Cisco ACI and OpenShift 3.11 Integration

Revised: May 10, 2021

New and Changed Information

The following table provides an overview of the significant changes up to this current release. The table does not provide an exhaustive list of all changes or of the new features up to this release.

Table 1: New Features and Changed Behavior in Cisco APIC

Cisco APIC Release Version	Feature	Description
5.0(x)	Support for VMware teaming policy.	You can configure VMware teaming policy when link aggregation groups (LAGs) are used. For more information, see the section Provisioning Cisco ACI to Work with OpenShift, on page 6.
5.0(x)	Support for creating multiple clusters in a Cisco Application Centric Infrastructure (ACI) tenant.	This release decouples the Cisco ACI Tenant name from the OpenShift cluster name—that is, the System ID provided in the acc-provision input file. The decoupling enables you to put one or more OpenShift clusters into one Cisco ACI tenant by deploying the cluster in a pre-existing tenant. For more information, see Provisioning Cisco ACI to Work with OpenShift, on page 6
4.2(x)	Kubernetes Replication Controller and OpenShift DeploymentConfig resources.	Support for adding endpoint group (EPG) and SecurityGroup (SG) annotations to Kubernetes ReplicationController and OpenShift DeploymentConfig resources. For more information, see the section Network Policy and EPGs, on page 17.
4.1(x)	Service account hardening	Every cluster is now configured with a dedicated administrative account that enables you to control read and write permissions for the whole fabric. For more information, see the section Service Account Hardening, on page 25.
4.0(1)	--	Added procedure on how to drain a node. For more information, see the procedure Draining a Node, on page 21.

Cisco APIC Release Version	Feature	Description
4.0(1)	Added support for OpenShift with ACI CNI plug-in to run nested on Red Hat OpenStack.	For more information, see the procedure Preparing OpenShift Nodes Running as OpenStack Virtual Machines, on page 9.
3.1(1)	• OpenShift on bare-metal servers • OpenShift nested on ESXi VMM-based VMs	The release supports integration of OpenShift on bare-metal servers and OpenShift nested on ESXi VMM-based VMs into the Cisco Application Centric Infrastructure (ACI).
3.1(1) and later	Service subnet advertisement	You can configure Cisco Application Centric Infrastructure (ACI) to advertise service subnets externally through a dynamic routing protocol. For more information, see the section Service Subnet Advertisement, on page 22.
3.1(1)	--	Added hardware dependency. For more information, see the section Hardware Requirements, on page 3.
3.1(1)	--	This guide was released.

Cisco ACI and OpenShift Integration

OpenShift is a container application platform that builds on top of Docker and Kubernetes that makes it accessible and easy for the developer to create applications, and a platform for operators that simplifies deployments of containers for both development and production workloads. Beginning with Cisco APIC Release 3.1(1), OpenShift can be integrated with Cisco Application Centric Infrastructure (ACI) by leveraging the ACI CNI Plugin.

To integrate Red Hat OpenShift with Cisco ACI you must perform a series of tasks. Some tasks are performed by the fabric administrator directly on APIC, while others are performed by the OpenShift Cluster administrator. Once you have integrated the Cisco ACI CNI Plugin for Red Hat OpenShift, you can use the APIC to view OpenShift endpoints and constructs in Cisco ACI.

This document provides the workflow for integrating OpenShift, Release 3.11, and specific instructions for setting up the Cisco APIC. However, it is assumed that you are familiar with OpenShift and containers and can install OpenShift. Specific instructions for installing OpenShift are beyond the scope of this document.

For documents related to OpenShift, Release 4.x, see the **ACI Virtualization** > **Virtualization — Containers** section on the Cisco APIC documentation landing page.

Hardware Requirements

This section provides the hardware requirements:

- Connecting the servers to Gen1 hardware of Cisco Fabric Extenders (FEXes) is not supported and results in a nonworking cluster.
- The use of symmetric policy-based routing (PBR) feature for load balancing external services requires the use of Cisco Nexus 9300-EX or -FX leaf switches.

  For this reason, the Cisco ACI CNI Plug-in is only supported for clusters that are connected to switches of those models.

Note UCS-B is supported as long as the UCS Fabric Interconnects are connected to Cisco Nexus 9300-EX or -FX leaf switches.

OpenShift Compatibility Matrix

Verify the compatibility matrix to review specific OpenShift and ACI release support:

https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/aci/virtualization/matrix/virtmatrix.html

Workflow for OpenShift Integration

This section provides a high-level description of the tasks required to integrate OpenShift into the Cisco ACI fabric.

- Prepare for the integration.

  Identify the subnets and VLANs that you will use in your network by following the instructions in the section Planning for OpenShift Integration, on page 4.

- Fulfill the required day-0 fabric configurations.

  Perform the required day-0 fabric configurations. Make sure that you perform all fabric configurations required prior to deploying the OpenShift cluster as per the section Prerequisites for Integrating OpenShift with Cisco ACI, on page 5.

- Configure the Cisco APIC for the OpenShift Cluster.

  Many of the required fabric configurations are performed directly by a provisioning tool (acc-provision). The tool is embedded in the plugin files from www.cisco.com. Once downloaded and installed modify the configuration file with the information from the planning phase and run the tool. For more information, see the section Provisioning Cisco ACI to Work with OpenShift, on page 6.

- If you plan to deploy the OpenShift cluster running nested on Red Hat OpenStack, see the section Preparing OpenShift Nodes Running as OpenStack Virtual Machines, on page 9.

- Considerations for OpenShift node network configurations.

  Set up networking for the node to support OpenShift installation. This includes configuring an uplink interface, subinterfaces, and static routes. See the section Preparing the OpenShift Nodes, on page 12.

- Install OpenShift and Cisco ACI containers.

  Use the appropriate method for your setup. See the section Installing OpenShift and Cisco ACI Containers, on page 15.

- Update the OpenShift router to use the ACI fabric.

  See the section Updating the Openshift Router to Use the ACI Fabric, on page 15.

- Verify the integration.

  Use the Cisco APIC GUI to verify that OpenShift has been integrated into the Cisco ACI. See the section Verifying the OpenShift Integration, on page 16.

Planning for OpenShift Integration

The OpenShift cluster requires various network resources all of which will be provided by the ACI fabric integrated overlay. The OpenShift cluster will require the following subnets:

- Node subnet—The subnet used for OpenShift control traffic. This is where the OpenShift API services are hosted. The acc-provisioning tool configures a private subnet. Ensure that it has access to the Cisco APIC management address.
- Pod subnet—The subnet from which the IP addresses of OpenShift pods are allocated. The acc-provisioning tool configures a private subnet.

  Note This subnet specifies the starting address for the IP pool that is used to allocate ip to pods as well as your ACI Bridge Domain IP. For example if you define it as 192.168.255.254/16, a valid configuration from an ACI perspective, your containers will not get an IP address as there are no free IPs after 192.168.255.254 in this subnet. We suggest to always use the first IP address in the POD subnet. In this example: 192.168.0.1/16.

- Node service subnet—The subnet used for internal routing of load-balanced service traffic. The acc-provisioning tool configures a private subnet.

  Note Similarly to the Pod subnet note, configure it with the first IP in the subnet.

- External service subnets—Pools from which load-balanced services are allocated as externally accessible service IPs.

  The externally accessible service IPs could be globally routable. Configure the next-hop router to send traffic to these IPs to the fabric. There are two such pools: One is used for dynamically allocated IPs and the other is available for services to request a specific fixed external IP.

