Contents

Executive Summary

OpenStack on Cisco UCS

OpenStack with Cisco ACI: OpFlex Architecture
  - Why Cisco ACI and OpenStack
  - OpFlex Modular Layer 2 Plug-in Architecture
  - OpFlex Agent
  - Distributed Neutron Services Using OpFlex ML2
  - OpFlex NAT Functions
  - OpenStack Versus Cisco ACI Constructs

Reference Architecture
  - Physical Topology
  - Cabling Diagram

Deployment Hardware and Software
  - Prerequisites
  - Required VLANs
  - Logical OpenStack Topology

Configuring Cisco UCS
  - Service Profile and vNIC Templates
  - Server Pools
  - Server vNICs and Placement
  - Maximum Transmission Unit and Quality-of-Service Configuration
  - Port-Channel Configuration

Configuring Cisco ACI
  - Perform Cisco APIC Initial Configuration
  - Convert from Cisco NX-OS to Cisco ACI Boot Mode
  - Discover Cisco ACI Fabric
  - Upgrade Cisco APIC and Switch Software
  - Configure Fabric Access Policies
  - Configure vPC for the Fabric Interconnect
  - Configure vPC for the OOB Switch
  - Validate the vPC Configuration
  - Deploy Infrastructure Tenant
  - Create a Bridge Domain, EPG, and VRF Instance in Infrastructure Tenant

Installing Red Hat OpenStack Cloud Platform
  - Red Hat OpenStack Cloud Platform
  - Red Hat OpenStack Deployment Summary
  - Installing the OpFlex ML2 Plug-in

Setting Up the OpenStack External Network
  - Prerequisites
  - Configure L3-Out in Cisco ACI
  - Configure L3-Out in OpenStack Controller
  - Create Route Reflector in APIC

Conclusion

For More Information
Appendix A: Layer 3 Outside Configuration Using XML
L3-Out Configuration Using XML

Appendix B: Overcloud Deployment .yaml files
network-environment.yaml
controller.yaml
compute.yaml
The openstack overcloud deploy command
Executive Summary

Next-generation data centers are transitioning to cloud computing models. As companies make this transition, they are seeking appropriate and trusted solutions for creating scalable, flexible, and secure cloud environments. In addition, they want an overall solution that addresses total cost of ownership (TCO), management capabilities, ease of use, and the application services that their users demand.

However, not all cloud solutions are the same or a good fit for all. As a result, many customers have been turning to OpenStack to deploy their new cloud environments. More than 125 industry leaders have contributed to OpenStack code.

In addition, because all cloud deployments must be properly supported from the foundation, choosing the correct hardware is critical to helping ensure predictable performance, reliability, and long-term success. When today’s clouds run business-critical applications, failure is not an option, so solutions from the market leaders, Cisco and Red Hat, are the obvious choice.

This technical white paper describes the reference architecture for implementing Cisco® Application Centric Infrastructure (Cisco ACI™) and Red Hat OpenStack Platform on the Cisco Unified Computing System™ (Cisco UCS®). The solution is implemented using Cisco blade servers and next-generation fabric interconnects to reap the benefits of Cisco’s simplified and programmable infrastructure. The document serves as a design and deployment guide and provides step-by-step instructions for configuring Cisco UCS hardware, installing Red Hat OpenStack Platform, and installing and integrating Cisco ACI with OpenStack. It also provides an overview of the OpenStack data center setup, end-to-end system architecture, and other miscellaneous features such as Cisco ACI external Layer 3 (L3-Out) network configuration.

Cisco UCS Integrated Infrastructure and Cisco ACI on Red Hat OpenStack Platform constitutes an all-in-one solution for deploying a Cisco ACI and Red Hat OpenStack private cloud using Cisco infrastructure. The solution is validated and supported by Cisco and Red Hat to increase the speed of infrastructure deployment and reduce the risk of scaling from a proof-of-concept environment to a full enterprise production environment.

In this reference architecture, the Cisco open-source plug-in for OpenStack Neutron allows OpenStack tenants to transparently configure and manage a network based on Cisco ACI. This plug-in automatically translates and maps OpenStack Neutron API commands for networks, subnets, routers, and so on into an application network profile.

OpenStack on Cisco UCS

Cloud-enabled applications can run on organization premises, in public clouds, or on a combination of the two (hybrid cloud) for greater flexibility and business agility. Finding a platform that supports all these scenarios is essential. With Cisco UCS, IT departments can take advantage of technological advancements and lower the cost of their OpenStack deployments.

- **Open architecture:** A market-leading, open alternative to expensive, proprietary environments, the simplified architecture of Cisco UCS running OpenStack software delivers greater scalability, manageability, and performance at a significant cost savings compared to traditional systems, both in the data center and the cloud. Using industry-standard x86-architecture servers and open-source software, IT departments can deploy cloud infrastructure today without concern for hardware or software vendor lock-in.

- **Accelerated cloud provisioning:** Cloud infrastructure must be able to flex on demand, providing infrastructure to applications and services on a moment's notice. Cisco UCS simplifies and accelerates cloud infrastructure deployment through automated configuration. The abstraction of Cisco UCS Integrated Infrastructure for Red Hat Enterprise Linux (RHEL) server identity, personality, and I/O connectivity from the
hardware allows these characteristics to be applied on demand. Every aspect of a server’s configuration, from firmware revisions and BIOS settings to network profiles, can be assigned through Cisco UCS service profiles. Cisco service profile templates establish policy-based configuration for server, network, and storage resources and can be used to logically preconfigure these resources even before they are deployed in the cloud infrastructure.

- **Simplicity at scale:** With IT departments challenged to deliver more applications and services in shorter time frames, the architectural silos that result from an impromptu approach to capacity scaling with traditional systems poses a barrier to successful cloud infrastructure deployment. Start with the computing and storage infrastructure needed today and then scale easily by adding components. Because servers and storage systems integrate into the unified system, they do not require additional supporting infrastructure or expert knowledge. The system simply, quickly, and cost effectively presents more computing power and storage capacity to cloud infrastructure and applications.

- **Virtual infrastructure density:** Cisco UCS enables cloud infrastructure to meet ever-increasing guest OS memory demands on fewer physical servers. The system’s high-density design increases consolidation ratios for servers, saving the capital, operating, physical space, and licensing costs that would be needed to run virtualization software on larger servers. With Cisco UCS B200 M4 Blade Servers using the latest Intel® Xeon® processor E5-2600 v3 series with up to 1536 GB of RAM (using 64-GB DIMMs), OpenStack deployments can host more applications using less-expensive servers without sacrificing performance.

- **Simplified networking:** In OpenStack environments, underlying infrastructure can become a sprawling complex of networked systems. Unlike traditional server architecture, Cisco UCS provides greater network density with less cabling and complexity. Cisco’s unified fabric integrates Cisco UCS servers with a single high-bandwidth, low-latency network that supports all system I/O. This approach simplifies the architecture and reduces the number of I/O interfaces, cables, and access-layer switch ports compared to the requirements for traditional cloud infrastructure deployments. This unification can reduce network complexity by up to a factor of three, and the system’s wire-once network infrastructure increases agility and accelerates deployment with zero-touch configuration.

- **Installation confidence:** Organizations that choose OpenStack for their cloud can take advantage of the Red Hat OpenStack Platform Director. This software performs the work needed to install a validated OpenStack deployment. Unlike other solutions, this approach provides a highly available, highly scalable architecture for OpenStack services.

- **Easy management:** Cloud infrastructure can be extensive, so it must be easy and cost effective to manage. Cisco UCS Manager provides embedded management of all software and hardware components in Cisco UCS. Cisco UCS Manager resides as embedded software on the Cisco UCS fabric interconnects, fabric extenders, servers, and adapters. No external management server is required, simplifying administration and reducing capital expenses for the management environment.
OpenStack with Cisco ACI: OpFlex Architecture

Why Cisco ACI and OpenStack
Cisco ACI offers programmable switching fabric with a robust set of APIs. Combined with Cisco UCS, the solution offers end-to-end infrastructure programmability. Programmability is the most important requirement for cloud automation and orchestration for service delivery. The Cisco ACI networking fabric in the OpenStack environment offers the following main benefits:

- **Policy-based infrastructure**: All applications have requirements from the network. Cisco ACI couples these requirements with the policies that facilitate communication between the application and the outside world. The APIC automatically renders policies in the fabric infrastructure. This mapping of application requirements with the network through policies accelerates application deployment. Furthermore, if the application requirements change in the future, these polices can be altered or even replaced with ease. OpenStack itself is a combination of Web Server Gateway Interface (WSGI) applications that are network centric and based on VLANs, subnets, and access control lists (ACLs).

- **Physical and virtual integration**: Cisco ACI handles all physical and virtual networking that OpenStack requires.

- **Service chaining**: Cisco ACI offers native service-chaining capabilities to transparently insert or remove services between two endpoints. Cisco ACI can be configured using the APIs of service-layer appliances such as firewalls, load balancers, and other Layer 4 through Layer (L4-L7) devices. This feature allows both tenants and cloud administrators to deploy applications that have complex security policy requirements.

- **Telemetry**: Cisco ACI offers real-time visibility and telemetry information. The APIC presents detailed data about the performance of individual endpoint groups (EPGs) and tenants in the network. This information includes real-time health metrics for the physical and virtual networks. This telemetry information helps reduce troubleshooting time significantly. Faults can be traced from virtual to physical connections down to the virtual machine. Cisco ACI networks can be debugged efficiently, and actions can be orchestrated based on the events triggered by the faults.

- **Single point of management**: In Cisco ACI infrastructure, the APIC cluster is the central point for management and provisioning using the GUI or the representational state transfer (REST) API.

- **Secure multitenancy**: In Cisco ACI, a tenant is generally a logical security construct that keeps resources isolated. Each tenant maintains its own network containers called contexts or Virtual Routing and Forwarding (VRF) instances. Theses contexts have their own address spaces, VLANs, subnets, and L4-L7 services.

- **Automation**: Cisco ACI offers a robust programmable REST API, which can be used to automate repetitive tasks to reduce errors, thus increasing network agility.

- **Fabric discovery**: Cisco ACI discovers leaf and spine switches within the fabric as soon as they are racked and cabled. This feature significantly enhances ease of scalability.

- **Integrated overlay and underlay**: Cisco ACI offers a fully managed underlay network for the Cisco ACI fabric through the APIC. It provides the capability to connect physical servers and multiple hypervisors to overlay networks.
OpFlex Modular Layer 2 Plug-in Architecture

The OpFlex™ Modular Layer 2 (ML2) framework in OpenStack allows integration of networking services based on type drivers and mechanism drivers. Common networking type drivers include local, flat, VLAN, and Virtual Extensible LAN (VXLAN). OpFlex is added as a new network type through ML2, with an actual packet encapsulation of either VXLAN or VLAN on the host defined in the OpFlex configuration. A mechanism driver is enabled to communicate networking requirements from the Neutron servers to the APIC cluster. The APIC mechanism driver translates Neutron networking elements such as a network (segment), subnet, router, or external network into APIC constructs within the Cisco ACI policy model.

