Installation

Installation Overview

Cisco Policy Suite VMs is deployed using either Nova boot commands or Heat templates.

Create CPS VMs using Nova Boot Commands

Step 1 Create cloud configuration files for each VM to be deployed (xxx-cloud.cfg). These configurations are used to define the OpenStack parameters for each CPS VM.

Refer to Sample Cloud Config Files, on page 4 to create these files.

Step 2 Run the following command on the control node:

```
source ~/keystonerc_core
```

Step 3 Deploy each CPS VM with the following nova boot command:

```
nova boot --config-drive true --user-data=<node>-cloud.cfg
 --image "base_vm" --flavor "<cluman|pcrfclient0x|sm|lb0x|qns0x>"
--nic net-id="<Internal n/w id>,v4-fixed-ip=
<Internal network private IP>"
--nic net-id="<Management network id>,v4-fixed-ip=
<Management n/w public ip>" --block-device-mapping
"/dev/vdb=<Volume id of iso>:::0"
--availability-zone "<availability zone:Host info>"
"cluman"
```

Note Configure the networks, internal IPs, management IPs and availability zones based on the requirements of your environment.

The following example shows the nova boot commands to deploy a Cluster Manager (cluman), two OAMs (pcrfclients), two sessionmgrs, two Policy Directors (load balancers), and four Policy Server (qns) VMs.

In the following example:

- 172.16.2.200 is the Internal VIP address.
- 172.18.11.156 is the management VIP address.
- 192.168.2.200 is the Gx VIP address

```
nova boot --config-drive true --user-data=cluman-cloud.cfg
--image "CPS_xx_x_x_Base" --flavor "cluman" --nic net-id=
"8c74819c-f3cb-46ad-b69a-d0d521b336d5,v4-fixed-ip=172.16.2.19"
--nic net-id="27a07da0-116f-4453-94b6-457bad9154b0,v4-fixed-ip=172.18.11.101"
--block-device-mapping "/dev/vdb=edf0113a-2ea0-4286-97f0-ee149f35b0d2:::0"
--availability-zone Zone1 "cluman"

nova boot --config-drive true --user-data=pcrfclient01-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "pcrfclient01" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.20" --nic
net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.152"
--block-device-mapping "/dev/vdb=139f2b90-eb74-4d5e-9e20-2af3876a7572:::0"
 --availability-zone "az-1:os8-compute-1.cisco.com" "pcrfclient01"

nova boot --config-drive true --user-data=pcrfclient02-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "pcrfclient02"  --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.21" --nic net-id=
"24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.153"
--block-device-mapping "/dev/vdb=27815c35-c5e8-463b-8ce4-fb1ec67d9446:::0"
--availability-zone "az-2:os8-compute-2.cisco.com" "pcrfclient02"

nova boot --config-drive true --user-data=sessionmgr01-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "sm"  --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.22"
--nic net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.157"
--block-device-mapping "/dev/vdb=8c3577d2-74f2-4370-9a37-7370381670e4:::0"
--availability-zone "az-1:os8-compute-1.cisco.com" "sessionmgr01"

nova boot --config-drive true --user-data=sessionmgr02-cloud.cfg
--image "base_vmCPS_xx_x_x_Base" --flavor "sm"
--nic net-id="2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.23"
--nic net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.158"
--block-device-mapping "/dev/vdb=67aa5cbd-02dd-497e-a8ee-797ac04b85f0:::0"
--availability-zone "az-2:os8-compute-2.cisco.com" "sessionmgr02"

nova boot --config-drive true --user-data=lb01-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "lb01" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.201"
--nic net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.154"
--nic net-id="d0a69b7f-5d51-424a-afbe-5f6486c6e90d,v4-fixed-ip=192.168.2.201"
--availability-zone "az-1:os8-compute-1.cisco.com" "lb01"

nova boot --config-drive true --user-data=lb02-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "lb02" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.202"
--nic net-id="24d71ec2-40b0-489f-9f0c-ca8a42a5c834,v4-fixed-ip=172.18.11.155"
--nic net-id="d0a69b7f-5d51-424a-afbe-5f6486c6e90d,v4-fixed-ip=192.168.2.202"
--availability-zone "az-2:os8-compute-2.cisco.com" "lb02"
```

```
nova boot --config-drive true --user-data=qns01-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "qps" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.24"
--availability-zone "az-1:os8-compute-1.cisco.com" "qns01"

nova boot --config-drive true --user-data=qns02-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "qps" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.25"
--availability-zone "az-1:os8-compute-1.cisco.com" "qns02"

nova boot --config-drive true --user-data=qns03-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "qps" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.26"
--availability-zone "az-2:os8-compute-2.cisco.com" "qns03"

nova boot --config-drive true --user-data=qns04-cloud.cfg --image
"CPS_xx_x_x_Base" --flavor "qps" --nic net-id=
"2544e49e-0fda-4437-b558-f834e73801bb,v4-fixed-ip=172.16.2.27"
--availability-zone "az-2:os8-compute-2.cisco.com" "qns04"
```

Note Use the `cinder list` command to query OpenStack for the block-device-mapping IDs for the above nova boot commands.

Step 4 Update the ports to allow address pairing on the Neutron ports:

a) Use the following command to find the Neutron port ID for the lb01 internal IP address:

```
openstack port list | grep "<lb01_internal_IP>"
```

b) Use the following command to find the Neutron port ID for the lb02 internal IP address:

```
openstack port list | grep "<lb02_internal_IP>"
```

c) Update the above two Neutron ports to allow Internal VIP address by running the following command for each of the above ports:

```
openstack port set --allowed-address-pair ip_address=IP_ADDR|CIDR[,mac_address=MAC_ADDR]
```

For example:

```
[root@os8-control cloud(keystone_core)]# openstack port list | grep "172.16.2.201"
| db8944f3-407d-41ef-b063-eabbab43c039 || fa:16:3e:b1:f3:ab |
ip_address='172.16.2.201',subnet_id='6cfd1d1b-0931-44ad-bdc9-5015dc69f9d0' | ACTIVE |

[root@os8-control cloud(keystone_core)]# openstack port set --allowed-address-pairs
ip-address=172.16.2.200 db8944f3-407d-41ef-b063-eabbab43c039
```

d) Repeat Step c for External VIP addresses using neutron ports for the lb01/lb02 Management IP address and also Gx VIP address using neutron ports for lb01/lb02 Gx IP addresses.

Step 5 Wait approximately 10 minutes for the Cluster Manager VM to be deployed, then check the readiness status of the Cluster Manager VM using the following API:

GET http://<*Cluster Manager IP*>:8458/api/system/status/cluman

Refer to /api/system/status/cluman for more information.

When this API response indicates that the Cluster Manager VM is in a ready state (`"status": "ready"`), continue with Deploy CPS, on page 25.

Refer also to the `/var/log/cloud-init-output.log` on the Cluster Manager VM for deployment details.

Sample Cloud Config Files

For nova boot installation of CPS, you must create a cloud configuration file for each CPS VM to be deployed.

The following sections show an example Cluster Manager cloud configuration (`cluman-cloud.cfg`), and a pcrfclient01 cloud configuration (`pcrfclient01-cloud.cfg`).

These files must be placed in the directory in which you execute the nova launch commands, typically `/root/cps-install/`.