All the above mentioned subnets must be specified on the acc-provisioning configuration file. The Node and POD subnet will be provisioned on corresponding ACI Bridge Domains that will be created by the provisioning tool. The endpoints on these subnets will be learnt as fabric endpoints and can be used to communicate directly with any other fabric endpoint without NAT, provided that contracts allow communication of course. The Node Service subnet and the External Service Subnet will not be seen as fabric endpoints, but will be instead used to manage ClusterIP and Load Balancer IP respectively, programmed on Open vSwitch via OpFlex. As mentioned before, the External Service subnet must be routable outside of the fabric.

OpenShift Nodes will need to be connected on an EPG using a VLAN encapsulation. PODs can connect to one or multiple EPGs and use either VLAN or VXLAN encapsulation. Additionally, PBR-based Load Balancing will require the use of a VLAN encapsulation to reach the opflex service endpoint IP of each Openshift node. The following VLAN IDs will therefore be required:

- Node VLAN ID—The VLAN ID used for the EPG mapped to a Physical domain for OpenShift nodes.
- Service VLAN ID—The VLAN ID used for delivery of load-balanced external service traffic.
- The Fabric Infra VLAN ID—The Infra VLAN used to extend OpFlex to the OVS on the OpenShift nodes.

In addition to providing network resources, read and understand the guidelines in the knowledge base article Cisco ACI and OpFlex Connectivity for Orchestrators.

Prerequisites for Integrating OpenShift with Cisco ACI

The following are required before you can integrate OpenShift with the Cisco ACI fabric:

- A working Cisco ACI fabric running a release that is supported for the desired OpenShift integration
- An attachable entity profile (AEP) set up with interfaces desired for the OpenShift deployment

  When running in nested mode, this is the AEP for the VMM domain on which OpenShift will be nested.

- A Layer 3 Outside connection, along with a Layer 3 external network to serve as external access
- Virtual routing and forwarding (VRF) configured

Note The VRF and L3Out in Cisco ACI that are used to provide outside connectivity to OpenShift external services can be in any tenant. The most common usage is to put the VRF and L3Out in the common tenant or in a tenant that is dedicated to the OpenShift cluster. You can also have separate VRFs, one for the OpenShift bridge domains and one for the L3Out, and you can configure route leaking between them.

- Any required route reflector configuration for the Cisco ACI fabric
- Ensure the subnet that is used for external services is routed by the next-hop router that is connected to the selected ACI L3Out interface. This subnet is not announced by default, so either static routes or appropriate configuration must be considered.

In addition, the OpenShift cluster must be up through the fabric-connected interface on all the hosts. The default route on the OpenShift nodes should be pointing to the ACI node subnet bridge domain. This is not mandatory, but it simplifies the routing configuration on the hosts and is the recommend configuration. If you do not follow this design, ensure the OpenShift node routing is correctly used so that all OpenShift cluster traffic is routed through the ACI fabric.

Provisioning Cisco ACI to Work with OpenShift

Use the acc_provision tool to provision the fabric for the OpenShift VMM domain and generate a `.yaml` file that OpenShift uses to deploy the required Cisco Application Centric Infrastructure (ACI) container components.

This tool requires a configuration file as input and will perform two actions as output:

- it will configure relevant parameters on the Cisco ACI fabric.
- it will generate a YAML file that OpenShift administrators can use to install the Cisco ACI CNI plug-in and containers on the cluster.

Note We recommended that when using ESXi nested for OpenShift hosts, you provision one OpenShift host for each OpenShift cluster for each ESXi server. Doing so ensures—in case of an ESXi host failure—that a single OpenShift node is affected for each OpenShift cluster.

Procedure

Step 1 Download the provisioning tool from cisco.com.

https://software.cisco.com/download/type.html?mdfid=285968390&i=rm

a) Click **APIC OpenStack and Container Plugins**.
b) Choose the package that you want to download.
c) Click **Download**.

Step 2 Generate a sample configuration file that you can edit.

Example:

```
terminal$ acc-provision--sample > aci-containers-config.yaml
```

Note Take note of the values if you are provisioning OpenStack to work with OpenShift.

The command generates a configuration file that looks like the following example:

```
#
# Configuration for ACI Fabric
#
aci_config:
  system_id: mykube             # Every opflex cluster must have a distict ID
  apic_hosts:                   # List of APIC hosts to connect for APIC API
- 10.1.1.101
  vmm_domain:                   # Kubernetes VMM domain configuration
    encap_type: vxlan           # Encap mode: vxlan or vlan
    mcast_range:                 # Every opflex VMM must use a distinct range
      start: 225.20.1.1
      end: 225.20.255.255

  # The following resources must already exist on the APIC,
  # they are used, but not created by the provisioning tool.
  aep: kube-cluster             # The AEP for ports/VPCs used by this cluster
  vrf:                                # This VRF used to create all kubernetes EPs
    name: mykube-vrf
    tenant: common              # This can be system-id or common
  l3out:
    name: mykube_l3out        # Used to provision external IPs
    external_networks:
    - mykube_extepg            # Used for external contracts

#
# Networks used by Kubernetes
#
net_config:
  node_subnet: 10.1.0.1/16      # Subnet to use for nodes
  pod_subnet: 10.2.0.1/16       # Subnet to use for Kubernetes Pods
  extern_dynamic: 10.3.0.1/24   # Subnet to use for dynamic external IPs
  node_svc_subnet: 10.5.0.1/24  # Subnet to use for service graph--This is not the same as
openshift_portal_net: Use different subnets.
  kubeapi_vlan: 4001            # The VLAN used by the physdom for nodes
  service_vlan: 4003            # The VLAN used by LoadBalancer services
  infra_vlan: 4093              # The VLAN used by ACI infra

#
# Configuration for container registry
# Update if a custom container registry has been setup
#
registry:
  image_prefix: noiro                   # e.g: registry.example.com/noiro
  # image_pull_secret: secret_name      # (if needed)
```

Notes:

- The Cisco Application Policy Infrastructure Controller (APIC) administrator must not modify the bridge domain configuration that is pushed by the acc-provisioning tool.
- Make sure you remove the following line from the `net_config` section:

  ```
  extern_static: 10.4.0.1/24    # Subnet to use for static external IPs
  ```

  This subnet is not used for OpenShift.
- Beginning Cisco APIC 5.0(1), you can configure VMware teaming policy when link aggregation groups (LAGs) are used. To do so, add the following to the configuration file:

  ```
  type: vmware
  elag_name: eLAG-name-used
  ```

- Starting in release Cisco APIC 5.0.(1), Cisco APIC resources created for the cluster are prepended with `aci-containers` and the `system_id` that are provided in the `.yaml` file.

  This naming convention ensures unique resource names when multiple clusters are provisioned under one tenant. Legacy clusters can use the `use_legacy_kube_naming_convention` field to enable the older naming convention. (Legacy clusters are those installed before Cisco ACI Container Network Interface (CNI) version 5.0.) However, you cannot use the `use_legacy_kube_naming_convention` field together with the `tenant` field.

- Beginning with Cisco APIC 5.0(x), you can configure a OpenShift cluster in an existing tenant. To do so, add the following to the configuration file:

```
aci_config:
    tenant:
        name: existing_tenant
```

  The **acc-provision** command creates the tenant if it does not yet exist. It also creates the application profile, bridge domain, endpoint groups (EPGs), and required contracts. However, the tenant is not deleted during unprovisioning.

