

OpenStack for Cisco DFA Install Guide

Using Cisco OpenStack Installer

September 26, 2014

Table of Contents

OpenStack for Cisco DFA Install Guide	1
Using Cisco OpenStack Installer	1
1 Hardware and Software Requirements.....	3
2 Pre-Requisite	3
3 OpenStack for DFA Overview.....	3
4 Before You Start.....	5
5 Servers/UCS Topology and OpenStack for DFA Installation	5
5.1 UCS resources required.....	5
5.2 Servers/UCS topology	5
5.3 Installation	6
5.4 Preparing the Build Server	7
5.5 Installing the Controller and Computes.....	14
5.6 Configuration in the Leaf or DCNM	15
5.6.1 Leaf Interface Config and Global Config.....	15
5.6.2 DHCP Configuration in DCNM	16
6 Create Project and Launch VM.....	17
6.1 KNOWN CAVEATS	17
6.2 Steps to create a Project:.....	17
6.3 Steps to create a User for the project:.....	17
6.4 Steps to create the network:.....	18
6.5 Steps to launch the VM:	18
7 Scalability.....	18
8 Known Limitations and Caveats	18
9 Technical Support Model.....	19

1 Hardware and Software Requirements

OpenStack for DFA software is currently packaged with COI (Cisco OpenStack Installer). The Cisco OpenStack Installer is qualified on:

- Ubuntu 12.04 LTS serves as a base operating system.
- KVM serves as the hypervisor.
- Cisco UCS B and C-Series Servers serve as physical compute/storage hardware
- OpenStack for DFA software is based on open source software Grizzly release with Cisco DFA plugin developed at Cisco

DFA support via other commercial OpenStack distributions are being actively worked on and should be available in near future.

2 Pre-Requirement

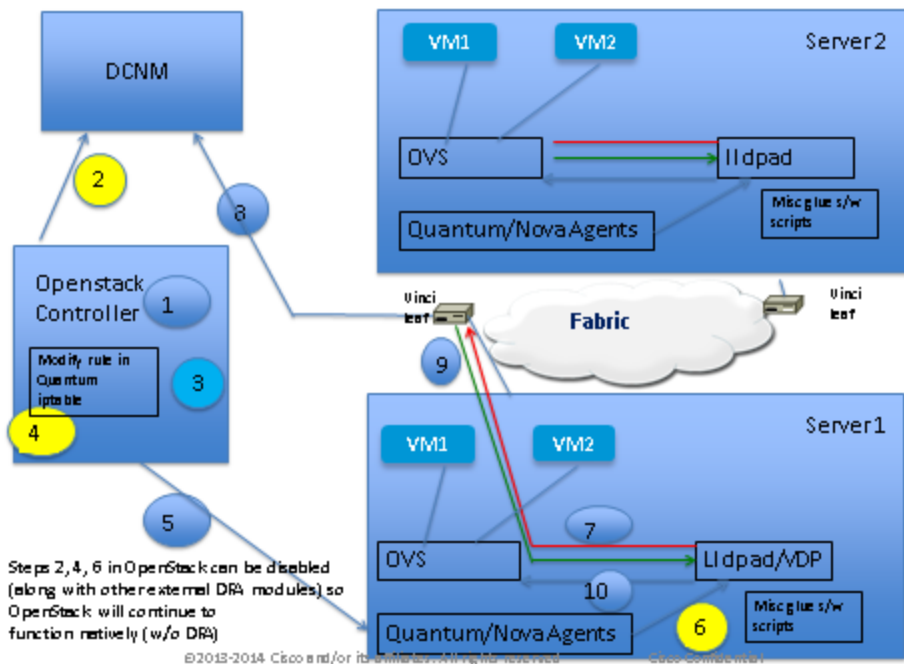
As a pre-requisite to run OpenStack for DFA, please follow Dynamic Fabric Automation Deployment Guideline document to bring up DFA system first, including Nexus switches and DCNM software.

3 OpenStack for DFA Overview

OpenStack serves as one of orchestrators of the cloud enabled through DFA. For this release, all the orchestration is done using OpenStack's dashboard Horizon graphic user interface.

The following diagram provides an overview of the system in the DFA context:

OpenStack for DFA System Flow



The control flow can be summarized as follows:

1. Create a Network of type tunnel with tunnel ID range > 4K. (In OpenStack, tunnel ID is Segment ID)
2. Network Information (Subnet/mask, tenant name), is sent to DCNM via DCNM REST APIs
3. Instance (VM) is launched, specifying the Network that the instance will be a part of.
4. The current security rules in Openstack blocks any incoming DHCP frames from the outside world as well as add host specific iptable rules that will only allow frames with source IP addresses assigned by Openstack. The host specific iptable rules in quantum agent is modified to unblock it.
5. Network information (Subnet/mask, tenant name) and VM information is sent to the compute node.
6. Lldpad (VDP) gets notified about the VM and the segment_id associated with the VM.
7. VDP communicates with the Leaf passing the VM's information along with the Segment ID.
8. Leaf contacts DCNM with the Segment_id for retrieving the Network attributes
9. Leaf responds back with the VLAN to be used for tagging the VM's traffic
10. Lldpad module configures OVS for tagging the packets from the VM destined to the network with the value provided by the Leaf. => VM's VNIC is operational ONLY at this point