Furthermore, the OpFlex ML2 software stack uses a modified Open vSwitch (OVS) package and local software agents (mechanisms) on each OpenStack computing host that communicates with the Neutron servers and OVS. An OpFlex proxy from the Cisco ACI leaf switch exchanges policy information with the agent-ovs instance in each computing host, effectively extending the Cisco ACI switch fabric and policy model into the virtual switch. This extension results in a cohesive system that can apply networking policy anywhere in the combined virtual and physical switching fabrics, starting from the virtual port at which a virtual machine instance attaches to the network. Figure 1 illustrates the interaction of the OpFlex ML2 APIC driver and the Cisco ACI fabric, and the extension of the OpFlex proxy to the agent-ovs service on the computing host.

Figure 1. OpFlex ML2 Plug-in: OpenStack with Cisco ACI Architecture
OpFlex Agent
On the computing node, the neutron-opflex-agent service receives information about OpenStack endpoints from the ML2 driver software on the Neutron server. This information is stored locally in the endpoint files located in /var/lib/opflex-agent-ovs/endpoints. The agent-ovs service uses the endpoint information to resolve policy for the endpoints through the OpFlex proxy on the connected Cisco ACI leaf switch. The agent-ovs service then programs policy on the OVS using OpenFlow for policies that can be enforced locally. Nonlocal policies are enforced on the upstream leaf switch. Figure 2 shows the interaction between OVS and the OpFlex modules running on the computing node.

Figure 2. Compute Hosts: OpFlex Agent Architecture

Distributed Neutron Services Using OpFlex ML2
In the OpenStack cloud environment, Neutron defines common networking constructs and services required by virtual machine instances. Availability of Neutron has improved considerably over several OpenStack releases. However, both the availability and scalability of Neutron services are still a concern when implementing all these functions on a single server or on a cluster of servers. The OpFlex ML2 driver software stack provides the capability to distribute these network services across the computing nodes in the cluster, using a scale-out approach to reduce the load on any single instance of a service while increasing overall service availability.

The following OpenStack Neutron services can be distributed to computing nodes when using the OpFlex ML2 driver software:

- **Network Address Translation (NAT) for external networks**: The OpFlex ML2 driver supports external networks by distributing Source NAT (SNAT) and floating IP functions for OpenStack to the Open vSwitch of the computing hosts. Packets destined for IP addresses not defined in the private OpenStack tenant space are automatically translated before they egress a computing host. The translated packets are then routed to the external routed network defined in the APIC. Distributed NAT services are inherent in the solution.

- **Layer 3 forwarding**: The Neutron reference software implementation of the Layer 3 agent is replaced by a combination of Layer 3 forwarding in the Cisco ACI fabric and local forwarding in the computing node. If two virtual machines connected to the same OpenStack tenant router reside on the same computing node,
Layer 3 traffic between them will be forwarded by OVS and remain local to that physical server. Distributed Layer 3 forwarding for traffic local to a computing node is inherent in the solution.

- **Dynamic Host Configuration Protocol (DHCP):** Neutron software reference implementations have a DHCP agent service centralized on the Neutron servers. The OpFlex ML2 driver software allows a distributed DHCP approach using the agent-ovs service. Distributing the DHCP function across the computing nodes keeps DHCP discovery, offer, request, and acknowledge (DORA) traffic local to the host and helps ensure reliable allocation of IP addresses to virtual machine instances. Centralized Neutron address management functions communicate DHCP addresses and options to the local agent-ovs service over the management network. This optimized DHCP approach is enabled by default in the solution, but if you want, you can revert to the traditional centralized mode.

- **Metadata proxy:** OpenStack virtual machines can receive instance-specific information such as instance IDs, host names, and Secure Shell (SSH) keys from the Nova metadata service. This service is normally reached through a Neutron service acting as a proxy on behalf of OpenStack virtual machine instances. The OpFlex ML2 software allows this proxy function to be distributed to each of the computing hosts. This optimized metadata proxy function is disabled by default, and either a traditional centralized or a distributed approach can be configured.

The logical topology diagram in the Figure 3 illustrates the connections to OpenStack network segments from Neutron servers and computing hosts, including the distributed Neutron services.

**Figure 3.** Logical OpenStack Network Connectivity with Distributed Neutron Services
OpFlex NAT Functions
OpFlex NAT consists of two functions:

- **Floating IP**: Maps a private IP address to an externally routable IP address
- **Source NAT**: Allows external communication without a designated routable IP address

Floating IP Function
The floating IP function in OpenStack is used when a virtual machine instance is allowed to claim a distinct static NAT address to support inbound connections to the virtual machine from outside the cloud. The floating IP subnet is the subnet assigned within OpenStack to the Neutron external network entity. Traffic egressing the cloud will carry a source IP address of either the SNAT subnet or floating subnet. The routing hops external to Cisco ACI need to have routes back to these subnets, either through a dynamic routing protocol or through static configuration, to allow return traffic to find its way back to OpenStack. Figure 4 shows the workflow of the floating IP function.

**Figure 4.** Floating IP Function Workflow

Source NAT Function
SNAT is a feature of the Cisco ACI platform’s OpFlex plugin. It gives hosts access to the outside world though an external routed network preconfigured in Cisco ACI. SNAT differs from the floating IP function in that OpenStack OpFlex assigns a public IP address to all computing hosts. The traffic from each guest virtual machine to the external network is translated directly to the OpenStack hypervisor through the Open vSwitch controlled by the OVS agent. As the traffic is translated, it is placed in a separate overlay in Cisco ACI that can route through the external L3-Out construct. Figure 5 shows the SNAT function workflow.
OpenStack Versus Cisco ACI Constructs
The OpFlex ML2 driver translates the Neutron networking requirements into the necessary Cisco ACI policy model constructs to achieve the desired connectivity. Table 1 lists the OpenStack Neutron constructs and the corresponding APIC policy objects that will be configured when they are created.

Table 1. Cisco ACI and OpenStack Mappings

<table>
<thead>
<tr>
<th>OpenStack Object</th>
<th>Cisco APIC Object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neutron instance</td>
<td>Cisco ACI tenant and virtual machine manager (VMM) domain</td>
</tr>
<tr>
<td>Tenant or project</td>
<td>Application profile or separate Cisco ACI tenant</td>
</tr>
<tr>
<td>Tenant network</td>
<td>EPG plus bridge domain</td>
</tr>
<tr>
<td>Subnet</td>
<td>Subnet</td>
</tr>
<tr>
<td>Security groups and rules</td>
<td>Not applicable (Linux iptables rules are maintained for each host)</td>
</tr>
<tr>
<td>Router</td>
<td>Contract plus EPG plus bridge domain</td>
</tr>
<tr>
<td>External network</td>
<td>L3-Out and outside EPG</td>
</tr>
</tbody>
</table>
Reference Architecture

Physical Topology

Figure 6 shows the lab topology and hardware components used in the reference infrastructure described in this document.

Figure 6. Physical Topology

[Diagram of Cisco Integrated Red Hat Enterprise Linux OpenStack Platform with Cisco ACI]
Hardware Components and Bill of Materials

Table 2 shows the hardware components required for this reference architecture.

<table>
<thead>
<tr>
<th>Component and Role</th>
<th>Part or Model Number</th>
<th>Quantity</th>
<th>Comments</th>
</tr>
</thead>
</table>
| Red Hat OpenStack Platform Director node | Cisco UCS B200 M4 Blade Server | 1        | • CPU: 2 x Intel Xeon processor E5-2630 v3  
  • Memory: 8 x 16-GB 2133-MHz DIMMs; total of 128 GB  
  • Local disk: 2 x 300-GB SAS drives for boot  
  • Network interface card (NIC): 1 x Cisco UCS Virtual Interface Card (VIC) 1340  
  • RAID controller: Cisco MegaRAID 12-GB SAS controller |
| OpenStack controller node                 | Cisco UCS B200 M4 Blade Server | 3        | • CPU: 2 x Intel Xeon processor E5-2630 v3  
  • Memory: 8 x 16-GB 2133-MHz DIMMs; total of 128 GB  
  • Local disk: 2 x 300-GB SAS drives for boot  
  • NIC: 1 x Cisco UCS VIC 1340  
  • RAID controller: Cisco MegaRAID 12-GB SAS controller |
| OpenStack computing node                  | Cisco UCS B200 M4 Blade Server | 4        | • CPU: 2 x Intel Xeon processor E5-2660 v3  
  • Memory: 16 x 16-GB 2133-MHz DIMMs; total of 256 GB  
  • Local disk: 2 x 300-GB SAS drives for boot  
  • NIC: 1 x Cisco UCS VIC 1340  
  • RAID controller: Cisco MegaRAID 12-GB SAS controller |
| Ceph storage node                         | Cisco UCS C240 Rack Server   | 3        | • CPU: 2 x Intel Xeon processor E5-2630 v3  
  • Memory: 8 x 16-GB 2133-MHz DIMMs; total of 128 GB  
  • Internal hard-disk drive (HDD): None  
  • Ceph object storage device (OSD): 8 x 6-TB SAS drives  
  • Ceph journal: 2 x 400-GB solid-state disk (SSD) drives  
  • OS boot drive: 2 x 1-TB SAS drives  
  • NIC: 1 x Cisco UCS VIC 1227  
  • RAID controller: Cisco MegaRAID 12-GB SAS controller |
| Chassis                                   | Cisco UCS 5108 Blade Server Chassis | 2        |                                                            |
| I/O module (IOM)                          | Cisco UCS 2104XP Fabric Extender | 4        | 2 per chassis                                             |
| Fabric interconnect                       | Cisco UCS 6248UP 48-Port Fabric Interconnect | 2        |                                                            |
| Cisco ACI leaf switch                     | Cisco Nexus 9372PX Switch     | 2        |                                                            |
| Cisco ACI spine switch                    | Cisco Nexus 9336PQ ACI Spine Switch | 2        |                                                            |
| Cisco APIC                                | Cisco UCS C220 Rack Server APIC appliance | 1        |                                                            |
Hardware and Software Revisions

Tables 3 and 4 list the hardware and software versions tested in the reference architecture described in this document.

