are the steps to enable cleaning if the manual cleaning has been disabled.

- Provide `cleaning_network_uuid` in `ironic-conductor.conf`
- Edit `~/helion/my_cloud/definition/data/ironic/ironic_config.yml` file and set `enable_node_cleaning` to true
- Then run the following set of commands:

```

cd ~/helion/hos/ansible
git add -A
git commit -m "enabling node cleaning"
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
ansible-playbook -i hosts/localhost ready-deployment.yml
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml

```

After performing these steps, the state of the node will become **Cleaning**.

Run the following command:

```
ironic --ironic-api-version latest node-set-provision-state 2123254e-8b31-4259-b
```

Node-information after a Manual cleaning:

Property	Value
chassis_uuid	
clean_step	{}
console_enabled	False
created_at	2016-06-14T14:58:07+00:00
driver	pxe_ilo
driver_info	{u'ilo_deploy_iso': u'd43e589a-07db-4fce-a06e-98e2f38', u'deploy_kernel': u'915c5c74-1ceb-4f78-bdb4-8547a90ac', u'ilo_address': u'10.1.196.116', u'deploy_ramdisk': u'154e7024-bf18-4ad2-95b0-726c09ce417a', u'ilo_password': u'Administrator'}
driver_internal_info	{u'agent_cached_clean_steps_refreshed': u'2016-06-15', u'agent_cached_clean_steps': {u'raid': [{u'interface': u'priority': 0, u'step': u'delete_configuration'}, {u'interface': u'priority': 0, u'step': u'create_configuration'}], u'priority': 10, u'interface': u'deploy', u'reboot': u'abortable': True, u'step': u'erase_devices'}]}, u'hardware_manager_version': {u'generic_hardware_manager': u'agent_erase_devices_iterations': 1, u'agent_url': u'http://192.168.245.143:9999', u'agent_last_heartbeat': None}, u'extra': {}}
extra	{}
inspection_finished_at	None
inspection_started_at	None
instance_info	{u'deploy_key': u'XXN2ON0V9ER429MECETJMUG5YHTKOQOZ'}
instance_uuid	None
last_error	None
maintenance	False
maintenance_reason	None
name	None
power_state	power off
properties	{u'cpu_arch': u'amd64', u'root_device': {u'wwn': u'0x', u'cpus': 2, u'capabilities':

	u'boot_mode:bios,raid_level:6,boot_option:local', u'm
	u'local_gb': 4}
provision_state	manageable
provision_updated_at	2016-06-15T07:16:27+00:00
raid_config	{u'last_updated': u'2016-06-15 07:16:14.584014', u'ph [ {u'status': u'ready', u'size_gb': 1024, u'interface_ u'firmware': u'HPGC', u'controller': u'Smart Array P4 (Embedded)', u'model': u'ATA MM1000GBKAL', u'disk u'id': u'1I:3:3'}, {u'status': u'ready', u'size_gb': u'interface_type': u'sata', u'firmware': u'HPGC', u'c Array P440ar in Slot 0 (Embedded)', u'model': u'ATA u'disk_type': u'hdd', u'id': u'1I:3:1'}, {u'status': u'size_gb': 1024, u'interface_type': u'sata', u'firmw u'controller': u'Smart Array P440ar in Slot 0 (Embedde u'ATA MM1000GBKAL', u'disk_type': u'hdd', u'id': {u'status': u'active', u'size_gb': 1024, u'interface_ u'firmware': u'HPGC', u'controller': u'Smart Array P4 (Embedded)', u'model': u'ATA MM1000GBKAL', u'disk u'id': u'2I:3:6'}, {u'status': u'active', u'size_gb': u'interface_type': u'sata', u'firmware': u'HPGC', u'c Array P440ar in Slot 0 (Embedded)', u'model': u'ATA u'disk_type': u'hdd', u'id': u'2I:3:5'}, {u'status': u'size_gb': 1024, u'interface_type': u'sata', u'firmw u'controller': u'Smart Array P440ar in Slot 0 (Embedde u'ATA MM1000GBKAL', u'disk_type': u'hdd', u'id': u'logical_disks': [{u'size_gb': 4, u'physical_disks': u'2I:3:6', u'2I:3:5', u'1I:3:4'}, u'raid_level': u'6' u'is_root_volume': True, u'root_device_hint': {u'wwn': u'0x600508b1001ce286'}, u'controller': u'Smart Array (Embedded)', u'volume_name': u'015E795CPDNLH0BRH8N406
reservation	padawan-ironic-cp1-c1-m2-mgmt
target_power_state	power off
target_provision_state	None
target_raid_config	{u'logical_disks': [{u'size_gb': 5, u'raid_level': u' u'is_root_volume': True}]}
updated_at	2016-06-15T07:44:22+00:00
uuid	22ab9f85-71a1-4748-8d6b-f6411558127e

After the manual cleaning, run the following command to change the state of a node to **available**:

```
ironic --ironic-api-version latest node-set-provision-state <node-uuid> provide
```

## Audit Support for Ironic

### API Audit Logging

Audit middleware supports delivery of CADF audit events via Oslo messaging notifier capability. Based on `notification_driver` configuration, audit events can be routed to messaging infrastructure (`notification_driver = messagingv2`) or can be routed to a log file (`notification_driver = log`).

Audit middleware creates two events per REST API interaction. The first event has information extracted from request data and the second one contains information on the request outcome (response).

# Enabling API Audit Logging

You can enable audit logging for Ironic by changing the configuration in the input model. Edit the file `~/helion/my_cloud/definition/cloudConfig.yml` and in the `audit-settings` section, change the default value to `enabled`. The `ironic-ansible` playbooks will now enable audit support for Ironic.

API audit events will be logged in the corresponding audit directory, for example, /var/audit/ironic/ironic-api-audit.log. An audit event will be logged in the log file for every request and response for an API call.

# Sample Audit Event

The following output is an example of an audit event for an ironic node-list command:

```
    "credential": {
        "token": "***",
        "identity_status": "Confirmed"
    },
    "host": {
        "agent": "python-ironicclient",
        "address": "10.1.200.129"
    },
    "project_id": "d8f52dd7d9e1475dbbf3ba47a4a83313",
    "id": "8c1a948bad3948929aa5d5b50627a174"
},
"action": "read",
"outcome": "pending",
"id": "061b7aa7-5879-5225-a331-c002cf23cb6c",
"requestPath": "/v1/nodes/?associated=True"
},
"priority": "INFO",
"publisher_id": "ironic-api",
"message_id": "2f61ebaa-2d3e-4023-afba-f9fca6f21fc2"
}
```

---

# Chapter 14. Installation for HPE Helion Entry-scale Cloud with Swift Only

This page describes the installation step requirements for the HPE Helion Entry-scale Cloud with Swift Only model.

## Important Notes

- If you are looking for information about when to use the GUI installer and when to use the CLI, see the Installation Overview.
- Review the Chapter 2, *Hardware and Software Support Matrix* that we have listed.
- Review the release notes to make yourself aware of any known issues and limitations.
- The installation process can occur in different phases. For example, you can install the control plane only and then add Compute nodes afterwards if you would like.
- If you run into issues during installation, we have put together a list of Chapter 18, *Troubleshooting the Installation* you can reference.
- Make sure all disks on the system(s) are wiped before you begin the install. (For Swift, refer to the section called “Swift Requirements for Device Group Drives”)
- There is no requirement to have a dedicated network for OS-install and system deployment, this can be shared with the management network. More information can be found in Chapter 10, *Example Configurations*.
- You may see the terms deployer and lifecycle manager used interchangeably. These are referring to the same nodes in your environment.
- When running the Ansible playbook in this installation guide, if a runbook fails you will see in the error response to use the `--limit` switch when retrying a playbook. This should be avoided. You can simply re-run any playbook without this switch.
- DVR is not supported with ESX compute.
- When you attach a Cinder volume to the VM running on the ESXi host, the volume will not get detected automatically. Make sure to set the image metadata `vmware_adaptertype=lsiLogicsas` for image before launching the instance. This will help to discover the volume change appropriately.

## Before You Start

We have put together a Chapter 2, *Pre-Installation Checklist* that should help with the recommended pre-installation tasks.

## Setting Up the Lifecycle Manager

### Installing the Lifecycle Manager

The lifecycle manager will contain the installation scripts and configuration files to deploy your cloud. You can set up the lifecycle manager on a dedicated node or you do so on your first controller node. The default choice is to use the first controller node as the lifecycle manager.

1. Sign in and download the product and signature files at the link below:
  - a. Software Entitlement Portal [<http://www.hpe.com/software/entitlements>]
2. You can verify the download was complete via the signature verification process outlined here [<https://h20392.www2.hpe.com/portal/swdepot/displayProductInfo.do?productNumber=HPLinuxCodeSigning>].
3. Boot your lifecycle manager from the ISO contained in the download.
4. Enter "install" to start installation.

## Note

"install" is all lower case

5. Select the language. Note that only the English language selection is currently supported.
6. Select the location.
7. Select the keyboard layout.
8. Select the primary network interface, if prompted:
  - a. Assign IP address, subnet mask, and default gateway
9. Create new account:
  - a. Enter a username.
  - b. Enter a password.
  - c. Enter time zone.

Once the initial installation is finished, complete the lifecycle manager setup with these steps:

1. Ensure your lifecycle manager has a valid DNS nameserver specified in `/etc/resolv.conf`.
2. Set the environment variable `LC_ALL`:

```
export LC_ALL=C
```

## Note

This can be added to `~stack/.bashrc` or `/etc/bash.bashrc`.

At the end of this section you should have a node set up with Linux for HPE Helion on it.

## Configure and Run the Lifecycle Manager

### Important

It is critical that you don't run any of the commands below as the `root` user or use `sudo`, unless it is stated explicitly in the steps. Run them as the user you just created (or `stack` if you left the default of "stack").

1. Log into your lifecycle manager as the user you created and mount the install media at `/media/cdrom`. It may be necessary to use `wget` or another file transfer method to transfer the install media to the lifecycle manager before completing this step. Here is the command to mount the media:

Example for HPE Helion OpenStack® 5.0:

```
sudo mount HelionOpenStack-5.0.iso /media/cdrom
```

2. Unpack the tarball that is in the `/media/cdrom/hos/` directory:

Example for HPE Helion OpenStack 5.0

```
tar xvf /media/cdrom/hos/hos-5.0.0-<timestamp>.tar
```

3. Run the `hos-init.bash` script which is included in the build:

Example for HPE Helion OpenStack® 5.0

```
~/hos-5.0.0/hos-init.bash
```

You will be prompted to enter an optional SSH passphrase when running `hos-init.bash`. This passphrase is used to protect the key used by Ansible when connecting to its client nodes. If you do not want to use a passphrase then just press **Enter** at the prompt.

For automated installation (e.g. CI) it is possible to disable SSH passphrase prompting by setting the `HOS_INIT_AUTO` environment variable before running `hos-init.bash`, like this:

```
export HOS_INIT_AUTO=y
```

If you have protected the SSH key with a passphrase then execute the following commands to avoid having to enter the passphrase on every attempt by Ansible to connect to its client nodes:

```
eval $(ssh-agent)
ssh-add ~/.ssh/id_rsa
```

At the end of this section you should have a local directory structure, as described below:

<code>~/helion/</code>	Top level directory
<code>~/helion/examples/</code>	Directory contains the config input files of the
<code>~/helion/my_cloud/definition/</code>	Directory contains the config input files
<code>~/helion/my_cloud/config/</code>	Directory contains .j2 files which are symlinks to
<code>~/helion/hos/</code>	Directory contains files used by the installer

## Warning

It is important that you do not add any extra files in the `~/helion` directory on your lifecycle manager. This includes temporary save files. Doing so will cause errors during the installation.

## Important

Be aware that the files in the `~/helion/my_cloud/config/` directory are symlinks to the `~/helion/hos/ansible/` directory. For example, `~/helion/my_cloud/config/keepalived/defaults.yml` is a symbolic link to `~/helion/hos/ansible/roles/keepalived/defaults/main.yml`.

```
ls -al ~/helion/my_cloud/config/keepalived/defaults.yml  
lwxrwxrwx 1 stack stack 55 May 24 20:38 /home/stack/helion/my_cloud/config/kee
```

If you are using a tool like **sed** to make edits to files in this directory, you might break the symbolic link and create a new copy of the file. To maintain the link, you will need to force **sed** to follow the link:

```
sed -i --follow-symlinks \  
's$keepalived_vrrp_offset: 0$keepalived_vrrp_offset: 2$' \  
~/helion/my_cloud/config/keepalived/defaults.yml
```

Alternatively, directly edit the target of the link `~/helion/hos/ansible/roles/keepalived/defaults/main.yml`.

For any troubleshooting information regarding these steps, see the section called “Issues during Lifecycle-manager Setup”.

## Configure Your Environment

This part of the install is going to depend on the specific cloud configuration you are going to use.

1. Setup your configuration files, as follows:

- a. See the sample sets of configuration files in the `~/helion/examples/` directory. Each set will have an accompanying `README.md` file that explains the contents of each of the configuration files.
- b. Copy the example configuration files into the required setup directory and edit them to contain the details of your environment:

```
cp -r ~/helion/examples/entry-scale-swift/* ~/helion/my_cloud/definition/  
  
c. Begin inputting your environment information into the configuration files in the ~/he-  
lion/my_cloud/definition directory.
```

Full details of how to do this can be found here: .

In many cases, the example models provide most of the data you need to create a valid input model. However, there are two important aspects you must plan and configure before starting a deploy as follows:

- Check the disk model used by your nodes. Specifically, check that all disk drives are correctly named and used as described in the section called “Swift Requirements for Device Group Drives”.
- Select an appropriate partition power for your rings. Detailed information about this is provided at the section called “Understanding Swift Ring Specifications”.

Optionally, you can use the `hosencrypt.py` script to encrypt your iLo passwords. This script uses OpenSSL.

a. Change to the Ansible directory:

```
cd ~/helion/hos/ansible
```

b. Put the encryption key into the following environment variable:

```
export HOS_USER_PASSWORD_ENCRYPT_KEY=<encryption key>
```

- c. Run the python script below and follow the instructions. Enter a password that you want to encrypt.

```
hosencrypt.py
```

- d. Take the string generated and place it in the "ilo\_password" field in your ~/helion/my\_cloud/definition/data/servers.yml file, remembering to enclose it in quotes.

- e. Repeat the above for each server.

## Note

Before you run any playbooks, remember that you need to export the encryption key in the following environment variable: `export HOS_USER_PASSWORD_ENCRYPT_KEY=<encryption key>`

Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*), as follows:

```
cd ~/helion/hos/ansible  
git add -A  
git commit -m "My config or other commit message"
```

## Important

This step needs to be repeated any time you make changes to your configuration files before you move onto the following steps. See Chapter 3, *Using Git for Configuration Management* for more information.

# Running the Configuration Processor

Once you have your configuration files setup you need to run the configuration processor to complete your configuration.

When you run the configuration processor you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the ansible `group_vars` and `host_vars` that it produces for subsequent deploy runs. You will need this password for subsequent ansible deploy and configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise just press **Enter** to bypass this.

Run the configuration processor with this command:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

For automated installation (e.g. CI) you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor

```
ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e rekey=""
```

If you receive an error during this step then there is probably an issue with one or more of your configuration files. We recommend that you verify that all of the information in each of your configuration files is correct

for your environment and then commit those changes to git using the instructions in the previous section before re-running the configuration processor again.

For any troubleshooting information regarding these steps, see the section called “Issues while Updating Configuration Files”.

## Provisioning Your Baremetal Nodes

To provision the baremetal nodes in your cloud deployment you can either use the automated operating system installation process provided by HPE Helion OpenStack or you can use the 3rd party installation tooling of your choice. We will outline both methods below:

### Using 3rd Party Baremetal Installers

If you do not wish to use the automated operating system installation tooling included with 5.0 then the requirements that have to be met using the installation tooling of your choice are:

- The operating system must be installed via the HPE Linux for HPE Helion OpenStack ISO provided on the Software Entitlement Portal [<http://www.hpe.com/software/entitlements>].
- Each node must have SSH keys in place that allows the same user from the lifecycle manager node who will be doing the deployment to SSH to each node without a password.
- Passwordless sudo needs to be enabled for the user.
- There should be a LVM logical volume as /root on each node.
- If the LVM volume group name for the volume group holding the "root" LVM logical volume is hlm-vg then it will align with the disk input models in the examples.
- Ensure that openssh-server, python, python-apt, and rsync are installed.

If you chose this method for installing your baremetal hardware, skip forward to the Chapter 10, *Installing Mid-scale and Entry-scale KVM* step.

If you would like to use the automated operating system installation tools provided by 5.0 then complete all of the steps below.

### Using the Automated Operating System Installation Provided by HPE Helion OpenStack

#### Part One: Deploy Cobbler

This phase of the install process takes the baremetal information that was provided in servers.yml and installs the Cobbler provisioning tool and loads this information into Cobbler. This sets each node to netboot-enabled: true in Cobbler. Each node will be automatically marked as netboot-enabled: false when it completes its operating system install successfully. Even if the node tries to PXE boot subsequently, Cobbler will not serve it. This is deliberate so that you can't reimagine a live node by accident.

The cobbler-deploy.yml playbook prompts for a password - this is the password that will be encrypted and stored in Cobbler, which is associated with the user running the command on the lifecycle manager, that you will use to log in to the nodes via their consoles after install. The username is the same as the user set up in the initial dialogue when installing the lifecycle manager from the iso, and is the same user that is running the cobbler-deploy play.

1. Run the following playbook which confirms that there is iLo connectivity for each of your nodes so that they are accessible to be re-imaged in a later step:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost bm-power-status.yml
```

2. Run the following playbook to deploy Cobbler:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

### Part Two: Image the Nodes

This phase of the install process goes through a number of distinct steps:

1. Powers down the nodes to be installed
2. Sets the nodes hardware boot order so that the first option is a network boot.
3. Powers on the nodes. (The nodes will then boot from the network and be installed using infrastructure set up in the previous phase)
4. Waits for the nodes to power themselves down (this indicates a success install). This can take some time.
5. Sets the boot order to hard disk and powers on the nodes.
6. Waits for the nodes to be ssh-able and verifies that they have the signature expected.

The reimage command is:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost bm-reimage.yml [-e nodelist=node1,node2,node3]
```

If a nodelist is not specified then the set of nodes in cobbler with netboot-enabled: True is selected. The playbook pauses at the start to give you a chance to review the set of nodes that it is targeting and to confirm that it's correct.

You can use the command below which will list all of your nodes with the netboot-enabled: True flag set:

```
sudo cobbler system find --netboot-enabled=1
```

For any troubleshooting information regarding these steps, see the section called “Issues while Provisioning your Baremetal Nodes”.

## Deploying the Cloud

1. Use the playbook below to create a deployment directory:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

2. [OPTIONAL] - Run the wipe\_disks.yml playbook to ensure all of your partitions on your nodes are completely wiped before continuing with the installation. If you are using fresh machines this step may not be necessary.

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts wipe_disks.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts wipe_disks.yml --ask-vault-pass
```

3. Run the site.yml playbook below:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts site.yml
```

If you have used an encryption password when running the configuration processor use the command below and enter the encryption password when prompted:

```
ansible-playbook -i hosts/verb_hosts site.yml --ask-vault-pass
```

### Note

The step above runs osconfig to configure the cloud and hlm-deploy to deploy the cloud. Therefore, this step may run for a while, perhaps 45 minutes or more, depending on the number of nodes in your environment.

4. Verify that the network is working correctly. Ping each IP in the /etc/hosts file from one of the controller nodes.

For any troubleshooting information regarding these steps, see the section called “Issues while Deploying the Cloud”.

## Post-Installation Verification and Administration

We recommend verifying the installation using the instructions in Chapter 22, *Cloud Verification*.

There are also a list of other common post-installation administrative tasks listed in the Chapter 28, *Other Common Post-Installation Tasks* list.

---

# Chapter 15. Installing SLES Compute

## SLES Compute Node Installation Overview

HPE Helion OpenStack 5.0 supports SLES compute nodes, specifically SLES 12 SP2. HPE does not ship a SLES ISO with HPE Helion OpenStack so you will need to download a copy of the SLES iso (SLE-12-SP2-Server-DVD-x86\_64-GM-DVD1.iso) and the SLES SDK ISO (SLE-12-SP2-SDK-DVD-x86\_64-GM-DVD1.iso) from SUSE. You can use the following link to download the ISO. To do so, you will need to sign in or create a SUSE customer service account before downloading: <https://www.suse.com/products/server/download/>.

There are two approaches for deploying SLES compute nodes in HPE Helion OpenStack:

- Using the lifecycle manager to automatically deploy SLES Compute nodes.
- Provisioning SLES nodes yourself, either manually or using a 3rd party tool, and then provide the relevant information to the lifecycle manager.

These two approaches can be used whether you are installing a cloud for the first time or adding a compute node to an existing cloud. Regardless of your approach, you should be certain to register your SLES compute nodes in order to get product updates as they come available. For more information, see Chapter 1, *Registering SUSE Linux*.

## SLES Support

SUSE Linux Enterprise Server (SLES) Host OS KVM and/or supported SLES guests have been tested and qualified by HPE to run on HPE Helion OpenStack. Refer to the published compatibility matrix for your version of HPE Helion OpenStack at the following URL <http://docs.hpccloud.com/#home.html>. HPE is one of SUSE's largest OEMs with follow the sun global support coverage.

- **One Number to Call**

HPE customers who have purchased both HPE Helion OpenStack and SLES subscriptions with support from HPE will have one number to call for troubleshooting, fault isolation and support from HPE technical support specialists in HPE Helion OpenStack and SUSE technologies. If the problem is isolated to SLES software itself the issue will be replicated on a SUSE certified platform and escalated to SUSE for resolution.

- **A Dual Support Model**

HPE will troubleshoot and fault isolate an issue at the Helion software level. If HPE Helion OpenStack software is excluded as the cause of the problem, then customers who did not purchase SLES support from HPE will be directed to the vendor from whom they purchased SLES for continued support.

## Using the Lifecycle Manager to Deploy SLES Compute Nodes

The method used for deploying SLES compute nodes using Cobbler on the lifecycle manager uses legacy BIOS.

## Note

UEFI and Secure boot are currently not supported on SLES compute.

# Deploying legacy BIOS SLES compute nodes

The installation process for SLES nodes is almost identical to that of HPE Linux nodes as described in the topic for . The key differences are:

- The standard SLES ISO (SLE-12-SP2-Server-DVD-x86\_64-GM-DVD1.iso) must be accessible via /home/stack/sles12.iso. Rename the ISO or create a symbolic link:  

```
mv SLE-12-SP2-Server-DVD-x86_64-GM-DVD1.iso /home/stack/sles12.iso
```
- The contents of the SLES SDK ISO (SLE-12-SP2-SDK-DVD-x86\_64-GM-DVD1.iso) must be mounted or copied to /opt/hlm\_packager/hlm/sles12/zypper/SDK/. If you choose to mount the ISO, we recommend creating an /etc/fstab entry to ensure the ISO is mounted after a reboot.
- You must identify the node(s) on which you want to install SLES, by adding the key/value pair distro-id: sles12sp2-x86\_64 to server details in servers.yml. You will also need to update net\_interfaces.yml, server\_roles.yml, disk\_compute.yml and control\_plane.yml. For more information on configuration of the Input Model for SLES, see the section called “SLES Compute Nodes”.
- HPE Helion OpenStack playbooks currently do not take care of the SDK, therefore it needs to be added manually. The following command needs to be run on every SLES compute node:  

```
deployer_ip=192.168.10.254
zypper addrepo --no-gpgcheck --refresh http://$deployer_ip:79/hlm/sles12/zypper/
```

There is no need to add OS repo as in case of the section called “Provisioning SLES Yourself”, because SLES already comes with OS repo populated after a Cobbler-managed install.

# Provisioning SLES Yourself

## Introduction

This article outlines the steps needed to manually provision a SLES node so that it can be added to a new or existing HPE Helion OpenStack 5.0 cloud.

# Configure Lifecycle Manager to Enable SLES

- Take note of the lifecycle manager's IP address. It will be used below during .
- Mount or copy the contents of SLE-12-SP2-Server-DVD-x86\_64-GM-DVD1.iso to /opt/hlm\_packager/hlm/sles12/zypper/OS/
- Mount or copy the contents of SLE-12-SP2-SDK-DVD-x86\_64-GM-DVD1.iso to /opt/hlm\_packager/hlm/sles12/zypper/SDK/

## Note

If you choose to mount an ISO, we recommend creating an /etc/fstab entry to ensure the ISO is mounted after a reboot.

## Install SLES 12 SP2

Install SLES 12 SP2 using the standard iso (SLE-12-SP2-Server-DVD-x86\_64-GM-DVD1.iso)

## Assign a static IP

1. Use the `ip addr` command to find out what network devices are on your system:

```
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eno1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether f0:92:1c:05:89:70 brd ff:ff:ff:ff:ff:ff
    inet 10.13.111.178/26 brd 10.13.111.191 scope global eno1
        valid_lft forever preferred_lft forever
    inet6 fe80::f292:1cff:fe05:8970/64 scope link
        valid_lft forever preferred_lft forever
3: eno2: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether f0:92:1c:05:89:74 brd ff:ff:ff:ff:ff:ff
```

2. Identify the one that matches the MAC address of your server and edit the corresponding config file in `/etc/sysconfig/network-scripts/ifcfg-eno1`.

```
vi /etc/sysconfig/network-scripts/ifcfg-eno1
```

3. Edit the `IPADDR` and `NETMASK` values to match your environment. Note that the `IPADDR` is used in the corresponding stanza in `servers.yml`. You may also need to set `BOOTPROTO` to none.

```
TYPE=Ethernet
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
NAME=eno1
UUID=36060f7a-12da-469b-a1da-ee730a3b1d7c
DEVICE=eno1
ONBOOT=yes
NETMASK=255.255.255.192
IPADDR=10.13.111.14
```

4. [OPTIONAL] Reboot your SLES node and ensure that it can be accessed from the lifecycle manager.

## Add stack user and home directory

```
useradd -m stack  
passwd stack
```

## Allow user stack to sudo without password

Setting up sudo on SLES is covered in the SUSE documentation, Configuring sudo [[https://www.suse.com/documentation/sles-12/book\\_sle\\_admin/data/sec\\_sudo\\_conf.html](https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_sudo_conf.html)].

The recommendation is to create user specific sudo config files under /etc/sudoers.d, therefore creating an /etc/sudoers.d/stack config file with the following content will allow sudo commands without the requirement of a password.

```
stack ALL=(ALL) NOPASSWD:ALL
```

## Add zypper repository

Using the ISO-based repositories created above, add the zypper repositories.

Follow these steps. Update the value of deployer\_ip as necessary.

```
deployer_ip=192.168.10.254  
zypper addrepo --no-gpgcheck --refresh http://$deployer_ip:79/hlm/sles12/zypper/OS  
zypper addrepo --no-gpgcheck --refresh http://$deployer_ip:79/hlm/sles12/zypper/SD
```

To verify that the repositories have been added, run:

```
zypper repos --detail
```

You can find more information on Using Zypper in the SLES 12 Admin guide: [https://www.suse.com/documentation/sles-12/book\\_sle\\_admin/data/sec\\_zypper.html](https://www.suse.com/documentation/sles-12/book_sle_admin/data/sec_zypper.html)

## Add Required Packages

As documented in the section of , you will need to add some extra packages that are required. Ensure that `openssh-server`, `python`, and `rsync` are installed.

## Set up passwordless SSH access

Once you have started your installation using the lifecycle manager, or if you are adding a SLES node to an existing cloud, you need to copy the lifecycle manager public key to the SLES node. One way of doing this is to copy the `/home/stack/.ssh/authorized_keys` from another node in the cloud to the same location on the SLES node. If you are installing a new cloud, this file will be available on the nodes after running the `bm-reimage.yml` playbook.

### Important

Ensure that there is global read access to the file `/home/stack/.ssh/authorized_keys`.

Now test passwordless ssh from the deployer and check your ability to remotely execute sudo commands:

```
ssh stack@<<ip of sles node>> "sudo tail -5 /var/log/messages"
```

# Using SLES as a Ceph Client

## Warning

SUSE Linux Enterprise Server 12 SP2 comes with Ceph packages from the Ceph Jewel release. HPE Helion OpenStack 5.0 Ceph server is based on the Ceph Hammer release, therefore it is not advised that client related Ceph management commands from a SLES compute node be performed. Doing so could trigger an update of the Ceph config which would not be compatible with other non-SLES Ceph clients in the same HPE Helion OpenStack 5.0 system.

---

# Chapter 16. HLM-Hypervisor instructions

## Introduction

You can read an overview of the design on the Helion lifecycle manager creation of VMs [<https://wiki.hpccloud.net/display/core/HLM+creation+of+VMs>] page. This page describes the steps needed to set up and use a HLM-Hypervisor (previously known as a VM-factory) and related functionality of HPE Helion OpenStack 5.0. This comprises model changes and running a couple of new playbooks to bring up the VMs.

## Model changes

### **passthrough-network-groups**

With PB3 the specification of `passthrough-network-groups` is now supported within the `Interfaces` section of the input model.

A `passthrough-network` group is a network group that can be accessed by a VM hosted by the hypervisor system regardless of whether the network group is required on the hypervisor system itself.

<code>network-groups:</code>	List of network groups that this server may require access to.  Access to the network is only configured if the network-group is required by at least one service hosted on this server.
<code>forced-network-groups:</code>	List of network-groups that this server requires access to.  Access to the network is configured regardless of whether the network-group is required by any service hosted on this server.
<code>passthrough-network-groups:</code>	Network groups should not appear in both <code>network-groups</code> and <code>forced-network-groups</code> .  List of network-groups that need to be passed through this server to hosted VMs.
	No access is configured for this server if the network group is listed only in <code>passthrough-network-groups</code> .  The <code>passthrough-network-groups</code> may include networks from either <code>network-groups</code> or <code>forced-network-groups</code> but may also include network groups not used directly by this server. If a <code>passthrough-network-groups</code> is listed in either of <code>network-groups</code> or <code>forced-network-groups</code> the access to the network group will be such that both the server and hosted VMs are network peers.

### **Example:**

The following hypervisor interface-model example specifies :

- `bond0`

- PXE available to hypervisor and Hosted-VMs
  - CLM available to hypervisor and Hosted-VMs
  - CAN available to Hosted-VMs only
  - bond1
    - VxLAN-VLAN1-TUL, EXT and VLAN2-TUL available to Hosted-VMs only
- ```
interface-models:  
  - name: HLM-HYPERVISOR-INTERFACES  
    network-interfaces:  
      - name: BOND0  
        device:  
          name: bond0  
        bond-data:  
          options:  
            mode: active-backup  
            miimon: 200  
            primary: hed5  
            provider: linux  
          devices:  
            - name: hed5  
    network-groups:  
      - PXE  
      - CLM  
    passthrough-network-groups:  
      - PXE  
      - CLM  
      - CAN  
  - name: BOND1  
    device:  
      name: bond1  
    bond-data:  
      options:  
        mode: active-backup  
        miimon: 200  
        primary: hed1  
        provider: linux  
      devices:  
        - name: hed1  
    passthrough-network-groups:  
      - VxLAN-VLAN1-TUL  
      - EXT  
      - VLAN2-TUL
```

## **hlm-hypervisor**

With PB3 we now support the specification of `hlm-hypervisor` boolean within the Servers section of the input model. This setting is used to identify nodes that can host VMs (non-NOVA).

This is related to the specification of `hypervisor-id` that is required to identify servers to be instantiated as hosted-VMs.

- Any server referenced as a target of a hypervisor-id must have hlm-hypervisor set to True.
- A server with hlm-hypervisor of True setting may or may not have Helion lifecycle manager Hosted-VMs associated with it.

## Example

In the following example from servers.yml

- a single server (hypervisor1) is declared as supporting Hosted-VMs using hlm-hypervisor: True
- three nodes (controller1, controller2 and controller3) are declared as Hosted-VMs on- hypervisor1 (hypervisor-id: hypervisor1)
  - id: hypervisor1  
ip-addr: 10.225.107.74  
role: HLM-HYPERVISOR-ROLE  
server-group: RACK2  
hlm-hypervisor: True  
nic-mapping: HP-BL460-6PORT  
mac-addr: 9c:7e:96:22:9e:d8  
ilo-ip: 10.1.18.42  
ilo-password: whatever  
ilo-user: whatever
  - id: controller1  
ip-addr: 10.225.107.75  
role: CONTROLLER-ROLE  
hypervisor-id: hypervisor1  
nic-mapping: VIRTUAL-1-PORT
  - id: controller2  
ip-addr: 10.225.107.76  
role: CONTROLLER-ROLE  
hypervisor-id: hypervisor1  
nic-mapping: VIRTUAL-1-PORT
  - id: controller3  
ip-addr: 10.225.107.77  
role: CONTROLLER-ROLE  
hypervisor-id: hypervisor1  
nic-mapping: VIRTUAL-1-PORT
  - id: compute1  
ip-addr: 10.225.107.78  
role: COMPUTE-ROLE  
server-group: RACK1  
nic-mapping: HP-BL460-6PORT  
mac-addr: 6c:9e:96:22:7e:d8  
ilo-ip: 10.1.18.44  
ilo-password: whatever  
ilo-user: whatever

The basic steps are as follows, with examples below. In all of these examples, it is assumed that hed1 is the physical interface to which the VMs need to be connected.

1. Create a new network interfaces group for the Hypervisor nodes, appropriate to the hardware in use. It is called HLM-HYPERVISOR-INTERFACES in this example. You may also want to create nic mappings for these nodes.

```
@@ -94,3 +94,12 @@
      network-groups:
        - MANAGEMENT

+      - name: HLM-HYPERVISOR-INTERFACES
+      network-interfaces:
+        - name: hed1
+        device:
+          name: hed1
+        network-groups:
+          - EXTERNAL-VM
+          - GUEST
+          - MANAGEMENT
```

2. Create a new disk model for the Hypervisor nodes. Here's an example for a Hypervisor node with 4 disks. In this specific example, we've included storage to be set-aside in the HPE Helion OpenStack input model for gluster.

You will need to tailor this disk model to the hardware that you are using. For example, if your Hypervisor nodes only have one disk, you would delete the line referencing /dev/sdb from the hlm-vg volume group definition, and remove the vg-images volume group altogether.

```
---
  product:
    version: 2
  disk-models:
    - name: HLM-HYPERVISOR-DISKS
      volume-groups:
        - name: hlm-vg
          physical-volumes:
            - /dev/sda_root
            - /dev/sdb
          logical-volumes:
            - name: root
              size: 35%
              fstype: ext4
              mount: /
            - name: log
              size: 50%
              mount: /var/log
              fstype: ext4
              mkfs-opt: -O large_file
            - name: crash
              size: 10%
              mount: /var/crash
              fstype: ext4
              mkfs-opt: -O large_file
      # optional VG dedicated to VM images to keep VM IOPS off the OS disk
```

```
# The consumer stanza allows user defined location of where to store the q
- name: hlm-images
physical-volumes:
- /dev/sdc
- /dev/sdd
logical-volumes:
- name: hlm-images
size: 95%
mount: /var/lib/hlm-images
fstype: ext4
mkfs-opt: -O large_file
consumer:
name: hlm-hypervisor
usage: hlm-hypervisor-images

# Optionally define some storage in the input model to be used
# by gluster
device-groups:
- name: gluster
devices:
- name: /dev/sde
- name: /dev/sdf
consumer:
name: gluster
suppress-warnings: True
```

Note the `suppress-warnings: True` flag for "gluster". This indicates to the configuration processor that it should not issue warnings when it cannot find a "gluster" service definition.

3. Create a new role that uses this interface and disk model.

```
@@ -34,3 +34,7 @@
- name: DEPLOYER-ROLE
  interface-model: DEPLOYER-INTERFACES
  disk-model: DEPLOYER-DISKS
+
+ - name: HLM-HYPERVISOR-ROLE
+   interface-model: HLM-HYPERVISOR-INTERFACES
+   disk-model: HLM-HYPERVISOR-DISKS
```

4. Define a new type of node in the "clusters" section of `control_plane.yml` so that nodes with this role will get their own CP group.

```
@@ -135,3 +135,10 @@
- ntp-client
- vsa

+
- name: hypervisor
  resource-prefix: vmf
+
  server-role: HLM-HYPERVISOR-ROLE
+
  allocation-policy: strict
+
  min-count: 0
+
  service-components:
-
  - lifecycle-manager-target
+
  - ntp-server
```

If the lifecycle manager is also a hypervisor then you should add the component `lifecycle-manager` instead of `lifecycle-manager-target`. It is required that the hypervisor nodes have the `ntp-server` component.

5. Assign the HLM-HYPERVERSOR-ROLE to the appropriate nodes in servers.yml
  - a. If the lifecycle manager node is also going to be a Hypervisor, then give it the HLM-HYPERVISOR-ROLE in servers.yml
  - b. Having a baremetal node that is both a Hypervisor and a Controller is not supported.
  - c. Also add definitions for all of the control plane VMs that you intend to create later. Note that you don't need iLO information or mac-addresses for VMs.
  - d. You will need to modify the IP addresses etc in this example to match your machine.

```
@@ -80,7 +80,7 @@

+     - id: hypervisor1
+       ip-addr: 10.225.107.74
+       role: HLM-HYPERVERSOR-ROLE
+       hlm-hypervisor: True
+       server-group: RACK2
+       nic-mapping: HP-BL460-6PORT
+       mac-addr: 9c:7e:96:22:9e:d8
+       ilo-ip: 10.1.18.42
+       ilo-password: whatever
+       ilo-user: whatever
+
+     - id: controller1
+       ip-addr: 10.225.107.75
+       role: CONTROLLER-ROLE
+       hypervisor-id: hypervisor1
+       nic-mapping: VIRTUAL-1-PORT
+
+     - id: controller2
+       ip-addr: 10.225.107.76
+       role: CONTROLLER-ROLE
+       hypervisor-id: hypervisor1
+       nic-mapping: VIRTUAL-1-PORT
+
+     - id: controller3
+       ip-addr: 10.225.107.77
+       role: CONTROLLER-ROLE
+       hypervisor-id: hypervisor1
+       nic-mapping: VIRTUAL-1-PORT
+
+     - id: compute1
+       ip-addr: 10.225.107.78
+       role: COMPUTE-ROLE
+       server-group: RACK1
+       nic-mapping: HP-BL460-6PORT
+       mac-addr: 6c:9e:96:22:7e:d8
+       ilo-ip: 10.1.18.44
```

```
+      ilo-password: whatever
+      ilo-user: whatever
```

6. Add a definition of this new nic-mapping to `nic_mappings.yml`:

```
- name: VIRTUAL-1-PORT
  physical-ports:
    - bus-address: '0000:01:01.0'
      logical-name: hed1
      type: simple-port
```

See the section NIC Mapping for Helion lifecycle manager Virtual-Controllers below for more information and examples.

7. You will also need to make model changes for the vms. The vm role above is CONTROLLER-ROLE. You need to add information to the model that specifies:

- a. The number of vcpus - this is done by adding a `cpu-model` to the vm role with a `vm-size` stanza
- b. The amount of RAM to be used - this is done by adding a `memory-model` to the vm role with a `vm-size` stanza
- c. The sizes of the virtual disks - this is done by adding a `vm-size` stanza to the disk-model used with the vm-role.

The CONTROLLER-ROLE definition will look like:

```
- name: CONTROLLER-ROLE
  interface-model: CONTROLLER-INTERFACES
  disk-model: CONTROLLER-DISKS
  cpu-model: CONTROLLER-CPU
  memory-model: CONTROLLER-MEMORY
```

The `cpu-model` will look like the following - place it in a yaml file and set `vcpus` to the value that you want to use:

```
---
product:
  version: 2

cpu-models:
  - name: CONTROLLER-CPU
    vm-size:
      vcpus: 4
```

The `memory-model` will look like the following - place it in a yaml file and set `ram` to the value that you want to use (valid values are in "KMG"):

```
---
product:
  version: 2
memory-models:
  - name: CONTROLLER-MEMORY
    vm-size:
      ram: 16G
```

The disk-model will have a stanza like the following added to it:

```
disk-models:  
- name: CONTROLLER-DISKS  
  vm-size:  
    disks:  
      - name: /dev/vda_root  
        size: 1T  
      - name: /dev/vdb  
        size: 1T  
      - name: /dev/vdc  
        size: 1T  
      - name: /dev/vdd  
        size: 1T
```

disks is a list of all the physical-volumes in volume-groups and devices in device-groups used in the disk-model, one entry for each with the name : field and the size : field. Valid values for size are KMGT

8. Note that the network that is associated with ansible traffic needs to be on the network interface in the vm which is on the "lowest" pci address. In the nic-mapping description below this would be hex1.
9. You do need to have a vda\_root described in your input model,

## Bootstrap Instructions

Start with one bare-metal node and install it directly from the HPE Helion OpenStack 5.0 iso e.g. through virtual media on its iLO. Follow the usual install instructions e.g. but make sure you use a model with the changes described above. Follow all the usual steps, including cobbler-deploy, bm-reimage and config-processor-run. Stop after running ready-deployment.yml though, and move to the instructions in the next section, to bring up the virtual control plane nodes.