4 Before You Start

- We assume you have some general understanding of DFA (Dynamic Fabric Automation) system architecture.
- We use COI package to install OpenStack and all the servers/hosts we qualified for this purpose are Cisco UCS.
- For general COI installation guide:
<http://docwiki.cisco.com/wiki/OpenStack>
- Please refer the Server/UCS Topology and OpenStack for DFA Installation section for pre-requisites
- Ensure CIMC connectivity is there
- Ensure that DCNM is reachable via the Openstack Controller as well as from the Vinci cluster

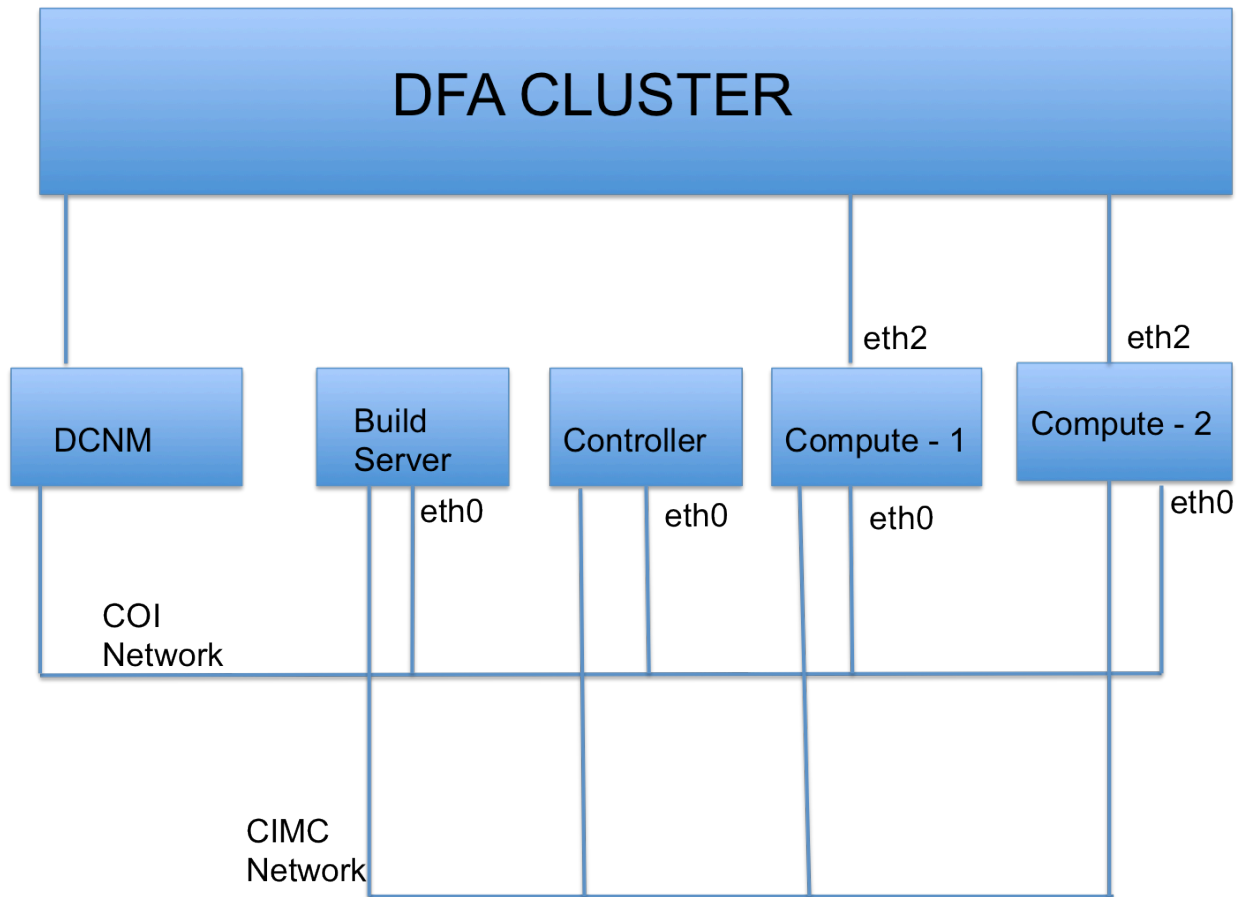
5 Servers/UCS Topology and OpenStack for DFA Installation

5.1 UCS resources required

- One UCS is dedicated for the puppet build server.
- One UCS is dedicated for the OpenStack Controller where the dashboard Horizon is running for doing orchestration.
- As many UCS as the number of Computes are needed. VM's run in the Computes.
- Ensure that all the nodes are configured with CIMC and they are reachable.