Cluster Manager Configuration File (for install type mobile)

```
#cloud-config
write_files:
 - path: /etc/sysconfig/network-scripts/ifcfg-eth0
   encoding: ascii
   content: |
     DEVICE=eth0
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.16.2.19    <---- Internal IP to access via private IP
     NETMASK=255.255.255.0
     NETWORK=172.16.2.0    <------ Internal network
   owner: root:root
   permissions: '0644'
 - path: /etc/sysconfig/network-scripts/ifcfg-eth1
   encoding: ascii
   content: |
     DEVICE=eth1
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.18.11.101   <---- Management IP to access via public IP
     NETMASK=255.255.255.0
     GATEWAY=172.18.11.1
     NETWORK=172.18.11.0
   owner: root:root
   permissions: '0644'
 - path: /var/lib/cloud/instance/payload/launch-params
   encoding: ascii
   owner: root:root
   permissions: '0644'
 - path: /root/.autoinstall.sh
   encoding: ascii
   content: |
     #!/bin/bash
     if [[ -d /mnt/iso ]] && [[ -f /mnt/iso/install.sh ]]; then
       /mnt/iso/install.sh << EOF
     mobile
     y
     1
     EOF
     fi
   permissions: '0755'
mounts:
 - [ /dev/vdb, /mnt/iso, iso9660, "auto,ro", 0, 0 ]
runcmd:
 - ifdown eth0
 - ifdown eth1
 - echo 172.16.2.19 installer >> /etc/hosts   <---- Internal/private IP of cluman
 - ifup eth0
 - ifup eth1
 - /root/.autoinstall.sh
```

Note If actual hostname for Cluster Manager VM is other than 'installer', then modify installer/cluman entry in `/etc/hosts` accordingly.

Example:

```
echo 172.16.2.19 installer <actual-hostname> >> /etc/hosts
```

Non-Cluster Manager Configuration File

- The following example configuration file is for pcrfclient01. You must create separate configuration files for each CPS VM to be deployed.

  For each file, modify the `NODE_TYPE`, and network settings (`IPADDR`, `GATEWAY`, `NETWORK`) accordingly.

  A typical CPS deployment would require the following files:

  - • pcrfclient01-cloud.cfg
    • pcrfclient02-cloud.cfg
    • lb01-cloud.cfg
    • lb02-cloud.cfg
    • sessionmgr01-cloud.cfg
    • sessionmgr02-cloud.cfg
    • qns01-cloud.cfg
    • qns02-cloud.cfg
    • qns03-cloud.cfg
    • qns04-cloud.cfg

    • pcrfclient01-cloud.cfg
    • pcrfclient02-cloud.cfg
    • lb01-cloud.cfg
    • lb02-cloud.cfg
    • sessionmgr01-cloud.cfg
    • sessionmgr02-cloud.cfg
    • qns01-cloud.cfg
    • qns02-cloud.cfg
    • qns03-cloud.cfg
    • qns04-cloud.cfg

- Modify `IPADDR` to the IP address used in nova boot command for that interface.

- Set `NETMASK`, `GATEWAY`, and `NETWORK` according to your environment.

```
#cloud-config
#hostname: pcrfclient01
fqdn: pcrfclient01
write_files:
 - path: /etc/sysconfig/network-scripts/ifcfg-eth0
   encoding: ascii
   content: |
     DEVICE=eth0
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.16.2.20
     NETMASK=255.255.255.0
     NETWORK=172.16.2.0
   owner: root:root
   permissions: '0644'
 - path: /etc/sysconfig/network-scripts/ifcfg-eth1
   encoding: ascii
   content: |
     DEVICE=eth1
     BOOTPROTO=none
     NM_CONTROLLED=none
     IPADDR=172.18.11.152
     NETMASK=255.255.255.0
     GATEWAY=172.18.11.1
     NETWORK=172.18.11.0
   owner: root:root
   permissions: '0644'
 - path: /var/lib/cloud/instance/payload/launch-params
   encoding: ascii
   owner: root:root
   permissions: '0644'
 - path: /etc/broadhop.profile
   encoding: ascii
   content: "NODE_TYPE=pcrfclient01\n"
   owner: root:root
   permissions: '0644'
runcmd:
 - ifdown eth0
 - ifdown eth1
 - echo 172.16.2.19 installer >> /etc/hosts
 - ifup eth0
 - ifup eth1
 - sed -i '/^HOSTNAME=/d' /etc/sysconfig/network && echo HOSTNAME=pcrfclient01 >>
/etc/sysconfig/network
 - echo pcrfclient01 > /etc/hostname
- hostname pcrfclient01
```

Create CPS VMs using Heat

To create the CPS VMs using OpenStack Heat, you must first create an environment file and a Heat template containing information for your deployment.

These files include information about the ISO, base image, availability zones, management IPs, and volumes. Modify the sample files provided below with information for your deployment.

- Sample Heat Environment File, on page 7
- Sample Heat Template File, on page 8

After populating these files, continue with Create Heat Stack, on page 24.

Sample Heat Environment File

Note Update the network/vlan names, internal and management IPs, VIPs, and volumes for your environment.

`az-1`, `az-2` shown in the following sample are for example purposes only. Update these for your environment accordingly.

Also update the heat template (hot-cps.yaml) with your availability zone variables (for example: `cps_az_1`, `cps_az_2`) after updating this heat environment file.

```
# cat hot-cps.env
# This is an example environment file parameters:
  cps_iso_image_name: CPS_9.0.0.release.iso
  base_vm_image_name: CPS_9.0.0_Base.release
  cps_az_1: az-1
  cps_az_2: az-2

  internal_net_name: internal
  internal_net_cidr: 172.16.2.0/24

  management_net_name: management
  management_net_cidr: 172.18.11.0/24
  management_net_gateway: 172.18.11.1

  gx_net_name: gx
  gx_net_cidr: 192.168.2.0/24

  cluman_flavor_name: cluman
  cluman_internal_ip: 172.16.2.19
  cluman_management_ip: 172.18.11.151

  lb_internal_vip: 172.16.2.200
  lb_management_vip: 172.18.11.156
  lb_gx_vip: 192.168.2.200
  lb01_flavor_name: lb01
  lb01_internal_ip: 172.16.2.201
  lb01_management_ip: 172.18.11.154
  lb01_gx_ip: 192.168.2.201
  lb02_flavor_name: lb02
  lb02_internal_ip: 172.16.2.202
  lb02_management_ip: 172.18.11.155
  lb02_gx_ip: 192.168.2.202

  pcrfclient01_flavor_name: pcrfclient01
  pcrfclient01_internal_ip: 172.16.2.20
  pcrfclient01_management_ip: 172.18.11.152
  pcrfclient02_flavor_name: pcrfclient02
  pcrfclient02_internal_ip: 172.16.2.21
  pcrfclient02_management_ip: 172.18.11.153

  qns01_internal_ip: 172.16.2.24
  qns02_internal_ip: 172.16.2.25
  qns03_internal_ip: 172.16.2.26
  qns04_internal_ip: 172.16.2.27

  sessionmgr01_internal_ip: 172.16.2.22
  sessionmgr01_management_ip: 172.18.11.157
```

```
  sessionmgr02_internal_ip: 172.16.2.23
  sessionmgr02_management_ip: 172.18.11.158

  mongo01_volume_id: "54789405-f683-401b-8194-c354d8937ecb"
  mongo02_volume_id: "9694ab92-8ddd-407e-8520-8b0280f5db03"
  svn01_volume_id: "5b6d7263-40d1-4748-b45c-d1af698d71f7"
  svn02_volume_id: "b501f834-eff9-4044-90c3-a24378f3734d"
  cps_iso_volume_id: "ef52f944-411b-42b1-b86a-500950f5b398"
```