## Customizing for VCP

The ready-deployment.yml playbook will have created ~/scratch/ansible/next/hos/ansible so change to that directory and perform the rest of your work from that directory.

```
cd ~/scratch/ansible/next/hos/ansible
```

There are a number of installation choices at this point:

1. You are installing a VCP system which is fully described and managed by input model.
2. You are installing a VCP system where additional steps need to be performed after provisioning the VCP VMs before running site.yml
3. You are installing a VCP system where additional steps need to be performed between configuring the hypervisor nodes, deploying the hypervisors, provisioning the VCP VMs, or running site.yml

## Installing a fully input model managed Virtual Control Plane based HPE Helion OpenStack Cloud

The plays to setup the hlm-hypervisor have been encapsulated into a single play that can be run before running site.yml to bring up the full cloud.

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts hlm-hypervisor-setup.yml
ansible-playbook -i hosts/verb_hosts site.yml
```

This will configure and deploy the Helion lifecycle manager Hypervisor nodes, provision the VCP VMs, and fully deploy the HPE Helion OpenStack Cloud.

## Provisioning just the Virtual Control Plane VMs

If your HPE Helion OpenStack Cloud installation requires that you perform additional actions between the provisioning of the VCP VMs and the running of the site.yml playbook then you can run the hlm-hypervisor-setup.yml playbook to provision the VCP VMs only, as follows:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts hlm-hypervisor-setup.yml
```

This will configure and deploy the Helion lifecycle manager Hypervisor nodes, and then provision the VCP VMs. Once you have performed the additional actions you can complete the installation by running the site.yml playbook.

## Manually running each phase of the Virtual Control Plane provisioning

If you need explicit control over when the different phases of the configuration and deployment of the HLM Hypervisor nodes and the provisioning of the VCP VMs happen then you can run through the following steps, performing any required additional actions before moving on to the next step.

### Configure OS and Networking on HLM Hypervisor nodes

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts osconfig-run.yml --limit=hlm-hypervisors
```

This will configure the OS environment and any relevant networking infrastructure on the HLM Hypervisor nodes that will be needed to support the associated VCP VMs.

### Deploy and Configure the HLM Hypervisor nodes

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts hlm-hypervisor-deploy.yml
```

This will install and configure any additional software that needs to be in place before you can provision the VCP VMs.

Provision the VCP VMs

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts hlm-hypervisor-vms-deploy.yml
```

This will provision and start the VCP VMs, but will not perform any additional setup operations on them.

When you have completed any additional actions you can then run site.yml to complete the installation of the HPE Helion OpenStack Cloud.

# NIC Mapping for HLM Virtual-Controllers

For a bare-metal system the set of nic-mappings specified within `nic_mappings.yml` file define the mapping of PCI bus-addresses to device name.

For a HLM Virtual-Controller the set of nic-mappings specified within `nic_mappings.yml` file defines the mapping of PCI bus-addresses to device name, but is also used to construct the HLM Virtual-Controller VM with network interfaces at those PCI bus-addresses.

When defining the network-interface model for a HLM Virtual-Controller it will become apparent how many network interfaces are required for each controller and therefore how many NICs need to be defined in the `nic-mappings` for this VM.

It is recommended that you define a separate `nic-mappings` for each number of NICs that you reference. For example the following defines three mappings, one with a single NIC, a second with two NICs and a third with eight NICs:

```
...
  nic-mappings:
  ...
    - name: HOS-VIRTUAL-ONE-PORT
      physical-ports:
        - logical-name: hed1
          bus-address: '0000:01:01.0'
          type: simple-port
    - name: HOS-VIRTUAL-TWO-PORT
      physical-ports:
        - logical-name: hed1
          type: simple-port
          bus-address: "0000:01:01.0"
        - logical-name: hed2
          type: simple-port
          bus-address: "0000:01:02.0"
    - name: HOS-VIRTUAL-EIGHT-PORT
      physical-ports:
        - bus-address: '0000:01:01.0'
          logical-name: hed1
          type: simple-port
        - bus-address: '0000:01:02.0'
          logical-name: hed2
          type: simple-port
        - bus-address: '0000:01:03.0'
          logical-name: hed3
          type: simple-port
        - bus-address: '0000:01:04.0'
          logical-name: hed4
          type: simple-port
        - bus-address: '0000:01:05.0'
          logical-name: hed5
          type: simple-port
        - bus-address: '0000:01:06.0'
```

```
logical-name: hed6
type: simple-port
- bus-address: '0000:01:07.0'
logical-name: hed7
type: simple-port
- bus-address: '0000:01:08.0'
logical-name: hed8
type: simple-port
```

## Recommended Mappings

It is recommended that the nic-mappings for HLM Virtual-Controllers use the PCI-bus addresses of the form "0000:01:XX.0", where XX is the value 01..1f (1..31 in hexadecimal).

This style isolates the virtual NIC interfaces onto PCI bus 01 and should prevent bus-address clashes with other PCI devices on the system.

## Notification of actions to be taken post upgrade

As part of an upgrade play actions that need to be taken - that cannot be automatically done since sharing the host with other vms will be placed in **/var/run/hos**.

The actions that are currently notified are

```
/var/run/hos/reload_apparmor
/var/run/hos/reload_libvirtd
/var/run/hos/restart_libvirtd
```

## Reboot of a HLM-Hypervisor node

Rebooting a HLM-Hypervisor node follows the standard procedure in the HPE Helion OpenStack documentation .

The sequence here is to target the HPE Helion OpenStack vms first then the host itself - then shutdown the vms before reboot. Note that there is an ansible host group in verb hosts called <hlm-hypervisor node name>-vms which targets the vms on the node with name of hlm-hypervisor-node-name. In this case **liam-vcp-cp1-vmf-m1-mgmt-vms**

```
ansible-playbook -i hosts/verb_hosts --limit liam-vcp-cp1-vmf-m1-mgmt-vms hlm-stop.yml
ansible-playbook -i hosts/verb_hosts --limit liam-vcp-cp1-vmf-m1-mgmt hlm-stop.yml
ansible-playbook -i hosts/verb_hosts --limit liam-vcp-cp1-vmf-m1-mgmt hlm-hypervis
```

### reboot node

```
ansible-playbook -i hosts/verb_hosts --limit liam-vcp-cp1-vmf-m1-mgmt hlm-hypervis
ansible-playbook -i hosts/verb_hosts --limit liam-vcp-cp1-vmf-m1-mgmt-vms hlm-star
```

```
ansible-playbook -i hosts/verb_hosts --limit liam-vcp-cp1-vmf-m1-mgmt hlm-start.y
```

# Staged install on HLM-Hypervisor individual play books.

This is a call out of the set of various playbooks to allow for a staged deployment of the hlm-hypervisor to allow for NFV vms to be installed as well at an appropriate time.

Once the node has been installed and the input model created from the NFV seed VM, the next stage is to configure this node and deploy the other nodes in the control plane.

The install of the other BM nodes can follow standard practice of running the cobbler deploy and the bm-reimage playbooks - then you need to run the osconfig playbook and hlm-hypervisor on these nodes.

This brings up the networking on the nodes and deploys a minimal set of packages to allow for virtual machine deployment. This is done by

```
ansible-playbook -i hosts/verb_hosts --limit hlm-hypervisors osconfig-run.yml  
ansible-playbook -i hosts/verb_hosts --limit hlm-hypervisors hlm-hypervisors-deploy
```

At this stage you now have the nodes up with the networking configured and the basic set of packages installed to allow for subsequent deploy of the NFV vms.

Once this is done we then would need to deploy the HPE Helion OpenStack vms - this can be done with

```
ansible-playbook -i hosts/verb_hosts --limit hlm-hypervisors hlm-hypervisor-vms-deploy
```

Note that these 3 plays have been encapsulated into a single play that can be run with

```
ansible-playbook -i hosts/verb_hosts hlm-hypervisor-setup.yml
```

At this point you now have all the nodes to be able to deploy the cloud - this can now be achieved by running

```
ansible-playbook -i hosts/verb_hosts site.yml
```

# Virtual Controller Replacement

There are several use cases here, all related to repair or replacement of VCP infrastructure. Essentially in each case you start by recovering the hypervisor (if necessary) and then create new VMs to replace the ones that were lost. Once they complete `hlm-hypervisor-vms-deploy.yml`, it is the equivalent of having just completed `bm-reimage.yml` for a physical node replacement, and can proceed with the existing instructions for controller recovery.

1. Repair or replacement of a single bad VM. One that has been damaged or lost, not just a simple reboot.
2. Of a single hypervisor, not causing e.g. RabbitMQ to lose quorum or similar effects, because all the VMs on this host are part of 3-node clusters and the other nodes are still up.
3. Disaster recovery - loss of all hypervisors (and therefore control plane VMs)

## Use case 1 - single bad VM.

```
ansible-playbook -i hosts/verb_hosts --limit hlm-hypervisors osconfig-run.yml  
ansible-playbook -i hosts/verb_hosts --limit hlm-hypervisors hlm-hypervisors-deploy
```

```
ansible-playbook - i hosts/verb_hosts --limit hlm-hypervisors hlm-hypervisor-vms-d
```

### Use case 2 - single lost hypervisor but services are still running on the rest of the control plane

This starts with recovering the hypervisor, which is very similar to recovering a dead physical controller.

1. You'll need to update servers.yml and cobbler if the node's mac address or ilo details have changed (because of physical repairs/replacements).
2. Reimage just this node and then run the hypervisor plays on it

```
ansible-playbook -i hosts/localhost bm-reimage.yml --limit=hypervisor1  
ansible-playbook -i hosts/verb_hosts --limit hypervisor1 osconfig-run.yml  
ansible-playbook -i hosts/verb_hosts --limit hypervisor1 hlm-hypervisors-deploy.  
ansible-playbook -i hosts/verb_hosts --limit hypervisor1 hlm-hypervisor-vms-dep
```

At this stage, the VMs are up with a bare operating system, and you should follow the physical controller recovery instructions from here.

### Use case 3 - lost all hypervisors and thus the complete control plane

Currently this use case is not correctly documented, even for a completely physical system - see DOCS-2921 [<https://jira.hpccloud.net/browse/DOCS-2921>]. However once there is a working procedure for physical it could be adapted using similar techniques to the above.

## Ansible host-specific variables for network-group access

From a 3rd-party view-point there are two important scenarios for accessing the network:

- 3rd-Party service on a HLM managed Server.
- 3rd-Party VM on a HLM-Hypervisor Server.

Each requires different sub-sets of data to establish the network configuration and potentially the end-point to which they need to attach. The following table indicates the set of data that is made available, and the scenarios that may find this data useful. The network-group specifications are provided under an attribute: host .my\_network\_groups : containing a list of dicts, one per network-group.

|  |                       |                    |                 | Consumed by 3rd-Party scenario |                   |                                                 |
|--|-----------------------|--------------------|-----------------|--------------------------------|-------------------|-------------------------------------------------|
|  | Index                 | Attribute          | Value           | 3rd-Party VM                   | 3rd-Party Service | Comment                                         |
|  | <net-work-group-name> |                    |                 | +                              | +                 | name of the net-work-group                      |
|  |                       | net-work-name      | <net-work-name> | +                              | +                 | name of the network                             |
|  |                       | passthrough-device | bridge-name>    | +                              |                   | open-vswitch-bridge to attach to for unfiltered |

|  |       |             |              | Consumed by 3rd-Party scenario |                   |                                                |
|--|-------|-------------|--------------|--------------------------------|-------------------|------------------------------------------------|
|  | Index | Attribute   | Value        | 3rd-Party VM                   | 3rd-Party Service | Comment                                        |
|  |       | tagged-vlan | true/false   | +                              |                   | access to the specified network                |
|  |       | vlanid      | <value>      | +                              |                   | is this network running tagged                 |
|  |       | device      | <dev-name>   |                                | +                 | which vlanid: is it tagged with                |
|  |       | address     | <ip-address> |                                | +                 | Device used for local-services                 |
|  |       | cidr        | <cidr>       | +                              | +                 | Address only used for local-services           |
|  |       | gate-way-ip | <ip-address> | +                              | +                 | CIDR in use by this network                    |
|  |       | mtu         | <mtu>        | +                              | +                 | gateway-ip in use on this network <optional>   |
|  |       |             |              |                                |                   | the MTU specified or inherited on this network |

For Example:

```

host:
  ...
my_network_groups:
  BLS:
    -
      address: 10.244.47.102
      cidr: 10.244.47.0/24
      device: br-vlan3537
      gateway-ip: 10.244.47.1
      network-name: BLS-NET
      passthrough-device: br-bond0
      tagged-vlan: true
      vlanid: 3537
  CAN:
    -
      cidr: 10.244.45.0/24
      device: br-bond0
      gateway-ip: 10.244.45.1
      network-name: CAN-NET
      passthrough-device: br-bond0

```

```

tagged-vlan: true
vlanid: 3535
CLM:
-
  address: 10.244.44.102
  cidr: 10.244.44.0/24
  device: vlan3534
  gateway-ip: 10.244.44.1
  network-name: CLM-NET
  passthrough-device: br-bond0
  tagged-vlan: true
  vlanid: 3534
PXE:
-
  address: 10.244.43.102
  cidr: 10.244.43.0/24
  device: br-bond0
  gateway-ip: 10.244.43.1
  network-name: PXE-NET
  passthrough-device: br-bond0
  tagged-vlan: false
  vlanid: 3533
VLAN1-TUL:
-
  device: br-bond1
  network-name: VLAN1-TUL-NET
  passthrough-device: br-bond1
  tagged-vlan: false
VLAN2-TUL:
-
  device: br-bond1
  network-name: VLAN2-TUL-NET
  passthrough-device: br-bond1
  tagged-vlan: false
VxLAN-TUL:
-
  cidr: 10.244.48.0/24
  device: br-bond1
  gateway-ip: 10.244.48.1
  network-name: VxLAN-TUL-NET
  passthrough-device: br-bond1
  tagged-vlan: false
  vlanid: 3538

```

Which decodes to mean:

- BLS: available locally (address: 10.244.47.102) and passthrough to hosted-VMs (passthrough-device: br-bond0, vlanid: 3537, cidr:, gateway: )
- CAN: not-available locally, but passthrough to Hosted-VMs (passthrough-device: br-bond0, vlanid: 3535, cidr:, gateway: )
- CLM: available locally (address: 10.244.44.102) and passthrough to hosted-VMs (passthrough-device: br-bond0, vlanid: 3534, cidr:, gateway: )
- PXE: available locally (address: 10.244.43.1102) and passthrough to hosted-VMs (passthrough-device: br-bond0, vlanid: 3533, cidr:, gateway: )
- VLAN1-TUL: not available locally, but passthrough to Hosted-VMs (passthrough-device: br-bond1, tagged-vlan: false)

- VLAN2-TUL: not available locally, but passthrough to Hosted-VMs (passthrough-device: br-bond1, tagged-vlan: false)
- VxLAN-TUL: not available locally, but passthrough to Hosted-VMs (passthrough-device: br-bond1, tagged-vlan: false, cidr:, gateway:)

A 3rd-Party application can provide an ansible-playbook that processes the specified set of ansible-vars and extracts the information required to support the configuration of the 3rd-party application.

## Example 3rd-Party Ansible for Network Configuration Access

The following is an example ansible playbook to find the name of the device associated with the MANAGEMENT network-group. It can be easily customised to select any known network and any attribute of that network shown in the table above.

```
#  
# (c) Copyright 2016 Hewlett Packard Enterprise Development LP  
#  
# Licensed under the Apache License, Version 2.0 (the "License"); you may  
# not use this file except in compliance with the License. You may obtain  
# a copy of the License at  
#  
# http://www.apache.org/licenses/LICENSE-2.0  
#  
# Unless required by applicable law or agreed to in writing, software  
# distributed under the License is distributed on an "AS IS" BASIS, WITHOUT  
# WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the  
# License for the specific language governing permissions and limitations  
# under the License.  
#  
# Example playbook to interrogate and recover network specific  
# configuration data for Third-Party applications  
- name: third-party | network-details | display all available network groups  
  debug:  
    var: host.my_network_groups  
- name: third-party | network-details | select a network group  
  set_fact:  
    tp_network_group_name: 'MANAGEMENT'  
- name: third-party | network-details | display which network group we're interrogating  
  debug:  
    var: tp_network_group_name  
- name: third-party | network-details | get the details of the selected network group  
  set_fact:  
    tp_network_group_data: "{{ host.my_network_groups | item(tp_network_group_name) }}"  
- name: third-party | network-details | display the details of the selected network group  
  debug:  
    var: tp_network_group_data  
- name: third-party | network-details | get the 'device' for the selected network group  
  set_fact:  
    tp_network_group_device: "{{ tp_network_group_data[0] | item('device') }}"  
- name: third-party | network-details | display the 'device' for the selected network group  
  debug:
```

```
var: tp_network_group_device
```

## Ansible host-specific variables for gluster

If disks have been set aside for gluster in the hypervisor input model as outlined earlier, then a list of these disks can be found in `host.my_device_groups.gluster.devices`, for example:

```
host:
  my_device_groups:
    gluster:
      - consumer:
          name: gluster
          suppress_warnings: true
        devices:
          - name: /dev/sde
          - name: /dev/sdf
      name: gluster
```

## "Empty" HLM Hypervisor Node

In order to create an empty hlm-hypervisor node then specifying `hlm-hypervisor: True` is necessary.

```
- id: hypervisor1
  ip-addr: 10.225.107.74
  role: HLM-HYPERVISOR-ROLE
  server-group: RACK2
  hlm-hypervisor: True
  nic-mapping: HP-BL460-6PORT
  mac-addr: 9c:7e:96:22:9e:d8
  ilo-ip: 10.1.18.42
  ilo-password: whatever
  ilo-user: whatever
```

In addition, if these nodes are being used in scenarios where there are no HPE Helion OpenStack servers to consume log data etc., you will need to ensure that in the input model the only `common-service-components` would be the `lifecycle-manager-target`.

## Monasca Monitoring of HLM Hypervisors

If your build includes a top-level play `hlm-hypervisors-monitoring-deploy.yml` then you have the option of enabling Monasca monitoring for the HLM Hypervisor nodes by running that playbook as follows:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts hlm-hypervisor-monitoring-deploy.yml
```

## Monitored Metrics

Currently two types of metrics are gathered:

1. Overall summary state for a HLM Hypervisor node - `hlm-hypervisor.vcp_vms`

- a. Reports 0 indicating healthy so long as all monitored VMs, and associated networks, are running/active.
  - b. If any of the VMs or associated networks are not running/active, reports a value of 1.
  - c. Associated additional dimensions:
    - i. service: hlm-hypervisor
    - ii. component: vcp
2. Summary state for each VCP VM running on a HLM Hypervisor node - `hlm-hypervisor.vcp_vm.<vm_name>`
    - a. Reports 0 indicating health so long as a VM, and its associated networks, are running/active.
    - b. If the VM or any of its networks are not running/active, reports a value of 1.
    - c. Associated additional dimensions:
      - i. service: hlm-hypervisor
      - ii. component: vcp\_vm
      - iii. domain: `<vm_name>`

## Configured Alarms

Alarms are configured to track these metrics; an alarm will be triggered if the metric starts reporting a state of 1.

If an alarm is triggered the metric's measurement value\_meta will include a detail section outlining the detected issues.

## HLM Hypervisor Monitoring Configuration

The monitoring mechanism is driven by per-VM JSON configuration files in `/etc/hlm-hypervisor/vms` with the following format:

```
{  
    "hhv_vm_config": {  
        "name": "<hostname>" ,  
        "domain": "<vm_name>" ,  
        "networks": [  
            "<vnet_1>" ,  
            "<vnet_2>"  
        ] ,  
        "dimensions": {  
            "service": "hlm-hypervisor" ,  
            "component": "vcp_vm" ,  
            "domain": "<vm_name>"  
        } ,  
        "check_type": "vm"  
    }  
}
```

The HLM Hypervisor Monasca detection plugin `hhv.py`, exposes two classes, `HLMHypervisorSummary` and `HLMHypervisorVMs`, which respectively generate the appropriate Monasca configuration to support both types of metric.

The HLM Hypervisor Monasca check plugin, `hhv.py`, processes the configuration generated by the detection plugin and updates the relevant metrics.

---

# Chapter 17. Integrations

Once you have completed your cloud installation, these are some of the common integrations you may want to perform.

## Ceph Overview

### Overview

HPE Helion OpenStack® 5.0 supports the Hammer version of Ceph cluster. Ceph cluster is a distributed object storage solution that can scale horizontally up to multiple petabytes and is based on the Reliable Autonomic Distributed Object Store (RADOS) object-based storage system. Ceph cluster stores all data as objects. It also provides components that support other storage protocols. For example: RBD (RADOS Block Devices) supports block storage protocol, RADOS Gateway support S3/Swift API protocol for object storage access and so on. For example, RBD (RADOS Block Devices) supports the block storage protocol, RADOS Gateway supports the S3/Swift API protocol for object storage access, and so on. The following are Ceph's key features:

1. Uses of any commodity hardware.
2. Scales horizontally up to petabytes (there is no theoretical limit).
3. Self healing, self managing.
4. No single point of failure.
5. Supports various client protocol for block device access, object storage access using the S3/Swift API and more.

Ceph deployment is flexible. Although it supports a variety of storage protocols based on deployed components, you can add extra components (except mandatory ones) *only* if you need to support specific storage protocols. For example, if you are not going to support the S3/Swift protocol, then you might choose not to deploy the RADOS Gateway. For easy readability, the various aspects of cluster deployment, such as hardware configuration, service configuration parameters, and deployment architecture, are categorized into separate sections. Each major section is further categorized into following segments:

- Core Ceph
- RADOS Gateway

Each sub-section provides recommendations vs. supported configurations. We recommend the production configuration, although all supported configurations are tested and validated. Use alternative supported configurations only if you have a strong need for them after evaluating their pros and cons.

#### Core Ceph

Core Ceph has two primary components, Object Storage Daemon (OSD) and Monitor, which are mandatory for cluster function. The following table briefly describes these components.

| Components | Description                                                                                                                                                                                                 |
|------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| OSD        | A Ceph OSD Daemon (OSD) stores data, handles data replication, recovery, backfilling, and rebalancing, and provides monitoring information to Ceph Monitors by checking other Ceph daemons for an activity. |

| Components     | Description                                                                                                                                                                                                                                                     |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Monitor</b> | Ceph Monitor maintains maps of the cluster state including the monitor map, the OSD map, the placement group (PG) map, and the CRUSH map. It also maintains a history (called an "epoch") of each state change in the Ceph Monitors, Ceph OSD Daemons, and PGs. |

### RADOS Gateway

The RADOS Gateway service is an object storage interface that enables user to perform HTTP-based CRUD operations on an object. It supports both the OpenStack Swift and Amazon S3 REST APIs. It has the following two types of users, unlike the rest of OpenStack services, which rely completely on Keystone for user management (see more details at ).

1. Keystone
2. RADOS Gateway user (managed by Ceph itself and does not require Keystone)

RADOS Gateway is an optional component. Deploy it only if you need to access objects storage functionality using Swift or S3 API. Features include the following:

1. RADOS Gateway is configured to run in a simple (non-federated or single region) mode.
2. The HAProxy on the HPE Helion OpenStack® controller node acts as a load balancer for RADOS Gateway servers (in least connection mode, the load balancer selects the server with the least number of connections) for RADOS Gateway servers.
3. Provides OpenStack Keystone integration (users having a configured set of roles can access Swift APIs served by radosgw).
4. The default HPE Helion OpenStack® configuration installs RADOS Gateway on standalone nodes.
5. Access to Amazon S3 APIs is limited to RADOS Gateway users.
6. The RADOS Gateway external and internal endpoints are SSL/TLS- enabled, including the public end points represented by HAProxy.

## Deployment Architecture

Consider the following issues when deploying Ceph:

- Ceph networking
- Placement of service components (such as OSD, monitor, RADOS Gateway) across nodes. For example, RADOS Gateway and monitor can be deployed on standalone nodes or together.

## Ceph Networking

Ceph clients transmits traffic directly to OSD daemons for storage operations instead of the client routing requests to a specific gateways. OSD daemons perform data replication and participate in recovery activities. In general, a storage pool is configured with a replica count of 3, causing daemons to transact three sets of client data over the cluster network. As a result, 4 MB of write traffic results in a total of 12 MB of data movement in the Ceph clusters (4 MB \* 3 replicas = 12MB). Also, Ceph clusters routinely share data among themselves. It is important to segregate Ceph data traffic into three segments, as follows:

- Management traffic, which includes monitoring and logging.
- Client traffic (often termed as data traffic) includes client requests sent to OSD daemons.
- Cluster traffic (often termed as replication traffic), which includes replication and recovery data traffic among OSD daemons.

For a high-performance clusters, a proper network configuration is very important. Use multiple networks for different data traffics. For a cluster of a reasonable size (a few terabytes), we recommend having a cluster with at least two networks, such as a single network for management and client data traffic (front-side) and a cluster (back-side) network. For large Ceph clusters, we recommend segregating all three traffic configurations. Segregation enables helps make for a secure connection because a cluster network is not required to be connected to the Internet directly. It allows OSD daemons to keep communicating without intervention so that placement groups can be brought to active and clean states relatively easily, whenever required. Apart from network separation (using VLANs), be sure to consider NICs used for Ceph servers too. We strongly recommend using bonded NICs to prevent single points of failure. Note that the network (and hence VLAN) separation is different from NIC separation even though they are linked to each other. In a multi-network model, emphasize VLAN separation, which ideally should be complemented by NIC separation, using the following priority order:

1. Separation of VLAN
2. Separation of NIC for VLANs.

The number of NICs you have will depend on your number of bonded interfaces, as follows:

1. Three bonded interfaces with six NICs: first for management, second for OSD client, and third for OSD internal (preferred setup).
2. Two bonded interfaces with four NICs: management VLANs hooked to the first bonded interface, and OSD networks hooked to the second bonded interface (most-used setup).
3. Only one bonded interface: two NICs for all VLANs (for a few NICs only).

HPE Helion OpenStack® Ceph software offers significant flexibility when defining and deploying OpenStack-based clouds, letting you implement a wide variety of different Ceph configurations. This allows you to design, model, and deploy cloud based on your requirements. You have the following VLAN choices based on traffic types:

1. Single: A single VLAN is used for all traffic, meant primarily for a small cluster.
2. Double deployment: One VLAN is used for cloud management and client traffic and another VLAN is used for Ceph internal traffic.
3. Triple deployment: Separate VLANs are used for management, client, and internal traffic.

As mentioned previously, you can link all VLANs to the same bonded interface, to a separate bonded interface, or a combination of interfaces. For a large cluster, you can use a separate bonded interface which in turn necessitates having at least six NICs for OSD nodes and four NICs for monitor nodes.

## Placement of service component

Although you can place service components in multiple ways, this section focuses on recommended deployments only, covering Core Ceph (OSD and monitors) and the RADOS Gateway. For details on alternative supported deployment architecture, please refer to .

- Core Ceph (i.e. OSD and monitors)
- Rados Gateway
- **Core Ceph**

We recommend the following deployment composition is recommended to avoid single points of failure for the Ceph cluster deployment.

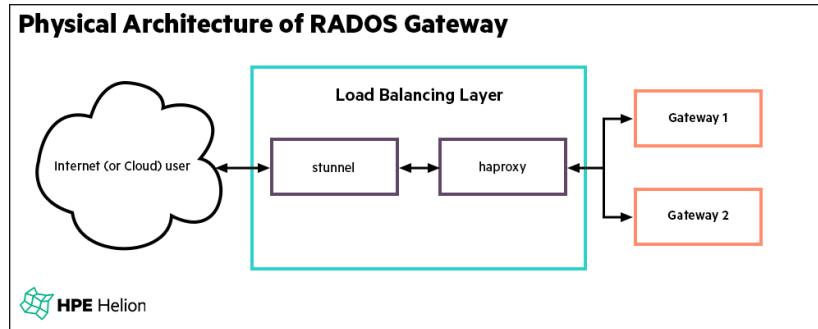
1. Three monitor nodes, to retain the odd number criteria of the monitor quorum.
2. At least three OSD nodes, to ensure that the object is replicated on three separate physical nodes, if the replica count of the pool is set to three (the recommended storage pool configuration is to set the replica count to three).

As mentioned in , we recommend using separate VLANS to separate Ceph traffic. The cloud management network is used for logging and/or monitoring, the OSD client network is used for Ceph client traffic, and the OSD internal network is used for internal Ceph traffic, such as replication.

- **RADOS Gateway**

We recommend deploying at least two instances of RADOS Gateway on a standalone node front- ended by HAProxy.

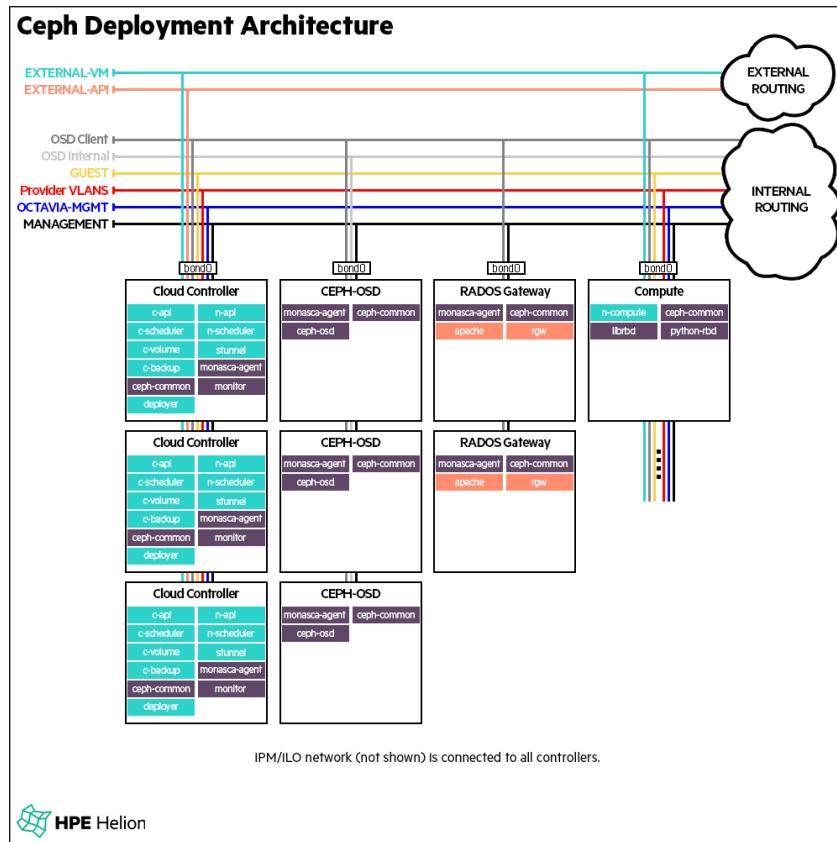
The following diagram illustrates the physical architecture of the RADOS Gateway and using the entry-scale-kvm-ceph configuration.



The default example model (entry-scale-kvm-ceph) is the recommended mechanism to deploy the Ceph cluster with the RADOS Gateway.

## Ceph Deployment Architecture

The following diagram illustrates the Ceph deployment architecture.



The preceding diagram illustrates the Ceph deployment scenario of `entry-scale-kvm-ceph` input model. Monitors are deployed on three controller nodes. Three standalone nodes are used for OSD and two standalone nodes are used for the RADOS Gateway, front-ended by HAProxy running on controller nodes. Management, client, and internal data traffic is separated using independent VLANs.

## Alternative supported architecture

HPE Helion OpenStack® 5.0 also supports an alternate deployment architecture, for which the following points are relevant.

- Core networking
- Placement of service components
  - Core Ceph
  - Monitor on standalone node
  - RadosGateway
    - Co-hosted on cluster nodes hosting monitor service components
    - Co-hosted on controller nodes

## Core networking

Ceph clients transmit traffic directly to OSD daemons for storage operations instead of the client routing requests to a specific gateway. For more information, refer to [\[link\]](#).

## Placement of service components

- **Core Ceph**

The architecture choices which is supported in HPE Helion OpenStack® 5.0 is as follows.

- Monitor on standalone node

You can deploy the Ceph monitor service on a one or more dedicated clusters or resource nodes. Ensure that you modify your environment after installing the lifecycle manager. For more details, refer to .

- **RADOS Gateway**

RADOS Gateway service can be co-hosted with other HPE Helion OpenStack® services as follows.

- Alternate RADOS Gateway deployment architecture choice
  - Co-hosted on cluster nodes hosting monitor service components
  - Co-hosted on controller nodes

The default `entry-scale-kvm-ceph` input model deploys the `radosgw` services on two dedicated cluster nodes. However, you can also install the RADOS Gateway on a cluster node hosting the Ceph monitor service or on controller nodes. Refer to or for more details.

### Note

Because the RADOS Gateway service shares server resources with multiple services, these alternate configurations will result in sub-optimal performance, as compared to the default configuration.

## Hardware recommendations

For hardware recommendations, refer to .

## Ceph Deployment and Configurations

HPE Helion OpenStack® 5.0 Ceph deployment leverages the cloud lifecycle operations supported by Helion lifecycle management. It provides the simplified lifecycle management of critical cluster operations such as service check, upgrade, and reconfiguring service components. This section assumes that you understand cloud input models and highlights only important aspects of cloud input models pertaining to Ceph. We focus on deployment aspects of the `entry-scale-kvm-ceph` input model, which is the most widely used configuration. You can see for the deployment of Ceph with various supported options. To ensure the proper deployment and verification of Ceph, it is important to read the topics and perform the steps in order. This section provides insight on how to alter the `entry-scale-kvm-ceph` input model to deploy Ceph with various supported options. We recommend that you deploy your supported choice only after evaluating all pros and cons.

1.
  - a. Define an OSD Disk Model for an OSD Disk
  - b. Customize Your Service Configuration
- 2.

3.

## Predeployment

Before you start deploying the HPE Helion OpenStack® cloud with Ceph, you must understand the following aspects of Ceph clusters,

- **Define an OSD Disk Model for an OSD Disk**

This section focus on expressing the storage requirements of an OSD (object-storage daemon) node. OSD nodes have system, data, and journal disks.

System disks are used for OSD components, logging, and other tasks. The configuration of data and journal disks is important for Ceph deployment. A sample disk model file for the entry-scale-kvm-ceph cloud is as follows.

```
---
product:
    version: 2

disk-models:
- name: OSD-DISKS
    # Disk model to be used for Ceph OSD nodes
    # /dev/sda_root is used as a volume group for /, /var/log and /var/crash
    # sda_root is a templated value to align with whatever partition is really used
    # This value is checked in os config and replaced by the partition actually used
    # on sda e.g. sdal or sda5

volume-groups:
- name: hlm-vg
    physical-volumes:
        - /dev/sda_root

logical-volumes:
    # The policy is not to consume 100% of the space of each volume group.
    # 5% should be left free for snapshots and to allow for some flexibility
        - name: root
            size: 30%
            fstype: ext4
            mount: /
        - name: log
            size: 45%
            mount: /var/log
            fstype: ext4
            mkfs-opt: -O large_file
        - name: crash
            size: 20%
            mount: /var/crash
            fstype: ext4
            mkfs-opt: -O large_file
consumer:
    name: os

    # Disks to be used by Ceph
    # Additional disks can be added if available
```

```
device-groups:
  - name: ceph-osd-data-and-journal
    devices:
      - name: /dev/sdc
    consumer:
      name: ceph
      attrs:
        usage: data
        journal_disk: /dev/sdd
  - name: ceph-osd-data-and-shared-journal-set-1
    devices:
      - name: /dev/sde
    consumer:
      name: ceph
      attrs:
        usage: data
        journal_disk: /dev/sdg
  - name: ceph-osd-data-and-shared-journal-set-2
    devices:
      - name: /dev/sdf
    consumer:
      name: ceph
      attrs:
        usage: data
        journal_disk: /dev/sdg
```

The disk model has the following parameters:

| Value                          | Description                                                                                                                                                                                                                 |
|--------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>device-groups</b>           | The name of the device group. There can be several device groups, which allows different sets of disks to be used for different purposes.                                                                                   |
| <b>name</b>                    | An arbitrary name for the device group. The name must be unique.                                                                                                                                                            |
| <b>devices</b>                 | A list of devices allocated to the device group. A name field containing <code>/dev/sdb</code> , <code>/dev/sdc</code> , <code>/dev/sde</code> , and <code>/dev/sdf</code> indicates that the device group is used by Ceph. |
| <b>consumer</b>                | The service that uses the device group. A name field containing <code>ceph</code> indicates that the device group is used by Ceph.                                                                                          |
| <b>attrs</b>                   | Attributes associated with the consumer.                                                                                                                                                                                    |
| <b>usage</b>                   | Devices for a particular service can have several uses. In the preceding sample, the <code>usage</code> field contains <code>data</code> , which indicates that the device is used for data storage.                        |
| <b>journal_disk [OPTIONAL]</b> | The disk to be used for storing journal data. When running multiple Ceph OSDs on a single node, a journal disk can be shared between OSDs of the node.                                                                      |

| Value | Description                                                                                                 |
|-------|-------------------------------------------------------------------------------------------------------------|
|       | If you do not specify this value, Ceph stores the journal on the OSD's data disk (in a separate partition). |

The preceding sample file represents the following:

- The first disk is used for OS and system purposes.
- There are three OSD data disks (sdc, sde, and sdf) and two journal disks (sdd and sdg). This configuration shows that we can share journal disks for multiple OSDs. It is recommended to use an OSD journal disk for four OSD data disks. See for more details.
- The drive type is not mentioned for the journal or data disks. You can consume any drive type but we **recommend** using an SSD (solid-state drive) for the journal disk.

Although the preceding model illustrates mixed use of the journal disk, we strongly advise that you keep journal data separate from OSD data, which means that your disk model **should not** have journal disks shared on the same data disks. For more information, see *Usage of Journal Disk*.

### Usage of Journal Disk

HPE Helion OpenStack® 5.0 recommends storing the Ceph OSD journal on an SSD and the OSD object data on a separate hard disk drive. SSD drives are costly, so it saves money to use multiple partitions in a single SSD drive for multiple OSD journals. We recommend not more than four or five OSD journals on each SSD disk as a reasonable balance between cost and optimal performance. If you have too many OSD journals on a single SSD, and the journal disk crashes, you might lose your data on those disks. Also, too many journals in a single SSD can negatively affect performance.

Using an OSD journal as a partition on the data disk itself is supported. However, you might see a significant decline in Ceph performance because each client request to store an object is first written to the journal disk before sending an acknowledgment to the client.

The Ceph OSD journal size defaults to 5120 MB (5 GB) in HPE Helion OpenStack® 5.0. This value can be changed, but it does not apply to any existing journal partitions. It will affect new OSDs created after the journal size is changed (whether the journal is on the same disk or a separate disk than the data disk). To change the journal size, edit the `osd_journal_size` parameter in the `~/helion/my_cloud/config/ceph/settings.yml` file.

To summarize:

1. Use SSD for the journal disk.
2. The ratio of OSD data disks to the journal disk is recommended to 4:1.
3. The default journal partition size is 5 GB, which you can change. Actual journal size depends upon your disk drive rpm and expected throughput. The formula is:

$$\text{OSD\_JOURNAL\_SIZE} = 2 * (\text{EXPECTED\_THROUGHPUT} * \text{FILESTORE\_MAX\_SYNC\_INTERVAL})$$

4. The journal size for previously configured OSD disks does not change even if you change the `osd_journal_size` parameter in the `~/helion/my_cloud/config/ceph/settings.yml` file. If you want to resize the journal partition of previously configured OSD disks, you should flush journal data, remove the OSD from the cluster, and then add it again.

- **Customize Your Service Configuration**

You must customize the parameters in the following files:

- Customize the parameters in the `~/helion/my_cloud/config/ceph/settings.yml` file.

HPE Helion OpenStack® makes it easy to configure service parameters. All common parameters are available in the `~/helion/my_cloud/config/ceph/settings.yml` file. You can deploy your cluster without altering any of the parameters but we advise that you review and understand the parameters before deploying your cluster. The following link will assist you in the understanding of CEPH placement-groups: <http://docs.ceph.com/docs/master/rados/operations/placement-groups/>. The following table provides details about parameters you can change and descriptions of those parameters.