5.2 Servers/UCS topology

A sample topology is shown below.



Ensure that all the UCS' are connected to each other. The sample topology has the build server, the controller and the computes connected through eth0. This is referred as the COI network.

The computes have their uplink interface(s) connected to the DFA cluster.

All the CIMC ports of the OpenStack nodes including the build server are connected to the CIMC network.

The DCNM is also connected to the COI network. But, the primary requirement is that the DCNM should be reachable from the Controller.

5.3 Installation

The installation procedure described here follows the instructions given in <http://docwiki.cisco.com/wiki/OpenStack:Grizzly-Multinode>.

So, always refer to that webpage for more details.

The first step is to prepare the build server node.

Ensure that the controller and compute nodes 'hostname' is reachable from the build server node. This means the controller and computes should have at least some bare minimal OS installed.

Then, the build server node fresh installs the controller and computes. This means fresh Ubuntu installation in controller and computes.

5.4 Preparing the Build Server

The procedure described in this document follows the second model of installation (manual method) as given in the wiki..

1. `sudo -H bash`
2. `http_proxy=http://your-proxy.address.com:80/`
3. `https_proxy=https://your-proxy.address.com:443/`
4. I set the `no_proxy` as below, where `vinci-ucs187` is my build server
`export no_proxy=vinci-ucs187,vinci-ucs187.mine.com,127.0.0.1,127.0.1.1,localhost`
5. `apt-get update && apt-get dist-upgrade -y && apt-get install -y puppet git ipmitool`

Please note that the configuration values specified in steps 2, 3 and 4 are specific to the network. Please find the proxy address specific to your network and substitute it for `http_proxy` and `https_proxy`. The last step of 'apt-get' usually runs into issues like the hash sum mismatch errors, failed dependancies, download errors etc. Most of these errors were due to that fact that we were running it in the lab behind the proxies and firewalls. Below lists some of the method we followed to get it to work. **Try the below suggestions ONLY if you run in to problems with the previous apt-get update**

Method 1:

Create the following file:

Edit `/etc/apt/apt.conf.d/40proxy` and insert the following statement:

```
Acquire::http { Proxy "http://proxy.esl.your-proxy.address.com:8080"; };
```

Next since you already ran the `apt- update` command you need to clean your corrupted package list; as root do the following:

```
rm -rf /var/lib/apt/lists
```

then run:

```
sudo apt-get update && apt-get upgrade
```

Method 2:

Prefix the below commands with with "sudo -i"

```
apt-get clean
```

```
rm /var/lib/apt/lists/*
```

```
rm /var/lib/apt/lists/partial/*
```

```
apt-get clean
```

```
apt-get update
```

Method 3:

```
sudo -i
apt-get clean
cd /var/lib/apt
mv lists lists.old
mkdir -p lists/partial
apt-get clean
apt-get update
exit
sudo apt-get update
```

Method 4:

```
sudo fuser -vfv /var/lib/dpkg/lock
sudo rm /var/lib/apt/lists/lock
sudo cp -arf /var/lib/dpkg /var/lib/dpkg.backup
sudo cp /var/lib/dpkg/status-old /var/lib/dpkg/status
sudo cp /var/lib/dpkg/available-old /var/lib/dpkg/available
sudo rm -rf /var/lib/dpkg/updates/*
sudo rm -rf /var/lib/apt/lists
sudo rm /var/cache/apt/*.bin
sudo mkdir /var/lib/apt/lists
sudo mkdir /var/lib/apt/lists/partial
LANG=C;sudo apt-get clean
LANG=C;sudo apt-get autoclean
LANG=C;sudo apt-get --purge autoremove
LANG=C;sudo apt-get update -o APT::Cache-Limit=25165824
sudo dpkg --clear-avail
sudo dpkg --configure -a
LANG=C;sudo apt-get -f install
LANG=C;sudo apt-get --fix-missing install
LANG=C;sudo apt-get update -o APT::Cache-Limit=25165824 && sudo apt-get dist-upgrade
sudo apt-get clean
cd /var/lib/apt
sudo mv lists lists.old
sudo mkdir -p lists/partial
sudo apt-get clean
sudo apt-get update
```

Continue with the below commands after apt-get update

6. git clone <https://github.com/CiscoSystems/grizzly-manifests> ~/cisco-grizzly-manifests/
7. cd ~/cisco-grizzly-manifests
8. git checkout -q dfa-multi-node
9. cp ~/cisco-grizzly-manifests/manifests/* /etc/puppet/manifests
10. cp ~/cisco-grizzly-manifests/templates/* /etc/puppet/templates

Remove the http_proxy and https_proxy environment variables as:

[

export http_proxy=

export https_proxy=

]

11. cd /etc/puppet/manifests
12. python /etc/puppet/manifests/puppet_modules.py)

Now it the time to create the site.pp file. **This is the most critical step.** Copy the site.pp.example to site.pp. Read through the site.pp in detail, it's well documented. Take your time in filling this file. If there's a mistake in filling information in this file, the subsequent step of puppet apply or even installing the controller/computes may not work. Also, follow the sample diff that is pasted below.