Table 3. Hardware Firmware Versions

<table>
<thead>
<tr>
<th>Layer</th>
<th>Production Hardware</th>
<th>Firmware Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing</td>
<td>Cisco UCS 6248UP fabric interconnect</td>
<td>2.2(5)</td>
</tr>
<tr>
<td></td>
<td>Cisco UCS B200 M4 server</td>
<td>2.2(5)</td>
</tr>
<tr>
<td></td>
<td>Cisco UCS Manager</td>
<td>2.2(5)</td>
</tr>
<tr>
<td></td>
<td>Cisco Ethernet NIC (eNIC) driver</td>
<td>2.1.1.93</td>
</tr>
<tr>
<td>Network</td>
<td>Cisco Nexus 9372PX leaf switch</td>
<td>11.3 (1g)</td>
</tr>
<tr>
<td></td>
<td>Cisco Nexus 9336PQ spine switch</td>
<td>11.3 (1g)</td>
</tr>
<tr>
<td></td>
<td>Cisco APIC server</td>
<td>1.3 (1g)</td>
</tr>
<tr>
<td>Storage</td>
<td>Cisco UCS C240 M4 server</td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Software Versions

<table>
<thead>
<tr>
<th>Layer</th>
<th>Production Hardware</th>
<th>Firmware Release</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating system</td>
<td>RHEL</td>
<td>7.2</td>
</tr>
<tr>
<td>OpenStack platform</td>
<td>RHEL OpenStack Platform</td>
<td>8 (Liberty based)</td>
</tr>
<tr>
<td></td>
<td>RHEL OpenStack Platform director</td>
<td>8</td>
</tr>
<tr>
<td>Storage</td>
<td>Red Hat Ceph Storage</td>
<td>1.3</td>
</tr>
<tr>
<td>Networking</td>
<td>Cisco ACI platform OpFlex plug-in</td>
<td>1.3.1-Liberty</td>
</tr>
</tbody>
</table>

Cabling Diagram

Figure 7 show the cabling diagram of the lab test setup.
Figure 7. Cabling Diagram

Note: For simplicity, in this reference environment Leaf 101 is configured to act as a border leaf, and an interface is configured in Cisco ACI for the external routed network for L3-Out. However, a good practice is to have redundant paths to the upstream existing external network.
Deployment Hardware and Software

Prerequisites
Before deploying the architecture described in this document, be sure that your system has the following:

- Active Red Hat subscription for registering RHEL hosts and subscribing to the required repositories
- Network Time Protocol (NTP) server: This server is essential for a Red Hat OpenStack Platform installation. The NTP server must be accessible from the RHEL hosts.
- Repository server: This server can be an RHEL server or any server accessible from the management network of the OpenStack server. This server is needed to stage the repository for the yum-based installation of Cisco OpFlex software.
- RHEL 7.2 ISO image
- Access to Cisco software downloads for firmware packages and the Cisco OpFlex plug-in

Required VLANs
Table 5 lists the VLAN requirements for this deployment and can be used as a worksheet for planning and designing this architecture.

Table 5: VLANs Used in the Reference Architecture

<table>
<thead>
<tr>
<th>VLAN Name</th>
<th>VLAN Purpose</th>
<th>VLAN ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>PXE</td>
<td>VLAN for preboot execution environment (PXE) provisioning network</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Used by the Red Hat OpenStack Platform director server to deploy RHEL 7.2 and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>OpenStack Platform</td>
<td></td>
</tr>
<tr>
<td>Management</td>
<td>VLAN for management network</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Used for managing all the devices in the infrastructure, including Linux</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kernel-based Virtual Machine (KVM) to the hosts</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Also used for OpenStack internal API network</td>
<td></td>
</tr>
<tr>
<td>Storage public</td>
<td>VLAN for Ceph client to send requests to Ceph OSD daemons</td>
<td>30</td>
</tr>
<tr>
<td>Storage cluster</td>
<td>VLAN for Ceph storage cluster for OSD replication and heartbeat traffic from</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>public network</td>
<td></td>
</tr>
<tr>
<td>External</td>
<td>VLAN for public network</td>
<td>219</td>
</tr>
<tr>
<td></td>
<td>Provides access to outside world</td>
<td></td>
</tr>
<tr>
<td>Tenant data</td>
<td>VLAN for tenant private networks for virtual machine data traffic</td>
<td>250 – 750</td>
</tr>
<tr>
<td>ACI infra</td>
<td>VLAN for Cisco ACI fabric in-band network infrastructure</td>
<td>3967</td>
</tr>
<tr>
<td></td>
<td>VLAN ID provided at the time of APIC setup; default ID is 4093</td>
<td></td>
</tr>
</tbody>
</table>

**Note:** This design does not use a Cisco UCS Manager plug-in to automatically provision tenant VLANs. Therefore, the entire tenant VLAN range, which is 250 to 750 in this case, has been preprovisioned in Cisco UCS Manager and added in the tenant data virtual NIC (vNIC).

Logical OpenStack Topology
Network data is logically segmented using VLANs. The entire range of VLANs, shown in Table 5, is configured ahead of time in Cisco UCS.

Figure 8 shows the logical topology of the director, controller, computing, and Ceph storage nodes.
Figure 8. Logical Topology
• **PXE:** This network is the provisioning network. It is common in all hosts. This network is used by Red Hat OpenStack Platform to build controller, computing, and storage nodes. The Red Hat OpenStack Platform director uses this network to boot hosts using PXE, deploy hosts, and configure hosts based on their roles.

**Note:** PXE VLAN must be configured as native in Cisco UCS Manager. This is the only native VLAN in the infrastructure.

• **Management:** This network is created for host management. It is also the out-of-band (OOB) management network for all the devices within the infrastructure such as host KVM devices, Cisco ACI leaf and spine switches, APIC appliances, and fabric interconnects. This network also carries OpenStack internal API traffic; however, this traffic can be separated from management traffic by adding another vNIC.

• **Storage public:** This network is also called the Ceph client network. In this network, computing hosts communicate with Ceph OSD nodes. This network performs request routing and dispatching on behalf of the Ceph client and sends requests directly to Ceph OSD daemons.

• **Storage cluster:** This network carries traffic required to maintain the Ceph storage cluster. It is a common network between the controller nodes and the Ceph OSD nodes (storage servers). This network offloads OSD replication and heartbeat traffic from the storage public network.

• **External:** This network is the public network. It provides external connectivity to the Red Hat content delivery network (CDN), Internet access, and so on. This network is configured only for director and controller nodes. The Red Hat OpenStack Platform Director node also acts as a network gateway for overcloud instances.

• **Tenant data:** This network is the tenant virtual machine data network.

• **ACI infra:** This is the Cisco ACI infrastructure network. The VLAN ID that carries this traffic is determined when the APIC servers are set up. In this design, VLAN ID 3967 is configured. However, the default VLAN ID of 4093 can also be used. This network is used to discover computing hosts within the Cisco ACI in-band network using the Link Layer Discovery Protocol (LLDP).

### Configuring Cisco UCS


### Service Profile and vNIC Templates

In the Cisco Validated Design documented in the deployment guide, service profiles were not bound to templates because of the limitations of the Cisco UCS Manager ML2 plug-in. In this design, the Cisco UCS Manager plug-in is not included. Therefore, service profiles are bound to updating vNIC and service profile templates. Any changes to a template will automatically update the service profiles and vNICs created from the template. Service profile templates for director, controller, computing, and storage nodes are created in the reference architecture.

### Server Pools

Server pools are used to divide the OpenStack server roles for ease of deployment and scalability. These pools also determine the placement of server roles within the infrastructure. Server pools for controller, computing, and storage nodes are created with respect to their placement within the chassis and their roles. Server pools of rack servers are created for Ceph storage nodes. These pools are later associated with their respective service profile templates (Figure 9).
Server vNICs and Placement

vNICs are needed for the various server roles, such as director, controller, computing, and storage. Note that vNIC placement policy should be established for consistent placement of host-level interfaces.

vNIC Configuration for Red Hat OpenStack Platform Director Node

Three vNICs are required for the Red Hat OpenStack Platform director, as shown in Figure 10. These vNICs are configured to fail over to an alternative fabric. For example, the PXE vNIC is mapped to fabric A and will fail over to fabric B. Similarly, the management vNIC is mapped to fabric B and configured to fail over to fabric A, and the external vNIC is mapped to fabric A and configured to fail over to fabric B. This information can be viewed in the Fabric ID column in Figure 10.

Figure 9. Server Pools for OpenStack Server Role Placement

Figure 10. vNICs for Red Hat OpenStack Platform Director Node

vNIC Configuration for Red Hat OpenStack Platform Controller Nodes

Figure 11 shows the vNICs for Red Hat OpenStack Platform controller nodes. The Fabric ID column shows the fabric mapping. For example, vNIC eth1, which carries tenant virtual machine data traffic, is mapped to fabric A and will fail over to fabric B if fabric A is unavailable.

Figure 11. vNICs for Controller Node
vNIC Configuration for Red Hat OpenStack Platform Computing Nodes

Figure 12 shows the vNICs for Red Hat OpenStack Platform computing nodes. The PXE, eth1 (tenant data), management, and storage public vNICs are configured with fabric failover. However, ACI_Infra_A is mapped to fabric A, and ACI_Infra_B is mapped to fabric B. Linux bonding is configured to provide failover and redundancy.

![Figure 12. vNICs for Computing Node](image)

vNIC Configuration for Red Hat OpenStack Platform Ceph Storage Nodes

Figure 13 shows the vNIC configuration for Ceph storage nodes.

![Figure 13. vNICs for Ceph Storage Nodes](image)

Maximum Transmission Unit and Quality-of-Service Configuration

You need to configure jumbo frames and enable quality of service (QoS) in Cisco UCS fabric as shown in Figure 14. For the Best Effort priority, specify a maximum transmission unit (MTU) value of 9000, which will be used in the vNIC templates for the storage public, storage management, and Cisco ACI infrastructure vNICs.

![Figure 14. QoS System Class for Jumbo Frames](image)
Port-Channel Configuration

Port-channel configuration is critical in this reference architecture. Configure port channels on interfaces in fabric A and fabric B (FI-A and FI-B) for uplink connectivity to Cisco ACI leaf switches. Figure 15 shows the port-channel configuration in Cisco UCS Manager.

Figure 15. Configuring Port Channels in Cisco UCS

![Port-Channel Configuration](image)

Configuring Cisco ACI

Perform Cisco APIC Initial Configuration

Log in to the APIC appliance’s Cisco Integrated Management Controller (IMC) using a web browser and launch KVM. The initial setup screen should be visible with the initial setup information shown here. Configure the infrastructure VLAN ID if you are using an ID other than the default; otherwise, the default ID 4093 will be used. Specify the out-of-band management IP address for the APIC. This IP address typically belongs to the management subnet.

Cluster configuration ...