Sample Heat Template File

Note

- Update the following sample heat template according to your environment, such as to add more VMs, networks to the VMs, and so on.
- For more information on MOG/PATS, contact your Cisco Technical Representative.
- Currently, eSCEF is an EFT product and is for Lab Use Only. This means it is not supported by Cisco TAC and cannot be used in a production network. The features in the EFT are subject to change at the sole discretion of Cisco.

```
#cat hot-cps.yaml
heat_template_version: 2014-10-16

description: A minimal CPS deployment for big bang deployment

parameters:
#=========================
# Global Parameters
#=========================
  base_vm_image_name:
    type: string
    label: base vm image name
    description: name of the base vm as imported into glance
  cps_iso_image_name:
    type: string
    label: cps iso image name
    description: name of the cps iso as imported into glance
  cps_install_type:
    type: string
    label: cps installation type (mobile|mog|pats|arbiter|andsf|escef)
    description: cps installation type (mobile|mog|pats|arbiter|andsf|escef)
    default: mobile
  cps_az_1:
    type: string
    label: first availability zone
    description: az for "first half" of cluster
    default: nova
  cps_az_2:
    type: string
    label: second availability zone
    description: az for "second half" of cluster
    default: nova

#=========================
# Network Parameters
#=========================
  internal_net_name:
    type: string
```

```
    label: internal network name
    description: name of the internal network
  internal_net_cidr:
    type: string
    label: cps internal cidr
    description: cidr of internal subnet

  management_net_name:
    type: string
    label: management network name
    description: name of the management network
  management_net_cidr:
    type: string
    label: cps management cidr
    description: cidr of management subnet
  management_net_gateway:
    type: string
    label: management network gateway
    description: gateway on management network
    default: ""

  gx_net_name:
    type: string
    label: gx network name
    description: name of the gx network
  gx_net_cidr:
    type: string
    label: cps gx cidr
    description: cidr of gx subnet
  gx_net_gateway:
    type: string
    label: gx network gateway
    description: gateway on gx network
    default: ""

  cps_secgroup_name:
    type: string
    label: cps secgroup name
    description: name of cps security group
    default: cps_secgroup

#=========================
# Volume Parameters
#=========================
  mongo01_volume_id:
    type: string
    label: mongo01 volume id
    description: uuid of the mongo01 volume

  mongo02_volume_id:
    type: string
    label: mongo02 volume id
    description: uuid of the mongo02 volume

  svn01_volume_id:
    type: string
    label: svn01 volume id
    description: uuid of the svn01 volume

  svn02_volume_id:
    type: string
    label: svn02 volume id
    description: uuid of the svn02 volume
```

```
  cps_iso_volume_id:
    type: string
    label: cps iso volume id
    description: uuid of the cps iso volume

#=========================
# Instance Parameters
#=========================
  cluman_flavor_name:
    type: string
    label: cluman flavor name
    description: flavor cluman vm will use
    default: cluman
  cluman_internal_ip:
    type: string
    label: internal ip of cluster manager
    description: internal ip of cluster manager
  cluman_management_ip:
    type: string
    label: management ip of cluster manager
    description: management ip of cluster manager

  lb_internal_vip:
    type: string
    label: internal vip of load balancer
    description: internal vip of load balancer
  lb_management_vip:
    type: string
    label: management vip of load balancer
    description: management vip of load balancer
  lb_gx_vip:
    type: string
    label: gx ip of load balancer
    description: gx vip of load balancer
  lb01_flavor_name:
    type: string
    label: lb01 flavor name
    description: flavor lb01 vms will use
    default: lb01
  lb01_internal_ip:
    type: string
    label: internal ip of load balancer
    description: internal ip of load balancer
  lb01_management_ip:
    type: string
    label: management ip of load balancer
    description: management ip of load balancer
  lb01_gx_ip:
    type: string
    label: gx ip of load balancer
    description: gx ip of load balancer
  lb02_flavor_name:
    type: string
    label: lb02 flavor name
    description: flavor lb02 vms will use
    default: lb02
  lb02_internal_ip:
    type: string
    label: internal ip of load balancer
    description: internal ip of load balancer
  lb02_management_ip:
    type: string
    label: management ip of load balancer
    description: management ip of load balancer
```

```
lb02_gx_ip:
  type: string
  label: gx ip of load balancer
  description: gx ip of load balancer

pcrfclient01_flavor_name:
  type: string
  label: pcrfclient01 flavor name
  description: flavor pcrfclient01 vm will use
  default: pcrfclient01
pcrfclient01_internal_ip:
  type: string
  label: internal ip of pcrfclient01
  description: internal ip of pcrfclient01
pcrfclient01_management_ip:
  type: string
  label: management ip of pcrfclient01
  description: management ip of pcrfclient01
pcrfclient02_flavor_name:
  type: string
  label: pcrfclient02 flavor name
  description: flavor pcrfclient02 vm will use
  default: pcrfclient02
pcrfclient02_internal_ip:
  type: string
  label: internal ip of pcrfclient02
  description: internal ip of pcrfclient02
pcrfclient02_management_ip:
  type: string
  label: management ip of pcrfclient02
  description: management ip of pcrfclient02

qns_flavor_name:
  type: string
  label: qns flavor name
  description: flavor qns vms will use
  default: qps
qns01_internal_ip:
  type: string
  label: internal ip of qns01
  description: internal ip of qns01
qns02_internal_ip:
  type: string
  label: internal ip of qns02
  description: internal ip of qns02
qns03_internal_ip:
  type: string
  label: internal ip of qns03
  description: internal ip of qns03
qns04_internal_ip:
  type: string
  label: internal ip of qns04
  description: internal ip of qns04


sessionmgr_flavor_name:
  type: string
  label: sessionmgr flavor name
  description: flavor sessionmgr vms will use
  default: sm
sessionmgr01_internal_ip:
  type: string
  label: internal ip of sessionmgr01
  description: internal ip of sessionmgr01
```

```
  sessionmgr01_management_ip:
    type: string
    label: management ip of sessionmgr01
    description: management ip of sessionmgr01
  sessionmgr02_internal_ip:
    type: string
    label: internal ip of sessionmgr02
    description: internal ip of sessionmgr02
  sessionmgr02_management_ip:
    type: string
    label: management ip of sessionmgr02
    description: management ip of sessionmgr02

resources:
#=========================
# Instances
#=========================

  cluman:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: cluman_flavor_name }
      networks:
        - port: { get_resource: cluman_internal_port }
        - port: { get_resource: cluman_management_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: cps_iso_volume_id }
      user_data_format: RAW
      user_data: { get_resource: cluman_config }
  cluman_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_internal_ip }}]
  cluman_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: cluman_management_ip }}]
  cluman_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
            permissions: "0644"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            permissions: "0644"
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: cluman_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            permissions: "0644"
            content:
```

```
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: cluman_management_ip }
                $gateway: { get_param: management_net_gateway }
          - path: /root/.autoinstall.sh
            permissions: "0755"
            content:
              str_replace:
                template: |
                  #!/bin/bash
                  if [[ -d /mnt/iso ]] && [[ -f /mnt/iso/install.sh ]]; then
                  /mnt/iso/install.sh << EOF
                  $install_type
                  y
                  1
                  EOF
                  fi
                params:
                  $install_type: { get_param: cps_install_type }
        mounts:
          - [ /dev/vdb, /mnt/iso, iso9660, "auto,ro", 0, 0 ]
        runcmd:
          - str_replace:
              template: echo $ip installer >> /etc/hosts
              params:
                $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - echo HOSTNAME=cluman >> /etc/sysconfig/network
          - echo cluman > /etc/hostname
          - hostname cluman
          - /root/.autoinstall.sh

  lb01:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: lb01_flavor_name }
      networks:
        - port: { get_resource: lb01_internal_port }
        - port: { get_resource: lb01_management_port }
        - port: { get_resource: lb01_gx_port }
      user_data_format: RAW
      user_data: { get_resource: lb01_config }
  lb01_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
```

```
    fixed_ips: [{ ip_address: { get_param: lb01_internal_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_internal_vip }
lb01_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_management_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_management_vip }
lb01_gx_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: gx_net_name }
    fixed_ips: [{ ip_address: { get_param: lb01_gx_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_gx_vip }
lb01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=lb01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: lb01_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: lb01_management_ip }
                $gateway: { get_param: management_net_gateway }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth2
          content:
            str_replace:
              template: |
                DEVICE=eth2
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: lb01_gx_ip }
                $gateway: { get_param: gx_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
```

```
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
            params:
              $cidr: { get_param: gx_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - ifdown eth2 && ifup eth2
        - echo HOSTNAME=lb01 >> /etc/sysconfig/network
        - echo lb01 > /etc/hostname
        - hostname lb01

lb02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: lb02_flavor_name }
    networks:
      - port: { get_resource: lb02_internal_port }
      - port: { get_resource: lb02_management_port }
      - port: { get_resource: lb02_gx_port }
    user_data_format: RAW
    user_data: { get_resource: lb02_config }
lb02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_internal_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_internal_vip }
lb02_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_management_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_management_vip }
lb02_gx_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: gx_net_name }
    fixed_ips: [{ ip_address: { get_param: lb02_gx_ip }}]
    allowed_address_pairs:
      - ip_address: { get_param: lb_gx_vip }
lb02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=lb02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
```

```
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: lb02_internal_ip }
              - path: /etc/sysconfig/network-scripts/ifcfg-eth1
                content:
                  str_replace:
                    template: |
                      DEVICE=eth1
                      BOOTPROTO=none
                      NM_CONTROLLED=no