13. cp /etc/puppet/manifests/site.pp.example /etc/puppet/manifests/site.pp
14. vi /etc/puppet/manifests/site.pp

A sample diff is given below. Here vinci-ucs116 is the build server, vinci-ucs119 is the controller, vinci-ucs117 and vinci-ucs120 are the computes.

```
diff site.pp.example site.pp
9c9
< #proxy = "http://proxy-server:port-number"
---
> $proxy = "http://proxy.esl.mine.com:80"
42c42
< $build_node_name      = 'build-server'
---
> $build_node_name      = 'vinci-ucs116'
49c49
< $ntp_servers = ['time-server.domain.name']
---
> $ntp_servers = ['ntp.esl.mine.com']
55,56c55,56
< $cobbler_node_ip      = '192.168.242.100'
< $node_subnet          = '192.168.242.0'
---
> $cobbler_node_ip      = '172.28.8.186'
> $node_subnet          = '172.28.8.0'
60c60
< $node_gateway         = '192.168.242.1'
---
> $node_gateway         = '172.28.8.1'
67c67
< $domain_name          = 'domain.name'
---
> $domain_name          = 'mine.com'
94c94
< $password_crypted     =
'$6$UfgWxrIv$k4KfzAEMqMg.fppmSOTd0usI4j6gfjs0962.JXsoJRWa5wMz8yQk4SfInn4.WZ3L
/MCt5u.62tHDGB36EhiKF1'
---
> $password_crypted     =
'$6$UfgWxrIv$ge2goE1NYynQRW2yRimZLsjp6qwfruhmMCi4NCQHmClSVvhYlGo4yZiSDxjAm843
3cU9t1hj1OGFjCbWbbR20.'
111,113c111,113
< $controller_node_address = '192.168.242.10'
```

```

< $controller_node_network      = '192.168.242.0'
< $controller_hostname          = 'control-server'
---
> $controller_node_address      = '172.28.8.189'
> $controller_node_network      = '172.28.8.0'
> $controller_hostname          = 'vinci-ucs119'
118c118
< $db_allowed_network           = '192.168.242.%'
---
> $db_allowed_network           = '172.28.8.0'
162c162
< $service_address = $ipaddress_eth0
---
> $service_address = '172.28.8.189'
188c188
< # $tenant_network_type = 'vlan'
---
> # $tenant_network_type = 'vlan'
200,201c200,201
< $admin_password               = 'pword123'
< $keystone_db_password         = 'keystone_db_pass'
---
> $admin_password               = 'pword123'
> $keystone_db_password         = 'pword123'
203c203
< $mysql_root_password          = 'mysql_db_pass'
---
> $mysql_root_password          = 'pword123'
205,206c205,206
< $nova_db_password             = 'nova_pass'
< $nova_user_password           = 'nova_pass'
---
> $nova_db_password             = 'pword123'
> $nova_user_password           = 'pword123'
208,209c208,209
< $glance_db_password           = 'glance_pass'
< $glance_user_password         = 'glance_pass'
---
> $glance_db_password           = 'pword123'
> $glance_user_password         = 'pword123'
212,216c212,216
< $cinder_user_password         = 'cinder_pass'
< $cinder_db_password           = 'cinder_pass'
< $quantum_user_password        = 'quantum_pass'
< $quantum_db_password          = 'quantum_pass'
< $rabbit_password              = 'openstack_rabbit_password'
---
> $cinder_user_password         = 'pword123'
> $cinder_db_password           = 'pword123'
> $quantum_user_password        = 'pword123'
> $quantum_db_password          = 'pword123'
> $rabbit_password              = 'pword123'
218c218
< $swift_password               = 'swift_pass'
---
> $swift_password               = 'pword123'
257c257

```

```

< # $quantum_core_plugin = 'ovs'
---
> $quantum_core_plugin = 'ovs'
304c304
< $test_file_image_type = 'kvm'
---
> $test_file_image_type = 'cirros'
343,346c343,346
<  cobbler_node { 'control-server':
<   mac           => '00:11:22:33:44:55',
<   ip            => '192.168.242.10',
<   power_address => '192.168.242.110',
---
>  cobbler_node { 'vinci-ucs119':
>   mac           => '7c:ad:74:6f:72:0c',
>   ip            => '172.28.8.189',
>   power_address => '24.1.0.119',
360,363c360,372
<  cobbler_node { 'compute-server01':
<   mac           => '11:22:33:44:55:66',
<   ip            => '192.168.242.21',
<   power_address => '192.168.242.121',
---
>  cobbler_node { 'vinci-ucs120':
>   mac           => '7c:ad:74:6f:69:84',
>   ip            => '172.28.8.190',
>   power_address => '24.1.0.120',
>   power_user    => 'admin',
>   power_password => 'password',
>   power_type    => 'ipmitool',
> }
>
>  cobbler_node { 'vinci-ucs117':
>   mac           => '7c:ad:74:6f:7b:10',
>   ip            => '172.28.8.187',
>   power_address => '24.1.0.117',
494c503
< node build-server inherits build-node { }
---
> node vinci-ucs116 inherits build-node { }
506c515
< $dcnm_ip_var = "10.1.1.1"
---
> $dcnm_ip_var = "172.28.11.156"
508c517
< $dcnm_pass_val = "xxxxx"
---
> $dcnm_pass_val = "pword123"
511c520
< $mysql_host_val = "30.1.1.1"
---
> $mysql_host_val = "172.28.10.177"
513c522
< $mysql_pass_val = "xxxxx"
---
> $mysql_pass_val = "pword123"
517,520c526,533

