JUNIPER TECHFEST 2019
ENTERPRISE MULTICLOUD – DC FABRIC
HANDS-ON LAB – SOLUTION GUIDE

Simple, Open, and Smart.

Juniper Networks | Engineering Simplicity

INTRODUCTION

The goal of the present lab session is get hands-on experience on the EM (Enterprise Multicloud) 5.1 fabric manager for the baseline networking capabilities implementation such as deploying the Edge-Routed-Bridging brownfield fabric overlay, implementing Virtual-Port-Groups for BMS server connectivity, Logical-Routers, L2 VXLAN gateway signalized by EVPN,

Lab topology

https://github.com/juniper-dc/techfest2019

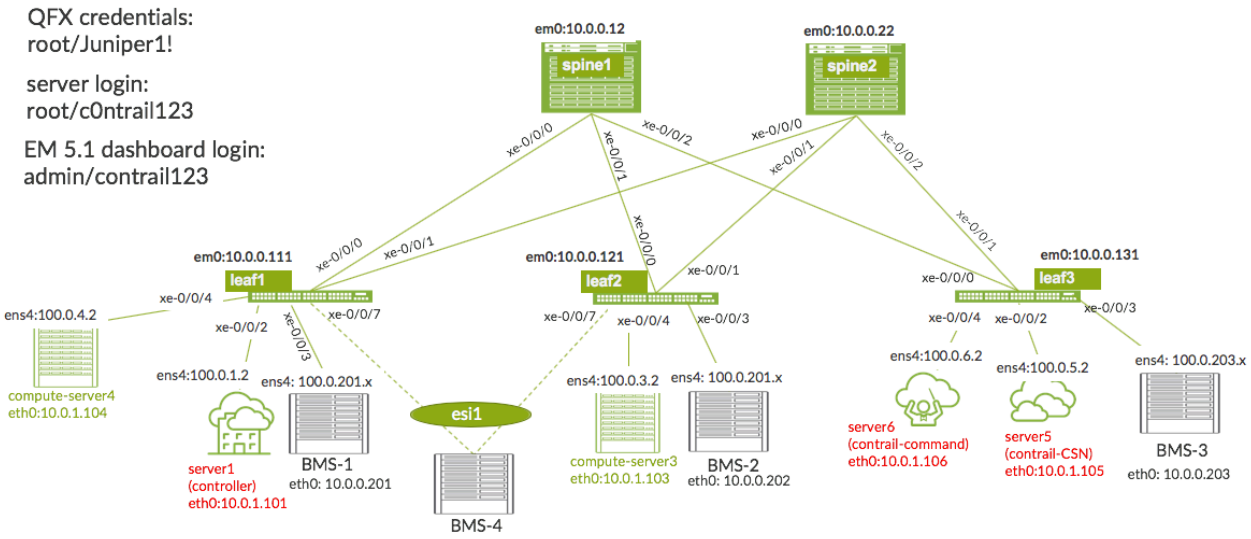


Figure 1 EM 5.1. for DC fabrics - main lab topology

The following diagram gives an overview of the flows between the EM main dashboard (server6) and the config API (server1)

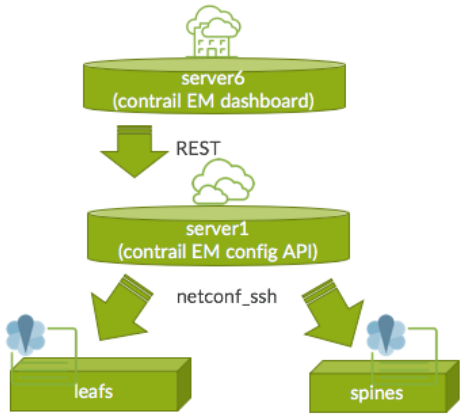


Figure 2 EM 5.1 - high level flow diagram

Lab access

The lab topology is delivered within the Ravello Oracle cloud system. Every user gets his own virtual lab setup as per the main topology diagram.

The Oracle Ravello cloud system URL will be allowing to get the nodes IP addresses and ravello login/password are on the printed lab material student gets during the in-person hands-on lab session.

You can ssh using putty or secure-crt with the dns names or IP@ listed in ravello interface using your internet connectivity

- the lab vqfx device access info: `root` and `Juniper1!`
- the contrail EM dashboard login/password is: `admin` and `contrail123`
- the servers ssh access login/password is: `root` and `c0ntrail123`

Lab objectives

The goal of the present lab is to build the overlay **Edge Routed Bridging** fabric architecture using the Juniper QFX series EVPN-VXLAN technologies in order to deliver L2 active/active forwarding within the datacenter between the BMS servers and VMs.

The ZTP process was already conducted for the present fabric so the first step is to onboard the devices from the topology and create an overlay iBGP fabric.

The iBGP EVPN sign. overlay will be used in order to advertise the MAC@ and MAC+IP in DC-1 and EVPN-type5 between the DC1 and DC2.

The inter-vni tenant routing will be taking place at the leaf1/leaf2/leaf3 level only i.e. spines should not be provisioned with any tenant information but should play the role of Route Reflectors and IP Forwarders between the leafs

Spine3-re is deployed in DC-2 is connected to the same overlay ASN 64512 and Spine1-re/Spine2-re but DC-1 to DC-2 exchanges only EVPN type-5 route for prefix-advertisement.

The ultimate goal of the present lab is to deliver:

- L2 communication between BMS1 and BMS2 - they should reside in the same virtual-network with VLAN-VNIs allocated dynamically by contrail
- L3 inter-vni communication should be symmetric and take place only at the leafs
- L2 VXLAN gateway for BMS1 to VM2 (on compute-server-3) should allow to communicate between the BMS1 and the VM2 using the same subnet

Lab environment

The environment is composed of the following vqfx nodes:

- 2 x vQFX Spines - spine1, spine2
- 3 x vQFX Leafs: leaf1, leaf2, leaf3
- 3 x BMS servers - bm1, bms2, bms3, bms4
- 2 x compute-servers - server3 and server4
- 1 x CSN server - server5
- 1 x controller - server1
- 1 x EM dashboard - server6

The underlay eBGP is already pre-provisioned in order to deliver full IP reachability between the loopback0.0 IP@ of the given IP fabric

All nodes from the lab are accessible via internet, so you can verify each individual node state from your laptop using SSH sessions
Use the username: `root` and password: `Juniper1!`

Lab tasks

Task 1.0 Verify the full IPv4 underlay reachability within the main topology

- Ensure the full IP reachability of your fabric
- Verify the eBGP underlay peering are all in the established state
- verify the contrail container and services are working fine in your lab

Expected result:

- two eBGP peerings per leaf device in the Established state,
- Full IP reachability between the lo0.0 IP addresses of each fabric node
- contrail components containers are running at server1 and server6

Note: for contrail UI based verifications you can login to the EM dashboard via

https://<server6_public_IP_address>:9091

or to the legacy contrail dashboard

https://<server1_public_IP_address>:8143

login/password is: `admin/contrail123`

- the servers ssh access login/password are: `root` and `c0ntrail123`

When ssh'ed into the contrail server1/server6 or the compute-nodes/CSN you can verify the baseline status by running the `contrail-status` command or by checking the state of the contrail containers `docker ps`
You can check the type of containers running on server1 vs server6 and the containers running on the compute-nodes server3/server4/server5(CSN)

Task 1.1 Deploy the overlay iBGP EVPN network using the BGP ASN 64512

- Discover the devices as based on the given range of subnet 10.0.0.0/24
- Associate the devices with the roles
- Deploy the overlay

Expected result:

- The leaf and spine devices are discovered and each device is associated with the Edge-Routed architecture roles
- Spines1/2 should have role: null (aka Lean Spine) and Route-Reflector
- Leafs1/2/3 should have the role: ERB-UCAST-Gateway

Task 1.2 Verify the iBGP peering is established between all the leafs and spines route reflectors

- Using the "show bgp summary” command make sure all leafs have a total of four BGP peerings - two for underlay and two for overlay

Expected results:

- four BGP peerings are in the Established state - two for the underlay and two for the overlay

---

Task 1.3 Enable the new network policy called my_allow_all_policy which will allow for the communication within the BMS1 and BMS2 virtual-network

- Go to the Overlay-> Network Policy EM 5.1 dashboard and enable the new network policy that allows the communication from/to network 100.0.201/24

Expected result: The new Network policy is enabled

Task 1.4 Enable the Virtual-Network1-2 used for BMS-1 and BMS-2 communication

- Go to the Overlay EM 5.1 dashboard and enable new Virtual-Network used for BMS1 and BMS2 in the subnet range 100.0.201.0/24

Expected result: The new Virtual-Network is created but not yet pushed to the leaf devices

Ask yourself: why the virtual-network is not pushed to the leaf devices in this case ?

Task 1.5 Create two VPG-1 and VPG-2 ports: one for BMS1 at leaf1 and one for BMS2 at leaf2 - use the interfaces from the topology diagram

- Go to the Overlay -> Virtual-Port-Group option of the EM dashboard and create two VPG ports associated with the virtual-network created in the previous task

Expected result: The virtual-network VLAN and IRB configuration is pushed to the leaf1 and leaf2. BMS1 can ping BMS2 and EVPN based MAC@ learning is taking place

Ask yourself:

- did you have to enable any VNI or VLAN ?
- are the new virtual-network configurations enabled on all leafs or just selected leafs ?

Task 1.6 Create a new Virtual-network for BMS3

- Go to the Overlay EM 5.1 dashboard and enable new Virtual-Network used for BMS3 in the range 100.0.203.0/24

Expected result: The new Virtual-Network is not yet pushed to the leaf devices but is successfuly created in the contrail config API

Task 1.7 Create third VPG-3 ports for BMS3 at leaf3

- Go to the Overlay -> Virtual-Port-Group option of the EM dashboard and create two VPG ports associated with the virtual-network created for BMS3 in the previous task

Expected result: The virtual-network VLAN and IRB configuration is pushed to the leaf3 only BMS3 cannot ping the BMS1 and BMS2

Task 1.8 Create a VPG-4 which will be ready to connect in the future the BMS-4 connected to leaf1 xe-0/0/7 and leaf2 xe-0/0/7

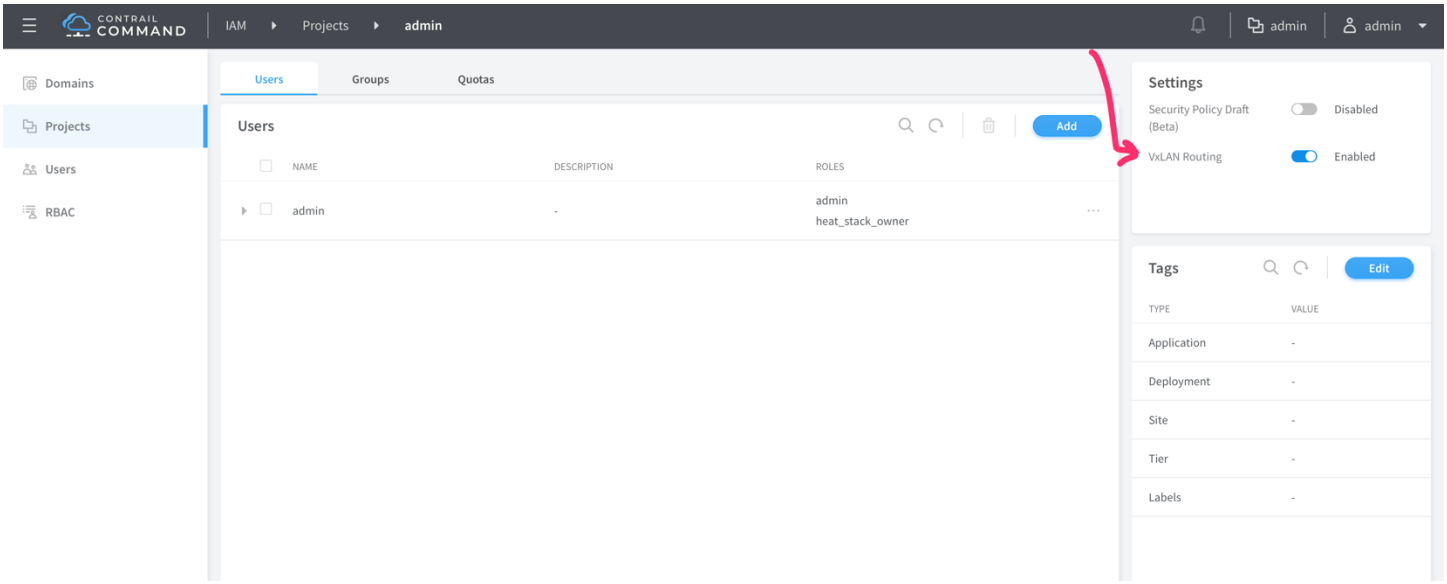
- Go to the Overlay -> Virtual-Port-Group option of the EM dashboard and create two VPG ports associated with the same virtual-network as used by BMS-1 and BMS-2

Expected result: interfae xe-0/0/7 on leaf1 and leaf2 should be part of a new ESI-LAG, ESI and LACP system-id values of the new ESI-LAG should be allocated automatically by the Fabric Manager

---

Task 1.9 Create a new Logical-Router in order to communicate between the BMS1/2 and BMS3 - integrate the BMS1 and BMS3 virtual-networks under the same logical-router

- Go to the Overlay -> Logical Router dashboard option and create a new Logical Router name that associates the two Virtual-Networks enabled in previous tasks
- Specify at which leaf devices the logical router (T5 instance) should be enabled Note: before enabling the logical-routers make sure in the EM 5.1 the vxlan-routing capability is explicitly enabled in the given fabric project



Note: the BMS-1/BMS-2 will have to get the static route for BMS-3 subnet and BMS-3 will have to get the static route for BMS-1/2 subnet You can add the routes by going to the BMS-1/2 and BMS-3 console and adding it: `route add -net 100.0.203.0/24 gw 100.0.201.1` on BMS-1 and BMS-2 `route add -net 100.0.201.0/24 gw 100.0.203.1` on BMS-3
Expected result:

- BMS3 is able to ping BMS1/2
- T5 instance is pushed dynamically to BMS1/2/3 when new LR is created in contrail EM

Task 1.10 Ensure the CSN is onboarded using server5 and services it offers are fully functional

- ensure the DHCP daemon is running at the CSN server
- check what other service are running on that server *Ask yourself:*
- what's the purpose of that server ?
- if we were deploying only the BMSes in the fabric do you still need CSN services ?

Task 1.11 Check the servers server3 and server4 from the topology are onboarded as compute-nodes (aka vrouters)

- server4 attached to leaf1 and server3 attached to leaf2 are successfully onboarded already so in this task we have to verify the IP full reachability from hypervisor point of view via the fabric underlay is delivered

Expected result:

- you can ping from these servers their default gateway
- ping between server3 and server4
- ping from server3/server4 to the CSN server works as well using the in-band network

Task 1.12 Using a Cirros linux image enable two VMs in the same virtual-network as BMS-1 and BM-2

Note: before enabing the VMs ensure the local compute-node /etc/hosts file is updated with the IP in-band and hostname Here's the example of adding the in-band IP@ at the server3 shell `100.0.3.2 server3 server3` is added manually You'll have to make the same changes for server4 and CSN server5 Here's the example of the change to be done at the server3 - first check the local IP@ of the in-band fabric interface and add it to /etc/hosts file

```
[root@server3 ~]# vi /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
10.0.1.101 server1.local server1
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

10.0.1.104 server4.local server4
10.0.1.105 server5.local server5
10.0.1.102 server2.local server2
10.0.1.103 server3.local server3
100.0.3.2 server3 server3
```

```
[root@server3 ~]# docker-compose down
[root@server3 ~]# docker-compose up -d
```

Repeat the same for the server4 by adding on server4 the local IP@ and server4 name

- VM-1 and VM-2 should be enabled in the already existing virtual-network dedicated to BMS-1 and BMS-2
- upload the cirros image. Using the contrail EM dashboard go to workloads > images and upload the Cirros tiny linux you can first download to your laptop

https://docs.openstack.org/image-guide/obtain-images.html

Note: In a CirrOS based VM default login is `cirros` and the password is `gocubsgo`
Note:

- the VM created may need a static arp entry for the BMS due to the vqfx limitation where tunneled arp messages sometimes use vni 0
- make sure the BMSes from the lab are not using as the last octet the IP@ `.1` or `.2` as these are reserved IP@ - change it to an unused IP@ from the given subnet at the BMS when needed

```
arp -s 100.0.201.12 2c:c2:60:63:51:e4 arp -an
[root@server4 ~]# route -n | grep 169
169.254.0.0     0.0.0.0         255.255.0.0     U     1002   0        0 ens3
169.254.0.3     0.0.0.0         255.255.255.255 UH    0      0        0 vhost0
169.254.0.4     0.0.0.0         255.255.255.255 UH    0      0        0 vhost0
[root@server4 ~]#
[root@server4 ~]# ssh cirros@169.254.0.3
cirros@169.254.0.3's password:
$
$
```

Expected result:

- BMS1 can ping the VM-2
- BMS-2 can ping VM-1

Task 1.13 Create a VM-3 an associate it with the existing virtual-network, created already for the BMS-3

Expected result:

- VM-3 can ping the BMS-3
- VM-3 can ping BMS-1

Ask yourself:

- which part of the routing is at the vrouter and which at the qfx leaf device ?

Implementation details:

- The fabric discovery network: `10.0.0.0/24`
- Overlay iBGP ASN number: `64512`
- BMS1 and BMS2 virtual network IP@ range: `100.0.201.30-100.0.201.45` subnet mask `/24`
- BMS3 virtual network IP@ range: `100.0.203.30-100.0.203.45` subnet mask `/24`
- BMS4 virtual network IP@ range: `100.0.204.30-100.0.204.45` subnet mask `/24`

- use the fabric, virtual-network, VPG and LR names that are significant for you as the fabric administrator

Solution guide for EVPN/VXLAN lab

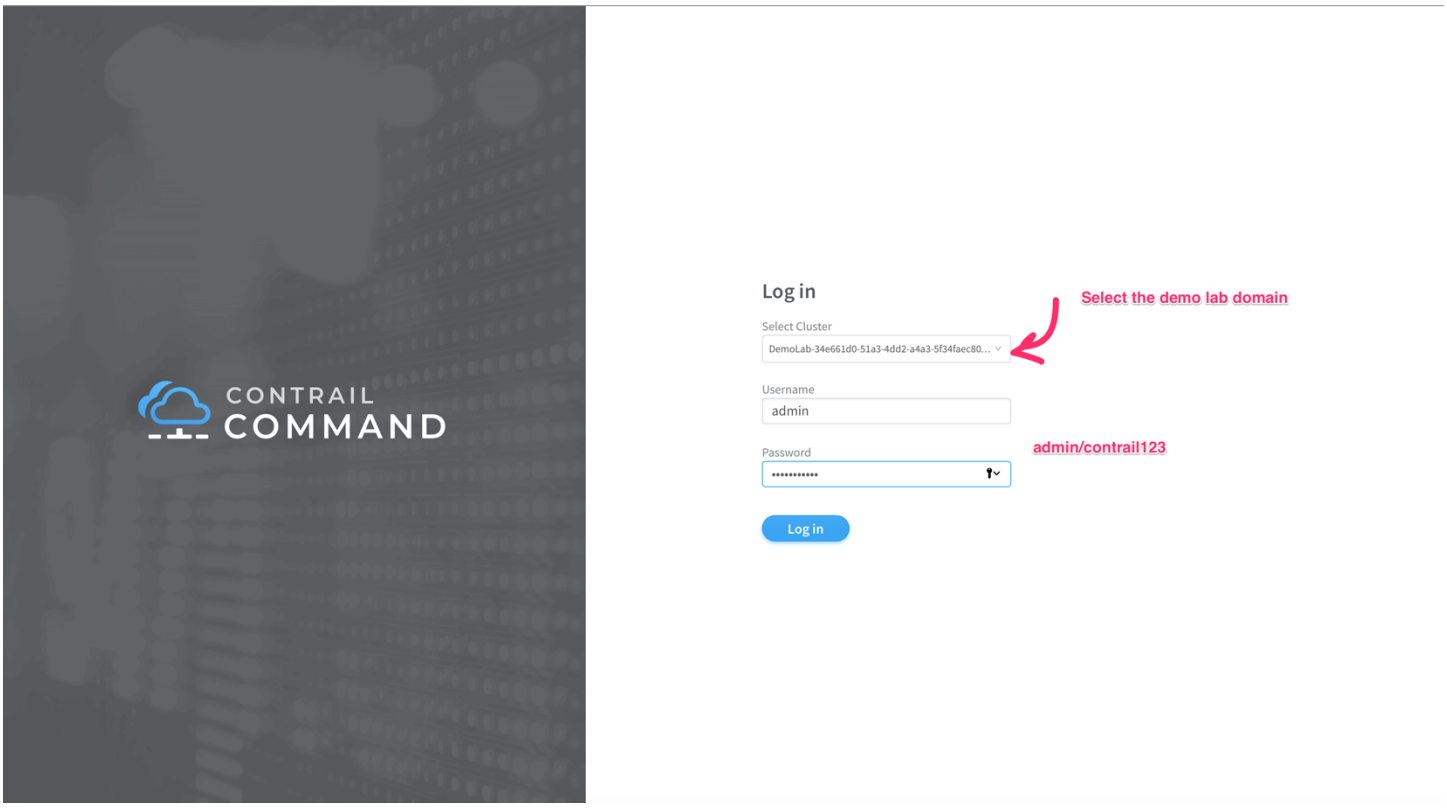
1.0 Verify the full IP reachability between the fabric nodes

- Using the Ravello system provided leafs and spines IP@ verify using the SSH access the full IP reachability between the loopbacks and eBGP peerings are fully operations within the global routing table

