Juniper Test Env for Contrail+OSP Deployments

Env Description:
- Undercloud
- Openstack Controllers
- Openstack Computes
- Contrail Controllers
- Network Topology

Env Preparation:

Hypervisor Preparation
- System Configuration
- Reimage with Redhat 7.2
- Attach RHEL/RHOSP subscription
- Enable required repos
- Update
- Install Virtualization packages
- Enable nested KVM support
- Enable Packet routing
- Reboot hypervisor for the changes to take effect
- Bridge Interfaces for Director access
- Define networks in libvirt

Undercloud VM (OSP-Director)
- Undercloud VM Preparation
- VM image preparation

Contrail VM Preparation

Conclusion

Env Description:

This documents helps build a contrail-OSPD integrated environment. Setup consists of following components

Undercloud

The Undercloud is the main director node. It is a single-system OpenStack installation that includes components for provisioning and managing the OpenStack nodes that form your OpenStack environment (the Overcloud)

Openstack Controllers

Nodes that provide administration, networking, and high availability for the OpenStack environment. An ideal OpenStack environment recommends three of these nodes together in a high availability cluster

A default Controller node contains the following components: horizon, keystone, nova API, neutron server, Open vSwitch, glance, cinder volume, cinder API, swift storage, swift proxy, heat engine, heat API, ceilometer, MariaDB, RabbitMQ. The Controller also uses Pacemaker and Galera for high availability services.

Apart from these components, this node will also have contrail-neutron plugin which provides contrail enhancements to the neutron server

Openstack Computes

These nodes provide computing resources for the OpenStack environment. You can add more Compute nodes to scale out your environment over time.

A default Compute node contains the following components: nova Compute, nova KVM, ceilometer agent, Open vSwitch

Apart from these components, Contrail vrouter and agent will be installed in the compute. Contrail Vrouter and Open vSwitch are incompatible and so Open vSwitch will be removed from the compute node

Contrail Controllers

Contrail provides components like Config, Control, Analytics, Database and Webui and they are collectively called as Contrail Controller.

Config: It keeps a persistent copy of the intended configuration state and translates the high-level data model into the lower level model suitable for interacting with network elements. Runs Contrail-discovery, neutron-server, contrail-api, ifmap, schema, svc-monitor, and AMQP services.

Control: Implements a logically centralized control plane responsible for maintaining network state. Control nodes interact with each other and with network elements to ensure that

network state is eventually consistent. Runs contrail-control, contrail-dns, contrail-named services