```

```

< #This is the start of the segment ID base. Ensure that this doesn't conflict
< #with other orchestrators segment ID
< $dfa_tunnel_base_var = "30000"
<
---
> $dfa_tunnel_base_var = "60000"
> #If a compute node is not directly connected to a leaf, but through any
other
> #physical bridge device, then VDP frames will not reach the leaf. In such
cases,
> #VDP frames should use other reserved MCast DMAC that will not be consumed
by
> #the intermediate bridge. The Leaf and intermediate bridge also may need to
be
> #appropriately configured. If there's no such configuration in your setup,
leave
> #the below field as it is, otherwise, enter the Mcast DMAC that you plan to
use.
> $non_nearest_brdmac = "01:88:C2:00:00:00"
528c541
< node 'control-server' inherits os_base {
---
> node 'vinci-ucs119' inherits os_base {
530c543
<     #enable_dhcp_agent      => false,
---
>     enable_dhcp_agent      => false,
536,545c549,560
<     uplink_intf           => 'eth1',
<     dcnm_ip_addr          => $dcnm_ip_var,
<     dcnm_username         => $dcnm_user_val,
<     dcnm_password        => $dcnm_pass_val,
<     mysql_host            => $mysql_host_val,
<     mysql_user            => $mysql_user_val,
<     mysql_password        => $mysql_pass_val,
<     compute               => 'false',
<     gateway_mac           => $gateway_mac_val,
<     dfa_tunnel_base       => $dfa_tunnel_base_var,
---
>     uplink_intf           => 'eth1',
>     dcnm_ip_addr          => $dcnm_ip_var,
>     dcnm_username         => $dcnm_user_val,
>     dcnm_password        => $dcnm_pass_val,
>     mysql_host            => $mysql_host_val,
>     mysql_user            => $mysql_user_val,
>     mysql_password        => $mysql_pass_val,
>     compute               => 'false',
>     gateway_mac           => $gateway_mac_val,
>     dfa_tunnel_base       => $dfa_tunnel_base_var,
>     non_nearest_bridge    => 'false',
>     non_nearest_bridge_mac => $non_nearest_brdmac,
597c612,646
< node 'compute-server01' inherits os_base {
---
> node 'vinci-ucs120' inherits os_base {
>     class { 'compute':
>         internal_ip        => '172.28.8.190',

```

```

>     #enable_dhcp_agent => true,
>     #enable_l3_agent   => true,
>     enable_ovs_agent   => true,
> }
> class { 'dfa':
>     # Modify this to the upstream interface of the compute server connected
>     # to the leaf. Currently, only one interface can be connected to the
>     # leaf
>     uplink_intf           => 'eth2',
>     dcnm_ip_addr         => $dcnm_ip_var,
>     dcnm_username        => $dcnm_user_val,
>     dcnm_password        => $dcnm_pass_val,
>     mysql_host           => $mysql_host_val,
>     mysql_user           => $mysql_user_val,
>     mysql_password       => $mysql_pass_val,
>     compute              => 'true',
>     gateway_mac          => $gateway_mac_val,
>     dfa_tunnel_base      => $dfa_tunnel_base_var,
>     # If Compute is not directly connected to leaf, but through any other
>     # intermediate physical bridge, enter 'true' here. It is case-
sensitive.
>     non_nearest_bridge   => 'false',
>     non_nearest_bridge_mac => $non_nearest_brdmac,
> }
>
> # Reduce OVS log verbosity.
> coe::ovs{ ['netdev', 'netdev_linux']:
>     facility   => 'file',
>     log_level  => 'err',
> }
> }
>
> node 'vinci-ucs117' inherits os_base {
599c648
<     internal_ip           => '192.168.242.21',
---
>     internal_ip           => '172.28.8.187',
602c651
<     #enable_ovs_agent    => true,
---
>     enable_ovs_agent     => true,
608,617c657,670
<     uplink_intf         => 'eth3',
<     dcnm_ip_addr        => $dcnm_ip_var,
<     dcnm_username       => $dcnm_user_val,
<     dcnm_password       => $dcnm_pass_val,
<     mysql_host          => $mysql_host_val,
<     mysql_user          => $mysql_user_val,
<     mysql_password      => $mysql_pass_val,
<     compute             => 'true',
<     gateway_mac         => $gateway_mac_val,
<     dfa_tunnel_base     => $dfa_tunnel_base_var,
---
>     uplink_intf         => 'eth2',
>     dcnm_ip_addr        => $dcnm_ip_var,
>     dcnm_username       => $dcnm_user_val,
>     dcnm_password       => $dcnm_pass_val,