Step 3 Edit the sample configuration file with the relevant values for each of the subnets, VLANs, etc., as appropriate to you planning and then save the file.

Step 4 Provision the Cisco ACI fabric.

Example:

```
acc-provision -f openshift-<version> -c aci-containers-config.yaml -o aci-containers.yaml \
-a -u [apic username] -p [apic password]
```

This command generates the file `aci-containers.yaml` that you use after installing OpenShift. It also creates the files `user-[system id].key` and `user-[system id].crt` that contain the certificate that is used to access Cisco APIC. Save these files in case you change the configuration later and want to avoid disrupting a running cluster because of a key change.

Notes:

- The file `aci-containers.yaml` is security sensitive. It contains keys necessary for connecting to the Cisco APIC administration API.
- Currently, the provisioning tool supports only the installation of a single OpenShift cluster for each Cisco ACI tenant on a single or multipod Cisco ACI fabric.

Step 5 (Optional): Advanced optional parameters can be configured to adjust to custom parameters other than the Cisco ACI default values or base provisioning assumptions:

- If your VMM's multicast address for the fabric is different from 225.1.2.3, you can configure it adding the following:

```
aci_config:
  vmm_domain:
     mcast_fabric: 225.1.2.3
```

- If you are using VLAN encapsulation, you can specify vlan-pool for it as follows:

```
aci_config:
  vmm_domain:
    encap_type: vlan
    vlan_range:
      start: 10
      end: 25
```

- If you want to use an existing User, or Key, or Cert, add the following:

```
aci_config:
  sync_login:
    username: <name>
    certfile: <pem-file>
    keyfile: <pem-file>
```

- If provisioning in a system nested inside of virtual machines, enter the name of an existing preconfigured VMM domain in Cisco ACI into the `aci_config` section under the `vmm_domain` of the configuration file.

```
nested_inside:
    type: vmware
    name: myvmware
```

Preparing OpenShift Nodes Running as OpenStack Virtual Machines

This section describes how to provision a Red Hat OpenStack installation that uses the Cisco Application Centric Infrastructure (ACI) OpenStack plug-in in order to work with a nested Red Hat OpenShift cluster using the Cisco ACI CNI Plug-in.

Before you begin

- Ensure that you have a running Red Hat OpenStack installation using the required release of the Cisco ACI OpenStack plugin.

  For more information, see the Cisco ACI Installation Guide for Red Hat OpenStack Using OSP Director.

  When installing the Overcloud, the ACIOpflexInterfaceType and ACIOpflexInterfaceMTU should be set as follows:

  ```
  ACIOpflexInterfaceType: ovs
  ```

  ```
  ACIOpflexInterfaceMTU: 8000
  ```

Procedure

Step 1 Create the neutron network that will be used to connect the OpenStack instances that will be running the OpenShift nodes:

```
neutron net-create  <name> \
--apic:nested-domain-name openshift-domain \
--apic:nested-domain-type openshift \
--apic:nested_domain_infra_vlan <infra-vlan> \
--apic:nested_domain_node_network_vlan <node_vlan> \
--apic:nested_domain_service_vlan <service_vlan> \
--apic:nested_domain_allowed_vlans "{'vlan_ranges': [{'start': <vlan-id-1>, 'end': <vlan-id-2>}]}"
```

Notes:

- Ensure you use the same values as you did in step 2 of the Provisioning Cisco ACI to Work with OpenShift, on page 6.

  The node_vlan and service_vlan must match those configured in the acc-provisioning tool settings.

- The nested_domain_allowed_vlans is only required if the Cisco ACI CNI Plug-in will be deployed in VLAN mode. This is not required when using VXLAN mode.

- When using VLAN mode for the Cisco ACI CNI Plugin, the VLAN range must be configured correctly on the acc-provisioning tool configuration file. This will result in a VLAN pool configured on Cisco Application Policy Infrastructure Controller (APIC), ensure that there is no VLAN overlap with other VLAN pools used for OpenStack.

Example:

```
neutron net-create os-net \
--apic:nested-domain-name openshift-domain \
--apic:nested-domain-type openshift \
--apic:nested_domain_infra_vlan 4093 \
--apic:nested_domain_node_network_vlan 4001 \
--apic:nested_domain_service_vlan 4003 \
--apic:nested_domain_allowed_vlans "{'vlan_ranges': [{'start': 4004, 'end': 4006}]}"

{"network": {"apic:nested_domain_name": "openshift domain", "apic:nested_domain_node_network_vlan":

4001, "apic:nested_domain_infra_vlan": 4093, "apic:distinguished_names": {"VRF":
"uni/tn-common/ctx-lab_UnroutedVRF", "EndpointGroup": "uni/tn-prj_9fb5e09e0d1e4bf6b81d4d3
54eeccd4a/ap-OpenStack/epg-net_5ffb14fc-99c3-4d2b-b9c9-d7ab90af459a", "BridgeDomain":
"uni/tn-prj_9fb5e09e0d1e4bf6b81d4d354eeccd4a/BD-net_5ffb14fc-99c3-4d2b-b9c9-d7ab90af459a"},
"ipv6_address_scope": null, "revision_number": 3, "port_security_enabled": true, "mtu":
1500, "apic:nested_domain_service_vlan": 4003, "id": "5ffb14fc-99c3-4d2b-b9c9-d7ab90af459a",
"apic:synchronization_state": "synced", "router:external": false, "availability_zone_hints":
[], "availability_zones": [], "provider:physical_network": "physnet1",
"apic:nested_domain_allowed_vlans": [4001, 4003, 4004, 4005, 4006, 4093], "ipv4_address_scope":
null, "shared": false, "project_id": "9fb5e09e0d1e4bf6b81d4d354eeccd4a", "status":
"ACTIVE", "subnets": [], "description": "", "tags": [], "updated_at":
"2018-04-26T18:44:17Z", "provider:segmentation_id": null, "apic:nested_domain_type":
"openshift", "name": "os-net", "admin_state_up": true, "tenant_id":
"9fb5e09e0d1e4bf6b81d4d354eeccd4a", "created_at": "2018-04-26T18:44:17Z",
"provider:network_type": "opflex"}}
```

Step 2 Launch the OpenStack VMs to host OpenShift.

Sub-interfaces have to be created for the node network, infra network and the appropriate MTUs have to be set.

Examples of the network-interface configuration files are shown in the following text. In the examples, we assume that the Cisco ACI infra VLAN is 3085, kube node VLAN is 2031, and VLAN 4094 is the VLAN used from OpenStack to provide IP and metadata to the VM.