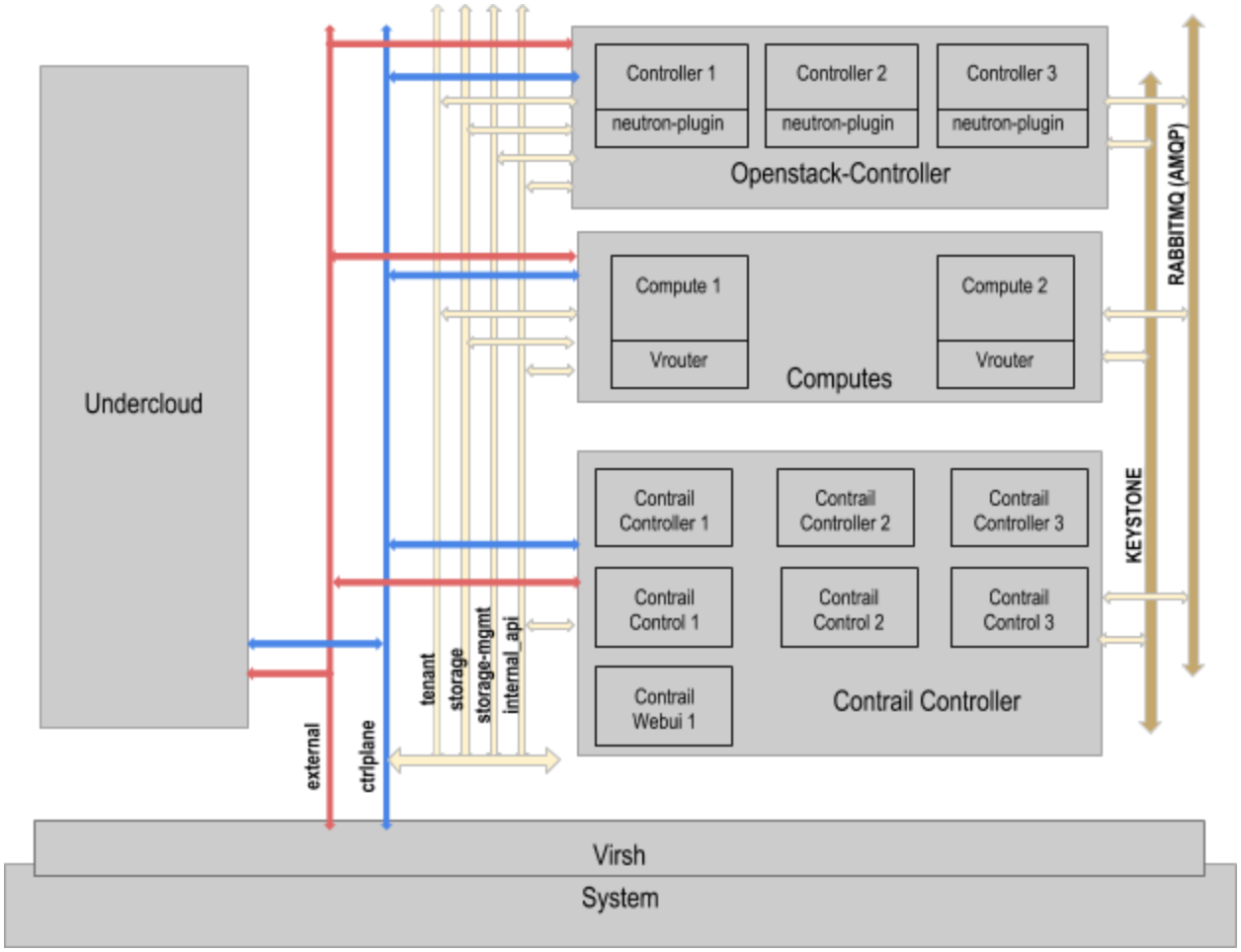
Analytics: Analytics nodes collect, store, correlate, and analyze information from network elements, virtual or physical. This information includes statistics, logs, events, and errors. Runs contrail-collector, analytics- api, query-engine, snmp-collector and contrail-topology services

Database: Runs Cassandra and Zookeeper services

WebUI: Provides a Web UI for managing contrail networking solution. Runs contrail-webui and webui- middleware services.

Network Topology

Network topology with OSP-Director + Contrail.



Env Preparation:

Hypervisor Preparation

System Configuration

The baremetal machine should meet the following minimum system requirements:

- Virtualization hardware extensions enabled
- 1 quad core CPU
- 32 GB free memory
- 500 GB disk space

Each virtual machine must consist of at least 8GB of memory and 80 GB of disk space

Note: The virtual machine disk files are thinly provisioned and will not take up the full 80GB initially.

Operating System: RHEL 7.2

Reimage with Redhat 7.2

Use “RHEL 7.2 Binary DVD ISO” to reimage hypervisor node from below url
https://access.redhat.com/downloads/content/69/ver=/rhel---7/7.2/x86_64/product-software

Attach RHEL/RHOSP subscription

Attach your redhat subscription to the hypervisor. Use list command to the list of available of pools and attach the appropriate pool ID.

```
subscription-manager register --username <username> --password <password> --force
sudo subscription-manager list --available --all
subscription-manager attach --pool <pool-id>
```

Enable required repos

Enable required repos in the hypervisor.

```
subscription-manager repos --enable=rhel-7-server-rpms
--enable=rhel-7-server-extras-rpms  --enable=rhel-7-server-openstack-8-rpms;

subscription-manager repos --disable=*;

subscription-manager repos --enable=rhel-7-server-rpms
--enable=rhel-7-server-rh-common-rpms --enable=rhel-7-server-extras-rpms
--enable=rhel-7-server-openstack-8-rpms
```

Update

Update the hypervisor to the latest packages and reboot system

```
yum -y upgrade
systemctl reboot
```

Install Virtualization packages

Install virtualization packages as listed below and enable/start libvirtd process

```
yum groupinstall "Virtualization Host"
yum install -y libguestfs-tools libvirt qemu-kvm virt-manager virt-install
xorg-x11-apps xauth virt-viewer libguestfs-xfs fence-virt fence-virtd
fence-virtd-multicast fence-virtd-libvirt
systemctl enable libvirtd && systemctl start libvirtd
```

