How to configure IP Fabric with OpenClos

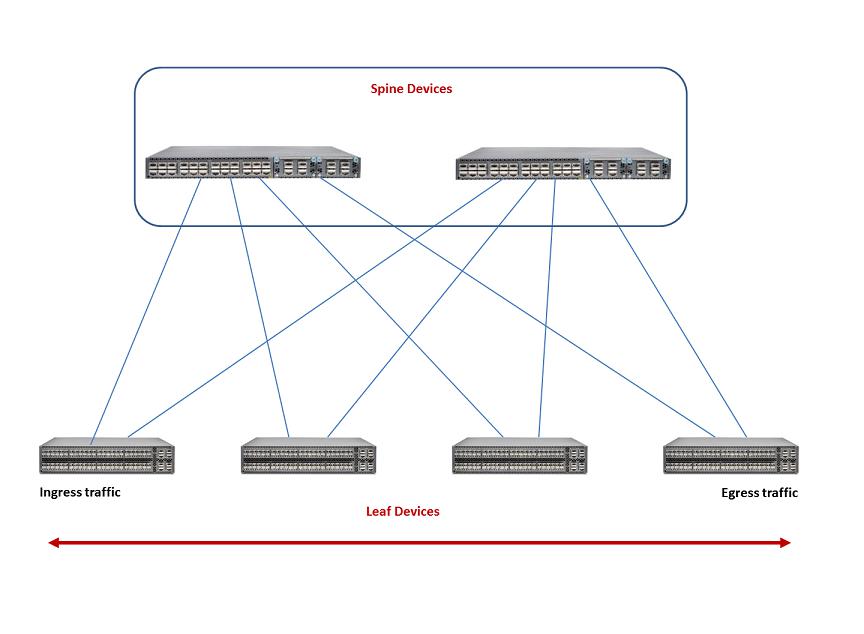
Understanding Layer 3 Fabrics



Most enterprises that host data centers are looking to increase resiliency and also support new technologies such as VMware NSX that allow them to deploy applications, servers, and virtual networks within seconds. Layer 3 Fabrics allow them to support better uptime, performance, and newer cloud infrastructures such as VMware NSX. In order to maintain the large scale required to host thousands of servers, the use of a multi-stage Clos architecture is required. Such an architecture allows the physical network to scale beyond the port density of a single switch. Layer 3 Fabrics use BGP as the control plane protocol to advertise prefixes, perform traffic engineering, and tag traffic. The most common designs in a multi-stage Clos architecture are a 3-stage and 5-stage networks that use the spine-and-leaf topology.

Spine-and-leaf topology is an alternate to the traditional three-layer network architecture, which consists of an access layer, aggregation layer, and a core. In the spine-and-leaf topology, all the leaf devices are connected to the spine devices in a mesh as shown in [Figure 1](#page600).

Figure 1: Layer 3 Fabric in a Spine and Leaf Topology



Typically, the spine devices are high-performance switches capable of Layer 3 switching and routing combined with high port density. Spine devices constitute the core and the leaf devices constitute the access layer in Layer 3 Fabrics. Leaf devices enable servers to connect to the Layer 3 Fabric. They also provide uplinks to spine devices.

OpenClos currently supports only the 3-stage design. The 3-stage design has two roles—the spine and the leaf. It is called a 3-stage design because the traffic must traverse three switches in the worst-case scenario.

The maximum number of spine devices that you can have in your Layer 3 Fabric depends on the number of 40-Gigabit Ethernet interfaces in your leaf devices. A Layer 3 Fabric that has 8 QFX5100-24Q spine devices and 32 QFX5100-96S leaf devices (each leaf supports 96 10-Gigabit Ethernet ports) can provide 3072 usable 10-Gigabit Ethernet ports.

Creating Layer 3 Fabrics

You create IP Fabric by performing following tasks in the order listed here:

Start OpenClos

1. Start REST Server

OpenClos REST server supports both HTTP and HTTPS.

**HTTP**:

Configure HTTP with following in <OPENCLOS\_INSTALLATION\_PATH>/jnpr/openclos/conf/openclos.yaml,

restServer :

version : 1

protocol : http

ipAddr : <REST server ip>

port : <REST server port>

Use HTTP when your client is located in the same machine as the OpenClos REST server. In this case, no authentication is performed. The only configurations required in this case are 'ipAddr' and 'port'.