                      IPADDR=$ip
                      GATEWAY=$gateway
                    params:
                      $ip: { get_param: lb02_management_ip }
                      $gateway: { get_param: management_net_gateway }
              - path: /etc/sysconfig/network-scripts/ifcfg-eth2
                content:
                  str_replace:
                    template: |
                      DEVICE=eth2
                      BOOTPROTO=none
                      NM_CONTROLLED=no
                      IPADDR=$ip
                      GATEWAY=$gateway
                    params:
                      $ip: { get_param: lb02_gx_ip }
                      $gateway: { get_param: gx_net_gateway }
            runcmd:
              - str_replace:
                  template: echo $ip installer >> /etc/hosts
                  params:
                    $ip: { get_param: cluman_internal_ip }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
                  params:
                    $cidr: { get_param: internal_net_cidr }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
                  params:
                    $cidr: { get_param: management_net_cidr }
              - str_replace:
                  template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth2
                  params:
                    $cidr: { get_param: gx_net_cidr }
              - ifdown eth0 && ifup eth0
              - ifdown eth1 && ifup eth1
              - ifdown eth2 && ifup eth2
              - echo HOSTNAME=lb02 >> /etc/sysconfig/network
              - echo lb02 > /etc/hostname
              - hostname lb02

      pcrfclient01:
        type: OS::Nova::Server
        properties:
          availability_zone: { get_param: cps_az_1 }
          config_drive: "True"
          image: { get_param: base_vm_image_name }
          flavor: { get_param: pcrfclient01_flavor_name }
          networks:
            - port: { get_resource: pcrfclient01_internal_port }
```

```
      - port: { get_resource: pcrfclient01_management_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: svn01_volume_id }
    user_data_format: RAW
    user_data: { get_resource: pcrfclient01_config }
pcrfclient01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient01_internal_ip }}]
pcrfclient01_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient01_management_ip }}]
pcrfclient01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=pcrfclient01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: pcrfclient01_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: pcrfclient01_management_ip }
                $gateway: { get_param: management_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=pcrfclient01 >> /etc/sysconfig/network
        - echo pcrfclient01 > /etc/hostname
        - hostname pcrfclient01
```

```
pcrfclient02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: pcrfclient02_flavor_name }
    networks:
      - port: { get_resource: pcrfclient02_internal_port }
      - port: { get_resource: pcrfclient02_management_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: svn02_volume_id }
    user_data_format: RAW
    user_data: { get_resource: pcrfclient02_config }
pcrfclient02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient02_internal_ip }}]
pcrfclient02_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: pcrfclient02_management_ip }}]
pcrfclient02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=pcrfclient02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: pcrfclient02_internal_ip }
        - path: /etc/sysconfig/network-scripts/ifcfg-eth1
          content:
            str_replace:
              template: |
                DEVICE=eth1
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
                GATEWAY=$gateway
              params:
                $ip: { get_param: pcrfclient02_management_ip }
                $gateway: { get_param: management_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
```

```
              $cidr: { get_param: internal_net_cidr }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
              params:
                $cidr: { get_param: management_net_cidr }
          - ifdown eth0 && ifup eth0
          - ifdown eth1 && ifup eth1
          - echo HOSTNAME=pcrfclient02 >> /etc/sysconfig/network
          - echo pcrfclient01 > /etc/hostname
          - hostname pcrfclient02

qns01:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns01_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns01_config }
qns01_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns01_internal_ip }}]
qns01_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns01\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns01_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns01 >> /etc/sysconfig/network
        - echo qns01 > /etc/hostname
        - hostname qns01

qns02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_1 }
    config_drive: "True"
```

```
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns02_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns02_config }
qns02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns02_internal_ip }}]
qns02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns02_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns02 >> /etc/sysconfig/network
        - echo qns02 > /etc/hostname
        - hostname qns02

qns03:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns03_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns03_config }
qns03_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns03_internal_ip }}]
qns03_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
```

```
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns03\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns03_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - ifdown eth0 && ifup eth0
        - echo HOSTNAME=qns03 >> /etc/sysconfig/network
        - echo qns03 > /etc/hostname
        - hostname qns03

qns04:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: qns_flavor_name }
    networks:
      - port: { get_resource: qns04_internal_port }
    user_data_format: RAW
    user_data: { get_resource: qns04_config }
qns04_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: qns04_internal_ip }}]
qns04_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=qns04\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: qns04_internal_ip }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
```

```
            params:
              $ip: { get_param: cluman_internal_ip }
          - str_replace:
              template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
              params:
                $cidr: { get_param: internal_net_cidr }
          - ifdown eth0 && ifup eth0
          - echo HOSTNAME=qns04 >> /etc/sysconfig/network
          - echo qns04 > /etc/hostname
          - hostname qns04

  sessionmgr01:
    type: OS::Nova::Server
    properties:
      availability_zone: { get_param: cps_az_1 }
      config_drive: "True"
      image: { get_param: base_vm_image_name }
      flavor: { get_param: sessionmgr_flavor_name }
      networks:
        - port: { get_resource: sessionmgr01_internal_port }
        - port: { get_resource: sessionmgr01_management_port }
      block_device_mapping:
        - device_name: vdb
          volume_id: { get_param: mongo01_volume_id }
      user_data_format: RAW
      user_data: { get_resource: sessionmgr01_config }
  sessionmgr01_internal_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: internal_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_internal_ip }}]
  sessionmgr01_management_port:
    type: OS::Neutron::Port
    properties:
      network: { get_param: management_net_name }
      fixed_ips: [{ ip_address: { get_param: sessionmgr01_management_ip }}]
  sessionmgr01_config:
    type: OS::Heat::CloudConfig
    properties:
      cloud_config:
        write_files:
          - path: /var/lib/cloud/instance/payload/launch-params
          - path: /etc/broadhop.profile
            content: "NODE_TYPE=sessionmgr01\n"
          - path: /etc/sysconfig/network-scripts/ifcfg-eth0
            content:
              str_replace:
                template: |
                  DEVICE=eth0
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                params:
                  $ip: { get_param: sessionmgr01_internal_ip }
          - path: /etc/sysconfig/network-scripts/ifcfg-eth1
            content:
              str_replace:
                template: |
                  DEVICE=eth1
                  BOOTPROTO=none
                  NM_CONTROLLED=no
                  IPADDR=$ip
                  GATEWAY=$gateway
                params:
```

```
                $ip: { get_param: sessionmgr01_management_ip }
                $gateway: { get_param: management_net_gateway }
      runcmd:
        - str_replace:
            template: echo $ip installer >> /etc/hosts
            params:
              $ip: { get_param: cluman_internal_ip }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
            params:
              $cidr: { get_param: internal_net_cidr }
        - str_replace:
            template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
            params:
              $cidr: { get_param: management_net_cidr }
        - ifdown eth0 && ifup eth0
        - ifdown eth1 && ifup eth1
        - echo HOSTNAME=sessionmgr01 >> /etc/sysconfig/network
        - echo sessionmgr01 > /etc/hostname
        - hostname sessionmgr01

sessionmgr02:
  type: OS::Nova::Server
  properties:
    availability_zone: { get_param: cps_az_2 }
    config_drive: "True"
    image: { get_param: base_vm_image_name }
    flavor: { get_param: sessionmgr_flavor_name }
    networks:
      - port: { get_resource: sessionmgr02_internal_port }
      - port: { get_resource: sessionmgr02_management_port }
    block_device_mapping:
      - device_name: vdb
        volume_id: { get_param: mongo02_volume_id }
    user_data_format: RAW
    user_data: { get_resource: sessionmgr02_config }
sessionmgr02_internal_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: internal_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr02_internal_ip }}]
sessionmgr02_management_port:
  type: OS::Neutron::Port
  properties:
    network: { get_param: management_net_name }
    fixed_ips: [{ ip_address: { get_param: sessionmgr02_management_ip }}]
sessionmgr02_config:
  type: OS::Heat::CloudConfig
  properties:
    cloud_config:
      write_files:
        - path: /var/lib/cloud/instance/payload/launch-params
        - path: /etc/broadhop.profile
          content: "NODE_TYPE=sessionmgr02\n"
        - path: /etc/sysconfig/network-scripts/ifcfg-eth0
          content:
            str_replace:
              template: |
                DEVICE=eth0
                BOOTPROTO=none
                NM_CONTROLLED=no
                IPADDR=$ip
              params:
                $ip: { get_param: sessionmgr02_internal_ip }
```

```
  - path: /etc/sysconfig/network-scripts/ifcfg-eth1
    content:
      str_replace:
        template: |
          DEVICE=eth1
          BOOTPROTO=none
          NM_CONTROLLED=no
          IPADDR=$ip
          GATEWAY=$gateway
        params:
          $ip: { get_param: sessionmgr02_management_ip }
          $gateway: { get_param: management_net_gateway }
runcmd:
  - str_replace:
      template: echo $ip installer >> /etc/hosts
      params:
        $ip: { get_param: cluman_internal_ip }
  - str_replace:
      template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth0
      params:
        $cidr: { get_param: internal_net_cidr }
  - str_replace:
      template: ipcalc -m $cidr >> /etc/sysconfig/network-scripts/ifcfg-eth1
      params:
        $cidr: { get_param: management_net_cidr }
  - ifdown eth0 && ifup eth0
  - ifdown eth1 && ifup eth1
  - echo HOSTNAME=sessionmgr02 >> /etc/sysconfig/network
  - echo sessionmgr02 > /etc/hostname
  - hostname sessionmgr02
```