### Core Service Parameters

| Parameter               | Description                                                                                                  | Default Value | Recommendation                                                                                                                                                                                                  |
|-------------------------|--------------------------------------------------------------------------------------------------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ceph_cluster            | The name of the Ceph clusters. The default value is Ceph.                                                    | Ceph          | Customize to suit your requirements.                                                                                                                                                                            |
| ceph_release            | The name of the Ceph release.                                                                                | hammer        | Do not change the default value.                                                                                                                                                                                |
| osd_pool_default_size   | The number of replicas for objects in the pool.                                                              | 3             | Do not lower the default value. The value can be increased to the maximum number of OSD nodes in the environment (increasing it beyond this limit will cause the cluster to never reach an active+clean state). |
| osd_pool_default_pg_num | The default number of placement groups for a pool. This value changes based on the number of OSDs available. | 128           | The value can be changed based on the number of OSD servers/nodes in the deployment. Refer to the Ceph PG calculator at <a href="http://ceph.com/pg-calc/">http://ceph.com/pg-calc/</a> to customize it.        |
| fstype                  | Storage filesystem type for OSDs.                                                                            | xfs           | Only the xfs file system is certified.                                                                                                                                                                          |
| zap_data_disk           | Zap partition table and contents of the disk.                                                                | True          | Not recommended to change the default value.                                                                                                                                                                    |
| persist_mountpoint      | Place to persist OSD data disk mount point.                                                                  | fstab         | Not recommended to change the default value (as it ensures that the OSD data disks are mounted automatically on a system reboot).                                                                               |

| Parameter               | Description                                                                                           | Default Value                    | Recommendation                                                                                                                                                                                                                                                            |
|-------------------------|-------------------------------------------------------------------------------------------------------|----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| osd_settle_time         | The time in seconds to wait for after starting/restarting Ceph OSD services.                          | 10 seconds                       | Increase this value only if the number of OSD servers is more than three or the servers have a slow network.                                                                                                                                                              |
| osd_journal_size        | The size of the journal in megabytes.                                                                 | 5120                             | You can increase this value to achieve optimal use of the journal disk (if it is shared between multiple OSDs).                                                                                                                                                           |
| data_disk_poll_attempts | The maximum number of attempts before attempting to activate an OSD for a new disk (default value 5). | 5                                | Increase this value only if the OSD data disk drives are under-performing or slower than expected. Because this parameter and <code>data_disk_poll_interval</code> (following) have a combined effect, we recommend that you consider both while tweaking either of them. |
| data_disk_poll_interval | The time interval in seconds to wait between <code>data_disk_poll_attempts</code> .                   | 12                               | You can customize this value to suit your requirements. However, because this parameter and <code>data_disk_poll_attempts</code> (preceding) have a combined effect, we recommend that you consider both while tweaking either of them.                                   |
| osd_max_open_files      | Maximum number of file descriptors for OSD.                                                           | 32768                            | Do not change the default value.                                                                                                                                                                                                                                          |
| mon_default_dir         | Directory to store monitor data.                                                                      | /var/lib/ceph/mon/<ceph_cluster> | Do not change the default value.                                                                                                                                                                                                                                          |
| mon_max_open_files      | Maximum number of file descriptors for monitor.                                                       | 16384                            | Do not change the default value.                                                                                                                                                                                                                                          |
| root_bucket             | Ceph CRUSH map root bucket name.                                                                      | default                          | Not recommended to change the default value.                                                                                                                                                                                                                              |

| Parameter | Description | Default Value | Recommendation                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------|-------------|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|           |             |               | <p>Changing this value affects CRUSH map and data placement. If you change the default value ensure to create a new rule set with a new root bucket and map the Ceph storage pools to use a new rule set.</p> <p>It is strongly recommended not to change this value post day-zero deployment. Changing the value after deployment results in a newly added OSD nodes to go to a newer <code>root_bucket</code>. It affects placement of the placement groups (data) of the storage pools.</p> <p>If you are upgrading cluster(s) from HPE Helion OpenStack® 3.0 to HPE Helion OpenStack® 5.0, you are strongly advised not to change the default value of <code>root_bucket</code>.</p> |

### RADOS Gateway (RGW) Parameters

| Parameter                                | Description                                                                                                       | Default Value                | Recommendation                                        |
|------------------------------------------|-------------------------------------------------------------------------------------------------------------------|------------------------------|-------------------------------------------------------|
| <code>radosgw_user</code>                | The name of the Ceph client user for <code>radosgw</code> .                                                       | <code>gateway</code>         | Customize to suit your requirements.                  |
| <code>radosgw_admin_email</code>         | The email address of the server administrator.                                                                    | <code>admin@hpe.com</code>   | Update the email address of the server administrator. |
| <code>rgw_keystone_service_type</code>   | <b>DEPRECATED</b><br>To configure RADOS Gateway before deployment refer to .                                      |                              |                                                       |
| <code>rgw_keystone_accepted_roles</code> | Only users having either of the roles listed here will be able to access the Swift APIs of <code>radosgw</code> . | <code>admin, _member_</code> | Do not change the default value.                      |

## Note

The default service password for the RADOS Gateway service can be modified by following the steps documented at .

### Configuring the RADOS Gateway service type in the Keystone catalog

#### Configuring the service type before deployment

1. You can configure the RADOS Gateway service type in Keystone catalog by replacing the `ceph-object-store` with desired value in `~/helion/hos/services/ceph/rgw.yml` file on the life cycle manager node.

```
advertises-to-services:  
  - service-name: KEY-API  
    entries:  
      - service-name: ceph-rgw  
        service-type: ceph-object-store  
        service-description: "Ceph Object Storage Service"  
        url-suffix: "/swift/v1"
```

2. After you have modified the **service-type**, commit the change to the local git repository.

```
cd ~/helion  
git checkout site  
git add ~/helion/hos/services/ceph/rgw.yml  
git commit -m "Updating the RADOS Gateway service type"
```

3. Rerun the configuration processor.

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

4. Rerun the deployment area preparation playbooks.

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

5. Run reconfiguration playbook in deployment area.

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts keystone-reconfigure.yml
```

### Configuring the RADOS Gateway service type after deployment

To update the RADOS Gateway service type in a deployed or a running cloud, you must delete the `ceph-rgw` service from the Keystone catalog and perform the same steps as mentioned in the preceding section () .

1. To delete the `ceph-rgw` service, you must know the service-id. Execute the following command from a controller node.

```
source ~/keystone.osrc  
openstack service list |grep ceph-rgw | awk '{print $2}'  
openstack service delete <service-id>
```

## Ceph client parameters

| Value                | Description                                                                                                                                             | Default Value | Recommendation                                                                                                                                                                                                                                                  |
|----------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pg_active_delay_time | The delay time for Ceph PGs to come into active state.                                                                                                  | 10            | You can increase this value if the number of OSD servers/nodes in the deployment is more than three. Because this parameter and <b>pg_active_retries</b> (following) have a combined effect, we recommend that you consider both while tweaking either of them. |
| pg_active_retries    | The number of retries for Ceph placement groups to come into active state with a duration of <code>pg_active_delay_time</code> seconds between entries. | 5             | You can customize this value to suit your requirements. However, because this parameter and <b>pg_active_delay_time</b> (preceding) have a combined effect, we recommend that you consider both while tweaking either of them.                                  |

- Customize parameters at `~/helion/hos/ansible/roles/_CEP-CMN/defaults/main.yml`.

The following table provides parameter descriptions. You can edit each parameter in the `main.yml` file.

| Value | Description                                                                                                                                                                                                            |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| fsid  | A unique identifier, File System ID, for the Ceph cluster that you should generate prior to deploying a cluster (use the <code>uuidgen</code> command to generate a new FSID). When set, this value cannot be changed. |

## Deploying Ceph

To deploy a new HPE Helion OpenStack® Ceph cloud using the default `entry-scale-kvm-ceph` model, follow these steps .

### Note

- In a multi-region cloud, Ceph can be deployed only as a shared service. In other words, Ceph services (Monitor, OSD, and RADOS Gateway servers) should be deployed in the shared control plane (such that those services will run in the same control plane as the Keystone service) ONLY.

- The non-shared control plane(s) need to ensure that the ceph-monitor service is specified in the **service\_components** section of uses and **imports** sections of the control plane definition.
- Ensure that all the regions that need access to Ceph have the shared control plane included.

## Edit Your Ceph Environment Input Files

Perform the following steps:

1. Log in to the lifecycle manager.
2. Copy the example configuration files into the required setup directory and edit them to contain the details of your environment:

```
cp -r ~/helion/examples/entry-scale-kvm-ceph/* ~/helion/my_cloud/definition/
```

Enter your environment information into the configuration files in the `~/helion/my_cloud/definition` directory.

You can find details of how to do this at the section called “Ring Specifications in the Input Model”.

3. Edit the `~/helion/my_cloud/definition/data/servers.yml` file and enter details. If you are using alternative RADOS Gateway deployments, see before editing the `servers.yml` file.

```
# Ceph OSD Nodes
- id: osd1
  ip-addr: 192.168.10.9
  role: OSD-ROLE
  server-group: RACK1
  nic-mapping: MY-2PORT-SERVER
  mac-addr: "8b:f6:9e:ca:3b:78"
  ilo-ip: 192.168.9.9
  ilo-password: password
  ilo-user: admin

- id: osd2
  ip-addr: 192.168.10.10
  role: OSD-ROLE
  server-group: RACK2
  nic-mapping: MY-2PORT-SERVER
  mac-addr: "8b:f6:9e:ca:3b:79"
  ilo-ip: 192.168.9.10
  ilo-password: password
  ilo-user: admin

- id: osd3
  ip-addr: 192.168.10.11
  role: OSD-ROLE
  server-group: RACK3
  nic-mapping: MY-2PORT-SERVER
  mac-addr: "8b:f6:9e:ca:3b:7a"
  ilo-ip: 192.168.9.11
  ilo-password: password
  ilo-user: admin

# Ceph RGW Nodes
```

```
- id: rgw1
  ip-addr: 192.168.10.12
  role: RGW-ROLE
  server-group: RACK1
  nic-mapping: MY-2PORT-SERVER
  mac-addr: "8b:f6:9e:ca:3b:62"
  ilo-ip: 192.168.9.12
  ilo-password: password
  ilo-user: admin

- id: rgw2
  ip-addr: 192.168.10.13
  role: RGW-ROLE
  server-group: RACK2
  nic-mapping: MY-2PORT-SERVER
  mac-addr: "8b:f6:9e:ca:3b:63"
  ilo-ip: 192.168.9.13
  ilo-password: password
  ilo-user: admin
```

The preceding sample file contains three OSD nodes and two RADOS Gateway nodes.

4. Edit the `~/helion/my_cloud/definition/data/disks_osd.yml` file to align the disk model to fit the server specification in your environment. For details on disk models, refer to .

Ceph service configuration parameters can be modified as described in the preceding section.

5. Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*):

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

6. After you set up your configuration files, continue with the installation procedure from .

## Note

For any troubleshooting information regarding the OSD node failure, see .

## Verifying Ceph Cluster Status

If you have deployed RADOS Gateway with core Ceph, then you need to ensure that all service components including RADOS Gateway are functioning as expected.

### Verify Core Ceph

Perform the following steps to check the status of the Ceph cluster:

1. Log in to the monitor node.
2. Execute the following command and make sure that the result is HEALTH\_OK or HEALTH\_WARN:

```
$ ceph health
```

Optionally, you can also set up the lifecycle manager as a Ceph client node (refer to ) and execute the preceding command from the lifecycle manager.

### Verify RADOS Gateway

To make sure that a Keystone user can access RADOS Gateway using Swift, perform the following steps:

1. Log in to a controller node.
2. Source the `service.osrc` file:

```
source ~/service.osrc
```

3. Execute the following command to generate a list of the containers associated with the user:

```
swift --os-service-type ceph-object-store list
```

If the containers are listed, this indicates that RADOS Gateway is accessible.

## Alternative Supported Choices

This section provides insight on how to alter the `entry-scale-kvm-ceph` input model to deploy Ceph with various supported options. We recommend that you deploy your supported choice only after evaluating all pros and cons. For technical details, please consult with the technical support team. The choices available can impact the performance and scaling of clusters. Choices are illustrated for reference purposes and you can combine one or more of them as needed. The content is categorized as follows:

1. Core Ceph

- 
- 
- 

2. RADOS Gateway

- 
- 
- 

- 3.

## Core Ceph

### Installing the Monitor Service on Standalone Nodes

The following section provides the procedure for installing the monitor service on standalone nodes instead of installing on controller nodes, as mentioned in `entry-scale-kvm-ceph`.

### Important

If you want to install the monitor service as a dedicated resource node, you must decide before deploying Ceph. HPE Helion OpenStack 5.0 does not support deployment transition. After Ceph is deployed, you cannot migrate the monitor service from controller nodes to dedicated resource nodes.

## Prerequisite

Perform the following steps to install the Ceph monitor on a dedicated node. Note that the Ceph requires at least three monitoring servers to form a cluster in case of a node failure.

1. Log in to the lifecycle manager.

2. Copy the `entry-scale-kvm-ceph` input model to the `~/helion/my_cloud/definition` directory before you begin the editing process:

```
cp -r ~/helion/examples/entry-scale-kvm-ceph/* ~/helion/my_cloud/definition/
```

3. Edit the `control_plane.yml` to create a new cluster, such as with `ceph-mon`, as shown here.

```
clusters:  
  - name: cluster1  
    cluster-prefix: c1  
    server-role: CONTROLLER-ROLE  
    member-count: 3  
    allocation-policy: strict  
    service-components:  
      - lifecycle-manager  
      - ntp-server  
      ...  
  
  - name: ceph-mon  
    cluster-prefix: ceph-mon  
    server-role: CEP-MON-ROLE  
    min-count: 3  
    allocation-policy: strict  
    service-components:  
      - ntp-client  
      - ceph-monitor  
  
  - name: rgw  
    cluster-prefix: rgw  
    server-role: RGW-ROLE  
    ...
```

4. Edit the `~/helion/my_cloud/definition/data/servers.yml` file to define the Ceph monitor node (monitor services). The following example shows three nodes for monitor services. We recommend using an odd number of monitor nodes.

```
# Ceph Monitor Nodes  
- id: ceph-mon1  
  ip-addr: 10.13.111.141  
  server-group: RACK1  
  role: CEP-MON-ROLE  
  nic-mapping: MY-4PORT-SERVER  
  mac-addr: "f0:92:1c:05:69:10"  
  ilo-ip: 10.12.8.217  
  ilo-password: password  
  ilo-user: admin  
  
- id: ceph-mon2
```

```
ip-addr: 10.13.111.142
server-group: RACK2
role: CEP-MON-ROLE
nic-mapping: MY-4PORT-SERVER
mac-addr: "83:92:1c:55:69:b0"
ilo-ip: 10.12.8.218
ilo-password: password
ilo-user: admin

- id: ceph-mon3
  ip-addr: 10.13.111.143
  server-group: RACK3
  role: CEP-MON-ROLE
  nic-mapping: MY-4PORT-SERVER
  mac-addr: "d9:92:1c:25:69:e0"
  ilo-ip: 10.12.8.219
  ilo-password: password
  ilo-user: admin

# Ceph RGW Nodes
- id: rgw1
  ...

```

5. Edit the `~/helion/my_cloud/definition/data/net_interfaces.yml` file to define a new network interface set for your Ceph monitors, as shown here.

```
## This defines the interface used for management
## traffic such as logging, monitoring, etc.
- name: CEP-MON-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed1
        provider: linux
        devices:
          - name: hed1
          - name: hed2
      network-groups:
        - MANAGEMENT

- name: RGW-INTERFACES
  network-interfaces:
    ...

```

6. Edit `~/helion/my_cloud/definition/data/disks_ceph_monitor.yml` to define the disk model for monitor nodes.

```
disk-models:
- name: CEP-MON-DISKS
  # Disk model to be used for Ceph monitor nodes
```

```
# /dev/sda_root is used as a volume group for /, /var/log and /var/crash
# sda_root is a templated value to align with whatever partition is really used
# This value is checked in os config and replaced by the partition actually used
# on sda e.g. sdal or sda5

volume-groups:
  - name: hlm-vg
    physical-volumes:
      - /dev/sda_root

logical-volumes:
  # The policy is not to consume 100% of the space of each volume group.
  # 5% should be left free for snapshots and to allow for some flexibility.
  - name: root
    size: 30%
    fstype: ext4
    mount: /
  - name: log
    size: 45%
    mount: /var/log
    fstype: ext4
    mkfs-opt: -O large_file
  - name: crash
    size: 20%
    mount: /var/crash
    fstype: ext4
    mkfs-opt: -O large_file
consumer:
  name: os
```

7. Edit the `~/helion/my_cloud/definition/data/server_roles.yml` file to define a new server role for your Ceph monitors:

```
- name: CEP-MON-ROLE
  interface-model: CEP-MON-INTERFACES
  disk-model: CEP-MON-DISKS
```

8. Commit your configuration:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "<commit message>"
```

9. Run the following playbook to add your nodes into Cobbler:

```
cd ~/helion/hos/ansible/
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

10. To reimage all the nodes using PXE, run the following playbook:

```
cd ~/helion/hos/ansible/
ansible-playbook -i hosts/localhost bm-reimage.yml
```

11. Run the configuration processor:

```
cd ~/helion/hos/ansible/
```

```
ansible-playbook -i hosts/localhost config-processor-run.yml
```

12.Update your deployment directory with this playbook:

```
cd ~/helion/hos/ansible/
ansible-playbook -i hosts/localhost ready-deployment.yml
```

13.Deploy these changes:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts site.yml
```

### Using a Single VLAN for All Ceph Traffic (Management, Client, and Internal OSD)

You can use a single VLAN to transmit all Ceph traffic. This configuration is recommended for a small cluster deployment.

Perform the following steps to configure Entry-scale-kvm-ceph-single-network.

1. Log in to the lifecycle manager.
2. Copy the entry-scale-kvm-ceph input model to the ~/helion/my\_cloud/definition directory before you begin the editing process:

```
cp -r ~/helion/examples/entry-scale-kvm-ceph/* ~/helion/my_cloud/definition/
```

3. Validate that NIC interfaces are correctly specified in nic\_mapping.yml for servers that are used in the cloud.
4. Ensure that you have at least two NICs for Ceph nodes to create a bonded interface for it.

5. Validate that your servers are mapped to a correct NIC interface specification in servers.yml. The following is an example of a server node used for OSD deployment:

```
# Ceph OSD Nodes
  - id: osd1
    ip-addr: 192.168.10.9
    role: OSD-ROLE
    server-group: RACK1
    nic-mapping: MY-2PORT-SERVER
    mac-addr: "8b:f6:9e:ca:3b:78"
    ilo-ip: 192.168.9.9
    ilo-password: password
    ilo-user: admin
```

6. Delete the OSD-INTERNAL and OSD-CLIENT network groups from network\_groups.yml. This is necessary because only the management network is used for Ceph traffic, thus OSD-INTERNAL and OSD-CLIENT network groups are not required.

7. Define net\_interfaces.yml to use only management network groups, as shown here.

```
- name: CONTROLLER-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
```

```
bond-data:
  options:
    mode: active-backup
    miimon: 200
    primary: hed3
  provider: linux
  devices:
    - name: hed3
    - name: hed4
network-groups:
  - EXTERNAL-API
  - EXTERNAL-VM
  - GUEST
  - MANAGEMENT

- name: COMPUTE-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
  bond-data:
    options:
      mode: active-backup
      miimon: 200
      primary: hed3
    provider: linux
    devices:
      - name: hed3
      - name: hed4
  network-groups:
    - EXTERNAL-VM
    - GUEST
    - MANAGEMENT

- name: OSD-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
  bond-data:
    options:
      mode: active-backup
      miimon: 200
      primary: hed3
    provider: linux
    devices:
      - name: hed3
      - name: hed4
  network-groups:
    - MANAGEMENT

- name: RGW-INTERFACES
  network-interfaces:
    - name: BOND0
```

```
device:
  name: bond0
bond-data:
  options:
    mode: active-backup
    mimon: 200
    primary: hed3
  provider: linux
  devices:
    - name: hed3
    - name: hed4
network-groups:
  - MANAGEMENT
```

8. Delete VLAN information for OSD-INTERNAL-NET and OSD-CLIENT-NET from `networks.yml`. Only Management VLANs are used.
9. After you set up your configuration files, perform in .

### Using Two VLANs: For Management and Client Traffic and for Internal OSD Traffic

You can use dual VLANs to transmit Ceph traffic. In this configuration one VLAN transmits management and client traffic and the other VLAN transmits internal OSD traffic. A separate bonded interface for two VLANs is used with four NICs. This configuration provides two aspects:

- Use of two networks, such as VLANs.
- Use of separate bonded interfaces for each VLAN (different from what is provided in `entry-scale-kvm-ceph`).

The use of separate NICs segregates traffic at the interface level and requires your server to have at least four NICs. But using a separate bonded interface for each VLAN is not mandatory, and thus you can use a single bonded interface (or server with only two NICs) for Ceph deployment.

Perform the following steps to configure `Entry-scale-kvm-ceph-dual-network`.

1. Log in to the lifecycle manager.
2. Copy the `entry-scale-kvm-ceph` input model to the `~/helion/my_cloud/definition` directory before you begin the editing process:

```
cp -r ~/helion/examples/entry-scale-kvm-ceph/* ~/helion/my_cloud/definition/
```

3. Validate that NIC interfaces are correctly specified in `nic_mapping.yml` for servers that are used in the cloud. For Ceph OSD nodes, four port servers are required. You can use **HP-DL360-4PORT** as it is defined in `nic_mapping.yml` of `entry-scale-kvm-ceph` or define a new NIC mapping (as shown here) for new sets of servers having four port servers.

```
- name: HP-4PORT-SERVER
  physical-ports:
    - logical-name: hed1
      type: simple-port
      bus-address: "0000:07:00.0"
    - logical-name: hed2
```

```
    type: simple-port
    bus-address: "0000:08:00.0"

    - logical-name: hed3
      type: simple-port
      bus-address: "0000:09:00.0"

    - logical-name: hed4
      type: simple-port
      bus-address: "0000:0a:00.0"
```

4. Modify OSD nodes to use four port servers, as shown here. Change the NIC mapping attribute from **HP-DL360-4PORT** to use any other name defined in `nic_mapping.yml`.

```
# Ceph OSD Nodes
- id: osd1
  ip-addr: 192.168.10.9
  role: OSD-ROLE
  server-group: RACK1
  nic-mapping: HP-DL360-4PORT
  mac-addr: "8b:f6:9e:ca:3b:78"
  ilo-ip: 192.168.9.9
  ilo-password: password
  ilo-user: admin
```

5. Delete OSD-CLIENT from `network_groups.yml`. Note that no dedicated network group exists for client traffic. Only the management network group is used for client traffic.
6. Edit `net_interfaces.yml` with a bonded NIC as shown here.

```
- name: CONTROLLER-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed1
        provider: linux
        devices:
          - name: hed1
          - name: hed2
  network-groups:
    - EXTERNAL-API
    - EXTERNAL-VM
    - GUEST
    - MANAGEMENT

- name: COMPUTE-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
```

```
bond-data:
  options:
    mode: active-backup
    miimon: 200
    primary: hed1
  provider: linux
  devices:
    - name: hed1
    - name: hed2
network-groups:
  - EXTERNAL-VM
  - GUEST
  - MANAGEMENT

- name: OSD-INTERFACES
network-interfaces:
  - name: BOND0
    device:
      name: bond0
  bond-data:
    options:
      mode: active-backup
      miimon: 200
      primary: hed1
    provider: linux
    devices:
      - name: hed1
      - name: hed2
  network-groups:
    - MANAGEMENT
- name: BOND1
  device:
    name: bond1
  bond-data:
    options:
      mode: active-backup
      miimon: 200
      primary: hed3
    provider: linux
    devices:
      - name: hed3
      - name: hed4
  network-groups:
    - OSD-INTERNAL

- name: RGW-INTERFACES
network-interfaces:
  - name: BOND0
    device:
      name: bond0
  bond-data:
    options:
      mode: active-backup
      miimon: 200
```

```
        primary: hed1
    provider: linux
    devices:
        - name: hed1
        - name: hed2
    network-groups:
        - MANAGEMENT
```

7. Delete OSD-CLIENT from `server_groups.yml`.
8. Delete VLAN information for OSD-CLIENT-NET from `networks.yml`. Only management VLANs are used for client traffic.
9. After you set up your configuration files, perform in .

## RADOS Gateway

### **Installing RADOS Gateway on Dedicated Cluster Nodes that Host the Ceph Monitor Service**

You can configure RADOS Gateway to install on one or more dedicated cluster nodes hosting the Ceph monitor service as follows:

1. Remove the sections for servers in the `~/helion/my_cloud/definition/data/server-s.yml` file that have the `role: RGW-ROLE` attribute.
2. Edit the `~/helion/my_cloud/definition/data/control_plane.yml` file and add the following lines to the `service-components` section for the cluster nodes that have the `server-role: MON-ROLE` attribute.
  - `ceph-radosgw`
  - `apache2`
3. Edit the `~/helion/my_cloud/definition/data/net_interfaces.yml` file to remove the `RGW-INTERFACES` section. This section defines RADOS Gateway network interfaces, which are not required in this configuration:

```
- name: RGW-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed3
      provider: linux
      devices:
        - name: hed3
        - name: hed4
    network-groups:
      - MANAGEMENT
      - OSD-CLIENT
```
4. After you set up your configuration files, perform in .

## Installing RADOS Gateway on Controller Nodes

You can configure RADOS Gateway to install on controller nodes. To do this, perform the following steps:

1. Remove the sections for servers in the `~/helion/my_cloud/definition/data/servers.yml` file that have the `role: RGW-ROLE` attribute.
2. Edit the `~/helion/my_cloud/definition/data/net_interfaces.yml` file to remove the `RGW-INTERFACES` section. This section defines RADOS Gateway network interfaces, which are not required in this configuration:

```
- name: RGW-INTERFACES
  network-interfaces:
    - name: BOND0
      device:
        name: bond0
      bond-data:
        options:
          mode: active-backup
          miimon: 200
          primary: hed3
        provider: linux
      devices:
        - name: hed3
        - name: hed4
  network-groups:
    - MANAGEMENT
    - OSD-CLIENT
```

3. Edit the `~/helion/my_cloud/definition/data/control_plane.yml` file and add the following line to `service-components` for the cluster with the `server-role: CONTROLLER-ROLE` attribute.

```
- ceph-radosgw
```

4. After you set up your configuration files, perform in .

## Installing More than Two RADOS Gateway Servers

To deploy more than two RADOS Gateway servers, you need to add a section to the `~/helion/my_cloud/definition/data/servers.yml` file for each additional RADOS Gateway node.

### Note

Installing additional RADOS Gateway servers is possible only if RADOS Gateway is installed on dedicated cluster nodes or on dedicated cluster nodes that host the Ceph monitor service. Additional RADOS Gateway servers cannot be added if RADOS Gateway is installed on a controller node.

## Ceph Deployment with Virtual Control Plane

HPE Helion OpenStack® 5.0 supports the deployment of control plane elements on virtual machines which can be co-located on one baremetal machine or spread across three baremetal machines. The baremetal machine in this context is termed as VM factory host(s). The following aspects must be considered while deploying Ceph with the virtual control plane.

1. Deploy OSD and monitor node on a single VM factor host - It is applicable to X1 cloud model. The number of VM factory host is one. Therefore, Ceph cluster will not have HA support as there will be only one instance of the monitor component of Ceph.
2. Deploy OSD and monitor on set of three VM factor hosts - It is applicable to S1 cloud model.
3. Deploy monitor on set of three VM factor hosts but OSD nodes are deployed independently as a resource nodes - It is applicable to M1 cloud model.

The following aspects must be considered while deploying Ceph with virtual control plane:

1. Scale-out of cluster - Adding a new set of monitor or OSD nodes is not validated by the engineering team. Although, technically it is feasible but not recommended because it can have a significant performance impact. However, one can increase a cluster capacity by adding more disks to the VM factory host and configuring them as OSD (a scale-in path to increase capacity) nodes.
2. Deployment of RADOS Gateway on VM factory host is not formally supported.
3. Performance of Ceph components (OSD and monitor nodes) are sensitive to compute resources i.e. memory, core CPU, and so on. It is strongly recommended to plan and allocate minimum amount of resources for Ceph components to avoid resource contention because same set of machines will be running the control plane elements and the Ceph components. Starvation of resources might causes impact on the performance and the stability of the Ceph clusters health. Consider the following resource aspect for planning:
  - CPU
  - RAM
  - Disk space for monitoring logs

The following section focus on the change of the input model for X1 and S1 model ONLY. The change in the input model for M1 model is similar except that OSD node is deployed as the resource nodes. For other aspects of cluster management like upgrade, adding new set of disks, stopping and starting services and so on, you can follow the similar approach that is used for the deployment of Ceph cluster using HPE Helion OpenStack® .

#### Steps to deploy Ceph on VM factory host(s)

1. Log in to the lifecycle manager.
2. Go to `~/helion/my_cloud/definition/`.

#### Note

We assume that you have already copied the cloud model representing control element deployment on VM factory host(s).

3. Edit your `control_plane.yml` file to add `ceph-osd` and `ceph-monitor` components to the `vmfactory` nodes. For example:

```
- name: vmfactory
  resource-prefix: vmf
  server-role: HLM-HYPERVISOR-ROLE
  min-count:
  allocation-policy: strict
```

```
service-components:
  - ntp-server
  - ceph-osd
  - ceph-monitor
  - lifecycle-manager
  - tempest
  - openstack-client
```

4. Edit `disks_vmfactory.yml` file of VM factory hosts to define data and journal disks for OSD. For example, the following disk model illustrates the usage of `/dev/sdd`, `/dev/sde`, and `/dev/sdf` as data disks and `/dev/sdg` as journal disks for OSD. Disks allocated to OSD must not be used for any other purpose.

```
---
product:
  version: 2

disk-models:
  - name: HLM-HYPERVISOR-DISKS
volume-groups:
  - name: hlm-vg
physical-volumes:
  - /dev/sda_root
logical-volumes:
  # The policy is not to consume 100% of the space of each volume group.
  # 5% should be left free for snapshots and to allow for some flexibility
  - name: root
    size: 35%
    fstype: ext4
    mount: /
  - name: log
    size: 50%
    mount: /var/log
    fstype: ext4
    mkfs-opt: -O large_file
  - name: crash
    size: 10%
    mount: /var/crash
    fstype: ext4
    mkfs-opt: -O large_file
  - name: vg-images
    # this VG is dedicated to libvirt images to keep VM IOPS off the OS disk
physical-volumes:
  - /dev/sdb
  - /dev/sdc
logical-volumes:
  - name: images
    size: 95%
    mount: /var/lib/libvirt/images
    fstype: ext4
    mkfs-opt: -O large_file

device-groups:
  - name: ceph-osd-disks
```

```
devices:
- name: /dev/sdd
- name: /dev/sde
- name: /dev/sdf
consumer:
  name: ceph
  attrs:
    usage: data
    journal_disk: /dev/sdg
```

5. Commit your configuration to the local git repo.

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

6. After setting up the configuration files, continue with the installation procedure mentioned at .

### Note

- a. Running `ceph-stop.yml` and `ceph-start.yml` playbooks on the VM factory host stops all the services (OSD and monitor) on the node. There is no playbook that can stop only one service.
- b. If you are deploying Ceph on a single VM factory host (i.e. X1 model), make the following changes before deployment.
  - Edit `~/helion/my_cloud/config/ceph/settings.yml` file to add the following content.

```
extra:
global:
  osd_crush_chooseleaf_type: 0
```
  - Commit your configuration to the local git repo and continue with the installation procedure mentioned at .

## Usage of Ceph Storage

Ceph is a versatile storage technology that facilitates the consumption of storage by using multiple protocols. Ceph is closely integrated with HPE Helion OpenStack® services to support various storage use cases such as the use of storage pools for cinder-volume and glance data store. HPE Helion OpenStack® further simplifies administrator tasks by providing automated playbooks for various storage use cases.

- Creating storage pools
- Deploying Ceph client components on client nodes
- Configuring OpenStack services with storage pools

The following section guides you through the following common scenarios that you might come across while consuming Ceph for various storage use cases.

- 1.
2.
  - a.

- b.
  - c.
  - d.
- 3.

Although you can also use Ceph storage pools as ephemeral storage backends for Nova, this practice is not formally supported.

## Set Up Your Deployer as a Ceph Client Node

By default, your deployer node is not configured as a Ceph client node. As a result, you cannot start your Ceph operation here. It is usually more helpful to set up your deployer node as an admin client node to perform various Ceph operations. This will help you manage the Ceph cluster without logging in to the monitor node. You can turn your deployer node into a client node by executing the following playbook.

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts ceph-setup-deployer-as-client.yml
```

After you complete the preceding playbook, your deployer node will be configured with an admin key ring and thus act as an admin node.

## Using Core Ceph OpenStack Services

The use of Cinder for glance, cinder-volume, and cinder-backup requires pool creation and setting up ceph-clients on respective OpenStack service nodes. To simplify the workflow, the playbook `ceph-client-prepare.yml` is provided. This playbook reads pool configurations following the specification provided in `ceph_user_model.yml` and sets up an OpenStack client node. Perform the following steps:

1. Define the user model.
2. Run `hosts/verb_hosts ceph-client-prepare.yml`.

The default user model provided with HPE Helion OpenStack® is mentioned below. The default configuration contains pool configuration for cinder-volume, glance, and cinder-backup. You can edit the file if you do not intend to use Ceph for all services or want to change pool attributes based on your cluster configuration. For example, if you have large number of disks then you can increase the Placement Group (PG) number for a given pool.

---

```
product:  
  version: 2  
  
ceph_user_models:  
  - user:  
      name: cinder  
      type: openstack  
      secret_id: 457eb676-33da-42ec-9a8c-9293d545c337  
    pools:  
      - name: volumes  
        attrs:  
          creation_policy: eager  
          type: replicated
```

```

replica_size: 3
permission: rwx
pg: 100
usage:
    consumer: cinder-volume
- name: vms
  attrs:
    creation_policy: eager
    type: replicated
    replica_size: 3
    permission: rwx
    pg: 100
  usage:
    consumer: nova
- name: images
  attrs:
    creation_policy: lazy
    permission: rx
- user:
  name: glance
  type: openstack
  pools:
    - name: images
      attrs:
        creation_policy: eager
        type: replicated
        replica_size: 3
        permission: rwx
        pg: 128
      usage:
        consumer: glance-datastore
- user:
  name: cinder-backup
  type: openstack
  pools:
    - name: backups
      attrs:
        creation_policy: eager
        type: replicated
        replica_size: 3
        permission: rwx
        pg: 128
      usage:
        consumer: cinder-backup

```

The following table provides the descriptions of the preceding parameters.

| Parameter | Value                  | Description                                                                                                      | Recommendation                                                                                                  |
|-----------|------------------------|------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------|
| user.name | A user-defined string. | Defines the name of the user who can access a set of pools. A user is created with same name in the Ceph system. | Retain the default names for some of the well-known OpenStack users as described in the default user model. For |

| Parameter                   | Value                  | Description                                                                                                                                                                                  | Recommendation                                                                                                                                                                                                                                                                                                 |
|-----------------------------|------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                             |                        |                                                                                                                                                                                              | example, cinder user for volume access who needs to access volumes, vms, and the images pool, as defined in the default model.                                                                                                                                                                                 |
| user.type                   | OpenStack   user.      | Indicates whether the user is specific for OpenStack services. This parameter does not have semantic implications.                                                                           | Use <b>openstack</b> for pools used by cinder-volume, cinder-backup, glance, and Nova services. For other services, you can choose <b>user</b> .                                                                                                                                                               |
| user.secret_id              | UUID instance.         | A unique UUID.                                                                                                                                                                               | Generate a value for your cluster before setting up the Cinder client. The libvirt process needs this value to access the cluster while attaching a block device from Cinder.                                                                                                                                  |
| pools.name                  | A user-defined string. | A user-defined name.                                                                                                                                                                         | The name has to be unique in the Ceph namespace.                                                                                                                                                                                                                                                               |
| pools.attrs.creation_policy | eager   lazy           | If creation_policy is "eager," the playbooks will create the user. If the creation_policy is set to "lazy," the pool will be created externally (not by Ceph ansible playbooks) out of band. | Retain default values for pools meant to be consumed by OpenStack services.                                                                                                                                                                                                                                    |
| pools.attrs.type            | Replicated.            | Defines the type of pool. Ceph supports erasure-coded and replicated pools.                                                                                                                  | Retain the value as replicated if you do want to use <code>ceph-client-prepare.yml</code> for pool creation.<br><br>This does not mean that you cannot create an erasure-code pool with your cluster deployed using HPE Helion OpenStack®. It just means that erasure-code pools are not officially supported. |
| pools.attrs.replica_size    | 1..n                   | Replica count.                                                                                                                                                                               | Use 3 as the replica count because it is used in most common scenarios providing the right                                                                                                                                                                                                                     |

| Parameter              | Value                   | Description                                                                                                                                                                                                                                           | Recommendation                                                                                                                                                  |
|------------------------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                        |                         |                                                                                                                                                                                                                                                       | level of protection and is the minimum setting that HPE Helion OpenStack supports.                                                                              |
| pools.attrs.permission | rwx                     | Read, write, and execute the permission a user will have on given pool.                                                                                                                                                                               | Retain the values for OpenStack user as mentioned in the default user model. Altering these values might affect your client setup configuration.                |
| pools.attrs.pg         | Placement group number. | Number of placement groups.                                                                                                                                                                                                                           | The default value is 128. You can change the value of this parameter if your disk size is larger.                                                               |
| pools.usage.consumer   | Predefined choices.     | Defines pool consumers, and indicates which services are expected to consume a given pool with which access. It has semantic meaning for the following predefined choices:<br>a. nova<br>b. glance-datastore<br>c. cinder-volume<br>d. cinder-back-up | Do not pick the value from glance-datastore, nova, cinder-volume, or cinder-backup for user-defined pools. For user-defined pools, you can skip this parameter. |

## Prerequisites

To use Ceph as a volume backend, the nodes running these services should have the Ceph client installed on them. Use the `ceph-client-prepare.yml` playbook to deploy the Ceph client on these nodes.

```
cd ~/scratch/ansible/next/hosts/ansible
ansible-playbook -i hosts/verb_hosts ceph-client-prepare.yml
```

This playbook also creates Ceph users and Ceph pools on the resource nodes.

### Note

The following steps install packages and configure the existing client nodes (such as the Cinder, Glance, and Nova Compute nodes) required to use the Ceph cluster. For any new client nodes added later on that need to be configured to use the Ceph cluster, just execute the preceding playbook with the addition of the `--limit <new-client-node>` switch.

## Use Ceph Storage Pools as Glance Data Stores

To enable Ceph as a Glance data store, perform the following steps:

1. Log in to the lifecycle manager.
2. Make the following changes in the `~/helion/my_cloud/config/glance/glance-api.conf.j2` file:
  - a. Copy the following content and paste it into the `glance-api.conf.j2` file under **[glance\_store]**:

```
[glance_store]
default_store = rbd
stores = rbd
rbd_store_pool = images
rbd_store_user = glance
rbd_store_ceph_conf = /etc/ceph/ceph.conf
rbd_store_chunk_size = 8
```

- b. In the same file, comment out the following references to Swift:

```
stores = {{ glance_stores }}
default_store = {{ glance_default_store }}
```

- 3. IMPORTANT:** If you have preexisting images in your Glance repo and want to use Ceph exclusively as a backend, use the Glance CLI or other tool to back up your images before configuring Ceph as your Glance backend:

- a. Snapshot or delete all Nova instances using those images.
  - b. Download the images locally that you want to save.
  - c. Delete all the images from Glance.

After you finish the Ceph configuration you will need to add those images again.

- 4. Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*), as follows:**

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

- 5. Run the configuration processor:**

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

- 6. Update your deployment directory:**

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

- 7. Run the Glance reconfigure playbook to configure Ceph as a Glance backend:**

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts glance-reconfigure.yml
```

After you execute the preceding command successfully, the glance is configured to use Ceph as a data store. You can use glance operations to upload images, which will be stored in the Ceph cluster.

## Use Ceph Storage Pools as Cinder Volume Backends

To enable Ceph as a Cinder volume backend, follow these steps:

1. Log in to the lifecycle manager.
2. Make the following changes to the `~/helion/my_cloud/config/cinder/cinder.conf.j2` file:
  - a. Add your Ceph backend to the `enabled_backends` section:

```
# Configure the enabled backends
enabled_backends=ceph1
```

### Important

If you are using multiple backend types, you can use a comma-delimited list here. For example, if you are going to use both VSA and Ceph backends, specify `enabled_backends=vsa-1,ceph1`.