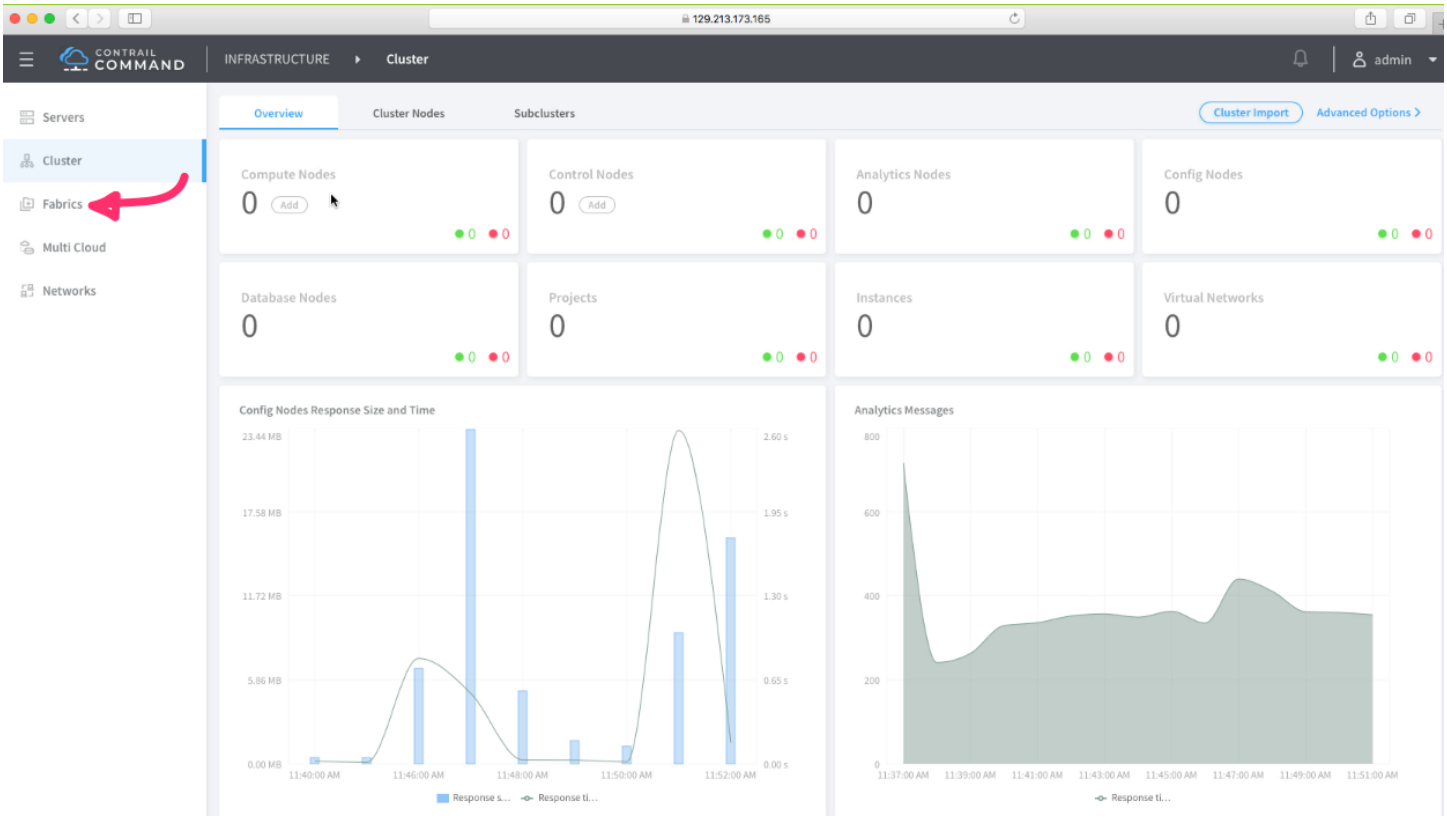
1.1 Auto-discovery of the fabric nodes role assignment

- Check the IP@ of your server6 and connect to the EM 5.1. main dashboard from the Safari or Firefox browser

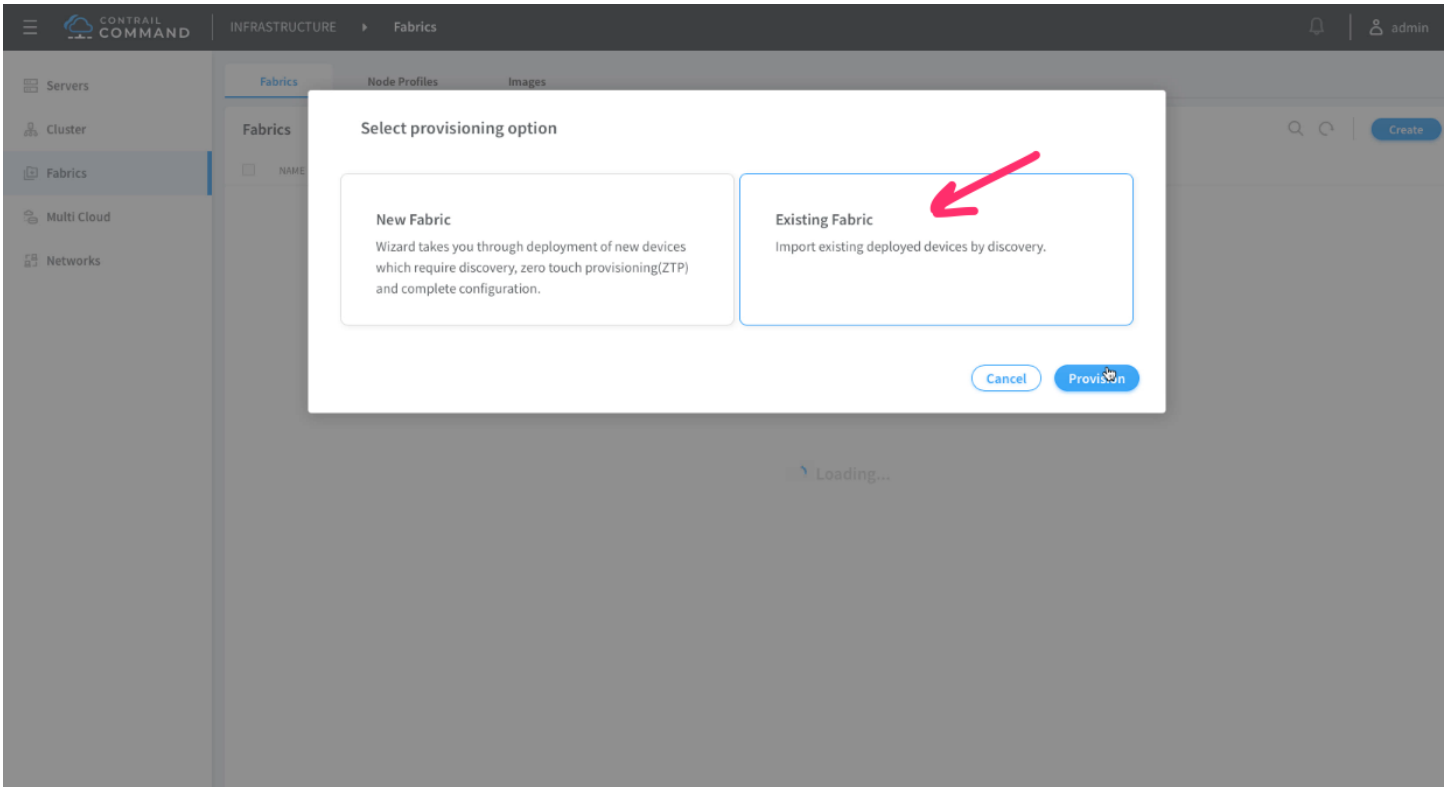
- Log into the EM 5.1 aka CEM fabric management tool

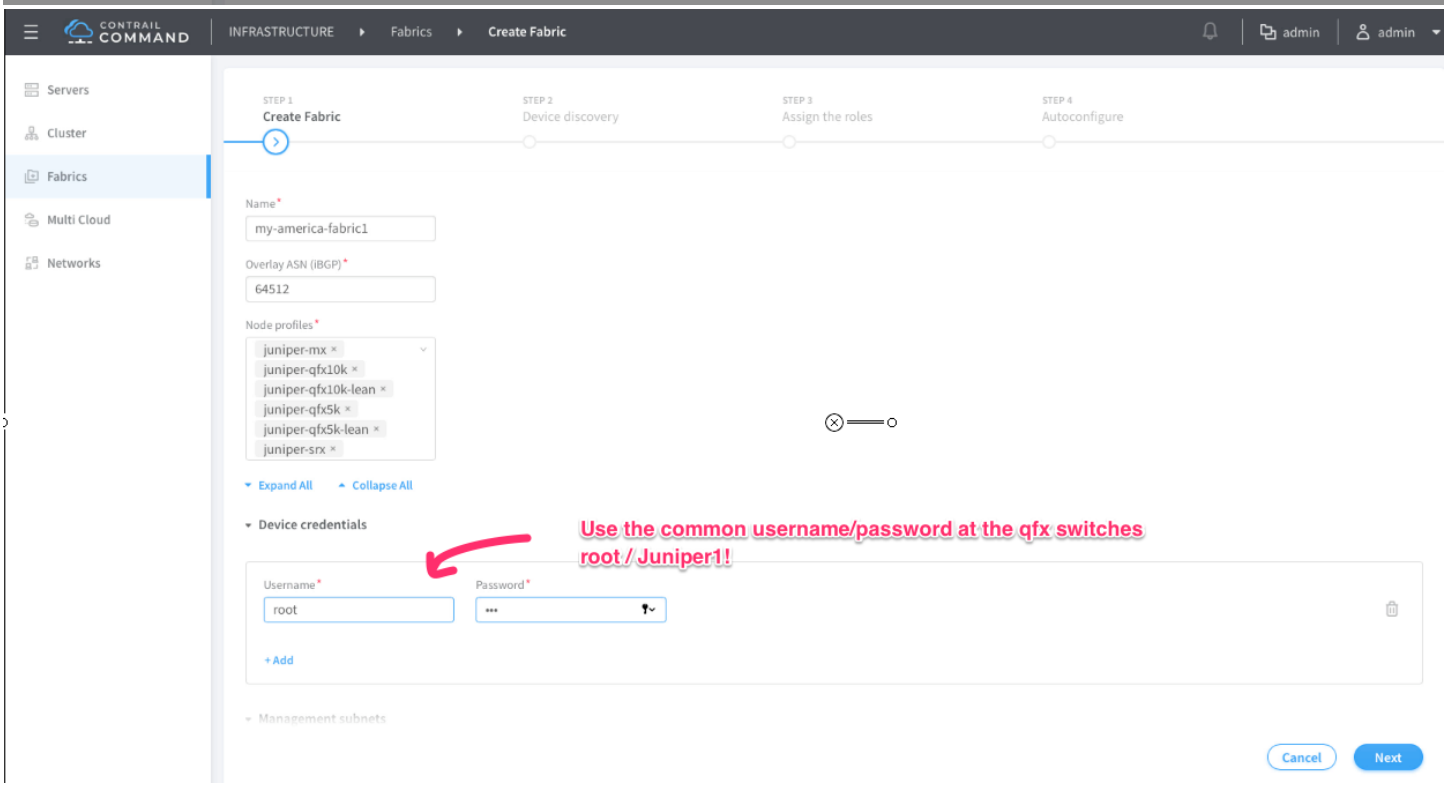


- Once logged-in go to the fabric tab of the EM 5.1 dashboard

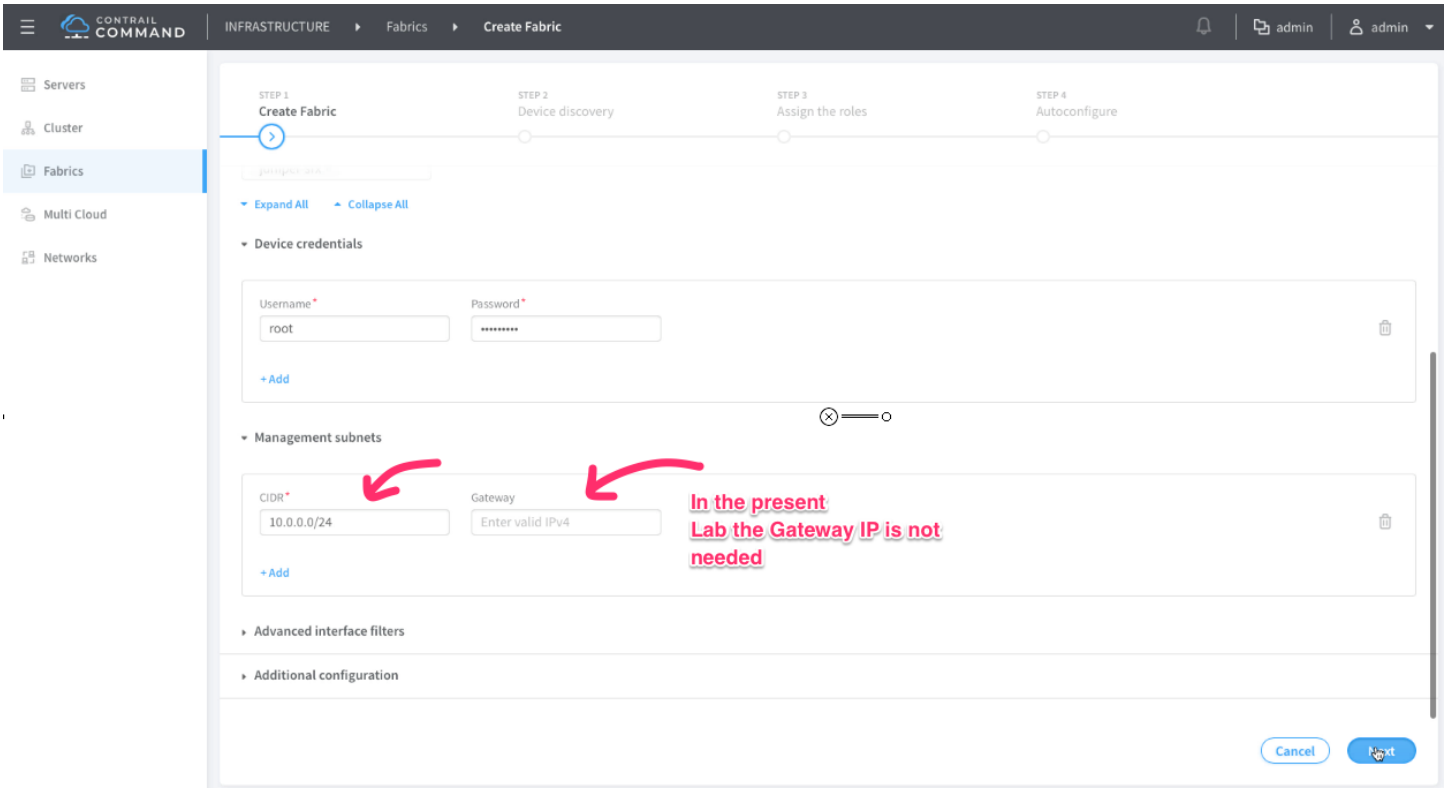


- Then select the “Existing Fabric” option aka ‘Brownfield ‘and click ‘Provision’

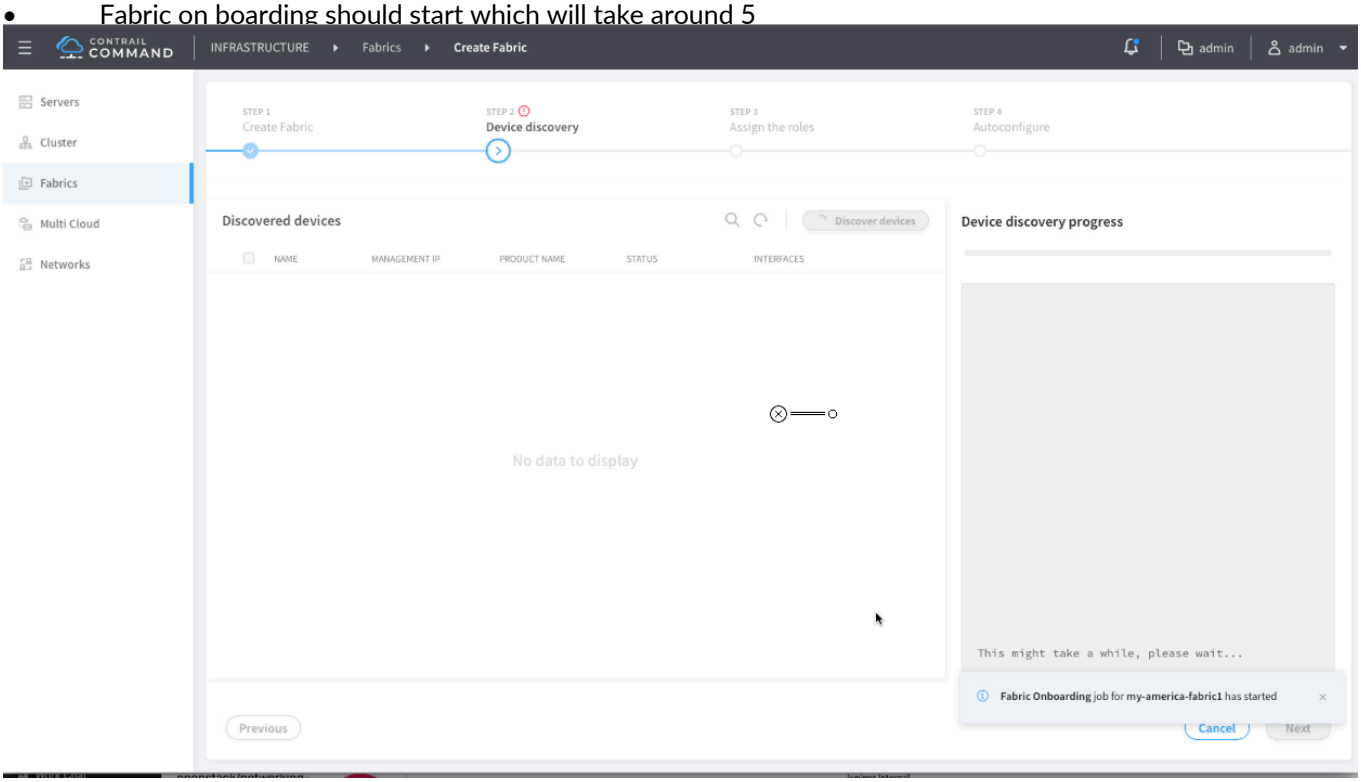




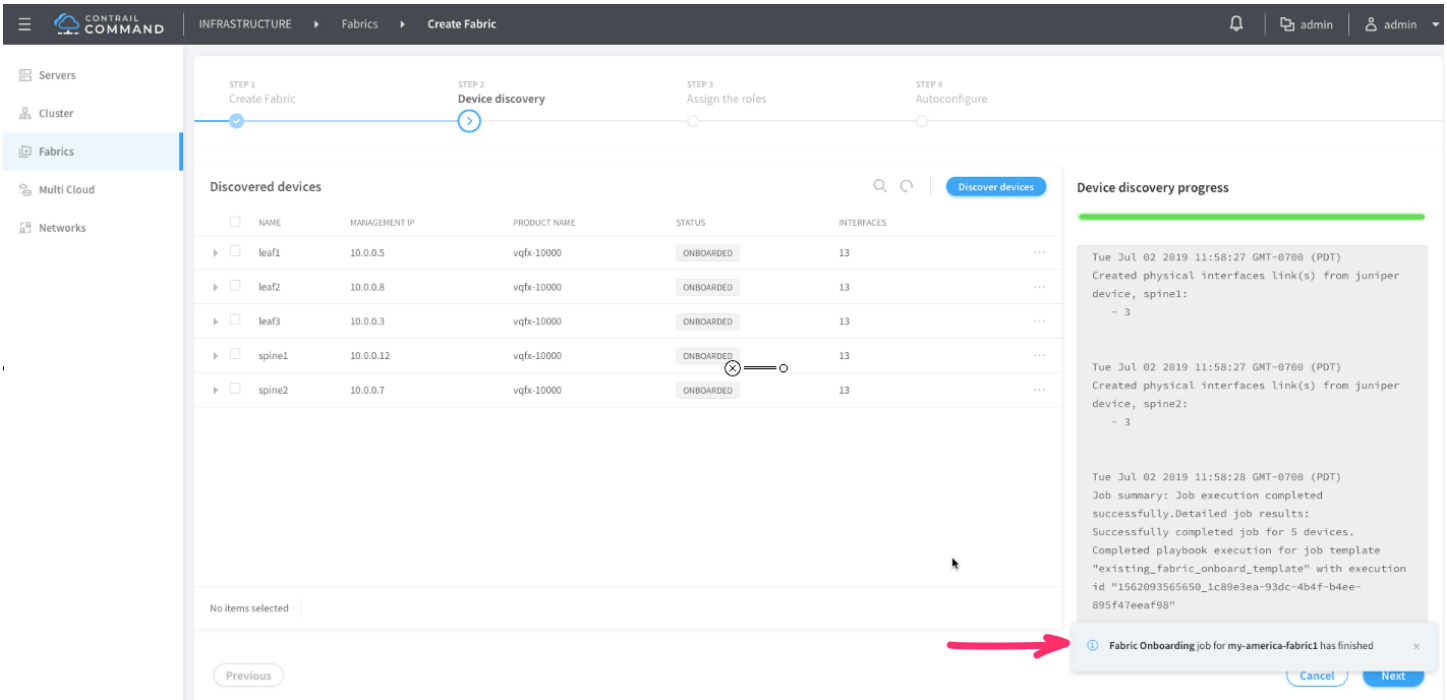
- Use the OOB management subnet that should be used for the auto discovery of the devices



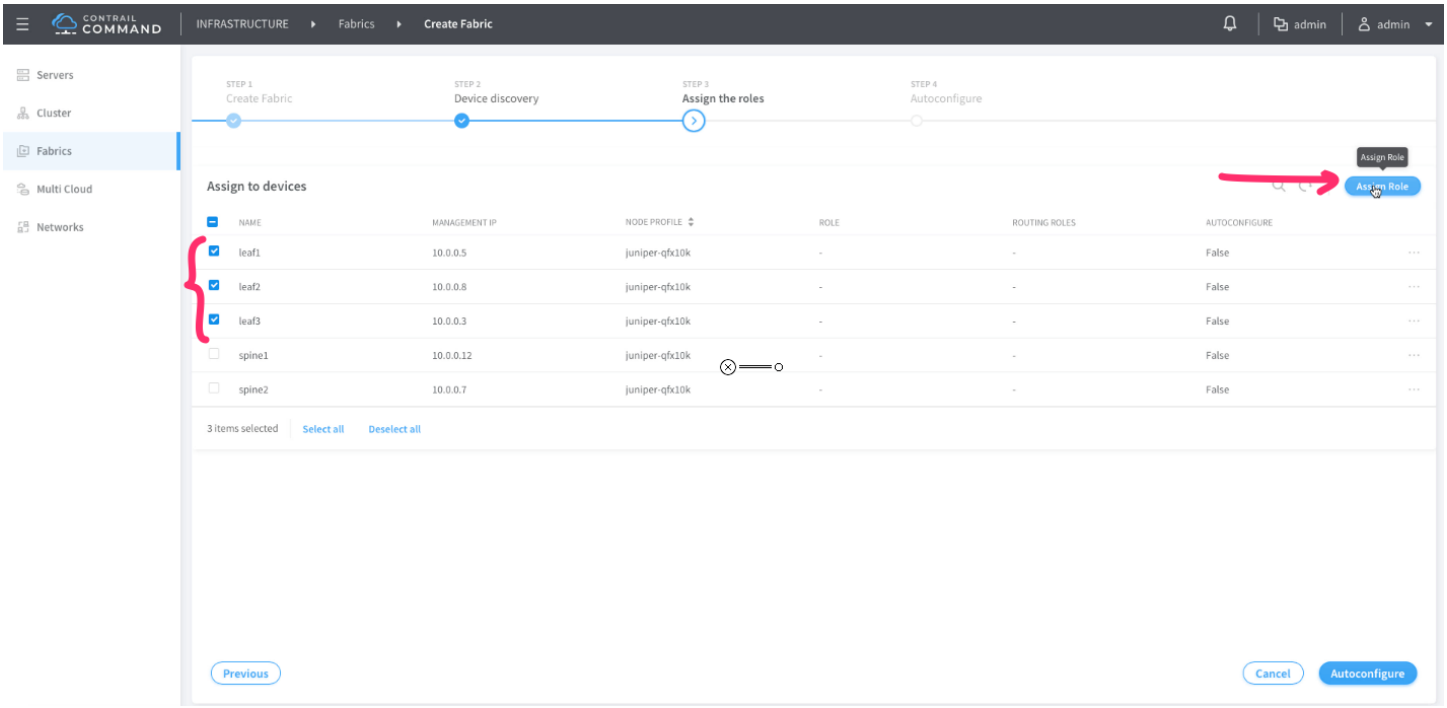
- Fabric on boarding should start which will take around 5

