Using NIC templates with RedHat Director/Tripleo

# What are NIC templates

TripleO provides the flexibility to have different NIC templates for different overcloud roles. For example: there might be differences between the NIC/networking layout for the overcloud-compute-nodes and the overcloud-contrail-controller-nodes.

These NIC templates provide data to the backend scripts that take care of provisioning the network on the overcloud nodes. The templates are written in standard JSON formats.

# How NIC templates work

The ‘resources’ section within the template includes all the networking information for the corresponding overcloud role. This information includes:

1. Number of NICs
2. Network associated with each NIC
3. Static routes associated with each NIC
4. Any VLAN configuration which is tied to a particular NIC
   1. Network associated with each VLAN interface
   2. Static routes associated with each VLAN

A lot of good information on what each of these sections look like can be accessed here:

<https://access.redhat.com/documentation/en-us/red_hat_openstack_platform/10/html/advanced_overcloud_customization/sect-isolating_networks>

The link mentioned above has great examples on how to define a NIC within the template. We’ll use this information in the subsequent sections.

Please note that Director-10 has a limitation wherein all the overcloud networks must be stretched at layer-2 to all the overcloud nodes. If the overcloud nodes are physical servers which are present in different racks/subnets of an IP fabric, then you’ll have to first stretch all the overcloud networks to the physical servers.

Deploying an overcloud using TripleO/Director across multiple subnets is an upstream feature and a work-in-progress at the time of writing this document (ETA Q4-2017).

# Contrail NIC templates

As part of deployment, a net template must be provided. The net template files are also available at the same location:

[stack@undercloud contrail]$ ls -lrt | grep contrail-net

-rw-rw-r--. 1 stack stack 1866 Sep 19 17:10 contrail-net-storage-mgmt.yaml

-rw-rw-r--. 1 stack stack 894 Sep 19 17:10 contrail-net-single.yaml

-rw-rw-r--. 1 stack stack 1528 Sep 19 17:10 contrail-net-dpdk.yaml

-rw-rw-r--. 1 stack stack 1504 Sep 19 17:10 contrail-net-bond-vlan.yaml

-rw-rw-r--. 1 stack stack 1450 Sep 19 17:12 contrail-net.yaml

The ‘resource\_registry’ section of this file specifies which NIC template must be used for each role:

[stack@undercloud contrail]$ cat contrail-net.yaml

resource\_registry:

OS::TripleO::Compute::Net::SoftwareConfig: contrail-nic-config-compute.yaml

OS::TripleO::ContrailDpdk::Net::SoftwareConfig: contrail-nic-config-compute-dpdk-bond-vlan.yaml

OS::TripleO::Controller::Net::SoftwareConfig: contrail-nic-config.yaml

OS::TripleO::ContrailController::Net::SoftwareConfig: contrail-nic-config.yaml

OS::TripleO::ContrailAnalytics::Net::SoftwareConfig: contrail-nic-config.yaml

OS::TripleO::ContrailAnalyticsDatabase::Net::SoftwareConfig: contrail-nic-config.yaml

OS::TripleO::ContrailTsn::Net::SoftwareConfig: contrail-nic-config-compute.yaml

parameter\_defaults:

ControlPlaneSubnetCidr: '24'

ControlPlaneDefaultRoute: 192.0.2.1

InternalApiNetCidr: 10.0.0.0/24

InternalApiAllocationPools: [{'start': '10.0.0.10', 'end': '10.0.0.200'}]

InternalApiDefaultRoute: 10.0.0.1

ManagementNetCidr: 10.1.0.0/24

ManagementAllocationPools: [{'start': '10.1.0.10', 'end': '10.1.0.200'}]

ManagementInterfaceDefaultRoute: 10.1.0.1

ExternalNetCidr: 10.2.0.0/24

ExternalAllocationPools: [{'start': '10.2.0.10', 'end': '10.2.0.200'}]

EC2MetadataIp: 192.0.2.1 # Generally the IP of the Undercloud

DnsServers: ["8.8.8.8","8.8.4.4"]

VrouterPhysicalInterface: vlan20

VrouterGateway: 10.0.0.1

VrouterNetmask: 255.255.255.0

ControlVirtualInterface: eth0

PublicVirtualInterface: vlan10

VlanParentInterface: eth1 # If VrouterPhysicalInterface is a vlan interface using vlanX notation

## NIC Templates for Control nodes

In this example, all the OpenStack-Controller and all the Contrail-Control-Plane roles use the NIC template named ‘contrail-nic-config.yaml’. Note that the Compute roles and the DPDK roles use different NIC templates.