- Enter the fabric name [Cisco ACI Fabric1]:
- Enter the number of controllers in the fabric (1-9) [3]:
- Enter the controller ID (1-3) [1]:
- Enter the controller name [apic1]:
- Enter address pool for TEP addresses [10.0.0.0/16]:
- Enter the VLAN ID for infra network (1-4094) [4093]:3967
- Enter address pool for BD multicast addresses (GIPO) [225.0.0.0/15]:

Out-of-band management configuration ...

- Enter the IP address for out-of-band management: 20.7.10.150/24
- Enter the IP address of the default gateway [None]: 20.7.10.1
  - Enter the interface speed/duplex mode [auto]:

Administrator user configuration...
Enable strong passwords? [Y]
Enter the password for admin:

**Note:** When the APIC boots after the initial setup for the first time, it might take about 5 minutes before you can log in using the admin credentials. The password provided in the initial configuration of APIC 1 will be the password for all the devices in the Cisco ACI fabric such as the leaf and spine switches and other APIC appliances.

**Convert from Cisco NX-OS to Cisco ACI Boot Mode**

If the Cisco Nexus® 9372PX Switch was previously set up in standalone NX-OS mode (for Cisco NX-OS Software), you need to convert it to ACI mode (for Cisco ACI boot) before proceeding to fabric discovery. Follow the guidelines in the following link to convert from NX-OS to ACI boot mode:


For a new Cisco Nexus installation the switches are booted in ACI mode and so the following steps are not required.

**Discover Cisco ACI Fabric**

After the initial setup is complete, log in to the APIC GUI.

1. Browse to https://20.7.10.150 (the OOB management IP address provided at the time of initial setup).
2. Log in using the admin username and password defined during initial setup as shown in Figure 16.

*Figure 16. Cisco APIC Login*

3. Click Fabric in the top menu bar, click the Inventory menu item, and expand Fabric Membership. At this point, at least one of the leaf switches should be discovered and visible.
4. Double-click the highlighted leaf switch in the right pane and assign 101 as the node ID and provide a node name. Fabric discovery will discover the second leaf and assign a node ID 102. Click Update when done.
5. As the fabric discovery continues, both spine switches will appear in the Fabric Membership pane. Assign node IDs and node names.
6. When the node IDs and node names are assigned, the APIC assigns IP addresses from the tunnel endpoint (TEP), the pool defined during initial setup, as shown in Figure 17.
Configure Fabric Access Policies

You need to define access policies such as Cisco Discovery Protocol, LLDP, and Link Aggregation Control Protocol (LACP), which are required to configure a virtual port-channel (vPC) and virtual machine manager (VMM) domain.

Create Cisco Discovery Protocol Interface Policy

Follow these steps to create Cisco Discovery Protocol policy:

2. Right-click CDP Interface and choose Create CDP Interface Policy.
3. Create Cisco Discovery Protocol policy as shown in Figure 18.
Follow these steps to create LLDP interface policy:

2. Right-click LLDP Interface and choose Create LLDP Interface Policy.
3. Create LLDP policy as shown in Figure 19.
Create Port-Channel Interface Policy

Follow these steps to create port-channel interface policy:

2. Right-click Port Channel and choose Create Port Channel Policy.
3. Create LACP policy as shown in Figure 20.

**Figure 20.** Configuring Port-Channel Policy
Configure vPC for the Fabric Interconnect

Follow these steps to configure vPC for the fabric interconnect:

2. Right-click Interface Policies and choose Configure Interface, PC, and VPC.
3. In the VPC Switch Pairs panel, click + to add a vPC.
4. Specify the domain ID, switch 1, and switch 2. Click Save. The vPC domain will be added in VPC Switch Pairs panel as shown in Figure 21.

**Figure 21. Configuring vPC Switch Pairs**

<table>
<thead>
<tr>
<th>VPC Domain Id</th>
<th>Switch 1</th>
<th>Switch 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>101</td>
<td>102</td>
</tr>
</tbody>
</table>

5. Configure switch interfaces for vPC. In the reference topology, ports 17 and 18 from each leaf switch are connected to the fabric interconnect as shown in the cable diagram.
6. Click + in the Configured Switch Interfaces pane. Select both Leaf101 and Leaf102 from the drop-down menu.
7. Click + to configure the switch interfaces. Configure interface 1/17 connecting to FI-A as shown in Figure 22. Also provide the infrastructure VLAN in the VLAN range field as shown in Table 5.
8. Click Submit.
Figure 22. Configuring Switch Interface for FI-A

9. Click + to configure switch interfaces and configure interface 1/18 connecting to FI-B as shown in Figure 23.

Figure 23. Configuring Switch Interface for FI-B
Configure vPC for the OOB Switch

Out-of-band management switch connectivity and configuration is based on the reference topology in the test setup and may vary from site to site. Therefore, it should be configured in Cisco ACI accordingly. This configuration can be used as a reference to set up an external Layer 2 bridge domain such as, in this case, an OOB management switch.

2. Right-click Interface Policies and choose Configure Interface, PC, and VPC.
3. Click + to select switches and select both the leaf switches. Click + to configure switch interfaces and configure interface 1/23 of both leaf switches connecting to the OOB management switch as indicated in the cable diagram and shown in Figure 24. Click Save and then click Submit.

**Figure 24.** Configuring Switch Interface for OOB Management Switch
Validate the vPC Configuration
The vPC configuration for the Cisco UCS fabric interconnect is critical for this reference architecture. To verify that vPC is configured correctly, perform the following steps:

1. From the GUI, choose Fabric > Inventory > Pod 1 > Leaf101 (Node-101) > VPC Interfaces.
2. Click Domain ID. Verify vPC consistency as shown in the Figure 25.
3. From the command-line interface (CLI), use SSH to go to one of the leaf switches and run the `show port-channel summary` command. If the port channel in Cisco UCS Manager is also configured correctly, the port channel will be listed as Switched and Up (SU).

Figure 25. Verifying vPC

Deploy Infrastructure Tenant
In this section, you will create a new tenant. This tenant will manage infrastructure connectivity between hosts. This tenant will also allow management connectivity from the OOB management switch to servers.

1. On the main menu bar, click Tenants and in the submenu click Add Tenant.
2. In the Create Tenant dialog box, enter `osp_infra` as the tenant name. Click Submit to add a tenant.
Create a Bridge Domain, EPG, and VRF Instance in Infrastructure Tenant

Follow these steps to create a VRF instance, bridge domain, and EPG for the infrastructure network in the osp_infra tenant.

1. Create a VRF instance named vrf_osp_infra.

2. Create a bridge domain for the PXE, management, external, storage public, and storage cluster networks.

   Create a subnet within each bridge domain, which will act as a switch virtual interface (SVI), as shown in Figure 26. Specify the VRF instance created in step 1 and enable unicast routing.

   **Figure 26. Creating a Bridge Domain**

3. Create an application profile in the application profiles of the infrastructure tenant.

4. Create EPGs as shown in Figure 27 in the application profile created in step 3. Deploy Static Binding (Paths) for the vPC and provide encapsulation for the required VLAN. As soon as the static binding path is deployed, the associated VLAN is added to the vPC.
5. For the management and external EPGs, also add Static Binding (Paths) for the OOB management switch as shown in Figure 27 and allow the required VLAN.

6. Add a physical domain association to the EPG as shown in Figure 28.
7. Create a contract to allow all traffic types under Security Policies. Add this contact as a provided contract in each EPG as shown in Figure 29.
Figure 29. Adding Provided Contract to EPGs

Now the system is ready for OpenStack deployment.

Installing Red Hat OpenStack Cloud Platform

This section provides an overview of Red Hat OpenStack Cloud Platform installation and configuration.

Red Hat OpenStack Cloud Platform

The Red Hat OpenStack Platform director offers a set of tools for installing and configuring the OpenStack environment. The Red Hat OpenStack installation is based on TripleO OpenStack deployment topology. The standard hardware configuration for Cisco UCS with Cisco ACI on Red Hat OpenStack Platform is as follow:

- One director node to host Red Hat OpenStack software and an undercloud installation
- Three overcloud controller nodes
- Four computing nodes
- Three Ceph storage nodes (Cinder storage)

The Cisco UCS with Cisco ACI on Red Hat OpenStack solution is based on Red Hat OpenStack Platform 8 (Liberty release) on RHEL 7.2.

Red Hat OpenStack Deployment Summary

To install and configure the RHEL and Red Hat OpenStack software, you will follow these steps:

1. Download RHEL 7.2 from [http://access.redhat.com](http://access.redhat.com).
2. Install RHEL 7.2 and configure the network on the director node.


The sample Heat templates used in the solution described here can be referenced in Appendix B. You can also customize your Heat templates based on your infrastructure setup.

After OpenStack is deployed, the system is ready for integration of the Cisco ACI platform’s OpFlex plug-in.

Installing the OpFlex ML2 Plug-in
This section covers the installation and configuration of the Cisco ACI platform’s OpenStack plug-in on Red Hat OpenStack Platform.

Set Up the Software Archive on the Repository Server
To install the Cisco ACI OpenStack plug-in, you need to set up a repository (repo) server, which acts as a central point for distributing consistent code versions to all the OpenStack server nodes in the system. Within your environment, you can use any web server for this purpose, depending on its IP connectivity to the OpenStack servers. However, in this reference architecture, the Red Hat OpenStack Platform director node is used to set up the repo server for the Cisco ACI plug-in. Perform the following steps to set up the Cisco ACI plug-in repository:

1. Download the Cisco ACI OpenStack plug-in from the Cisco Software Download website.
2. Create an aciplugin folder and upload the TAR file to the director node in the folder: for example, /root/aciplugin.
3. Extract the TAR file using the following command:
   ```
   # tar -xzf dist-rpms-1.3.1-liberty-20160515.tar.gz
   ```
4. Create a folder:
   ```
   # mkdir /var/www/html/opflex
   ```
5. Copy the extracted Red Hat Package Manager (RPM) files and set appropriate permissions:
   ```
   # cp /root/aciplugin/*.rpm /var/www/html/opflex
   # chmod -R 755 /var/www/html/opflex
   ```
6. Create a repo server using the createrepo command. This will create the repodata folder.
   ```
   # createrepo -v /var/www/html/opflex
   ```
7. At this point, the director node is configured as a repo server for the Cisco ACI plug-in. To test it, type the base URL in your browser as http://173.36.219.206/opflex. You should see all the RPM files and the repodata folder.

   **Note:** Be sure that all the controller and computing nodes reach to the repo server. If the repo server is not set up in the director node, make sure that firewall settings are not blocking HTTP access and IP connectivity. If you see any HTTP error message such as a “forbidden” error message while yum is installing, RPM access has been denied in the permissions.