Note VLAN 4094 is currently hardcoded for the OpenShift on OpenStack nested plug-in configuration and should not be changed.

```
$ cat /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE=eth0
ONBOOT=yes
HOTPLUG=no
DEVICETYPE=ethernet
BOOTPROTO=none

$ cat /etc/sysconfig/network-scripts/ifcfg-eth0.2031
VLAN=yes
TYPE=Vlan
DEVICE=eth0.2031
PHYSDEV=eth0
VLAN_ID=2031
REORDER_HDR=yes
GVRP=no
MVRP=no
BOOTPROTO=none
IPADDR=12.11.11.100
PREFIX=16
GATEWAY=12.11.0.1
MTU=8000
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
```

```
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eth0.2031
ONBOOT=yes
PROXY_METHOD=none
BROWSER_ONLY=no
DNS1=8.8.8.8

$ cat /etc/sysconfig/network-scripts/ifcfg--eth0.3085
DEVICE=eth0.3085
NAME=eth0.3085
ONBOOT=yes
PEERDNS=yes
TYPE=vlan
PHYSDEV=eth0
BOOTPROTO=dhcp
VLAN=yes
ONPARENT=yes
MTU=8000

$ cat /etc/sysconfig/network-scripts/ifcfg-eth0.4093
VLAN=yes
TYPE=Vlan
PHYSDEV=eth0
VLAN_ID=4093
ONBOOT=yes
BOOTPROTO=dhcp
DEFROUTE=no
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=eth0.4093
MTU=8000

$ cat /etc/sysconfig/network-scripts/route-opflex-conn
ADDRESS0=224.0.0.0
NETMASK0=240.0.0.0
METRIC0=1000

$ cat /etc/sysconfig/network-scripts/route-eth0.3085
ADDRESS0=224.0.0.0
NETMASK0=240.0.0.0
GATEWAY0=0.0.0.0
METRIC0=1000
```

The configurations in the previous text can be added or updated after initial VM bringup.

The dhcp request for eth0.4093 fails if the dhcp client is not configured. The following configuration needs to be added to the `/etc/dhcp/dhcpclient.conf`:

```
$ cat /etc/dhcp/dhclient-eth0.4093.conf
send dhcp-client-identifier 01:mac-address;

Obtain IP address for eth0.4093
```

Example:

```
sudo dhclient eth0.4093
```

The following configuration needs to be added to the `/etc/dhcp/dhcpclient.conf`:

```
$ more /etc/dhcp/dhclient-eth0.3085.conf
send dhcp-client-identifier 01:mac-address-of-the-VM;

Obtain IP address for eth0.3085
```

Example:

```
sudo dhclient eth0.3085
```

Step 3 For OpenShift to work correctly, the default route on the VM needs to be configured to point to the subinterface created for the Kubernetes node network. In the above example, the infra network is 10.0.0.0/16, the node network is 12.11.0.0/16 and the Neutron network is 1.1.1.0/24. Note that the metadata subnet 169.254.0.0/16 route is added to the eth0.4094 interface.

Example:

```
$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         12.11.0.1       0.0.0.0         UG    400    0        0 eth0.2031
1.1.1.0         0.0.0.0         255.255.255.0   U     401    0        0 eth0.4094
10.0.0.0        0.0.0.0         255.255.0.0     U     402    0        0 eth0.3085
12.11.0.0       0.0.0.0         255.255.0.0     U     400    0        0 eth0.2031
169.254.0.0     1.1.1.1         255.255.0.0     UG    401    0        0 eth0.4094
172.17.0.0      0.0.0.0         255.255.0.0     U     0      0        0 docker0
224.0.0.0       0.0.0.0         240.0.0.0       U     1000   0        0 eth0.3085
```

You are now ready to install OpenShift. Skip the next section and proceed to Installing OpenShift and Cisco ACI Containers, on page 15.

Preparing the OpenShift Nodes

After you provision Cisco Cisco Application Centric Infrastructure (ACI), you prepare networking for the OpenShift nodes.

Procedure

Step 1 Configure your uplink interface with NIC bonding or not, depending on how your AEP is configured.

Set the MTU on this interface to at least 1600, preferably 9000.

Note The Node Uplink Interface MTU needs to account for the VXLAN header overhead of the container traffic. By default, the container interface MTU is 1500 but it can be increased up to 8900 through the interface_mtu parameter.

Step 2 Create a subinterface on your uplink interface on your infra VLAN.

Configure this subinterface to obtain an IP address using DHCP. Set the MTU on this interface to 1600.

Step 3 Configure a static route for the multicast subnet 224.0.0.0/4 through the uplink interface that is used for VXLAN traffic.

Step 4 Create a subinterface on your uplink interface on your node VLAN. For example, which is called `kubeapi_vlan` in the configuration file.

Configure an IP address on this interface in your node subnet. Then set this interface and the corresponding node subnet router as the default route for the node.

Note Many OpenShift installer tools look specifically for the default route to choose interfaces for API server traffic and other traffic. It's possible to install with the default route on another interface. To do this, you set up static routes into this interface and override your installer configuration. However, we recommend setting up the default route through the node uplink.

Step 5 Create the `/etc/dhcp/dhclient-eth0.4093.conf` file with the following content, inserting the MAC address of the Ethernet interface for each server on the first line of the file:

Example:

Note If you have a single interface, you could name the file just `dhclient.conf` and not need the interface name, as in `dhclient-eth0.4093.conf`.

```
send dhcp-client-identifier 01:<mac-address of infra VLAN interface>;
request subnet-mask, domain-name, domain-name-servers, host-name;
send host-name <server-host-name>;

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;
```

The network interface on the infra VLAN requests a DHCP address from the Cisco APIC Infrastructure network for OpFlex communication. The server must have a dhclient configuration for this interface to receive all the correct DHCP options with the lease.

If you need information on how to configure a VPC interface for the OpenShift servers, see "Manually Configure the Host vPC" in the Cisco ACI with OpenStack OpFlex Deployment Guide for Red Hat on Cisco.com.

Note The infra VLAN interface in your environment may be a basic Linux-level subinterface, such as eth0.4093.

Step 6 If you have a separate management interface for the node being configured, configure any static routes required to access your management network on the management interface.

Step 7 Ensure that Open vSwitch (OVS) is not running on the node.

Step 8 Informational: Here is an example of the interface configuration (/etc/network/interfaces):

```
# Management network interface (not connected to ACI)
# /etc/sysconfig/network-scripts/ifcfg-eth0
NAME=eth0
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
IPADDR=192.168.66.17
NETMASK=255.255.255.0
PEERDNS=no
DNS1=192.168.66.1

# /etc/sysconfig/network-scripts/route-eth0
ADDRESS0=10.0.0.0
NETMASK0=255.0.0.0
GATEWAY0=192.168.66.1

# Interface connected to ACI
# /etc/sysconfig/network-scripts/ifcfg-eth1
```

```
NAME=eth1
DEVICE=eth1
ONBOOT=yes
BOOTPROTO=none
TYPE=Ethernet
MTU=1600

# ACI Infra VLAN
# /etc/sysconfig/network-scripts/ifcfg-4093
VLAN=yes
TYPE=Vlan
PHYSDEV=eth1
VLAN_ID=4093
REORDER_HDR=yes
BOOTPROTO=dhcp
DEFROUTE=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=4093
DEVICE=eth1.4093
ONBOOT=yes
MTU=1600

# /etc/sysconfig/network-scripts/route-4093
ADDRESS0=224.0.0.0
NETMASK0=240.0.0.0
METRIC0=1000

# Node Vlan
# /etc/sysconfig/network-scripts/ifcfg-node-vlan-4001
VLAN=yes
TYPE=Vlan
PHYSDEV=eth1
VLAN_ID=4001
REORDER_HDR=yes
BOOTPROTO=none
IPADDR=12.1.0.101
PREFIX=24
GATEWAY=12.1.0.1
DNS1=192.168.66.1
DEFROUTE=yes
IPV6INIT=no
NAME=node-vlan-4001
DEVICE=eth1.4001
ONBOOT=yes
MTU=1600
```

Step 9 Add the following Iptables rule to each host to allow IGMP traffic:

```
$ iptables -A INPUT -p igmp -j ACCEPT
```

To make this change persistent across reboots, add the command either to `/etc/rc.d/rc.local` or to a cron job that runs after reboot.