Install GUI packages (Optional but Recommended). This bring in packages related to fonts, which will be required for virt-manager fonts.

```
yum -y groupinstall "Server with GUI"
```

Enable nested KVM support

```
cat << EOF > /etc/modprobe.d/kvm_intel.conf
options kvm-intel nested=1
options kvm-intel enable_shadow_vmcs=1
options kvm-intel enable_apicv=1
options kvm-intel ept=1
EOF
```

Enable Packet routing

```
cat << EOF > /etc/sysctl.d/98-rp-filter.conf
net.ipv4.conf.default.rp_filter = 0
net.ipv4.conf.all.rp_filter = 0
EOF
```

Reboot hypervisor for the changes to take effect

```
systemctl reboot
```

Bridge Interfaces for Director access

From the hypervisor, create below interface config files at /etc/sysconfig/network-scripts/

Bridge (br-ex) for infrastructure connectivity (Will be eth0 on director vm. An example bridge interface

```
[root@hypervisor] # vi /etc/sysconfig/network-scripts/ifcfg-br-ex
DEVICE=br-ex
TYPE=Bridge
ONBOOT="yes"
BOOTPROTO="dhcp"
NM_CONTROLLED="no"
DELAY=0
```

Bridge (br-ctrlplane) for RHEL OSP PXE/Provision Network (Will become eth1)

```
DEVICE=br-ctrlplane
TYPE=Bridge
ONBOOT="yes"
BOOTPROTO="none"
NM_CONTROLLED="no"
DELAY=0
```

Attach bridge br-ex to external or mgmt interface of hypervisor. Update external or mgmt interface file with bridge info and restart network. In below example eno1 is the mgmt address of the hypervisor. NAME,DEVICE,UUID are all example, please retain the original values

```
NAME="eno1"
DEVICE="eno1"
ONBOOT=yes
NETBOOT=yes
UUID="3baae127-5bb0-4a73-b8f9-04c3c9d65c89"
IPV6INIT=yes
BOOTPROTO=none
TYPE=Ethernet
BRIDGE=br-ex
NM_CONTROLLED=no
```

Restart networking

```
systemctl restart network
```

Define networks in libvirt

Define two networks, external and ctrlplane using below virsh commands with an example template of the xml files as shown below.

```
# cat << EOF > /tmp/external.xml
<network>
  <name>external</name>
  <forward mode='bridge'/>
  <bridge name='br-ex'/>
</network>
EOF
```

```
# cat << EOF > /tmp/ctrlplane.xml
<network>
  <name>ctrlplane</name>
  <forward mode='bridge'/>
  <bridge name='br-ctrlplane'/>
</network>
EOF
```

```
virsh net-define /tmp/external.xml
virsh net-autostart external
virsh net-start external

virsh net-define /tmp/ctrlplane.xml
virsh net-autostart ctrlplane
virsh net-start ctrlplane

To view the added networks:
# virsh net-list
 Name                 State      Autostart     Persistent
----------------------------------------------------------
 ctrlplane            active     yes           yes
 default              active     yes           yes
 external             active     yes           yes

# virsh net-info ctrlplane
Name:           ctrlplane
UUID:           16015702-ff4d-4f07-864f-27badb775366
Active:         yes
Persistent:     yes
Autostart:      yes
Bridge:         br-ctrlplane

# virsh net-info external
Name:           external
UUID:           2b84eaf4-d8a9-4d17-80b1-356ea63bd2fd
Active:         yes
Persistent:     yes
Autostart:      yes
Bridge:         br-ex
```

Undercloud VM (OSP-Director)

This section explains creating VM image for undercloud host and provision undercloud. All virt-customize commands are run from hypervisor.

Undercloud VM Preparation

Below package provides rhel-7 cloud image which could be used as the base image for undercloud VM image

```
[root@hypervisor] # yum -y install rhel-guest-image-7
```

VM image preparation

Copy cloud image to a temporary directory

```
[root@hypervisor] # mkdir /tmp/uc-prep
[root@hypervisor] # cd /tmp/uc-prep
[root@hypervisor] # cp
/usr/share/rhel-guest-image-7/rhel-guest-image-*.x86_64.qcow2
/tmp/uc-prep/undercloud.qcow2
```