These NIC template files can be accessed at this location:

[stack@undercloud contrail]$ ls -lrt | grep contrail-nic-config-

-rw-rw-r--. 1 stack stack 5615 Sep 19 17:10 contrail-nic-config-vlan.yaml

-rw-rw-r--. 1 stack stack 5568 Sep 19 17:10 contrail-nic-config-storage-mgmt.yaml

-rw-rw-r--. 1 stack stack 3861 Sep 19 17:10 contrail-nic-config-single.yaml

-rw-rw-r--. 1 stack stack 5669 Sep 19 17:10 contrail-nic-config-compute-storage-mgmt.yaml

-rw-rw-r--. 1 stack stack 3864 Sep 19 17:10 contrail-nic-config-compute-single.yaml

-rw-rw-r--. 1 stack stack 5385 Sep 19 17:10 contrail-nic-config-compute-dpdk.yaml

-rw-rw-r--. 1 stack stack 5839 Sep 19 17:10 contrail-nic-config-compute-bond-vlan.yaml

-rw-rw-r--. 1 stack stack 5666 Sep 19 17:10 contrail-nic-config-compute-bond-vlan-dpdk.yaml

-rw-rw-r--. 1 stack stack 5538 Sep 19 17:10 contrail-nic-config-compute-bond-dpdk.yaml

-rw-rw-r--. 1 stack stack 5132 Sep 19 17:13 contrail-nic-config-compute.yaml

-rw-r--r--. 1 stack stack 5503 Sep 19 17:13 contrail-nic-config-compute-dpdk-bond-vlan.yaml

Note that these NIC template files are examples, and you may have to modify these according to your cluster’s topology.

Also, these examples call out NIC-names in the format of nic1, nic2, nic3 etc (nic.$<number>). Think of these as variables, and Director’s backend scripts translate these nic numbers into actual interface names based on the interface boot order. So if you specify nic1, nic2 and nic3 in the template and the boot order of interfaces is eth0, eth1 and eth2, then the mapping of these nic variables to actual interfaces would look like:

* Nic1 mapped to eth0
* Nic2 mapped to eth1
* Nic3 mapped to eth2

A common mistake while defining NIC templates is that the boot order of NICs is not set correctly. Because of this, your deployment might go beyond the network\_configuration stage but there might be connectivity issues as the IP/Subnet/route information might not be configured correctly for the NICs of overcloud nodes.

As a next step, let’s zoom into the network\_config NIC template used by the Controllers: contrail-nic-config.yaml:

network\_config:

-

type: interface

name: nic1

use\_dhcp: false

dns\_servers: {get\_param: DnsServers}

addresses:

-

ip\_netmask:

list\_join:

- '/'

- - {get\_param: ControlPlaneIp}

- {get\_param: ControlPlaneSubnetCidr}

routes:

-

ip\_netmask: 169.254.169.254/32

next\_hop: {get\_param: EC2MetadataIp}

-

type: vlan

use\_dhcp: false

vlan\_id: {get\_param: InternalApiNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: InternalApiIpSubnet}

routes:

-

default: true

next\_hop: {get\_param: InternalApiDefaultRoute}

-

type: vlan

vlan\_id: {get\_param: ManagementNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: ManagementIpSubnet}

-

type: vlan

vlan\_id: {get\_param: ExternalNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: ExternalIpSubnet}

-

type: vlan

vlan\_id: {get\_param: StorageNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: StorageIpSubnet}

-

type: vlan

vlan\_id: {get\_param: StorageMgmtNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: StorageMgmtIpSubnet}