Step 10 Tune the `igmp_max_memberships` kernel parameter.

The Cisco ACI Container Network Interface (CNI) plug-in gives you the flexibility to place Namespaces, Deployment, and PODs into dedicated endpoint groups (EPGs).

To ensure that Broadcast, Unknown Unicast and Multicast traffic (BUM) is not flooded to all the EPGs, Cisco ACI allocates a dedicated multicast address for BUM traffic replication for every EPG that is created.

Recent kernel versions set the default for the `igmp_max_memberships` parameter to 20, limiting the maximum number of EPGs that can be utilized to 20. To have more than 20 EPGs, you can increase the `igmp_max_memberships` with the following steps:

a) Check the current configured maximum with the following command:

```
sysctl net.ipv4.igmp_max_memberships
```

b) Edit the `/etc/sysctl.d/99-sysctl.conf` file and add the following line:

```
net.ipv4.igmp_max_memberships = Max_Number_Of_EPGs
```

c) Restart the node or issue the following command:

```
sysctl -p
```

d) Verify that the new limit is correctly configured with the following command:

```
sysctl net.ipv4.igmp_max_memberships
```

Note For more details about the `igmp_max_memberships` kernel parameter, see the article "Documentation for /proc/sys" on The Linux Kernel Archives website. Go to the Documentation section and search for "Documentation for `pro/sys/net`."

Installing OpenShift and Cisco ACI Containers

After you provision Cisco Application Centric Infrastructure (ACI) and prepare the OpenShift nodes, you can install OpenShift and Cisco ACI containers. You can use any installation method appropriate to your environment. We recommend that you use this procedure to install the OpenShift and Cisco ACI containers.

Note When installing OpenShift, ensure that the API server is bound to the IP addresses on the node subnet and not to management or other IP addresses. Issues with node routing table configuration, API server advertisement addresses, and proxy are the most common problems during installation. If problems occur, check these issues first.

Procedure

Install OpenShift, following the installation procedure documented in the article Planning your installation for the 3.11 release on the Red Hat OpenShift website.

Use the configuration overrides for Cisco ACI CNI Ansible inventory configuration on the GitHub website: https://github.com/openshift/openshift-ansible/tree/release-3.11/roles/aci.

Note Make sure you provide the`aci_deployment_yaml_file` with the path to the yaml generated with acc-provision tool in the section Provisioning Cisco ACI to Work with OpenShift, on page 6.

Updating the Openshift Router to Use the ACI Fabric

This section describes how to update the OpenShift router to use the ACI fabric.

Procedure

Step 1 Remove the old router, enter the following commands:

Example:

```
oc delete svc router
oc delete dc router
```

Step 2 Create the container networking router, enter the following command:

Example:

```
oc adm router --service-account=router --host-network=false
```

Step 3 Expose the router service externally, enter the following command:

Example:

```
oc patch svc router -p '{"spec":{"type": "LoadBalancer"}}'
```

Unprovisioning OpenShift from the ACI Fabric

This section describes how to unprovision OpenShift from the ACI fabric.

Procedure

To unprovision, enter the following command:

Example:

```
acc-provision -c aci-containers-config.yaml -o aci-containers.yaml -a -d -u [apic username] -f
openshift-<version> -p [apic
password]
```

This command unprovisions the resources that have been allocated for this OpenShift.

This also deletes the tenant. If you are using a shared tenant, this is very dangerous.

Uninstalling the CNI Plug-In

This section describes how to uninstall the CNI plug-in.

Procedure

Uninstall the CNI plug-in using the following command:

Example:

```
oc delete -f aci-containers.yaml
```

Verifying the OpenShift Integration

After you have performed the previous steps, you can verify the integration in the Cisco APIC GUI. The integration creates a tenant, three EPGs, and a VMM domain.

Procedure

Step 1 Log in to the Cisco APIC.

Step 2 Go to **Tenants** > ***tenant***.

The tenant should have the name that you specified in the configuration file that you edited and used in installing OpenShift and the ACI containers.

Step 3 In the tenant navigation pane, expand the following: ***tenant*** > **Application Profiles** > ***application profile*** > **Application EPGs**.

You should see three folders inside the **Application EPGs** folder:

- **kube-default**—The default EPG for containers that are otherwise not mapped to any specific EPG.
- **kube-nodes**—The EPG for the OpenShift nodes.
- **kube-system**—The EPG for the kube-system OpenShift namespace. This typically contains the kube-dns pods that provide DNS services for a OpenShift cluster.

Step 4 In the tenant navigation pane, expand the **Networking** and **Bridge Domains** folders.

You should see two bridge domains:

- **node-bd**—The bridge domain used by the node EPG.
- **pod-bd**—The bridge domain used by all pods.

Step 5 If you deploy OpenShift with a load balancer, go to **Tenants** > **common**, expand **L4-L7 Services**, and perform the following steps:

a) Open the **L4-L7 Service Graph Templates** folder; you should see a template for OpenShift.
b) Open the **L4-L7 Devices** folder; you should see a device for OpenShift.
c) Open the **Deployed Graph Instances** folder; you should see an instance for OpenShift.

Step 6 Go to **VM Networking** > **Inventory**.

Step 7 In the **Inventory** navigation pane, expand the **OpenShift** folder.

You should see that a VMM domain, with the name that you provided in the configuration file, is created and that the domain contains a folder called **Nodes** and a folder called **Namespaces**.

Using Policy

Network Policy and EPGs

The Cisco ACI and OpenShift integration was designed to offer a highly flexible approach to policy. It was based on two premises: that OpenShift templates not need to change when they run on Cisco ACI, and that developers not be forced to implement any APIC configuration to launch new applications. At the same time, the solution optionally exposes Cisco ACI EPGs and contacts to OpenShift users if they choose to leverage them for application isolation.

By default, Cisco plug-ins create an EPG and a bridge domain in APIC for the entire OpenShift cluster. All pods by default are attached to the new EPG, has no special properties. The container team or the network team do not need to take any further action for a fully functional OpenShift cluster—as one might find in a public cloud environment. Additionally, security enforcement can occur based on usage of the OpenShift NetworkPolicy API. NetworkPolicy objects are transparently mapped into Cisco ACI and enforced for containers within the same EPG and between EPGs.