Resize the undercloud image to the desired size. At least 80G of disk is recommended. Note that the fdisk command will return an error but it actually completes. The error is a warning that it is editing a mounted device and the change will not be visible until it is remounted.

```
[root@hypervisor] # qemu-img resize undercloud.qcow2 +80G
[root@hypervisor] # virt-customize -a undercloud.qcow2 --run-command 'echo -e
"d\nn\n\n\n\n\nw\n" | fdisk /dev/sda'
[root@hypervisor] # virt-customize -a undercloud.qcow2 --run-command 'xfs_growfs /'
[root@hypervisor] # virt-customize -a undercloud.qcow2 --run-command 'yum remove
cloud-init* -y'
[root@hypervisor] # virt-customize -a undercloud.qcow2 --root-password
password:<password>
```

Add eth1/eth0 interfaces to the undercloud vm image

```
[root@hypervisor] # virt-customize -a undercloud.qcow2 --run-command 'cp
/etc/sysconfig/network-scripts/ifcfg-eth{0,1} && sed -i s/DEVICE=.*/DEVICE=eth1/g
/etc/sysconfig/network-scripts/ifcfg-eth1'

[root@hypervisor] # virt-customize -a undercloud.qcow2 --run-command 'cat << EOF >
/etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
ONBOOT="yes"
TYPE="Ethernet"
PEERDNS="yes"
IPV6INIT="no"
BOOTPROTO=dhcp
EOF'
```

Copy prepared undercloud image to the standard image path of libvirt and the prepared qcow2 can be backed up to reused when anything goes wrong with undercloud.qcow2

```
[root@hypervisor] # cp /tmp/uc-prep/undercloud.qcow2
/var/lib/libvirt/images/undercloud.qcow2
```

Boot undercloud image using below command

```
[root@hypervisor] # virt-install --ram 16384 --vcpus 4 \
--os-variant rhel7 \
--disk path=/var/lib/libvirt/images/undercloud.qcow2 \
--import --noautoconsole --vnc \
--bridge br-ex \
--bridge br-ctrlplane \
--name undercloud
```

Connect to undercloud console and create director installation user and provide sudo access

```
[root@hypervisor ~]$ virsh console undercloud
[root@undercloud ~]$ useradd stack
[root@undercloud ~]$ passwd stack
[root@undercloud ~]$ echo "stack ALL=(root) NOPASSWD:ALL" | tee -a
/etc/sudoers.d/stack
[root@undercloud ~]$ chmod 0440 /etc/sudoers.d/stack
```

Use stack user to perform rest of the operations

```
[root@undercloud ~]$ su - stack
```

Set hostname for undercloud image

```
[stack@undercloud ~]$ sudo hostnamectl set-hostname undercloud.example.com
[stack@undercloud ~]$ sudo hostnamectl set-hostname --transient
undercloud.example.com
[stack@undercloud ~]$ sudo echo "<ip address of undercloud> undercloud.example.com
undercloud" >> /etc/hosts
[stack@undercloud ~]$ cat /etc/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
<ip address of undercloud> undercloud.example.com undercloud
```

Attach RHEL subscription to the undercloud and enable required packages

```
[stack@undercloud ~]$ sudo subscription-manager register --username <username>
--password <password> --force
[stack@undercloud ~]$ sudo subscription-manager list --available --all
[stack@undercloud ~]$ sudo subscription-manager attach --pool <pool-id>
```

```
[stack@undercloud ~]$ sudo subscription-manager repos --enable=rhel-7-server-rpms
--enable=rhel-7-server-extras-rpms  --enable=rhel-7-server-openstack-8-rpms;
[stack@undercloud ~]$ sudo subscription-manager repos --disable=*;

[stack@undercloud ~]$ sudo subscription-manager repos --enable=rhel-7-server-rpms
--enable=rhel-7-server-extras-rpms   --enable=rhel-7-server-openstack-8-rpms
--enable=rhel-7-server-openstack-8-director-rpms
```

Create directories to store image and templates

```
[stack@undercloud ~]$ mkdir ~/images
[stack@undercloud ~]$ mkdir ~/templates
```

Enable IP forwarding in the undercloud

```
[stack@undercloud ~]$ cat << EOF > /tmp/97-ip_forward.conf
net.ipv4.ip_forward = 1
EOF
[stack@undercloud ~]$ sudo cp /tmp/97-ip_forward.conf
/etc/sysctl.d/97-ip_forward.conf
```

Install useful packages in the undercloud

```
[stack@undercloud ~]$ sudo yum -y install screen vim
```