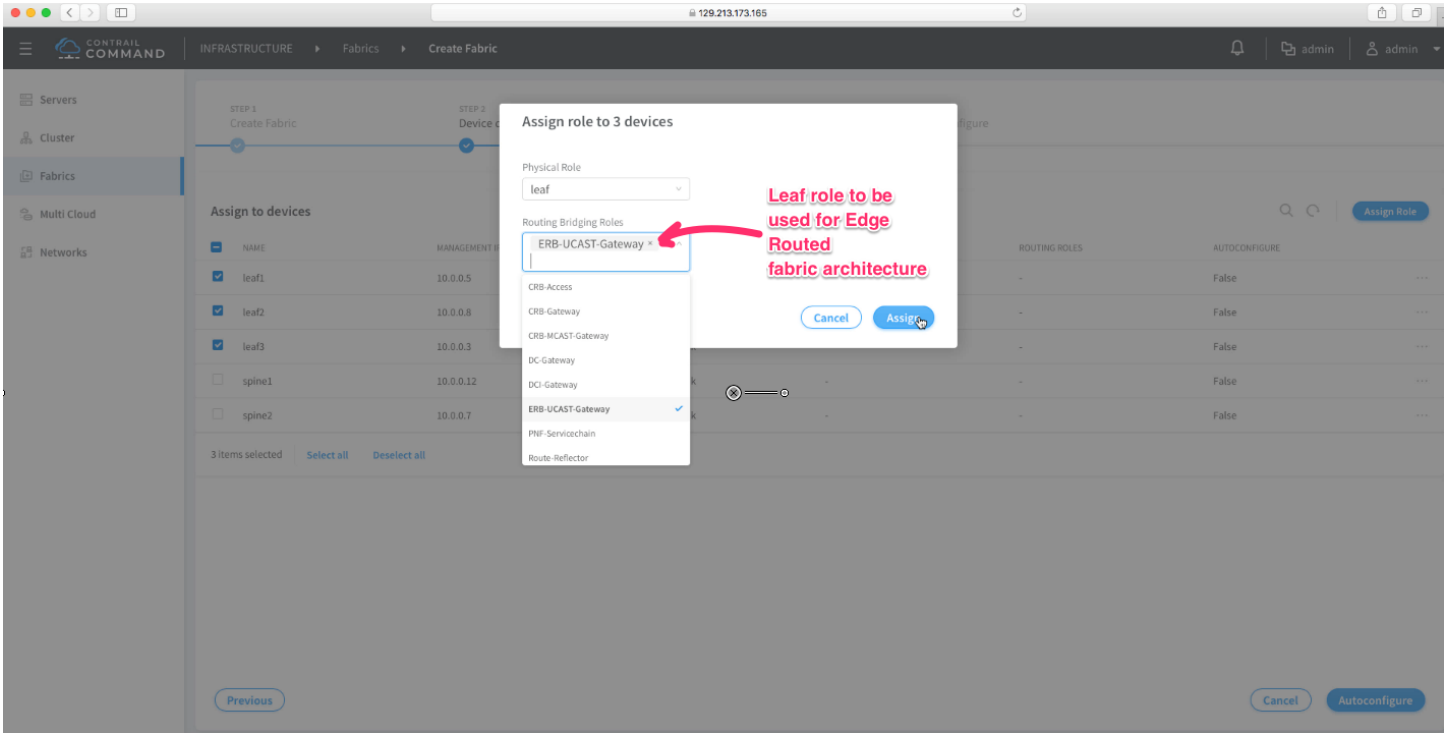


- The devices should be first onboarded and then the role assignment phase will start

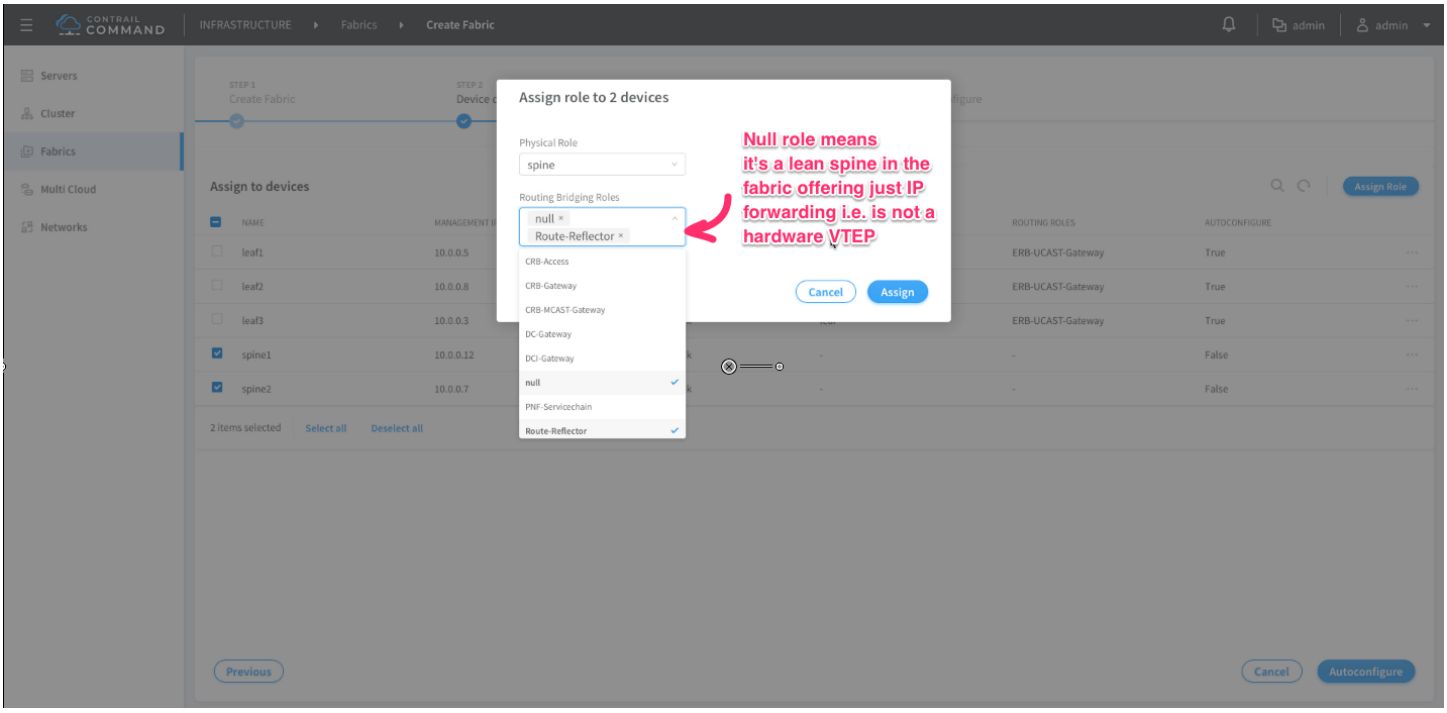


- Select the onboarded leafs and assign the leaf roles

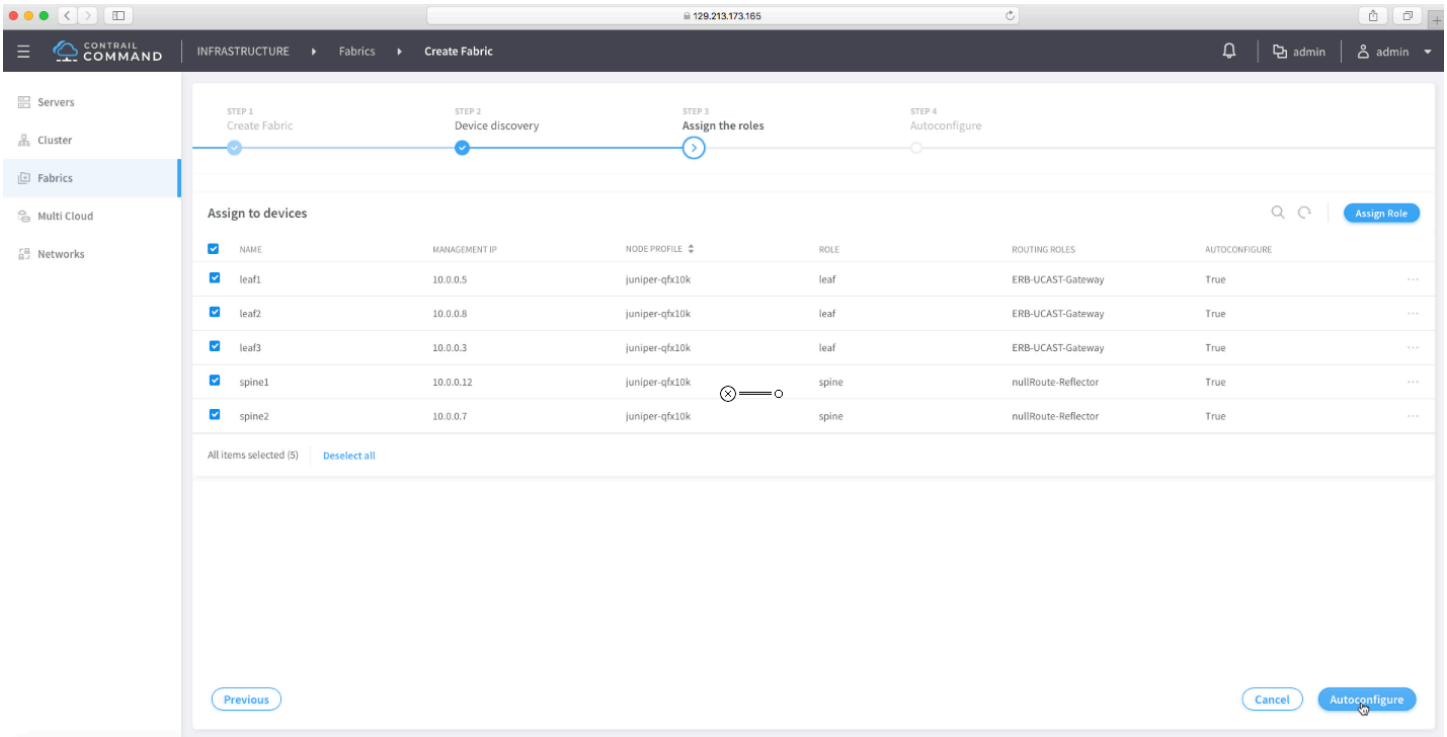




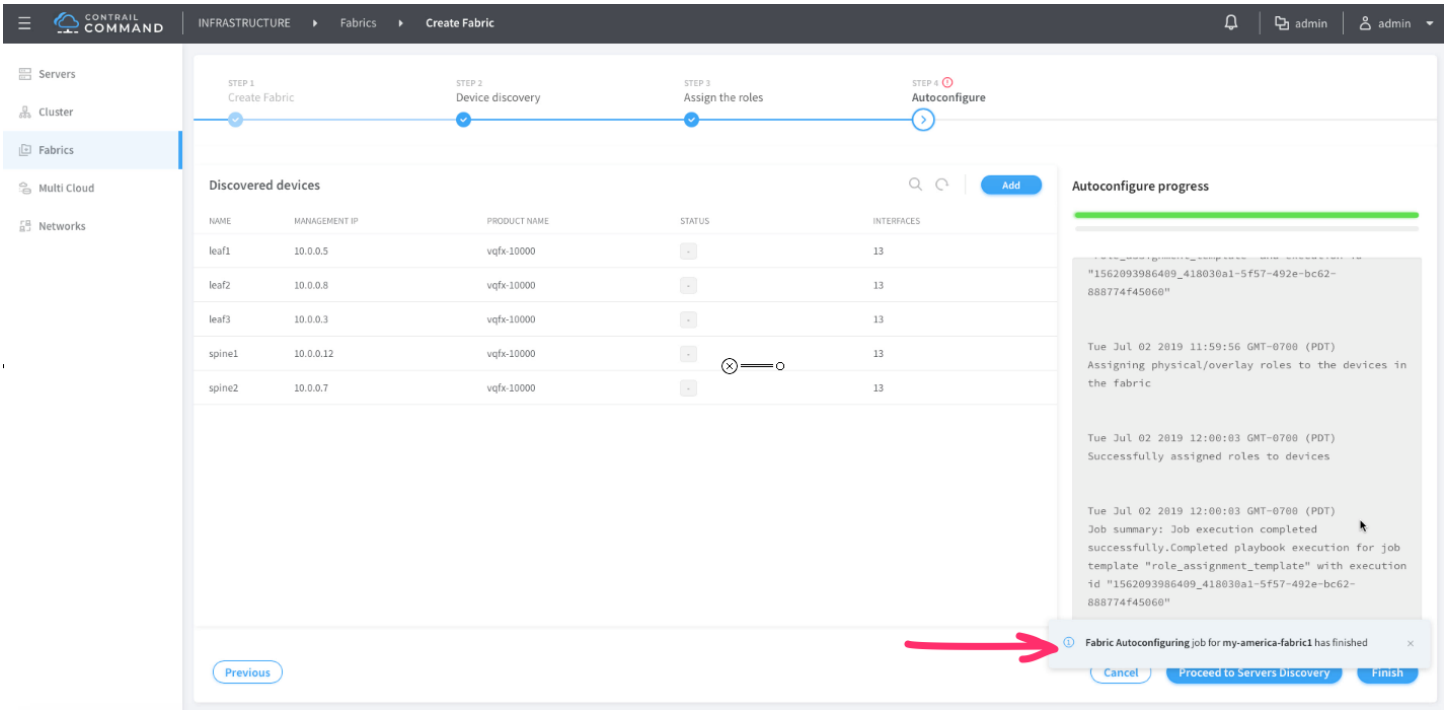
- Select and assign the spine roles in the Edge Routed architecture