The following is an example of NetworkPolicy in OpenShift:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  ingress:
  - from:
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
```

However, in many cases, you may want to leverage EPGs and contracts in a more flexible way to define policy. Additional EPGs and contracts may be created either directly through the APIC.

To move namespaces, deployments, or pods into these EPGs, a OpenShift user simply applies an annotation to any of these objects specifying the application profile and EPG. The running pods automatically shift to the new EPG, and any configured contracts are applied. In this model, it is still possible to use OpenShift NetworkPolicy, which is honored regardless of how pods are mapped to EPGs.

OpenShift supports a DeploymentConfig resource that defines the template for a pod and manages deploying new images or configuration changes. A single deployment configuration is usually analogous to a single microservice. It can support many different deployment patterns, including full restart, customizable rolling updates, and fully custom behaviors, as well as pre- and post-deployment hooks. Each individual deployment is represented as a ReplicationController. The same annotation support that was available earlier for Deployments is available for ReplicationController and—as a result—is available to OpenShift DeploymentConfig as well.

For details about the DeploymentConfig resource, see the page **v1.DeploymentConfig** in the REST API Reference section on the OpenShift website.

For details about the ReplicationController, see the concept **ReplicationController** in the Documentation section of the Kubernetes website.

Mapping to Cisco APIC

Each OpenShift cluster is represented by a tenant within Cisco APIC. By default, all pods are placed in a single EPG created automatically by the plug-ins. However, it is possible to map a namespace, deployment, or pod to a specific application profile and EPG in OpenShift through OpenShift annotations.

While this is a highly flexible model, there are three typical ways to use it:

- EPG=OpenShift Cluster—This is the default behavior and provides the simplest solution. All pods are placed in a single EPG, kube-default.
- EPG=Namespace—This approach can be used to add namespace isolation to OpenShift. While OpenShift does not dictate network isolation between namespaces, many users may find this desirable. Mapping EPGs to namespaces accomplishes this isolation.
- EPG=Deployment—A OpenShift deployment represents a replicated set of pods for a microservice. You can put that set of pods in its EPG as a means of isolating specific microservices and then use contracts between them.

Creating and Mapping an EPG

Use this procedure to create and EPG, using annotations to map namespaces or deployments into it.

For information about EPGs and bridge domains, see the *Cisco APIC Basic Configuration Guide*.

Procedure

Step 1 Log in to Cisco APIC.

Step 2 Create the EPG and add it to the bridge domain kube-pod-bd.

Step 3 Attach the EPG to the VMM domain.

Step 4 Configure the EPG to consume contracts in the OpenShift tenant:

- Consume: arp, dns, kube-api

  kube-api is optional.

- Provide: arp, health-check

  Optional: You can use contract inheritance and have kub-defaul as your "EPG Contract Master". For more information about "About Contract Inheritance", see the *Cisco Application Centric Infrastructure Fundamentals Guide* at:

  https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/aci-fundamentals/b_ACI-Fundamentals/b_ACI-Fundamentals_chapter_010001.html#id_51661

Step 5 Configure the EPG to consume contracts in the common tenant:

Consume: kube-l3out-allow-all (optional)

Step 6 Create any contracts you need for your application and provide and consume them as needed.

Step 7 Apply annotations to the namespaces or deployments.

You can apply annotations in three ways:

- Through oc:
  - Deployment example:

    ```
    oc --namespace=namespace annotate deployment deployment
    opflex.cisco.com/endpoint-group='{"tenant":"tenant","app-profile":"app-profile","name":"EPG"}'
    ```

  - Namespace example:

    ```
    oc annotate namespace namespace
    opflex.cisco.com/endpoint-group='{"tenant":"tenant","app-profile":"app-profile","name":"EPG"}'
    ```

- Through acikubectl:
  - Deployment example:

    ```
    acikubectl set default-eg deployment deployment -n namespace -t kube -a app-profile -g EPG
    ```

  - Namespace example:

    ```
    acikubectl set default-eg namespace namespace -t kube -a app-profile -g EPG
    ```

- Through the .yaml file:

```
annotations:
opflex.cisco.com/endpoint-group: {
"tenant":"tenant",
"app-profile":"app-profile",
"name":"EPG"
}
```

Note OpenShift system infrastructure components such as the OpenShift router and the registry are non annotated pods that are discovered by ACI in the default EPG. When annotating objects to map to an EPG other than default, an ACI contract to allow traffic between the default EPG and the new EPG will need to be created for the router to access pods in the new EPG.

The acikubectl Command

The **acikubectl** command is a command-line utility that provides an abbreviated way to manage Cisco ACI policies for OpenShift objects and annotations. It also enables you to debug the system and collect logs.

The **acikubectl** command includes a **--help** option that displays descriptions of the command's supported syntax and options, as seen in the following example:

```
acikubectl --help

Available Commands:
  debug       Commands to help diagnose problems with ACI containers
  get         Get a value
  help        Help about any command
  set         Set a value
```

Load Balancing External Services

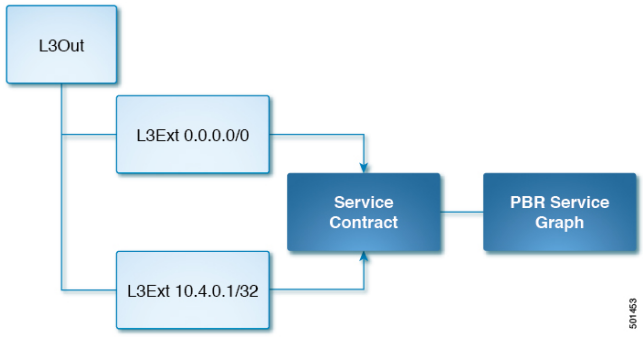
For OpenShift services that are exposed externally and need to be load balanced, OpenShift does not handle the provisioning of the load balancing. It is expected that the load balancing network function is implemented separately. For these services, Cisco ACI takes advantage of the symmetric policy-based routing (PBR) feature available in the Cisco Nexus 9300-EX or FX leaf switches in ACI mode.

On ingress, incoming traffic to an externally exposed service is redirected by PBR to one of the OpenShift nodes that hosts at least one pod for that particular service. Each node hosts a special service endpoint that handles the traffic for all external services hosted for that endpoint. Traffic that reaches the service endpoint is not rewritten by the fabric, so it retains its original destination IP address. It is not expected that the OpenShift pods handle traffic that is sent to the service IP address, so Cisco ACI performs the necessary network address translation (NAT).

If a OpenShift worker node contains more than one IP pod for a particular service, the traffic is load balanced a second time across all the local pods for that service.

A simplified view of the Cisco ACI policy model required for the north-south load balancer is shown in the following illustration.

Figure 1: Load Balancer Logical Path



In the example, service IP 10.4.0.1 is exposed as an external service. It is presented as a /32 Layer 3 external network. This network provides a contract that is consumed by the default /0 Layer 3 external network. Traffic that comes into the fabric from the outside hits this contract and is redirected by the service graph to the correct set of OpenShift service endpoints.

The Cisco ACI OpenShift integration components are responsible for automatically writing the Cisco ACI policy that implements the external service into Cisco ACI.

Load Balancing External Services Across Multiple L3Out

ACI CNI supports load balancing external services across multiple L3Outs. However, the service contract must be configured manually on the external EPG in the consumer direction. Also, if the L3Out is in a different tenant and/or VRF, then you need to change the contract scope accordingly. This can be done by annotating the service with "`opflex.cisco.com/ext_service_contract_scope=<scope>`". If the **ext_service_contract_scope** annotation is not set, or if it set as an empty string (i.e. `opflex.cisco.com/ext_service_contract_scope=""`) then, the contract scope is set to context (VRF). Setting any scope other than context will also set the "import-security" attribute for the subnet associated with the External EPG that is consuming the contract. This allows the service to be reachable across the VRFs.

Draining a Node

This section describes how to drain the pods on a node.

Procedure

Step 1 Drain the node:

Example:

```
oc adm drain --ignore-daemonsets <nodename>
```

Step 2 After the drain is finished, scale the controller down to 0 replicas:

Example:

```
oc scale deployment aci-containers-controller --replicas=0 -n aci-containers-system
```

Step 3 Remove the old annotations from the node by editing the node description and remove the `opflex.cisco.com` annotations and save the changes:

Example:

```
oc edit node <nodename>
```

Step 4 Find and delete the hostagent pod for the drained node.

Example:

```
oc delete pod aci-containers-host-xxxx -n aci-containers-system
```

Note The pod remains in **Terminating** state until you uncordon it.

Step 5 Bring the controller up again:

Example:

```
oc scale deployment aci-containers-controller --replicas=1 -n aci-containers-system
```

Step 6 Bring back the node:

Example:

```
oc adm uncordon <nodename>
```

Step 7 Verify annotations are present and that the hostagent pod is running:

Example:

```
oc describe node <nodename> | grep cisco
oc get pods -n aci-containers-system -o wide | grep <nodename>
```

Moving a Node Between ESXi Hosts in Nested Mode

Follow the procedure in this section to move a node from one ESXI host to another when you have nested ESXi hosts.

Note Using VMware vMotion to move nodes is not supported for nested ESXi hosts. Using VMware vMotion under these circumstances results in a loss of connectivity on the node that you are trying to move.

Procedure

Step 1 Create a new VM and add it to the cluster.

Step 2 Drain the old VM.

Step 3 Delete the old VM.

Service Subnet Advertisement

By default, service subnets are not advertised externally, requiring that external routers be configured with static routes. However, you can configure Cisco Application Centric Infrastructure (ACI) to advertise the service subnets through a dynamic routing protocol.

To configure and verify service subnet advertisement, complete the procedures in this section.

Configuring Service Subnet Advertisement

Complete the following steps to configure service subnet advertisement.

Note Perform this procedure for all border leafs that are required to advertise the external subnets.

Procedure

Step 1 Add routes to null to the subnets:

a) Log in to Cisco Application Policy Infrastructure Controller (APIC).

b) Go to **Tenant** > **Networking** > **External Routed Networks** > ***your_L3-Out*** > **Logical Node Profile** > ***your_node_profile***.

c) In the node profile work pane, double-click a node.

d) In the **Node Association** dialog box, in the **Static Routes** area, click the **+** (plus) icon.

e) In the **Create Static Route** dialog box, in the **Prefix** field, enter the static route IP address and network mask that is assigned to the outside network.

Note Add a static route for external dynamic subnets. Leave the **Next Hop Addresses** field empty. A null interface is automatically created.

f) Click **Submit**.

In the **Node Association** dialog box, click **Close**.

g) In the **Node Association** dialog box, in the **Static Routes** area, repeat steps 1.d/1.4 through 1.f/1.6 for each node.

Step 2 Create match rules for the route map and add the subnet.

a) Go to **Tenant** > **Networking** > **External Routed Networks** > **Match Rules for Route Map**.

b) Right-click **Match Rules for Route Map** and choose **Create Match Rule for a Route Map**.

c) In the **Create Match Rule** dialog box, in the **Name** field, enter a name for the match rule.

d) In the **Match Prefix** area, click the **+** (plus) icon.

e) In the **IP** field, enter the static route IP address and network mask that you entered in step 1e.

f) Click **Update** and then click **Submit**.

Step 3 Create a route map for the L3 Out, add a context to it, and choose the match rules.

a) Go to **Tenant** > **Networking** > **External Routed Networks** > ***your_L3-Out*** > **Route Maps/Profiles**.

b) Right-click **Route Maps/Profiles**, and choose **Create Route Maps/Profiles**.

c) In the **Create Route Map** dialog box, from the **Name** drop-down list, choose **default-export**.

d) In the **Type** area, make sure that **Match Prefix AND Routing Policy** is chosen.

e) In the **Contexts** area, click the **+** (plus) icon.

f) In the **Create Route Control Context** dialog box, with the **Order** selector, choose **0**.

g) In the **Name** field, enter a name for the policy context.

h) In the **Action** area, make sure that **Permit** is chosen.

i) From the **Match Rule** drop-down list, choose the rule that you created in Step 2c.

j) Click **OK** and then click **Submit**.

What to do next

Verify that the external routers have been configured with the external routes. See the section Verifying Service Subnet Advertisement, on page 24.

Verifying Service Subnet Advertisement

Use NX-OS style CLI to verify that the external routers have been configured with the external routes. Perform the commands for each for the border leafs.

Before you begin

Ensure that your border leafs are configured with routes to Null0 for extern_static and extern_dynamic subnets (10.3.0.1/24 in the following example):

```
fab2-apic1# fabric 203 show ip route vrf common:os | grep null0 -B1
10.3.0.1/24, ubest/mbest: 1/0
    *via , null0, [1/0], 04:31:23, static
```

Procedure

Step 1 Check the route maps applied to your dynamic routing protocol that permits the advertisement of the subnets.

a) Find your route map for static routes:

Example:

```
fabric 203 show ip ospf vrf common:k8s  | grep route-map
 Table-map using route-map exp-ctx-2981889-deny-external-tag
   static route-map exp-ctx-st-2981889
   direct route-map exp-ctx-st-2981889
   bgp route-map exp-ctx-proto-2981889
   eigrp route-map exp-ctx-proto-2981889
   coop route-map exp-ctx-st-2981889
```

Step 2 Find the specific route for each of the nodes, looking for entries that match the name of the match rule:

Example:

In the example, `os-svc-export` is the name of the match rule in Cisco Application Policy Infrastructure Controller (APIC).

```
fabric 203 show route-map exp-ctx-st-2981889 | grep os-svc-export
   ip address prefix-lists: IPv4-st19-2981889-exc-ext-out-os-svc-export2os-svc-export0os-svc-export-dst

fabric 204 show route-map  exp-ctx-proto-2981889 | grep os-svc-export
    ip address prefix-lists:
IPv4-proto19-2981889-exc-ext-out-os-svc-export2os-svc-export0os-svc-export-dst
```

Step 3 Verify that the IP addresses are correct for each of the nodes:

Example:

```
fab2-apic1# fabric 203 show ip prefix-list
IPv4-st19-2981889-exc-ext-out-os-svc-export2os-svc-export0os-svc-export-dst
----------------------------------------------------------------
 Node 203 (Leaf203)
----------------------------------------------------------------
ip prefix-list IPv4-st19-2981889-exc-ext-out-os-svc-export2os-svc-export0os-svc-export-dst: 2 entries