8. Create the following file in the controller and computing nodes:
   ```
   # vi /etc/yum.repos.d/opflex.repo
   ```
9. Add the following content to the file:
[opflex]
name=opflex repo
baseurl=http://173.36.219.206/opflex
enabled=1
gpgcheck=0

10. The repo server is now set up. Run the following commands:

# yum clean all
# yum repolist

Configure OpenStack Computing Nodes for the Cisco ACI Infrastructure Network
The OpenStack computing nodes need to be prepared to properly interact with the Cisco ACI fabric for OpFlex. This preparation includes interface configuration, DHCP configuration for the interface on the ACI infra VLAN, and LLDP communication.

Install the LLDP Package
For the Cisco ACI fabric to be able to dynamically discover the OpenStack nodes, a software LLDP stack is needed on the server.

1. To install the LLDP package, run the following commands:
   
   # yum -y install lldpad

2. Start and enable the lldpd service, which will be reloaded after a server reboot:

   # service lldpd start
   # chconfig lldpd on

Configure Cisco UCS Manager for the ACI Infra VLAN
Verify that the following settings are configured in Cisco UCS Manager as a part of Cisco ACI infrastructure connectivity for the computing hosts:

1. Set the MTU value for the vNIC carrying the Cisco ACI infrastructure traffic to 9000.
2. Configure VLAN 3967 in the Cisco ACI infrastructure vNICs as shown in Figure 30.

Figure 30. ACI Infra VLAN Configuration in vNIC

3. Turn on multicast in the fabric interconnect. Broadcast domains for virtual networks are implemented as multicast channels in the physical network. To allow the broadcast traffic for virtual networks, multicast traffic
should be allowed through the fabric interconnects. Create a multicast policy by choosing LAN > Policies > Multicast Policies as shown in Figure 31.

**Figure 31.** Creating Multicast Policy

![Multicast Policy](image)

4. Add multicast policy in the ACI infra VLAN as shown in Figure 32.

**Figure 32.** Adding the Multicast Policy Name in ACI Infra VLAN

![Multicast Policy in ACI Infra VLAN](image)

Configure Cisco ACI Leaf Switches

The ACI infra VLAN should also be allowed in the vPC that connects to the fabric interconnect in the leaf switches. You need to enable the infrastructure VLAN in the attachable access entity profiles for FI-A and FI-B. Perform the following steps:

1. Click Fabric on the menu bar and then click the Access Policies submenu.
3. Click Attachable Entity Access Profile for FI-A (FI-A_AttEntityP in this case).
4. Select the Enable Infrastructure VLAN checkbox as shown in Figure 33.
5. Repeat steps 3 and 4 for Attachable Entity Access Profile for FI-B (FI-B_AttEntityP in this case).
6. After you performing the preceding steps, VLAN 3967 is added in the vPC. To verify this addition, use SSH on the Cisco ACI leaf switch to run the `show vpc` command, or in the GUI choose Fabric > Inventory > Pod 1 > Leaf101 > Interfaces < VPC Interfaces.

Prepare the Interface for the ACI Infra VLAN

Perform the following steps in all the computing hosts:

1. Log in to the computing host. Shut down the interfaces dedicated to ACI in fra VLAN traffic. In the reference test setup, these interfaces are enp14s0 and enp15s0.

2. Edit the `/etc/sysconfig/network-scripts/ifcfg-enp14s0` file as follows:

   ```
   [root@overcloud-compute-0 network-scripts]# cat ifcfg-enp14s0
   
   DEVICE="enp14s0"
   ONBOOT="yes"
   NM_CONTROLLED="no"
   HOTPLUG=no
   MASTER=bond0
   SLAVE=yes
   BOOTPROTO=none
   ```

3. Edit the `/etc/sysconfig/network-scripts/ifcfg-enp15s0` file as follows:

   ```
   [root@overcloud-compute-0 network-scripts]# cat ifcfg-enp15s0
   
   DEVICE="enp15s0"
   ONBOOT="yes"
   NM_CONTROLLED="no"
   HOTPLUG=no
   MASTER=bond0
   SLAVE=yes
   ```
BOOTPROTO=none

4. Create file `/etc/sysconfig/network-scripts/ifcfg-bond0` with the following contents:

   ```
   [root@overcloud-compute-0 network-scripts]# vi ifcfg-bond0
   DEVICE="bond0"
   ONBOOT="yes"
   NM_CONTROLLED="no"
   HOTPLUG=no
   BONDING_OPTS="mode=1 miimon=1"
   MTU=1600
   ```

   **Note:** The miimon parameter is set to 1, which is required to change the NIC state from active to backup and from backup to active, depending on the link status of the NIC.

5. Create `/etc/sysconfig/network-scripts/ifcfg-vlan3967` file with the following contents:

   ```
   DEVICE=vlan3967
   ONBOOT=yes
   HOTPLUG=no
   NM_CONTROLLED=no
   VLAN=yes
   BOOTPROTO=dhcp
   PHYSDEV=bond0
   ```

6. Create the `/etc/sysconfig/network-scripts/route-vlan3967` file. Specifically apply a multicast route to the OpFlex infrastructure VLAN interface:

   ```
   224.0.0.0/4 via 0.0.0.0 dev vlan3967
   ```

7. Create the `/etc/dhcp/dhclient-vlan3967.conf` file with the following content:

   ```
   send dhcp-client-identifier 01:00:25:b5:00:0a:6d;
   request subnet-mask, domain-name, domain-name-servers, host-name;
   send host-name overcloud-compute-0;
   option rfc3442-classless-static-routes code 121 = array of unsigned integer 8;
   option ms-classless-static-routes code 249 = array of unsigned integer 8;
   option wpad code 252 = string;
   also request rfc3442-classless-static-routes;
   also request ms-classless-static-routes;
   also request static-routes;
   also request wpad;
   also request ntp-servers;
   ```

   **Note:** In this reference architecture, 3967 was specified as the ACI infra VLAN at the time of initial APIC setup. If your VLAN ID is other than 3967, use that VLAN ID. For example, if the ACI infra VLAN is set to the default 4093 VLAN ID, then these files should be named ifcfg-vlan4093, route-vlan4093, and dhclient-vlan4093. You can see the infrastructure VLAN in the APIC GUI by choosing System > Controllers > Controllers > apic 1 > Interfaces.
8. Bring the interface back up:
   
   ```
   # ifup vlan3967
   ```

9. The DHCP discovery process should begin. If everything is configured correctly, interface vlan3967 will receive a DHCP offer from the Cisco ACI fabric, and the Cisco ACI infrastructure IP address will be assigned to this interface as shown in Figure 34.

![Figure 34. ACI Infra VLAN Interface](image)

Configure OpenStack Controller Hosts

This section describes how to update the OpenStack Neutron servers, which are controllers in this case. The Neutron servers in OpenStack provide the primary interaction with the Cisco ACI fabric through the APIC for dynamic provisioning of networking for OpenStack tenants.

This section describes the installation of the OpFlex agent and driver along with specific configuration file edits required to enable APIC communication. The Cisco ACI platform’s OpenStack plug-in allows the Cisco ACI fabric to replace the Layer 3 forwarding functions typically provided by the OpenStack Neutron Layer 3 agent (neutron-l3-agent) service. That service will not be used.

Follow these steps in all three controller nodes:

1. Disable the neutron-l3-agent service because it is no longer needed by entering the following commands:
   
   ```
   # systemctl stop neutron-l3-agent
   # systemctl disable neutron-l3-agent
   ```

   **Note:** In a cluster environment, it is recommended to use pcs resource disable neutron-l3-agent to disable neutron L3 agent service.

2. On the OpenStack controller node, install the Neutron OpFlex agent (neutron-opflex-agent), APIC API, and ML2 driver along with the required supporting modules. These packages are pulled from the Extra Packages for Enterprise Linux (EPEL) repository, so be that EPEL is enabled on the node for successful installation.

   Supporting modules python-pip and python-pbr are required as prerequisites:
   
   ```
   # yum install -y python-pip
   # yum install -y python-pbr
   ```

3. Install the OpFlex agent, APIC API, and ML2 driver:
   
   ```
   # yum install -y neutron-opflex-agent
   # yum install -y apicapi
   # yum install -y neutron-ml2-driver-apic
   # yum install -y agent-ovs
   ```
4. **After installation is complete, modify the /etc/neutron/neutron.conf file to point to the APIC for networking services.** Edit the file with the following service plug-in information:

   ```
   service_plugins = cisco_apic_l3,qos
   ```

   **Note:** You need to be sure not to remove services that do not clash with services required by this plug-in. For example, if load balancing as a service (LBaaS) or metering service is enabled, you need to make sure that the service remains enabled.

5. **Edit the /etc/neutron/plugins/ml2/ml2_conf.ini file with the following configuration.** These changes are required to enable the correct mechanism driver for the APIC and add OpFlex as a new network type.

   ```
   type_drivers = opflex,vxlan,vlan,flat,gre
   tenant_network_types = opflex
   mechanism_drivers = cisco_apic_ml2
   ```

6. **Verify the network VLAN ranges.** This configuration is set by the openstack overcloud deploy command.

   ```
   network_vlan_ranges =physnet-tenant:250:749
   ```

7. **Edit the /etc/neutron/dhcp_agent.ini file.** Change the dhcp_driver parameter and verify other values.

   ```
   ovs_integration_bridge = br-int
dhcp_driver = apic_ml2.neutron.agent.linux.apic_dhcp.ApicDnsmasq
   enable_isolated_metadata = True
   ```

   **Note:** Make sure that br-int is not commented out.

8. **The OpFlex agent-ovs component provides local DHCP lease delivery to virtual machine instances on each computing node by default.** The ml2_conf_cisco_apic.ini file contains a setting to control distributed behavior: `enable_optimized_dhcp`. This parameter has a default value of `True` if it is not overridden in the file. The neutron-dhcp-agent process is still required on the Neutron servers to handle IP address management and properly communicate dnsmasq processes to the agent-ovs DHCP function. Restart neutron-dhcp-agent to apply any configuration changes:

   ```
   # systemctl restart neutron-dhcp-agent
   ```

9. **The ml2_conf_cisco_apic.ini file is the primary configuration file on the controllers for customizing the interaction of the Cisco ACI platform’s OpenStack plug-in with the APIC.** In this file, the APIC IP address, login credentials, and default names for objects in the Cisco ACI policy model are configured. Edit the /etc/neutron/plugins/ml2/ml2_conf_cisco_apic.ini file as follows:

   ```
   [DEFAULT]
apic_system_id = osp8apic # Any string. This would show up as ACI Tenant for OpenStack System
   
   [opflex]
networks = '*'
   
   [ml2_cisco_apic]
   # Hostname:port list of APIC controllers
   apic_hosts = 20.7.10.150
   # Username for the APIC controller
   apic_username = admin
   # Password for the APIC controller
   apic_password = <apic_password>
   ```
# Whether use SSL for connecting to the APIC controller or not
apic_use_ssl = True

# How to map names to APIC: use_uuid or use_name
apic_name_mapping = use_name

# Agent timers for State reporting and topology discovery
apic_agent_report_interval = 30
apic_agent_poll_interval = 2
apic_provision_infra = False
apic_provision_hostlinks = False
enable_optimized_dhcp = True
enable_optimized_metadata = True

Here, apic_provision_infra = True is required when the system is first launched to create the VMM domain in the APIC. If preexisting server connections are being used and are already defined in the APIC, the VMM domain created will also need to be manually associated with the attachable entity profile (AEP) used when those connections were created. The True setting also enables the VLAN pool to be created for tenant networks if the VLAN encapsulation mode is in use.