- b. [optional] If you want your volumes to use a default volume type, enter the name of the volume type in the `[DEFAULT]` section with the following syntax. **Make note of this value because you will need it when you create your volume type later.**

### Important

If you do not specify a default type, then your volumes will default to a nonredundant RAID configuration. We recommend that you create a volume type that meets your environments needs and specify it here.

```
[DEFAULT]
# Set the default volume type
default_volume_type = <your new volume type>
```

- c. Uncomment the `ceph` section and supply the values to match your cluster information. If you have more than one cluster, you will need to add another similar section with respective values. The following example adds only one cluster.

```
[ceph1]
rbd_secret_uuid = <secret-uuid>
rbd_user = <ceph-cinder-user>
rbd_pool = <ceph-cinder-volume-pool>
rbd_ceph_conf = <ceph-config-file>
rbd_cluster_name = <cluster_name>
volume_driver = cinder.volume.drivers.rbd.RBDDriver
volume_backend_name = <ceph-backend-name>
```

This example has the following values:

| Value           | Description                                                                                                                               |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| rbd_secret_uuid | The <code>secret_id</code> value from the <code>~/helion/my_cloud/config/ceph/user_model.yml</code> file, as highlighted here:<br>- user: |

| Value               | Description                                                                                                                                                                                                                     |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                     | <pre> name: cinder type: openstack secret_id: &lt;secret ID will be here&gt; pools: - name: volumes </pre> <p><b>Important</b></p> <p>You should generate and use your own secret ID. You can use any UUID generation tool.</p> |
| rbd_user            | The username value from the ~/helion/my_cloud/config/ceph/user_model.yml file, as highlighted here: <pre> - user:   name: cinder   type: openstack   secret_id: &lt;secret ID will be here&gt; pools: - name: volumes </pre>    |
| rbd_pool            | The pool name value from the ~/helion/my_cloud/config/ceph/user_model.yml file, as highlighted here: <pre> - user:   name: cinder   type: openstack   secret_id: 457eb676-33da-42ec-9a8c-9293d5 pools: - name: volumes </pre>   |
| rbd_ceph_conf       | The Ceph configuration file location, usually /etc/ceph/ceph.conf.                                                                                                                                                              |
| rbd_cluster_name    | Name of the Ceph cluster.                                                                                                                                                                                                       |
| volume_driver       | The Cinder volume driver. Leave this as the default value specified for Ceph.                                                                                                                                                   |
| volume_backend_name | The name given to the Ceph backend.                                                                                                                                                                                             |

3. To enable attaching Ceph volumes to Nova provisioned instances, make the following changes to the ~/helion/my\_cloud/config/nova/kvm-hypervisor.conf.j2 file:

- a. Uncomment the Ceph backend lines and edit them as follows:

```
[libvirt]
rbd_user = <ceph-user>
rbd_secret_uuid = <secret-uuid>
```

The values are as follows:

| Value           | Description                                                                                                                                                                                                                                                           |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| rbd_user        | <p>The username value from the <code>~/helion/my_cloud/config/ceph/user_model.yml</code> file, as highlighted here:</p> <pre data-bbox="915 375 1632 498">- user:   name: <b>cinder</b>   type: openstack   secret_id: 457eb676-33da-42ec-9a8c-9293d545</pre>         |
| rbd_secret_uuid | <p>The secret_id value from the <code>~/helion/my_cloud/config/ceph/user_model.yml</code> file, as highlighted here:</p> <pre data-bbox="915 618 1632 760">- user:   name: <b>cinder</b>   type: openstack   secret_id: <b>457eb676-33da-42ec-9a8c-9293d545</b></pre> |

## Note

To attach a volume provisioned out of a newly added Ceph backend to an existing OpenStack instance, the instance must be rebooted after the new backend has been added.

## Important

Do not use `backend_host` variable in `cinder.conf` file. If `backend_host` is set, it will override the [DEFAULT]/host value which HPE Helion OpenStack 5.0 is dependent on.

4. Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*), as follows:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

5. Run the configuration processor:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Update your deployment directory:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Cinder reconfigure playbook:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
```

8. If Nova has been configured to attach Ceph backend volumes, run the Nova reconfigure playbook:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

After you execute the preceding command successfully, the cinder-volume is configured to use Ceph as a data store. You can use volume lifecycle operations as described at *FIXME: broken external xref*

## Use Ceph Storage Pools as Cinder Backup Devices

To enable Cinder backup devices for Ceph, make the following changes to the `~/helion/my_cloud/config/cinder/cinder.conf.j2` file:

1. Uncomment the `cinder backup` section and supply the values:

```
[DEFAULT]
backup_driver = cinder.backup.drivers.ceph
backup_ceph_conf = <ceph-config-file>
backup_ceph_user = <ceph-backup-user>
backup_ceph_pool = <ceph-backup-pool>
```

The values are as follows:

| Value            | Description                                                                                                                                                                                                   |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| backup_driver    | The Cinder volume driver. Leave this as the default value specified for Ceph.                                                                                                                                 |
| backup_ceph_conf | The Ceph configuration file location, usually: <code>/etc/ceph/ceph.conf</code> .                                                                                                                             |
| backup_ceph_user | The username value from the <code>~/helion/my_cloud/config/ceph/user_model.yml</code> file, as highlighted here:<br><br>- user:<br>name: <b>cinder-backup</b><br>type: openstack<br>pools:<br>- name: backups |
| backup_ceph_pool | The pool name value from the <code>~/helion/my_cloud/config/ceph/user_model.yml</code> file, as highlighted here:<br><br>pools:<br>- name: <b>backups</b>                                                     |

2. Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*), as follows:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

3. Run the configuration processor:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

4. Update your deployment directory:

```
cd ~/helion/hos/ansible
```

```
ansible-playbook -i hosts/localhost ready-deployment.yml
```

5. Run the Cinder reconfigure playbook:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
```

After you execute the preceding command successfully, the cinder-backup service is configured to use Ceph as a data store. You can use volume lifecycle operations as described in .

## Use RADOS Gateway to Access Objects Using S3/Swift API

RADOS Gateway provides object storage functionality through S3 and Swift API. It has a built-in authentication mechanism and it can use Keystone as an authentication backend, unlike the OpenStack Services, where Keystone is the only authentication backend. Users primarily see the following combinations of object accessibility:

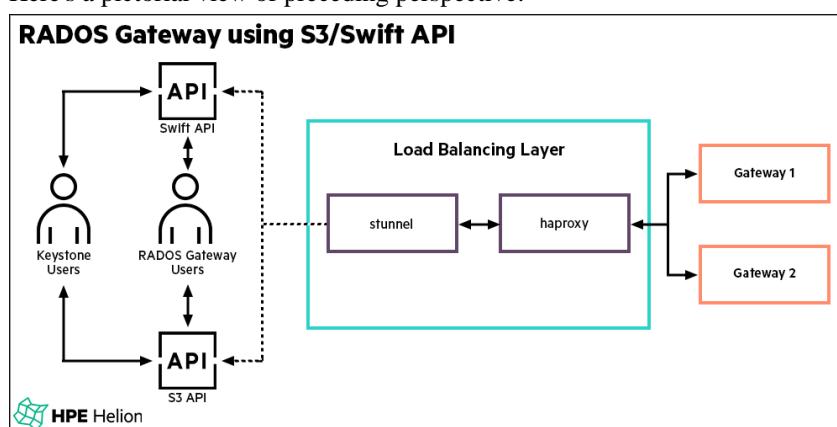
1. Keystone users using:

- a. Swift API
- b. S3 API

2. RADOS Gateway users using:

- a. Swift API
- b. S3 API

Here's a pictorial view of preceding perspective:



The first view uses Keystone as the authentication backend while the second one uses RADOS Gateway (internal authentication backend). Default deployment of RADOS Gateway in HPE Helion OpenStack® 5.0 enables 1.a, 2.a, and 2.b only.

Enabling choice 1.b for Keystone users accessing S3 API requires Keystone to be the default authentication backend. In this case all user authentication goes first to Keystone, with failover to RADOS Gateway even for non-Keystone users (such as in case 2.b), but this situation causes a serious performance drop for RADOS Gateway users.

Lab tests performed on a 10 TB Ceph cluster have shown that enabling S3 API access for Keystone users degrades the performance of S3 API access for RADOS Gateway users roughly by 50%. We ran the rest-bench tool for 10 seconds with a concurrency of 16 and an object size of 4096 (4K) bytes on Ceph deployed using the entry-scale-kvm-ceph input model. Our lab finding was that the system performed an

average of 1350 (object size 4K) write operations when Keystone was enabled as the default authentication backend, in contrast to 2500 (object size 4K) operations in the default configuration, which was a degradation of roughly 49%. Note that these numbers are for illustration purposes to provide an insight on the degree of degradation. However, storing all credentials in Keystone provides the advantage of reducing the maintenance burden. It is not required to create or manage credentials in a database for S3 authentication. You can use standard administrative tools such as Horizon instead. Aspect 1.b does not apply if you do not intend to enable S3 API access for Keystone users.

## Steps to Access S3 API for RADOS Gateway Users

Perform these steps:

1. Create a user by executing the following command:

```
sudo radosgw-admin user create --uid="{username}" --display-name="{Display Name}
```

Example:

```
sudo radosgw-admin user create --uid=rgwuser --display-name="RadosGatewayUser" -
```

2. Perform the object operation. You can use S3 clients. Example: python-boto.

## Steps to Access Swift API for RADOS Gateway Users

Perform these steps:

1. Create a user by executing the following command:

```
sudo radosgw-admin user create --uid="{username}" --display-name="{Display Name}
```

Example:

```
sudo radosgw-admin user create --uid=rgwuser --display-name="RadosGatewayUser" -
```

2. Create a subuser by executing the following command:

```
sudo radosgw-admin subuser create --uid={username} --subuser={username}:{subuser}
```

Example:

```
sudo radosgw-admin subuser create --uid="rgwuser" --subuser="rgwuser:rgwswiftuse
```

3. Perform the object operation using Swift CLI or any compatible client.

## Steps to Access Swift API for Keystone Users

When RADOS Gateway is deployed per the default entry-scale-kvm-ceph model, it exists alongside OpenStack Swift (this is because the Keystone service type for RADOS Gateway defaults to ceph-object-store). In this (coexisting) mode, OpenStack Horizon communicates only with OpenStack Swift for all the object store operations. However, the OpenStack Swift command line interface can be tuned to communicate with both OpenStack Swift and RADOS Gateway as follows:

- To interact with Swift (default):

```
export OS_SERVICE_TYPE=object-store
```

- To interact with RADOS Gateway:

```
export OS_SERVICE_TYPE=ceph-object-store
```

You can use Swift CLI to perform object operations on Ceph.

**Example:**

Log in to the monitor node to list the objects in Ceph using the credentials available in the `~/service.osrc` file.

```
source ~/service.osrc
```

Execute the following command to list objects in the Ceph object store:

```
export OS_SERVICE_TYPE=ceph-object-store
swift list
```

Execute the following command to list objects in the Swift object store:

```
export OS_SERVICE_TYPE=object-store
swift list
```

## Steps to Access S3 API for Keystone Users

By default, a Keystone user can access the RADOS Gateway functionality using the Swift API. To configure a Keystone user for S3 API to access the RADOS Gateway, perform the following steps:

1. Login to lifecycle manager.
2. Edit the `~/helion/my_cloud/config/ceph/settings.yml` file and set the `rgw_s3_auth_use_keystone` value to **true**.  
`rgw_s3_auth_use_keystone: true`
3. Commit your configuration to local repo.

```
cd ~/helion/hos/ansible
git add -A
git commit -m <"commit message">
```

4. Run ready-deployment playbook.

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

5. Run the ceph-reconfigure playbook.

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts ceph-reconfigure.yml
```

### Create the EC2 credentials

- a. Login to a controller node and source the admin user credentials.

```
source ~/service.osrc
```

- b. Execute the following commands to generate the EC2 credentials for the OpenStack user.

```
openstack ec2 credentials create --project <project-name> --user <user-name>
```

For example, to generate the EC2 credentials for user demo for the project demo.

```
openstack ec2 credentials create --project demo --user demo
```

## Important

Please contact the Professional Services team for details on how to perform the preceding steps.

# Managing Ceph Clusters After Deployment

After you successfully deploy your Ceph cluster and it is functioning as expected, you can perform various maintenance operations for the Ceph cluster when required. For example, you can change the configuration of Ceph service components, or scale up storage nodes. The following list provides details about various maintenance options that you might encounter for cluster management. Text links take you to the instructions for each step.

1. Modify service configuration.

2. Scale out cluster.

a. .

b. .

c. See .

3. Scale down cluster.

a. .

b. .

c. .

4. Repair scenarios.

a. .

b. .

c. .

d. .

e. .

5. .

Note that the preceding maintenance procedure is based on the `entry-scale-kvm-ceph` input model. If you have altered your deployment, such as by deploying the monitor service on standalone nodes, you might have to adjust the procedure accordingly.

# Configuring for VSA Block Storage Backend

## Abstract

This page describes how to configure your VSA backend for the Helion Entry-scale with KVM cloud model. This procedure can be performed during or after installation.

## Prerequisites

- Your HPE Helion OpenStack cloud must be fully deployed using the KVM cloud model with VSA added. The Entry-Scale KVM with VSA cloud model is an example of such a model. For more details on the installation refer to Chapter 10, *Installing Mid-scale and Entry-scale KVM*.
- It's important that all of your systems have the correct date/time because the HPE StoreVirtual VSA license has a start date. If the start date is later than the system time then the installation will fail.

## Notes

- The license for the HPE StoreVirtual VSA license is bundled with HPE Helion OpenStack and comes with a free trial which allows a maximum limit of 50 TB per node. Hence the total amount of the configured storage on an individual VSA node should not exceed 50 TB. To extend the 50 TB per node limit, you can add nodes. A VSA cluster can support up to 16 nodes, which means configured storage on a VSA cluster can be as much as 800 TB.

You can deploy VSA with Adaptive Optimization (AO) or without. The deployment process for each of these options is similar, you just need to make a change in the disk input model. For more detailed information, refer to **VSA with or without Adaptive Optimization (AO)** in Chapter 11, *Modifying the Entry-scale KVM with VSA Model for Your Environment*.

- A single VSA node can have a maximum of seven raw disks (excluding the operating system disks) attached to it, which is defined in the disk input model for your VSA nodes. It is expected that no more than seven disks are specified (including Adaptive Optimization disks) per VSA node. For example, if you want to deploy VSA with two disks for Adaptive Optimization then your disk input model should not specify more than five raw disks for data and two raw disks for Adaptive Optimization. Exceeding the disk limit causes VSA deployment failure.
- You can create more than one VSA cluster of same or different type by specifying the configuration in cloud model. For more details, refer to .
- Usernames and passwords designated during VSA configuration or repair must be alphabetic only.

**Concerning using multiple backends:** If you are using multiple backend options, ensure that you specify each of the backends you are using when configuring your `cinder.conf.j2` file using a comma delimited list. An example would be `enabled_backends=vsa-1,3par_iSCSI,ceph1` and is included in the steps below. You will also want to create multiple volume types so you can specify which backend you want to utilize when creating volumes. These instructions are included below as well. In addition to our documentation, you can also read the OpenStack documentation at Configure multiple storage backends [[http://docs.openstack.org/admin-guide/blockstorage\\_multi\\_backend.html](http://docs.openstack.org/admin-guide/blockstorage_multi_backend.html)] as well.

**VSA driver has updated name:** In the OpenStack Mitaka release, the VSA driver used for HPE Helion integration had its name updated from `hplefthand_` to `hpelefthand_`. To prevent issues when upgrading from previous HPE Helion OpenStack releases, we left the names as-is in the release and provided a mapping so that the integration would continue to work. This will produce a warning in the `cinder-volume.log` file advising you of the deprecated name. The warning will look similar to this:

```
Option "hplefthand_username" from group "<your section>" is deprecated. Use option  
should be ignored.
```

These are just warnings and can be ignored.

## Configure HPE StoreVirtual VSA

The process for configuring HPE StoreVirtual VSA for HPE Helion OpenStack involves two major steps:

- Creating a cluster
- Configuring VSA as the block storage backend

You can perform these tasks in an automated process using the provided Ansible playbooks or manually using the HPE StoreVirtual Centralized Management Console (CMC) GUI.

### Note

Using Ansible playbooks to create clusters is strongly recommended. Cluster creation using HPE StoreVirtual Centralized Management Console (CMC) GUI is deprecated in HPE Helion OpenStack 5.0. Clusters created using CMC manually cannot be reconfigured using the Ansible playbooks. If you used CMC to create the clusters, you must continue to use CMC to administer those clusters.

## Using Ansible

Perform the following steps to create the cluster using the provided Ansible playbooks.

1. Log in to the lifecycle manager.
2. Run the following playbook, which will both create your management group and your cluster:

```
cd ~/scratch/ansible/next/hos/ansible/
ansible-playbook -i hosts/verb_hosts vsalm-configure-cluster.yml
```

3. When prompted, enter an administrative user name and password you will use to administer the CMC utility.

Administrative user naming requirements:

- 3 to 30 characters
- Must begin with a letter
- Allowed characters: 0-9, a-z, A-Z, hyphen (-), underline (\_)
- Disallowed characters: Multibyte character set

Administrative user password requirements:

- 5 to 40 characters
- Must begin with a letter
- Many ASCII and extended ASCII characters, Multibyte character set
- Disallowed characters: dot (.), colon (:), forward slash (/), comma (,), backward slash (\), semi-colon (;), single-quote ('), space ( ), equals (=)

### Important

You will need to remember these values to access the cluster using the CMC utility or to re-run this playbook later if further management of your VSA system is needed. Do not change the credentials for one or more clusters using CMC as it will make the cluster automation playbook ignore those clusters for reconfiguration.

4. In the output of the Ansible playbook you will see different steps where you can locate the values for your VSA cluster name, the virtual IP address, and the IP addresses of each of the VSA hosts in the cluster. You will use these values later when configuring your backend.

The following is an example, with the important information bolded.

This step displays the VSA cluster name and the virtual IP address of the cluster:

```
TASK: [vsalm | _create_cluster | Display the status of cluster creation performed]
ok: [helion-cpl-vsa0001-mgmt] => {
    "msg": "Created cluster - Name=cluster-vsa, IP=10.13.111.142"
}
```

5. You can view the added node to the cluster in CMC GUI as follows:

- a. Launch the CMC utility. For more information, see the section called “Using the CMC Utility”.
- b. Find the new VSA system.
- c. View the added node in the cluster.

## Using the CMC Utility

In HPE Helion OpenStack, cluster creation using HPE StoreVirtual Centralized Management Console (CMC) GUI is deprecated. The clusters created by CMC manually cannot be configured using Ansible playbooks.

Perform the following steps to create the cluster using CMC.

### Launching the CMC utility GUI

The HPE StoreVirtual Centralized Management Console (CMC) GUI is an interface where you can configure VSA.

The CMC utility requires a GUI to access it. You can use either of the following methods to launch the CMC GUI.

- the section called “RDP/VNC Connection Method”
- the section called “Install (any) X Display Tool”

## RDP/VNC Connection Method

Set up VNC connect on the Controller Node

The following steps will allow you to setup a VNC connect to your controller node so you can view the GUI.

1. Log in to your first controller node.
2. Run the following command to install the package that is required to launch CMC:

```
sudo apt-get install -y xrdp
```

3. Start vnc4server using the instructions below. You will be prompted for a password (minimum 6 characters). Enter a password and proceed. A sample output is shown below:

```
stack@helion-cpl-c1-m1-mgmt:~$ vnc4server

You will require a password to access your desktops.

Password:
Verify:
xauth:  file /home/stack/.Xauthority does not exist

New 'helion-cpl-c1-m1-mgmt:3 (stack)' desktop is helion-cpl-c1-m1-mgmt:3

Creating default startup script /home/stack/.vnc/xstartup
Starting applications specified in /home/stack/.vnc/xstartup
Log file is /home/stack/.vnc/helion-cpl-c1-m1-mgmt:3.log
```

## Note

If you use **xrdp** directly to connect to the first controller node without using the VNC server, then a remote session is created whenever you login. To avoid this, a dedicated VNC server instance is launched and connected to that instance by **xrdp**. This helps to maintain the session.

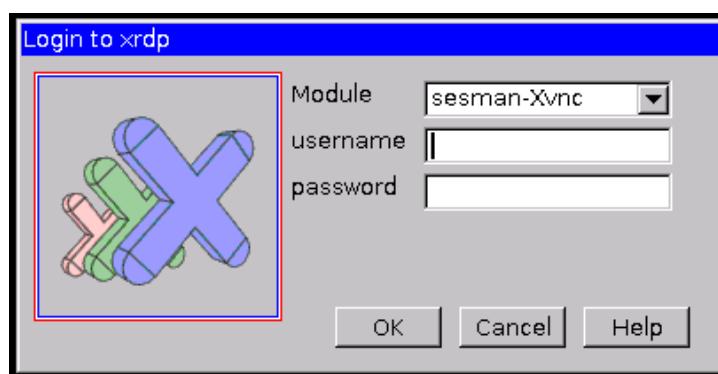
- Run **netstat -anp | grep vnc** to determine the public port that VNC is using. In the example below, the port is 5903:

```
stack@helion-cpl-c1-m1-mgmt:~$ netstat -anp | grep vnc
(Not all processes could be identified, non-owned process info
will not be shown, you would have to be root to see it all.)
tcp        0      0 0.0.0.0:6003          0.0.0.0:*                  LISTEN
tcp6       0      0 :::::5903           ::::*                  LISTEN
```

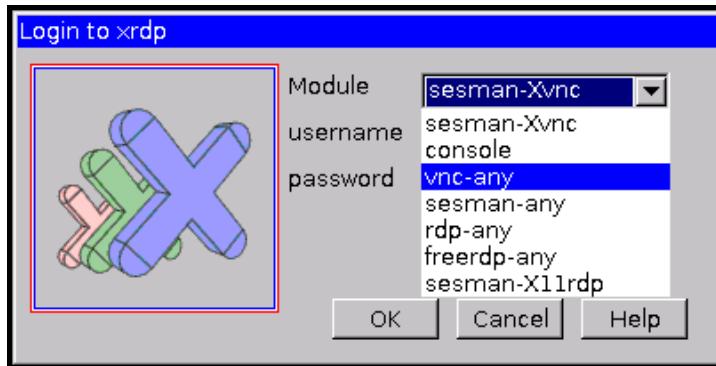
## Note

If you reboot the controller node, repeat the instructions starting from Step 3.

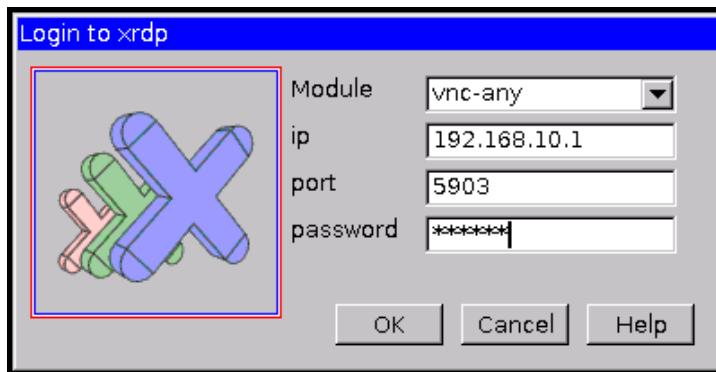
- Connect to your controller node through any remote desktop or VNC client. We will show the **xrdp** method first and the VNC method is below it:
  - Connecting through remote desktop client
    - Login to your remote desktop. You will be prompted with xrdp login screen.



- ii. Click the Module drop-down list and select vnc-any.

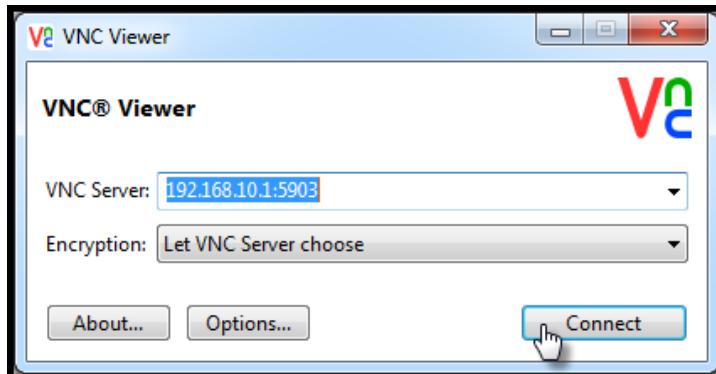


- iii. Enter the IP address, port and password in the respective fields.



- iv. Click **Ok**.

- b. Connecting through a VNC client, such as VNC Viewer [<https://www.realvnc.com/download/viewer/>]:
- Enter the IP address and port and click **Connect**. You will be prompted for your password once the connection is established.



A terminal emulator will be displayed where you can enter the CMC launch command. Note that the CMC launch with this method will have the following limitations: by default, CMC-xterm disables all the title bars and borders. This is an expected behavior.

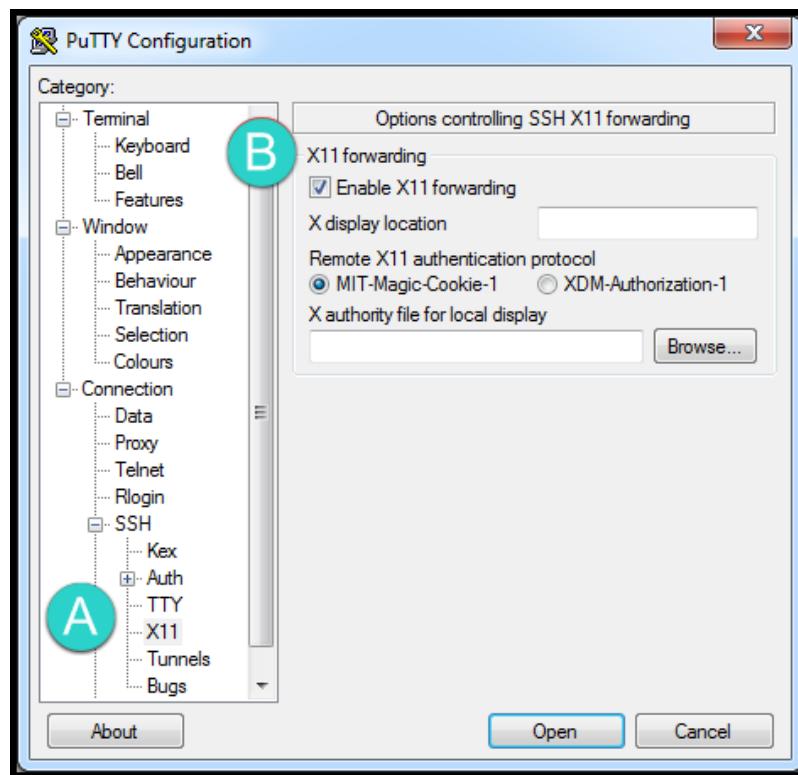
## Install (any) X Display Tool

### Note

You can use SSH to an X server but the performance may be poor.

You must configure an X display tool to launch CMC. You can choose any X display tool. In this section we are using the tool Xming as an example for launching CMC. The following example provides the steps to installing Xming and launching CMC in a Windows environment. Another alternative (not shown in the documentation) is MobaXterm [<http://mobaxterm.mobatek.net/>].

1. Download and install Xming on a Windows machine that can access the lifecycle manager. You can download Xming from <http://sourceforge.net/projects/xming/>.
2. Select Enable X11 forwarding checkbox on the PuTTY session for lifecycle manager. You can do this in PuTTY by:
  - a. Navigate to the Connection+SSH+X11 option in PuTTY.
  - b. Make sure Enable X11 forwarding is activated.



3. SSH to first control plane node.

```
ssh -X
```

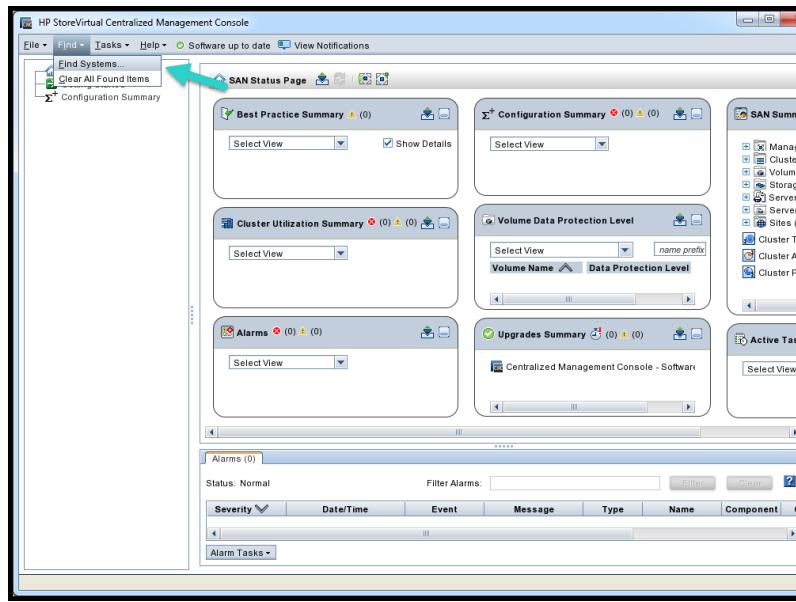
and enter the CMC command (as mentioned below) to launch CMC.

1. Run the following command from your first controller node which will open the CMC GUI on your local machine:

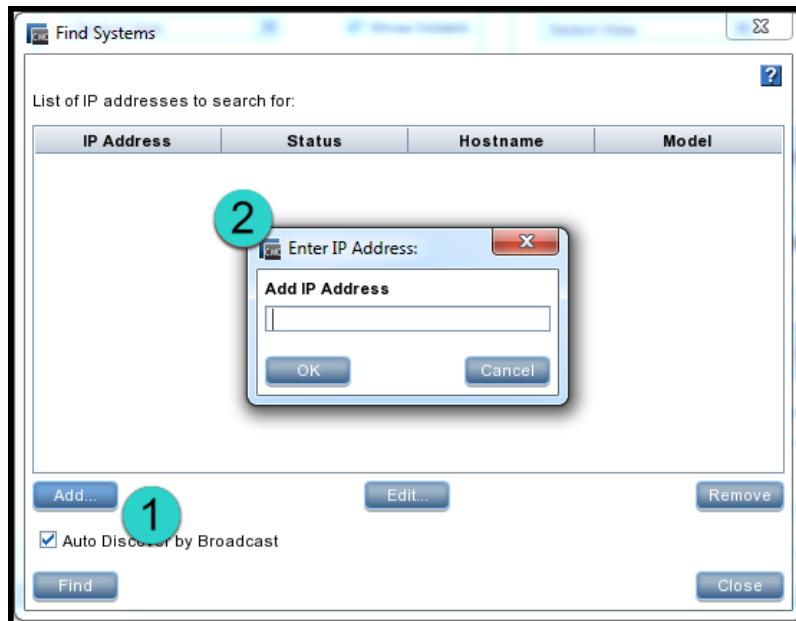
```
/opt/HP/StoreVirtual/UI/jre/bin/java -jar /opt/HP/StoreVirtual/UI/UI.jar
```

By default, the CMC GUI is configured to discover the VSA nodes in the subnet in which it is installed. This discovery functionality of VSA nodes using the CMC controller node is not supported in HPE Helion OpenStack. Instead, you must manually add each VSA node, as shown below.

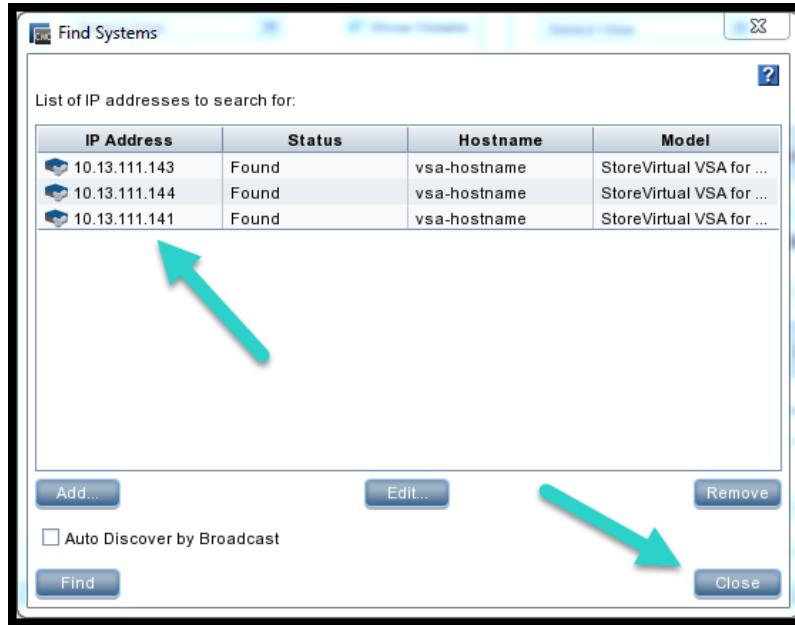
2. In the CMC GUI, click the Find menu and then select the Find Systems options.



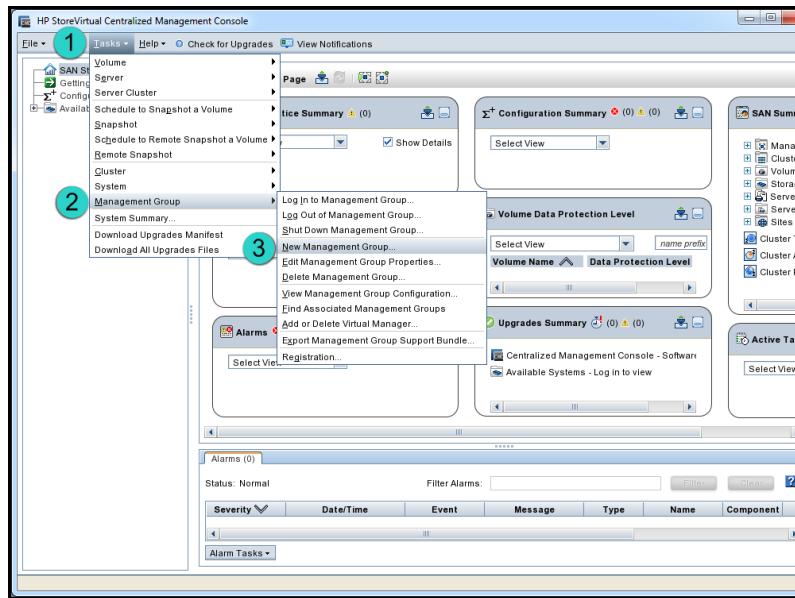
3. Click the Add button which will open the Enter IP Address dialogue box where you can enter the IP address of your VSA nodes which you noted earlier from your ~/scratch/ansible/next/my\_cloud/stage/info/net\_info.yml file.



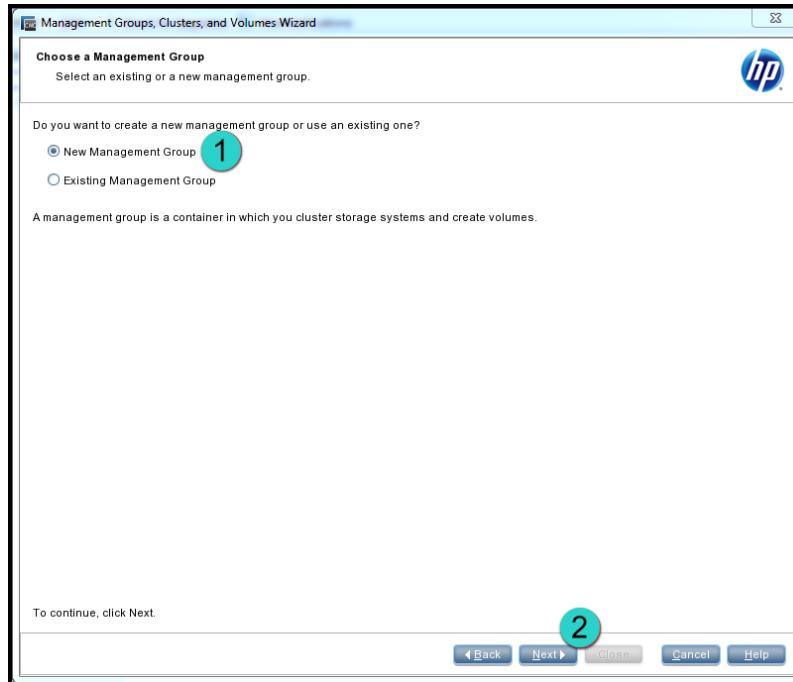
4. Once you have all of your VSA nodes entered, click the **Close** button.



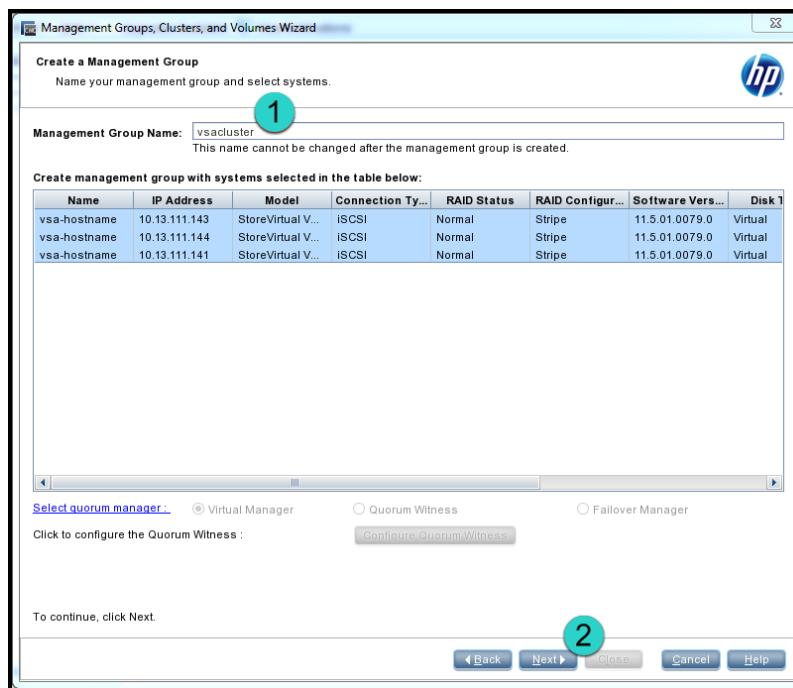
- Next, click the Tasks menu and then navigate to the Management Group submenu and select the New Management Group option.



- In the Management Group wizard, click Next and then select New Management Group and then Next again to continue.



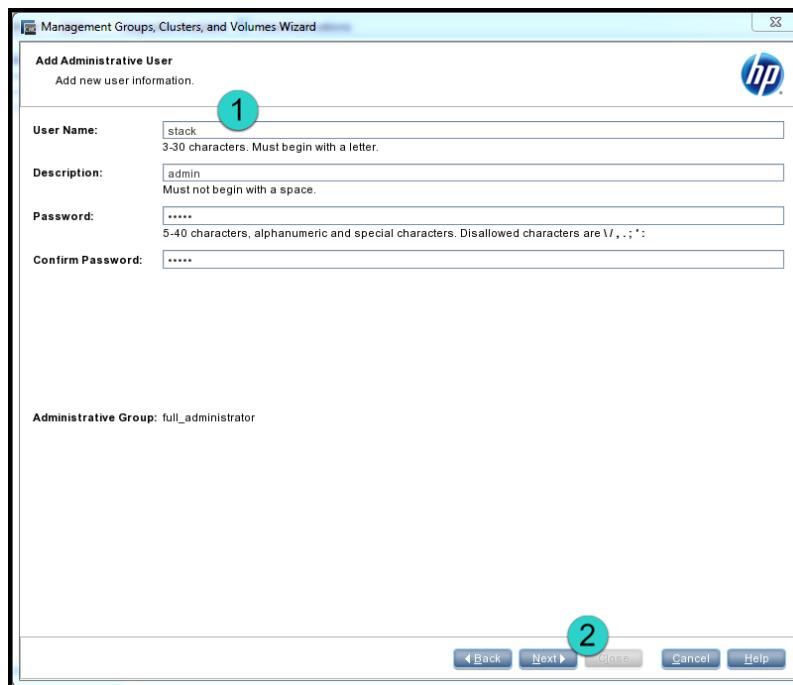
7. Enter a name in the New Management Group Name field and then click Next.



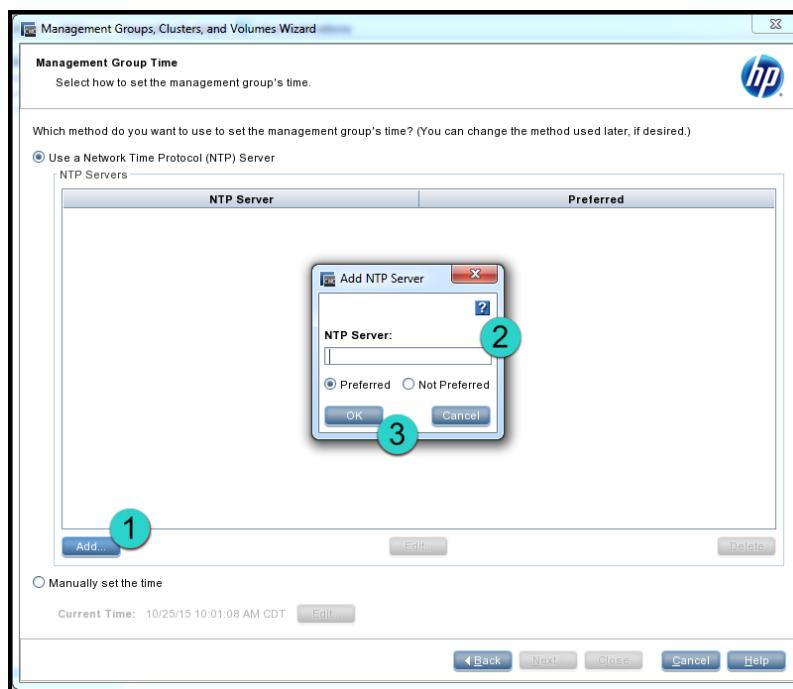
8. On the Add Administrative User, enter a username and password you will use to administer the CMC utility.

## Important

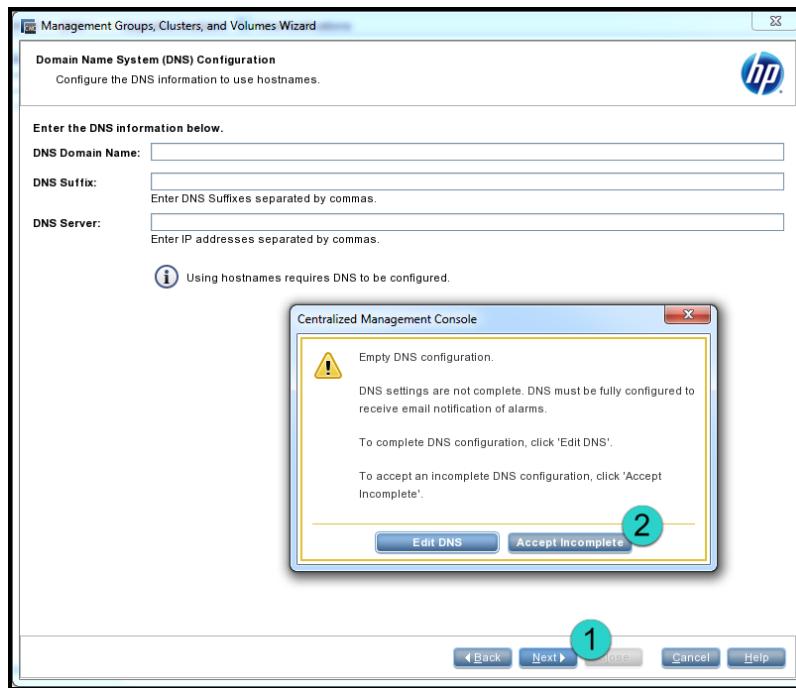
You will need to remember these values as you will input them into your `cinder.conf` file later.



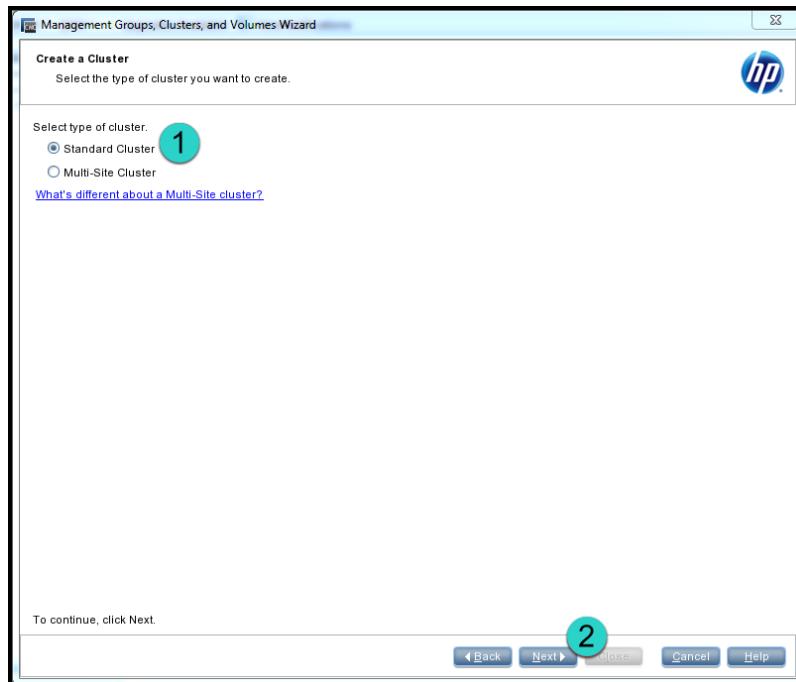
9. Click Nextto display the Management Group Time page.
10. Add your NTP server information and click **Next**



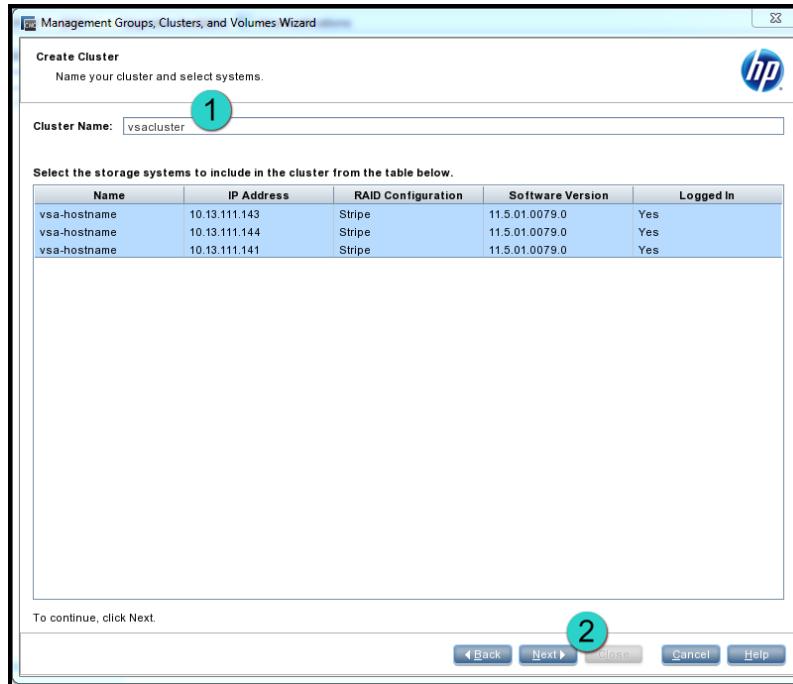
11. Skip the DNS and SMTP sections. To do so, click Next and a popup will display where you can choose the Accept Incomplete option. Repeat this to skip SMTP section as well.



12. On the Create a Cluster options, select Standard Cluster from the displayed options and click Next.



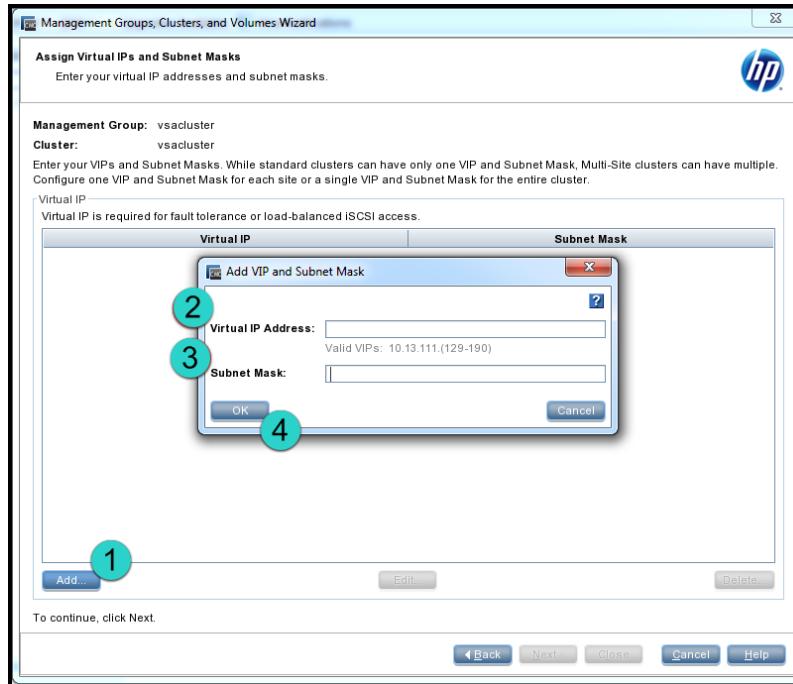
13. In the Cluster Name field, enter a name for the cluster and click Next.



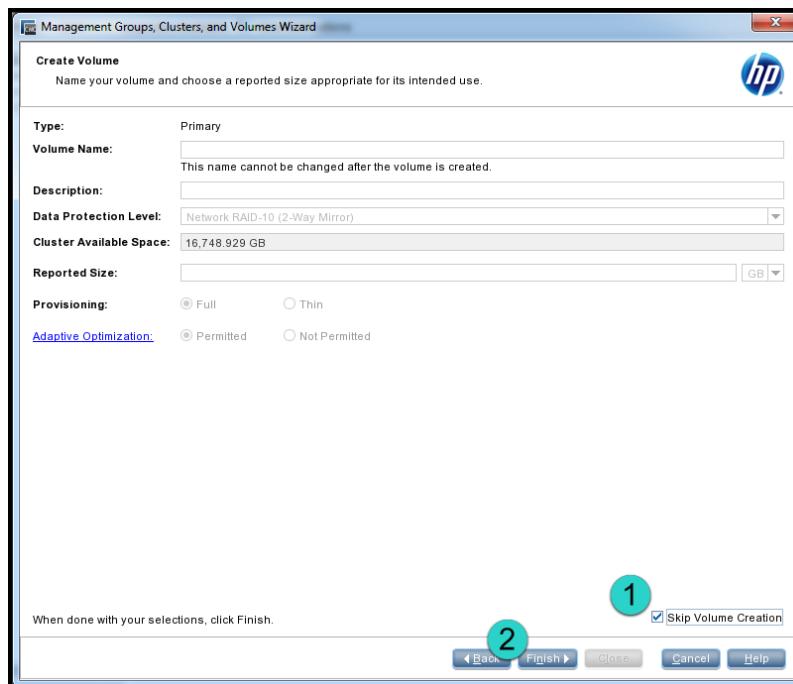
14. On the Assign Virtual IPs and Subnet Masks page, click Add and enter the virtual IP address and subnet mask of the cluster in the respective boxes and click OK.

### Note

The virtual IP address will be found as the `cluster_ip` value in your `~/scratch/ansible/next/my_cloud/stage/info/net_info.yml` file and your subnet mask will be the subnet address from the network your VSA nodes are attached to, usually your MANAGEMENT network.



15. The CMC utility will verify the virtual IP address information and then you can click the Next button.
16. Select the checkbox for Skip Volume Creation and click the Finish button which will display your VSA management cluster.



Cluster creation using CMC takes approximately 10 minutes or longer.

## Tip

You may get a pop-up notice telling you that your hostnames are not unique. This can be ignored by clicking the OK button.

17. If this process is successful you will see a summary page at the end which outlines what you have completed.

## Important

You can create more than one VSA cluster of same or different type by specifying the configuration in cloud model. For more details, refer to [Modifying Cloud Model to Create Multiple Clusters](#)

# Configure VSA as the Backend

You will use the information you input to the CMC utility to configure your Cinder backend to use your VSA environment.

To update your Cinder configuration to add VSA storage you must modify the `~/helion/my_cloud/config/cinder/cinder.conf` file on your lifecycle manager as follows:

1. Log in to the lifecycle manager.
2. Make the following changes to the `~/helion/my_cloud/config/cinder/cinder.conf` file:
  - a. Add your VSA backend to the `enabled_backends` section:

```
# Configure the enabled backends
enabled_backends=vsa-1
```

## Important

If you are using multiple backend types, you can use a comma delimited list here. For example, if you are going to use both VSA and Ceph backends, you would specify something like this: `enabled_backends=vsa-1,ceph1`.

- b. [OPTIONAL] If you want your volumes to use a default volume type, then enter the name of the volume type in the `[ DEFAULT ]` section with the syntax below. **Remember this value when you create your volume type in the next section.**

## Important

If you do not specify a default type then your volumes will default to a non-redundant RAID configuration. It is recommended that you create a volume type and specify it here that meets your environments needs.

```
[ DEFAULT ]
# Set the default volume type
default_volume_type = <your new volume type>
```

- c. Uncomment the StoreVirtual (VSA) cluster section and fill the values as per your cluster information. If you have more than one cluster, you will need to add another similar section with its respective values. In the following example only one cluster is added.

```
[vsal-1]
hplefthand_password: <vsal-cluster-password>
hplefthand_clustername: <vsal-cluster-name>
hplefthand_api_url: https://<vsal-cluster-vip>:8081/lhos
hplefthand_username: <vsal-cluster-username>
hplefthand_iscsi_chap_enabled: false
volume_backend_name: <vsal-backend-name>
volume_driver: cinder.volume.drivers.san.hp.hp_lefthand_iscsi.HPLeftHandISCS
hplefthand_debug: false
```

where:

| Value                         | Description                                                                                                                                                                                                               |
|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| hplefthand_password           | Password entered during cluster creation in the CMC utility. If you have chosen to encrypt this password, enter the value in this format:<br><br>hplefthand_password: {{ '<encrypted vsal-cluster-password>' }}           |
| hplefthand_clustername        | Name of the VSA cluster provided while creating a cluster in the CMC utility.                                                                                                                                             |
| hplefthand_api_url            | Virtual IP address of your VSA cluster, found in your ~/scratch/ansible/next/my_cloud/stage/info/net_info.yml file.                                                                                                       |
| hplefthand_username           | Username given during cluster creation in the CMC utility.                                                                                                                                                                |
| hplefthand_iscsi_chap_enabled | If you set this option as true then the hosts will not be able to access the storage without the generated secrets. And if you set this option as false then no CHAP authentication is required for the ISCSI connection. |
| volume_backend_name           | Name given to the VSA backend. You will specify this value later in the Associate the Volume Type to a Backend steps.                                                                                                     |
| volume_driver                 | Cinder volume driver. Leave this as the default value for VSA.                                                                                                                                                            |
| hplefthand_debug              | If you set this option as true then the Cinder driver for the VSA will generate logging in debug mode; these logging entries can be found in cinder-volume.log.                                                           |

[OPTIONAL] HPE Helion OpenStack 5.0 supports VSA deployment for KVM hypervisor only but it can be used as pre-deployed (or out of the band deployed) Lefthand storage boxes or VSA appliances (running on ESX/hyper-v/KVM hypervisor). It also supports Cinder configuration

of physical Lefthand storage device and VSA appliances. Depending upon your setup, you will have to edit the below section if your storage array is running LeftHand OS lower than version 11:

```
[<unique-section-name>]
volume_driver=cinder.volume.drivers.san.hp.hp_lefthand_iscsi.HPLeftHandISCS
volume_backend_name=lefthand-cliq
san_ip=<san-ip>
san_login=<san_username>
If adding a password here, then the password can be encrypted using the
mechanism specified in the documentation. If the password has been encrypted
add the value and the hos_user_password_decrypt filter like so:
san_password= {{ '<encrypted san_password>' | hos_user_password_decrypt }}
Note that the encrypted value has to be enclosed in quotes
If you choose not to encrypt the password then the unencrypted password
must be set as follows:
san_password=<san_password>
san_ssh_port=16022
san_clustername=<vsas-cluster-name>
volume_backend_name=<vsas-backend-name>
```

## Important

Similar to your hplefthand\_password in the previous example, encryption for your san\_password is supported. If you chose to use encryption you would use the syntax below to express that:

```
san_password= {{ '<encrypted san_password>' | hos_user_password_decrypt }}
```

See for more details.

## Important

Do not use backend\_host variable in cinder.conf file. If backend\_host is set, it will override the [DEFAULT]/host value which HPE Helion OpenStack 5.0 is dependent on.

3. Commit your configuration to a local repository (Chapter 3, *Using Git for Configuration Management*):

```
cd ~/helion/hos/ansible
git add -A
git commit -m "configured VSA backend"
```

## Note

Before you run any playbooks, remember that you need to export the encryption key in the following environment variable:

```
export HOS_USER_PASSWORD_ENCRYPT_KEY=ENCRYPTION_KEY
```

For more information, see Chapter 10, *Installing Mid-scale and Entry-scale KVM*.

4. Run the configuration processor:

```
cd ~/helion/hos/ansible
```

```
ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Run the following command to create a deployment directory:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the Cinder Reconfigure Playbook:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
```

## Post-Installation Tasks

After you have configured VSA as your Block Storage backend, here are some tasks you will want to complete:

- - the section called “Verifying Your Block Storage Backend”

## Configuring for 3PAR Block Storage Backend

### Abstract

This page describes how to configure your 3PAR backend for the HPE Helion Entry-scale with KVM cloud model.

## Prerequisites

- You must have the license for the following software before you start your 3PAR backend configuration for the Helion Entry-scale with KVM cloud model:
  - Thin Provisioning
  - Virtual Copy
  - System Reporter
  - Dynamic Optimization
  - Priority Optimization
- Your HPE Helion Entry-scale KVM Cloud should be up and running. Installation steps can be found in Chapter 10, *Installing Mid-scale and Entry-scale KVM*.
- Your 3PAR Storage Array should be available in the cloud management network or routed to the cloud management network and the 3PAR FC and iSCSI ports configured.
- The 3PAR management IP and iSCSI port IPs must have connectivity from the controller and compute nodes.
- Please refer to the system requirements for 3PAR in the OpenStack configuration guide, which can be found here: 3PAR System Requirements [<http://docs.openstack.org/liberty/config-reference/content/hp-3par-sys-reqs.html>].

## Notes

**Encrypted 3Par Volume:** Attaching an encrypted 3Par volume is possible after installation by setting `volume_use_multipath = true` under the `libvirt` stanza in the `nova/kvm-hypervisor.conf.j2` file and reconfigure nova.

**Concerning using multiple backends:** If you are using multiple backend options, ensure that you specify each of the backends you are using when configuring your `cinder.conf.j2` file using a comma delimited list. An example would be `enabled_backends=vsa-1,3par_iSCSI,ceph1` and is included in the steps below. You will also want to create multiple volume types so you can specify which backend you want to utilize when creating volumes. These instructions are included below as well. In addition to our documentation, you can also read the OpenStack documentation at Configure multiple storage backends [[http://docs.openstack.org/admin-guide-cloud/blockstorage\\_multi\\_backend.html](http://docs.openstack.org/admin-guide-cloud/blockstorage_multi_backend.html)] as well.

**Concerning iSCSI and Fiber Channel:** You should not configure cinder backends so that multipath volumes are exported over both iSCSI and Fiber Channel from a 3PAR backend to the same Nova compute server.

**3PAR driver has updated name:** In the OpenStack Mitaka release, the 3PAR driver used for HPE Helion integration had its name updated from `HP3PARFCDriver` and `HP3PARISCSIDriver` to `HPE3PARFCDriver` and `HPE3PARISCSIDriver`. To prevent issues when upgrading from previous HPE Helion OpenStack releases, we left the names as-is in the release and provided a mapping so that the integration would continue to work. This will produce a warning in the `cinder-volume.log` file advising you of the deprecated name. The warning will look similar to this:

```
Option "hp3par_api_url" from group "<your section>" is deprecated. Use option "hpe
```

These are just warnings and can be ignored.

## Multipath Support

If you want to enable multipath for Cinder volumes carved from 3PAR FC/iSCSI storage system, please go through the `~/helion/hos/ansible/roles/multipath/README.md` file on the lifecycle manager. The `README.md` file contains detailed procedures for configuring multipath for 3PAR FC/iSCSI Cinder volumes.

We have also included an additional set of steps needed if you are using 3PAR FC/iSCSI multipath which is included below.

If you are using 3PAR FC/iSCSI multipath, an additional configuration is required:

### Note

If you are planning on attaching an encrypted 3Par volume after installation, ensure that you `volume_use_multipath = true` under the `libvirt` section in the `nova/kvm-hypervisor.conf.j2` file before configuring Cinder.

1. Log in to the lifecycle manager.
2. Edit the `~/helion/my_cloud/config/nova/kvm-hypervisor.conf.j2` file add this line under the `[libvirt]` section:

Example:

```
[libvirt]
...
iscsi_use_multipath=true
```

3. Edit the `~/helion/my_cloud/config/cinder/cinder.conf.j2` file add this line under the `[DEFAULT]` section:

Example:

```
[DEFAULT]
...
use_multipath_for_image_xfer=true
```

4. Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*), as follows:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

5. Run the configuration processor:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

6. Use the playbook below to create a deployment directory:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

7. Run the Nova reconfigure playbook:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts nova-reconfigure.yml
```

## Configure 3PAR FC as a Cinder Backend

You must modify the `cinder.conf.j2` to configure the FC details.

Perform the following steps to configure 3PAR FC as Cinder backend:

1. Log in to lifecycle manager.
2. Make the following changes to the `~/helion/my_cloud/config/cinder/cinder.conf.j2` file:
  - a. Add your 3PAR backend to the `enabled_backends` section:

```
# Configure the enabled backends
enabled_backends=3par_FC
```

### Important

If you are using multiple backend types, you can use a comma delimited list here. For example, if you are going to use both VSA and 3par backends, you would specify something like this: `enabled_backends=vsa-1,3par_FC`.

- b. [OPTIONAL] If you want your volumes to use a default volume type, then enter the name of the volume type in the `[DEFAULT]` section with the syntax below. **You will want to remember this value when you create your volume type in the next section.**

## Important

If you do not specify a default type then your volumes will default to a non-redundant RAID configuration. It is recommended that you create a volume type and specify it here that meets your environments needs.

```
[DEFAULT]
# Set the default volume type
default_volume_type = <your new volume type>
```

- c. Uncomment the StoreServ (3par) iscsi cluster section and fill the values per your cluster information. Here is an example:

```
[3par_FC]
san_ip: <3par-san-ipaddr>
san_login: <3par-san-username>
san_password: <3par-san-password>
hp3par_username: <3par-username>
hp3par_password: <hp3par_password>
hp3par_api_url: https://<3par-san-ipaddr>:8080/api/v1
hp3par_cpg: <3par-cpg-name-1>[,<3par-cpg-name-2>, ...]
volume_backend_name: <3par-backend-name>
volume_driver: cinder.volume.drivers.san.hp.hp_3par_fc.HP3PARFCDriver
```

## Important

Do not use backend\_host variable in `cinder.conf` file. If `backend_host` is set, it will override the [DEFAULT]/host value which HPE Helion OpenStack 5.0 is dependent on.

3. Commit your configuration to the local git repo (Chapter 3, *Using Git for Configuration Management*), as follows:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "My config or other commit message"
```

4. Run the configuration processor:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

5. Update your deployment directory:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the following playbook to complete the configuration:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
```

## Configure 3PAR iSCSI as Cinder backend

You must modify the `cinder.conf.j2` to configure the iSCSI details.

Perform the following steps to configure 3PAR iSCSI as Cinder backend:

1. Log in to lifecycle manager.
2. Make the following changes to the `~/helion/my_cloud/config/cinder/cinder.conf.j2` file:
  - a. Add your 3PAR backend to the `enabled_backends` section:

```
# Configure the enabled backends
enabled_backends=3par_iSCSI
```

- b. Uncomment the `StoreServ (3par) iscsi cluster` section and fill the values per your cluster information. Here is an example:

```
[3par_iSCSI]
san_ip: <3par-san-ipaddr>
san_login: <3par-san-username>
san_password: <3par-san-password>
hp3par_username: <3par-username>
hp3par_password: <hp3par_password>
hp3par_api_url: https://<3par-san-ipaddr>:8080/api/v1
hp3par_cpg: <3par-cpg-name-1>[,<3par-cpg-name-2>, ...]
volume_backend_name: <3par-backend-name>
volume_driver: cinder.volume.drivers.san.hp.HP3PARISCSIDriver
hp3par_iscsi_ips: <3par-ip-address-1>[,<3par-ip-address-2>,<3par-ip-address-3>]
hp3par_iscsi_chap_enabled=true
```

## Important

Do not use `backend_host` variable in `cinder.conf` file. If `backend_host` is set, it will override the `[DEFAULT]/host` value which HPE Helion OpenStack 5.0 is dependent on.

3. Commit your configuration to your local git repository:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "<commit message>"
```

4. Run the configuration processor:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml
```

When you run the configuration processor you will be prompted for two passwords. Enter the first password to make the configuration processor encrypt its sensitive data, which consists of the random inter-service passwords that it generates and the Ansible `group_vars` and `host_vars` that it produces for subsequent deploy runs. You will need this key for subsequent Ansible deploy runs and subsequent configuration processor runs. If you wish to change an encryption password that you have already used when running the configuration processor then enter the new password at the second prompt, otherwise press **Enter**.

For CI purposes you can specify the required passwords on the ansible command line. For example, the command below will disable encryption by the configuration processor

```
ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e re
```

If you receive an error during either of these steps then there is an issue with one or more of your configuration files. We recommend that you verify that all of the information in each of your configuration files is correct for your environment and then commit those changes to git using the instructions above.

5. Run the following command to create a deployment directory.

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

6. Run the following command to complete the configuration:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts cinder-reconfigure.yml
```

## Post-Installation Tasks

After you have configured 3PAR as your Block Storage backend, here are some tasks you will want to complete:

- - the section called “Verifying Your Block Storage Backend”

## Ironic OneView Integration

HPE Helion OpenStack 5.0 supports integration of Ironic (Baremetal) service with HPE OneView using *agent\_pxe\_oneview* driver. Please refer to OpenStack Documentation [<https://docs.openstack.org/developer/ironic/drivers/oneview.html>] for more information.

## Prerequisites

1. Installed HPE Helion OpenStack 5.0 with entry-scale-ironic-flat-network or entry-scale-ironic-multi-tenancy model.
2. OneView 3.0 instance is running and connected to management network.
3. OneView configuration is set into `definition/data/ironic/ironic_config.yml` (and `ironic-reconfigure.yml` playbook ran if needed). This should enable *agent\_pxe\_oneview* driver in ironic conductor.
4. Managed node(s) should support PXE booting in legacy BIOS mode.
5. Managed node(s) should have PXE boot NIC listed first. That is, embedded 1Gb NIC must be disabled (otherwise it always goes first).

## Integrating with OneView

1. On the lifecycle manager, open the file `~/helion/my_cloud/definition/data/ironic/ironic_config.yml`

```
~$ cd ~/helion  
~/helion$ vi my_cloud/definition/data/ironic/ironic_config.yml
```

2. Modify the settings listed below:

- a. enable\_oneview: should be set to "true" for OneView integration
  - b. oneview\_manager\_url: HTTPS endpoint of OneView management interface, for example:  
**https://10.0.0.10/**
  - c. oneview\_username: OneView username, for example: **Administrator**
  - d. oneview\_encrypted\_password: OneView password in encrypted or clear text form. Encrypted form is distinguished by presence of '@hos@' at the beginning of the string. Encrypted form can be created by running hosencrypt.py program. This program is shipped as part of HPE Helion OpenStack and can be found in ~/helion/hos/ansible directory on lifecycle manager.
  - e. oneview\_allow\_insecure\_connections: should be set to "true" if OneView is using self-generated certificate.
3. Once you have saved your changes and exited the editor, add files, commit changes to local git repository, and run config-processor-run.yml and ready-deployment.yml playbooks, as described in Chapter 3, *Using Git for Configuration Management*.

```
~/helion$ git add my_cloud/definition/data/ironic/ironic_config.yml  
~/helion$ cd hos/ansible  
~/helion/hos/ansible$ ansible-playbook -i hosts/localhost config-processor-run.yml  
...  
~/helion/hos/ansible$ ansible-playbook -i hosts/localhost ready-deployment.yml
```

4. Run ironic-reconfigure.yml playbook.

```
$ cd ~/scratch/ansible/next/hos/ansible/  
  
# This is needed if password was encrypted in ironic_config.yml file  
~/scratch/ansible/next/hos/ansible$ export HOS_USER_PASSWORD_ENCRYPT_KEY=your_password  
  
~/scratch/ansible/next/hos/ansible$ ansible-playbook -i hosts/verb_hosts ironic-reconfigure.yml  
...
```

## Registering Node in OneView

In the OneView web interface:

1. Navigate to Menu -> **Server Hardware**. Add new **Server Hardware** item, using managed node iLO IP and credentials. If this is the first node of this type being added, corresponding **Server Hardware Type** will be created automatically.
2. Navigate to Menu -> **Server Profile Template**. Add **Server Profile Template**. Use **Server Hardware Type** corresponding to node being registered. In **BIOS Settings** section, set **Manage Boot Mode** and **Manage Boot Order** options must be turned on:

The screenshot shows the 'Edit server-profile-template-1' interface. At the top, there's a 'Connections' dropdown menu. Below it, under 'Local Storage', it says 'Integrated storage controller mode' and 'managed manually'. Under 'SAN Storage', it says 'not supported for this server hardware type'. In the 'Boot Settings' section, there are two checkboxes: 'Manage boot mode' (checked) and 'Manage boot order' (checked). The 'Boot mode' dropdown is set to 'Legacy BIOS'. Below these, a list of boot devices is shown: 1. CD, 2. USB, 3. Hard disk, 4. PXE. A note at the bottom says 'Drag and drop or edit rows to re-order'.

- Verify that node is powered off. Power the node off if needed.

## Provisioning Ironic Node

- Login to the lifecycle manager and source respective credentials file (e.g. service.osrc for admin account).
- Review glance images with `glance image-list`

```
$ glance image-list
+-----+-----+
| ID | Name |
+-----+-----+
c61da588-622c-4285-878f-7b86d87772da	cirros-0.3.4-x86_64
6616ef6a-ee7a-478c-b1bf-4c93088a123f	ir-deploy-iso-HOS5.0
633d379d-e076-47e6-b56d-582b5b977683	ir-deploy-kernel-HOS5.0
d5828785-edf2-49fa-8de2-3ddb7f3270d5	ir-deploy-ramdisk-HOS5.0
d6b5d971-42fd-492b-b33e-6e5f2d88e942	Ubuntu Trusty 14.04 BIOS
+-----+-----+
```

Ironic deploy images ("ir-deploy-\*") were created automatically by HOS installer. We will be using **agent\_pxe\_oneview** Ironic driver which needs **ir-deploy-kernel** and **ir-deploy-ramdisk** images. Managed node will be deployed using **Ubuntu Trusty 14.04 BIOS** image.

3. Create node using `agent_pxe_oneview` driver.

```
$ ironic --ironic-api-version 1.22 node-create -d agent_pxe_oneview --name test-node-1 \
    -p 'capabilities=boot_mode:bios,boot_option:local,server_hardware_type_uri:/rest/server-hardware-types/E5366BF8-7CBF-48DF-A752-8670CF780BB2,server_profile_template_uri:/rest/server-profile-templates/00614918-77f8-4146-a8b8-9fc276cd6ab2' \
    -i 'server_hardware_uri=/rest/server-hardware/32353537-3835-584D-5135-313930373046' \
    -i dynamic_allocation=True \
    -i deploy_kernel=633d379d-e076-47e6-b56d-582b5b977683 \
    -i deploy_ramdisk=d5828785-edf2-49fa-8de2-3ddb7f3270d5
```

| Property          | Value                                                                                                                                                                                                                                                                                                          |
|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| chassis_uuid      |                                                                                                                                                                                                                                                                                                                |
| driver            | agent_pxe_oneview                                                                                                                                                                                                                                                                                              |
| driver_info       | {u'server_hardware_uri': u'/rest/server-hardware/32353537-3835-584D-5135-313930373046', u'dynamic_allocation': u'True', u'deploy_ramdisk': u'd5828785-edf2-49fa-8de2-3ddb7f3270d5', u'deploy_kernel': u'633d379d-e076-47e6-b56d-582b5b977683'}                                                                 |
| extra             | {}                                                                                                                                                                                                                                                                                                             |
| name              | test-node-1                                                                                                                                                                                                                                                                                                    |
| network_interface | neutron                                                                                                                                                                                                                                                                                                        |
| properties        | {u'memory_mb': 131072, u'cpu_arch': u'x86_64', u'local_gb': 2, u'capabilities': u'boot_mode:bios,boot_option:local,server_hardware_type_uri:/rest/server-hardware-types/E5366BF8-7CBF-48DF-A752-8670CF780BB2,server_profile_template_uri:/rest/server-profile-templates/00614918-77f8-4146-a8b8-9fc276cd6ab2'} |
| resource_class    | None                                                                                                                                                                                                                                                                                                           |
| uuid              | c202309c-97e2-4c90-8ae3-d4c95afdaf06                                                                                                                                                                                                                                                                           |

## Note

- For deployments created via Ironic/OneView integration, `memory_mb` property must reflect physical amount of RAM installed in managed node. That is, for a server with 128 Gb of RAM it works out to  $128 * 1024 = 13072$ .
- Boot mode in `capabilities` property must reflect boot mode used by the server, i.e. '`bios`' for Legacy BIOS and '`uefi`' for UEFI.
- Values for `server_hardware_type_uri`, `server_profile_template_uri` and `server_hardware_uri` can be grabbed from browser URL field while navigating to respective objects in OneView UI. URI corresponds to the part of URL which starts from the token '`/rest`'. That is, URL `https://oneview.mycorp.net/#/profile-templates/show/overview/r/rest/server-profile-templates/12345678-90ab-cdef-0123-012345678901` corresponds to URI `/rest/server-profile-templates/12345678-90ab-cdef-0123-012345678901`.
- Grab IDs of `deploy_kernel` and `deploy_ramdisk` from **glance image-list** output above.

4. Create port.

```
$ ironic --ironic-api-version 1.22 port-create \
    --address aa:bb:cc:dd:ee:ff \
    --node c202309c-97e2-4c90-8ae3-d4c95afdaf06 \
    -l switch_id=ff:ee:dd:cc:bb:aa \
    -l switch_info=MY_SWITCH \
    -l port_id="Ten-GigabitEthernet 1/0/1" \
    --pxe-enabled true

+-----+-----+
| Property | Value
+-----+-----+
| address | 8c:dc:d4:b5:7d:1c
| extra | {}
| local_link_connection | {u'switch_info': u'C20DATA', u'port_id': u'Ten-GigabitEthernet 1/0/1', u'switch_id': u'ff:ee:dd:cc:bb:aa'}
| node_uuid | c202309c-97e2-4c90-8ae3-d4c95afdaf06
| pxe_enabled | True
| uuid | 75b150ef-8220-4e97-ac62-d15548dc8ebe
+-----+-----+
```

## Warning

Ironic Multi-Tenancy networking model is used in this example. Therefore, ironic port-create command contains information about the physical switch. OneView integration can also be performed using the Ironic Flat Networking model. For more information, see the section called “Ironic Examples”.

5. Move node to manageable provisioning state. The connectivity between Ironic and OneView will be verified, Server Hardware Template settings validated, and Server Hardware power status retrieved from OneView and set into the Ironic node.

```
$ ironic node-set-provision-state test-node-1 manage
```

6. Verify that node power status is populated.

```
$ ironic node-show test-node-1
+-----+-----+
| Property | Value
+-----+-----+
| chassis_uuid | 
| clean_step | {}
| console_enabled | False
| created_at | 2017-06-30T21:00:26+00:00
| driver | agent_pxe_oneview
| driver_info | {u'server_hardware_uri': u'/rest/server-hardware/32353537-3835-584D-5135-313930373046', u'dynamic': u'True', u'deploy_ramdisk': u'd5828785-edf2-49fa-8de2-633d379d-e076-47e6-b56d-582b5b977', u'deploy_kernel': u'633d379d-e076-47e6-b56d-582b5b977'}
| driver_internal_info | {}
| extra | {}
| inspection_finished_at | None
| inspection_started_at | None
| instance_info | {}
| instance_uuid | None
+-----+-----+
```

|                        |                                                                                                                                                                                                                                                                                           |
|------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| last_error             | None                                                                                                                                                                                                                                                                                      |
| maintenance            | False                                                                                                                                                                                                                                                                                     |
| maintenance_reason     | None                                                                                                                                                                                                                                                                                      |
| name                   | test-node-1                                                                                                                                                                                                                                                                               |
| network_interface      |                                                                                                                                                                                                                                                                                           |
| power_state            | power off                                                                                                                                                                                                                                                                                 |
| properties             | {u'memory_mb': 131072, u'cpu_arch': u'x86_64', u'local_gb': 2, u'capabilities': u'boot_mode:bios,boot_option:local,server_hardware_type:server-hardware-types/E5366BF8-7CBF-48DF-A752-8670CF780BB2,server_profile_template_uri:/redacted/templates/00614918-77f8-4146-a8b8-9fc276cd6ab2'} |
| provision_state        | manageable                                                                                                                                                                                                                                                                                |
| provision_updated_at   | 2017-06-30T21:04:43+00:00                                                                                                                                                                                                                                                                 |
| raid_config            |                                                                                                                                                                                                                                                                                           |
| reservation            | None                                                                                                                                                                                                                                                                                      |
| resource_class         |                                                                                                                                                                                                                                                                                           |
| target_power_state     | None                                                                                                                                                                                                                                                                                      |
| target_provision_state | None                                                                                                                                                                                                                                                                                      |
| target_raid_config     |                                                                                                                                                                                                                                                                                           |
| updated_at             | 2017-06-30T21:04:43+00:00                                                                                                                                                                                                                                                                 |
| uuid                   | c202309c-97e2-4c90-8ae3-d4c95afdaf06                                                                                                                                                                                                                                                      |

7. Move node to available provisioning state. The Ironic node will be reported to Nova as available.

```
$ ironic node-set-provision-state test-node-1 provide
```

8. Verify that node resources were added to Nova hypervisor stats.

| \$ nova hypervisor-stats |        |
|--------------------------|--------|
| Property                 | Value  |
| count                    | 1      |
| current_workload         | 0      |
| disk_available_least     | 80     |
| free_disk_gb             | 80     |
| free_ram_mb              | 131072 |
| local_gb                 | 80     |
| local_gb_used            | 0      |
| memory_mb                | 131072 |
| memory_mb_used           | 0      |
| running_vms              | 0      |
| vcpus                    | 2      |
| vcpus_used               | 0      |

9. Create Nova flavor.

```
$ nova flavor-create m1.ironic auto 131072 80 2
```

| ID                                   | Name      | Memory_MB | Disk | Ephemera |
|--------------------------------------|-----------|-----------|------|----------|
| 33c81884-b8aa-46b7-ab3f-076f3b72f8d8 | m1.ironic | 131072    | 80   | 0        |

```
+-----+-----+-----+
$ nova flavor-key m1.ironic set capabilities:boot_mode="bios"
$ nova flavor-key m1.ironic set capabilities:boot_option="local"
$ nova flavor-key m1.ironic set cpu_arch=x86_64
```

## Note

All parameters (specifically, amount of RAM and boot mode) must correspond to ironic node parameters.

10.Create Nova keypair if needed.