**HTTPS**:

Configure HTTPS with following in <OPENCLOS\_INSTALLATION\_PATH>/jnpr/openclos/conf/openclos.yaml,

restServer :

version : 1

protocol : https

ipAddr : <REST server ip>

port : <REST server port>

username : <username>

password : <2-way encrypted password>

certificate : <full path of the server certificate>

Use HTTPS when your client is accessing OpenClos REST server remotely. In this case, Basic Authentication is done over HTTPS.

***Default username/password/server certificate***:

OpenClos comes with a built-in username/password 'juniper/juniper' in openclos.yaml. OpenClos automatically generates a default server certificate whose CN subject is set to 'ipAddr' value in openclos.yaml when you start the REST server for the first time. The default server certificate is stored in ~/openclos.pem. The default username/password/server certificate enables user to use OpenClos HTTPS REST server out of box. The only configruations required in this case are 'ipAddr' and 'port'.

***Non default username/password/server certificate***:

If user decides to use non default username/password/server certificate, follow below instructions:

* Change 'username' to the new username
* Use "python crypt.py " to generate a 2-way encrypted password:
* root@sw-ubuntu25:# cd <OPENCLOS\_INSTALLATION\_PATH>/jnpr/openclos

root@sw-ubuntu25:.../jnpr/openclos# python crypt.py foobar

$9$lusvWxZGi5QnVwYoZG.m

Then copy-paste the output ($9$lusvWxZGi5QnVwYoZG.m) to 'password' in openclos.yaml

* Change 'certificate' to the full path of your certificate. Make sure your cerfiicate's CN subject is set to 'ipAddr' value in openclos.yaml
* Note the 'username', 'password' and 'cerficate' are required in HTTPS mode. OpenClos REST server won't start without them.
* Note the 'ipAddr' value needs to be set properly. OpenClos REST server won't start if 'ipAddr' value is 0.0.0.0
* Note Openclos also generates a public key file ~/openclos.pem.cer for the default certificate. User might want to import this file to their client if their client does not automatically accept the server cert.

Now Start REST Server

* root@sw-ubuntu25:# cd <OPENCLOS\_INSTALLATION\_PATH>/jnpr/openclos

root@sw-ubuntu25:.../jnpr/openclos# python rest.py

1. Start Trap Server

* root@sw-ubuntu25:# cd <OPENCLOS\_INSTALLATION\_PATH>/jnpr/openclos

root@sw-ubuntu25:.../jnpr/openclos# python trapd.py

Create POD

Send HTTP POST request with following request body.

POST http://<server>:<port>/openclos/v1/underlay/pods

{

"pod":{

"name":"aks",

"spineCount":2,

"spineAS":5,

"spineSettings":[{"deviceType":"qfx5200-32c-32q","downlinkPorts":["et-0/0/[0-1]"]}],

"leafCount":2,

"leafAS":10,

"topologyType":"threeStage",

"loopbackPrefix":"12.1.1.1/21",

"vlanPrefix":"15.1.1.1/21",

"interConnectPrefix":"14.1.1.1/21",

"outOfBandAddressList":"10.204.244.27",

"managementPrefix":"192.168.2.1/24",

"description":"",

"hostOrVmCountPerLeaf":2,

"devicePassword":"viren123",

"outOfBandGateway":"192.168.2.1",

"devices":[

{"role":"spine","family":"qfx5200-32c-32q","name":"aks-spine-0","username":"root",

"serialNumber":"VG3714070310","password":"viren123","deployStatus":"deploy"},

{"role":"spine","family":"qfx5200-32c-32q","name":"aks-spine-1","username":"root",

"serialNumber":"VG3714070311","password":"viren123","deployStatus":"deploy"},

{"role":"leaf","family":"qfx5100-48s-6q","name":"aks-leaf-0","username":"root",

"password":"viren123","deployStatus":"deploy"},

{"role":"leaf","family":"qfx5100-48s-6q","name":"aks-leaf-1","username":"root",

"password":"viren123","deployStatus":"deploy"}

],

"leafSettings":[

{

"deviceType":"qfx5100-96s-8q"

},

{

"deviceType":"ex4300-32f"

},

{

"deviceType":"qfx5100-48s-6q"