Create Heat Stack

Before beginning, verify you have populated your information in the environment (.env) file and heat template (.yaml) file and loaded both files on the control node.

Step 1 Run the following command on control node at the location where your environment and heat template files are located:

```
source ~/keystonerc_core
```

Step 2 Add/assign the heat stack owner to core tenant user:

```
openstack role add --project core --user core admin
```

Step 3 Verify that no existing CPS stack is present:

```
[root@os8-control ~(keystone_core)]# heat stack-list

+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
+--------------------------------------+------------+-----------------+----------------------+
```

Step 4 Create the stack using the heat template (hot-cps.yaml) and environment file (hot-cps.env) you populated earlier.

```
[root@os8-control mbuild(keystone_core)]# heat stack-create --environment-file hot-cps.env
--template-file hot-cps.yaml cps
+--------------------------------------+------------+--------------------+----------------------+
| id                                   | stack_name | stack_status       | creation_time        |
+--------------------------------------+------------+--------------------+----------------------+
```

```
| 3f1ab6c2-673d-47b3-ae01-8946cac9e9e9 | cps        | CREATE_IN_PROGRESS | 2016-03-03T16:58:53Z |
+--------------------------------------+------------+--------------------+----------------------+
```

Step 5 Check the status using the `heat stack-list` command:

```
[root@os8-control mbuild(keystone_core)]# heat stack-list
+--------------------------------------+------------+-----------------+----------------------+
| id                                   | stack_name | stack_status    | creation_time        |
+--------------------------------------+------------+-----------------+----------------------+
| 3f1ab6c2-673d-47b3-ae01-8946cac9e9e9 | cps        | CREATE_COMPLETE | 2016-01-19T16:58:53Z |
+--------------------------------------+------------+-----------------+----------------------+
```

`CREATE_COMPLETE` will be reported when the heat stack is finished.

Step 6 Wait approximately 10 minutes for the Cluster Manager VM to be deployed, then check the readiness status of the Cluster Manager VM using the following API:

GET http://<*Cluster Manager IP*>:8458/api/system/status/cluman

Refer to /api/system/status/cluman for more information.

When this API responds that the Cluster Manager VM is in a ready state (`"status": "ready"`), continue with Deploy CPS, on page 25.

Refer also to the `/var/log/cloud-init-output.log` on the Cluster Manager VM for deployment details.

Deploy CPS

The following steps outline how to create a consolidated CPS configuration file and use the CPS platform orchestration APIs to deploy the CPS VMs on OpenStack:

Step 1 Create a consolidated CPS configuration file. This file contains all the information necessary to deploy VMs in the CPS cluster, including a valid CPS license key. Contact your Cisco representative to receive the CPS license key for your deployment.