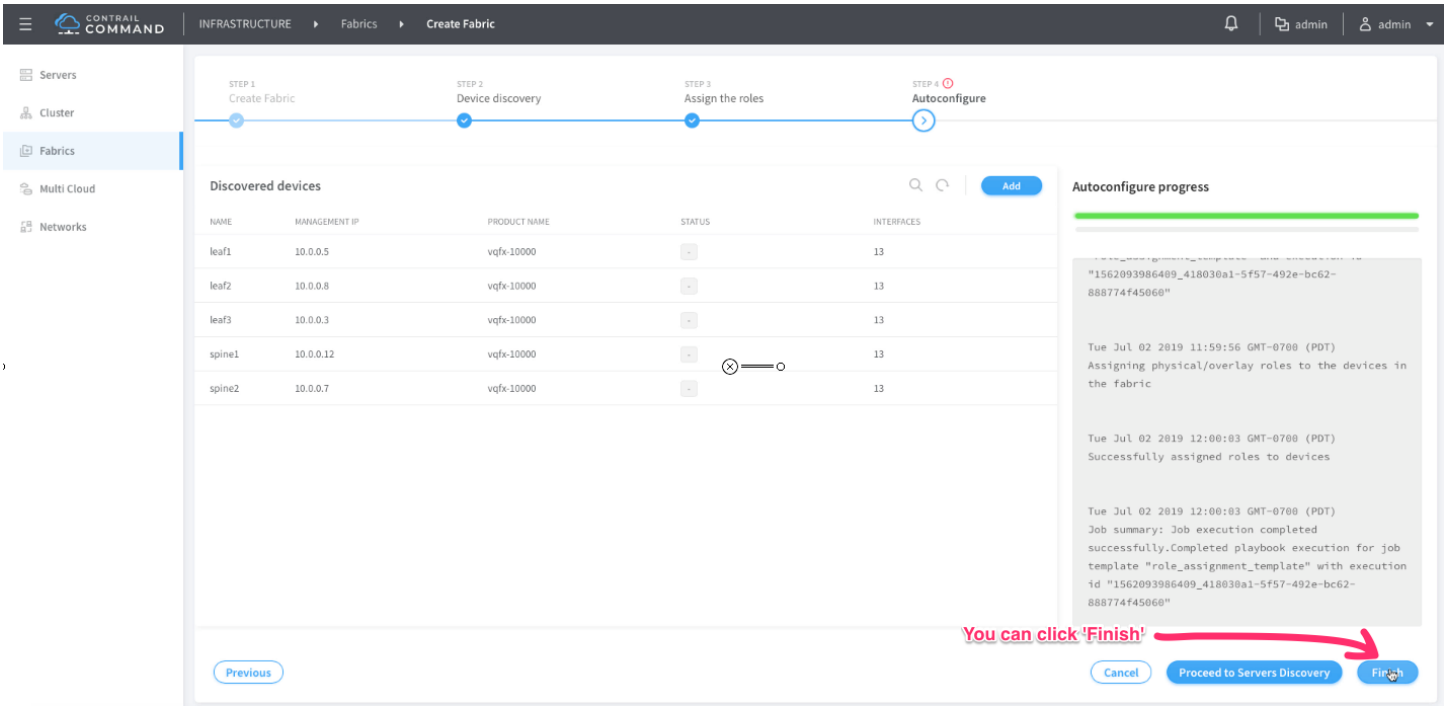


- Then proceed to 'autoconfigure' the new ERB fabric

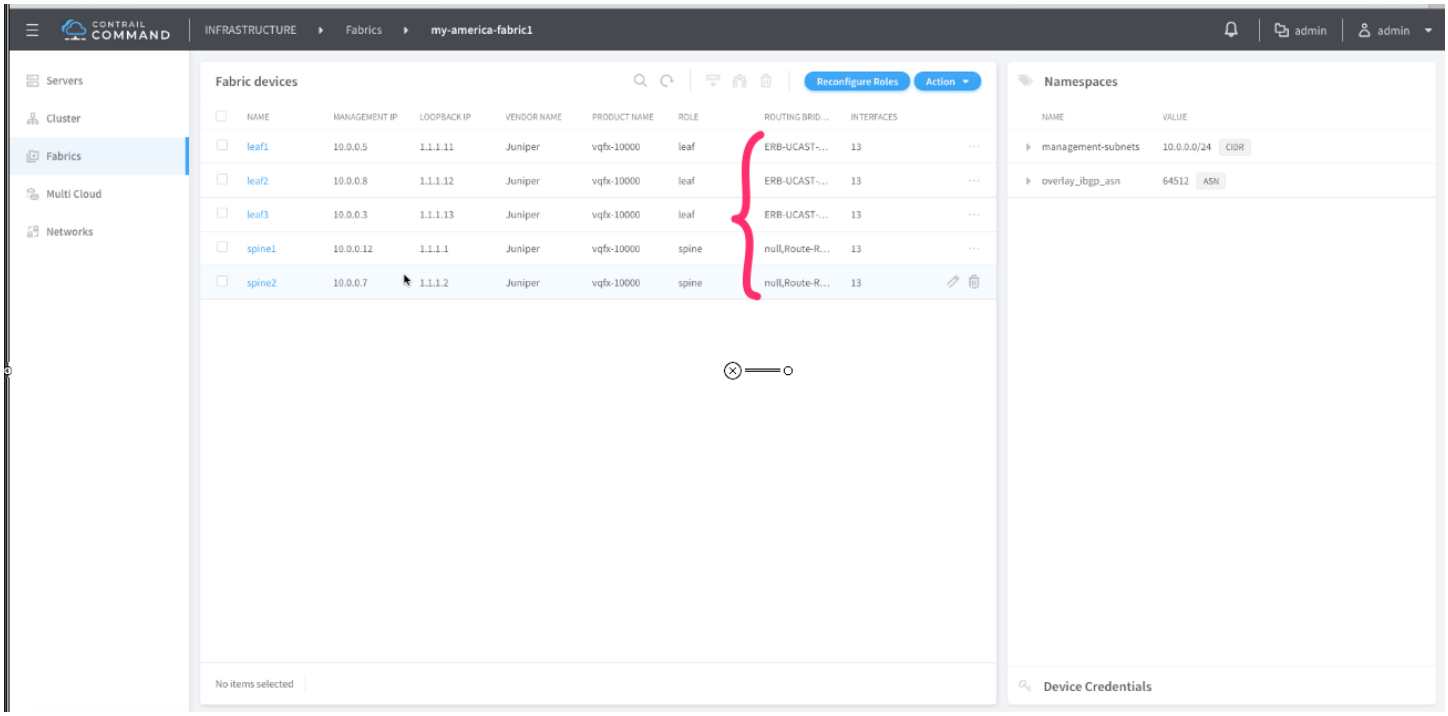


- The log from the tasks execution should show up in the dashboard

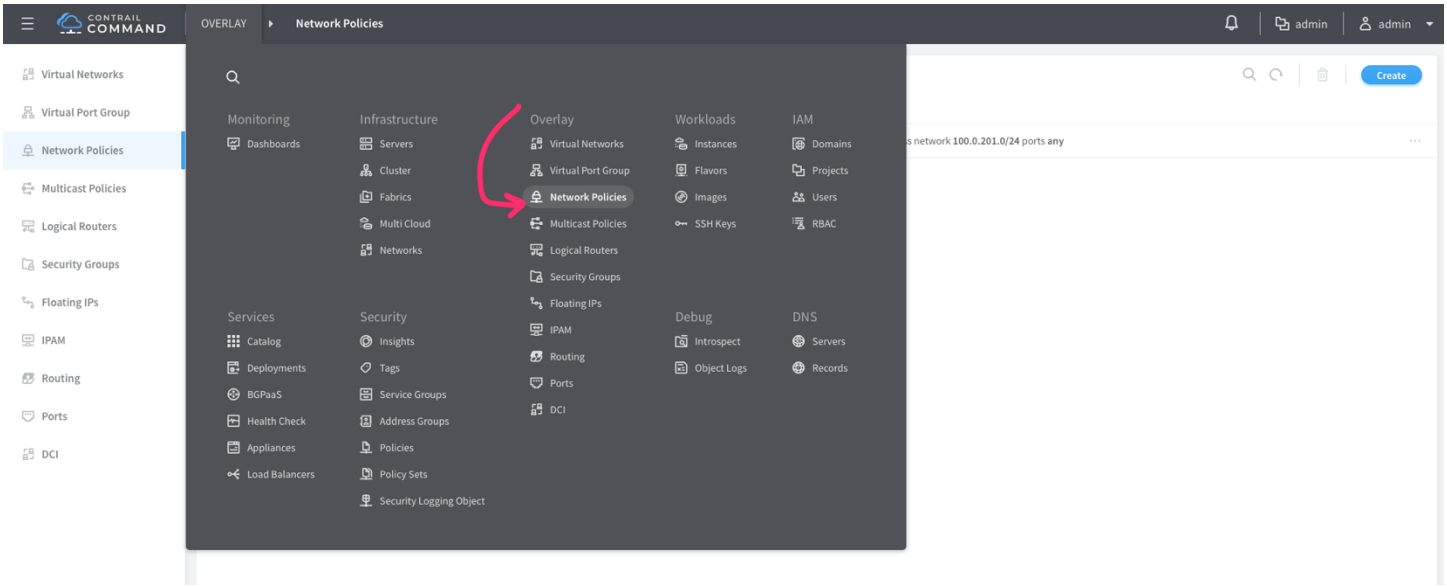


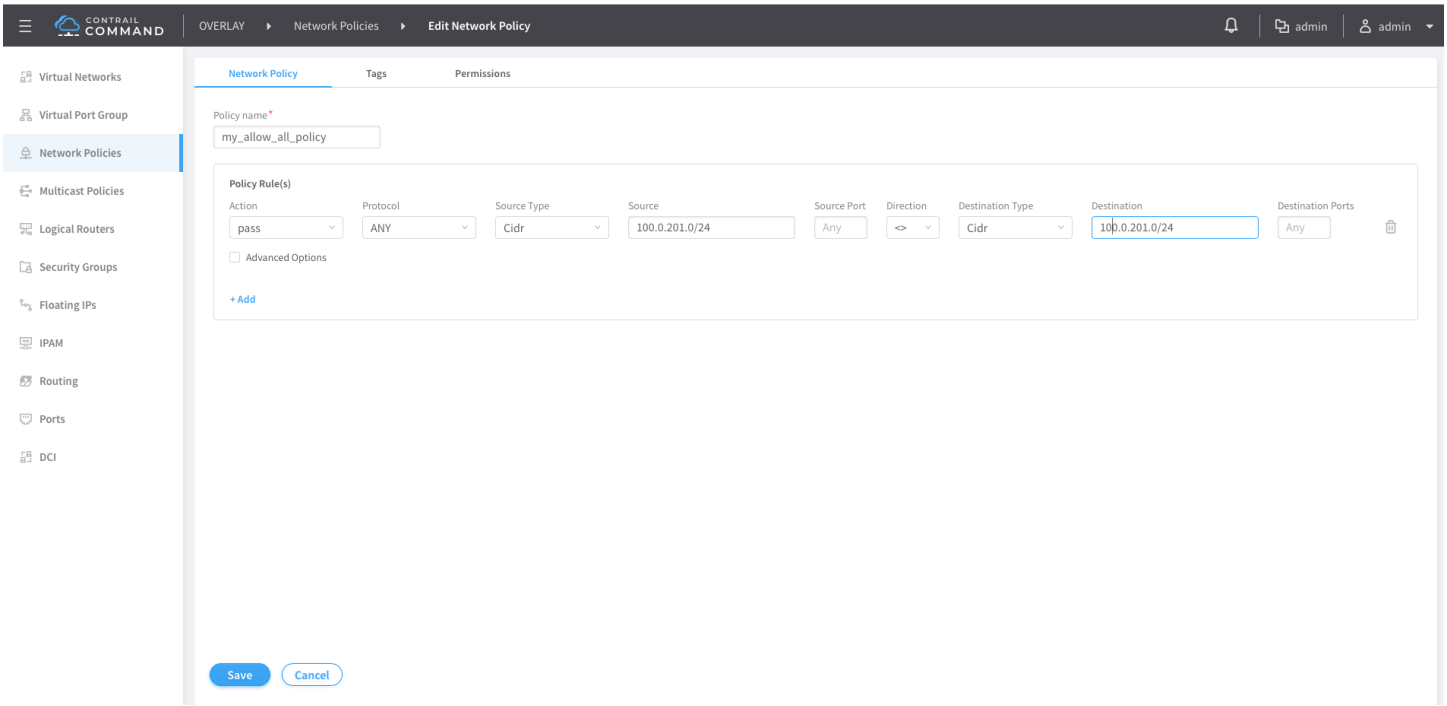


- The roles should be at that moment associated with the given leafs and spines

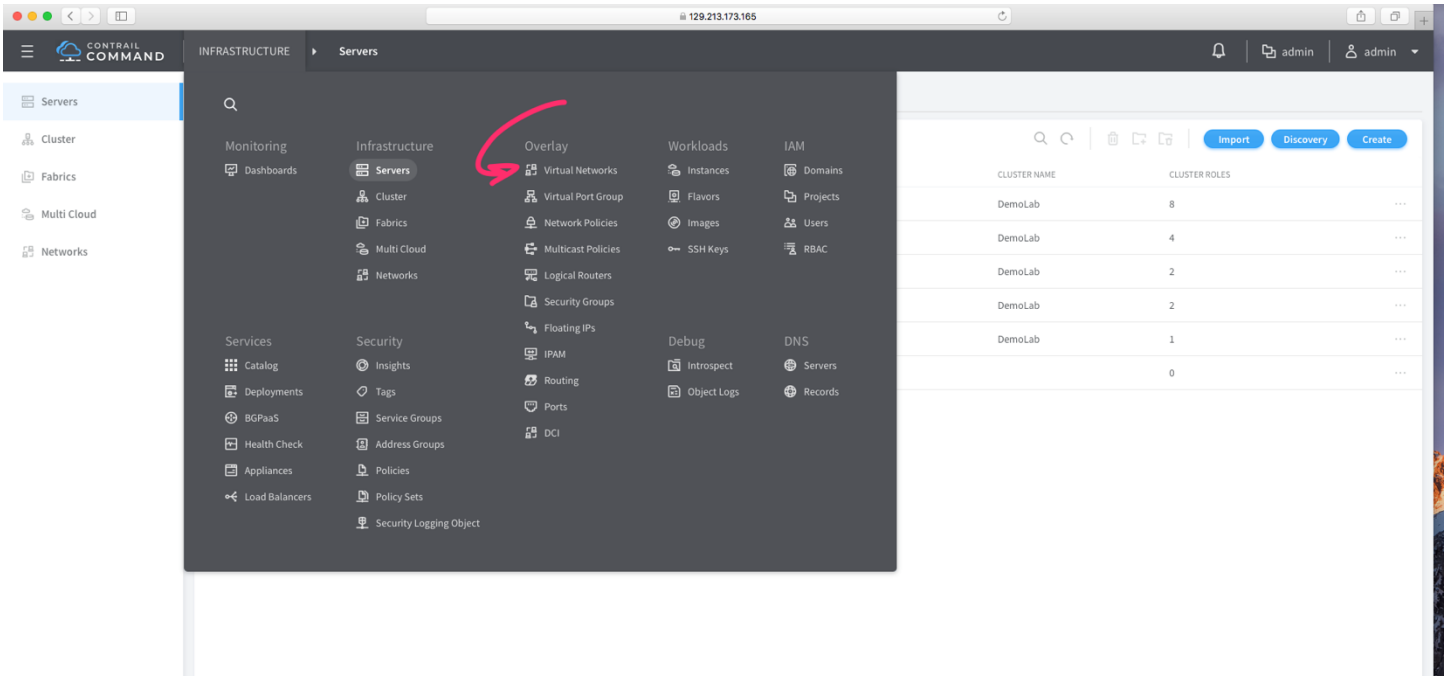


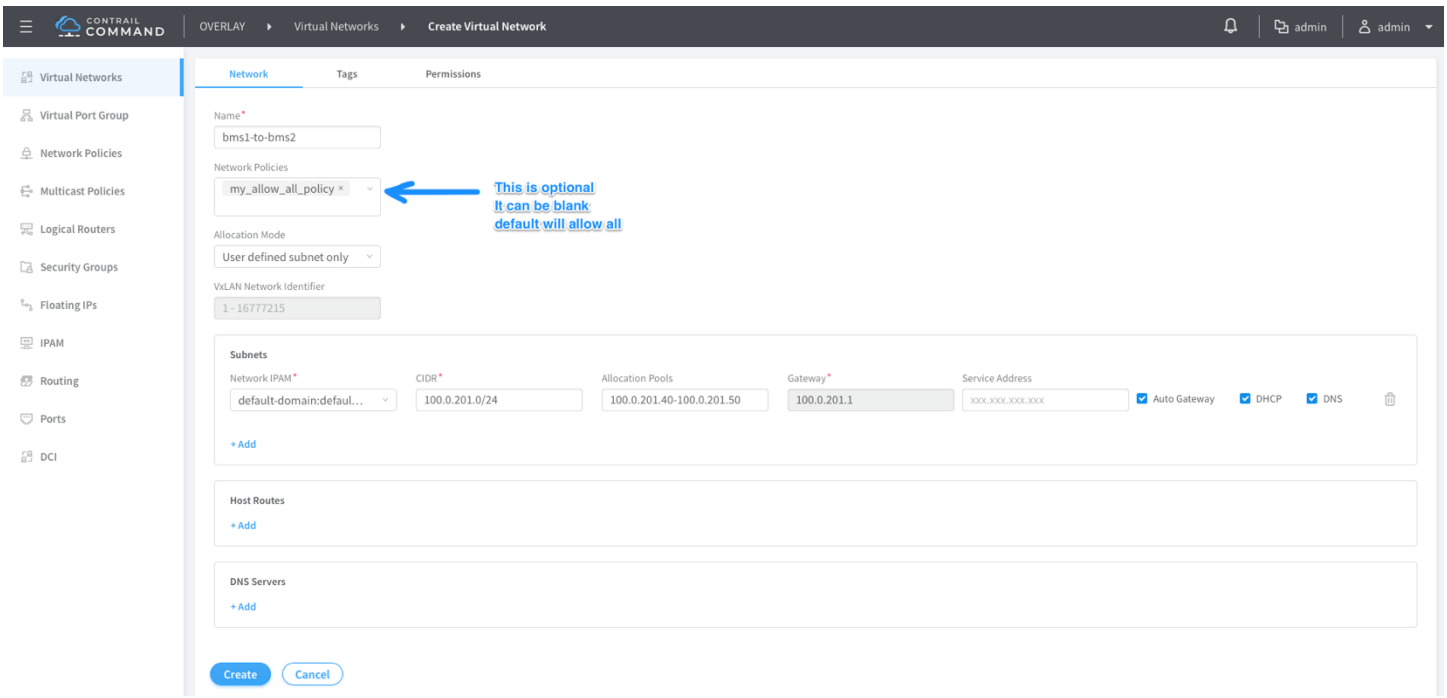
1.3. Enable the Network Policy for BMS-1 to BMS-2 communication (this step is optional - can be skipped)

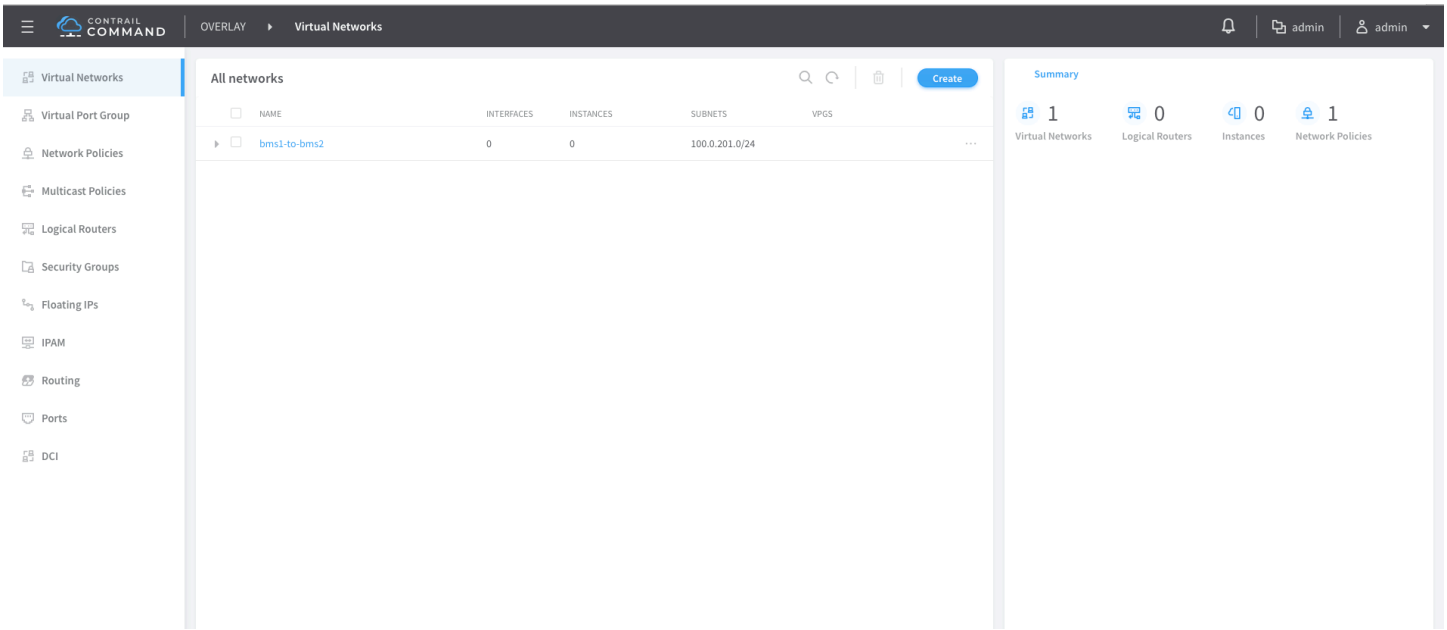




1.4. Create a virtual-network for BMS-1 to BMS-2 communication:

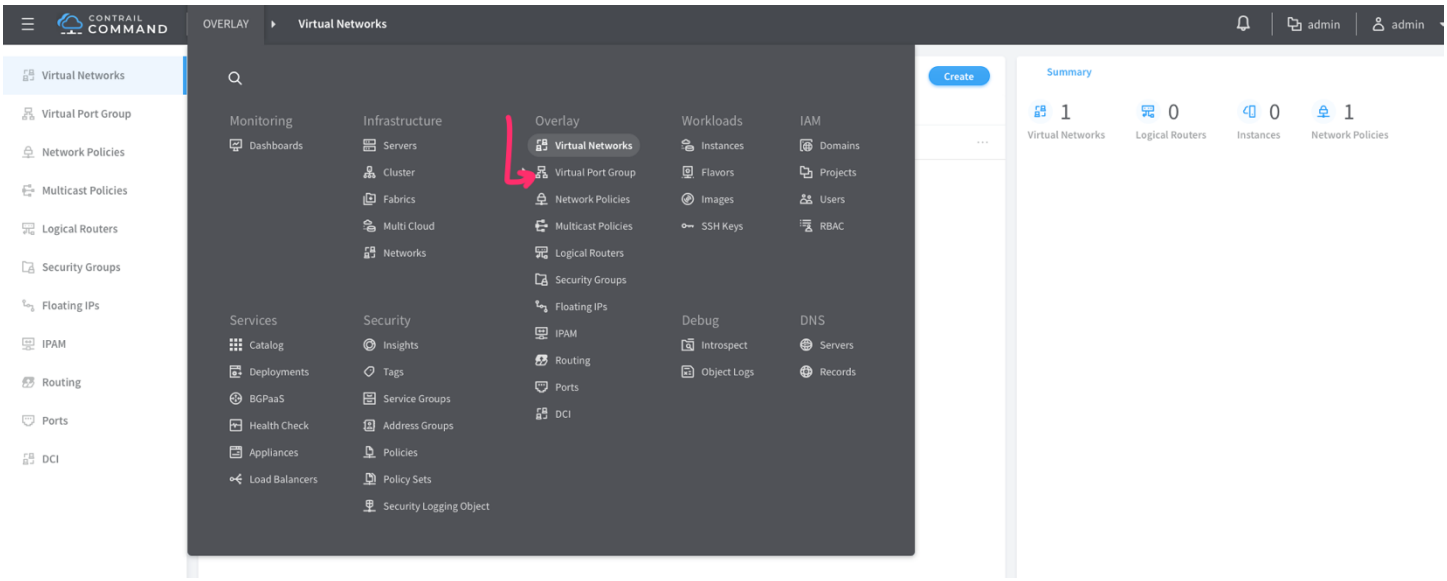




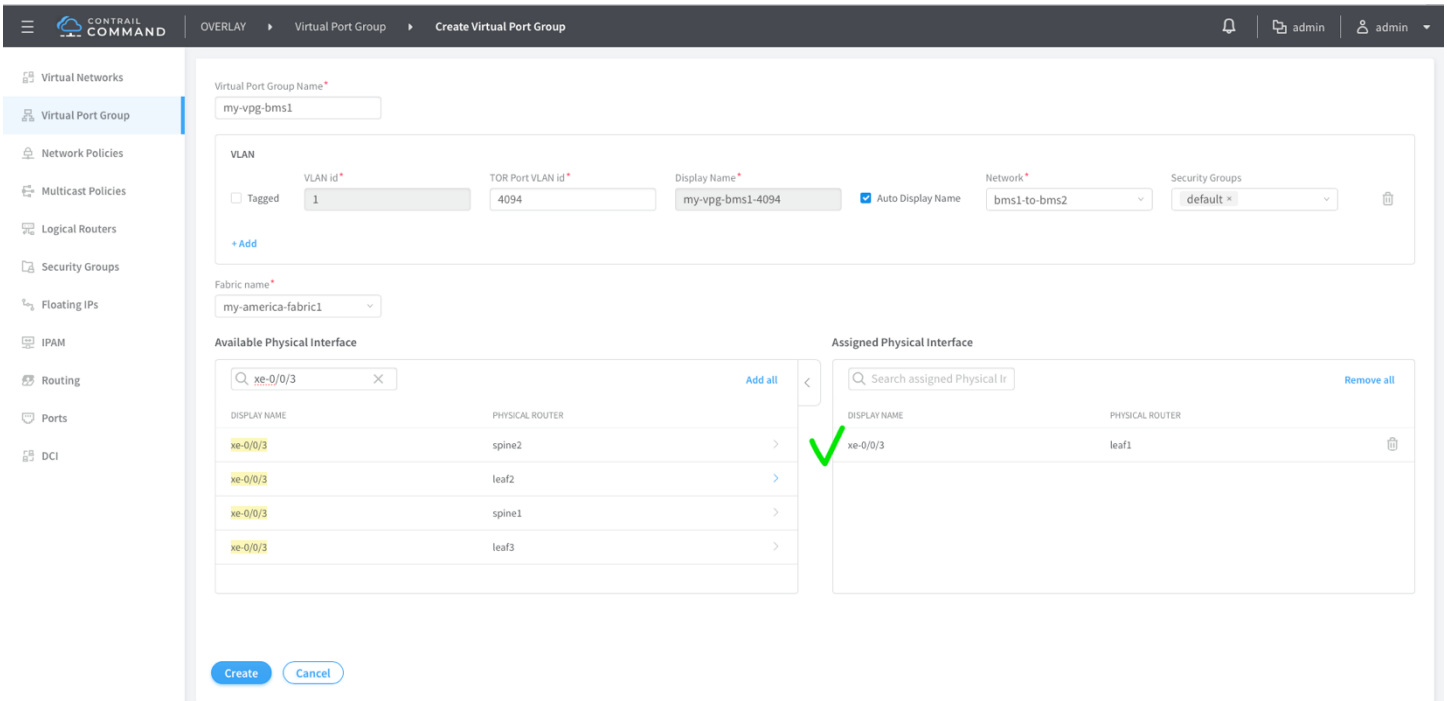


- At that stage because we didn't specify which leaf port is allocated with the virtual-network the JunOS configurations won't be pushed to the leafs yet

1.5. Enable the VPG1 for BMS1 at leaf1 and VPG2 for BMS2 at leaf2 and associate the virtual-network created in previous task



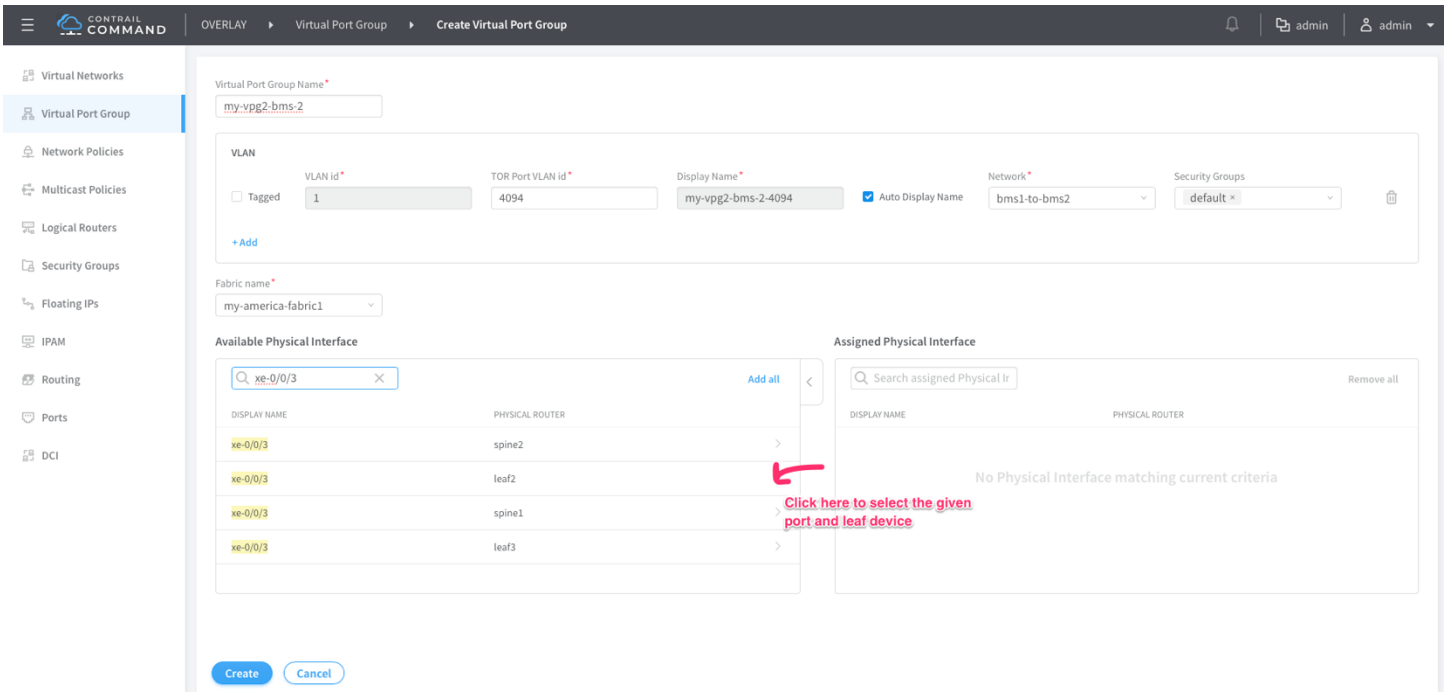
- Associate the VPG port name my-vpg-bms1 with the physical interface connected to BMS-1 as per the topology diagram