},

{

"deviceType":"qfx5100-48t-6q"

},

{

"deviceType":"qfx5200-32c-32q",

"downlinkPorts":["et-0/0/[2-35]"]

},

{

"deviceType":"ex4300-48p"

},

{

"deviceType":"ex4300-24p"

},

{

"deviceType":"ex4300-48t"

},

{

"deviceType":"ex4300-24t"

}

]

}

}

HTTP Response from OpenClos

200 OK

{

"pod": {

"l2Report": {

"uri": "http://192.168.56.101:20080/openclos/v1/underlay/pods/d477551a-fa94-485e-89fd-97ca1235e92c/l2-report"

},

"spineAS": 5,

"cablingPlan": {

"uri": "http://192.168.56.101:20080/openclos/v1/underlay/pods/d477551a-fa94-485e-89fd-97ca1235e92c/cabling-plan"

},

"vlanPrefix": "15.1.1.1/21",

"outOfBandAddressList": "10.204.244.27",

"id": "d477551a-fa94-485e-89fd-97ca1235e92c",

"topologyType": "threeStage",

"leafCount": 2,

"managementPrefix": "192.168.2.1/24",

"outOfBandGateway": "192.168.2.1",

"deviceConfiguration": {

"uri": "http://192.168.56.101:20080/openclos/v1/underlay/pods/d477551a-fa94-485e-89fd-97ca1235e92c/device-configuration"

},

"loopbackPrefix": "12.1.1.1/21",

"devicePassword": "viren123",

"description": "",

"interConnectPrefix": "14.1.1.1/21",

"l3Report": {

"uri": "http://192.168.56.101:20080/openclos/v1/underlay/pods/d477551a-fa94-485e-89fd-97ca1235e92c/l3-report"

},

"leafSettings": [

{

"downlinkPorts": [],

"uplinkPorts": [],

"deviceType": "ex4300-24p",

"junosImage": null

},

{

"downlinkPorts": [],

"uplinkPorts": [],

"deviceType": "ex4300-24t",

"junosImage": null

},

{

"downlinkPorts": [],

"uplinkPorts": [],

"deviceType": "ex4300-32f",

"junosImage": null

},

{

"downlinkPorts": [],

"uplinkPorts": [],

"deviceType": "ex4300-48p",

"junosImage": null

},

{

"downlinkPorts": [],

"uplinkPorts": [],

"deviceType": "ex4300-48t",

"junosImage": null

},

{

"downlinkPorts": [],

"uplinkPorts": [],

"deviceType": "qfx5100-48s-6q",

"junosImage": null

},

{

"downlinkPorts": [],

"uplinkPorts": [],

"deviceType": "qfx5100-48t-6q",

"junosImage": null

},

{

"downlinkPorts": [],

"uplinkPorts": [],

"deviceType": "qfx5100-96s-8q",

"junosImage": null

},

{

"downlinkPorts": [

"et-0/0/[2-35]"

],

"uplinkPorts": [],

"deviceType": "qfx5200-32c-32q",

"junosImage": null

}

],

"ztpConfiguration": {

"uri": "http://192.168.56.101:20080/openclos/v1/underlay/pods/d477551a-fa94-485e-89fd-97ca1235e92c/ztp-configuration"

},

"hostOrVmCountPerLeaf": 2,

"spineCount": 2,

"name": "aks",

"uri": "http://192.168.56.101:20080/openclos/v1/underlay/pods/d477551a-fa94-485e-89fd-97ca1235e92c",

"devices": {

"total": 4,

"uri": "http://192.168.56.101:20080/openclos/v1/underlay/pods/d477551a-fa94-485e-89fd-97ca1235e92c/devices"

},

"spineSettings": [

{

"downlinkPorts": [

"et-0/0/[0-1]"

],

"uplinkPorts": [],

"deviceType": "qfx5200-32c-32q",

"junosImage": null

}

],

"leafAS": 10,

"leafUplinkcountMustBeUp": 2

}

}

Configure ZTP (Zero Touch Provision)

* Create ZTP Configuration

Send HTTP PUT request with empty request body.

PUT http://<server>:<port>/openclos/v1/underlay/pods/<pod-id>/ztp-configuration

HTTP Response from OpenClos

200 OK

* Create Device Configuration

Send HTTP PUT request with empty request body.