```
$ nova keypair-add ironic_kp --pub-key ~/.ssh/id_rsa.pub
```

11.Boot Nova instance.

| Property                             | Value                                   |
|--------------------------------------|-----------------------------------------|
| OS-DCF:diskConfig                    | MANUAL                                  |
| OS-EXT-AZ:availability_zone          | -                                       |
| OS-EXT-SRV-ATTR:host                 | -                                       |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                       |
| OS-EXT-SRV-ATTR:instance_name        | -                                       |
| OS-EXT-STS:power_state               | 0                                       |
| OS-EXT-STS:task_state                | scheduling                              |
| OS-EXT-STS:vm_state                  | building                                |
| OS-SRV-USG:launched_at               | -                                       |
| OS-SRV-USG:terminated_at             | -                                       |
| accessIPv4                           |                                         |
| accessIPv6                           |                                         |
| adminPass                            | pE3m7wRACvYy                            |
| config_drive                         |                                         |
| created                              | 2017-06-30T21:08:42Z                    |
| flavor                               | m1.ironic (33c81884-b8aa-46b7-ab3f-076f |
| hostId                               | b47c9f2a-e88e-411a-abcd-6172aea45397    |
| id                                   | Ubuntu Trusty 14.04 BIOS (d6b5d971-42fd |
| image                                |                                         |
| key_name                             | ironic_kp                               |
| metadata                             | {}                                      |
| name                                 | test-node-1                             |
| os-extended-volumes:volumes_attached | []                                      |
| progress                             | 0                                       |
| security_groups                      | default                                 |
| status                               | BUILD                                   |
| tenant_id                            | c8573f7026d24093b40c769ca238fddc        |
| updated                              | 2017-06-30T21:08:42Z                    |
| user_id                              | 2eae99221545466d8f175eeb566cc1b4        |

During nova instance boot, the following operations will be performed by Ironic via OneView REST API.

- In OneView, new Server Profile is generated for specified Server Hardware, using specified Server Profile Template. Boot order in Server Profile is set to list PXE as the first boot source.
- Managed node is powered on and boots IPA image from PXE.
- IPA image writes user image onto disk and reports success back to ironic.
- ironic modifies Server Profile in OneView to list 'Disk' as default boot option.
- ironic reboots the node (via OneView REST API call).

# Chapter 18. Troubleshooting the Installation

We have gathered some of the common issues that occur during installation and organized them by when they occur during the installation. These sections will coincide with the steps labeled in the installation instructions.

- the section called “Issues during Lifecycle-manager Setup”
  - the section called “Issues while Provisioning your Baremetal Nodes”
  - the section called “Issues while Updating Configuration Files”
  - the section called “Issues while Deploying the Cloud”
  - Chapter 19, *Troubleshooting the Block Storage Backend Configuration*

# **Issues during Lifecycle-manager Setup**

**Issue: Running the hos-init.bash script when configuring your lifecycle manager does not complete**

Part of what the `~/hos-3.0.0/hos-init.bash` script does is install git and so if your DNS name-server(s) is not specified in your `/etc/resolv.conf` file, is not valid, or is not functioning properly on your lifecycle manager then it won't be able to complete.

To resolve this issue, double check your nameserver in your /etc/resolv.conf file and then re-run the script.

# Issues while Provisioning your Baremetal Nodes

**Issue:** The `bm-power-status.yml` playbook returns an error.

If the output of the `bm-power-status.yml` playbook gives you an error like the one below for one of your servers then the most likely cause is an issue with the information located in the `~/helion/my_cloud/definition/servers.yml` file. It could be that the information is incorrect or the iLO username you are using does not have administrator privileges. Verify all of the information for that server and recommit the changes to git and re-run the playbook if the information was incorrect. If it's a rights issue, ensure the iLO user has administrative privileges.

Error:

```
failed: [vsal] => {"cmd": "ipmitool -I lanplus -E -N 5 -R 12 -U iLOAdmin -H 10.12.1.100 -O raw -r 0x1000000000000000 -w 0x1000000000000000", "stderr": "Error: Unable to establish IPMI v2 / RMCP+ session"}
```

This error below, which may come with running the `bm-power-status.yml` playbook, may be resolved by placing the `ilo-password` values in your `servers.yml` file in double quotes.

Error:

```
failed: [controller2] => { "failed": true }
```

```
msg: ipmi: 'int' object has no attribute 'startswith'
```

### Cobbler Deployment Fails due to CIDR Error

The error will look similar to this:

```
TASK: [cobbler | set variables | Find baremetal nic] ****
failed: [localhost] => {"failed": true}
msg: matchcidr: no nic matching 10.242.115.0/24
  lo 127.0.0.1/8
  eth0 10.242.115.18/26
```

The most likely cause of this is that your netmask entry in the `~/helion/my_cloud/definition/data/servers.yml` file is incorrect.

To resolve this issue ensure that you have the correct subnet and netmask entries in the `~/helion/my_cloud/definition/data/servers.yml` for your Management network.

You can also verify these values exist in the `/etc/network/interfaces.d/ethX` file on your lifecycle manager. These should match the entries in the `servers.yml` file.

Once you make your corrections to your configuration files, commit the changes with these steps:

1. Ensuring that you stay within the `~/helion` directory, commit the changes you just made:

```
cd ~/helion
git commit -a -m "commit message"
```

2. Redeploy Cobbler:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

### Issue: Configuration changes needed after Cobbler deploy

If you've made a mistake or wish to change your `~/helion/my_cloud/definition/data/servers.yml` configuration file after you've already run the `cobbler-deploy.yml` playbook, follow these steps to ensure Cobbler gets updated with the new server information:

1. Ensure your `servers.yml` file is updated

2. Ensuring that you stay within the `~/helion` directory, commit the changes you just made:

```
cd ~/helion
git commit -a -m "commit message"
```

3. Determine which nodes you have entered into Cobbler by using:

```
sudo cobbler system list
```

4. Remove the nodes that had the old information from Cobbler using:

```
sudo cobbler system remove --name <nodename>
```

5. Re-run the `cobbler-deploy.yml` playbook to update the new node definitions to Cobbler:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

## Note

If you've also already run the `bm-reimage.yml` playbook then read the section below for how to ensure your nodes get re-imaged when re-running this playbook.

### **Issue: `bm-reimage.yml` playbook doesn't find any nodes to image**

You may receive this error when running the `bm-reimage.yml` playbook:

```
TASK: [cobbler | get-nodelist | Check we have targets] ****
failed: [localhost] => {"failed": true}
msg: There is no default set of nodes for this command, use -e nodelist

FATAL: all hosts have already failed -- aborting
```

This behavior occurs when you don't specify the `-e nodelist` switch to your command and all of your nodes are marked as `netboot-enabled: false` in Cobbler. By default, without the `-e nodelist` switch, the `bm-reimage.yml` playbook will only reimagine the nodes marked as `netboot-enabled: True`, which you can verify which nodes you have that are marked as such with this command:

```
sudo cobbler system find --netboot-enabled=1
```

If this is on a fresh install and none of your nodes have been imaged with the HPE Helion OpenStack ISO, then you should be able to remove all of your nodes from Cobbler and re-run the `cobbler-deploy.yml` playbook. You can do so with these commands:

1. Get a list of your nodes in Cobbler:

```
sudo cobbler system list
```

2. Remove each of them with this command:

```
sudo cobbler system remove --name SYSTEM_NAME
```

3. Confirm they are all removed in Cobbler:

```
sudo cobbler system list
```

4. Re-run `cobbler-deploy.yml`:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost cobbler-deploy.yml
```

5. Confirm they are all now set to `netboot-enabled: True`:

```
sudo cobbler system find --netboot-enabled=1
```

### **Issue: The `--limit` switch does not do anything**

If the `cobbler-deploy.yml` playbook fails, it makes a mention that to retry you should use the `--limit` switch, as seen below:

```
to retry, use: --limit @/home/heatadmin/cobbler-deploy.retry
```

This is a standard Ansible message but it is not needed in this context. Please use our instructions described above to remove your systems, if necessary, from Cobbler before re-running the playbook as the `--lim-`

it is not needed. Note that using the `--limit` switch does not cause any harm and won't prevent the playbook from completing, it's just not a necessary step.

### **Issue: Dealing With Nodes that Fail to Install**

The `bm-reimage.yml` playbook will take every node as far as it can through the baremetal install process. Nodes that fail will not prevent the others from continuing to completion. At the end of the run you will get a list of the nodes (if any) that failed to install.

If you run `bm-reimage.yml` a second time, by default it will target only the failed nodes the second time round (because the others are already marked as "completed"). Alternatively you can target specific nodes for reimage using `-e nodelist` as described in the section below.

The places where you are most likely to see a node fail is timeout in the "wait for shutdown" step, which means that the node did not successfully install an operating system (e.g. it could be stuck in POST) or a timeout in "wait for ssh" at the end of the baremetal install. This means that the node did not come back up after being powered on. To fix the issues you'll need to connect to the nodes' consoles and investigate.

### **How to re-image an existing node**

Once Linux for HPE Helion has been successfully installed on a node, that node will be marked as "installed" and subsequent runs of `bm-reimage.yml` (and related playbooks) will not target it. This is deliberate so that you can't reimage the node by accident. If you do need to reimage existing nodes you will need to use the `-e nodelist` option to target them specifically. For example:

```
ansible-playbook -i hosts/localhost bm-reimage.yml -e nodelist=cpn-0044,cpn-0045
```

### **Note**

You can target all nodes with `-e nodelist=all`

This will power cycle the specified nodes and reinstall the operating system on them, using the existing settings stored in Cobbler.

If you want to change settings for a node in the configuration files, see the section called "Issues while Provisioning your Baremetal Nodes".

### **Issue: Wait for SSH phase in bm-reimage.yml hangs**

This issue has been observed during deployment to systems configured with QLogic based BCM578XX network adapters utilizing the `bnx2x` driver and is currently under investigation. The symptom manifests following a cold boot during deployment at the "wait for SSH" phase in `bm-reimage.yml` which will result in a hang and eventually timeout, causing the baremetal install to fail. The presence of this particular issue can be further confirmed by connecting to the remote console of the server via iLO or by checking the server's `dmesg` output for the presence of `bnx2_panic_dump` messages, similar to the following:

```
bnx2x: [bnx2x_prev_unload_common:10433(eth%d)]Failed to empty BRB, hope for the be  
bnx2x: [bnx2x_stats_update:1268(eth0)]storm stats were not updated for 3 times  
bnx2x: [bnx2x_stats_update:1269(eth0)]driver assert  
bnx2x: [bnx2x_panic_dump:929(eth0)]begin crash dump ----- ..  
bnx2x: [bnx2x_panic_dump:1163(eth0)]end crash dump -----
```

This workaround is completed by rebooting the server.

### **Subsequent runs of bm-reimage.yml sometimes fail to trigger network install**

If you've run the `site.yml` playbook during an install and have a reason to run the `bm-reimage.yml` playbook afterwards, it may fail to trigger the network install over PXE due to the name of the interface changing during this process.

To resolve this issue, you can run the `cobbler-deploy.yml` playbook to refresh these settings and then proceed to run `bm-reimage.yml`.

#### **Issue: Soft lockup at imaging**

When imaging your nodes, if you see a kernel error of the type "BUG: soft lockup - CPU#10 stuck...." you should reset the node and make sure it is imaged properly next time. Note that depending on when you get the error, you may have to rerun `cobbler-deploy.yml`. If the `bm-reimage` playbook says it failed to image the node, then Cobbler knows this has occurred and you can reset the node. If not, you can follow the instructions above for .

#### **Blank Screen Seen When Monitoring the Imaging Step**

If you are watching the `os-install` process on a node via the console output, there can be a pause between 2-3 minutes in length where nothing gets reported to the console screen. This is just after the grub menu has been displayed on a UEFI-based system.

This is normal and nothing to be concerned with. The imaging process will continue on after this pause.

#### **Unable to SSH to a Compute Node after the bm-reimage Step Even Though SSH Keys Are In Place**

If after running the `bm-reimage.yml` playbook you get a reported failure when attempting to SSH to one or more Compute nodes stating that permission was denied and you've confirmed that the SSH keys are correctly copied from the lifecycle manager to the Compute node then you can follow the steps below to resolve this issue.

The suspected root cause of this issue is that the `bm-reimage` playbook is finding a different MAC address when attempting to SSH to the Compute node than was specified in the `servers.yml` file. This could be because you entered an incorrect MAC address or you may have specified the same IP address to two different Compute nodes.

To resolve this issue you can follow these steps:

1. Remove the Compute node that failed from Cobbler using this command:

```
sudo cobbler system remove --name <node name>
```

#### **Note**

Use `sudo cobbler system list` to get a list of your nodes in Cobbler.

2. Correct the IP and MAC address information in your `~/helion/my_cloud/definition/data/servers.yml` file.

3. Commit your changes to git:

```
cd ~/helion/hos/ansible  
git add -A  
git commit -m "My config or other commit message"
```

4. Re-run the `bm-reimage.yml` playbook and confirm the error is not received again.

#### **Server Crashes During Imaging**

If your server suffers a kernel panic just after it gets PXE booted, a possible cause could be that you have an incorrect bus address for the server specified in the `nic_mappings.yml` file.

To resolve this issue you will need to obtain the proper bus address. You can do this by installing HP Linux for OpenStack on the node and once you receive a terminal prompt you can use the command below to obtain the proper bus address:

```
sudo lspci -D | grep -i eth
```

Once you have the proper bus address, follow these steps to finish the reimage:

1. Remove the node that failed from Cobbler using this command:

```
sudo cobbler system remove --name <node name>
```

### Note

Use `sudo cobbler system list` to get a list of your nodes in Cobbler.

2. Correct the bus address in your `~/helion/my_cloud/definition/data/nic_mappings.yml` file.

3. Commit your changes to git:

```
cd ~/helion/hos/ansible  
git add -A  
git commit -m "My config or other commit message"
```

4. Re-run the `bm-reimage.yml` playbook and confirm the error is not received again.

## Issues while Updating Configuration Files

### Configuration Processor Fails Due to Wrong yml Format

If you receive the error below when running the configuration processor then you may have a formatting error:

```
TASK: [fail msg="Configuration processor run failed, see log output above for details"]
```

First you should check the ansible log in the location below for more details on which yml file in your input model has the error:

```
~/.ansible/ansible.log
```

Check the configuration file to locate and fix the error, keeping in mind the following tips below.

Check your files to ensure that they don't contain the following:

- Non-ascii characters
- Unneeded spaces

Once you have fixed the formatting error in your files, commit the changes with these steps:

1. Commit your changes to git:

```
cd ~/helion/hos/ansible
```

```
git add -A  
git commit -m "My config or other commit message"
```

2. Re-run the configuration processor playbook and confirm the error is not received again.

### Configuration processor fails with provider network OCTAVIA-MGMT-NET error

If you receive the error below when running the configuration processor then you have not correctly configured your VLAN settings for Octavia.

```
" #####  
" # The configuration processor failed. ",  
"# config-data-2.0      ERR: Provider network OCTAVIA-MGMT-NET host_routes:  
" #####
```

To resolve the issue, ensure that your settings in `~/helion/my_cloud/definition/data/neutron/neutron_config.yml` are correct for the VLAN setup for Octavia.

### Changes Made to your Configuration Files

If you have made corrections to your configuration files and need to re-run the Configuration Processor, the only thing you need to do is commit your changes to your local git:

```
cd ~/helion/hos/ansible  
git add -A  
git commit -m "commit message"
```

You can then re-run the configuration processor:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

### Configuration Processor Fails Because Encryption Key Does Not Meet Requirements

If you choose to set an encryption password when running the configuration processor, you may receive the following error if the chosen password does not meet the complexity requirements:

```
" #####  
# The configuration processor failed.  
# encryption-key ERR: The Encryption Key does not meet the following  
#           The Encryption Key must be at least 12 characters  
#           The Encryption Key must contain at least 3 of following classe  
" #####
```

If you receive the above error, simply run the configuration processor again and select a password that meets the complexity requirements detailed in the error message:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

## Issues while Deploying the Cloud

### Issue: If the site.yml playbook fails, you can query the log for the reason

Ansible is good about outputting the errors into the command line output, however if you'd like to view the full log for any reason the location is:

```
~/.ansible/ansible.log
```

This log is updated real time as you run ansible playbooks.

## Tip

Use grep to parse through the log. Usage: `grep <text> ~/.ansible/ansible.log`

### Issue: How to Wipe the Disks of your Machines

If you have re-run the `site.yml` playbook, you may need to wipe the disks of your nodes

You would generally run the playbook below after re-running the `bm-reimage.yml` playbook but before you re-run the `site.yml` playbook.

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts wipe_disks.yml
```

The playbook will show you the disks to be wiped in the output and allow you to confirm that you want to complete this action or abort it if you do not want to proceed. You can optionally use the `--limit <NODE_NAME>` switch on this playbook to restrict it to specific nodes.

If you receive an error stating that `osconfig` has already run on your nodes then you will need to remove the `/etc/hos/osconfig-ran` file on each of the nodes you want to wipe with this command:

```
sudo rm /etc/hos/osconfig-ran
```

That will clear this flag and allow the disk to be wiped.

### Issue: Errors during create\_db Task

If you are doing a fresh installation on top of a previously used system and you have not completely wiped your disk prior to the installation, you may run into issues when the installation attempts to create new Vertica databases. The recommendation here is to wipe your disks and re-start the installation, however we have included a playbook for cleaning your Vertica databases if they need to be re-used.

To run this playbook, follow these steps:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts monasca-vertica-dbclean.yml
```

### Issue: Freezer installation fails if an independent network is used for the External\_API.

Currently the Freezer installation fails if an independent network is used for the `External_API`. If you intend to deploy the External API on an independent network, the following changes need to be made:

In `roles/freezer-agent/defaults/main.yml` add the following line:

```
backup_freezer_api_url: "{{ FRE_API | item('advertises.vips.private[0].url', default='')) }}
```

In `roles/freezer-agent/templates/backup.osrc.j2` add the following line:

```
export OS_FREEZER_URL={{ backup_freezer_api_url }}
```

### Error Received if Root Logical Volume is Too Small

When running the `site.yml` playbook, you may receive the error below if your root logical-volume is too small:

```
2015-09-29 15:54:02,751 p=26345 u=stack | TASK: [osconfig | disk config | Extend  
2015-09-29 15:54:03,021 p=26345 u=stack | failed: [helion-ccp-swpac-m1-mgmt] => (  
'name': 'hlm-vg'}, {'mount': '/', 'fstype': 'ext4', 'name': 'root', 'size': '10%'}  
vg/root"], "delta": "0:00:00.022983", "end": "2015-09-29 10:54:18.925855", "failed":  
"os"}, "name": "hlm-vg", "physical_volumes": [{"/dev/sda_root"}], {"fstype": "ext4"}  
09-29 10:54:18.902872", "stdout_lines": [], "warnings": []}  
2015-09-29 15:54:03,022 p=26345 u=stack | stderr: New size given (7128 extents)
```

The specific part of this error to parse out and resolve is:

```
stderr: New size given (7128 extents) not larger than existing size (7629 extent
```

The error also references the root volume:

```
"name": "root", "size": "10%"
```

The problem is that the root logical-volume, as specified in the `disks_controller.yml` file, is set to 10% of the overall physical volume and this value is too small.

To resolve this issue you need to ensure that the percentage is set properly for the size of your logical-volume. The default values in the configuration files is based on a 500GB disk, so if your logical-volumes are smaller you may need to increase the percentage so there is enough room.

### **Multiple Keystone Failures Received during Site.yml**

If you receive the Keystone error below during your `site.yml` run then follow these steps:

```
TASK: [OPS-MON | _keystone_conf | Create Ops Console service in Keystone] *****  
failed: [helion-cpl-c1-m1-mgmt] => {"failed": true}  
msg: An unexpected error prevented the server from fulfilling your request. (HTTP  
  
FATAL: all hosts have already failed -- aborting
```

The most likely cause of this error is that the virtual IP address is having issues and the Keystone API communication through the virtual IP address is not working properly. You will want to check the Keystone log on the controller where you will likely see authorization failure errors.

Verify that your virtual IP address is active and listening on the proper port on all of your controllers using this command:

```
netstat -tplan | grep 35357
```

Ensure that your lifecycle manager did not pick the wrong (unusable) IP address from the list of IP addresses assigned to your Management network.

The lifecycle manager will take the first available IP address after the `gateway-ip` defined in your `~/helion/my_cloud/definition/data/networks.yml` file. This IP will be used as the virtual IP address for that particular network. If this IP address is used and reserved for another purpose outside of your HPE Helion OpenStack deployment then you will receive the error above.

To resolve this issue we recommend that you utilize the `start-address` and possibly the `end-address` (if needed) options in your `networks.yml` file to further define which IP addresses you want your cloud deployment to use. See for more details.

After you have made changes to your `networks.yml` file, follow these steps to commit the changes:

1. Ensuring that you stay within the `~/helion` directory, commit the changes you just made:

```
cd ~/helion  
git commit -a -m "commit message"
```

2. Run the configuration processor:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

3. Update your deployment directory:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

4. Re-run the site.yml playbook:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts site.yml
```

### VSA installer failed during deployment

VSA deployment might fail sporadically due to a defect in the HPE StoreVirtual VSA installation files for the KVM host. The following error messages are displayed and VSA installation will hang for more than 45-60 minutes:

```
14:10:13 TASK: [VSA-DEP | deploy | Create VSA appliance] ****  
14:18:32 changed: [hlm003-cpl1-vsa0003-mgmt]  
15:32:36 fatal: [hlm003-cpl1-vsa0002-mgmt] => SSH Error: Shared connection to 10.24  
15:32:36 It is sometimes useful to re-run the command using -vvvv, which prints SS  
15:32:36 fatal: [hlm003-cpl1-vsa0001-mgmt] => SSH Error: Shared connection to 10.24  
15:32:36 It is sometimes useful to re-run the command using -vvvv, which prints SS  
15:32:37
```

### Note

VSA deployment can take approximately usually 15-30 minutes, depending on how many disks are configured. Do not terminate the installation until 60 minutes have passed and/or you see the error.

To correct this issue:

1. Log into the failed VSA node.

2. Change to the root user:

```
sudo -i
```

3. Destroy the VSA VM:

```
virsh destroy <VSA_VM_Name>  
virsh undefine <VSA_VM_Name>
```

For example:

```
virsh destroy VSA-VM-vsa2  
virsh undefine VSA-VM-vsa2
```

To get the name of the VSA VM, execute the following command:

```
virsh list --all
Id      Name               State
-----
2       VSA-VM-vsa2        running
```

4. Destroy the VSA network:

```
virsh net-destroy vsa-network
virsh net-undefine vsa-network
```

5. Destroy the VSA storage-pool:

```
virsh pool-destroy vsa-storage-pool
virsh pool-undefine vsa-storage-pool
```

6. Remove the vsa-installer directory:

```
rm -rf /home/vsa-installer
```

7. Remove the VSA image:

```
rm -rf /mnt/state/vsa-kvm-storage
```

8. Reboot the VSA VM:

```
reboot
```

9. Log into your lifecycle manager.

10. Run the deployment playbooks from the resulting scratch directory:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts site.yml
```

---

# Chapter 19. Troubleshooting the Block Storage Backend Configuration

We have gathered some of the common issues that occur during the Block Storage configuration steps and organized them by product.

## **Issue: You have forgotten the identifying information about your VSA cluster**

If you have forgotten the cluster name or IP address after creating the cluster in the CMC utility then you can perform the following steps to retrieve them:

1. Log in to CMC console using the `/opt/HP/StoreVirtual/UI/jre/bin/java -jar /opt/HP/StoreVirtual/UI/UI.jar` command from the lifecycle manager/deployer node.
2. Log in to the Management group.
3. On the left side of the CMC console screen, click **mgmt**. It expands and you will be able to see the cluster name.
4. Click the cluster name. The page on the right hand side will populate with the cluster information.
5. Click **iSCSI** on the right hand side of the page. The virtual IP address will be displayed in the tabular form. You can view your cluster ID here.

## **Issue: Error During VSA Deployment Exhibits When Running site.yml or hlm-deploy.yml**

When you are running either `site.yml` or `hlm-deploy.yml` and you get either of the errors below during the VSA deployment task, there is an error in your locale settings.

Errors:

```
TASK: [VSA-DEP | deploy | Create VSA appliance] *****
failed: [helion-cp1-vsa0001-mgmt] => {"changed": true, "cmd": ["./vsas_automation"], "stderr": "libvirt: Network Driver error : Network not found: no network with matching name\nTraceback (most recent call last):\n  File \"./vsas_automation\", line 83, in <module>\n    deployer.install_vsa()\n  File \"/opt/stack/venv/storevirtual-installer-20151022T080854Z/lib/python2.7/site\n    self._roll_back_installation()\n  File \"/opt/stack/venv/storevirtual-installer-20151022T080854Z/lib/python2.7/site\n    self._vsas_network_destroy()\n  File \"/opt/stack/venv/storevirtual-installer-20151022T080854Z/lib/python2.7/site\n    raise vsas_excs.NetworkDestroyFailed(msg)\nstorevirtual_installer.vsa_excs.NetworkDestroyFailed: Failed to destroy and undefine\n\nFATAL: all hosts have already failed -- aborting
```

OR

```
Unable to get XML definition of storage pool -----
error: failed to get pool '-----'
error: Storage pool not found: no storage pool with matching name '-----'
```

Fix:

Make sure that the locale variables on the lifecycle manager are valid and then run the playbook again. You can set this variable to set your locale:

```
export LC_ALL=C
```

---

# Chapter 20. Troubleshooting the ESX

This section contains troubleshooting tasks for your HPE Helion OpenStack® 5.0 for ESX.

## Issue: If `hlm-ux-services.service` is not running, the `eon-resource-activate` and `eon-resource-deactivate` commands fails

If you perform any maintenance work or reboot the lifecycle manager/deployer node, make sure to restart the HLM UX service for standalone deployer node and shared HLM/controller node based on your environment.

For standalone deployer node, execute `hlm-start.yml` playbook to restart the HLM UX services on the deployer node after a reboot.

For shared deployer/controller node, execute `hlm-start.yml` playbook on all the controllers to restart HLM UX services.

For example:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts hlm-start.yml --limit <host name of the HLM n
```

## Issue: ESX onboard Compute

1. If there is a failure in provisioning phase
  - a. The spawned appliances will be deleted.
  - b. Check for `/var/log/eon/eon-conductor.log` for specific errors to troubleshoot the error.
2. If there is a failures in activating phase
  - a. The spawned appliances will be left intact, and you can try executing activate command again.
  - b. Check the `~/.ansible/ansible.log` or `/var/log/hlm-ux-service/service.log` in the `hlm-deployer`.

In case of SSH failure, check the background connectivity and try to ping again.

3. Failures leading to inconsistent Database (DB) state, execute the following command:

```
eon resource-deactivate <id> --forced True
```

## Issue: Migrate eon-conductor

Currently `eon-conductor` service is running in the first node of the control plane.

### Procedure to migrate `eon-conductor`

Perform the following steps to migrate `eon-conductor` service to any other node in the control plane.

1. SSH to lifecycle manager.

2. Change the directory.

```
cd /home/user/scratch/ansible/next/hos/ansible
```

3. Search where the eon conductor is running currently. In the following example, eon-conductor is running in helion-cpl-c1-m1-mgmt.

```
(helion-cpl-c1-m1-mgmt)
stack@helion-cpl-c1-m1-mgmt:~$ ps aux | grep eon
eon      11236  5.5  0.0 142104 52088 ?          Ss   18:53  0:01 /opt/stack/venv
eon      11281 23.2  0.1 401964 238080 ?          Ss   18:53  0:06 /opt/stack/venv
eon      11590  1.5  0.0 144420 49084 ?          S    18:53  0:00 /opt/stack/venv
eon      11591  1.4  0.0 144416 49056 ?          S    18:53  0:00 /opt/stack/venv
eon      11592  1.1  0.0 144420 48828 ?          S    18:53  0:00 /opt/stack/venv
stack    12086  0.0  0.0 12736  2064 pts/9    S+   18:54  0:00 grep eon

(helion-cpl-c1-m2-mgmt)
stack@helion-cpl-c1-m2-mgmt:~$ ps aux | grep eon
eon      7050  1.9  0.0 142100 52052 ?          Ss   18:53  0:03 /opt/stack/venv
eon      7677  1.1  0.0 144552 49156 ?          S    18:53  0:02 /opt/stack/venv
eon      7678  1.2  0.0 144544 49256 ?          S    18:53  0:02 /opt/stack/venv
eon      7679  1.2  0.0 144548 49136 ?          S    18:53  0:02 /opt/stack/venv
stack    13523  0.0  0.0 12732  2076 pts/0    S+   18:56  0:00 grep eon
```

4. Stop the eon service.

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts eon-stop.yml
```

Example:

```
stack@helion-cpl-c1-m1-mgmt:~/scratch/ansible/next/hos/ansible$ ansible-playbook
PLAY [EON-API] ****
TASK: [eon-common | stop | stop eon-api service] ****
changed: [helion-cpl-c1-m3-mgmt]
changed: [helion-cpl-c1-m1-mgmt]
changed: [helion-cpl-c1-m2-mgmt]

PLAY [EON-CND--first-member] ****
TASK: [eon-common | stop | stop eon-conductor service] ****
changed: [helion-cpl-c1-m1-mgmt]

PLAY [EON-ONEVIEW--first-member] ****
skipping: no hosts matched

PLAY RECAP ****
eon-common | stop | stop eon-conductor service ----- 8.39s
eon-common | stop | stop eon-api service ----- 0.42s
-----
Total: ----- 8.99s
helion-cpl-c1-m1-mgmt      : ok=2    changed=2    unreachable=0    failed=0
helion-cpl-c1-m2-mgmt      : ok=1    changed=1    unreachable=0    failed=0
```

```
helion-cpl-c1-m3-mgmt : ok=1     changed=1     unreachable=0     failed=0
```

5. Edit the `/home/user/scratch/ansible/next/hos/ansible/hosts/verb_hosts`. In the following example, we change the node name from `helion-cpl-c1-m1-mgmt` to `helion-cpl-c1-m2-mgmt` in the `[EON-CND--first-member:children]` group to move the eon-conductor service to node `helion-cpl-c1-m2-mgmt`.

```
[EON-CND--first-member:children] (ORIGINAL)
helion-cpl-c1-m1-mgmt <- original node that eon-conductor is setup on.
```

```
[EON-CND--first-member:children] (MODIFIED)
helion-cpl-c1-m2-mgmt <- new node that eon-conductor is setup on.
```

6. Execute the following playbook in sequence.

```
ansible-playbook -i hosts/verb_hosts eon-deploy.yml
ansible-playbook -i hosts/verb_hosts eon-start.yml
```

Example:

```
TASK: [eon-conductor | start | start eon-conductor service] ****
changed: [helion-cpl-c1-m2-mgmt]

PLAY RECAP ****
eon-api | start | start eon-api service ----- 0.26s
eon-conductor | start | start eon-conductor service ----- 0.23s
eon-api | start | activate the latest installed version ----- 0.07s
eon-api | start | restart eon-api service ----- 0.07s
eon-conductor | start | restart eon-conductor service ----- 0.02s
eon-conductor | start | activate the latest installed version ----- 0.02s
-----
Total: ----- 1.09s
helion-cpl-c1-m1-mgmt : ok=1     changed=0     unreachable=0     failed=0
helion-cpl-c1-m2-mgmt : ok=2     changed=0     unreachable=0     failed=0 <- S
helion-cpl-c1-m3-mgmt : ok=1     changed=0     unreachable=0     failed=0
```

7. Ensure that the conductor is running on the node that you have migrated. In the following example observe that the eon-conductor have migrated to node `helion-cpl-c1-m2-mgmt`.

```
(helion-cpl-c1-m1-mgmt)
stack@helion-cpl-c1-m1-mgmt:~$ ps aux | grep eon
stack    2624  0.0  0.0 12732  2132 pts/9    S+   18:46   0:00 grep eon
eon      27786  1.3  0.0 142112 52064 ?          Ss   16:21   1:55 /opt/stack/venv
eon      27816  1.2  0.0 149712 55060 ?          S    16:21   1:50 /opt/stack/venv
eon      27817  1.2  0.0 150380 55724 ?          S    16:21   1:49 /opt/stack/venv
eon      27818  1.2  0.0 148516 53844 ?          S    16:21   1:49 /opt/stack/venv
```

```
(helion-cpl-c1-m2-mgmt)
stack@helion-cpl-c1-m2-mgmt:~$ ps aux | grep eon
eon      1567  1.3  0.0 142108 52076 ?          Ss   16:21   1:56 /opt/stack/venv
eon      1606  1.2  0.0 144816 49888 ?          S    16:21   1:51 /opt/stack/venv
eon      1607  1.2  0.0 145068 49888 ?          S    16:21   1:50 /opt/stack/venv
eon      1608  1.2  0.0 145064 49888 ?          S    16:21   1:49 /opt/stack/venv
eon      22720 0.0  0.1 404872 241220 ?          Ss   16:15   0:08 /opt/stack/venv
stack   31965 0.0  0.0 12732  2308 pts/0    S+   18:50   0:00 grep eon
```

# Issue: ESX Cluster shows UNKNOWN in OpsConsole

In the OpsConsole Alarms dashboard, if all the alarms for ESX cluster are showing UNKNOWN then restart the monasca-agent running in ESX compute proxy.

1. SSH to the respective compute proxy. You can find the hostname of the proxy from the dimensions list shown against the respective alarm.
2. Restart the monasca-agent service.

```
sudo systemctl restart monasca-agent
```

# Issue: Unable to view the VM console in Horizon UI

By default the gdbserver firewall is disabled in ESXi host which results in a Handshake error when accessing the VM instance console in the Horizon UI.



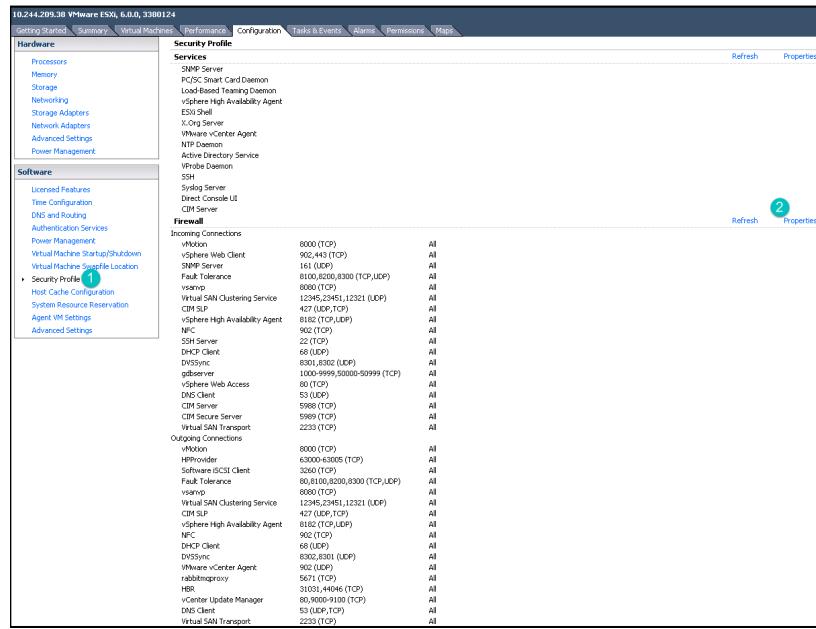
## Procedure to enable gdbserver

1. Login to vSphere Client.
2. Select the ESXi Host and click **Configuration** tab in the menu bar. You must perform the following actions on all the ESXi hosts in the compute clusters.



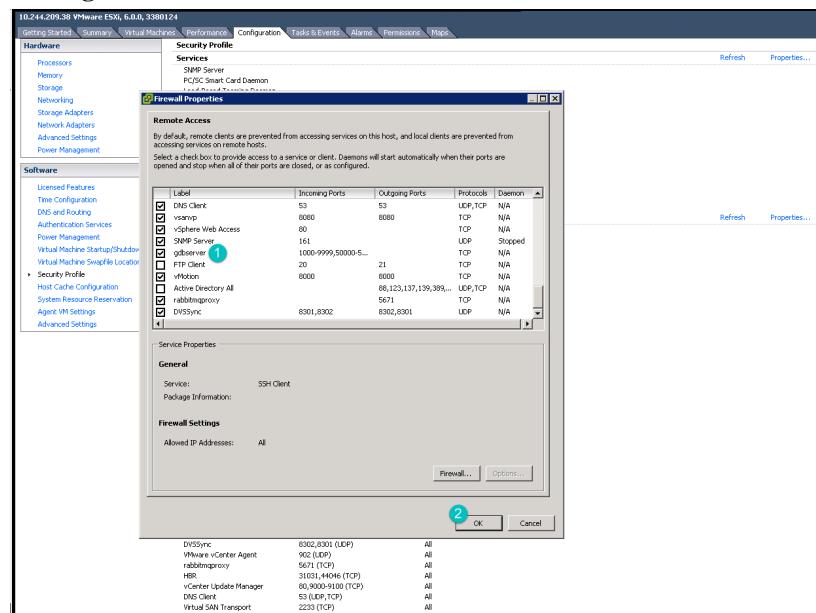
3. On the left hand side select **Security Profile** from the list of **Software**. Click **Properties** on the right hand side.

## Troubleshooting the ESX



Firewall Properties box displays.

#### 4. Select **gdbserver** checkbox and click **OK**.



---

## **Part III. Post-Installation**

---

# Table of Contents

|                                                                      |     |
|----------------------------------------------------------------------|-----|
| 21. Overview .....                                                   | 278 |
| 22. Cloud Verification .....                                         | 279 |
| API Verification .....                                               | 279 |
| Prerequisites .....                                                  | 279 |
| Tempest Integration Tests .....                                      | 279 |
| Running the Tests .....                                              | 280 |
| Viewing Test Results .....                                           | 280 |
| Customizing the Test Run .....                                       | 281 |
| Run Tests for Specific Services and Exclude Specific Features .....  | 281 |
| Run Tests Matching a Series of White and Blacklists .....            | 281 |
| 23. UI Verification .....                                            | 283 |
| Verifying Your Block Storage Backend .....                           | 283 |
| Create a Volume .....                                                | 283 |
| Attach Volume to an Instance .....                                   | 284 |
| Detach Volume from Instance .....                                    | 284 |
| Delete Volume .....                                                  | 285 |
| Verifying Your Object Storage (Swift) .....                          | 285 |
| Verify the Object Storage (Swift) Operations .....                   | 286 |
| Uploading an Image for Use .....                                     | 287 |
| Running the Playbook .....                                           | 287 |
| How to Curate Your Own Images .....                                  | 287 |
| Using the GlanceClient CLI to Create Images .....                    | 287 |
| Creating an External Network .....                                   | 287 |
| Notes .....                                                          | 288 |
| Using the Ansible Playbook .....                                     | 288 |
| Using the NeutronClient CLI .....                                    | 288 |
| Next Steps .....                                                     | 289 |
| 24. Installing OpenStack Clients .....                               | 290 |
| 25. Configuring Transport Layer Security (TLS) .....                 | 293 |
| Configuring TLS in the input model .....                             | 294 |
| User-provided certificates and trust chains .....                    | 295 |
| Edit the input model to include your certificate files .....         | 296 |
| Generate a self-signed CA .....                                      | 297 |
| Generate a certificate signing request .....                         | 299 |
| Generate a server certificate .....                                  | 299 |
| Upload to the lifecycle manager .....                                | 302 |
| Configuring the cipher suite .....                                   | 302 |
| Testing .....                                                        | 303 |
| Verifying that the trust chain is correctly deployed .....           | 303 |
| Turning TLS on or off .....                                          | 303 |
| 26. Configuring Availability Zones .....                             | 305 |
| 27. Configuring Load Balancer as a Service .....                     | 306 |
| Prerequisites .....                                                  | 307 |
| Octavia Load Balancing Provider .....                                | 307 |
| Setup of prerequisites .....                                         | 307 |
| Create Load Balancers .....                                          | 317 |
| Create Floating IPs for Load Balancer .....                          | 320 |
| Testing the Octavia Load Balancer .....                              | 321 |
| 28. Other Common Post-Installation Tasks .....                       | 323 |
| Determining Your User Credentials .....                              | 323 |
| Configure your lifecycle manager to use the command-line tools ..... | 323 |

|                                            |     |
|--------------------------------------------|-----|
| Protect home directory .....               | 323 |
| Back up Your SSH Keys .....                | 324 |
| Retrieving Service Endpoints .....         | 324 |
| Other Common Post-Installation Tasks ..... | 324 |

---

# **Chapter 21. Overview**

Once you have completed your cloud deployment, these are some of the common post-installation tasks you may need to perform. Take a look at the descriptions below to determine which of these you need to do.

---

# Chapter 22. Cloud Verification

Once you have completed your cloud deployment, these are some of the common post-installation tasks you may need to perform to verify your cloud installation.

## API Verification

HPE Helion OpenStack 5.0 provides a tool, Tempest, that you can use to verify that your cloud deployment completed successfully:

- the section called “Prerequisites”
- the section called “Tempest Integration Tests”
  - the section called “Running the Tests”
  - the section called “Viewing Test Results”
  - the section called “Customizing the Test Run”
- the section called “Verifying Your Block Storage Backend”
- the section called “Verify the Object Storage (Swift) Operations”

## Prerequisites

The verification tests rely on you having an external network setup and a cloud image in your image (Glance) repository. Run the following playbook to configure your cloud:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts hlm-cloud-configure.yml
```

### Note

In HPE Helion OpenStack 5.0, the EXT\_NET\_CIDR setting for the external network is now specified in the input model - see the section called “neutron-external-networks”.

## Tempest Integration Tests

Tempest is a set of integration tests for OpenStack API validation, scenarios, and other specific tests to be run against a live OpenStack cluster. In HPE Helion OpenStack 5.0, Tempest has been modeled as a service and this gives you the ability to locate Tempest anywhere in the cloud. It is recommended that you install Tempest on your lifecycle manager node - that is where it resides by default in a new installation.

A version of the upstream Tempest [<http://docs.openstack.org/developer/tempest/>] integration tests is pre-deployed on the HPE Helion OpenStack lifecycle manager node. For details on what Tempest is testing, you can check the contents of this file on your lifecycle manager:

```
/opt/stack/tempest/run_filters/ci.txt
```

You can use these embedded tests to verify if the deployed cloud is functional.

For more information on running Tempest tests, see Tempest - The OpenStack Integration Test Suite [<https://git.openstack.org/cgit/openstack/tempest/tree/README.rst>].

## Important

Running these tests requires access to the deployed cloud's identity admin credentials

Tempest creates and deletes test accounts and test resources for test purposes.

In certain cases Tempest might fail to clean-up some of test resources after a test is complete, for example in case of failed tests.

## Running the Tests

To run the default set of Tempest tests:

1. Log in to the lifecycle manager.
2. Ensure you can access your cloud:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts cloud-client-setup.yml
source /etc/environment
```

3. Run the tests:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts tempest-run.yml
```

Optionally, you can the section called "Customizing the Test Run".

## Viewing Test Results

Tempest is deployed under `/opt/stack/tempest`. Test results are written in a log file in the following directory:

`/opt/stack/tempest/logs`

A detailed log file is written to:

`/opt/stack/tempest/tempest.log`

The results are also stored in the `testrepository` database.

To access the results after the run:

1. Log in to the lifecycle manager.
2. Change to the `tempest` directory and list the test results:

```
cd /opt/stack/tempest
./venv/bin/testr last
```

## Important

If you encounter an error saying "local variable 'run\_subunit\_content' referenced before assignment", you may need to log in as the `tempest` user to run this command. This is due to a known issue reported at <https://bugs.launchpad.net/testrepository/+bug/1348970>.

See Test Repository Users Manual [<https://testrepository.readthedocs.org/en/latest/>] for more details on how to manage the test result repository.

## Customizing the Test Run

There are several ways available to customize which tests will be executed.

- 
- 

## Run Tests for Specific Services and Exclude Specific Features

Tempest allows you to test specific services and features using the `tempest.conf` configuration file.

A working configuration file with inline documentation is deployed under `/opt/stack/tempest/etc/`.

To use this, follow these steps:

1. Log in to the lifecycle manager.
2. Edit the `/opt/stack/tempest/configs/tempest_region1.conf` file.
3. To test specific service, edit the `[service_available]` section and clear the comment character `#` and set a line to `true` to test that service or `false` to not test that service.

```
cinder = true  
neutron = false
```

4. To test specific features, edit any of the `*_feature_enabled` sections to enable or disable tests on specific features of a service.