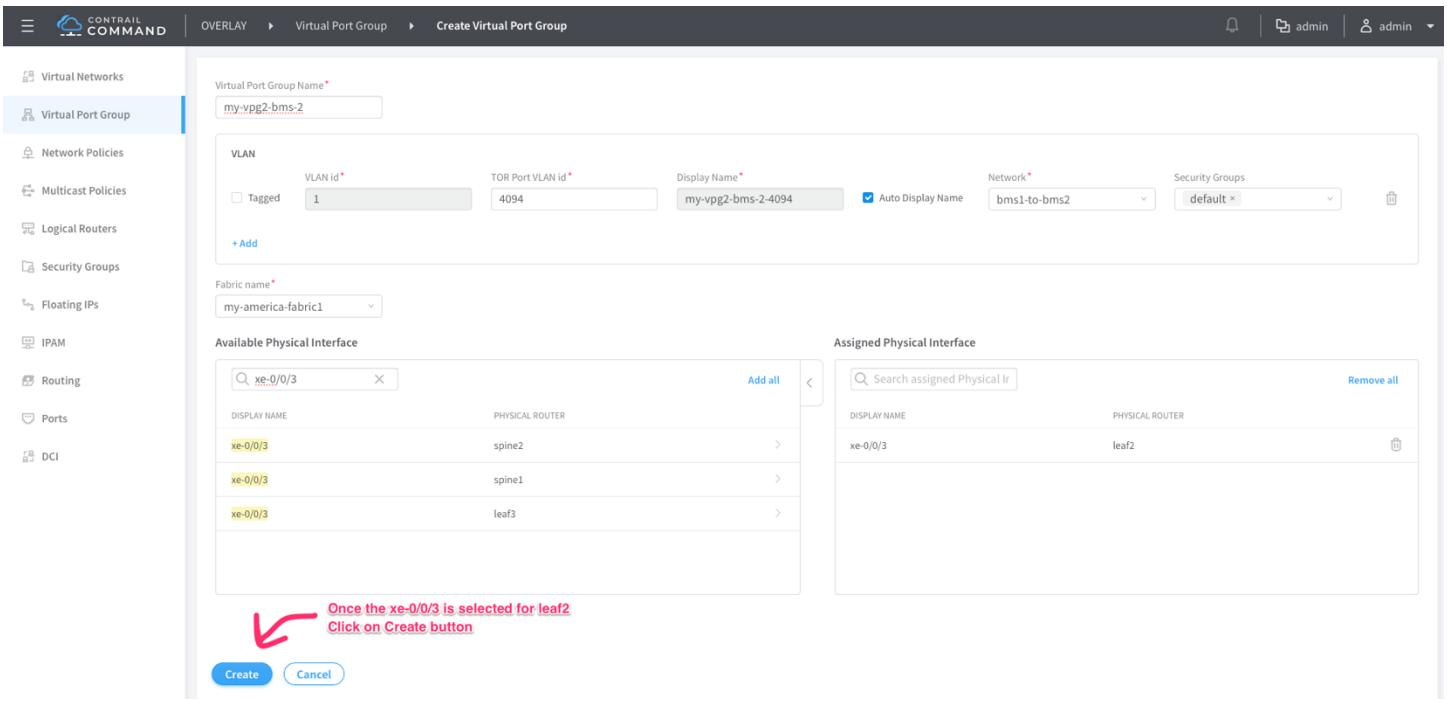
- Once you clicked on ‘create’ the leaf1 should enable the dynamically allocated VLAN with the port and the IRB configuration for the same Virtual-Network will be also pushed

```
root@leaf1> show ethernet-switching interface xe-0/0/3
Routing Instance Name : default-switch
Logical Interface flags (DL - disable learning, AD - packet action drop,
                         LH - MAC limit hit, DN - interface down,
                         MMAS - Mac-move action shutdown,  AS - Autostate-exclude enabled,
                         SCTL - shutdown by Storm-control, MI - MAC+IP limit hit)

Logical         Vlan                    TAG    MAC     MAC+IP  STP          Logical          Tagging
interface       members                        limit   limit   state        interface flags
xe-0/0/3.0                                     8192    8192                                  untagged
                bd-6                    4094   1024    1024    Forwarding                    untagged

{master:0}
root@leaf1> show interfaces irb terse routing-instance all
Interface         Admin Link Proto          Local                 Instance
irb.6             up    up   inet       100.0.201.1/24
                                                                  default

{master:0}
root@leaf1>
```

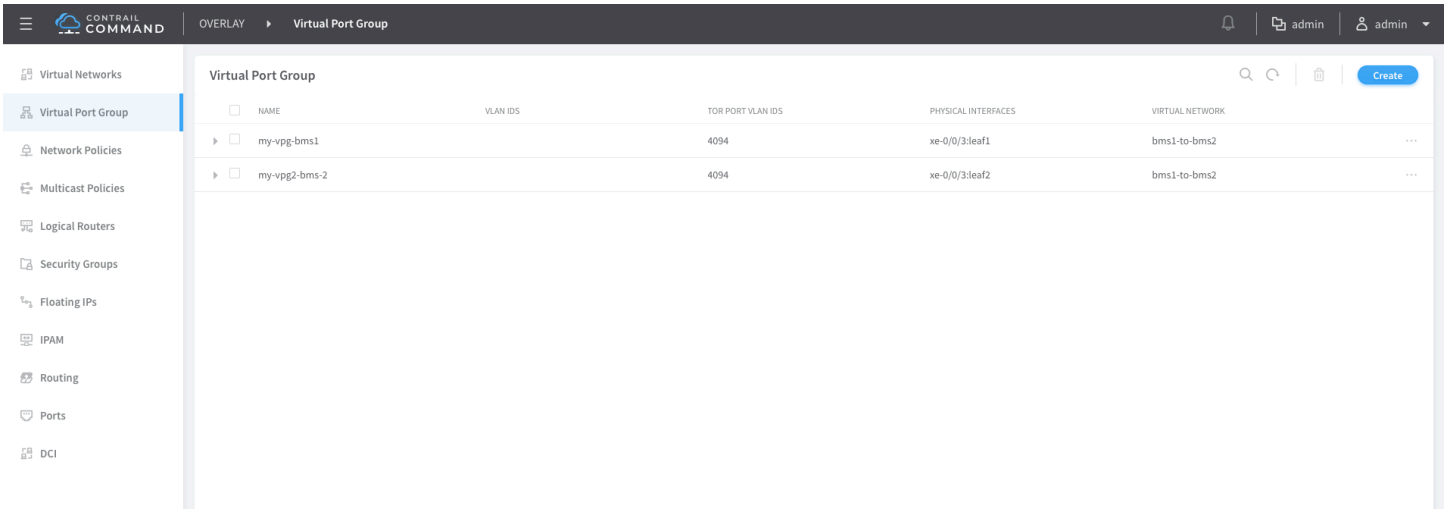
- Then repeat the same task for the BMS-2 to leaf-2 connectivity by creating a dedicated VPG port name and associating the physical interfaces from the selected leaf device



- Associate the my-vpg2-bms2 port name my-vpg-bms2 with the physical interface connected to BMS-2 as per the topology diagram



- At that stage both BMS-1 and BMS-2 VPG ports should be created and the BMS-1 to BMS-2 ping should work



- Here are the example of on how to verify if the given port was provisioned with the given bridge-domain bd-6 (it can be a different value as the bd-id and vlan are dynamically allocated for the given virtual-network)

```
root@leaf2> show ethernet-switching interface xe-0/0/3
Routing Instance Name : default-switch
Logical Interface flags (DL - disable learning, AD - packet action drop,
                         LH - MAC limit hit, DN - interface down,
                         MMAS - Mac-move action shutdown,  AS - Autostate-exclude enabled,
                         SCTL - shutdown by Storm-control, MI - MAC+IP limit hit)

Logical         Vlan                  TAG   MAC    MAC+IP STP         Logical          Tagging
interface       members                     limit  limit  state       interface flags
xe-0/0/3.0                                  8192   8192                                untagged
                bd-6                  4094  1024   1024   Forwarding                   untagged

{master:0}
root@leaf2>

{master:0}
root@leaf2> show interfaces irb terse routing-instance all
Interface        Admin Link Proto          Local               Instance
irb.6            up    up   inet      100.0.201.1/24
                                                              default

{master:0}
root@leaf2>
```

```
root@leaf2> show vlans

Routing instance        VLAN name             Tag          Interfaces
default-switch          bd-6                  4094
                                                           vtep.32769*
                                                           xe-0/0/3.0*

{master:0}
root@leaf2>
```

```
root@leaf1> show ethernet-switching vxlan-tunnel-end-point remote ip 1.1.1.12

MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC , P -Pinned MAC)

Logical system   : <default>
Routing instance : default-switch
 Bridging domain : bd-6+4094, VLAN : 4094
 VXLAN ID : 6, Multicast Group IP : 0.0.0.0
 Remote VTEP : 1.1.1.12, Nexthop ID : 1772
   MAC                MAC      Logical          Remote VTEP
   address            flags    interface        IP address
   2c:c2:60:42:5d:cc  D        vtep.32769       1.1.1.12

{master:0}
root@leaf1>
```

```
root@leaf2> show ethernet-switching vxlan-tunnel-end-point remote ip 1.1.1.11

MAC flags (S -static MAC, D -dynamic MAC, L -locally learned, C -Control MAC
           SE -Statistics enabled, NM -Non configured MAC, R -Remote PE MAC , P -Pinned MAC)

Logical system   : <default>
Routing instance : default-switch
 Bridging domain : bd-6+4094, VLAN : 4094
 VXLAN ID : 6, Multicast Group IP : 0.0.0.0
 Remote VTEP : 1.1.1.11, Nexthop ID : 1771

{master:0}
root@leaf2>

root@leaf1> show ethernet-switching table

MAC flags (S - static MAC, D - dynamic MAC, L - locally learned, P - Persistent static
           SE - statistics enabled, NM - non configured MAC, R - remote PE MAC, O - ovsdb MAC)


Ethernet switching table : 2 entries, 2 learned
Routing instance : default-switch
   Vlan                MAC                 MAC      Logical                 Active
   name                address             flags    interface               source
   bd-6                2c:c2:60:42:5d:cc   D        vtep.32769              1.1.1.12
   bd-6                2c:c2:60:63:51:e4   D        xe-0/0/3.0

{master:0}
root@leaf1>

root@leaf1> show route evpn-mac-address 2c:c2:60:42:5d:cc active-path table default-switch.evpn.0

default-switch.evpn.0: 6 destinations, 9 routes (6 active, 0 holddown, 0 hidden)
+ = Active Route, - = Last Active, * = Both

2:1.1.1.12:7999::6::2c:c2:60:42:5d:cc/304 MAC/IP
                   *[BGP/170] 00:43:57, localpref 100, from 1.1.1.1
                      AS path: I, validation-state: unverified
                    > to 172.16.0.0 via xe-0/0/0.0
                      to 172.16.0.2 via xe-0/0/1.0
2:1.1.1.12:7999::6::2c:c2:60:42:5d:cc::100.0.201.3/304 MAC/IP
                   *[BGP/170] 00:43:57, localpref 100, from 1.1.1.1
                      AS path: I, validation-state: unverified
                      to 172.16.0.0 via xe-0/0/0.0
                    > to 172.16.0.2 via xe-0/0/1.0

{master:0}
root@leaf1>
```

```
root@leaf1> show route evpn-mac-address 2c:c2:60:42:5d:cc active-path table default-switch.evpn.0 match-prefix 2:1.1.1.12:7999::6::2c:c2:60:42:5d:cc::100.0.201.3/304 extensive

default-switch.evpn.0: 6 destinations, 9 routes (6 active, 0 holddown, 0 hidden)
2:1.1.1.12:7999::6::2c:c2:60:42:5d:cc::100.0.201.3/304 MAC/IP (2 entries, 1 announced)
        *BGP    Preference: 170/-101
                Route Distinguisher: 1.1.1.12:7999
                Next hop type: Indirect, Next hop index: 0
                Address: 0xbfaad10
                Next-hop reference count: 12
                Source: 1.1.1.1
                Protocol next hop: 1.1.1.12
                Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                State: <Secondary Active Int Ext>
                Local AS: 64512 Peer AS: 64512
                Age: 45:04      Metric2: 0
                Validation State: unverified
                Task: BGP_64512.1.1.1.1
                Announcement bits (1): 0-default-switch-evpn
                AS path: I  (Originator)
                Cluster list:  0.0.0.100
                Originator ID: 1.1.1.12
                Communities: target:64512:8000005 encapsulation:vxlan(0x8)
                Import Accepted
                Route Label: 6
                ESI: 00:00:00:00:00:00:00:00:00:00
                Localpref: 100
                Router ID: 1.1.1.1
                Primary Routing Table bgp.evpn.0
                Indirect next hops: 1
                        Protocol next hop: 1.1.1.12
                        Indirect next hop: 0x2 no-forward INH Session ID: 0x0
                        Indirect path forwarding next hops: 2
                                Next hop type: Router
                                Next hop: 172.16.0.0 via xe-0/0/0.0
                                Session Id: 0x0
                                Next hop: 172.16.0.2 via xe-0/0/1.0
                                Session Id: 0x0
                                1.1.1.12/32 Originating RIB: inet.0
                                  Node path count: 1
                                  Forwarding nexthops: 2
                                        Nexthop: 172.16.0.0 via xe-0/0/0.0
                                        Session Id: 0
                                        Nexthop: 172.16.0.2 via xe-0/0/1.0
                                        Session Id: 0

{master:0}
root@leaf1>
```

leaf2 lo0.0

Single homed server

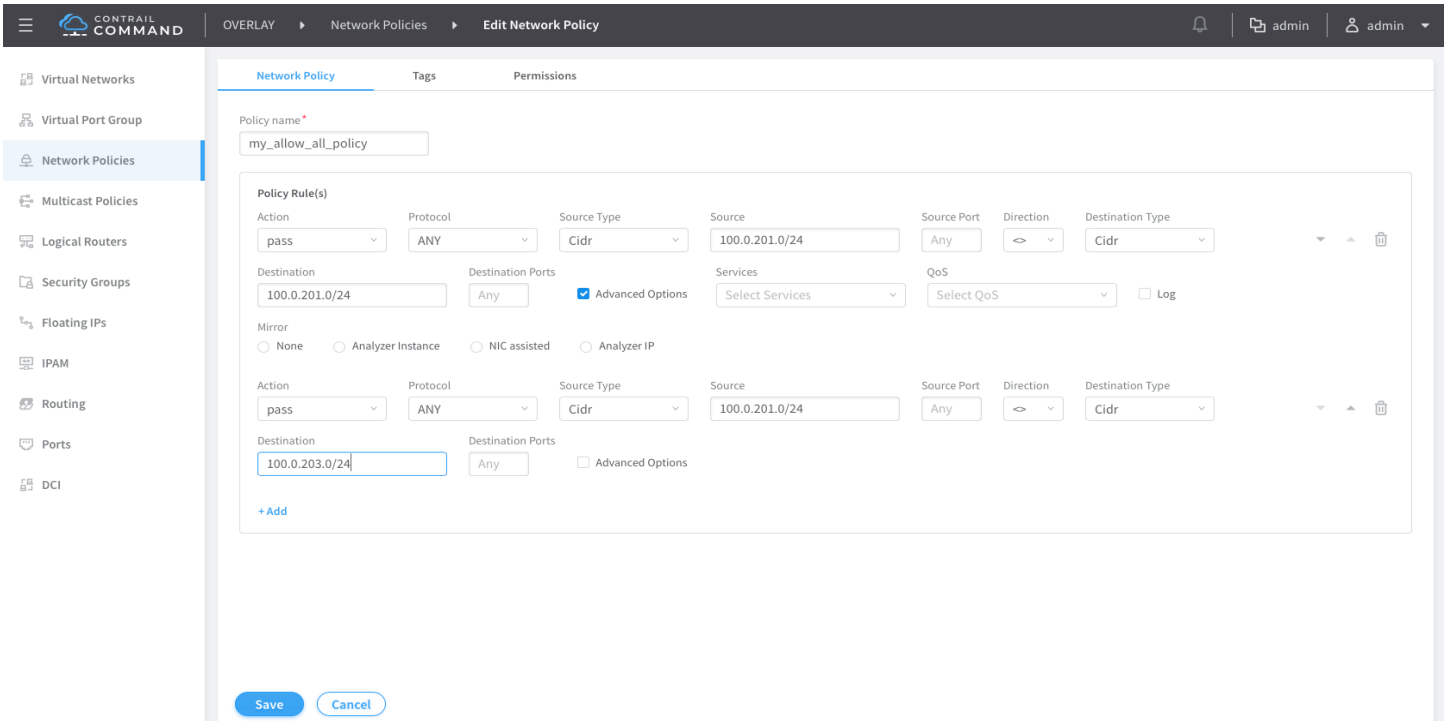
- The verification can be also conducted at the BMS1 and BMS2 server level

```
[root@bms1 ~]# ping 100.0.201.3
PING 100.0.201.3 (100.0.201.3) 56(84) bytes of data.
64 bytes from 100.0.201.3: icmp_seq=1 ttl=64 time=2578 ms
64 bytes from 100.0.201.3: icmp_seq=2 ttl=64 time=1579 ms
64 bytes from 100.0.201.3: icmp_seq=3 ttl=64 time=579 ms
64 bytes from 100.0.201.3: icmp_seq=4 ttl=64 time=303 ms
64 bytes from 100.0.201.3: icmp_seq=5 ttl=64 time=316 ms
64 bytes from 100.0.201.3: icmp_seq=6 ttl=64 time=400 ms
64 bytes from 100.0.201.3: icmp_seq=7 ttl=64 time=261 ms
64 bytes from 100.0.201.3: icmp_seq=8 ttl=64 time=374 ms
^C
--- 100.0.201.3 ping statistics ---
9 packets transmitted, 8 received, 11% packet loss, time 8005ms
rtt min/avg/max/mdev = 261.104/799.367/2578.740/785.053 ms, pipe 3
[root@bms1 ~]#
```

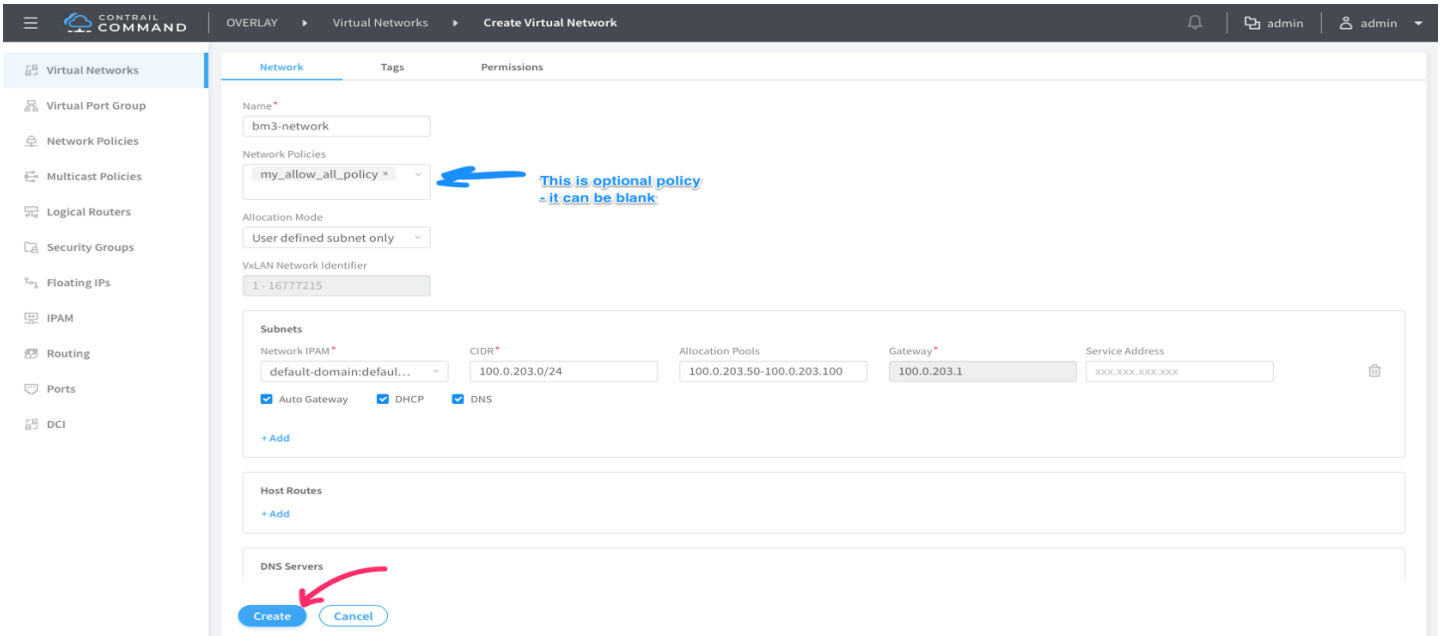
```
[root@bms2 ~]# tcpdump -i ens4 host 100.0.201.2
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
13:28:31.169094 IP pool-100-0-201-2.bstnma.fios.verizon.net > pool-100-0-201-3.bstnma.fios.verizon.net: ICMP echo request, id 3752, seq 1, length 64
13:28:31.169139 IP pool-100-0-201-3.bstnma.fios.verizon.net > pool-100-0-201-2.bstnma.fios.verizon.net: ICMP echo reply, id 3752, seq 1, length 64
13:28:32.278592 IP pool-100-0-201-2.bstnma.fios.verizon.net > pool-100-0-201-3.bstnma.fios.verizon.net: ICMP echo request, id 3752, seq 2, length 64
13:28:32.278617 IP pool-100-0-201-3.bstnma.fios.verizon.net > pool-100-0-201-2.bstnma.fios.verizon.net: ICMP echo reply, id 3752, seq 2, length 64
^C
4 packets captured
4 packets received by filter
0 packets dropped by kernel
[root@bms2 ~]#
```

1.6. Enable the BM3 dedicated new virtual-network and connecting with the existing BMS1 and BMS2 network

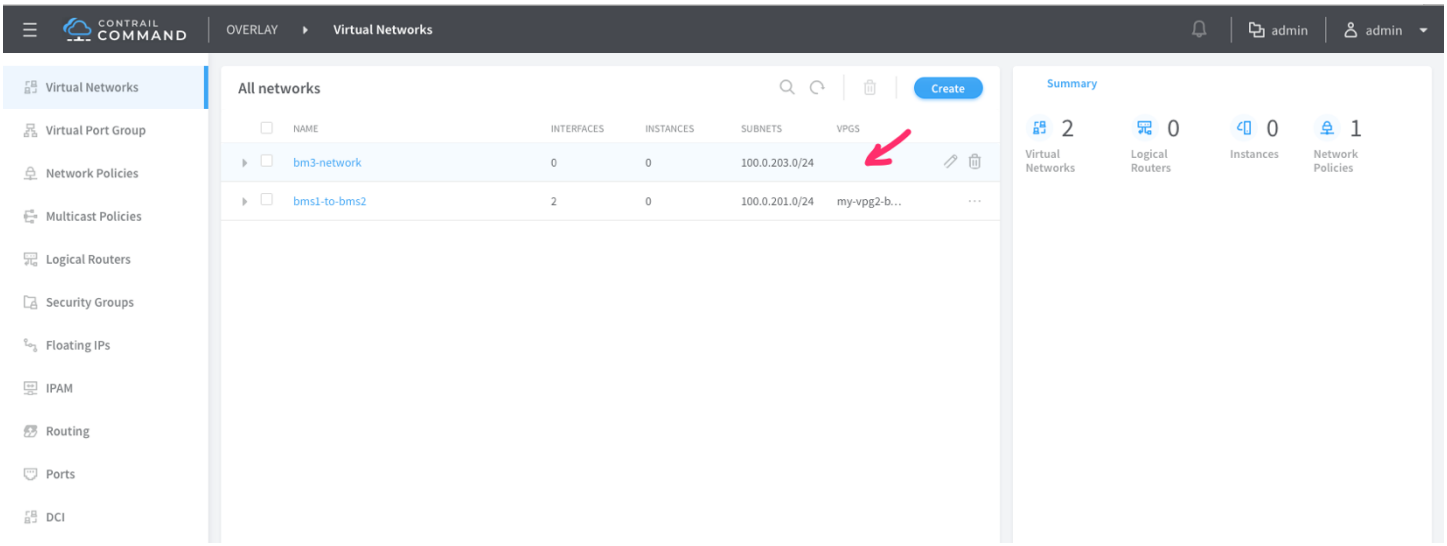
- Before enabling the new virtual-network we need to update the network policy to allow the BMS1/2/3 communication