Install tripleoclient package

```
[stack@undercloud ~]$ sudo yum install -y python-tripleoclient
```

Copy sample undercloud.conf and update desired password and local interface name

```
[stack@undercloud ~]$ cp /usr/share/instack-undercloud/undercloud.conf.sample
/home/stack/undercloud.conf
sed -i -e '/^\[DEFAULT\]/a local_interface = <ctrlplane interface name (ex:eth1)>'
/home/stack/undercloud.conf
sed -i -e '/^\[auth\]/a undercloud_admin_password = <password of undercloud>'
/home/stack/undercloud.conf
```

Reboot undercloud VM for the changes to take effect

```
[stack@undercloud ~]$ sudo yum update -y
[stack@undercloud ~]$ sudo reboot
```

Install undercloud

```
[stack@undercloud ~]$ openstack undercloud install
```

After the undercloud installation, stackrc file will be created

```
[stack@undercloud ~]$ source ~/stackrc
```

Below packages provide overcloud image

```
[stack@undercloud ~]$ sudo yum install rhosp-director-images
rhosp-director-images-ipa
```

Copy Overcloud images to local path and customize it if necessary

```
[stack@undercloud ~]$ cp
/usr/share/rhosp-director-images/overcloud-full-latest-8.0.tar ~/images/.
[stack@undercloud ~]$ cp
/usr/share/rhosp-director-images/ironic-python-agent-latest-8.0.tar ~/images/.
[stack@undercloud ~]$ cd ~/images
[stack@undercloud ~]$ for tarfile in *.tar; do sudo tar -xf $tarfile; done
```

Upload overcloud images to the undercloud glance

```
[stack@undercloud ~]$ openstack overcloud image upload --image-path
/home/stack/images/
[stack@undercloud ~]$ openstack image list
```

Verify below introspection PXE images are created at /httpboot directory in the undercloud

```
[stack@undercloud ~]$ ls -l /httpboot
total 388332
-rwxr-xr-x. 1 root   root     5153536 Oct 26 13:18 agent.kernel
-rw-r--r--. 1 root   root   392481403 Oct 26 13:18 agent.ramdisk
-rw-r--r--. 1 root root       409 Nov 15 00:53 inspector.ipxe
```

Overcloud nodes require a nameserver so that they can resolve hostnames through DNS. Update ctrlplane network with a DNS server address

```
[stack@undercloud ~]$ neutron subnet-list
[stack@undercloud ~]$ neutron subnet-update [subnet-uuid] --dns-nameserver
[nameserver-ip]
[stack@undercloud ~]$ neutron subnet-show [subnet-uuid]
```

Tune the configuration for this virtualized environment

```
[stack@undercloud ~]$ sudo openstack-config --set /etc/nova/nova.conf DEFAULT
rpc_response_timeout 600
[stack@undercloud ~]$ sudo openstack-config --set /etc/ironic/ironic.conf DEFAULT
rpc_response_timeout 600
[stack@undercloud ~]$ sudo openstack-service restart nova
[stack@undercloud ~]$ sudo openstack-service restart ironic
```

Generate and copy ssh keys from the hypervisor to the undercloud and vice versa, so that no password is required to login undercloud VM from hypervisor and vice versa

```
# Create ssh key if not exists in hypervisor
[root@hypervisor] # ssh-keygen -t rsa
```

```
# Create undercloud entry in /etc/hosts of hypervisor
[root@hypervisor] # echo "<ip-address-of-undercloud> undercloud" >> /etc/hosts

# Copy ssh key of hypervisor to root and stack users of undercloud
[root@hypervisor] # cat /root/.ssh/id_rsa.pub | ssh root@<undercloud> 'cat >>
/root/.ssh/authorized_keys'
[root@hypervisor] # cat /root/.ssh/id_rsa.pub | ssh stack@<undercloud> 'cat >>
/home/stack/.ssh/authorized_keys'

# Create ssh key if not exists in undercloud
[root@undercloud] # ssh-keygen -t rsa
[stack@undercloud] # ssh-keygen -t rsa

# Create hypervisor entry in /etc/hosts of undercloud
[root@undercloud] # echo "<ip-address-of-hypervisor> <hyerpervisor-hostname>" >>
/etc/hosts

# Copy ssh key of root and stack user of undercloud to hypervisor
[root@undercloud] # cat /root/.ssh/id_rsa.pub | ssh root@<hypervisor> 'cat >>
/root/.ssh/authorized_keys'
[stack@undercloud] # cat /home/stack/.ssh/id_rsa.pub | ssh root@<hypervisor> 'cat
>> /root/.ssh/authorized_keys'

Change the permission of authorized_keys on hypervisor and undercloud root and
stack users

[stack@undercloud .ssh]$ chmod 600 authorized_keys
[root@undercloud .ssh]$ chmod 600 authorized_keys
[root@hypervisor .ssh]# chmod 600 authorized_keys
```