Note Cisco Smart Licensing is supported for CPS 10.0.0 and later releases. For information about what Smart Licensing is and how to enable it for CPS, refer to *CPS Operations Guide*.

a) Refer to Sample YAML Configuration File - HA Setup for a sample CPS configuration to use as a template.
b) Refer to Configuration Parameters - HA System for a description of all parameters within this file.

Important Verify that all VM IP addresses and host names are configured properly in the YAML and Heat template files. You cannot modify the IP addresses or host names manually on the VMs (excluding Cluster Manager) after deploying the VMs, and CPS does not support modification of IP addresses or host names of deployed VMs.

Step 2 Load the consolidated configuration file you created in Step 1, on page 25 using the following API:

POST http://<*Cluster Manager IP*>:8458/api/system/config/

For example:

```
curl -v -X POST --data-binary @CPS_config_yaml.txt -H "Content-type: application/yaml"
http://x.x.x.x:8458/api/system/config/
```

Refer to /api/system/config/ for more information.

Step 3 (Optional) To confirm the configuration was loaded properly onto the Cluster Manager VM, perform a GET with the same API:

GET http://<*Cluster Manager IP*>:8458/api/system/config/

Step 4 Apply the configuration using the following API:

POST http://<*Cluster Manager IP*>:8458/api/system/config/action/apply

For example:

```
curl -v -X POST -H  "Content-type: application/json" http://x.x.x.x:8458/api/system/config/action/apply
```

Refer to /api/system/config/ for more information.

This API applies the CPS configuration file, triggers the Cluster Manager VM to deploy and bring up all CPS VMs, and performs all post-installation steps.

Important The VMs are rebooted in rescue mode for the first time for CentOS to adjust disk/hardware to the new version. Subsequent reboots if necessary is a normal operation.

Step 5 Run `change_passwd.sh` script on Cluster Manager to change the password of root user across the system.

For more information, refer to *Update Default Credentials* in *CPS Installation Guide for VMware*.

What to do next

To enable the feature **Disable Root SSH Login**, check whether there exists a user with uid 1000 on Cluster Manager.

Use the following command to check there exists a user with uid 1000:

```
cat /etc/passwd | grep x:1000
```

If a user with uid 1000 exists on the Cluster Manager, change the uid on the Cluster Manager by executing the following command:

```
usermod -u <new-uid> <user-name-with-uid-as-1000>
```

This is done because the feature **Disable Root SSH Login** creates a user with uid 1000.

Validate CPS Deployment

Step 1 To monitor the status of the deployment, use the following API:

GET http://<*Cluster Manager IP*>:8458/api/system/config/status

Refer to /api/system/config/status for more information.

Step 2 After the deployment has completed, verify the readiness of the entire CPS cluster using the following API:

GET http://<*Cluster Manager IP*>:8458/api/system/status/cps

Refer to /api/system/status/cps for more information.

Step 3 Connect to the Cluster Manager and issue the following command to run a set of diagnostics and display the current state of the system.

```
/var/qps/bin/diag/diagnostics.sh
```

What to do next

Important After the validation is complete, take a backup of the Cluster Manager configuration. For more information on taking the backup, refer to *CPS Backup and Restore Guide*. In case the Cluster Manager gets corrupted this backup can be used to recover the Cluster Manager.

Troubleshooting

- CPS clusters deployed using the orchestration APIs report the following licensing errors in `/var/log/broadhop/qns.log` on the OAM (pcrfclient) VMs:

  ```
  [LicenseManagerTimer] ERROR c.b.licensing.impl.LicenseManager - Unable to load the
  license file.  Server is not licensed!
  ```

  This error can be ignored.

SR-IOV Support

CPS supports single root I/O virtualization (SR-IOV) on Intel NIC adapters.

CPS also supports bonding of SR-IOV sub-interfaces for seamless traffic switchover.

The Intel SR-IOV implementation includes anti-spoofing support that will not allow MAC addresses other than the one configured in the VF to communicate. As a result, the active failover mac policy is used.

To support seamless failover of interfaces, the VLAN interfaces should be created directly on top of the VF interfaces (for example, eth0.123 and eth1.123) and then those interfaces are bonded. If VLAN interfaces are created on top of a bond, their MAC address will not follow the bonds when a failover occurs and the old MAC will be used for the new active interface.

The following sample configuration shows the bonding of two interfaces using a single IP address:

```
[root@qns0x ~]# cat /proc/net/bonding/bond0310
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: None
Currently Active Slave: eth1.310
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1.310
MII Status: up
Speed: 10000 Mbps
Duplex: full
```

```
Link Failure Count: 1
Permanent HW addr: fa:16:3e:aa:a5:c8
Slave queue ID: 0

Slave Interface: eth2.310
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:26:e3:9e
Slave queue ID: 0
[root@qns02 ~]# cat /proc/net/bonding/bond0736
Ethernet Channel Bonding Driver: v3.7.1 (April 27, 2011)

Bonding Mode: fault-tolerance (active-backup) (fail_over_mac active)
Primary Slave: None
Currently Active Slave: eth1.736
MII Status: up
MII Polling Interval (ms): 100
Up Delay (ms): 0
Down Delay (ms): 0

Slave Interface: eth1.736
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:aa:a5:c8
Slave queue ID: 0

Slave Interface: eth2.736
MII Status: up
Speed: 10000 Mbps
Duplex: full
Link Failure Count: 1
Permanent HW addr: fa:16:3e:26:e3:9e
Slave queue ID: 0

[root@qns0x ~]# more /etc/sysconfig/network-scripts/ifcfg-*
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-bond0310
::::::::::::::
DEVICE=bond0310
BONDING_OPTS="mode=active-backup miimon=100 fail_over_mac=1"
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV6INIT=no
IPADDR=172.16.255.11
NETMASK=255.255.255.192
NETWORK=172.16.255.0
IPV4_FAILURE_FATAL=no
IPV6INIT=no
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
ONBOOT=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-bond0736
```

```
::::::::::::::
DEVICE=bond0736
BONDING_OPTS="mode=active-backup miimon=100 fail_over_mac=1"
TYPE=Bond
BONDING_MASTER=yes
BOOTPROTO=none
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV6INIT=yes
IPV6ADDR=fd00:4888:1000:30c2::23/64
IPV6_DEFAULTGW=fd00:4888:1000:30c2::1
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=no
IPV6_DEFROUTE=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
IPV6_FAILURE_FATAL=no
ONBOOT=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth0
::::::::::::::
DEVICE=eth0
ONBOOT=yes
BOOTPROTO=none
IPADDR=192.168.66.34
NETMASK=255.255.255.0
NETWORK=192.168.66.0
IPV6INIT=yes
IPV6ADDR=fd00:4888:1000:f000::aab1/64
IPV6_DEFAULTGW=fd00:4888:1000:f000::1
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1
::::::::::::::
DEVICE=eth1
TYPE=Ethernet
ONBOOT=yes
BOOTPROTO=none
USRCTL=no
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1.310
::::::::::::::
DEVICE=eth1.310
ONBOOT=yes
MASTER=bond0310
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth1.736
::::::::::::::
DEVICE=eth1.736
ONBOOT=yes
MASTER=bond0736
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2
::::::::::::::
DEVICE=eth2
```

```
ONBOOT=yes
BOOTPROTO=none
USRCTL=no
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2.310
::::::::::::::
DEVICE=eth2.310
ONBOOT=yes
MASTER=bond0310
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
::::::::::::::
/etc/sysconfig/network-scripts/ifcfg-eth2.736
::::::::::::::
DEVICE=eth2.736
ONBOOT=yes
MASTER=bond0736
BOOTPROTO=none
USRCTL=no
SLAVE=yes
VLAN=yes
```

Consistent Network Device Naming

CPS instances require that network interfaces be assigned IP addresses statically. The names of network interfaces (eth0, eth1, and so on) are assumed to reflect network interfaces representing neutron ports passed to OpenStack nova-boot or heat template in that order. In this case, eth0 is assumed to reflect the first neutron port, eth1 the second, and so on.