In addition, note the following:

- apic_provision_hostlinks = False is the manual server port provisioning.
- enable_optimized_dhcp = True is true by default.
- enable_optimized_metadata = True is if you want to distribute metadata.

10. After the ml2_conf_cisco_apic.ini file has been edited, it needs to be added to the service definition for the OpenStack Neutron server service to be read for options when the service starts. Edit the /usr/lib/systemd/system/neutron-server.service file and add --config-file /etc/neutron/plugins/ml2/ml2_conf_cisco_apic.ini to the ExecStart line as shown here:

```
[Unit]
Description=OpenStack Neutron Server
After=syslog.target network.target

[Service]
Type=notify
User=neutron


PrivateTmp=true
NotifyAccess=all
```
KillMode=process

Install]
WantedBy=multi-user.target

11. With the Neutron server service definition updated to read the configuration file, reload the systemd manager configuration and then restart the Neutron server using the following commands:
   # systemctl daemon-reload
   # service neutron-server restart

12. Verify that the /etc/neutron/plugins/ml2/openvswitch_agent.ini file contains the settings shown in the following example:
   
   [ovs]
   enable_tunneling=False
   integration_bridge=br-int

13. Verify that the configuration lines for tunnel_bridge, vxlan_udp_port, and tunnel_types are either deleted or commented out.

14. Stop and disable the neutron-openvswitch-agent service. Enter the following commands:
   
   # service neutron-openvswitch-agent stop
   # systemctl disable neutron-openvswitch-agent

Configure OpenStack Computing Hosts
Now configure the OpenStack computing hosts.

1. Install neutron-opflex-agent and agent-ovs from the yum repository:
   
   # yum install -y neutron-opflex-agent
   # yum install -y agent-ovs

2. Verify that the /etc/neutron/plugins/ml2/openvswitch_agent.ini file contains the settings shown in the following example:
   
   [ovs]
   enable_tunneling=False
   integration_bridge=br-int

3. Verify that the configuration lines for tunnel_bridge, vxlan_udp_port, and tunnel_types are either deleted or commented out.

4. Stop and disable the neutron-openvswitch-agent service. Enter the following commands:
   
   # service neutron-openvswitch-agent stop
   # systemctl disable neutron-openvswitch-agent

5. To help ensure that the system is running cleanly with the new configuration and OVS module up to this point, reboot each of the servers.

6. After the servers have completed the reboot process, log in and change to the /etc/opflex-agent-ovs/conf.d directory.

7. The agent-ovs service reads its configuration from the /etc/opflex-agent-ovs/opflex-agent-ovs.conf file, and the conf.d subdirectory allows you to override specific settings in that file using smaller JSON-formatted files. Create a new /etc/opflex-agent-ovs/conf.d/10-opflex-connection.conf file with the contents shown in the following example:
Here,

- `<apic system id>` is the ID that was specified in the `/etc/neutron/plugins/ml2/ml2_conf_cisco_apic.ini` file in the Neutron server. In the case here, the ID is `osp8apic`.
- `<host name of this system>` is the Linux server host name local to each OpenStack host: for example, `overcloud-compute-0`.
- The IP address next to the host name in this example is a fabric interface for OpFlex communication, if the Cisco ACI fabric was installed with the default IP address pool for tunnel endpoints: `10.0.0.0/16`. If this IP address pool was altered during the fabric installation, change the address used here to match the fabric. SSH to a leaf switch. Use the `show ip interface` command to identify the addresses in use in your fabric. The host-name address for the OpFlex peer is the anycast IP address assigned to the SVI of the infrastructure VLAN on the leaf switches.

8. The OpFlex configuration requires a second set of override values specific to the VLAN configuration between the host and the leaf switch. Create a new `/etc/opflex-agent-ovs/conf.d/20-vlan-aci-renderer.conf` file using the following example:

```json
{
    "renderers": {
        "stitched-mode": {
            "ovs-bridge-name": "br-int",
            "encap": {
                "vlan": {
                    "encap-iface": "<tenant-VLAN-trunk>"
                }
            },
            "flowid-cache-dir": "'/var/lib/opflex-agent-ovs/ids"
        }
    }
}
```
Here, `<tenant-VLAN-trunk>` is the name of the tenant VLAN trunk interface. In this reference topology, the name is `enp7s0`.

9. Depending on the installation tool used to provision OpenStack, you may have extra ports and bridges configured in your OVS setup that are no longer required. For example, an OVS bridge named `br-ex` is commonly provisioned for an external network on Neutron nodes, and it will not be required here. An interface bridge such as `br-ethX` or `br-tenant`, in this case, is commonly provisioned by a VLAN-mode installation to carry VLAN traffic. Its function is replaced by the tenant network interface that is added directly to `br-int`. If the tenant interface is part of any other bridge, use the `ovs-vsctl del-port` command to remove that interface from that bridge, and use the `ovs-vsctl add-port` command to add the tenant interface to `br-int` until the simplified OVS configuration looks similar to the `ovs-vsctl show` output shown here:

```
Bridge br-int
  fail_mode: secure
  Port br-int
    Interface br-int
      type: internal
  Port <tenant-VLAN-trunk>
    Interface <tenant-VLAN-trunk>

ovs_version: "2.4.0"
```

When virtual machine instances are started on a computing node, the system will dynamically add OVS interfaces starting with `qvo` to `br-int` to link them to the individual Linux bridges used to attach each virtual machine. Virtual machine traffic transits `br-int` as programmed by `agent-ovs` out the tenant VLAN interface to the Cisco ACI fabric.

10. Run the following command to support OpenFlow versions other than OpenFlow 1.0.

```
# ovs-vsctl set bridge br-int protocols=[]
```

Start and Enable Agent Services

This section describes how to start and enable the agent services.

1. With the OpFlex configuration in place, start and enable the `neutron-opflex-agent` and `agent-ovs` services.
   Enter the commands as shown in the following example:

```
# service agent-ovs restart
# service neutron-opflex-agent restart
# systemctl enable agent-ovs
# systemctl enable neutron-opflex-agent
```

2. The APIC host agent is required to provide LLDP autodiscovery of host-to-server connectivity between OpenStack servers and Cisco ACI leaf switches. The host agent listens to LLDP information coming from the Cisco ACI fabric to identify the leaf switch and physical port to which each computing node is connected. It then updates that information on the OpenStack controller. Run the commands in the following example to start the agent and make it persistent:

```
# service neutron-cisco-apic-host-agent restart
# systemctl enable neutron-cisco-apic-host-agent
```

3. After all the services are running, verify that the interface for the OpFlex infrastructure VLAN is in the Up state, or bring up the interface using the `ifup <interface name>` command; for example, `# ifup vlan3967`. 

At this point, computing hosts are listed in Cisco ACI under Hypervisors in the OpenStack VMM domain.

**Note:** It is also recommended that neutron-opflex-agent and ovs-agent to be added to pacemaker in the controllers so that it manages those services as well.

Initialize the Cisco ACI Tenant
This section describes how to initialize the Cisco ACI tenant.

1. The Cisco ACI platform’s OpenStack plug-in software is now up and running and ready to provision tenant networks in OpenStack. The population of the APIC with OpenStack configuration will not begin until the first network segment is created in OpenStack with the Cisco ACI platform’s OpenStack plug-in active. Create a test Neutron network under the Admin Project in OpenStack to trigger initial creation of the Cisco ACI tenant and VMM domains in the APIC.

2. After the network is created, log in to the APIC GUI. On the menu bar, click Tenants to verify that a new Cisco ACI tenant has been created and named using the Cisco ACI platform’s OpenStack plug-in system name.

3. Click VM Networking. In the navigation pane, click OpenStack to verify that the VMM domain created was named to reflect the Cisco ACI system name. All the computing hosts will also be discovered and listed under Hypervisors as shown in Figure 35.

**Figure 35.** OpenStack VMM Domain

4. This VMM domain must be associated with an attachable access entity profile. Click Fabric in the top menu bar and then click the Access Policies submenu.

5. Expand Global Policies and Attachable Access Entity Profiles in the left navigation pane. Click the AEP that is referenced from Interface Policies > Policy Groups: in this case, FI-A_AttEntityP.

6. Click + to add the VMM domain in the right pane as shown in Figure 36 and click Submit.

7. Repeat step 6 for AEP FI-B_AttEntityP.
Enable Optimized Neutron Services

This section describes the configuration of distributed DHCP and metadata proxy functions.

Enable Optimized DHCP Services
The /etc/neutron/plugins/ml2/ml2_conf_cisco_apic.ini file contains the configuration line enable_optimized_dhcp. The default setting for this parameter is True. If this parameter is left at the default or set to True, the discovery, offer, response, and acknowledgment (or DORA) functions that interact with the virtual machine instances remain local to each computing node.

The local agent-ovs service handles this interaction for each computing node. Address allocation is still handled on the Neutron servers by the neutron-dhcp-agent and communicated to the agent-ovs instances over the management network. If this parameter is set to False, then the system reverts to a centralized DHCP function, with all address allocation and DORA functions on the Neutron servers.

Enable Optimized Metadata Services
This section describes how to enable optimized metadata proxy services in the Cisco ACI environment.

1. Edit the /etc/neutron/plugins/ml2/ml2_conf_cisco_apic.ini file as follows:
   ```ini
   enable_optimized_metadata=True
   ```
2. Disable the metadata agent in the Neutron servers and restart neutron-server service as follows:
   ```bash
   # systemctl stop neutron-metadata-agent
   # systemctl disable neutron-metadata-agent
   # systemctl restart neutron-server
   ```
3. Copy the /etc/neutron/metadata-agent.ini file from Neutron servers and restart the following services in all computing hosts:
Setting Up the OpenStack External Network

The OpFlex configuration of an external OpenStack network requires an external routed network for an L3-Out connection to exist in the APIC tenant or the common tenant. L3-Out provides a path to an external routing entity for communication outside the Cisco ACI fabric. L3-Out can be configured in several ways, such as using Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), Border Gateway Protocol (BGP), or static routing. A routed interface, a vPC with an SVI, or a routed subinterface can be configured. The Cisco ACI platform’s OpenStack plug-in requires a preconfigured L3-Out connection. The Cisco ACI network administrator is responsible for setting up appropriate routing for the environment. L3-Out configuration varies depending on the site requirements.