```

```

>     mysql_host           => $mysql_host_val,
>     mysql_user          => $mysql_user_val,
>     mysql_password      => $mysql_pass_val,
>     compute              => 'true',
>     gateway_mac         => $gateway_mac_val,
>     dfa_tunnel_base     => $dfa_tunnel_base_var,
>     # If Compute is not directly connected to leaf, but through any other
>     # intermediate physical bridge, enter 'true' here. It is case-
sensitive.
>     non_nearest_bridge  => 'false',
>     non_nearest_bridge_mac => $non_nearest_brdmac,

```

Please make sure gateway_mac above has the same value as the following in the leaf PoAP configuration:

```
fabric forwarding anycast-gateway-mac 2020.0000.00AA
```

Now after modifying the site.pp file, the puppet apply is done as shown:

```

15. puppet apply -v /etc/puppet/manifests/site.pp
16. puppet plugin download

```

The puppet apply command takes some time to complete. In my case, it takes around 500 secs. Ignore the errors related to Graphite and collectd

Now you are ready to install the controller and computes.

This output of this command should list the controller and compute nodes that you have configured in site.pp. If not recheck site.pp.

```
17. cobbler system list
```

5.5 Installing the Controller and Computes

Now go to the CIMC of the controller and computes and ensure that the boot order has PXE boot as the first option. From the CIMC screen, go to BIOS -> Configure Boot Order.

A node is fresh installed as follows. Run this command for each controller and compute resources. 'name' is the hostname of the controller or compute nodes and can be seen in the output of the previous cobbler command.

```
/etc/puppet/manifests/clean_node.sh {name}
```

No "\" is needed in the above command. Wiki displays it that way.

- Now you will see that fresh Ubuntu installation happens in the nodes. You will see from the console screen that it obtains its initial DHCP address from the build server, downloads the file and starts the installation.
- After installation is complete, the node boots up and you should see the login prompt.
- Now, wait for the puppet installation to complete, you will see the following lines in /var/log/syslog. First time you may need to wait for more than 30 mins.

```
puppet-agent[2380]: Finished catalog run in 500 seconds
```

- Now, associate an image to be used by VM's launched using Openstack. For Cirros image, the following is done.

wget http://download.cirros-cloud.net/0.3.1/cirros-0.3.1-x86_64-disk.img

```
glance add name="cirros-x86_64" is_public=true disk_format=qcow2
container_format=ovf < cirros-0.3.1-x86_64-disk.img
```

- For the compute node, after the puppet run is complete, reboot the node. **This is an important step** and is needed as a work around for the current LLDPAD implementation.
- After Compute node is rebooted and is up, wait for the puppet run to complete and then ensure that the daemons LLDPAD and Packet Capture are running as follows:

```
pidof lldpad
pidof pktcpt
```

5.6 Configuration in the Leaf or DCNM

5.6.1 Leaf Interface Config and Global Config

- For the leaf interface connected to the UCS server, “fabric forwarding port-tracking” configuration needs to be done:

```
int e1/35
switchport mode trunk
```

- Need to set the speed of the interface connected to compute node correctly (currently DCNM PoAP screen does not allow you to configure this)

```
fabric forwarding port-tracking
speed 1000
no shut
```

- Add the global command:

```
evb batch-response disable
```

- Again, make sure the leaf node connected to the OpenStack compute node needs to have the same anycast gateway mac as in the site.pp file of the build server:

```
N6k-23# sh run | inc anycast

fabric forwarding anycast-gateway-mac 2020.0000.00AA
```

- There are some LC's where the interface connecting the leaf and UCS does not come up. Try the following workaround.