For CPS deployments on OpenStack which use SR-IOV, often two or more network drivers are used. When more than one network driver is used, network interface names can become unpredictable and can change based on the order in which the network drivers are loaded into the kernel.

The following section describes how to map a network interface for a given network drivers type to its correct expected name in the guest OS.

Requirements:

- Correct IP address assignment requires that network names used in the network interfaces file must match the name of the network interface in the guest OS.
- The order of neutron ports of a given type (non-SR-IOV or SR-IOV) in nova-boot or heat template directly maps to the order of the PCI device slot of the associated network interfaces in the guest OS.
- The mapping between the network interface of a given network driver type and network driver name are passed during the creation of an instance through the cloud-init configuration.

The expected network interface name configuration is passed into CPS instance's guest OS using a YAML format configuration file located at: `/var/lib/cloud/instance/payload/ifrename.yaml`.

The file should have a section for each driver type and list the interfaces for that driver type with the following information:

- Rank order (0, 1, 2…) for the interface among other interfaces of the same driver type, as is specified in the nova boot command/heat template
- Expected name of the interface (eth0, eth1, eth2 etc.)

For example:

```
- path: /var/lib/cloud/instance/payload/ifrename.yaml
  encoding: ascii
  owner: root:root
  permissions: '0644'
  content:  |
    ---
      -  virtio_net
        0 : eth0
      -  ixgbevf:
        0 : eth1
        1 : eth2
```

Driver names for SR-IOV ports can be determined by checking the interface card vendor documentation. For regular virtio ports, the driver name is 'virtio_net'.

This `ifrename.yaml` file must be added in the existing write_files: section of cloud-init configurations for each CPS VM.

The configuration file above instructs cloud-init to create a file `ifrename.yaml` at `/var/lib/cloud/instance/payload`, owned by root, with permissions of 644 and contents as mentioned in "content:" section. In this example:

- the first SR-IOV neutron port (managed by 'ixgbevf' driver) is mapped to to eth1
- the second SR-IOV port (managed by 'ixgbevf' driver) is mapped to eth2
- the only non-SR-IOV port (managed by 'virtio-net' driver) to eth0.

Regardless of the order in which neutron ports are passed, or order in which network drivers are loaded, this configuration file specifies which network interface name should go to which network interface.

Enable Custom Puppet to Configure Deployment

Some customers may need to customize the configuration for their deployment. When customizing the CPS configuration, it is important to make the customization in a way that does not impact the normal behavior for VM deployment and redeployment, upgrades/migration, and rollbacks.

For this reason, customizations should be placed in the `/etc/puppet/env_config` directory. Files within this directory are given special treatment for VM deployment, upgrade, migrations, and rollback operations.

Note If system configurations are manually changed in the VM itself after the VM has been deployed, these configurations will be overridden if that VM is redeployed.

The following section describes the steps necessary to make changes to the puppet installer.

Customizations of the CPS deployment are dependent on the requirements of the change. Examples of customizations include:

- deploying a specific facility on a node (VM)
- overriding a default configuration.

To explain the process, let us consider that we modify all VMs built from an installer, so we use the Policy Server (QNS) node definition.

For the above mentioned example, add custom routes via the examples42-network Puppet module. (For more information on the module, refer to https://forge.puppetlabs.com/example42/network).

Step 1 Make sure that the proper paths are available:

```
mkdir -p /etc/puppet/env_config/nodes
```

Step 2 Install the necessary Puppet module. For example:

```
puppet module install \
--modulepath=/etc/puppet/env_config/modules:/etc/puppet/modules \
example42-network
Notice: Preparing to install into /etc/puppet/env_config/modules ...
Notice: Downloading from https://forge.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppet/env_config/modules
example42-network (v3.1.13)
```

Note For more information on installing and updating Puppet modules, refer to https://docs.puppetlabs.com/puppet/latest/reference/modules_installing.html.

Step 3 Copy the existing node definition into the env_config nodes:

```
cp /etc/puppet/modules/qps/nodes/qps.yaml \

/etc/puppet/env_config/nodes
```

Step 4 Add a reference to your custom Puppet manifest:

```
echo ' custom::static_routes:' >> \

/etc/puppet/env_config/nodes/qps.yaml
```

Step 5 Create your new manifest for static routes:

```
cat
 >/etc/puppet/env_config/modules/custom/manifests/static_routes.pp <<EOF class custom::static_routes
 {
   network::route {'eth0':
     ipaddress => ['192.168.1.0',],
     netmask   => ['255.255.255.0',],
     gateway   => ['10.105.94.1',],
   }
}
EOF
```

Step 6 Validate the syntax of your newly created puppet script(s):

```
puppet parser validate

/etc/puppet/env_config/modules/custom/manifests/static_routes.pp
```

Step 7 Rebuild your Environment Configuration:

```
/var/qps/install/current/scripts/build/build_env_config.sh
```

Step 8 Reinitialize your environment:

```
/var/qps/install/current/scripts/upgrade/reinit.sh
```

At this point your new manifest is applied across the deployment. For more details, refer to the installer image in the `/etc/puppet/env_config/README`.

What to do next

It is recommended that version control is used to track changes to these Puppet customizations.

For example, to use 'git', perform the following steps:

1. Initialize the directory as a repository:

   ```
   # git init
   ```

   Initialized empty Git repository in `/var/qps/env_config/.git/`.

2. Add everything:

   ```
   # git add .
   ```

3. Commit your initial check-in:

   ```
   # git commit -m 'initial commit of env_config'
   ```

4. If you are making more changes and customizations, make sure you create new revisions for those:

   ```
   # git add .

   # git commit -m 'updated static routes'
   ```

HTTPS Support for Orchestration API

Installation

By default, the Orchestration API service starts with the HTTP mode on Cluster Manager.

You can change the mode to start with HTTPS self-signed certificate by setting the `api_https=one_way_ssl` facter value in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file in Cluster Manager. This ensures that the API server starts by using the pre-loaded self-signed SSL certificates.

Important You cannot upload certificates using the API.