PUT http://<server>:<port>/openclos/v1/underlay/pods/<pod-id>/device-configuration

HTTP Response from OpenClos

200 OK

* Configure DHCP and TFTP
* DHCP configuration

Send HTTP GET request with empty request body.

GET http://<server>:<port>/openclos/v1/underlay/pods/<pod-id>/ztp-configuration

HTTP Response from OpenClos

200 OK

#

# ZTP DHCP jinja template for ISC-DHCP server

# Includes basic configuration for ISC dhcpd for Debian

#

# The ddns-updates-style parameter controls whether or not the server will

# attempt to do a DNS update when a lease is confirmed. We default to the

# behavior of the version 2 packages ('none', since DHCP v2 didn't

# have support for DDNS.)

ddns-update-style none;

# option definitions common to all supported networks...

#option domain-name "example.org";

#option domain-name-servers ns1.example.org, ns2.example.org;

default-lease-time 600;

max-lease-time 7200;

# If this DHCP server is the official DHCP server for the local

# network, the authoritative directive should be uncommented.

#authoritative;

# Use this to send dhcp log messages to a different log file (you also

# have to hack syslog.conf to complete the redirection).

log-facility local7;

option space ztp-ops;

option ztp-ops.image-file-name code 0 = text;

option ztp-ops.config-file-name code 1 = text;

option ztp-ops.image-file-type code 2 = text;

option ztp-ops.transfer-mode code 3 = text;

option ztp-ops-encap code 43 = encapsulate ztp-ops;

option ztp-file-server code 150 = { ip-address };

class "VG3714070310-client" {

match if substring (option host-name, 0,12) = "VG3714070310";

}

class "VG3714070311-client" {

match if substring (option host-name, 0,12) = "VG3714070311";

}

class "ex4300-24p-clients" {

match if substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-24p";

}

class "ex4300-24t-clients" {

match if substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-24t";

}

class "ex4300-32f-clients" {

match if substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-32f";

}

class "ex4300-48p-clients" {

match if substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-48p";

}

class "ex4300-48t-clients" {

match if substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-48t";

}

class "qfx5100-48s-6q-clients" {

match if substring (option vendor-class-identifier, 0,22) = "Juniper-qfx5100-48s-6q";

}

class "qfx5100-48t-6q-clients" {

match if substring (option vendor-class-identifier, 0,22) = "Juniper-qfx5100-48t-6q";

}

class "qfx5100-96s-8q-clients" {

match if substring (option vendor-class-identifier, 0,22) = "Juniper-qfx5100-96s-8q";

}

class "qfx5200-32c-32q-clients" {

match if substring (option vendor-class-identifier, 0,23) = "Juniper-qfx5200-32c-32q";

}

subnet 192.168.48.128 netmask 255.255.255.128 {

#range 192.168.48.218 192.168.48.223;

option routers 192.168.48.129;

option broadcast-address 192.168.48.255;

option ztp-file-server 0.0.0.0;

option host-name "netboot";

option ztp-ops.image-file-name "openclos/v1/underlay/images/jinstall-qfx-5-14.1X53-D10.4-domestic-signed.tgz";

option ztp-ops.transfer-mode "http";

pool {

allow members of "ex4300-24p-clients";

allow members of "ex4300-24t-clients";

allow members of "ex4300-32f-clients";

allow members of "ex4300-48p-clients";

allow members of "ex4300-48t-clients";

allow members of "qfx5100-48s-6q-clients";

allow members of "qfx5100-48t-6q-clients";

allow members of "qfx5100-96s-8q-clients";

allow members of "qfx5200-32c-32q-clients";

range 192.168.48.218 192.168.48.223;

if substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-24p" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/ex4300-24p";

}

elsif substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-24t" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/ex4300-24t";

}

elsif substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-32f" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/ex4300-32f";

}

elsif substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-48p" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/ex4300-48p";

}

elsif substring (option vendor-class-identifier, 0,18) = "Juniper-ex4300-48t" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/ex4300-48t";

}

elsif substring (option vendor-class-identifier, 0,22) = "Juniper-qfx5100-48s-6q" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/qfx5100-48s-6q";

}

elsif substring (option vendor-class-identifier, 0,22) = "Juniper-qfx5100-48t-6q" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/qfx5100-48t-6q";