1. On the leaf, load the debug plugin.
2. cd /mnt/pss

3. rm norcal_disable_qsf
4. Reload the leaf

5.6.2 DHCP Configuration in DCNM

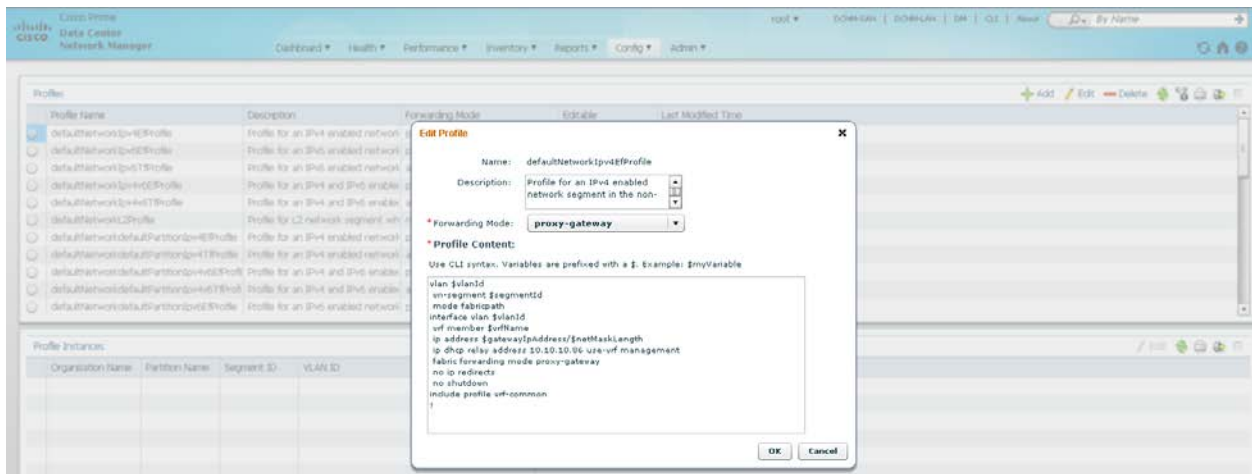
- For auto-configuration to work, the profile's DHCP address needs to be modified manually in DCNM. This is because Openstack does not populate the DHCP address to DCNM, since it has no way of knowing this information. For example, under a profile, you will see a config like the below:

```
ip dhcp relay address $dhcpServerAddr use-vrf management
```

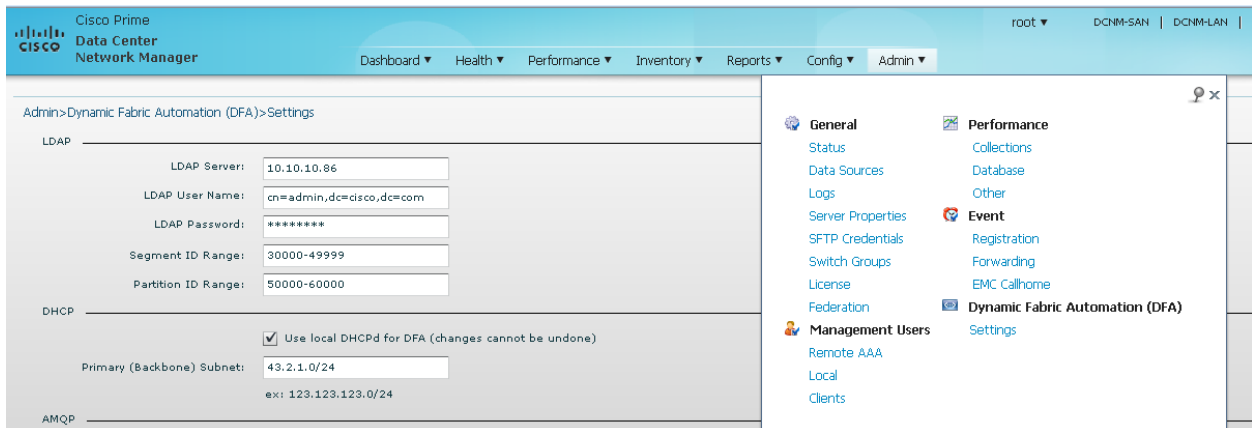
Modify this to the actual IP address of the DHCP server. Like:

```
ip dhcp relay address 10.10.2.86 use-vrf management
```

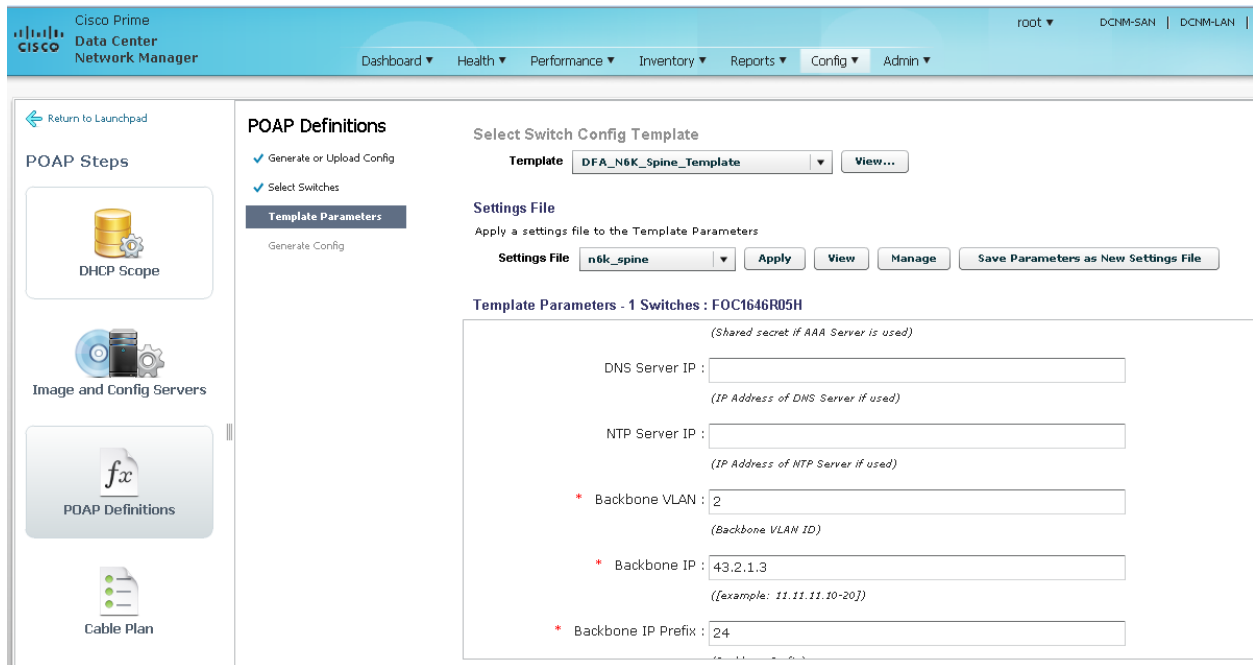
Go to “Config”, then DFA Profiles:



- Also fill the DFA backbone vlan subnet value in the “Settings” of Dynamic Fabric Automation (DFA) section under Admin tap in DCNM console.



- Need make sure that Primary (Backbone) subnet with “Use local DHCPd for DFA” checked is consistent with the backbone vlan subnet for the leaf and spine nodes in the DFA fabric. If the fabric is brought up from PoAP in DCNM, the configuration screen will look like the following:



6 Create Project and Launch VM

The information provided in this section is generic to OpenStack and it is provided here for your convenience with the exception of “ConfigProfile”, which is DFA specific.

6.1 KNOWN CAVEATS

- When a project is created, do not have hyphen "-" in the project name.
- Bulk Create and Delete of VM's is supported only to an extent. Pls refer the caveats section.
- Do not give space for a network name or project name or a VM name.

6.2 Steps to create a Project:

- Login to Horizon as admin, password “pword123”. Click on "projects" and "create project". If you had set a different admin password in site.pp, use that to login.
- Enter a Name for the project, DO NOT have hyphen "-" in the project name.
- Click on "Create Project".

6.3 Steps to create a User for the project:

- Click on Users and then do "Create User"

- Fill in all the fields, select the project you just created and select the role as "admin". The Network information will not be populated to DCNM correctly, if you fail to specify the role as "admin".

6.4 Steps to create the network:

- Login as the user just created.
- Click on the "project" tab.
- Click on "Networks" and then "Create Network". Specify a Name for the network and go to the subnet tab. This is MANDATORY.
- Specify a Network Address for the subnet.
- Click on the "Subnet Detail" tab and then uncheck the "Enable DHCP" option.
- Go to the "ConfigProfile" tab and select the profile from the pull down menu

6.5 Steps to launch the VM:

- Click on "Instances" and then "Launch Instance"
- Click on "Image" drop down menu and select the image. There will be cirros image by default.
- Give a name for the Instance.
- Go to the "Networking" tab and select the network from "Available network".
- Click "Launch".

7 Scalability

OpenStack is supposed to support many compute nodes from the control node running the Horizon dashboard. This should not be related to the added DFA functionality.

By default without specific configuration, OpenStack supports 10 VMs per compute node. This is the recommend number for a user. Our QA has qualified about 20 or less VM's/instances per each OpenStack compute node.

Batching VM creation and deletion with OpenStack should be used with great caution. We support 10 or less VMs per batch creation and deletion on two compute nodes (i.e., tested). Batching capability beyond that is best-effort.

8 Known Limitations and Caveats

- Currently live-migration/vmotion is not supported as it is not supported in COI Grizzly release
- OpenStack does not support IPv6 yet
- No service integration supported
- All added DFA functionality in OpenStack is done through the Horizon dashboard for this release.

- If a compute node with live VMs gets reloaded, OpenStack will lose these VM's
- Currently a single port of a server running OpenStack is allowed to connect to a DFA leaf port.
- It is required that the compute nodes should always be connected to the build server at least during reboots and puppet should not be disabled
- vm's vnic interface up/down not detected by vdp (use VM deletion/creation as potential workaround)
- Observed LLDPad does not come up after COI installation so a manual reload is necessary as a workaround for now
- All the subnet creation for network should be done through Horizon project/network menu. Other way of creating network subnet might not work.
- All orchestration tasks are done through the Horizon dashboard for this release
- VM provisioned through Horizon will always get its IP address via DHCP server in DCNM
- Cirros and mini Ubuntu images are currently tested

9 Technical Support Model

- OpenStack is based on open source and is generally supported through its community using a best effort approach.
- All general COI installation support will be from Cisco COI team. See the Before You Start section above.
- COI-DFA part support will be from Cisco DFA team.
- DFA part of the OpenStack DFA support will come from Cisco DFA team.