```
[volume-feature-enabled]  
[compute-feature-enabled]  
[identity-feature-enabled]  
[image-feature-enabled]  
[network-feature-enabled]  
[object-storage-feature-enabled]  
  
#Is the v2 identity API enabled (boolean value)  
api_v2 = true  
#Is the v3 identity API enabled (boolean value)  
api_v3 = false
```

5. Then run tests normally

## Run Tests Matching a Series of White and Blacklists

You can run tests against specific scenarios by editing or creating a run filter file.

Run filter files are deployed under `/opt/stack/tempest/run_filters`.

Use run filters to whitelist or blacklist specific tests or groups of tests:

- lines starting with # or empty are ignored
- lines starting with + are whitelisted
- lines starting with - are blacklisted
- lines not matching any of the above conditions are blacklisted

If whitelist is empty, all available tests are fed to blacklist. If blacklist is empty, all tests from whitelist are returned.

Whitelist is applied first. The blacklist is executed against the set of tests returned by the whitelist.

To run whitelist and blacklist tests:

1. Log in to the lifecycle manager.
2. Make sure you can access the cloud:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts cloud-client-setup.yml
source /etc/environment
```

3. Run the tests:

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts tempest-run.yml -e run_filter <run_filter_
```

Note that the run\_filter\_name is the name of the run\_filter file except for the extension. For instance, to run using the filter from the file /opt/stack/tempest/run\_filters/ci.txt, use the following:

```
ansible-playbook -i hosts/verb_hosts tempest-run.yml -e run_filter=ci
```

Documentation on the format of white and black-lists is available at:

```
/opt/stack/tempest/tests2skip.py
```

Example:

The following entries run API tests, exclude tests that are less relevant for deployment validation, such as negative, admin, cli and thirdparty (EC2) tests:

```
+tempest\api\.*
*[Aa]dmin.*/
*[Nn]egative.*/
- tempest\cli.*/
- tempest\thirdparty\.*
```

# Chapter 23. UI Verification

Once you have completed your cloud deployment, these are some of the common post-installation tasks you may need to perform to verify your cloud installation.

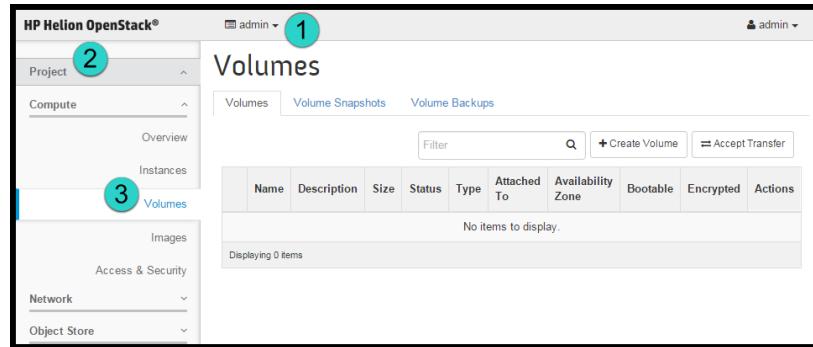
## Verifying Your Block Storage Backend

The sections below will show you the steps to verify that your Block Storage backend was setup properly.

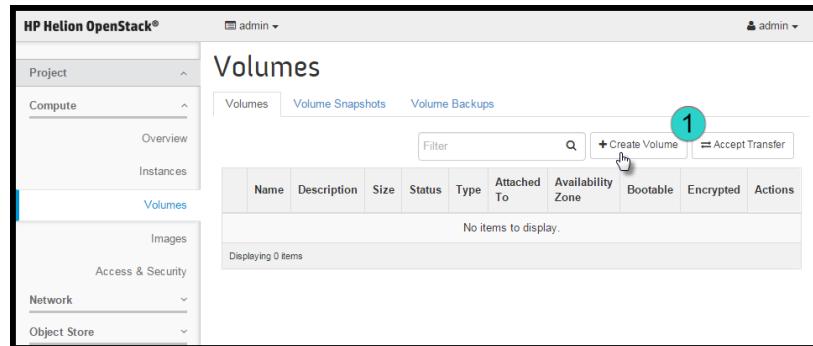
### Create a Volume

Perform the following steps to create a volume using Horizon dashboard.

1. Log into the Horizon dashboard. See for details.
2. Under the **Project** menu in the navigation pane, click on **Volumes** under the **Compute** subheading.

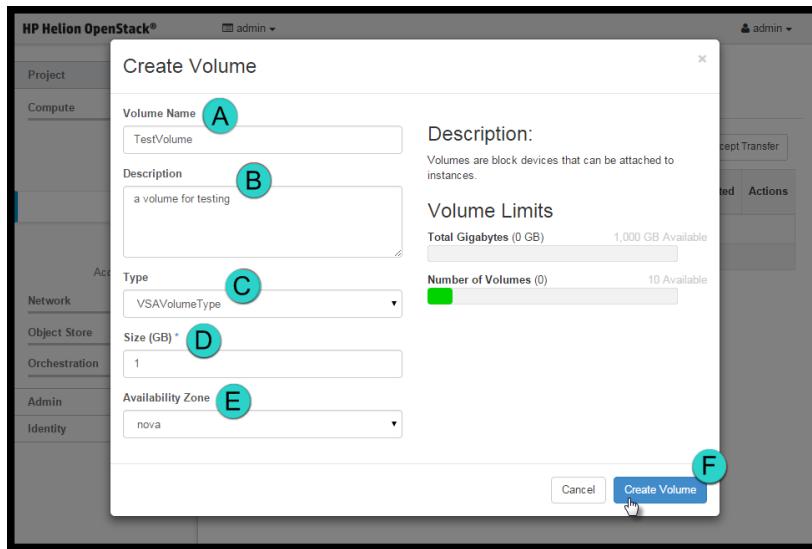


3. On the **Volumes** tabs, click the **Create Volume** button to create a volume.



4. In the **Create Volume** options, enter the required details into the fields and then click the **Create Volume** button:
  - a. Volume Name - This is the name you specify for your volume.
  - b. Description (optional) - This is an optional description for the volume.
  - c. Type - Select the volume type you have created for your volumes from the drop down.
  - d. Size (GB) - Enter the size, in GB, you would like the volume to be.

- e. Availability Zone - You can either leave this at the default option of **Any Availability Zone** or select a specific zone from the drop down.



The dashboard will then show the volume you have just created.

## Attach Volume to an Instance

Perform the following steps to attach a volume to an instance:

1. Log into the Horizon dashboard. See for details.
2. Under the **Project** menu in the navigation pane, click the **Instances** under the **Compute** subheading.
3. In the **Action** column, choose the **Edit Attachments** in the menu drop down box next to the instance you want to attach the volume to.
4. In the **Attach To Instance** drop-down, select the volume that you want to attach.
5. Edit the **Device Name** if necessary.
6. Click **Attach Volume** to complete the action.
7. Verify that the volume you attached is displayed in the **Attached To** columns on the **Volumes** screen.

## Detach Volume from Instance

Perform the following steps to detach the volume from instance:

1. Log into the Horizon dashboard. See for details.
2. Under the **Project** menu in the navigation pane, click the **Instances** under the **Compute** subheading.
3. Click the check box next to the name of the volume you want to detach.
4. In the **Action** column, choose the **Edit Attachments** in the menu drop down box next to the instance you want to attach the volume to.

5. Click **Detach Attachment**. A confirmation dialog box appears.
6. Click **Detach Attachment** to confirm the detachment of the volume from the associated instance.

## Delete Volume

Perform the following steps to delete a volume using Horizon dashboard:

1. Log into the Horizon dashboard. See for details.
2. Under the **Project** menu in the navigation pane, click on **Volumes** under the **Compute** subheading.

3. In the **Actions** column, click **Delete Volume** next to the volume you would like to delete.

4. To confirm and delete the volume, click **Delete Volume** again.

5. Verify that the volume was removed from the **Volumes** screen.

## Verifying Your Object Storage (Swift)

### Validate That All Servers Have Been Added to the Swift Rings

1. Run the swift-compare-model-rings.yml playbook as follows:

```
cd ~/scratch/ansible/next/hosts/ansible
```

```
ansible-playbook -i hosts/verb_hosts swift-compare-model-rings.yml
```

2. Search for output similar to the following. Specifically, look at the number of drives that are proposed to be added.

```
TASK: [swiftlm-ring-supervisor | validate-input-model | Print report] *****
ok: [helion-cp1-c1-m1-mgmt] => {
    "var": {
        "report.stdout_lines": [
            "Rings:",
            "  ACCOUNT:",
            "    ring exists",
            "    no device changes",
            "    ring will be rebalanced",
            "  CONTAINER:",
            "    ring exists",
            "    no device changes",
            "    ring will be rebalanced",
            "  OBJECT-0:",
            "    ring exists",
            "    no device changes",
            "    ring will be rebalanced"
        ]
    }
}
```

3. If the text contains "no device changes" then the deploy was successful and no further action is needed.
4. If there are more drives need to be added, it indicates that the deploy failed on some nodes and that you restarted the deploy to include those nodes. However, the nodes are not in the Swift rings because enough time has not elapsed to allow the rings to be rebuilt. You have two options to continue:
  - a. Repeat the deploy. There are two steps involved as follows:
    - i. Delete the ring builder files as described in .
    - ii. Repeat the installation process starting by running the site.yml playbook as described in the section of the installation instructions.
  - b. Rebalance the rings several times until all drives are incorporated in the rings. This process may take several hours to complete (because you need to wait one hour between each rebalance). The steps are as follows:
    - i. Change the min-part-hours to 1 hour. See .
    - ii. Use the "First phase of ring rebalance" and "Final rebalance phase" as described in . The "Weight change phase of ring rebalance" does not apply because you have not set the weight-step attribute at this stage.
    - iii. Set the min-part-hours to the recommended 16 hours as described in .

If you receive errors during the validation, read the documentation.

## Verify the Object Storage (Swift) Operations

Full details on how to verify the operations can be found in .

# Uploading an Image for Use

To create a Compute instance, you need to obtain an image that you can use. The HPE Helion OpenStack lifecycle manager provides an Ansible playbook that will download a CirrOS Linux image, and then upload it as a public image to your image repository for use across your projects.

## Running the Playbook

Use the following command to run this playbook:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts glance-cloud-configure.yml -e proxy=<PROXY>
```

The table below shows the optional switch that you can use as part of this playbook to specify environment-specific information:

| Switch                          | Description                                                                                                        |
|---------------------------------|--------------------------------------------------------------------------------------------------------------------|
| -e proxy="<proxy_address:port>" | Optional. If your environment requires a proxy for the internet, use this switch to specify the proxy information. |

## How to Curate Your Own Images

OpenStack has created a guide to show you how to obtain, create, and modify images that will be compatible with your cloud:

OpenStack Virtual Machine Image Guide [<http://docs.openstack.org/image-guide/content/>]

## Using the GlanceClient CLI to Create Images

You can use the GlanceClient on a machine accessible to your cloud or it's also installed automatically on your lifecycle manager.

The GlanceClient allows you to create, update, list, and delete images as well as manage your image member lists, which allows you to share access to images across multiple tenants. As with most of the OpenStack CLI tools, you can use the `glance help` command to get a full list of commands as well as their syntax.

If you would like to use the `--copy-from` option when creating an image, you will need to have your Administrator enable the http store in your environment using the instructions outlined at .

## Creating an External Network

You must have an external network set up to allow your Compute instances to reach the internet. There are multiple methods you can use to create this external network and we provide two of them here. The HPE Helion OpenStack installer provides an Ansible playbook that will create this network for use across your projects. We also show you how to create this network via the command line tool from your lifecycle manager.

- the section called “Using the Ansible Playbook”
- the section called “Using the NeutronClient CLI”

## Notes

If you have multiple regions in your cloud environment, it is important that when you set up your external networks in each region that you allocate unique IP ranges for each. If you have overlapping CIDR's then you risk having the same floating IP being allocated to two different virtual machines which will cause a conflict.

## Using the Ansible Playbook

This playbook will query the Networking service for an existing external network, and then create a new one if you do not already have one. The resulting external network will have the name `ext-net` with a subnet matching the CIDR you specify in the command below.

If you need to specify more granularity, for example specifying an allocation pool for the subnet then you should utilize the the section called “Using the NeutronClient CLI”.

```
cd ~/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts neutron-cloud-configure.yml -e EXT_NET_CIDR=<
```

The table below shows the optional switch that you can use as part of this playbook to specify environment-specific information:

| Switch                                    | Description                                                                                                                                                                                                                                                                                                                                     |
|-------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>-e EXT_NET_CIDR=&lt;CIDR&gt;</code> | <p>Optional. You can use this switch to specify the external network CIDR. If you choose not to use this switch, or use a wrong value, the VMs will not be accessible over the network.</p> <p>This CIDR will be from the EXTERNAL VM network.</p> <p><b>Note</b></p> <p>If this option is not defined the default value is "172.31.0.0/16"</p> |

## Using the NeutronClient CLI

For more granularity you can utilize the Neutron command line tool to create your external network.

1. Log in to the lifecycle manager.
2. Source the Admin credentials:

```
source ~/service.osrc
```

3. Create the external network and then the subnet using these commands below.

Creating the network:

```
neutron net-create --router:external <external-network-name>
```

Creating the subnet:

```
neutron subnet-create <external-network-name> <CIDR> --gateway <gateway> --alloc
```

Where:

| Value                       | Description                                                                                                                                                                                                                       |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| external-network-name       | This is the name given to your external network. This is a unique value that you will choose. The value <code>ext-net</code> is usually used.                                                                                     |
| CIDR                        | You can use this switch to specify the external network CIDR. If you choose not to use this switch, or use a wrong value, the VMs will not be accessible over the network.<br><br>This CIDR will be from the EXTERNAL VM network. |
| --gateway                   | Optional switch to specify the gateway IP for your subnet. If this is not included then it will choose the first available IP.                                                                                                    |
| --allocation-pool start end | Optional switch to specify a start and end IP address to use as the allocation pool for this subnet.                                                                                                                              |
| --disable-dhcp              | Optional switch if you want to disable DHCP on this subnet. If this is not specified then DHCP will be enabled.                                                                                                                   |

## Next Steps

Once the external network is created, users can create a Private Network to complete their networking setup. For instructions, see .

---

# Chapter 24. Installing OpenStack Clients

The way OpenStack clients are installed changed in HPE Helion OpenStack 4.0. If you have a standalone deployer, the OpenStack CLI and other clients will not be installed automatically on that node. If you require access to these clients, you will need to follow the procedure below to add the appropriate software.

## Important

In the `entry-scale-esx-kvm-vsa` and `entry-scale-kvm-esx-vsa-mml` example configurations, `eon-client` is installed only on `esx-compute` and `esx-ovsvapp` resources. With the changes to the OpenStack client installation, `eon-client` won't get installed on either dedicated HLM node or Controller nodes. You need to install the `eon-client` on either the dedicated deployer node or controller nodes so that you can use it to add the vCenter and activate the ESX compute clusters.

1. [OPTIONAL] Connect to your standalone deployer and try to use the OpenStack CLI:

```
source ~/keystone.osrc
openstack project list

-bash: openstack: command not found
```

2. Edit the configuration file containing details of your Control Plane, typically `~/helion/my_cloud/definition/data/control_plane`.
3. Locate the stanza for the cluster where you want to install the client(s). For a standalone deployer, this will look like the following extract:

```
clusters:
  - name: cluster0
    cluster-prefix: c0
    server-role: LIFECYCLE-MANAGER-ROLE
    member-count: 1
    allocation-policy: strict
    service-components:
      - ntp-server
      - lifecycle-manager
```

4. Choose the client(s) you wish to install from the following list of available clients:

- `openstack-client`
- `ceilometer-client`
- `cinder-client`
- `designate-client`
- `glance-client`
- `heat-client`
- `ironic-client`
- `keystone-client`
- `neutron-client`

```
- nova-client
- swift-client
- monasca-client
- barbican-client
```

5. Add the client(s) to the list of service-components - in this example, we add the openstack-client to the standalone deployer:

```
clusters:
  - name: cluster0
    cluster-prefix: c0
    server-role: LIFECYCLE-MANAGER-ROLE
    member-count: 1
    allocation-policy: strict
    service-components:
      - ntp-server
      - lifecycle-manager
      - openstack-client
      - ceilometer-client
      - cinder-client
      - designate-client
      - glance-client
      - heat-client
      - ironic-client
      - keystone-client
      - neutron-client
      - nova-client
      - swift-client
      - monasca-client
      - barbican-client
```

6. Commit the configuration changes:

```
cd ~/helion/hos/ansible
git add -A
git commit -m "Add explicit client service deployment"
```

7. Run the configuration processor, followed by the ready-deployment playbook:

```
cd ~/helion/hos/ansible
ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e re
ansible-playbook -i hosts/localhost ready-deployment.yml
```

8. Add the software for the clients using the following command:

```
cd /home/stack/scratch/ansible/next/hos/ansible
ansible-playbook -i hosts/verb_hosts clients-upgrade.yml
```

9. Check that the software has been installed correctly. In this instance, connect to your standalone deployer and try to use the OpenStack CLI:

```
source ~/keystone.osrc
openstack project list
```

You should now see a list of projects returned:

```
stack@helion-cp1-c0-m1-mgmt:~$ openstack project list
```

| ID                               | Name             |
|----------------------------------|------------------|
| 076b6e879f324183bbd28b46a7ee7826 | kronos           |
| 0b81c3a9e59c47cab0e208ea1bb7f827 | backup           |
| 143891c2a6094e2988358afc99043643 | octavia          |
| 1d3972a674434f3c95a1d5ed19e0008f | glance-swift     |
| 2e372dc57cac4915bf06bbe059fc547  | glance-check     |
| 383abda56aa2482b95fb9da0b9dd91f4 | monitor          |
| 606dd3b1fa6146668d468713413fb9a6 | swift-monitor    |
| 87db9d1b30044ea199f0293f63d84652 | admin            |
| 9fbb7494956a483ca731748126f50919 | demo             |
| a59d0c682474434a9ddc240ddfe71871 | services         |
| a69398f0f66a41b2872bcf45d55311a7 | swift-dispersion |
| f5ec48d0328d400992c1c5fb44ec238f | cinderinternal   |

---

# Chapter 25. Configuring Transport Layer Security (TLS)

## Abstract

TLS is enabled by default during the installation of HPE Helion OpenStack 5.0 and additional configuration options are available to secure your environment, as described below.

In HPE Helion OpenStack 5.0, you can provide your own certificate authority and certificates for internal and public virtual IP addresses (VIPs), and you should do so for any production cloud. The certificates automatically generated by HPE Helion OpenStack are useful for testing and setup, but you should always install your own for production use. Certificate installation is discussed below.

Please read the following if you're using the default cert-name: `my-public-cert` in your model.

The bundled test cert for public endpoints, located at `~/helion/my_cloud/config/tls/certs/my-public-cert`, is now expired but was left in the product in case you changed the content with your valid cert. Please verify if the certificate is expired and generate your own by following the guidelines further down on this page or by using a generic instruction from the web.

You can verify the expiry by running this command:

```
openssl x509 -in ~/helion/my_cloud/config/tls/certs/my-public-cert -noout -enddate  
notAfter=Oct 8 09:01:58 2016 GMT
```

Before you begin, the following list of terms will be helpful when generating and installing certificates.

Helion generated public CA

A Helion OpenStack generated public CA (`helion_frontend_cacert.crt`) is available for you to use in `/usr/local/share/ca-certificates`.

Fully qualified domain name (FQDN) of the public VIP

The registered domain name. A FQDN is not mandatory. It is perfectly valid to have no FQDN and use IP addresses instead. Note that you can use FQDNs on public endpoints, and you may change them whenever the need arises.

Certificate authority (CA) certificate

Your certificates must be signed by a CA, such as your internal IT department or a public certificate authority. For this example we will use a self-signed certificate.

Server certificate

It is easy to confuse server certificates and CA certificates. Server certificates reside on the server and CA certificates reside on the client. A server certificate affirms that the server that sent it serves a set of IP addresses, domain names, and set of services. A CA certificate is used by the client to authenticate this claim.

SAN (subject-alt-name)

The set of IP addresses and domain names in a server certificate request: A template for a server certificate.

Certificate signing request (CSR)

A blob of data generated from a certificate request and sent to a CA, which would then sign it, produce a server certificate, and send it back.

|              |                             |
|--------------|-----------------------------|
| External VIP | External virtual IP address |
| Internal VIP | Internal virtual IP address |

The major difference between an external VIP certificate and an internal VIP certificate is that the internal VIP has approximately 40 domain names in the SAN. This is because each service has a different domain name in HPE Helion OpenStack 5.0. So it is unlikely that you can create an internal server certificate before running the configuration processor. But after a configuration processor run, a certificate request would be created for each of your cert-names.

## Configuring TLS in the input model

For this example certificate configuration, let's assume there's no FQDN for the external VIP and that you're going to use the default IP address provided by HPE Helion OpenStack 5.0. Let's also assume that for the internal VIP you will use the defaults as well. If you were to call your certificate authority "example-CA," the CA certificate would then be called "example-CA.crt" and the key would be called "example-CA.key." In the following examples, the external VIP certificate will be named "example-public-cert" and the internal VIP certificate will be named "example-internal-cert."

### Note

Cautions:

Any time you make a cert change when using your own CA:

- You should use a distinct name from those already existing in config/tls/cacerts. This also means that you should not *reuse* your CA names (and use unique and distinguishable names such as MyCompanyXYZ\_PrivateRootCA.crt). A new name is what indicates that a file is new or changed, so reusing a name means that the file is not considered changed even its contents have changed.
- You should not remove any existing CA files from config/tls/cacerts
- If you want to remove an existing CA you must
  1. First remove the file.
  2. Then run:

```
ansible -i hosts/verb_hosts FND-STN -a 'sudo keytool -delete -alias debian:<file> -keystore /usr/lib/jvm/java-7-openjdk-amd64/jre/lib/security/cacerts -storepass changeit'
```

### Important

Be sure to install your own certificate for all production clouds after installing and testing your cloud. If you ever want to test or troubleshoot later, you will be able to revert to the sample certificate to get back to a stable state for testing.

### Note

Unless this is a new deployment, do not update both the certificate and the CA together. Add the CA first and then run a site deploy. Then update the certificate and run tls-reconfigure, FND-CLU-stop, FND-CLU-start and then hlm-reconfigure. If a playbook has failed, rerun it with -vv to get detailed error information. The configure, HAproxy restart, and reconfigure steps are included below. If this is a new deployment and you are adding your own certs/CA before running site.yml this caveat does not apply.

You can add your own certificate by following the instructions below. All changes must go into the following file:

```
~/helion/my_cloud/definition/data/network_groups.yml
```

Below are the entries for TLS for the internal and admin load balancers:

```
- provider: ip-cluster
  name: lb
  tls-components:
    - default
    components:
      # These services do not currently support TLS so they are not listed
      # under tls-components
      - nova-metadata
      roles:
        - internal
        - admin
      cert-file: helion-internal-cert
      # The helion-internal-cert is a reserved name and
      # this certificate will be autogenerated. You
      # can bring in your own certificate with a different name

      # cert-file: customer-provided-internal-cert
      # replace this with name of file in "config/tls/certs/"
```

The configuration processor will also create a request template for each named certificate under `info/cert_reqs/`. This will be of the form:

```
info/cert_reqs/customer-provided-internal-cert
```

These request templates contain the subject Alt-names that the certificates need. You can add to this template before generating your certificate signing request .

You would then send the CSR to your CA to be signed, and once you receive the certificate, place it in `config/tls/certs`

When you bring in your own certificate, you may want to bring in the trust chains (or CA certificate) for this certificate. This is usually not required if the CA is a public signer that is typically bundled with the operating system. However, we suggest you include it anyway by copying the file into the directory `config/cacerts/`.

## User-provided certificates and trust chains

HPE Helion OpenStack generates its own internal certificates but is designed to allow you to bring in your own certificates for the VIPs. Here is the general process.

1. You must have a server certificate and a CA certificate to go with it (unless the signer is a public CA and it's already bundled with most distributions).
2. You must decide the names of the server certificates and configure the `network_groups.yml` file in the input model such that each load balancer provider has at least one cert-name associated with it.
3. Run the configuration processor. Note that you may or may not have the certificate file at this point. The configuration processor would create certificate request file artifacts under `info/cert_reqs/` for each of the cert-name(s) in the `network_groups.yml` file. While there's no special reason to

use the request file created for an external endpoint VIP certificate, it is important to use the request files created for internal certificates since the canonical names for the internal VIP can be many and service specific and each of these need to be in the Subject Alt Names attribute of the certificate.

4. Create a certificate signing request for this request file and send it to your internal CA or a public CA to get it certified and issued with a certificate. You will now have a server certificate and possibly a trust chain or CA certificate.
5. Next, upload it to the lifecycle manager. Server certificates should be added to `config/tls/certs` and CA certificates should be added to `config/tls/cacerts`. The file extension should be `.crt` for the CA certificate to be processed by HPE Helion OpenStack. Detailed steps are next.

## Edit the input model to include your certificate files

Edit the load balancer configuration in `helion/my_cloud/definition/data/network_groups.yml`:

```
load-balancers:  
  - provider: ip-cluster  
    name: lb  
    tls-components:  
      - default  
    components:  
      - vertica  
      - nova-metadata  
    roles:  
      - internal  
      - admin  
    cert-file: example-internal-cert #<<<----- Certificate name for the internal VIP  
  
  - provider: ip-cluster  
    name: extlb  
    external-name: myhelion.test #<<<----- Use just IP for the external VIP in this  
    tls-components:  
      - default  
    roles:  
      - public  
    cert-file: example-public-cert #<<<----- Certificate name for the external VIP
```

Commit your changes to the local git repository and run the configuration processor:

```
cd ~/helion/hos/ansible  
git add -A  
git commit -m "changed VIP certificates"  
ansible-playbook -i hosts/localhost config-processor-run.yml
```

Verify that certificate requests have been generated by the configuration processor for every certificate file configured in the `networks_groups.yml` file. In this example, there are two files, as shown from the list command:

```
ls ~/helion/my_cloud/info/cert_reqs  
example-internal-cert  
example-public-cert
```

# Generate a self-signed CA

## Note

In a production setting you will not perform this step. You will use your company's CA or a valid public CA.

This section demonstrates to how you can create your own self-signed CA and then use this CA to sign server certificates. This CA can be your organization's IT internal CA that is self-signed and whose CA certificates are deployed on your organization's machines. This way the server certificate becomes legitimate.

## Note

Please use a unique CN for your example Certificate Authority and do not install multiple CA certificates with the same CN into your cloud.

Copy the commands below to the command line and execute. This will cause the two files, example-CA.key and example-CA.crt to be created:

```
export EXAMPLE_CA_KEY_FILE='example-CA.key'
export EXAMPLE_CA_CERT_FILE='example-CA.crt'
openssl req -x509 -batch -newkey rsa:2048 -nodes -out "${EXAMPLE_CA_CERT_FILE}" \
-keyout "${EXAMPLE_CA_KEY_FILE}" \
-subj "/C=UK/O=hp/CN=YourOwnUniqueCertAuthorityName" \
-days 365
```

You can tweak the subj and days settings above to meet your needs, or to test. For instance, if you want to test what happens when a CA expires, you can set 'days' to a very low value. Grab the configuration processor-generated request file from info/cert\_reqs/:

```
cat ~/helion/my_cloud/info/cert_reqs/example-internal-cert
```

Now, copy this file to your working directory and appended a .req extension to it.

```
cp ~/helion/my_cloud/info/cert_reqs/example-internal-cert example-internal-cert.req
```

### Example 25.1. Certificate request file

```
[req]
distinguished_name = req_distinguished_name
req_extensions = v3_req
prompt = no

[ req_distinguished_name ]
CN = "helion-vip"

[ v3_req ]
basicConstraints = CA:FALSE
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = "deployerincloud-ccp-c0-m1-mgmt"
```

```
DNS.2 = "deployerincloud-ccp-vip-CEI-API-mgmt"
DNS.3 = "deployerincloud-ccp-vip-CND-API-mgmt"
DNS.4 = "deployerincloud-ccp-vip-DES-API-mgmt"
DNS.5 = "deployerincloud-ccp-vip-FND-MDB-mgmt"
DNS.6 = "deployerincloud-ccp-vip-FND-RMQ-mgmt"
DNS.7 = "deployerincloud-ccp-vip-FND-VDB-mgmt"
DNS.8 = "deployerincloud-ccp-vip-FRE-API-mgmt"
DNS.9 = "deployerincloud-ccp-vip-GLA-API-mgmt"
DNS.10 = "deployerincloud-ccp-vip-GLA-REG-mgmt"
DNS.11 = "deployerincloud-ccp-vip-HEA-ACF-mgmt"
DNS.12 = "deployerincloud-ccp-vip-HEA-ACW-mgmt"
DNS.13 = "deployerincloud-ccp-vip-HEA-API-mgmt"
DNS.14 = "deployerincloud-ccp-vip-HUX-SVC-mgmt"
DNS.15 = "deployerincloud-ccp-vip-HZN-WEB-mgmt"
DNS.16 = "deployerincloud-ccp-vip-KEY-API-mgmt"
DNS.17 = "deployerincloud-ccp-vip-KEYMGR-API-mgmt"
DNS.18 = "deployerincloud-ccp-vip-LOG-API-mgmt"
DNS.19 = "deployerincloud-ccp-vip-LOG-SVR-mgmt"
DNS.20 = "deployerincloud-ccp-vip-MON-API-mgmt"
DNS.21 = "deployerincloud-ccp-vip-NEU-SVR-mgmt"
DNS.22 = "deployerincloud-ccp-vip-NOV-API-mgmt"
DNS.23 = "deployerincloud-ccp-vip-NOV-MTD-mgmt"
DNS.24 = "deployerincloud-ccp-vip-OCT-API-mgmt"
DNS.25 = "deployerincloud-ccp-vip-OPS-WEB-mgmt"
DNS.26 = "deployerincloud-ccp-vip-SHP-API-mgmt"
DNS.27 = "deployerincloud-ccp-vip-SWF-PRX-mgmt"
DNS.28 = "deployerincloud-ccp-vip-admin-CEI-API-mgmt"
DNS.29 = "deployerincloud-ccp-vip-admin-CND-API-mgmt"
DNS.30 = "deployerincloud-ccp-vip-admin-DES-API-mgmt"
DNS.31 = "deployerincloud-ccp-vip-admin-FND-MDB-mgmt"
DNS.32 = "deployerincloud-ccp-vip-admin-FRE-API-mgmt"
DNS.33 = "deployerincloud-ccp-vip-admin-GLA-API-mgmt"
DNS.34 = "deployerincloud-ccp-vip-admin-HEA-ACF-mgmt"
DNS.35 = "deployerincloud-ccp-vip-admin-HEA-ACW-mgmt"
DNS.36 = "deployerincloud-ccp-vip-admin-HEA-API-mgmt"
DNS.37 = "deployerincloud-ccp-vip-admin-HUX-SVC-mgmt"
DNS.38 = "deployerincloud-ccp-vip-admin-HZN-WEB-mgmt"
DNS.39 = "deployerincloud-ccp-vip-admin-KEY-API-mgmt"
DNS.40 = "deployerincloud-ccp-vip-admin-KEYMGR-API-mgmt"
DNS.41 = "deployerincloud-ccp-vip-admin-MON-API-mgmt"
DNS.42 = "deployerincloud-ccp-vip-admin-NEU-SVR-mgmt"
DNS.43 = "deployerincloud-ccp-vip-admin-NOV-API-mgmt"
DNS.44 = "deployerincloud-ccp-vip-admin-OPS-WEB-mgmt"
DNS.45 = "deployerincloud-ccp-vip-admin-SHP-API-mgmt"
DNS.46 = "deployerincloud-ccp-vip-admin-SWF-PRX-mgmt"
DNS.47 = "192.168.245.5"
IP.1 = "192.168.245.5"

=====end of certificate request file.
```

## Note

In the case of a public VIP certificate, please add all the FQDNs you want it to support. Currently Helion does not add the hostname for the external-name specified in `network_groups.yml`

to the certificate request file . However, you can add it to the certificate request file manually. Here we assume that `myhelion.test` is your external-name. In that case you would add this line (to the full that is shown above):

```
DNS.48 = "myhelion.test"
```

### Note

Any attempt to use IP addresses rather than FQDNs in certificates must use subject alternate name entries that list both the IP address (needed for Google) and DNS with an IP (needed for a Python bug workaround). Failure to create the certificates in this manner will cause future installations of Go-based tools (such as Cloud Foundry, Stackato and other PaaS components) to fail.

## Generate a certificate signing request

Now that you have a CA and a certificate request file, it's time to generate a CSR.

```
export EXAMPLE_SERVER_KEY_FILE='example-internal-cert.key'
export EXAMPLE_SERVER_CSR_FILE='example-internal-cert.csr'
export EXAMPLE_SERVER_REQ_FILE=example-internal-cert.req
openssl req -newkey rsa:2048 -nodes -keyout "$EXAMPLE_SERVER_KEY_FILE" -out "$EXAMPLE_SERVER_CSR_FILE"
```

Note that in production you would usually send the generated `example-internal-cert.csr` file to your IT department. But in this example you are your own CA, so sign and generate a server certificate.

## Generate a server certificate

### Note

In a production setting you will not perform this step. You will send the CSR created in the previous section to your company CA or a to a valid public CA and have them sign and send you back the certificate.

This section demonstrates how you would use your own self-signed CA that you created earlier to sign and generate a server certificate. A server certificate is essentially a signed public key, the signer being a CA and trusted by a client. When you install this the signing CA's certificate (called CA certificate or trust chain) on the client machine, you are telling the client to trust this CA, and thereby implicitly trusting any server certificates that are signed by this CA, thus creating a trust anchor.

### CA configuration file

When the CA signs the certificate, it uses a configuration file that tells it to verify the CSR. Note that in a production scenario the CA takes care of this for you.

Create a file called `openssl.cnf` and add the following contents to it.

```
# Copyright 2010 United States Government as represented by the
# Administrator of the National Aeronautics and Space Administration.
# All Rights Reserved.
#...
# OpenSSL configuration file.
#
```

```
# Establish working directory.

dir = .

[ ca ]
default_ca = CA_default

[ CA_default ]
serial = $dir/serial
database = $dir/index.txt
new_certs_dir = $dir/
certificate = $dir/cacert.pem
private_key = $dir/cakey.pem
unique_subject = no
default_crl_days = 365
default_days = 365
default_md = md5
preserve = no
email_in_dn = no
nameopt = default_ca
certopt = default_ca
policy = policy_match
copy_extensions = copy

# NOTE(dprince): stateOrProvinceName must be 'supplied' or 'optional' to
# work around a stateOrProvince printable string UTF8 mismatch on
# RHEL 6 and Fedora 14 (using openssl-1.0.0-4.el6.x86_64 or
# openssl-1.0.0d-1.fc14.x86_64)
[ policy_match ]
countryName = optional
stateOrProvinceName = optional
organizationName = optional
organizationalUnitName = optional
commonName = supplied
emailAddress = optional

[ req ]
default_bits = 1024 # Size of keys
default_keyfile = key.pem # name of generated keys
default_md = md5 # message digest algorithm
string_mask = nombstr # permitted characters
distinguished_name = req_distinguished_name
req_extensions = v3_req
x509_extensions = v3_ca

[ req_distinguished_name ]
# Variable name Prompt string
#-----
0.organizationName = Organization Name (company)
organizationalUnitName = Organizational Unit Name (department, division)
emailAddress = Email Address
emailAddress_max = 40
localityName = Locality Name (city, district)
stateOrProvinceName = State or Province Name (full name)
```

```
countryName = Country Name (2 letter code)
countryName_min = 2
countryName_max = 2
commonName = Common Name (hostname, IP, or your name)
commonName_max = 64

# Default values for the above, for consistency and less typing.
# Variable name Value
#-----
0.organizationName_default = Hewlett-Packard-Enterprise
localityName_default = Bristol
stateOrProvinceName_default = Bristol
countryName_default = UK

[ v3_ca ]
basicConstraints = CA:TRUE
subjectKeyIdentifier = hash
authorityKeyIdentifier = keyid:always,issuer:always
subjectAltName = @alt_names

[ v3_req ]
basicConstraints = CA:FALSE
subjectKeyIdentifier = hash

[ alt_names ]

##### end of openssl.cnf #####
```

### Sign and create a server certificate

Now you can sign the server certificate with your CA. Copy the commands below to the command line and execute. This will cause the one file, example-internal-cert.crt, to be created:

```
export EXAMPLE_SERVER_CERT_FILE='example-internal-cert.crt'
export EXAMPLE_SERVER_CSR_FILE='example-internal-cert.csr'
export EXAMPLE_CA_KEY_FILE='example-CA.key'
export EXAMPLE_CA_CERT_FILE='example-CA.crt'

touch index.txt
openssl rand -hex -out serial 6

openssl ca -batch -notext -md sha256 -in "$EXAMPLE_SERVER_CSR_FILE" \
-crt "$EXAMPLE_CA_CERT_FILE" \
-keyfile "$EXAMPLE_CA_KEY_FILE" \
-out "$EXAMPLE_SERVER_CERT_FILE" \
-config openssl.cnf -extensions v3_req
```

Finally, concatenate both the server key and certificate in preparation for uploading to the lifecycle manager.

```
cat example-internal-cert.key example-internal-cert.crt > example-internal-cert
```

Note that you have only created the internal-cert in this example. Repeat the above sequence for example-public-cert. Make sure you use the appropriate certificate request generated by the configuration processor.

# Upload to the lifecycle manager

The following two files created from the example run above will need to be uploaded to the lifecycle manager and copied into config/tls:

- example-internal-cert
- example-CA.crt

Once on the lifecycle manager, execute the following two copy commands to copy to their respective directories. Note if you had created an external cert, you can copy that in a similar manner, specifying its name using the copy command as well.

```
cp example-internal-cert ~/helion/my_cloud/config/tls/certs/  
cp example-CA.crt ~/helion/my_cloud/config/tls/cacerts/
```

## Continue with the deployment

Next, log into the lifecycle manager node, and save and commit the changes to the local git repository:

```
cd ~/helion/hos/ansible  
git add -A  
git commit -m "updated certificate and CA"
```

Next, rerun the config-processor-run playbook, and run ready-deployment.yml:

```
cd ~/helion/hos/ansible  
ansible-playbook -i hosts/localhost config-processor-run.yml  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

If you receive any prompts, enter the required information.

## Note

For automated installation (e.g. CI) you can specify the required passwords on the Ansible command line. For example, the command below will disable encryption by the configuration processor:

```
ansible-playbook -i hosts/localhost config-processor-run.yml -e encrypt="" -e r
```

Run this series of runbooks to complete the deployment:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts tls-reconfigure.yml  
ansible-playbook -i hosts/verb_hosts FND-CLU-stop.yml  
ansible-playbook -i hosts/verb_hosts FND-CLU-start.yml  
ansible-playbook -i hosts/verb_hosts monasca-stop.yml  
ansible-playbook -i hosts/verb_hosts monasca-start.yml  
ansible-playbook -i hosts/verb_hosts hlm-reconfigure.yml
```

# Configuring the cipher suite

By default, the cipher suite is set to: HIGH:!aNULL:!eNULL:!DES:!3DES. This setting is recommended in the OpenStack documentation site [<http://docs.openstack.org/security-guide/secure-communications.html>].

cation/introduction-to-ssl-and-tls.html]. You may override this. To do so, open config/haproxy/default.yml and edit it. The parameters can be found under the haproxy\_globals list.

```
- "ssl-default-bind-ciphers HIGH:!aNULL:!eNULL:!DES:!3DES"  
- "ssl-default-server-ciphers HIGH:!aNULL:!eNULL:!DES:!3DES"
```

Make the changes as needed. It's best to keep the two options identical.

## Testing

You can easily determine if an endpoint is behind TLS. To do so, run the following command, which probes a Keystone identity service endpoint that's behind TLS:

```
echo | openssl s_client -connect 192.168.245.5:5000 | openssl x509 -fingerprint -n  
depth=0 CN = helion-vip  
verify error:num=20:unable to get local issuer certificate  
verify return:1  
depth=0 CN = helion-vip  
verify error:num=27:certificate not trusted  
verify return:1  
depth=0 CN = helion-vip  
verify error:num=21:unable to verify the first certificate  
verify return:1  
DONE  
SHA1 Fingerprint=C6:46:1E:59:C6:11:BF:72:5E:DD:FC:FF:B0:66:A7:A2:CC:32:1C:B8
```

The next command probes a MySQL endpoint that is not behind TLS:

```
echo | openssl s_client -connect 192.168.245.5:3306 | openssl x509 -fingerprint -n  
140448358213264:error:140770FC:SSL routines:SSL23_GET_SERVER_HELLO:unknown protocol  
unable to load certificate  
140454148159120:error:0906D06C:PEM routines:PEM_read_bio:no start line:pem_lib.c:7
```

## Verifying that the trust chain is correctly deployed

You can determine if the trust chain is correctly deployed by running the following commands:

```
echo | openssl s_client -connect 192.168.245.9:5000 2>/dev/null | grep code  
Verify return code: 21 (unable to verify the first certificate)  
echo | openssl s_client -connect 192.168.245.9:5000 -CAfile /usr/local/share/ca-certificates  
Verify return code: 0 (ok)
```

Here, the first command produces error 21, which is then fixed by providing the CA certificate file. This verifies that the CA certificate matches the server certificate.