}

elsif substring (option vendor-class-identifier, 0,22) = "Juniper-qfx5100-96s-8q" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/qfx5100-96s-8q";

}

elsif substring (option vendor-class-identifier, 0,23) = "Juniper-qfx5200-32c-32q" {

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/leaf-generic-configurations/qfx5200-32c-32q";

}

}

pool {

allow members of "VG3714070310-client";

range dynamic-bootp 192.168.2.1;

option host-name "aks-spine-0";

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/devices/89467dd0-062b-464a-ab8d-13ad4fb2ed20/config";

}

pool {

allow members of "VG3714070311-client";

range dynamic-bootp 192.168.2.2;

option host-name "aks-spine-1";

option ztp-ops.config-file-name "openclos/v1/underlay/pods/7d35b31f-6aa5-4f75-b2dd-5779f4379a91/devices/418f9729-0c47-4ecd-a359-bbf5708ed1c2/config";

}

} # subnetEnd

Save the entire response as dhcpd.conf and copy it to /etc/dhcp/dhcpd.conf

Restart dhcpd: service dhcpd restart

* Device configuration

Send HTTP GET request with empty request body.

GET http://<server>:<port>/openclos/v1/underlay/pods/<pod-id>/device-configuration

HTTP Response from OpenClos

200 OK

<Response omitted>

Save the entire response as config.zip. Unzip it and copy all configs to /var/lib/tftpboot

Restart tftp: service xinetd restart

Cabling

* Create Cabling Plan

Send HTTP PUT request with empty request body.

PUT http://<server>:<port>/openclos/v1/underlay/pods/<pod-id>/cabling-plan

HTTP Response from OpenClos

200 OK

* Connect physical cables based on Cabling Plan

Send HTTP GET request with empty request body.

GET http://<server>:<port>/openclos/v1/underlay/pods/<pod-id>/cabling-plan

HTTP Response from OpenClos

200 OK

{

"cablingPlan": {

"devices": [

{"id": "c2515533-641e-4be3-9ace-d6b46bb804b9", "name": "aks-leaf-0", "family": "qfx5100-48s-6q", "role":"leaf"},

{"id": "fe4ca39f-787b-4973-9f1c-c10dfe89b990", "name": "aks-leaf-1", "family": "qfx5100-48s-6q", "role":"leaf"},

{"id": "89467dd0-062b-464a-ab8d-13ad4fb2ed20", "name": "aks-spine-0", "family": "qfx5200-32c-32q", "role":"spine"},

{"id": "418f9729-0c47-4ecd-a359-bbf5708ed1c2", "name": "aks-spine-1", "family": "qfx5200-32c-32q", "role":"spine"}

],

"links": [

{ "linkType": "interconnect", "device1": "aks-leaf-0", "port1": "et-0/0/48", "ip1":"14.1.0.1/31", "device2": "aks-spine-0", "port2": "et-0/0/0", "ip2":"14.1.0.0/31"},

{ "linkType": "interconnect", "device1": "aks-leaf-0", "port1": "et-0/0/49", "ip1":"14.1.0.5/31", "device2": "aks-spine-1", "port2": "et-0/0/0", "ip2":"14.1.0.4/31"},

{ "linkType": "interconnect", "device1": "aks-leaf-1", "port1": "et-0/0/48", "ip1":"14.1.0.3/31", "device2": "aks-spine-0", "port2": "et-0/0/1", "ip2":"14.1.0.2/31"},

{ "linkType": "interconnect", "device1": "aks-leaf-1", "port1": "et-0/0/49", "ip1":"14.1.0.7/31", "device2": "aks-spine-1", "port2": "et-0/0/1", "ip2":"14.1.0.6/31"}

]

}

}

CAUTION: Ensure that you always perform the physical connections based on the cabling plan that OpenClos generates for your fabric. Not following this set order might render your Layer 3 Fabric defunct.

Bring up the IP Fabric

Power on ALL spines first. Power on leaves **after** ALL spines are up and running.

When spines start up, they go through ZTP process by exchanging DHCP messages with dhcpd (Note dhcpd needs to be running in the same subnet). Spines configure themselves by downloading the configurations (specifically generated for each spine by OpenClos in previous step) from tftp server.