- Then create the new virtual network dedicated to the BMS3

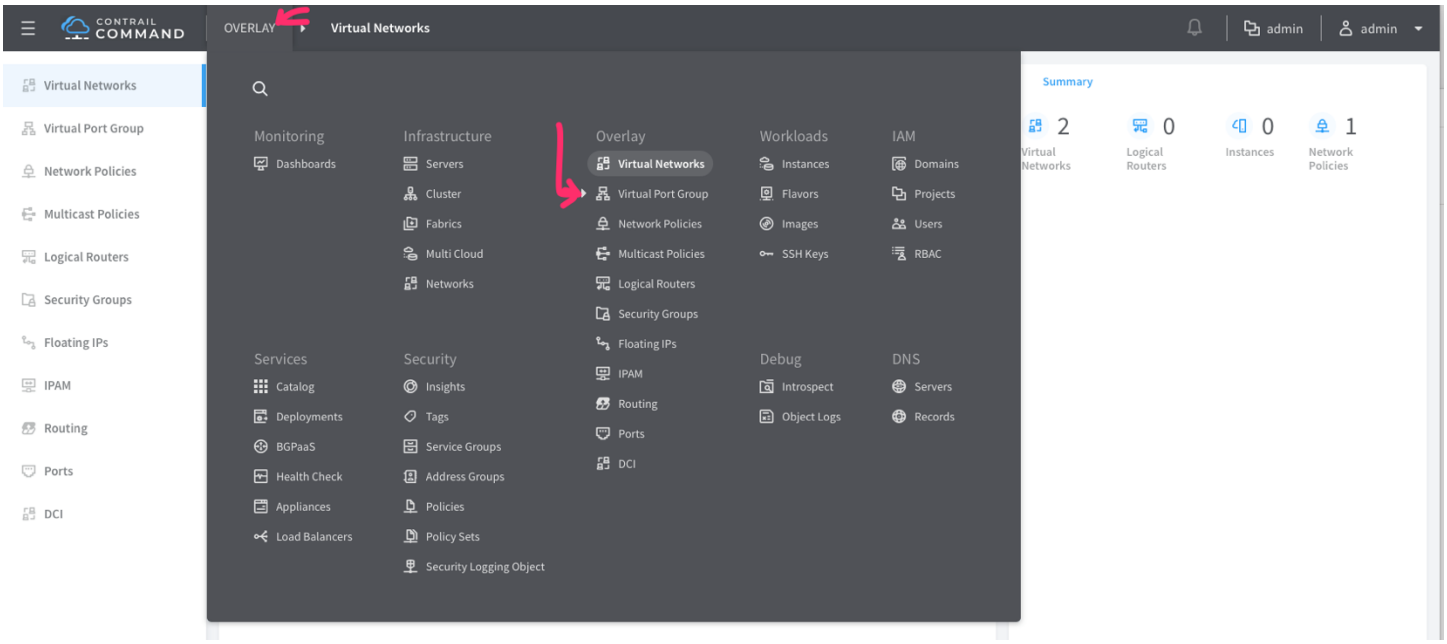


- At that stage we still don't see any VPG ports associated with the given new virtual-network

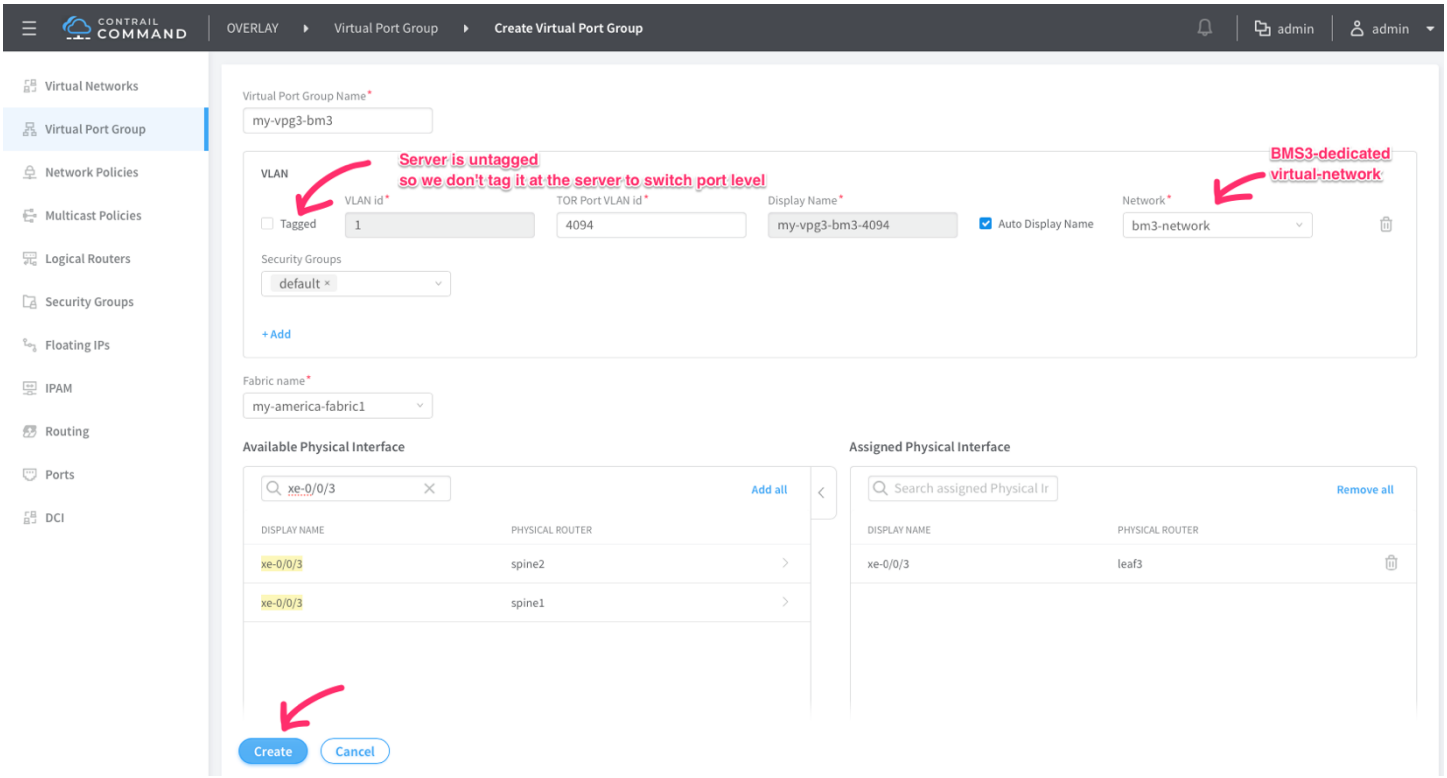


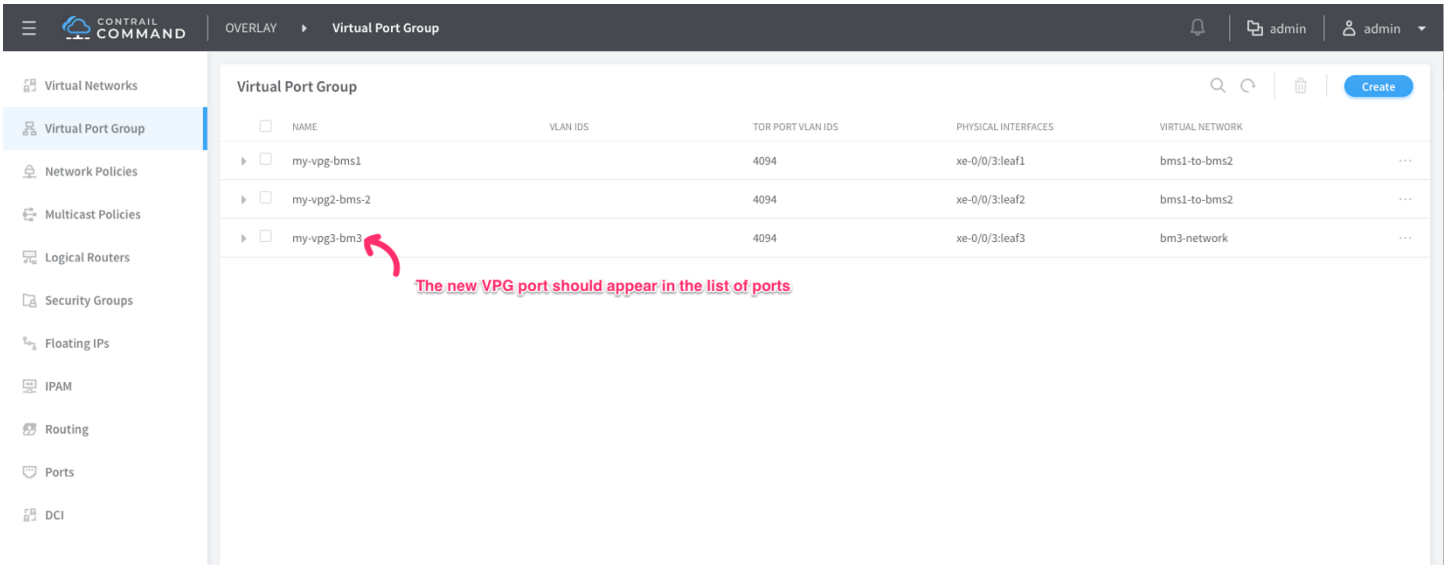
1.7 Creating the third VPG port for BMS3 at leaf 3

- Now we need to proceed on adding the new VPG port for the BMS3

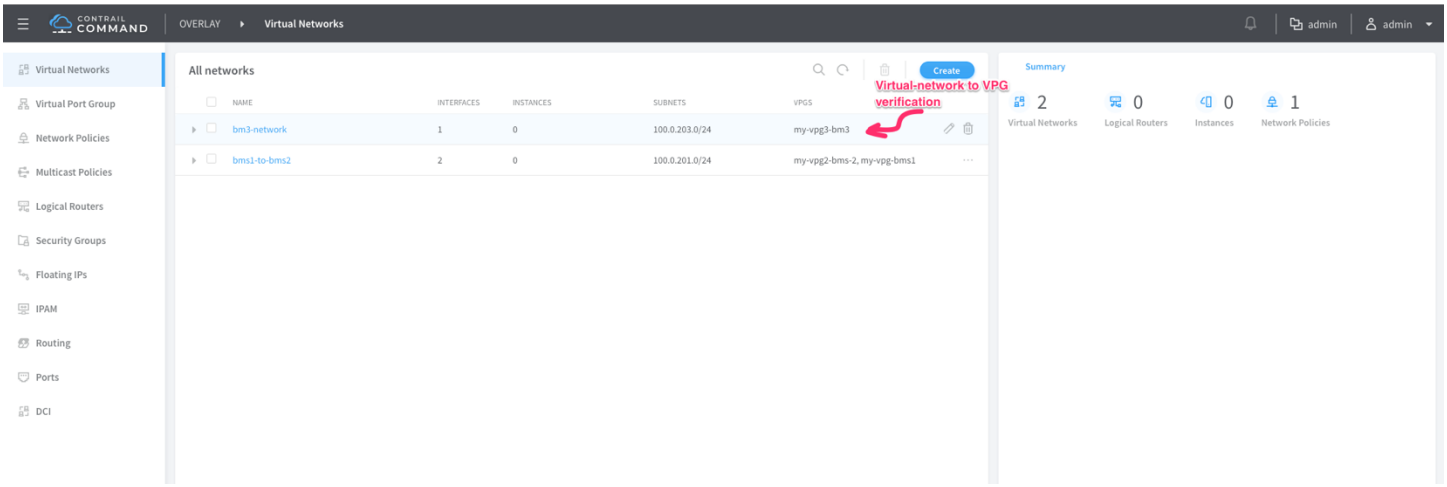


- Associate the BMS3 virtual-network created previously with the physical interface on the given leaf3 node and click “create”





- The given new VPG should show-up now as being associated with the bms3 virtual-network

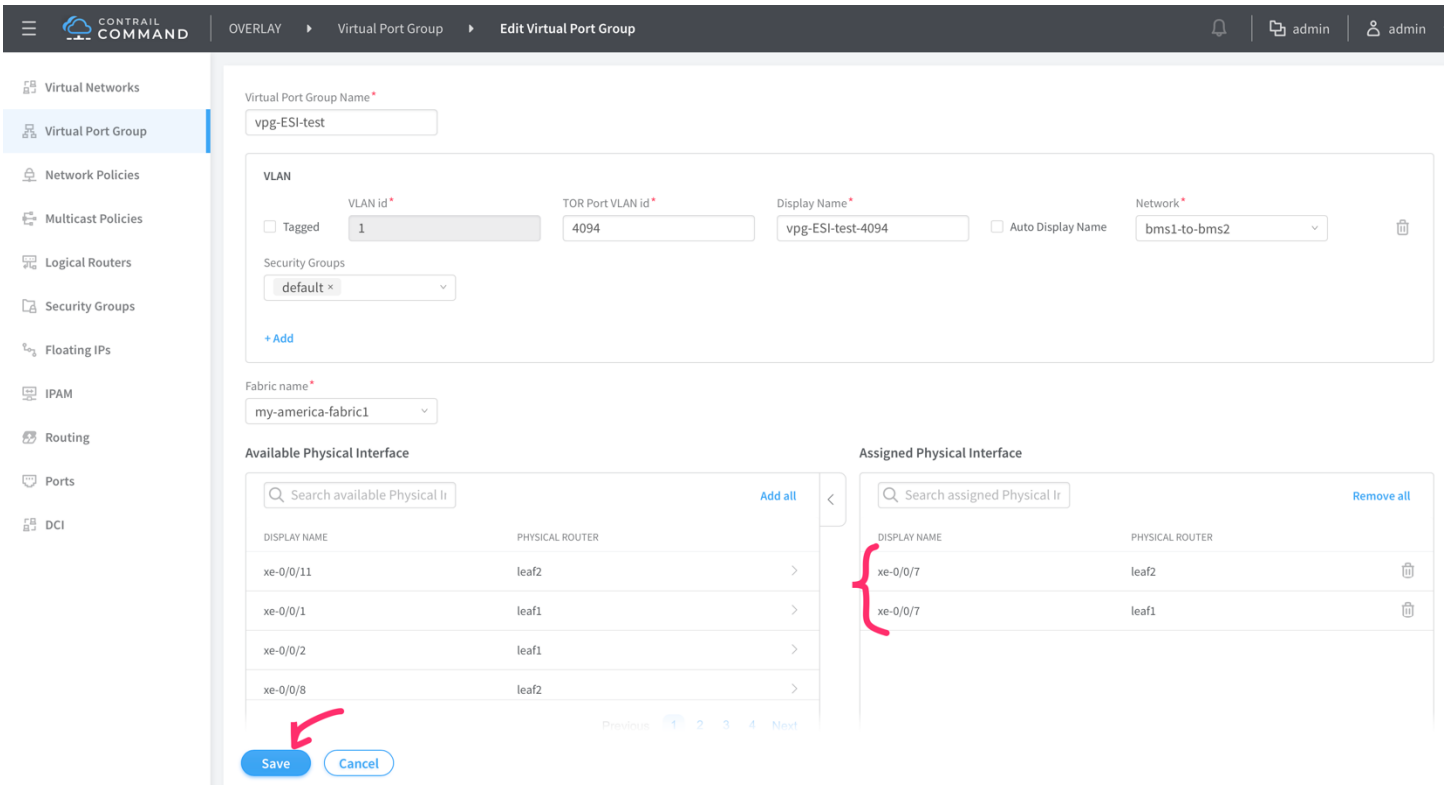



```
root@leaf3> show interfaces irb terse
Interface               Admin Link Proto    Local                 Remote
irb                     up    up
irb.7                   up    up   inet     100.0.203.1/24

{master:0}
root@leaf3> show interfaces irb terse routing-instance all
Interface         Admin Link Proto          Local                 Instance
irb.7             up    up   inet       100.0.203.1/24
                                                                  default

{master:0}
root@leaf3>

{master:0}
root@leaf3> show ethernet-switching interface xe-0/0/3
Routing Instance Name : default-switch
Logical Interface flags (DL - disable learning, AD - packet action drop,
                         LH - MAC limit hit, DN - interface down,
                         MMAS - Mac-move action shutdown,  AS - Autostate-exclude enabled,
                         SCTL - shutdown by Storm-control, MI - MAC+IP limit hit)

Logical         Vlan                 TAG   MAC    MAC+IP STP          Logical          Tagging
interface       members                    limit  limit  state        interface flags
xe-0/0/3.0                                 8192   8192                                 untagged
                bd-7                 4094  1024   1024   Forwarding                    untagged

{master:0}
root@leaf3>
```

At that stage the IRB for BMS3 is not associated with any routing-instance so it's present in the global routing-table until we associate it with a Logical-Router in EM dashboard

A new BD-7 was also generated for the BMS3 virtual-network and pushed to the switch once we associated it with the given VPG port

1.8. Enable the ESI-LAG for BMS-4 on leaf1/2 using the interface xe-0/0/7 - ensure the ESI value is automatically generated

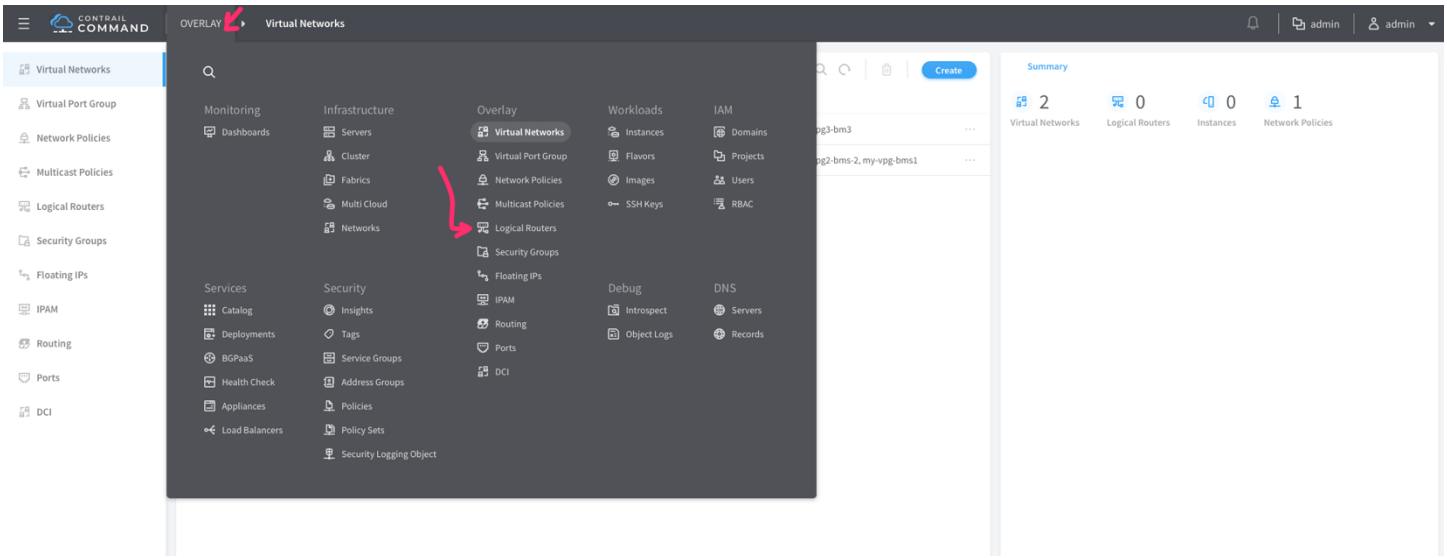


- The following should be the end-result of the VPG multihomed provisioning task

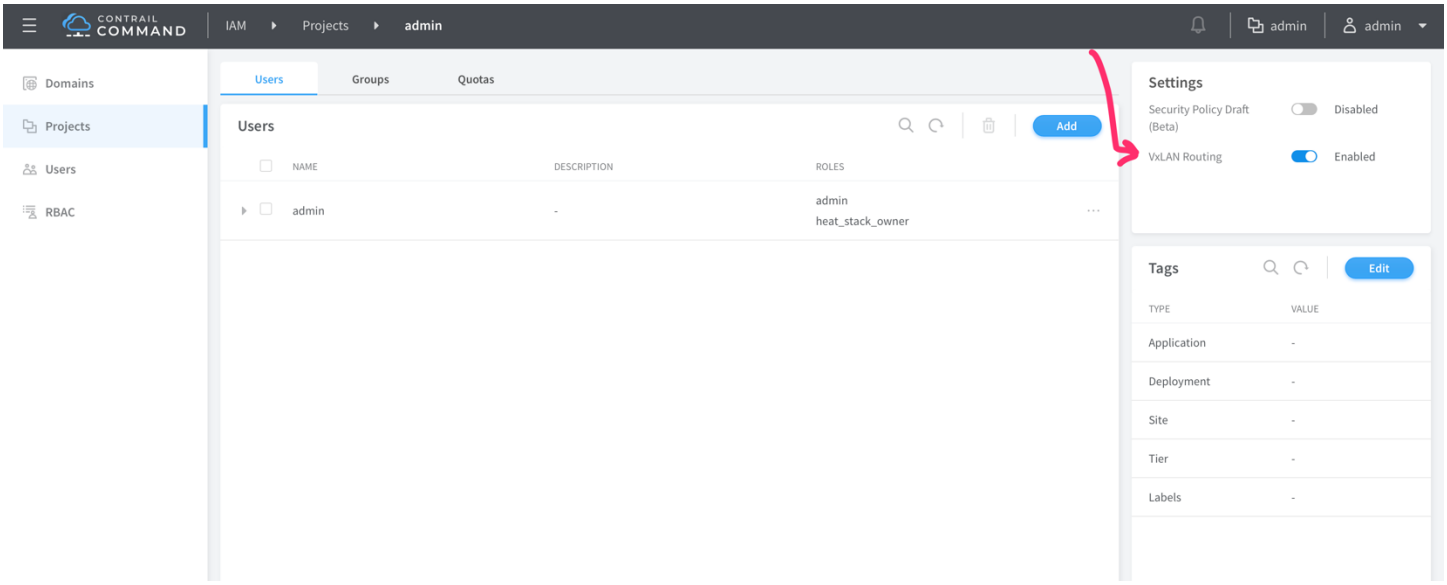
```
root@leaf1> show configuration interfaces xe-0/0/7 |display inheritance |except #
gigether-options {
    802.3ad ae2;
}

{master:0}
root@leaf1> show configuration interfaces ae2 | display inheritance |except #
mtu 9192;
esi {
    00:d4:10:2f:39:da:e6:d6:14:00;        <- Automatically generated ESI
    all-active;
}
aggregated-ether-options {
    lacp {
        active;
        periodic fast;
        system-priority 100;
        system-id 00:41:6d:6e:ad:93;      <- Automatically generated LACP system-id
        admin-key 1;
    }
}
unit 0 {
    family ethernet-switching {
        interface-mode access;
        vlan {
            members bd-6;
        }
    }
}

{master:0}
root@leaf1>
```