Let’s look at the sub-sections of this template:

1. Definition for NIC-1:

-

type: interface

name: nic1

use\_dhcp: false

dns\_servers: {get\_param: DnsServers}

addresses:

-

ip\_netmask:

list\_join:

- '/'

- - {get\_param: ControlPlaneIp}

- {get\_param: ControlPlaneSubnetCidr}

routes:

-

ip\_netmask: 169.254.169.254/32

next\_hop: {get\_param: EC2MetadataIp}

Observations:

* This is the definition for an interface called ‘nic1’
* The DNS server is defined. Make sure that this parameter has a valid value. Most commonly, this variable is assigned a value in the contrail-services.yaml file
* An IP and subnet is provided under the ‘addresses’ section. Note that these values are again variables, and the format is: $(Network\_Name)IP and $(Network\_Name)SubnetCidr.
  + This means that this particular NIC is on the ControlPlane Network. In the background, this NIC might be connected to an access-port on a switch for the ControlPlane VLAN.
* In the ‘routes’ section, there’s a /32 route out of this NIC. At the time of planning the networking for your cluster, you may need to provision static routes on the overcloud roles. Use the format mentioned under the ‘routes’ section to specify any such static routes.

1. Definition for NIC-2:

-

type: vlan

use\_dhcp: false

vlan\_id: {get\_param: InternalApiNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: InternalApiIpSubnet}

routes:

-

default: true

next\_hop: {get\_param: InternalApiDefaultRoute}

-

type: vlan

vlan\_id: {get\_param: ManagementNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: ManagementIpSubnet}

-

type: vlan

vlan\_id: {get\_param: ExternalNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: ExternalIpSubnet}

-

type: vlan

vlan\_id: {get\_param: StorageNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: StorageIpSubnet}

-

type: vlan

vlan\_id: {get\_param: StorageMgmtNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: StorageMgmtIpSubnet}

Observations:

* In this example, NIC-2 has multiple VLANs defined on it.
  + In the background, NIC-2 might be connected to a switch’s trunk port. And all these corresponding VLANs are allowed on the trunk.
  + Given that Director based deployments need the administrator to use a bunch f networks, it’s a very common requirement/design to use VLAN interface interfaces on the overcloud nodes. This way the administrators do not have to worry about having 6-7 physical NICs on each overcloud node.
* For each VLAN interface, the vlan\_id is defined. Note that the vlan\_id again points to a variable. These variables can be assigned values in the contrail-net.yaml that we discussed earlier.
* Another very important observation here is setting the default route. In this example, the default route was provisioned on the VLAN interface in the InternalAPI network. Note that the next-hop points to a variable. This variable can also be set in the contrail-net.yaml file.

## NIC templates for compute nodes

The NIC definitions for compute roles are slightly different from the ones for Control nodes. This is because Contrail provisions a logical interface called ‘vhost0’ on all compute nodes, and this interface must be provided in the NIC definition file for a compute node. Vhost0 is the logical interface that gets attached to the control/data network (or the InternalAPI network in TripleO based installation).

In the contrail-net.yaml example provided above, the NIC template being used for the compute nodes was ‘contrail-nic-config-compute.yaml’. Let’s look at the contents of the ‘resources’ section this file:

resources:

OsNetConfigImpl:

type: OS::Heat::StructuredConfig

properties:

group: os-apply-config

config:

os\_net\_config:

network\_config:

-

type: interface

name: nic1

use\_dhcp: false

dns\_servers: {get\_param: DnsServers}

addresses:

-

ip\_netmask:

list\_join:

- '/'

- - {get\_param: ControlPlaneIp}

- {get\_param: ControlPlaneSubnetCidr}

routes:

-

ip\_netmask: 169.254.169.254/32

next\_hop: {get\_param: EC2MetadataIp}

-

type: vlan

vlan\_id: {get\_param: InternalApiNetworkVlanID}

device: nic2

-

type: interface

name: vhost0

use\_dhcp: false

addresses:

-

ip\_netmask: {get\_param: InternalApiIpSubnet}

routes:

-

default: true

next\_hop: {get\_param: InternalApiDefaultRoute}

-

type: vlan

vlan\_id: {get\_param: ManagementNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: ManagementIpSubnet}

-

type: vlan

vlan\_id: {get\_param: ExternalNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: ExternalIpSubnet}

-

type: vlan

vlan\_id: {get\_param: StorageNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: StorageIpSubnet}

-

type: vlan

vlan\_id: {get\_param: StorageMgmtNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: StorageMgmtIpSubnet}

Details of each NIC:

1. NIC-1:

-

type: interface

name: nic1

use\_dhcp: false

dns\_servers: {get\_param: DnsServers}

addresses:

-

ip\_netmask:

list\_join:

- '/'

- - {get\_param: ControlPlaneIp}

- {get\_param: ControlPlaneSubnetCidr}

routes:

-

ip\_netmask: 169.254.169.254/32

next\_hop: {get\_param: EC2MetadataIp}

This looks very similar to the NIC definition template for the control-nodes mentioned earlier. So in this example topology, the first NIC for all the compute nodes is connected to the ControlPlane network. Note that this is again untagged, so this NIC might be connected to an access port on the underlay switch.

1. NIC-2:

-

type: interface

name: vhost0

use\_dhcp: false

addresses:

-

ip\_netmask: {get\_param: InternalApiIpSubnet}

routes:

-

default: true

next\_hop: {get\_param: InternalApiDefaultRoute}

-

type: vlan

vlan\_id: {get\_param: InternalApiNetworkVlanID}

device: nic2

-

type: vlan

vlan\_id: {get\_param: ManagementNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: ManagementIpSubnet}

-

type: vlan

vlan\_id: {get\_param: ExternalNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: ExternalIpSubnet}

-

type: vlan

vlan\_id: {get\_param: StorageNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: StorageIpSubnet}

-

type: vlan

vlan\_id: {get\_param: StorageMgmtNetworkVlanID}

device: nic2

addresses:

-

ip\_netmask: {get\_param: StorageMgmtIpSubnet}

This also looks very similar to the parameters defined in the NIC definition template for the control-plane nodes. There are two major differences though:

* VLAN sub-interface for InternalApiNetwork does not have an IP address
* Vhost0 interface holds the IP address for InternalApiNetwork
  + If you’re using stock TripleO based installation, then the IP address for the InternalApiNetwork will always be configured on the vhost0 interface

There are some more additional parameters that are required to successfully provision compute nodes. These values are normally specified in the contrail-net.yaml file:

VrouterPhysicalInterface: vlan20

VrouterGateway: 10.0.0.1

VrouterNetmask: 255.255.255.0

VlanParentInterface: eth1 # If VrouterPhysicalInterface is a vlan interface using vlanX notation

Details of all of these variables:

* VrouterPhysicalInterface: This is the interface on which vhost0 interface gets attached. This may be a physical NIC (e.g. eth2 or enps0f0), or a VLAN interface (e.g. Vlan20)
* VrouterGateway: This is the IP address of the SDN gateway. In a lot of deployments, this might be the IP address of the MX router’s IP address. This IP must be reachable via the InternalAPI network
* VrouterNetmask: subnet mask for the vhost0 interface (this is provisioned in the compute nodes’ config files).
* VlanParentInterface: This is optional, and needed only if vhost0 needs to be attached to a VLAN interface.

The topology used in this document has the following constraints:

1. First NIC must be connected to the ControlPlane network
2. Second NIC must have separate VLAN interfaces for every other network.

Keeping these limitations in mind, we specified ‘eth1’ as the VlanParentInterface. Note that ‘nic-2’ was specified as the interface with multiple VLAN sub-interfaces in the NIC definition template. In RHEL 7.3/7.4 (at the time of writing this document), the NIC’s manifest as eth0, eth1 and so on. Because of this, NIC-2 translated to eth-1.

As part of the development process, there are several NIC templates that are made available to the users. These templates are named according to the topology that they’re trying to solve, and are available in the environments/contrail/ directory. Please modify these templates according to your topology before deploying Contrail with TripleO/RedHat-Director.