When leaves start up, they go through ZTP process by exchanging DHCP messages with dhcpd and configure themselves by downloading a generic configuration from tftp server. Once leaves are up and running, they send out SNMP trap to OpenClos trap server. Upon receiving the trap, OpenClos will generate a tailored configuration and push to each leaf. This tailored configuration is specific to each leaf.

Verify Layer 2 and Layer 3 connectivity

* Layer 2 connections

Send HTTP GET request with empty request body.

GET http://<server>:<port>/openclos/v1/underlay/pods/<pod-id>/l2-report

HTTP Response from OpenClos

200 OK

{

"l2Report": {

"devices": [

{

"id": "c2515533-641e-4be3-9ace-d6b46bb804b9",

"name": "aks-leaf-0",

"family": "qfx5100-48s-6q",

"role": "leaf",

"status": "unknown",

"statusReason": "None",

"deployStatus": "deploy"

},

{

"id": "fe4ca39f-787b-4973-9f1c-c10dfe89b990",

"name": "aks-leaf-1",

"family": "qfx5100-48s-6q",

"role": "leaf",

"status": "unknown",

"statusReason": "None",

"deployStatus": "deploy"

},

{

"id": "89467dd0-062b-464a-ab8d-13ad4fb2ed20",

"name": "aks-spine-0",

"family": "qfx5200-32c-32q",

"role": "spine",

"status": "unknown",

"statusReason": "None",

"deployStatus": "deploy"

},

{

"id": "418f9729-0c47-4ecd-a359-bbf5708ed1c2",

"name": "aks-spine-1",

"family": "qfx5200-32c-32q",

"role": "spine",

"status": "unknown",

"statusReason": "None",

"deployStatus": "deploy"

}

],

"links": [

{

"linkType": "interconnect",

"device1": "aks-leaf-0",

"port1": "et-0/0/48",

"ip1": "14.1.0.1/31",

"device2": "aks-spine-0",

"port2": "et-0/0/0",

"ip2": "14.1.0.0/31",

"status": "unknown"

},

{

"linkType": "interconnect",

"device1": "aks-leaf-0",

"port1": "et-0/0/49",

"ip1": "14.1.0.5/31",

"device2": "aks-spine-1",

"port2": "et-0/0/0",

"ip2": "14.1.0.4/31",

"status": "unknown"

},

{

"linkType": "interconnect",

"device1": "aks-leaf-1",

"port1": "et-0/0/48",

"ip1": "14.1.0.3/31",

"device2": "aks-spine-0",

"port2": "et-0/0/1",

"ip2": "14.1.0.2/31",

"status": "unknown"

},

{

"linkType": "interconnect",

"device1": "aks-leaf-1",

"port1": "et-0/0/49",

"ip1": "14.1.0.7/31",

"device2": "aks-spine-1",

"port2": "et-0/0/1",

"ip2": "14.1.0.6/31",

"status": "unknown"

}

]

}

}

* Layer 3 connections

Send HTTP GET request with empty request body.

GET http://<server>:<port>/openclos/v1/underlay/pods/<pod-id>/l3-report

HTTP Response from OpenClos

200 OK

{

"l3Report": {

"devices": [

{

"id": "c2515533-641e-4be3-9ace-d6b46bb804b9",

"name": "aks-leaf-0",

"family": "qfx5100-48s-6q",

"role": "leaf",

"status": "unknown",

"statusReason": "None",

"deployStatus": "deploy"

},

{

"id": "fe4ca39f-787b-4973-9f1c-c10dfe89b990",

"name": "aks-leaf-1",

"family": "qfx5100-48s-6q",

"role": "leaf",

"status": "unknown",

"statusReason": "None",

"deployStatus": "deploy"

},

{

"id": "89467dd0-062b-464a-ab8d-13ad4fb2ed20",

"name": "aks-spine-0",

"family": "qfx5200-32c-32q",

"role": "spine",

"status": "unknown",

"statusReason": "None",

"deployStatus": "deploy"

},

{

"id": "418f9729-0c47-4ecd-a359-bbf5708ed1c2",

"name": "aks-spine-1",

"family": "qfx5200-32c-32q",

"role": "spine",

"status": "unknown",

"statusReason": "None",

"deployStatus": "deploy"

}

],

"peers": []

}

}