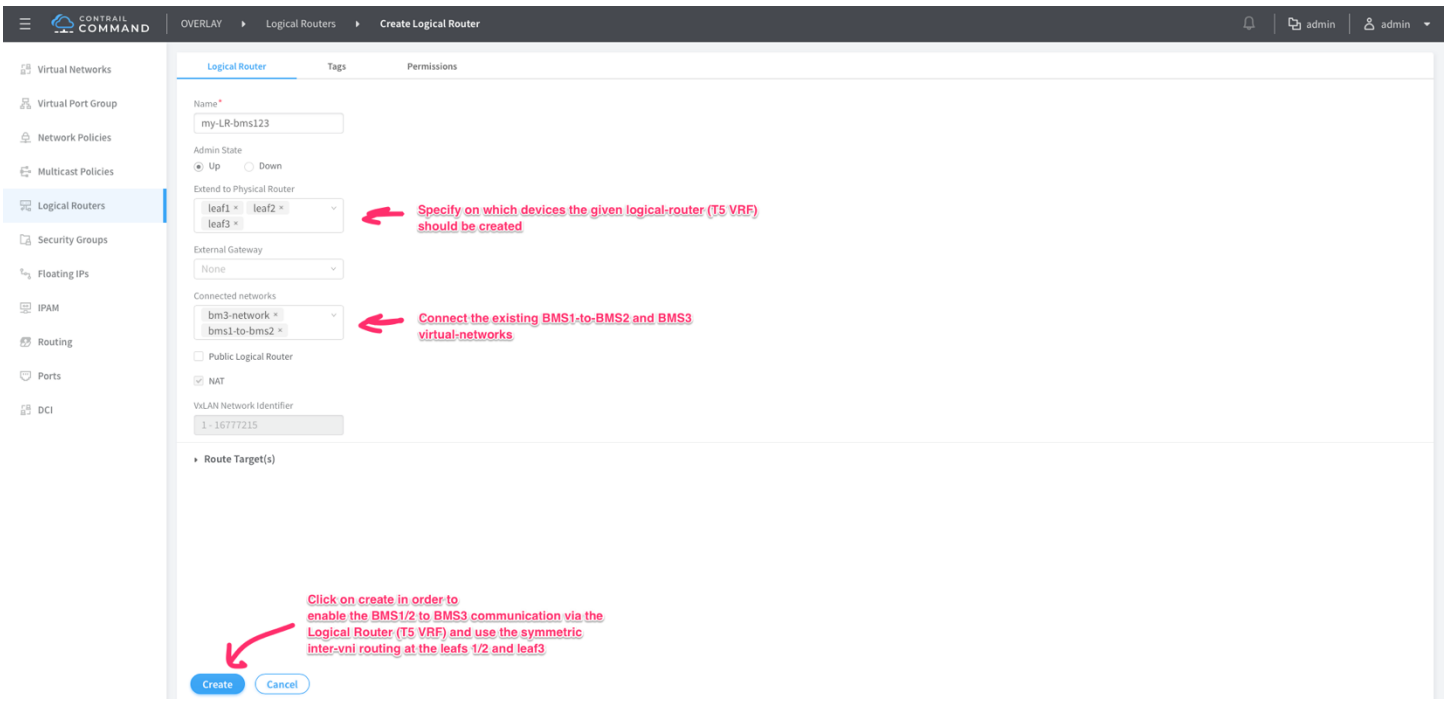
1.9. Enable the BMS1/2 to BMS3 communication using the Logical-Router dashboard in EM 5.1

- In order to make sure the BMS1/2 and BMS3 can communicate between each other we'll enable a new logical router which interconnects the BMS1/2 and BMS3 virtual networks. In the Contrail Command click on the Overlay and then Logical-Router option - this will create the routing-instances type-5 at the leafs1/2/3 which glues both virtual-networks

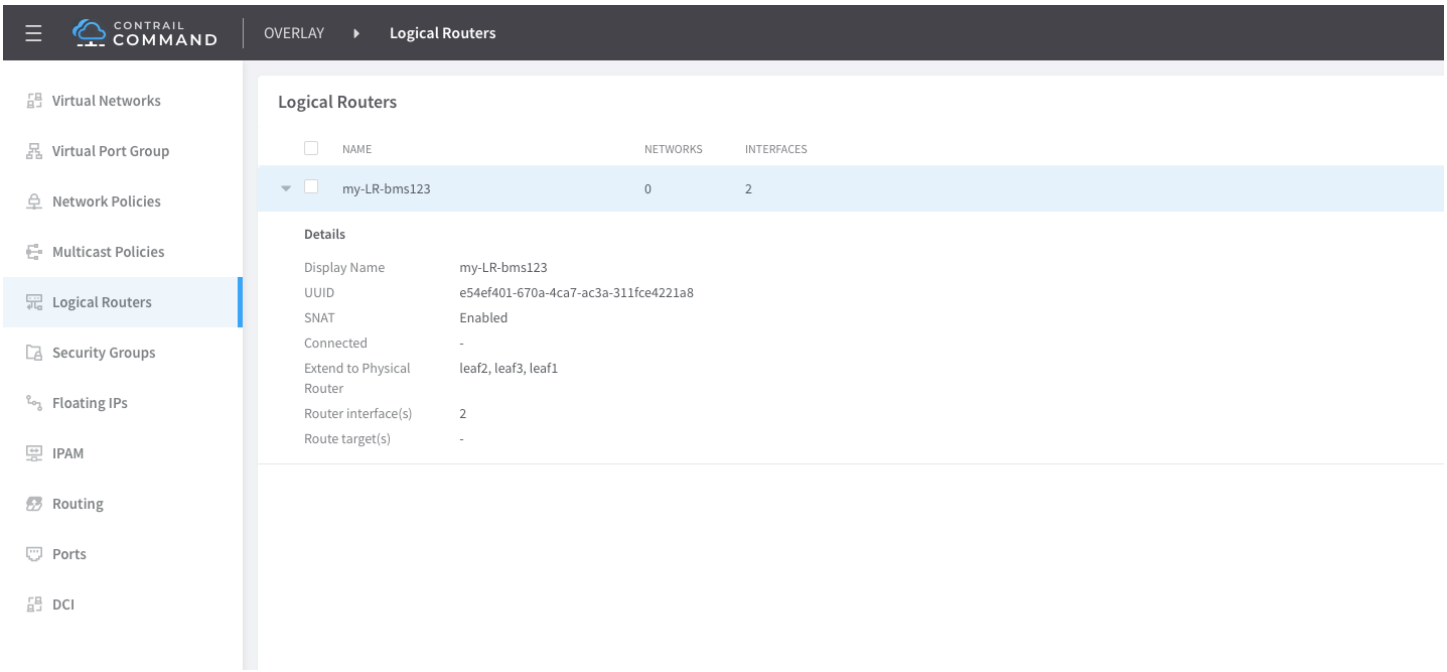


- Associate the existing virtual-networks with the new Logical-Router name and explicitly define on which devices the logical-router should be enabled in order to allow the BMS1/2 and BMS3 symmetric inter-vni communication





- We should see two router interfaces associated with the given new Logical Router - the two interfaces correspond to two IRBs (irb.6 and irb.7) in the given example

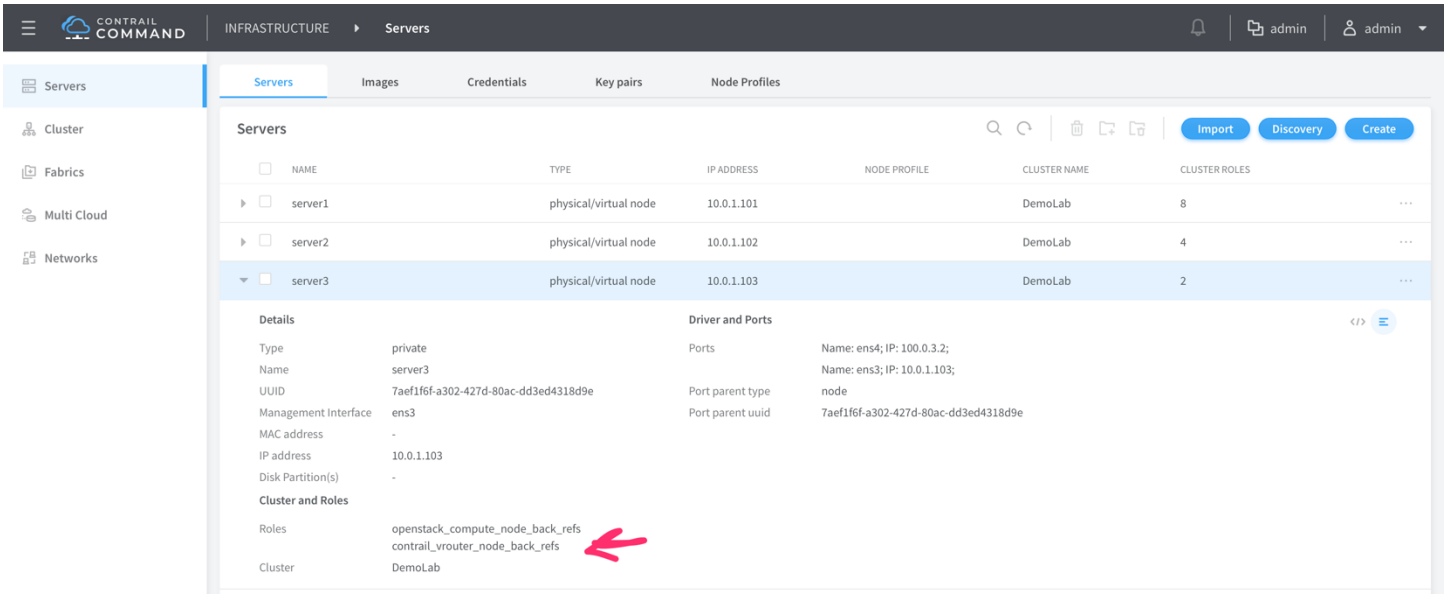


- The following is the end-result of the LR provisioning at the edge routed leaf QFX switch

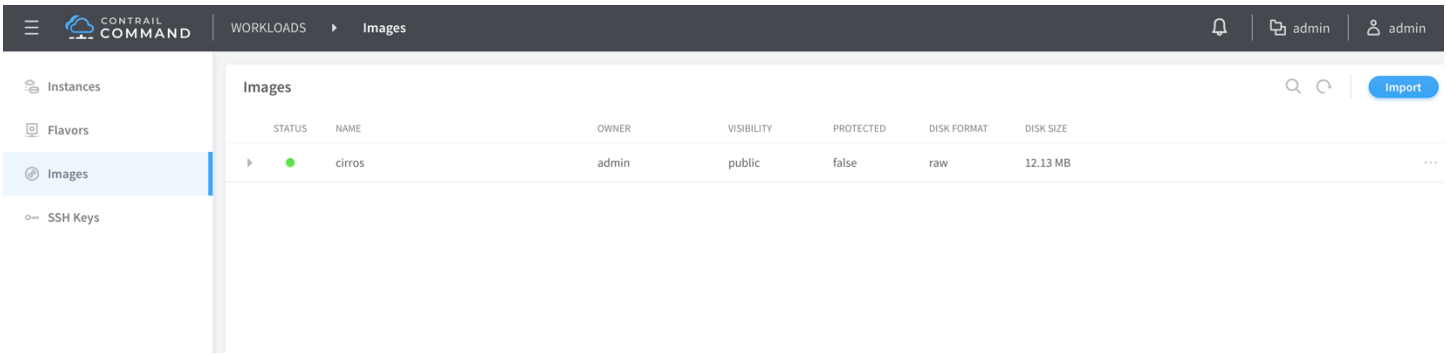
The IRB.6 and IRB.7 from the two different virtual-networks are now under the common logical-router aka routing-instance

```
root@leaf1> show configuration routing-instances |display inheritance |except #
_contrail___contrail_lr_internal_vn_5626ab97-2736-4c01-9e95-7785b33ba4c9__-l3-9 {
    instance-type vrf;
    interface lo0.1009;
    interface irb.7;
    interface irb.6;
    vrf-import _contrail___contrail_lr_internal_vn_5626ab97-2736-4c01-9e95-7785b33ba4c9__-l3-9-import;
    vrf-export _contrail___contrail_lr_internal_vn_5626ab97-2736-4c01-9e95-7785b33ba4c9__-l3-9-export;
    routing-options {
        rib _contrail___contrail_lr_internal_vn_5626ab97-2736-4c01-9e95-7785b33ba4c9__-l3-9.inet6.0 {
            multipath;
        }
        static {
            route 172.16.0.1/32 discard;
        }
        multipath;
    }
    protocols {
        evpn {
            ip-prefix-routes {
                advertise direct-nexthop;
                encapsulation vxlan;
                vni 9;
                export dummy_type5;
            }
        }
    }
}

{master:0}
root@leaf1>
```

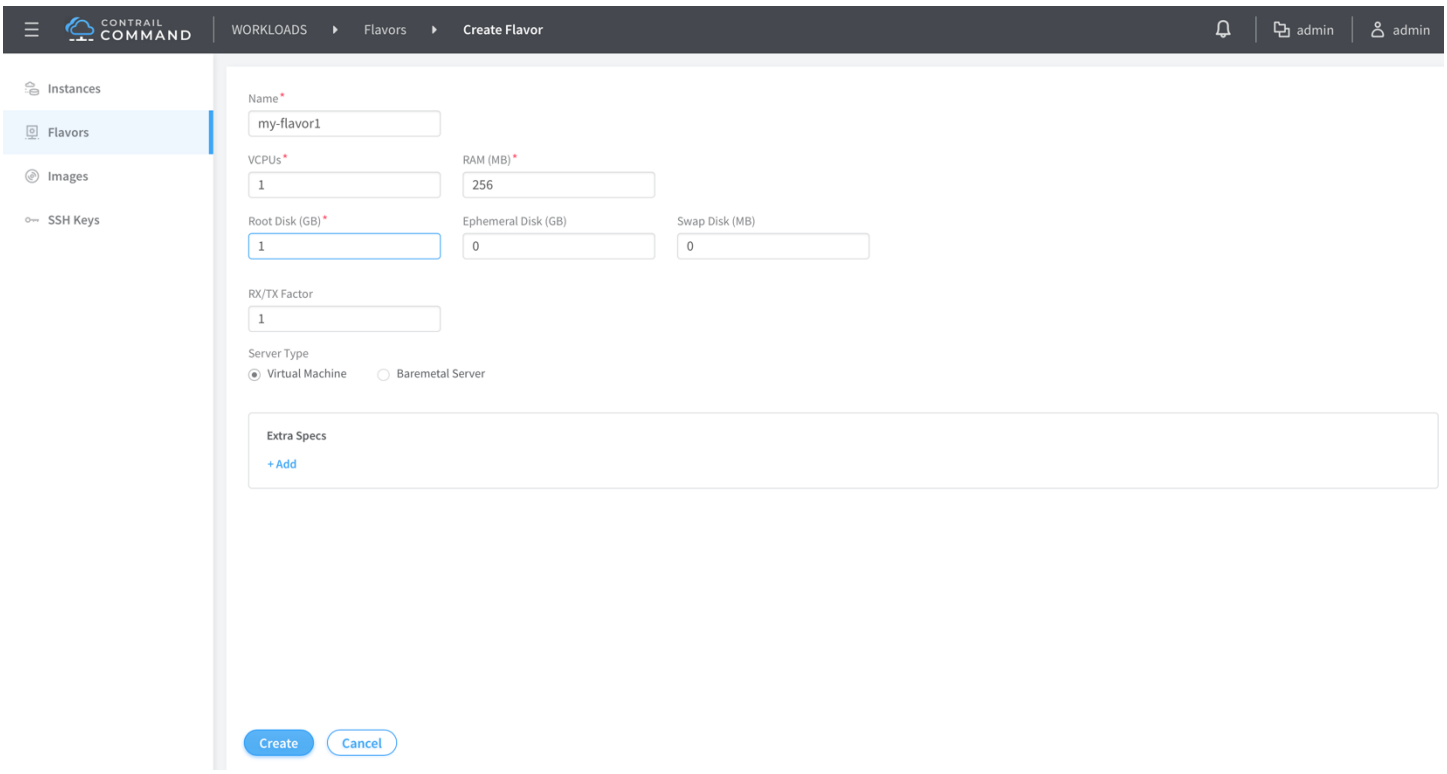
1.10 Verifying compute-node integration - this step is already done so there's no need to integrate it again

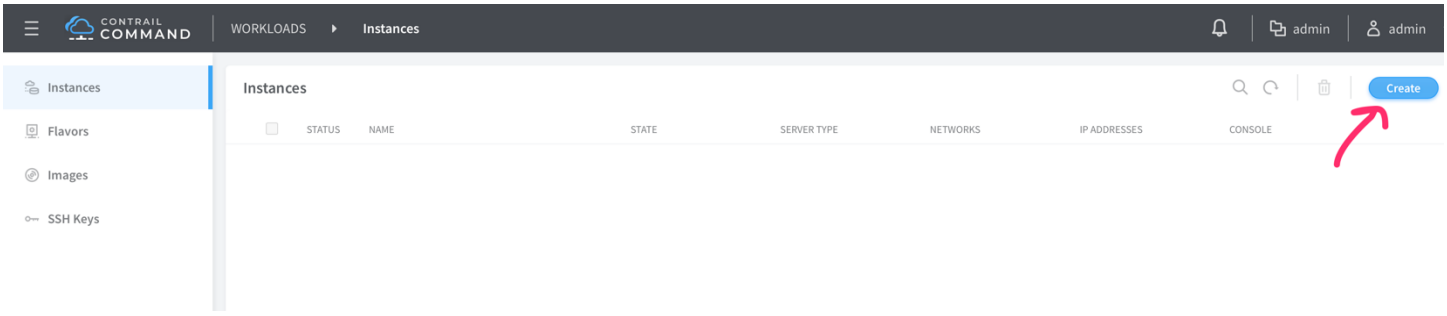


- Loading a cirrus image



In a CirrOS image, the login account is **`cirros`**. The password is **`gocubsgo`**.


CONTRAIL COMMAND
WORKLOADS
Flavors
Create Flavor
admin
admin
Instances
Flavors
Images
SSH Keys
Name*
my-flavor1
VCPUs*
1
RAM (MB)*
256
Root Disk (GB)*
1
Ephemeral Disk (GB)
0
Swap Disk (MB)
0
RX/TX Factor
1
Server Type
Virtual Machine
Baremetal Server
Extra Specs
+ Add
Create
Cancel


CONTRAIL COMMAND
WORKLOADS
Instances
admin
admin
Instances
Flavors
Images
SSH Keys
Instances
Create
STATUS
NAME
STATE
SERVER TYPE
NETWORKS
IP ADDRESSES
CONSOLE

Baseline troubleshooting of the EM 5.1

- Verifying the containers state at the contrail command server6 and contrail control/config server1