Prerequisites

- Two separate IP subnets are required for SNAT and floating IP communication on the external network.
- The upstream routers outside the Cisco ACI fabric must be configured with IP routes for these two subnets, either through the routing protocol in use or statically.
- The Cisco ACI fabric must have a BGP route reflector enabled to use an OpenStack external network.

Configure L3-Out in Cisco ACI

In OpFlex plug-in architecture, NAT functions occur in the OVS of the computing hosts. Physical switches in Cisco ACI need only to route external traffic to and from the external next-hop router. This external routing is performed through the VRF object associated with the L3-Out connection. This L3-Out VRF instance has an interface associated with the physical link to the external next-hop router. The same VRF instance also has an SVI with an IP address of the assigned SNAT subnet and the floating IP subnet. The loopback interface is also a part of this VRF instance for routing protocol interactions. Figure 37 shows the OpFlex OVS NAT architecture.
Figure 37. OpFlex OVS NAT Architecture

The L3-Out VRF instance associated with the OpenStack Neutron external network processes NAT traffic that egresses the OVS on the computing host. Non-NAT traffic is processed by a tenant VRF instance according to the OpenStack tenant association of the guest virtual machine.

L3-Out in Cisco ACI is required to operate as a Neutron external network in OpenStack. In this reference architecture, a shared L3-Out connection is configured under the common tenant in the APIC. However, a private L3-Out connection can also be added to the APIC tenant allocated to the OpenStack instance. The routed connection can be configured using available routing protocols depending on the upstream router protocol in use.

In this design, for simplicity and reference purposes, static routing is configured. This approach is useful for test and lab deployments of the Cisco ACI platform’s OpenStack plug-in. For a production-class deployment, you should use more advanced routing that is consistent with rest of the specific Cisco ACI environment and upstream routers.

Note: The interface to be used for L3-Out connectivity must be configured under Fabric > Access Policies > Interface Policies. Right-click and choose Configure Interface, PC, and VPC.

Use the following steps to configure L3-Out under the common tenant in the test setup using the APIC GUI. You can also configure L3-Out using XML as shown in Appendix A.

1. In the APIC GUI, click Tenant and then click Common.
2. In the left navigation pane, expand Networking > External Routed Networks. Right-click External Routed Network and choose Create Routed Outside.
3. In the Create Routed Outside dialog box, enter the name of the L3-Out connection (for example, enter L3-Out). In the VRF drop-down menu, choose Create VRF and specify a name for the VRF instance (for example, enter vrf1).
4. Click + to add nodes and interface protocol profiles.

5. Enter a name (for example, enter BorderLeaf) in the Create Node Profile dialog box.

6. Click + for Nodes. The Select Node dialog box will appear. Select the node ID from the drop-down menu. In this reference topology, the interface is configured in Leaf101. Therefore, select topology/pod-1/node-101.

7. Specify the router ID. The Router ID field is used for OSPF and BGP communication. This ID can be set to an address of your choice for a static routed setup (for example, enter 2.2.2.2).

8. Click + for Static Routes. Specify 0.0.0.0/0 in the Prefix field of the Create Static Route dialog box. Click + for Next Hop Addresses. Enter the next hop address, such as 20.7.60.1, which is the address assigned to the external router outside the Cisco ACI fabric on the link subnet. Click Update. Then click OK.

9. In the Select Node dialog box, click OK.

10. Click + for Interface Profiles in the Create Node Dialog box. The Create Interface Profile dialog box will appear. Enter the name of the interface profile (for example, enter portProfile).

11. Configure the routed interface. Click + for Interfaces. The Select Routed Interface dialog box will appear. Select the interface from the Path drop-down menu. In this topology, for example, you can choose Leaf101 (Node 101) > 1/37. The path will appear as topology/pod-1/paths-101/pathep[eth1/37]. Specify 20.7.60.2/24 in the IPv4 Primary Address field. Click OK to save your settings and close the Select Routed Interface dialog box.

12. Click OK to create the interface profile and click OK to create the node profile.

13. Click Next to create an external EPG network. Click + to add an external EPG network. Enter a name (for example, enter extroute) in the Create External Network dialog box.

14. Click + to add a subnet. Enter 0.0.0.0/0 in the IP address field of the Create Subnet dialog box. Click OK.

15. In the Create External Network dialog box, click OK.

Note: After the external EPG network has been created, it will appear under External Routed Network in the APIC in the Networks folder. When viewed in the Network folder, it will not be referred to as an EPG.

16. Click Finish to complete the L3-Out configuration.

17. Verify the configuration by running the following commands on the leaf switch:
   Leaf101# show vrf
   Leaf101# show ip interface
   Leaf101# iping 20.7.60.1 -V common:vrf1

The show vrf command displays a VRF instance added to the leaf switch for L3-Out.

The show ip interface command shows the interface that carries the IP address assigned.

You should be able to ping inbound to the IP address assigned to Cisco ACI on the link subnet from the external router at the address assigned in step 8.

Configure L3-Out in OpenStack Controller
Once L3-Out is configured and tested in Cisco ACI, configure OpenStack plugin to communicate with External Routed Network on the Neutron servers(s).

1. Edit the /etc/neutron/plugins/ml2/ml2_conf_cisco_apic.ini file. Add the following section:
   [apic_external_network:L3-Out]
Here, **L3-Out** is the name of the external routed network and **extroute** is the EPG previously configured in Cisco ACI under the common tenant. The **host_pool_cidr** entry is the IP address of the SNAT default gateway with a prefix. This is the subnet used for SNAT and should be identified with prefix notation. This is the address that the Cisco ACI platform’s OpenStack plug-in will use to add the subnet to the correct bridge domain in the APIC.

2. After the ml2_conf_cisco_apic.ini file is modified for L3-Out on all the Neutron nodes, restart the Neutron server to apply the new configuration:

   ```
   # service neutron-server restart
   ```

3. Create the external network in OpenStack using the OpenStack Horizon dashboard or the CLI. Log in to the undercloud node and run the following commands. The subnet added here will be used to provide floating IP addresses to the instances. It will be added in the bridge domain in the APIC by the APIC appliance’s OpenStack plug-in.

   ```
   # source overcloudrc
   # neutron net-create L3-Out --router:external --shared
   # neutron subnet-create L3-Out 10.1.3.0/24 --name L3-Out-Subnet
   ```

   **Note:** The name of external network in OpenStack must match the name specified in the ml2_conf_cisco_apic.ini file.

4. Run the Neutron list command. You will see a secondary network created by the driver named host-nat-network-for-internal-use as shown in Figure 38. This network allows OpenStack and Cisco ACI to properly process SNAT traffic for virtual machine instances that do not have floating IP addresses assigned. The OpFlex system will automatically assign a SNAT address from the host_pool_cidr subnet to each computing host in the OpenStack cluster.

   **Figure 38.** SNAT Subnet

   ![SNAT Subnet](image)

5. Subnets for the L3-Out OpenStack network and SNAT network will be created in bridge domains under the common tenant through the Cisco ACI platform’s OpenStack plug-in as shown in Figure 39.
Create Route Reflector in APIC

A BGP route reflector must be enabled in Cisco ACI for the external network to work. This can be achieved through the web interface following the system documentation or by entering the following command on the OpenStack controller node after the Cisco ACI platform’s OpenStack plug-in has been installed:

```
# apic route-reflector-create --apic-ip <APIC IP Address> --apic-username admin--apic-password <APIC Password> --no-secure
```

Conclusion

The combination of Cisco UCS, Cisco ACI, and Red Hat OpenStack Platform enables agility, flexibility, and programmability across public and private clouds. Provisioning of tenant resources in OpenStack can deploy the network configuration across the Cisco ACI data center fabric. Cisco ACI treats the network as a single entity rather than a collection of switches, using a central controller to implicitly automate common practices such as Cisco ACI fabric startup, upgrades, and individual element configuration. These innovations are designed to ease the operations of data center networks and provide a path to a fully automated fabric solution. The ultimate goal of an integrated Red Hat OpenStack and Cisco ACI solution is to significantly reduce operating expenses and achieve zero-touch provisioning that can be configured dynamically and on demand to meet specific application requirements.

For More Information

- Connecting Cisco ACI to outside Layer 2 and Layer 3 networks:
- Cisco APIC firmware management guide:
Cisco ACI and OpenStack OpFlex deployment guide for Red Hat:
http://www.cisco.com/c/en/us/td/docs/sAir/switches/datacenter/aci/apic/sw1-

Red Hat OpenStack Platform 8 architecture guide:

Cisco UCS Manager configuration guide:

Cisco UCS Integrated Infrastructure with Red Hat OpenStack Platform and Red Hat Ceph Storage deployment guide:

Appendix A: Layer 3 Outside Configuration Using XML

This appendix shows the L3-Out configuration using XML.

L3-Out Configuration Using XML

```xml
<imdata totalCount="1">
  <l3extOut descr="" dn="uni/tn-common/out-L3-Out" enforceRtctrl="export" name="L3-Out" ownerKeys="" ownerTag="" targetDscp="unspecified">
    <l3extRsEctx tnFvCtxName="vrf1"/>
  </l3extOut>

  <l3extLNodeP configIssues="" descr="" name="borderLeaf" ownerKey="" ownerTag="" tag="yellow-green" targetDscp="unspecified">
    <l3extRsNodeL3OutAtt rtrId="2.2.2.2" rtrIdLoopBack="yes" tDn="topology/pod-1/node-101">
      <ipRouteP aggregate="no" descr="" ip="0.0.0.0/0" name="" pref="1" rtCtrl="">
        <ipNexthopP descr="" name="" nhAddr="20.7.60.1" pref="unspecified" type="prefix"/>
      </ipRouteP>
    </l3extRsNodeL3OutAtt>

    <l3extLIfP descr="" name="portProfile" ownerKey="" ownerTag="" tag="yellow-green">
      <l3extRsNdIfPol tnNdIfPolName=""/>
      <l3extRsIngressQosDppPol tnQosDppPolName=""/>
      <l3extRsEgressQosDppPol tnQosDppPolName=""/>
      <l3extRsPathL3OutAtt addr="20.7.60.2/24" descr="" encap="unknown" encapScope="local" ifInstT="13-port" l1Addr="::" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit" tDn="topology/pod-1/paths-101/pathep-[eth1/37]" targetDscp="unspecified"/>
    </l3extLIfP>
  </l3extLNodeP>
</imdata>
```
Appendix B: Overcloud Deployment .yaml files

This appendix shows the .yaml files used in the test setup.

network-environment.yaml
[root@osp8-aci-host templates]# cat network-environment.yaml

resource_registry:

OS::TripleO::NodeUserData:
  /home/stack/templates/wipe_disk.yaml
OS::TripleO::Compute::Net::SoftwareConfig:
  /home/stack/templates/nic-configs/compute.yaml
OS::TripleO::Controller::Net::SoftwareConfig:
  /home/stack/templates/nic-configs/controller.yaml
OS::TripleO::CephStorage::Net::SoftwareConfig:
  /home/stack/templates/nic-configs/ceph-storage.yaml
# OS::TripleO::BlockStorage::Net::SoftwareConfig:
#  /home/stack/templates/nic-configs/cinder-storage.yaml
# OS::TripleO::ObjectStorage::Net::SoftwareConfig:
#  /home/stack/templates/nic-configs/swift-storage.yaml

parameter_defaults:

# This section is where deployment-specific configuration is done

# Customize the IP subnets to match the local environment

InternalApiNetCidr: 20.7.10.0/24  
StorageNetCidr: 20.7.30.0/24  
StorageMgmtNetCidr: 20.7.40.0/24  
#TenantNetCidr: 10.23.200.0/24  
TenantNetCidr: 10.0.0.0/12  
ExternalNetCidr: 173.36.219.0/24  
# CIDR subnet mask length for provisioning network
ControlPlaneSubnetCidr: '24'
# Customize the IP ranges on each network to use for static IPs and VIPs
InternalApiAllocationPools: ['20.7.10.60', '20.7.10.140']
StorageAllocationPools: ['20.7.30.51', '20.7.30.150']
StorageMgmtAllocationPools: ['20.7.40.51', '20.7.40.150']
TenantAllocationPools: [{'start': '10.0.0.10', 'end': '10.15.255.250'}]
ExternalAllocationPools: [{'start': '173.36.219.207', 'end': '173.36.219.212'}]
ExternalInterfaceDefaultRoute: "173.36.219.1"
ControlPlaneDefaultRoute: 20.7.20.55
EC2MetadataIp: 20.7.20.55
DnsServers: ['8.8.8.8', '8.8.4.4']
StorageNetworkVlanID: 30
StorageMgmtNetworkVlanID: 40
InternalApiNetworkVlanID: 10
ExternalNetworkVlanID: 219
# TenantNetworkVlanID: 130
NeutronExternalNetworkBridge: ""

controller.yaml
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with a VLANs attached for the controller role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
type: string
  InternalApiIpSubnet:
    default: ''
    description: IP address/subnet on the internal API network
type: string
  StorageIpSubnet:
    default: ''
    description: IP address/subnet on the storage network
type: string
  StorageMgmtIpSubnet:
    default: ''
    description: IP address/subnet on the storage mgmt network
type: string
  TenantIpSubnet:
    default: ''
    description: IP address/subnet on the tenant network
type: string
ManagementIpSubnet: # Only populated when including environments/network-management.yaml
    default: ''
    description: IP address/subnet on the management network
type: string
ExternalNetworkVlanID:
    default: 219
    description: Vlan ID for the external network traffic.
type: number
InternalApiNetworkVlanID:
    default: 10
    description: Vlan ID for the internal_api network traffic.
type: number
StorageNetworkVlanID:
    default: 30
    description: Vlan ID for the storage network traffic.
type: number
StorageMgmtNetworkVlanID:
    default: 40
    description: Vlan ID for the storage mgmt network traffic.
type: number
# TenantNetworkVlanID:
#    default: 70
#    description: Vlan ID for the tenant network traffic.
#    type: number
# ManagementNetworkVlanID:
#    default: ''
#    description: Vlan ID for the management network traffic.
#    type: number
ExternalInterfaceDefaultRoute:
    default: ''
    description: default route for the external network
type: string
ControlPlaneSubnetCidr: # Override this via parameter_defaults
    default: '24'
    description: The subnet CIDR of the control plane network.
type: string
DnsServers: # Override this via parameter_defaults
    default: ['8.8.8.8','8.8.4.4']
    description: A list of DNS servers (2 max for some implementations) that will be added to resolv.conf.
type: comma_delimited_list
EC2MetadataIp: # Override this via parameter_defaults
    description: The IP address of the EC2 metadata server.
type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
                -
                  ip_netmask:
                    list_join:
                      - '/'
                      - {get_param: ControlPlaneIp}
                      - {get_param: ControlPlaneSubnetCidr}
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
            -
              type: ovs_bridge
              name: {get_input: bridge_name}
              use_dhcp: false
              members:
                -
                  type: interface
                  name: nic4
                  use_dhcp: false
                  primary: true
                -
                  type: vlan
                  vlan_id: {get_param: ExternalNetworkVlanID}
                  addresses:
                    -
                      ip_netmask: {get_param: ExternalIpAddress}
                  routes:
ip_netmask: 0.0.0.0/0

next_hop: {get_param: ExternalInterfaceDefaultRoute}

- type: ovs_bridge
  name: br-mgmt
  use_dhcp: false
  members:
    - type: interface
      name: nic3
      use_dhcp: false
      primary: true
    - type: vlan
      vlan_id: {get_param: InternalApiNetworkVlanID}
      addresses:
        - ip_netmask: {get_param: InternalApiIpSubnet}

- type: ovs_bridge
  name: br-storage-pub
  use_dhcp: false
  mtu: 9000
  members:
    - type: interface
      name: nic5
      use_dhcp: false
      mtu: 9000
      primary: true
    - type: vlan
      mtu: 9000
      vlan_id: {get_param: StorageNetworkVlanID}
      addresses:
        - ip_netmask: {get_param: StorageIpSubnet}

- type: ovs_bridge
  name: br-storage-clus
  use_dhcp: false
  mtu: 9000
  members:
    - type: interface
name: nic6
use_dhcp: false
mtu: 9000
primary: true
-
  type: vlan
  mtu: 9000
  vlan_id: {get_param: StorageMgmtNetworkVlanID}
  addresses:
  -
    ip_netmask: {get_param: StorageMgmtIpSubnet}
-
  type: ovs_bridge
  name: br-tenant
  use_dhcp: false
  members:
  -
    type: interface
    name: nic2
    use_dhcp: false
    primary: true

outputs:
OS::stack_id:
  description: The OsNetConfigImpl resource.
  value: {get_resource: OsNetConfigImpl}

compute.yaml
heat_template_version: 2015-04-30

description: >
  Software Config to drive os-net-config with 2 bonded nics on a bridge
  with a VLANs attached for the compute role.

parameters:
  ControlPlaneIp:
    default: ''
    description: IP address/subnet on the ctlplane network
    type: string
  ExternalIpSubnet:
    default: ''
    description: IP address/subnet on the external network
    type: string
  InternalApiIpSubnet:
    default: ''
description: IP address/subnet on the internal API network
type: string

StorageIpSubnet:
  default: ''
  description: IP address/subnet on the storage network
type: string

StorageMgmtIpSubnet:
  default: ''
  description: IP address/subnet on the storage mgmt network
type: string

TenantIpSubnet:
  default: ''
  description: IP address/subnet on the tenant network
type: string

ManagementIpSubnet: # Only populated when including environments/network-management.yaml
  default: ''
  description: IP address/subnet on the management network
type: string

InternalApiNetworkVlanID:
  default: 10
  description: Vlan ID for the internal_api network traffic.
type: number

StorageNetworkVlanID:
  default: 30
  description: Vlan ID for the storage network traffic.
type: number

# TenantNetworkVlanID:
#   default: 70
#   description: Vlan ID for the tenant network traffic.
#   type: number

# ManagementNetworkVlanID:
#   default: ''
#   description: Vlan ID for the management network traffic.
#   type: number

ControlPlaneSubnetCidr: # Override this via parameter_defaults
  default: '24'
  description: The subnet CIDR of the control plane network.
type: string

ControlPlaneDefaultRoute: # Override this via parameter_defaults
  description: '20.7.20.55'
type: string

DnsServers: # Override this via parameter_defaults
  default: ['8.8.8.8','8.8.4.4']
description: A list of DNS servers (2 max for some implementations) that will be added to resolv.conf.
  type: comma_delimited_list
EC2MetadataIp: # Override this via parameter_defaults
description: The IP address of the EC2 metadata server.
type: string

resources:
  OsNetConfigImpl:
    type: OS::Heat::StructuredConfig
    properties:
      group: os-apply-config
      config:
        os_net_config:
          network_config:
            -
              type: interface
              name: nic1
              use_dhcp: false
              dns_servers: {get_param: DnsServers}
              addresses:
                -
                  ip_netmask:
                    list_join: - '/'
                    - (get_param: ControlPlaneIp)
                    - (get_param: ControlPlaneSubnetCidr)
              routes:
                -
                  ip_netmask: 169.254.169.254/32
                  next_hop: {get_param: EC2MetadataIp}
                -
                  default: true
                  next_hop: {get_param: ControlPlaneDefaultRoute}
              type: ovs_bridge
              name: br-storage-pub
              use_dhcp: false
              mtu: 9000
              members:
                -
                  type: interface
                  name: nic4
                  use_dhcp: false
mtu: 9000
primary: true
-
  type: vlan
  mtu: 9000
  vlan_id: {get_param: StorageNetworkVlanID}
  addresses:
  -
    ip_netmask: {get_param: StorageIpSubnet}
-
  type: ovs_bridge
  name: br-mgmt
  use_dhcp: false
  members:
  -
    type: interface
    name: nic3
    use_dhcp: false
    primary: true
  -
    type: vlan
    vlan_id: {get_param: InternalApiNetworkVlanID}
    addresses:
    -
      ip_netmask: {get_param: InternalApiIpSubnet}
-
  type: ovs_bridge
  name: br-tenant
  use_dhcp: false
  members:
  -
    type: interface
    name: nic2
    use_dhcp: false
    primary: true

outputs:
OS::stack_id:
  description: The OsNetConfigImpl resource.
  value: {get_resource: OsNetConfigImpl}

The openstack overcloud deploy Command
  openstack overcloud deploy --templates \

-e /usr/share/openstack-tripleo-heat-templates/environments/network-isolation.yaml \
-e /home/stack/templates/network-environment.yaml \
-e /home/stack/templates/storage-environment.yaml \
-e /home/stack/templates/timezone.yaml \
--control-flavor control --compute-flavor compute --ceph-storage-flavor ceph-storage \
--compute-scale 4 --control-scale 3 --ceph-storage-scale 3 \
--libvirt-type kvm \
--ntp-server 171.68.38.66 \
--neutron-network-type vlan \
--neutron-tunnel-type vlan \
--neutron-bridge-mappings datacentre:br-ex,physnet-tenant:br-tenant \
--neutron-network-vlan-ranges physnet-tenant:250:749 \
--neutron-disable-tunneling --timeout 300 \
--log-file overcloud_new.log