To configure the Orchestration API server to start with the HTTPS self-signed certificate mode, make the following changes to the Heat template. These changes create the `/etc/facter/facts.d/cluman_facts.yaml` file and also set the puppet facter value to `api_https=one_way_ssl` in the configuration file in Cluster Manager.

```
cluman_api_name:
 type: string
 label: cluman orch api
 description: cluman orch
 default: one_way_ssl
# This will set the default value to one_way_ssl
```

```
- path: /etc/facter/facts.d/cluman_facts.yaml
 permissions: "0755"
 content:
 str_replace:
 template: |
   api_https: $kval
 params:
   $kval: { get_param: cluman_api_name }
```

Sample YAML configuration to run the Orchestration API server:

- Using self-signed certificates (one_way_ssl):

```
cat /etc/facter/facts.d/cluman_facts.yaml
 api_https: one_way_ssl
```

- Using trusted certificates (one_way_ssl):

```
cat /etc/facter/facts.d/cluman_facts.yaml
 api_https: one_way_ssl
 api_keystore_path: /var/certs/keystore.jks
 api_keystore_password: yoursecret
 api_keystore_type: JKS
 api_cert_alias: server-tls
 api_tls_version: TLSv1.2
 api_validate_certs: FALSE
 api_validate_peers: FALSE
```

- Using mutual authentication (two_way_ssl):

```
cat /etc/facter/facts.d/cluman_facts.yaml
 api_https: two_way_ssl
 api_keystore_path: /var/certs/keystore.jks
 api_keystore_password: yoursecret
 api_keystore_type: JKS
 api_cert_alias: server-tls
 api_tls_version: TLSv1.2
 api_truststore_path: /var/certs/truststore.jks
 api_truststore_password: yoursecret
 api_truststore_type: JKS
 api_validate_certs: TRUE
 api_validate_peers: TRUE
 api_enable_crldp: TRUE
```

Note

- For more information on how to add certificates to the keystore or truststore, see Adding Certificates to Keystore and Truststore, on page 36.
- Trusted certificates, keystores, or the truststore should not be located at `/opt/orchestration_api_server/`.
- For a list of the configuration parameters for HTTPS, see Configuration Parameters for HTTPS, on page 37.

After Cluster Manager is deployed, you can reconfigure the API server to run on HTTP (default) or HTTPS mode. The prerequisites to configure the HTTPS mode are as follows:

- For self-signed certificates, set `api_https=one_way_ssl` in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file.
- For trusted certificates:

1. Install the certificates on Cluster Manager.
2. Import the certificates into the keystore and the truststore.
3. Set `api_https` value to `one_way_ssl` or `two_way_ssl` (mutual authentication) in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file.

To apply the configuration run the following **puppet** commands on Cluster Manager. These commands reconfigure Cluster Manager only.

1. `cd /opt/cluman`
2. `CLUMAN_DIR="/opt/cluman";`
3. `puppet apply --logdest /var/log/cluman/puppet-run.log --modulepath=${CLUMAN_DIR}/puppet/modules --config ${CLUMAN_DIR}/puppet/puppet.conf ${CLUMAN_DIR}/puppet/nodes/node_repo.pp`

Note

1. For fresh installation, only HTTP or HTTPS with self-signed certificates mode is allowed.
2. For `one_way_ssl`, the `api_validate_peers` parameter should be set to *FALSE*.
3. In case some parameters are missing in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file:
   - For one way ssl, the Orchestration API server starts by using the self-signed certificates.
   - For two way ssl, the Orchestration API server rolls back to the default HTTP mode.

Upgrade

Upgrade CPS to run the Orchestration API server on HTTP or HTTPS. To change the behavior, configuration parameters must be configured before triggering the upgrade.

Follow the steps below to upgrade CPS:

- For self-signed certificates, set `api_https=one_way_ssl` in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file and then trigger the upgrade.
- For trusted certificates:
  1. Install the certificates on Cluster Manager.
  2. Import the certificates into the keystore and the truststore.
  3. Set `api_https` value to `one_way_ssl` or `two_way_ssl` (mutual authentication) in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file.
  4. Trigger the upgrade.

Note To roll back the configuration to default, that is HTTP mode, do the following:

1. Move the `/etc/facter/facts.d/cluman_facts.yaml` configuration file to the `/root/` folder.
2. Run the following **puppet** commands on Cluster Manager:
   1. `cd /opt/cluman`
   2. `CLUMAN_DIR="/opt/cluman";`
   3. `puppet apply --logdest /var/log/cluman/puppet-run.log --modulepath=${CLUMAN_DIR}/puppet/modules --config ${CLUMAN_DIR}/puppet/puppet.conf ${CLUMAN_DIR}/puppet/nodes/node_repo.pp`

Adding Certificates to Keystore and Truststore

A keystore contains private keys and certificates used by the TLS and SSL servers to authenticate themselves to TLS and SSL clients respectively. Such files are referred to as keystores. When used as a truststore, the file contains certificates of trusted TLS and SSL servers or of certificate authorities. There are no private keys in the truststore.

Note Your trusted certificates and keystores or truststores should not be located at `/opt/orchestration_api_server/`

Step 1 Create the PKCS12 file for key and certificate chains.

```
openssl pkcs12 –export-name <cert name> -n chain.crt –inkey <cert_private_key> - out server.p12
```

For example: `openssl pkcs12 -export -name server-tls -in chain.crt -inkey server.key -out server.p12`

Step 2 Create the Java KeyStore on the server.

```
keytool -importkeystore -destkeystore <keystore_name.jks> -srckeystore server.p12 -srcstoretype pkcs12 -alias server-tls
```

```
keytool -importkeystore -destkeystore keystore.jks -srckeystore server.p12 -srcstoretype pkcs12 -alias server-tls
```

Step 3 Import the root certificate or CA certificate in the truststore.

```
# Import your root certificate into a new trust store and follow the prompts
```

```
keytool -import -alias root -file root.crt -keystore truststore.jks
```

You must remember the keystore password and this needs to be updated in the `/etc/facter/facts.d/cluman_facts.yaml` file.

Configuration Parameters for HTTPS

The following parameters can be defined in the `/etc/facter/facts.d/cluman_facts.yaml` configuration file. This file is loaded only onto the Cluster Manager VM. All parameters and values are case sensitive.

Note Before loading the configuration file to the Cluster Manager VM, verify that the YAML file uses the proper syntax. There are many publicly-available Websites that you can use to validate your YAML configuration file.

Table 1: HTTPS Configuration Parameters

Parameter	Description
`api_https`	Runs the application with or without HTTPS (one way or mutual authentication). Valid options: • disabled (default) • one_way_ssl • two_way_ssl
`api_tls_version`	List of protocols that are supported. Valid options: • TLSv1.1 • TLSv1.2
`api_keystore_path`	Path to the Java keystore which contains the host certificate and private key. Required for one_way_ssl and two_way_ssl.
`api_keystore_type`	Type of keystore. Valid options: • Java KeyStore (JKS) • PKCS12 • JCEKS`` • Windows-MY} • Windows-ROOT Required for one_way_ssl and two_way_ssl.
`api_keystore_password`	Password used to access the keystore. Required for one_way_ssl and two_way_ssl.

Parameter	Description
`api_cert_alias`	Alias of the certificate to use. Required for one_way_ssl and two_way_ssl.
`api_truststore_path`	Path to the Java keystore which contains the CA certificates used to establish trust. Required for two_way_ssl.
`api_truststore_type`	The type of keystore. Valid options: • Java KeyStore (JKS) • PKCS12 • JCEKS`` • Windows-MY} • Windows-ROOT Required for two_way_ssl.
`api_truststore_password`	Password used to access the truststore. Required for two_way_ssl.
`api_validate_certs`	Decides whether or not to validate TLS certificates before starting. If enabled, wizard refuses to start with expired or otherwise invalid certificates. Valid options: • true • false Required for one_way_ssl and two_way_ssl.
`api_validate_peers`	Decides whether or not to validate TLS peer certificates. Valid options: • true • false Required for one_way_ssl and two_way_ssl.
`api_need_client_auth`	Decides whether or not client authentication is required. Valid options: • true • false Required for one_way_ssl and two_way_ssl.

Parameter	Description
`api_enable_crldp`	Decides whether or not CRL Distribution Points (CRLDP) support is enabled. Valid options: • true • false Required for two_way_ssl.