[root@server6 ~]# docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d8155e5827ef hub.juniper.net/contrail/contrail-command:5.1.0-0.38 "/bin/contrail -c /e,Ä¶" 4 weeks ago Up 3 hours contrail_command

e709dc7c2509 circleci/postgres:10.3-alpine "docker-entrypoint.s,Ä¶" 4 weeks ago Up 3 hours contrail_psql

systemctl status docker.service

docker logs -f contrail_command_deployer

docker ps

- The following are the containers running at the Contrail controller server 1 - we can check the container status and count in the following way

```
[root@server1 ~]# docker ps  | grep "4 weeks ago"
d47adeb7dde7        hub.juniper.net/contrail-nightly/contrail-analytics-snmp-topology:5.1.0-0.38-queens       "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      analytics_snmp_topology_1
43541a653d40        hub.juniper.net/contrail-nightly/contrail-analytics-snmp-collector:5.1.0-0.38-queens      "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      analytics_snmp_snmp-collector_1
f19d76e88126        hub.juniper.net/contrail-nightly/contrail-nodemgr:5.1.0-0.38-queens                       "/entrypoint.sh /bin,Ä¶"   4 weeks ago         Up 4 hours                      analytics_snmp_nodemgr_1
1729728ce4cc        hub.juniper.net/contrail-nightly/contrail-analytics-alarm-gen:5.1.0-0.38-queens           "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      analytics_alarm_alarm-gen_1
b9f5d772d89d        hub.juniper.net/contrail-nightly/contrail-nodemgr:5.1.0-0.38-queens                       "/entrypoint.sh /bin,Ä¶"   4 weeks ago         Up 4 hours                      analytics_alarm_nodemgr_1
a70a417b4682        hub.juniper.net/contrail-nightly/contrail-external-kafka:5.1.0-0.38-queens               "/docker-entrypoint.,Ä¶"   4 weeks ago         Up 4 hours                      analytics_alarm_kafka_1
59359bfc21aa        hub.juniper.net/contrail-nightly/contrail-analytics-api:5.1.0-0.38-queens                "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      analytics_api_1
5d4cc91582fd        hub.juniper.net/contrail-nightly/contrail-analytics-collector:5.1.0-0.38-queens          "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      analytics_collector_1
054c875ad008        hub.juniper.net/contrail-nightly/contrail-nodemgr:5.1.0-0.38-queens                       "/entrypoint.sh /bin,Ä¶"   4 weeks ago         Up 4 hours                      analytics_nodemgr_1
dfa982b86c7c        hub.juniper.net/contrail-nightly/contrail-analytics-query-engine:5.1.0-0.38-queens       "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      analytics_database_query-engine_1
800e825fb6c2        hub.juniper.net/contrail-nightly/contrail-nodemgr:5.1.0-0.38-queens                       "/entrypoint.sh /bin,Ä¶"   4 weeks ago         Up 4 hours                      analytics_database_nodemgr_1
eb84acebdb8b        hub.juniper.net/contrail-nightly/contrail-external-cassandra:5.1.0-0.38-queens           "/contrail-entrypoin,Ä¶"   4 weeks ago         Up 4 hours                      analytics_database_cassandra_1
e4b9c43e3415        hub.juniper.net/contrail-nightly/contrail-controller-control-named:5.1.0-0.38-queens     "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      control_named_1
70a9425c8d2a        hub.juniper.net/contrail-nightly/contrail-nodemgr:5.1.0-0.38-queens                       "/entrypoint.sh /bin,Ä¶"   4 weeks ago         Up 4 hours                      control_nodemgr_1
3dae2464898e        hub.juniper.net/contrail-nightly/contrail-controller-control-control:5.1.0-0.38-queens   "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      control_control_1
bf42d8863e85        hub.juniper.net/contrail-nightly/contrail-controller-control-dns:5.1.0-0.38-queens       "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      control_dns_1
e795ed755a02        hub.juniper.net/contrail-nightly/contrail-controller-webui-web:5.1.0-0.38-queens         "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      webui_web_1
e20e2983f685        hub.juniper.net/contrail-nightly/contrail-controller-webui-job:5.1.0-0.38-queens         "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      webui_job_1
02643df85766        hub.juniper.net/contrail-nightly/contrail-controller-config-devicemgr:5.1.0-0.38-queens "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      config_devicemgr_1
e40fb13c210f        hub.juniper.net/contrail-nightly/contrail-controller-config-svcmonitor:5.1.0-0.38-queens "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      config_svcmonitor_1
ae2885035776        hub.juniper.net/contrail-nightly/contrail-nodemgr:5.1.0-0.38-queens                       "/entrypoint.sh /bin,Ä¶"   4 weeks ago         Up 4 hours                      config_nodemgr_1
44802203f32d        hub.juniper.net/contrail-nightly/contrail-controller-config-api:5.1.0-0.38-queens        "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      config_api_1
646b61c983c9        hub.juniper.net/contrail-nightly/contrail-controller-config-schema:5.1.0-0.38-queens     "/entrypoint.sh /usr,Ä¶"   4 weeks ago         Up 4 hours                      config_schema_1
a774c2cc10c0        hub.juniper.net/contrail-nightly/contrail-external-dnsmasq:5.1.0-0.38-queens             "/entrypoint.sh dnsm,Ä¶"   4 weeks ago         Up 4 hours                      config_dnsmasq_1
a921a0509176        hub.juniper.net/contrail-nightly/contrail-nodemgr:5.1.0-0.38-queens                       "/entrypoint.sh /bin,Ä¶"   4 weeks ago         Up 4 hours                      config_database_nodemgr_1
1c2738fea9c4        hub.juniper.net/contrail-nightly/contrail-external-zookeeper:5.1.0-0.38-queens           "/contrail-entrypoin,Ä¶"   4 weeks ago         Up 4 hours                      config_database_zookeeper_1
98ca0a3dcb4e        hub.juniper.net/contrail-nightly/contrail-external-rabbitmq:5.1.0-0.38-queens            "/contrail-entrypoin,Ä¶"   4 weeks ago         Up 4 hours                      config_database_rabbitmq_1
36863ca8d87c        hub.juniper.net/contrail-nightly/contrail-external-cassandra:5.1.0-0.38-queens           "/contrail-entrypoin,Ä¶"   4 weeks ago         Up 4 hours                      config_database_cassandra_1
cda8debe5ba1        hub.juniper.net/contrail-nightly/contrail-external-redis:5.1.0-0.38-queens               "/contrail-entrypoin,Ä¶"   4 weeks ago         Up 4 hours                      redis_redis_1
[root@server1 ~]#
[root@server1 ~]#
[root@server1 ~]# docker ps  | grep "4 weeks ago" | wc -l
29
[root@server1 ~]#
```

- checking the logs related to the ansible execution at the config API server1

[root@server1 ~]#

[root@server1 ~]# tailf /var/log/contrail/contrail-fabric-ansible-playbooks.log

07/03/2019 14:01:33.403 [ansible] pid=6477 [WARNING]: TASK [commit_fabric_config : Process additional error handler, if present] *****

07/03/2019 14:01:33.430 [ansible] pid=6477 [WARNING]: skipping: [localhost]

07/03/2019 14:01:33.442 [ansible] pid=6477 [WARNING]: TASK [commit_fabric_config : fail] **********

07/03/2019 14:01:33.472 [ansible] pid=6477 [WARNING]: skipping: [localhost]

07/03/2019 14:01:33.485 [ansible] pid=6477 [WARNING]: PLAY RECAP
****

07/03/2019 14:01:33.485 [ansible] pid=6477 [WARNING]: localhost :
ok=8 changed=0 unreachable=0 failed=1

07/03/2019 14:01:33.485 [ansible] pid=6477 [WARNING]: localhost :
ok=8 changed=0 unreachable=0 failed=1

07/03/2019 14:01:33.486 [ansible] pid=6477 [WARNING]:

- Before starting any VM at the compute nodes we need to make sure the local compute-node server /etc/hosts file by adding the local compute-node in-band IP addressing and the name - repeat the same at server3/server4 and CSN server5

[root@server3 ~]#

[root@server3 ~]# vi /etc/hosts

127.0.0.1 localhost localhost.localdomain localhost4 localhost4.localdomain4

10.0.1.101 server1.local server1

::1 localhost localhost.localdomain localhost6 localhost6.localdomain6

10.0.1.104 server4.local server4

10.0.1.105 server5.local server5

10.0.1.102 server2.local server2

10.0.1.103 server3.local server3

100.0.3.2 server3 server3 ← that part needs to correspond to your local server IP@ fabric in-band connectivity

[root@server3 ~]# docker-compose down

[root@server3 ~]# docker-compose up -d

- Checking the compute-node aka vrouter contrail state

```
[root@server3 ~]# contrail-status
Pod      Service  Original Name           State    Id            Status
vrouter  agent    contrail-vrouter-agent  running  ecfa89041623  Up 2 hours
vrouter  nodemgr  contrail-nodemgr        running  b0c8641a06c1  Up 2 hours

vrouter kernel module is PRESENT
== Contrail vrouter ==
nodemgr: active
agent: active

[root@server3 ~]# _
```

- In order to debug the VM creation we can observe the following log at the compute-node

[root@server3 ~]# **tail -f /var/lib/docker/volumes/kolla_logs/_data/nova/nova-compute.log**

2019-07-09 15:59:36.936 5 INFO nova.compute.resource_tracker [req-5a3e55d5-9bee-4cfe-81e9-a039245cc21c - - - - -] Final resource view: name=server3.local phys_ram=49151MB used_ram=512MB phys_disk=499GB used_disk=0GB total_vcpus=2 used_vcpus=0 pci_stats=[]

2019-07-09 16:00:38.940 5 INFO nova.compute.resource_tracker [req-5a3e55d5-9bee-4cfe-81e9-a039245cc21c - - - - -] Final resource view: name=server3.local phys_ram=49151MB used_ram=512MB phys_disk=499GB used_disk=0GB total_vcpus=2 used_vcpus=0 pci_stats=[]

2019-07-09 16:01:41.010 5 INFO nova.compute.resource_tracker [req-5a3e55d5-9bee-4cfe-81e9-a039245cc21c - - - - -] Final resource view: name=server3.local phys_ram=49151MB used_ram=512MB phys_disk=499GB used_disk=0GB total_vcpus=2 used_vcpus=0 pci_stats=[]

2019-07-09 16:02:40.955 5 INFO nova.compute.resource_tracker [req-5a3e55d5-9bee-4cfe-81e9-a039245cc21c - - - - -] Final resource view: name=server3.local phys_ram=49151MB used_ram=512MB phys_disk=499GB used_disk=0GB total_vcpus=2 used_vcpus=0 pci_stats=[]

2019-07-09 16:03:43.000 5 INFO nova.compute.resource_tracker [req-5a3e55d5-9bee-4cfe-81e9-a039245cc21c - - - - -] Final resource view: name=server3.local phys_ram=49151MB used_ram=512MB phys_disk=499GB used_disk=0GB total_vcpus=2 used_vcpus=0 pci_stats=[]

2019-07-09 16:07:43.947 5 INFO nova.compute.resource_tracker [req-5a3e55d5-9bee-4cfe-81e9-a039245cc21c - - - - -] Final resource view: name=server3.local phys_ram=49151MB used_ram=512MB phys_disk=499GB used_disk=0GB total_vcpus=2 used_vcpus=0 pci_stats=[]

[root@server3 ~]#

- Verifying the compute-node ip addresses

```
[root@server3 ~]# ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 2c:c2:60:2a:4f:94 brd ff:ff:ff:ff:ff:ff
    inet 10.0.1.103/16 brd 10.0.255.255 scope global ens3
       valid_lft forever preferred_lft forever
    inet6 fe80::2ec2:60ff:fe2a:4f94/64 scope link
       valid_lft forever preferred_lft forever
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 2c:c2:60:3d:29:29 brd ff:ff:ff:ff:ff:ff
4: pkt1: <UP,LOWER_UP> mtu 65535 qdisc noqueue state UNKNOWN group default qlen 1000
    link/void 4a:04:cf:98:91:35 brd 00:00:00:00:00:00
5: pkt3: <UP,LOWER_UP> mtu 65535 qdisc noqueue state UNKNOWN group default qlen 1000
    link/void ce:11:bf:42:5a:32 brd 00:00:00:00:00:00
6: pkt2: <UP,LOWER_UP> mtu 65535 qdisc noqueue state UNKNOWN group default qlen 1000
    link/void d2:f7:ad:50:17:d4 brd 00:00:00:00:00:00
7: vhost0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 1000
    link/ether 2c:c2:60:3d:29:29 brd ff:ff:ff:ff:ff:ff
    inet 100.0.3.2/24 brd 100.0.3.255 scope global vhost0
       valid_lft forever preferred_lft forever
    inet6 fe80::2ec2:60ff:fe3d:2929/64 scope link
       valid_lft forever preferred_lft forever
8: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:83:87:b6:56 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
       valid_lft forever preferred_lft forever
10: pkt0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 1000
    link/ether 66:d9:63:53:51:8b brd ff:ff:ff:ff:ff:ff
    inet6 fe80::64d9:63ff:fe53:518b/64 scope link
       valid_lft forever preferred_lft forever
[root@server3 ~]#
```

- Verifying the state of the compute-node interfaces

```
[root@server3 ~]# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 2c:c2:60:2a:4f:94 brd ff:ff:ff:ff:ff:ff
3: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether 2c:c2:60:3d:29:29 brd ff:ff:ff:ff:ff:ff
4: pkt1: <UP,LOWER_UP> mtu 65535 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/void 4a:04:cf:98:91:35 brd 00:00:00:00:00:00
5: pkt3: <UP,LOWER_UP> mtu 65535 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/void ce:11:bf:42:5a:32 brd 00:00:00:00:00:00
6: pkt2: <UP,LOWER_UP> mtu 65535 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/void d2:f7:ad:50:17:d4 brd 00:00:00:00:00:00
7: vhost0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 2c:c2:60:3d:29:29 brd ff:ff:ff:ff:ff:ff
8: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode DEFAULT group default
    link/ether 02:42:83:87:b6:56 brd ff:ff:ff:ff:ff:ff
10: pkt0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN mode DEFAULT group default qlen 1000
    link/ether 66:d9:63:53:51:8b brd ff:ff:ff:ff:ff:ff
[root@server3 ~]#
```

- Restarting the local compute-node vrouter agent

[root@server4 ~]#
[root@server4 ~]# docker restart vrouter_vrouter-agent_1

- Tcpdump the vxlan tunnel from the vrouter agent

[root@server4 ~]# docker exec -it vrouter_vrouter-agent_1 bash

(vrouter-agent)[root@server4 /]$ tcpdump -i ens4 -n -v udp port 4789 -e

```
(vrouter-agent)[root@server4 /]$ tcpdump -i ens4 -n -v udp port 4789 -e
tcpdump: listening on ens4, link-type EN10MB (Ethernet), capture size 262144 bytes
16:23:54.767969 02:05:86:71:5c:13 > 2c:c2:60:1d:8a:86, ethertype IPv4 (0x0800), length 96: (tos 0x0, ttl 64, id 0, offset 0, flags [none], proto UDP (17), length 82)
    1.1.1.11.26958 > 100.0.4.2.4789: VXLAN, flags [I] (0x08), vni 7
00:00:5e:01:00:01 > 01:00:5e:00:00:01, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 33655, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.203.1 > 224.0.0.1: igmp query v2
16:23:56.378787 02:05:86:71:5c:13 > 2c:c2:60:1d:8a:86, ethertype IPv4 (0x0800), length 96: (tos 0x0, ttl 62, id 0, offset 0, flags [none], proto UDP (17), length 82)
    1.1.1.12.27073 > 100.0.4.2.4789: VXLAN, flags [I] (0x08), vni 7
00:00:5e:01:00:01 > 01:00:5e:00:00:16, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 9094, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.203.1 > 224.0.0.22: igmp v2 report 224.0.0.22
16:23:56.870122 02:05:86:71:5c:13 > 2c:c2:60:1d:8a:86, ethertype IPv4 (0x0800), length 96: (tos 0x0, ttl 62, id 0, offset 0, flags [none], proto UDP (17), length 82)
    1.1.1.13.12508 > 100.0.4.2.4789: VXLAN, flags [I] (0x08), vni 7
00:00:5e:01:00:01 > 01:00:5e:00:00:02, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 3493, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.203.52 > 224.0.0.2: igmp v2 report 224.0.0.2
16:24:08.420742 02:05:86:71:5c:13 > 2c:c2:60:1d:8a:86, ethertype IPv4 (0x0800), length 96: (tos 0x0, ttl 62, id 0, offset 0, flags [none], proto UDP (17), length 82)
    1.1.1.12.26958 > 100.0.4.2.4789: VXLAN, flags [I] (0x08), vni 7
00:00:5e:01:00:01 > 01:00:5e:00:00:01, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 10336, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.203.1 > 224.0.0.1: igmp query v2
16:24:09.322961 02:05:86:71:5c:13 > 2c:c2:60:1d:8a:86, ethertype IPv4 (0x0800), length 96: (tos 0x0, ttl 62, id 0, offset 0, flags [none], proto UDP (17), length 82)
    1.1.1.13.24276 > 100.0.4.2.4789: VXLAN, flags [I] (0x08), vni 7
00:00:5e:01:00:01 > 01:00:5e:00:00:16, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 4763, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.203.52 > 224.0.0.22: igmp v2 report 224.0.0.22
16:24:10.123877 02:05:86:71:5c:13 > 2c:c2:60:1d:8a:86, ethertype IPv4 (0x0800), length 96: (tos 0x0, ttl 62, id 0, offset 0, flags [none], proto UDP (17), length 82)
    1.1.1.13.12508 > 100.0.4.2.4789: VXLAN, flags [I] (0x08), vni 7
00:00:5e:01:00:01 > 01:00:5e:00:00:02, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 4895, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.203.52 > 224.0.0.2: igmp v2 report 224.0.0.2
^C
6 packets captured
6 packets received by filter
0 packets dropped by kernel
(vrouter-agent)[root@server4 /]$
```

- Checking the drop stats at the router agent of the compute node

```
(vrouter-agent)[root@server4 /]$ dropstats | grep -v ' 0$'


Discards                        907
Cloned Original                 1336

Invalid NH                      2


Invalid VNID                    247
Invalid Source                  317
No L2 Route                     2

(vrouter-agent)[root@server4 /]$
(vrouter-agent)[root@server4 /]$
(vrouter-agent)[root@server4 /]$ dropstats | grep -v ' 0$'


Discards                        913
Cloned Original                 1342

Invalid NH                      2


Invalid VNID                    247
Invalid Source                  317
No L2 Route                     2

(vrouter-agent)[root@server4 /]$
```

- List the VMs (aka instance) tap interfaces on the compute-node

```
(vrouter-agent)[root@server4 /]$ vif --list | grep tap
vif0/3      OS: tap129d2c06-71
vif0/4      OS: tapa47e5af3-48
(vrouter-agent)[root@server4 /]$  tcpdump -i tap129d2c06-71 -n -v -e
tcpdump: listening on tap129d2c06-71, link-type EN10MB (Ethernet), capture size 262144 bytes
16:31:07.464802 00:00:5e:00:01:00 > 02:12:9d:2c:06:71, ethertype ARP (0x0806), length 42: Ethernet (len 6), IPv4 (len 4), Request who-has 100.0.201.42 tell 100.0.201.2, length 28
16:31:07.465518 02:12:9d:2c:06:71 > 00:00:5e:00:01:00, ethertype ARP (0x0806), length 42: Ethernet (len 6), IPv4 (len 4), Reply 100.0.201.42 is-at 02:12:9d:2c:06:71, length 28
16:31:09.205657 00:00:5e:01:00:01 > 01:00:5e:00:00:01, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 48942, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.201.1 > 224.0.0.1: igmp query v2
16:31:09.933237 00:00:5e:01:00:01 > 01:00:5e:00:00:16, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 19285, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.201.41 > 224.0.0.22: igmp v2 report 224.0.0.22
16:31:11.886948 00:00:5e:01:00:01 > 01:00:5e:00:00:02, ethertype IPv4 (0x0800), length 46: (tos 0xc0, ttl 1, id 19518, offset 0, flags [none], proto IGMP (2), length 32, options (RA))
    100.0.201.41 > 224.0.0.2: igmp v2 report 224.0.0.2
^C
5 packets captured
9 packets received by filter
0 packets dropped by kernel
(vrouter-agent)[root@server4 /]$
```

```
(vrouter-agent)[root@server4 /]$ vif --list | grep tap
vif0/3      OS: tap129d2c06-71
vif0/4      OS: tapa47e5af3-48
(vrouter-agent)[root@server4 /]$ vif --get 3
Vrouter Interface Table

Flags: P=Policy, X=Cross Connect, S=Service Chain, Mr=Receive Mirror
       Mt=Transmit Mirror, Tc=Transmit Checksum Offload, L3=Layer 3, L2=Layer 2
       D=DHCP, Vp=Vhost Physical, Pr=Promiscuous, Vnt=Native Vlan Tagged
       Mnp=No MAC Proxy, Dpdk=DPDK PMD Interface, Rfl=Receive Filtering Offload, Mon=Interface is Monitored
       Uuf=Unknown Unicast Flood, Vof=VLAN insert/strip offload, Df=Drop New Flows, L=MAC Learning Enabled
       Proxy=MAC Requests Proxied Always, Er=Etree Root, Mn=Mirror without Vlan Tag, Ig=Igmp Trap Enabled

vif0/3      OS: tap129d2c06-71
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:100.0.201.42
            Vrf:4 Mcast Vrf:4 Flags:PL3L2DEr QOS:-1 Ref:7
            RX packets:934  bytes:45734 errors:0
            TX packets:1287  bytes:73831 errors:0
            ISID: 0 Bmac: 02:12:9d:2c:06:71
            Drops:975

(vrouter-agent)[root@server4 /]$ vif --get 4
Vrouter Interface Table

Flags: P=Policy, X=Cross Connect, S=Service Chain, Mr=Receive Mirror
       Mt=Transmit Mirror, Tc=Transmit Checksum Offload, L3=Layer 3, L2=Layer 2
       D=DHCP, Vp=Vhost Physical, Pr=Promiscuous, Vnt=Native Vlan Tagged
       Mnp=No MAC Proxy, Dpdk=DPDK PMD Interface, Rfl=Receive Filtering Offload, Mon=Interface is Monitored
       Uuf=Unknown Unicast Flood, Vof=VLAN insert/strip offload, Df=Drop New Flows, L=MAC Learning Enabled
       Proxy=MAC Requests Proxied Always, Er=Etree Root, Mn=Mirror without Vlan Tag, Ig=Igmp Trap Enabled

vif0/4      OS: tapa47e5af3-48
            Type:Virtual HWaddr:00:00:5e:00:01:00 IPaddr:100.0.201.43
            Vrf:4 Mcast Vrf:4 Flags:PL3L2DEr QOS:-1 Ref:7
            RX packets:1232  bytes:88738 errors:0
            TX packets:1817  bytes:113790 errors:0
            ISID: 0 Bmac: 02:a4:7e:5a:f3:48
            Drops:182

(vrouter-agent)[root@server4 /]$
```

- Dump the local vrouter on the compute-node server bridging table

```
(vrouter-agent)[root@server4 /]$ rt --dump 2 --family bridge
Flags: L=Label Valid, Df=DHCP flood, Mm=Mac Moved, L2c=L2 Evpn Control Word, N=New Entry, Ec=EvpnControlProcessing
vRouter bridge table 0/2
Index       DestMac                   Flags          Label/VNID      Nexthop          Stats
30052       2c:c2:60:1d:8a:86           Df                   -            3              0
31264       0:0:5e:0:1:0                Df                   -            3              0
54652       2c:c2:60:33:c4:a2         LDf                    7           53              0
112924      ff:ff:ff:ff:ff:ff         LDf                    7           51            349
115240      2:0:0:0:0:2                 Df                   -           12              0
205564      2:0:0:0:0:1                 Df                   -           12              0
243740      0:0:5e:1:0:1              LDf                    7           58              0
(vrouter-agent)[root@server4 /]$
```

```
(vrouter-agent)[root@server4 /]$ nh --get 53
Id:53         Type:Tunnel         Fmly: AF_INET  Rid:0  Ref_cnt:5          Vrf:0
              Flags:Valid, Vxlan, Etree Root,
              Oif:0 Len:14 Data:02 05 86 71 5c 13 2c c2 60 1d 8a 86 08 00
              Sip:100.0.4.2 Dip:1.1.1.13

(vrouter-agent)[root@server4 /]$
```

[root@server4 ~]# **docker exec -it vrouter_vrouter-agent_1 bash**

(vrouter-agent)[root@server4 /]$ **rt --get 2c:c2:60:1d:8a:86 --vrf 2 --family bridge**

- How to ssh to the vm OS from the compute-node

[root@server4 ~]# route -n | grep 169

169.254.0.0 0.0.0.0 255.255.0.0 U 1002 0 0 ens3

169.254.0.3 0.0.0.0 255.255.255.255 UH 0 0 0 vhost0

169.254.0.4 0.0.0.0 255.255.255.255 UH 0 0 0 vhost0

[root@server4 ~]#

[root@server4 ~]# ssh cirros@169.254.0.3

cirros@169.254.0.3's password:

login account is **cirros**. The password is **gocubsgo**

```
$ arp -na
? (100.0.201.2) at 00:00:5e:00:01:00 [ether]  on eth0
? (100.0.201.43) at 02:a4:7e:5a:f3:48 [ether]  on eth0
? (100.0.201.12) at <incomplete>  on eth0
? (100.0.201.1) at 00:00:5e:00:01:00 [ether]  on eth0
$
```

Corporate and Sales Headquarters
Juniper Networks, Inc.
1194 North Mathilda Avenue
Sunnyvale, CA 94089 USA
Phone: 888.JUNIPER (888.586.4737)
or 408.745.2000
Fax: 408.745.2100
www.juniper.net

APAC and EMEA Headquarters
Juniper Networks International B.V.
Boeing Avenue 240
1119 PZ Schiphol-Rijk
Amsterdam, The Netherlands
Phone: 31.0.207.125.700
Fax: 31.0.207.125.701