   seq 1 permit 10.3.0.1/24

fab2-apic1# fabric 204 show ip prefix-list
```

```
IPv4-proto19-2981889-exc-ext-out-os-svc-export2os-svc-export0os-svc-export-dst
------------------------------------------------------------------
 Node 204 (Leaf204)
------------------------------------------------------------------
ip prefix-list IPv4-proto19-2981889-exc-ext-out-os-svc-export2os-svc-export0os-svc-export-dst: 2
entries
```

Service Account Hardening

Every time that you create a cluster, a dedicated user account is automatically created in Cisco Application Policy Infrastructure Controller (APIC). This account is an administrative account with read and write permissions for the whole fabric.

Read and write permissions at the fabric level could be a security concern in case of multitenant fabrics where you do not want the cluster administrator to have administrative access to the Cisco Application Centric Infrastructure (ACI) fabric.

You can modify the dedicated user account limits and permissions. The level and scope of permission that is required for the cluster account depend on the location of the networking resources:

(The networking resources include the bridge domain, virtual routing and forwarding (VRF), and Layer 3 outside (L3Out).)

- When cluster resources are in the cluster dedicated tenant, the account needs read and write access to the cluster tenant and the cluster container domain.
- When cluster resources are in the common tenant, the account needs read and write access to the common tenant, the cluster tenant, and the cluster container domain.

Checking the Current Administrative Privileges

Complete the following procedure to see the current administrator privileges for the Cisco Application Centric Infrastructure (ACI) fabric.

Procedure

Step 1 Log in to Cisco Application Policy Infrastructure Controller (APIC).

Step 2 Go to **Admin** > **AAA** > **Users**.

Step 3 Click the username associated with your cluster.

Step 4 Scroll to the security domain and verify the following:

- That the security domain "all" has role admin "writePriv" and "readPriv"
- That the security domain "common" has role read-all "readPriv"

Modifying Administrative Account Permissions

After you configure the fabric, you can see a new tenant and Cisco Application Centric Infrastructure (ACI) user. Its name is equal to the *system_id* parameter specified in the Cisco ACI Container Network Interface (CNI) configuration file. Complete the following procedure to modify administrative permissions:

Note This procedure works whether cluster resources are in the same tenant or when virtual routing and forwarding (VRF) and Layer 3 Outside (L3Out) are in the common tenant. However, if VRF and L3Out are in the common tenant, you must give write permission to the common tenant In Step 3.

Procedure

Step 1 Log in to the Cisco Application Policy Infrastructure Controller (APIC).

Step 2 Create a new security domain by completing the following steps:

a) Go to **Admin** > **AAA** > **Security**.
b) Right-click **Security** and choose **Create Security Domain**.
c) In the **Create Security Domain** dialog box, enter a name for the security name and click **Submit**.

We recommend that you use the OpenShift *system_id* name. Entering a description is optional.

Step 3 Go to **Admin** > **AAA** > **Users** and complete the following steps:

a) Double-click the *system_id* username.
b) In the **Local User** dialog box, scroll to the **Security Domains** area.
c) Expand **Security Domain all**, right-click **Role Admin**, and then click **Delete** from the drop-down list.
d) Click the + (plus) icon.
e) In the **Add User Domain** dialog box, choose the container *system_id* domain.

You now add two new roles.

f) Click the + (plus) icon.
g) From the **Name** drop-down list, choose **Admin**, from the **Access Type** drop-down list, choose **Write Privilege**, and then click **Update**.

Note Complete the following three steps *only* if the cluster resources—such as the bridge domain, virtual routing and forwarding (VRF), and Layer 3 outside (L3Out)—are in the common tenant.

h) From the drop-down list, choose **common**.
i) Click the + (plus) icon.
j) From the **Name** drop-down list, choose **Admin**, from the **Access Type** drop-down list, choose **Write Privilege**, and then click **Update**
k) Click **Submit**.

Step 4 Create a custom role-based access control (RBAC) rule to allow the container account to write information into the container domain by completing the following steps:

a) Go to **Admin** > **Security** and in the **User Management - Security** central pane, choose **RBAC Rules** and **Explicit Rules**.
b) Click the tools icon and choose **`Create RBAC Rule`** from the drop-down list.
c) In the **Create RBAC Rule** dialog box, in the **DN** field, enter **`comp/prov-OpenShift/ctrlr-[`*`system_id`*`]-`*`system_id`*.

Example:

If your *system_id* is "OS," the DN is **`comp/prov-OpenShift/ctrlr-[OS]-OS`**

d) From the **Domain** drop-down list, choose the security domain that you created in Step 2.

e) Set the **Allow Writes** option to **Yes**.
f) Click **Submit**.

Step 5 Map the security domain to the cluster tenant by completing the following steps:

a) Go to **Tenants**, choose the tenant for the OpenShift *system_id*, and then in the **Tenant** central pane, choose **Policy**.
b) In the **Security Domains** area, click the **+** (plus) icon.
c) From the **Name** drop-down list, choose the newly created security name, click **Update**, and then click **Submit**.

Troubleshooting OpenShift Integration

This section contains instructions for troubleshooting the OpenShift integration with Cisco ACI.

For additional troubleshooting information, see the *Cisco ACI Troubleshooting Kubernetes and OpenShift* at:

Troubleshooting Checklist

This section contains a checklist to troubleshoot problems that occur after you integrate OpenShift with Cisco ACI.

Procedure

Step 1 Check for faults on the fabric and resolve any that are relevant.

Step 2 Check that the API server advertisement addresses use the node subnet, and that the nodes are configured to route all OpenShift subnets over the node uplink.

Typically, the API server advertisement address is pulled from the default route on the node during installation. If you are putting the default route on a different network than the node uplink interfaces, you should do so—in addition to configuring the subnets from the planning process and the cluster IP subnet used internally for OpenShift.

Step 3 Check the logs for the container `aci-containers-controller` for errors using the following command on the OpenShift master node: **`acikubectl debug logs controller acc`**

Step 4 Check these node-specific logs for errors using the following commands on the OpenShift master node:

a) Host agent: `acikubectl debug logs node -n [nodename] host-agent`
b) OpFlex agent: `acikubectl debug logs node -n [nodename] opflex-agent`
c) Open vSwitch: `acikubectl debug logs node -n [nodename] openvswitch`

Troubleshooting Specific Problems

This section describes how to troubleshoot specific problems.

Collecting and Exporting Logs

Collecting and exporting logs can help you and Cisco Support troubleshoot problems.

Procedure

Enter the following command to collect and export OpenShift logs: **`acikubectl debug cluster-report -o cluster-report.tar.gz`**

Troubleshooting External Connectivity

Follow the instructions in this section if external connectivity is not working.

Procedure

Check configuration of the next-hop router.

Note You cannot access external services from the next-hop router will not work because contracts are not enforced in this case by the fabric. Instead, access external services from an IP address that is not in the subnet configured on the next-hop router interface.

Troubleshooting POD EPG Communication

Follow the instructions in this section if communication between two pod EPGs is not working.

Procedure

Check the contracts between the pod EPGs.

Ensure that you verify that there is a contract that allows ARP traffic. All pods are in the same subnet so Address Resolution Protocol (ARP) is required.

Troubleshooting Endpoint Discovery

If an endpoint is not automatically discovered, either EPG does not exist or mapping of the annotation to EPG is not in place. Follow the instructions in this section to troubleshoot the problem.

Procedure

Step 1 Ensure that the EPG name, tenant, and application are spelled correctly.

Step 2 Make sure that the VMM domain is mapped to an EPG.

Troubleshooting Pod Health Check

Follow the instructions in this section if the pod health check does not work.

Procedure

Ensure that the health check contract exists between the pod EPG and the node EPG.

Troubleshooting aci-containers-host

Follow the instructions in this section if the mcast-daemon inside aci-containers-host fails to start.

Procedure

Check the mcast-daemon log messages:

Example:

```
oc -n kube-system logs aci-containers-host-[xxxxx] mcast-daemon
```

If the following error message is present, `Fatal error: open: Address family not supported by protocol`, ensure that IPv6 support is enabled in the kernel. IPv6 must be enabled in the kernel for the mcast-daemon to start.

Contacting Support

If you need help with troubleshooting problems, generate a cluster report file and contact Cisco TAC for support.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA 95134-1706
USA

Asia Pacific Headquarters
CiscoSystems(USA)Pte.Ltd.
Singapore

Europe Headquarters
CiscoSystemsInternationalBV
Amsterdam,TheNetherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at www.cisco.com/go/offices.