## Turning TLS on or off

You should leave TLS enabled in production. However, if you need to disable it for any reason, you must change "tls-components" to "components" in network\_groups.yml (as shown earlier) and comment out the cert-file. Additionally, if you have a network\_groups.yml file from a previous installation, you won't have TLS enabled unless you change "components" to "tls-components" in that file. By default,

Horizon is configured with TLS in the input model. Note that you should not disable TLS in the input model for Horizon as that is a public endpoint and is required. Additionally, you should keep all services behind TLS, but using the input model file `network_groups.yml` you may turn TLS off for a service for troubleshooting or debugging. TLS should always be enabled for production environments.

If you are using an example input model on a clean install, all supported TLS services will be enabled before deployment of your cloud. If you want to change this setting later, for example, when upgrading, you can change the input model and reconfigure the system. The process is as follows:

Edit the input model `network_groups.yml` file appropriately, as described above. Then, commit the changes to the git repository:

```
cd ~/helion/hos/ansible/  
git add -A  
git commit -m "TLS change"
```

Change directories again and run the configuration processor and ready deployment playbooks:

```
ansible-playbook -i hosts/localhost config-processor-run.yml  
ansible-playbook -i hosts/localhost ready-deployment.yml
```

Change directories again and run the reconfigure playbook:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts hlm-reconfigure.yml
```

---

# Chapter 26. Configuring Availability Zones

The lifecycle manager only creates a default availability zone during installation. If your system has multiple failure/availability zones defined in your input model, these zones will not get created automatically.

Once the installation has finished, you can run the `nova-cloud-configure.yml` playbook to configure availability zones and assign compute nodes to those zones based on the configuration specified in the model.

You can run the playbook `nova-cloud-configure.yml` any time you make changes to the configuration of availability zones in your input model. Alternatively, you can use Horizon or the command line to perform the configuration.

For more details, see .

---

# Chapter 27. Configuring Load Balancer as a Service

## Abstract

The HPE Helion OpenStack Neutron LBaaS service supports several load balancing providers. By default, both Octavia and the namespace haproxy driver are configured to be used. We describe this in more detail here.

## Warning

If you are planning to upgrade from HPE Helion OpenStack 3.0 or HPE Helion OpenStack 4.0, please contact F5 and HPE PointNext to determine which F5 drivers have been certified for use with Helion OpenStack. Loading drivers not certified by HPE may result in catastrophic failure of your cloud deployment. The last tested versions are 8.0.8 for HPE Helion OpenStack 3.x and 9.0.3 for HPE Helion OpenStack 4.x. More information is expected in 4th quarter 2017, including the correct drivers for HPE Helion OpenStack 5.x.

The HPE Helion OpenStack Neutron LBaaS service supports several load balancing providers. By default, both Octavia and the namespace haproxy driver are configured to be used. A user can specify which provider to use with the `--provider` flag upon load balancer creation.

Example:

```
neutron lbaas-loadbalancer-create --name <name> --provider [octavia|haproxy] <subn
```

If you don't specify the `--provider` option it will default to Octavia. The Octavia driver provides more functionality than the haproxy namespace driver which is deprecated. The haproxy namespace driver will be retired in a future version of HPE Helion OpenStack.

There are additional drivers for 3rd party hardware load balancers. Please refer to the vendor directly. The `neutron service-provider-list` command displays not only the currently installed load balancer drivers but also other installed services such as VPN. You can see a list of available services as follows:

```
$ neutron service-provider-list
+-----+-----+-----+
| service_type | name   | default |
+-----+-----+-----+
LOADBALANCERV2	octavia	True
VPN	openswan	True
LOADBALANCERV2	haproxy	False
LOADBALANCERV2	octavia	True
VPN	openswan	True
LOADBALANCERV2	haproxy	False
+-----+-----+-----+
```

## Note

In the example above, the providers are listed twice. This is a limitation in HPE Helion OpenStack 3.0. Also note that the Octavia load balancer provider is listed as the default.

## Prerequisites

You will need to create an external network and create an image to test LBaaS functionality. If you have already created an external network and registered an image, this step can be skipped.

**Creating an external network:** the section called “Creating an External Network”.

**Creating and uploading a Glance image:**

## Octavia Load Balancing Provider

The Octavia Load balancing provider bundled with HPE Helion OpenStack 5.0 is an operator grade load balancer for OpenStack. It is based on the Mitaka version of Octavia. It differs from the namespace driver by starting a new nova virtual machine to house the haproxy load balancer software, called an *amphora*, that provides the load balancer function. A virtual machine for each load balancer requested provides a better separation of load balancers between tenants and makes it easier to grow load balancing capacity alongside compute node growth. Additionally, if the virtual machine fails for any reason Octavia will replace it with a replacement VM from a pool of spare VM's, assuming that the feature is configured.

### Note

The Health Monitor will not create or replace failed amphorae. If the pool of spare VM's is exhausted there will be no additional virtual machines to handle load balancing requests.

Octavia uses two-way SSL encryption to communicate with the amphora. There are demo Certificate Authority (CA) certificates included with HPE Helion OpenStack 5.0 in /home/stack/scratch/ansible/next/hos/ansible/roles/octavia-common/files on the lifecycle manager. For additional security in production deployments, all certificate authorities should be replaced with ones you generated yourself by running the following commands:

```
openssl genrsa -passout pass:foobar -des3 -out cakey.pem 2048
openssl req -x509 -passin pass:foobar -new -nodes -key cakey.pem -out ca_01.pem
openssl genrsa -passout pass:foobar -des3 -out servercakey.pem 2048
openssl req -x509 -passin pass:foobar -new -nodes -key cakey.pem -out serverca_01.pem
```

For more details refer to the openssl man page [<https://www.openssl.org/docs/manmaster/apps/openssl.html>].

### Note

If you change the certificate authority and have amphora running with an old CA you won't be able to control the amphora. The amphora's will need to be failed over so they can utilize the new certificate. If you change the CA password for the server certificate you need to change that in the Octavia configuration files as well. See the guide for more information.

## Setup of prerequisites

### Octavia Network and Management Network Ports

The Octavia management network and Management network must have access to each other. If you have a configured firewall between the Octavia management network and Management network, you must open up the following ports to allow network traffic between the networks.

- From Management network to Octavia network
  - TCP 9443 (amphora API)
- From Octavia network to Management network
  - TCP 9876 (Octavia API)
  - UDP 5555 (Octavia Health Manager)

### Installing the Amphora Image

Octavia is utilizing Nova VMs for its load balancing function and HPE provides images used to boot those VM's called `octavia-amphora-haproxy`.

### Warning

Without these images the Octavia load balancer will not work.

You can download the **Octavia Amphora HA Proxy Guest Image** from HPE Software Entitlement Portal.

- Sign in and download the product and signature files from the following link: Software Entitlement Portal [<http://www.hpe.com/software/entitlements>]
- Once the image is downloaded it needs to be placed on the lifecycle manager node and the image registered.

### Register the image.

1. Switch to the ansible directory and register the image by giving the full path and name (e.g. `/tmp/octavia-amphora-haproxy-guest-image.tgz`) as argument to service\_package:

```
$ cd ~/scratch/ansible/next/hos/ansible/  
$ ansible-playbook -i hosts/verb_hosts -e service_package=<image path/name> serv
```

2. Source the service user (this can be done on a different computer)

```
$ source service.osrc
```

3. Verify that the image was uploaded and registered (this can be done on a computer with access to the glance CLI client)

```
$ glance image-list  
+-----+-----+  
| ID | Name |  
+-----+-----+  
| 01ff1f0d-fc35-4e3e-bae2-e7e2ee1f65b6 | cirros-0.3.3-x86_64 |  
| e64cb914-15d2-4ad8-a63c-b7c60a6c232e | octavia-amphora-x64-haproxy_hos-3.0.0 |  
+-----+-----+
```

### Note

In rare circumstances the image won't be registered and you will have to run the whole registration again. If you run the registration by accident, the system will only upload a new image if the underlying image has been changed.

4. Check the status of the image by running `glance image-show <image-id>`.

```
$ glance image-show e64cb914-15d2-4ad8-a63c-b7c60a6c232e
+-----+-----+
| Property | Value |
+-----+-----+
| checksum | 094a553d38fb5bfdb43f4a662d84ec2e
| container_format | bare
| created_at | 2016-04-14T23:05:21Z
| disk_format | qcow2
| id | e64cb914-15d2-4ad8-a63c-b7c60a6c232e
| min_disk | 0
| min_ram | 0
| name | octavia-amphora-x64-haproxy_hos-3.0.0
| owner | 0671a8d4d71c44ffb210c11cb5d11f7f
| protected | False
| size | 434379264
| status | active
| tags | []
| updated_at | 2016-04-14T23:05:36Z
| virtual_size | None
| visibility | private
+-----+-----+
```

## **Important**

In the example above, the status of the image is *active* which means the image was successfully registered. If a status of the images is *queued*, you must run the image registration again.

Please be aware that if you have already created load balancers they won't receive the new image. Only load balancers created after the image has been successfully installed will use the new image. If existing load balancers need to be switched to the new image please follow the instructions in the guide.

### **Setup network, subnet, router, security and IP's**

If you have already created a network, subnet, router, security settings and IP's you can skip the following steps and go directly to creating the load balancers.

1. Create a network.

```
neutron net-create lb_net1
+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | True
| id | 71a1ac88-30a3-48a3-a18b-d98509fbef5c
| mtu | 0
| name | lb_net1
| provider:network_type | vxlan
| provider:physical_network |
| provider:segmentation_id | 1061
| router:external | False
| shared | False
| status | ACTIVE
| subnets |
```

```
+----+-----+-----+
| tenant_id | 4b31d0508f83437e83d8f4d520cda22f |
+----+-----+
```

2. Create a subnet.

```
neutron subnet-create --name lb_subnet1 lb_net1 10.247.94.128/26 --gateway 10.247.94.129
+-----+-----+
| Field | Value |
+-----+-----+
allocation_pools	[ {"start": "10.247.94.130", "end": "10.247.94.190"} ]
cidr	10.247.94.128/26
dns_nameservers	
enable_dhcp	True
gateway_ip	10.247.94.129
host_routes	
id	6fc2572c-53b3-41d0-ab63-342d9515f514
ip_version	4
ipv6_address_mode	
ipv6_ra_mode	
name	lb_subnet1
network_id	71a1ac88-30a3-48a3-a18b-d98509fbef5c
subnetpool_id	
tenant_id	4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

3. Create a router.

```
neutron router-create --distributed False lb_router1
+-----+-----+
| Field | Value |
+-----+-----+
admin_state_up	True
distributed	False
external_gateway_info	
ha	False
id	6aaafc9a9-93f6-4d7e-94f2-3068b034b823
name	lb_router1
routes	
status	ACTIVE
tenant_id	4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

4. Add interface to router. In this example, `neutron router-interface-add lb_router1 lb_subnet1` will add interface `426c5898-f851-4f49-b01f-7a6fe490410c` to router `lb_router1`.

```
neutron router-interface-add lb_router1 lb_subnet1
```

5. Set gateway for router.

```
neutron router-gateway-set lb_router1 ext-net
```

6. Check networks.

```
neutron net-list
+-----+-----+-----+
| id | name | subnets |
+-----+-----+-----+
```

| d3cb12a6-a000-4e3e-82c4-ee04aa169291 | ext-net          | f4152001-2500-4ebe-ba9d-a8d6149a50df | 10.247.96.0/23   |
|--------------------------------------|------------------|--------------------------------------|------------------|
| 8306282a-3627-445a-a588-c188b6a13163 | OCTAVIA-MGMT-NET | f00299f8-3403-45ae-ac4b-58af41d57bdc | 10.247.94.128/26 |
| 71a1ac88-30a3-48a3-a18bd98509fbef5c  | lb_net1          | 6fc2572c-53b3-41d0-ab63-342d9515f514 | 10.247.94.128/26 |

7. Create security group.

| neutron security-group-create lb_secgroup1 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Field                                      | Value                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| description                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| id                                         | 75343a54-83c3-464c-8773-802598afaaee9                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| name                                       | lb_secgroup1                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| security_group_rules                       | {"remote_group_id": null, "direction": "egress", "remote_ip_prefix": "4b31d0508f83437e83d8f4d520cda22f", "port_range_max": null, "port_range_min": null, "ethertype": "IPv4", "id": "20a"}, {"remote_group_id": null, "direction": "egress", "remote_ip_prefix": "4b31d0508f83437e83d8f4d520cda22f", "port_range_max": null, "port_range_min": null, "ethertype": "IPv6", "id": "563"}, {"remote_group_id": null, "direction": "ingress", "remote_ip_prefix": null, "port_range_max": null, "port_range_min": null, "ethertype": "IPv4", "id": "20b"}, {"remote_group_id": null, "direction": "ingress", "remote_ip_prefix": null, "port_range_max": null, "port_range_min": null, "ethertype": "IPv6", "id": "564"}] |
| tenant_id                                  | 4b31d0508f83437e83d8f4d520cda22f                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

8. Create icmp security group rule.

| neutron security-group-rule-create lb_secgroup1 --protocol icmp |                                       |
|-----------------------------------------------------------------|---------------------------------------|
| Field                                                           | Value                                 |
| direction                                                       | ingress                               |
| ethertype                                                       | IPv4                                  |
| id                                                              | 16d74150-a5b2-4cf6-82eb-a6c49a972d93  |
| port_range_max                                                  |                                       |
| port_range_min                                                  |                                       |
| protocol                                                        | icmp                                  |
| remote_group_id                                                 |                                       |
| remote_ip_prefix                                                |                                       |
| security_group_id                                               | 75343a54-83c3-464c-8773-802598afaaee9 |
| tenant_id                                                       | 4b31d0508f83437e83d8f4d520cda22f      |

9. Create TCP port 22 rule.

| neutron security-group-rule-create lb_secgroup1 --protocol tcp --port-range-min |         |
|---------------------------------------------------------------------------------|---------|
| Field                                                                           | Value   |
| direction                                                                       | ingress |
| ethertype                                                                       | IPv4    |

|                   |                                       |
|-------------------|---------------------------------------|
| id                | 472d3c8f-c50f-4ad2-97a1-148778e73af5  |
| port_range_max    | 22                                    |
| port_range_min    | 22                                    |
| protocol          | tcp                                   |
| remote_group_id   |                                       |
| remote_ip_prefix  |                                       |
| security_group_id | 75343a54-83c3-464c-8773-802598afaaee9 |
| tenant_id         | 4b31d0508f83437e83d8f4d520cda22f      |

10.Create TCP port 80 rule.

|                                                                                 |                                       |
|---------------------------------------------------------------------------------|---------------------------------------|
| neutron security-group-rule-create lb_secgroup1 --protocol tcp --port-range-min | +-----+-----+                         |
| +-----+-----+                                                                   | Field   Value                         |
| +-----+-----+                                                                   | direction   ingress                   |
| ethertype   IPv4                                                                |                                       |
| id                                                                              | 10a76cad-8b1c-46f6-90e8-5ddd279e5f7   |
| port_range_max                                                                  | 80                                    |
| port_range_min                                                                  | 80                                    |
| protocol                                                                        | tcp                                   |
| remote_group_id                                                                 |                                       |
| remote_ip_prefix                                                                |                                       |
| security_group_id                                                               | 75343a54-83c3-464c-8773-802598afaaee9 |
| tenant_id                                                                       | 4b31d0508f83437e83d8f4d520cda22f      |

11.If you haven't already created a keypair, create one now with nova keypair-add. You'll use the keypair to boot images.

```
nova keypair-add lb_kp1 > lb_kp1.pem
chmod 400 lb_kp1.pem
cat lb_kp1.pem
-----BEGIN RSA PRIVATE KEY-----
MIIEqAIBAAKCAQEakbw5W/XWGRGC0LAJI7lttR7EdDfiTDeFJ7A9b9Cff+OMXjhx
WL26eKIr+jp8DR64YjV2mNnQLsDyCxeKFpkjnjnGRId3KVAeV5sRQqXgtaCXI+Rvd
IyUtd8p1cp3DRgTd1dx00oL6bBmwrZatNrrRn4HgKc2c7ErekeXrwLHyE0Pia/pz
C6qs0coRdfIeXxsmS3kXExP0YfsswRS/OyD18QhRAF0ZW/zV+DQIi8+HpLZT+RW1
8sTTYZ6b0kXoH9wLER4IUBj1I1IyrYdx1Ahe2VIn+tF0Ec4nDbn1py9iwEfGmn0+
N2jHCJAkrK/QhWdXO408zeXfL4mCZ9FybW4nzQIDAQABoIBACe0PvgB+v8FuIGp
FjR32J8b7ShF+hIOpufzrCoFzRCKLruV4bzuphstBZK/0QG6Nz/71X99Cq9SwCGp
pXrK7+3EoGl8CB/xmTUylVA4gRb6BNNsdkuXW9ZigrJirs0rkk8uIwRV0GsYbP5A
Kp7ZNTmjqDN75aC1ngRfhGgT1QUOdxBH+4xSb7GukekD13V8V5MF1Qft089asdWp
1/TpvhYeW9O92xEnZ3qXQYpXYQgEFQoM2PKa3VW7FGLgfw9gdS/MSqpHuHGyKmj1
uT6upUX+Lofbe7V+9kfxuV32sLL/S5YFvkBy2q8VpuEV1sXI7O7Sc411WX4cqmlb
YoFwhrkCggCBALkYE70MTtdCAGcMotJhTiis518d4U/fn1x0zus43XV5Y7wCnMuU
r5vCoK+a+TR9Ekzc/GjccAx7Wz/YYKp6G8FXW114dLcADXZjqjIlX7ifUud4sLCS
y+x3KAJa7LqyzH53I6Fots9RaB5xx4gZ2WjcJquCTbATZWj7jlyGeNgvAoIAgQDJ
h0r0Te5IliYbCRg+ES9YRZzH/PSLuIn00bbLvpOPNEoKe2PxS+KI8Fqp6ZIDAB3c
4EPOK5QrJvAny9Z58ZArrNZi15t84KEVAKWUATl+c4SmHc8sW/atgmUoqIzgDQXe
AlwadHLY7JCdg7EYDuUxuTKLLOdqfpf6fKkhNxtEwwKCAIAMxi+d5aIPUxvKAOI/
2L1XKYRCrkI9i/ZooBsjustH1+JG8iQWFozy/aDhExlJKoBMiQOIerpABHIZYqqtJ
```

```
OLIvrsK8ebK8aoGDWS+G1HN9v2kuVnMDTK5MPJEDUJkj7XEVju1lNZSCTGD+MOYP
a5FInmA1zZbX4tRKoNjZFh0uwKCAIEAiLs7drAdOLBu4C72fL4K11jwu5t7jATD
zRAwduIxMzq/lYcMU2RaEdEJonivsUt193NNbeeRWwnLLSUWupvT114pAt0ISNzb
TbbB4F5IVOwpls9ozc8DecubuM9K7YTIC02kkepqNZWjtMsx74HDru3a5iSSSkvj
73Z/BeMupCMCggCAS48BsrgsDsHSHE3tO4D8pAIr1r+6WPQn49pT3GIRDQNc7a0
d9PfXmPoe/PxUlqaXoNAvT99+nNEadp+GTId21VM0Y28pn3EkIGE1Cqoeyl3BEO8
f9SUiRNruDnH4F4OclsDBlmqWXImuXRfeiDHxM8X03UDZoqyHmGD3RqA53I=
-----END RSA PRIVATE KEY-----
```

12.Check and boot images.

```
nova image-list
+-----+-----+
| ID | Name |
+-----+-----+
| 04b1528b-b1e2-45d4-96d1-fbe04c6b2efd | cirros-0.3.3-x86_64 |
| 8aa51e01-e6f4-480f-b91e-08ea14178f2f | octavia-amphora-x64-haproxy_hos-3.0.0 |
+-----+-----+
```

Boot first VM.

```
nova boot --flavor 1 --image 04b1528b-b1e2-45d4-96d1-fbe04c6b2efd --key-name lb_kp1
+-----+-----+
| Property | Value |
+-----+-----+
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	-
OS-EXT-SRV-ATTR:host	-
OS-EXT-SRV-ATTR:hypervisor_hostname	-
OS-EXT-SRV-ATTR:instance_name	instance-00000031
OS-EXT-STS:power_state	0
OS-EXT-STS:task_state	scheduling
OS-EXT-STS:vm_state	building
OS-SRV-USG:launched_at	-
OS-SRV-USG:terminated_at	-
accessIPv4	
accessIPv6	
adminPass	NeVvhP5E8iCy
config_drive	
created	2016-06-15T16:53:00Z
flavor	m1.tiny (1)
hostId	
id	dfdfe15b-ce8d-469c-a9d8-2cea0e7ca287
image	cirros-0.3.3-x86_64 (04b1528b-b1e2-45d4-96d1-fbe04c6b2efd)
key_name	lb_kp1
metadata	{}
name	lb_vml
os-extended-volumes:volumes_attached	[]
progress	0
security_groups	lb_secgroup1
status	BUILD
tenant_id	4b31d0508f83437e83d8f4d520cda22f
updated	2016-06-15T16:53:00Z
user_id	fd471475faa84680b97f18e55847ec0a
```

```
+-----+
      Server building... 100% complete
      Finished
```

Boot second VM.

```
nova boot --flavor 1 --image 04b1528b-b1e2-45d4-96d1-fbe04c6b2efd --key-name lb_
+-----+
| Property          | Value
+-----+
| OS-DCF:diskConfig | MANUAL
| OS-EXT-AZ:availability_zone |
| OS-EXT-SRV-ATTR:host |
| OS-EXT-SRV-ATTR:hypervisor_hostname |
| OS-EXT-SRV-ATTR:instance_name | instance-00000034
| OS-EXT-STS:power_state | 0
| OS-EXT-STS:task_state | scheduling
| OS-EXT-STS:vm_state | building
| OS-SRV-USG:launched_at |
| OS-SRV-USG:terminated_at |
| accessIPv4 |
| accessIPv6 |
| adminPass | 3nFXjNrTrmNm
| config_drive |
| created | 2016-06-15T16:55:10Z
| flavor | m1.tiny (1)
| hostId |
| id | 3844bb10-2c61-4327-a0d4-0c043c674344
| image | cirros-0.3.3-x86_64 (04b1528b-b1e2-45d4
| key_name | lb_kp1
| metadata | {}
| name | lb_vm2
| os-extended-volumes:volumes_attached | []
| progress | 0
| security_groups | lb_secgroup1
| status | BUILD
| tenant_id | 4b31d0508f83437e83d8f4d520cda22f
| updated | 2016-06-15T16:55:09Z
| user_id | fd471475faa84680b97f18e55847ec0a
+-----+
```

```
      Server building... 100% complete
      Finished
```

13.List the running VM with nova list

```
nova list
+-----+-----+-----+-----+-----+
| ID      | Name    | Status | Task State | Power St |
+-----+-----+-----+-----+-----+
| dfdfe15b-ce8d-469c-a9d8-2cea0e7ca287 | lb_vm1  | ACTIVE | -          | Running   |
| 3844bb10-2c61-4327-a0d4-0c043c674344 | lb_vm2  | ACTIVE | -          | Running   |
+-----+-----+-----+-----+-----+
```

14.Check ports.

```
neutron port-list
+-----+-----+-----+
| id | name | mac_address | fixed_ips
+-----+-----+-----+
...
| 7e5e0038-88cf-4f97-a366-b58cd836450e | fa:16:3e:66:fd:2e | {"subnet_id": "ip_address"
| ca95cc24-4e8f-4415-9156-7b519eb36854 | fa:16:3e:e0:37:c4 | {"subnet_id": "ip_address"
|                               |                               |
+-----+-----+-----+
```

15.Create the first floating IP.

```
neutron floatingip-create ext-net --port-id 7e5e0038-88cf-4f97-a366-b58cd836450e
+-----+-----+
| Field | Value
+-----+-----+
| fixed_ip_address | 10.247.94.132
| floating_ip_address | 10.247.96.26
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291
| id | 3ce608bf-8835-4638-871d-0efe8ebf55ef
| port_id | 7e5e0038-88cf-4f97-a366-b58cd836450e
| router_id | 6aafc9a9-93f6-4d7e-94f2-3068b034b823
| status | DOWN
| tenant_id | 4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

16.Create the second floating IP.

```
neutron floatingip-create ext-net --port-id ca95cc24-4e8f-4415-9156-7b519eb36854
+-----+-----+
| Field | Value
+-----+-----+
| fixed_ip_address | 10.247.94.133
| floating_ip_address | 10.247.96.27
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291
| id | 680c0375-a179-47cb-a8c5-02b836247444
| port_id | ca95cc24-4e8f-4415-9156-7b519eb36854
| router_id | 6aafc9a9-93f6-4d7e-94f2-3068b034b823
| status | DOWN
| tenant_id | 4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

17.List the floating IP's.

```
neutron floatingip-list
+-----+-----+-----+
| id | fixed_ip_address | floating_ip_address
+-----+-----+-----+
| 3ce608bf-8835-4638-871d-0efe8ebf55ef | 10.247.94.132 | 10.247.96.26
| 680c0375-a179-47cb-a8c5-02b836247444 | 10.247.94.133 | 10.247.96.27
+-----+-----+-----+
```

18.Show first Floating IP.

```
neutron floatingip-show 3ce608bf-8835-4638-871d-0efe8ebf55ef
+-----+-----+
| Field | Value |
+-----+-----+
fixed_ip_address	10.247.94.132
floating_ip_address	10.247.96.26
floating_network_id	d3cb12a6-a000-4e3e-82c4-ee04aa169291
id	3ce608bf-8835-4638-871d-0efe8ebf55ef
port_id	7e5e0038-88cf-4f97-a366-b58cd836450e
router_id	6aafc9a9-93f6-4d7e-94f2-3068b034b823
status	ACTIVE
tenant_id	4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

19.Show second Floating IP.

```
neutron floatingip-show 680c0375-a179-47cb-a8c5-02b836247444
+-----+-----+
| Field | Value |
+-----+-----+
fixed_ip_address	10.247.94.133
floating_ip_address	10.247.96.27
floating_network_id	d3cb12a6-a000-4e3e-82c4-ee04aa169291
id	680c0375-a179-47cb-a8c5-02b836247444
port_id	ca95cc24-4e8f-4415-9156-7b519eb36854
router_id	6aafc9a9-93f6-4d7e-94f2-3068b034b823
status	ACTIVE
tenant_id	4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

20.Ping first Floating IP.

```
ping -c 1 10.247.96.26
PING 10.247.96.26 (10.247.96.26) 56(84) bytes of data.
64 bytes from 10.247.96.26: icmp_seq=1 ttl=62 time=3.50 ms

--- 10.247.96.26 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.505/3.505/3.505/0.000 ms
```

21.Ping second Floating IP.

```
ping -c 1 10.247.96.27
PING 10.247.96.27 (10.247.96.27) 56(84) bytes of data.
64 bytes from 10.247.96.27: icmp_seq=1 ttl=62 time=3.47 ms

--- 10.247.96.27 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 3.473/3.473/3.473/0.000 ms
```

22.Listing the VM's will give you both the fixed and floating IP's for each virtual machine.

```
nova list
+-----+-----+-----+-----+
| ID | Name | Status | Task State | Power St |
+-----+-----+-----+-----+
```

| dfdfe15b-ce8d-469c-a9d8-2cea0e7ca287 | lb_vm1 | ACTIVE | - |  | Running |
|--------------------------------------|--------|--------|---|--|---------|
| 3844bb10-2c61-4327-a0d4-0c043c674344 | lb_vm2 | ACTIVE | - |  | Running |

23.List Floating IP's.

| neutron floatingip-list              |                     |                     |
|--------------------------------------|---------------------|---------------------|
|                                      |                     |                     |
| id                                   | fixed_ip_address    | floating_ip_address |
| +-----+-----+-----+                  | +-----+-----+-----+ | +-----+-----+-----+ |
| 3ce608bf-8835-4638-871d-0efe8ebf55ef | 10.247.94.132       | 10.247.96.26        |
| 680c0375-a179-47cb-a8c5-02b836247444 | 10.247.94.133       | 10.247.96.27        |
| +-----+-----+-----+                  | +-----+-----+-----+ | +-----+-----+-----+ |

## Create Load Balancers

The following steps will setup new Octavia Load Balancers.

### Note

The following examples assume names and values from the previous section.

1. Create load balancer for subnet

| neutron lbaas-loadbalancer-create --provider octavia --name lb1 6fc2572c-53b3-41 |                                      |
|----------------------------------------------------------------------------------|--------------------------------------|
| Field                                                                            | Value                                |
| admin_state_up                                                                   | True                                 |
| description                                                                      |                                      |
| id                                                                               | 3d9170a1-8605-43e6-9255-e14a8b4aae53 |
| listeners                                                                        |                                      |
| name                                                                             | lb1                                  |
| operating_status                                                                 | OFFLINE                              |
| provider                                                                         | octavia                              |
| provisioning_status                                                              | PENDING_CREATE                       |
| tenant_id                                                                        | 4b31d0508f83437e83d8f4d520cda22f     |
| vip_address                                                                      | 10.247.94.134                        |
| vip_port_id                                                                      | da28aed3-0eb4-4139-afcf-2d8fd3fc3c51 |
| vip_subnet_id                                                                    | 6fc2572c-53b3-41d0-ab63-342d9515f514 |

2. List load balancers. You will need to wait until the load balancer provisioning\_status is ACTIVE before proceeding to the next step.

| neutron lbaas-loadbalancer-list      |                           |                           |                           |
|--------------------------------------|---------------------------|---------------------------|---------------------------|
|                                      |                           |                           |                           |
| id                                   | name                      | vip_address               | provisioning_st           |
| +-----+-----+-----+-----+            | +-----+-----+-----+-----+ | +-----+-----+-----+-----+ | +-----+-----+-----+-----+ |
| 3d9170a1-8605-43e6-9255-e14a8b4aae53 | lb1                       | 10.247.94.134             | ACTIVE                    |
| +-----+-----+-----+-----+            | +-----+-----+-----+-----+ | +-----+-----+-----+-----+ | +-----+-----+-----+-----+ |

3. Once the load balancer is created, create the listener. This may take some time.

```
neutron lbaas-listener-create --loadbalancer lbl --protocol HTTP --protocol-port
+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | True |
| connection_limit | -1 |
| default_pool_id |
| default_tls_container_ref |
| description |
id	c723b5c8-e2df-48d5-a54c-fc240ac7b539
loadbalancers	{"id": "3d9170a1-8605-43e6-9255-e14a8b4aae53"}
name	lbl_listener
protocol	HTTP
protocol_port	80
sni_container_refs	
tenant_id	4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

4. Create the load balancing pool. During the creation of the load balancing pool, the status for the load balancer goes to PENDING\_UPDATE. Use `neutron lbaas-loadbalancer-list` to watch for the change to ACTIVE. Once the load balancer returns to ACTIVE, proceed with the next step.

```
neutron lbaas-pool-create --lb-algorithm ROUND_ROBIN --listener lbl_listener --p
+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | True |
| description |
| healthmonitor_id |
id	0f5951ee-c2a0-4e62-ae44-e1491a8988e1
lb_algorithm	ROUND_ROBIN
listeners	{"id": "c723b5c8-e2df-48d5-a54c-fc240ac7b539"}
members	
name	lbl_pool
protocol	HTTP
session_persistence	
tenant_id	4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

5. Create first member of the load balancing pool.

```
neutron lbaas-member-create --subnet 6fc2572c-53b3-41d0-ab63-342d9515f514 --addr
+-----+-----+
| Field | Value |
+-----+-----+
address	10.247.94.132
admin_state_up	True
id	61da1e21-e0ae-4158-935a-c909a81470e1
protocol_port	80
subnet_id	6fc2572c-53b3-41d0-ab63-342d9515f514
tenant_id	4b31d0508f83437e83d8f4d520cda22f
weight	1
+-----+-----+
```

#### 6. Create the second member.

```
neutron lbaas-member-create --subnet 6fc2572c-53b3-41d0-ab63-342d9515f514 --addr
+-----+-----+
| Field      | Value
+-----+-----+
| address    | 10.247.94.133
| admin_state_up | True
| id         | 459c7f21-46f7-49e8-9d10-dc7da09f8d5a
| protocol_port | 80
| subnet_id   | 6fc2572c-53b3-41d0-ab63-342d9515f514
| tenant_id   | 4b31d0508f83437e83d8f4d520cda22f
| weight      | 1
+-----+-----+
```

7. You should check to make sure the load balancer is active and check the pool members.

```
neutron lbaas-loadbalancer-list
+-----+-----+-----+
| id | name | vip_address | provisioning_sta
+-----+-----+-----+
| 3d9170a1-8605-43e6-9255-e14a8b4aae53 | lb1 | 10.247.94.134 | ACTIVE
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```
neutron lbaas-member-list lb1_pool
+-----+-----+-----+-----+
| id | address | protocol_port | weight
+-----+-----+-----+-----+
| 61dale21-e0ae-4158-935a-c909a81470e1 | 10.247.94.132 | 80 | 1
| 459c7f21-46f7-49e8-9d10-dc7da09f8d5a | 10.247.94.133 | 80 | 1
+-----+-----+-----+-----+-----+-----+-----+-----+
```

8. You can view the details of the load balancer, listener and pool.

```
neutron lbaas-loadbalancer-show 3d9170a1-8605-43e6-9255-e14a8b4aae53
+-----+-----+
| Field | Value |
+-----+-----+
| admin_state_up | True
| description |
| id | 3d9170a1-8605-43e6-9255-e14a8b4aae53
| listeners | {"id": "c723b5c8-e2df-48d5-a54c-fc240ac7b539"}
| name | lb1
| operating_status | ONLINE
| provider | octavia
| provisioning_status | ACTIVE
| tenant_id | 4b31d0508f83437e83d8f4d520cda22f
| vip_address | 10.247.94.134
| vip_port_id | da28aed3-0eb4-4139-afcf-2d8fd3fc3c51
| vip_subnet_id | 6fc2572c-53b3-41d0-ab63-342d9515f514
+-----+
```

```
| id | default_pool_id |
+-----+
| c723b5c8-e2df-48d5-a54c-fc240ac7b539 | 0f5951ee-c2a0-4e62-ae44-e1491a8988e1 |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

neutron lbaas-pool-list

```
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | name | protocol | admin_state_up |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 0f5951ee-c2a0-4e62-ae44-e1491a8988e1 | lb1_pool | HTTP | True |
+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
```

# Create Floating IPs for Load Balancer

To create the floating IP's for the load balancer, you will need to list the current ports to get the load balancer id. Once you have the id, you can then create the floating IP.

1. List the current ports.

| neutron port-list                    |                                                       |
|--------------------------------------|-------------------------------------------------------|
| id                                   | name                                                  |
| 7e5e0038-88cf-4f97-a366-b58cd836450e |                                                       |
| a3d0b0fe-e7ff-4b00-a033-44e833b55efe |                                                       |
| ca95cc24-4e8f-4415-9156-7b519eb36854 |                                                       |
| da28aed3-0eb4-4139-afcf-2d8fd3fc3c51 | loadbalancer-<br>3d9170a1-8605-43e6-9255-e14a8b4aae53 |

- ## 2. Create the floating IP for the load balancer.

```
neutron floatingip-create ext-net --port-id da28aed3-0eb4-4139-afcf-2d8fd3fc3c51
   Created a new floatingip:
+-----+-----+
| Field          | Value           |
+-----+-----+
fixed_ip_address	10.247.94.134
floating_ip_address	10.247.96.28
floating_network_id	d3cb12a6-a000-4e3e-82c4-ee04aa169291
id	9a3629bd-b0a6-474c-abe9-89c6ecb2b22c
port_id	da28aed3-0eb4-4139-afcf-2d8fd3fc3c51
router_id	6aafc9a9-93f6-4d7e-94f2-3068b034b823
status	DOWN
tenant_id	4b31d0508f83437e83d8f4d520cda22f
+-----+-----+
```

# Testing the Octavia Load Balancer

To test the load balancers, create the following web server script so you can run it on each virtual machine. You will use curl <ip address> to test if the load balance services are responding properly.

1. Start running web servers on both of the virtual machines. Create the webserver.sh script with below contents. In this example, the port is 80.

```
vi webserver.sh

#!/bin/bash

MYIP=$( /sbin/ifconfig eth0 | grep 'inet addr' | awk -F: '{print $2}' | awk '{print $1}' )
while true; do
    echo -e "HTTP/1.0 200 OK

Welcome to $MYIP" | sudo nc -l -p 80
done
```

2. Deploy the web server and run it on the first virtual machine.

```
ssh-keygen -R 10.247.96.26
/home/stack/.ssh/known_hosts updated.
Original contents retained as /home/stack/.ssh/known_hosts.old

scp -o StrictHostKeyChecking=no -i lb_kp1.pem webserver.sh cirros@10.247.96.26:
webserver.sh   100%   263      0.3KB/s   00:00

ssh -o StrictHostKeyChecking=no -i lb_kp1.pem cirros@10.247.96.26 'chmod +x ./we
ssh -o StrictHostKeyChecking=no -i lb_kp1.pem cirros@10.247.96.26 ./webserver.sh
```

3. Test the first web server.

```
curl 10.247.96.26
Welcome to 10.247.94.132
```

4. Deploy and start the web server on the second virtual machine like you did in the previous steps. Once the second web server is running, list the floating IP's.

```
neutron floatingip-list
+-----+-----+
| id          | fixed_ip_address | floating_ip_address |
+-----+-----+
| 3ce608bf-8835-4638-871d-0efe8ebf55ef | 10.247.94.132    | 10.247.96.26
| 680c0375-a179-47cb-a8c5-02b836247444 | 10.247.94.133    | 10.247.96.27
| 9a3629bd-b0a6-474c-abe9-89c6ecb2b22c | 10.247.94.134    | 10.247.96.28
+-----+-----+
```

5. Display the floating IP for the load balancer.

```
neutron floatingip-show 9a3629bd-b0a6-474c-abe9-89c6ecb2b22c
+-----+
| Field          | Value           |
+-----+
| fixed_ip_address | 10.247.94.134 |
```

|                     |                                      |
|---------------------|--------------------------------------|
| floating_ip_address | 10.247.96.28                         |
| floating_network_id | d3cb12a6-a000-4e3e-82c4-ee04aa169291 |
| id                  | 9a3629bd-b0a6-474c-abe9-89c6ecb2b22c |
| port_id             | da28aed3-0eb4-4139-afcf-2d8fd3fc3c51 |
| router_id           | 6aafc9a9-93f6-4d7e-94f2-3068b034b823 |
| status              | ACTIVE                               |
| tenant_id           | 4b31d0508f83437e83d8f4d520cda22f     |

- 
- Finally, test the load balancing.

```
curl 10.247.96.28
Welcome to 10.247.94.132

curl 10.247.96.28
Welcome to 10.247.94.133

curl 10.247.96.28
Welcome to 10.247.94.132

curl 10.247.96.28
Welcome to 10.247.94.133

curl 10.247.96.28
Welcome to 10.247.94.132

curl 10.247.96.28
Welcome to 10.247.94.133
```

---

# Chapter 28. Other Common Post-Installation Tasks

## Determining Your User Credentials

On your lifecycle manager, in the `~/scratch/ansible/next/hos/ansible/group_vars/` directory you will find several files. In the one labeled as first control plane node you can locate the user credentials for both the Administrator user (`admin`) and your Demo user (`demo`) which you will use to perform many other actions on your cloud.

For example, if you are using the Entry-scale KVM with VSA model and used the default naming scheme given in the example configuration files, you can use these commands on your lifecycle manager to GREP for your user credentials:

### **Administrator**

```
grep keystone_admin_pwd entry-scale-kvm-vsa-control-plane-1
```

### **Demo**

```
grep keystone_demo_pwd entry-scale-kvm-vsa-control-plane-1
```

## Configure your lifecycle manager to use the command-line tools

This playbook will do a series of steps to update your environment variables for your cloud so you can use command-line clients.

Run the following command, which will replace `/etc/hosts` on the lifecycle manager:

```
cd ~/scratch/ansible/next/hos/ansible  
ansible-playbook -i hosts/verb_hosts cloud-client-setup.yml
```

As the `/etc/hosts` file no longer has entries for Helion lifecycle management, sudo commands may become a bit slower. To fix this issue, once this step is complete, add "hlm" after "127.0.0.1 localhost". The result will look like this:

```
...  
# Localhost Information  
127.0.0.1 localhost hlm
```

## Protect home directory

The home directory of the user that owns the HPE Helion OpenStack 5.0 scripts should not be world readable. Change the permissions so that they are only readable by the owner:

```
chmod 0700 ~
```

## Back up Your SSH Keys

As part of the cloud deployment setup process, SSH keys to access the systems are generated and stored in `~/ .ssh` on your lifecycle manager.

These SSH keys allow access to the subsequently deployed systems and should be included in the list of content to be archived in any backup strategy.

## Retrieving Service Endpoints

1. Log in to your lifecycle manager.
2. Source the keystone admin credentials:

```
unset OS_TENANT_NAME  
source ~/keystone.osrc
```

3. Using the OpenStack command-line tool you can then query the Keystone service for your endpoints:

```
openstack endpoint list
```

### Tip

You can use `openstack -h` to access the client help file and a full list of commands.

To learn more about Keystone, see [Keystone Documentation](#).

## Other Common Post-Installation Tasks

Here are the links to other common post-installation tasks that either the Administrator or Demo users can perform:

- 
- the section called “Creating an External Network”
- 
- 
-