Verify no password required is to run any command in the undercloud VM from hypervisor. With undercloud host-to-ip mapping added to the /etc/hosts in the hypervisor

```
[root@hypervisor]# ssh stack@<undercloud> -t 'virsh --connect
qemu+ssh://root@<hypervisor>/system list --all'
```

Use below scriptlet to generate overcloud VM images. Qcow2 images created at /var/lib/libvirt/images/ Copy below scriplet to a file /tmp/generate-vm-images.sh and save. Run from hypervisor.

```
[root@hypervisor] # chmod 755 /tmp/generate-vm-images.sh
[root@hypervisor] # /tmp/generate-vm-images.sh
```

```
#!/bin/bash

# If you want to change the size of the overcloud VMs based on release, uncomment
this code
#if [ $# -ne 1 ]; then
#echo "$0: usage: $0 rhos_release"
```

```
#exit 1
#fi
#rhos_release="$1"
#if [ "$rhos_release" != "7" ] &&  [ "$rhos_release" != "8" ] && [ "$rhos_release"
!= "7GA" ] && [ "$rhos_release" != "8GA" ];then
#  echo "Unsupported rhos_release version:"
#  echo "'$rhos_release'"
#  exit 1
#fi

# 2016-08-05 I was having issues with deployment (networking of overcloud) and
removed the ,model=e1000 and it started working
# set e1000 for provisioning network in an attempt to avoir
https://bugzilla.redhat.com/show_bug.cgi?id=1326481 (whatever this issue is ...)

mkdir -p /var/lib/libvirt/images/
cd /var/lib/libvirt/images/
for i in {1..5}; do qemu-img create -f qcow2 -o preallocation=metadata
overcloud-node$i.qcow2 60G; done

# Controllers
for i in {1..3}; do
   virt-install \
  --ram 8192 --vcpus 4 \
  --os-variant rhel7 \
  --disk
path=/var/lib/libvirt/images/overcloud-node$i.qcow2,device=disk,bus=virtio,format=q
cow2 \
  --noautoconsole --vnc \
  --network network:external \
  --network network:ctrlplane \
  --name overcloud-node$i \
  --cpu SandyBridge,+vmx  --dry-run --print-xml > overcloud-node$i.xml;
 virsh define --file overcloud-node$i.xml;
done

# Computes
for i in {4..5}; do
  virt-install \
  --ram 8192 --vcpus 4 \
  --os-variant rhel7 \
  --disk
path=/var/lib/libvirt/images/overcloud-node$i.qcow2,device=disk,bus=virtio,format=q
cow2 \
  --noautoconsole --vnc \
  --network network:external \
  --network network:ctrlplane \
  --name overcloud-node$i \
  --cpu SandyBridge,+vmx  --dry-run --print-xml > overcloud-node$i.xml;
  virsh define --file overcloud-node$i.xml;
done

echo "$0 complete"
```

Use below scriptlet to generate instackenv.json, a template to register overcloud VMs to ironic. Update IP address of Hypervisor node in below script before executing the script. Run below script from undercloud node.
Copy below scriptlet to /tmp/generate-instackjson.sh

```
[stack@undercloud] # chmod 755 /tmp/generate-instackjson.sh
[stack@undercloud] # /tmp/generate-instackjson.sh
```

```
#!/bin/bash
set -x

hypervisor=<IP ADDRESS OF HYPERVISOR HOST>
mkdir -p ~/hwinstackfiles
cd ~/hwinstackfiles
for i in {1..5}; do \
    virsh -c qemu+ssh://root@${hypervisor}/system domiflist overcloud-node$i | awk
'$3 == "ctrlplane" {print $5}'; \
    done > nodemacs.txt
jq . << EOF > ~/hwinstackfiles/instackenv.json
{
  "ssh-user": "root",
  "ssh-key": "$(cat ~/.ssh/id_rsa)",
  "power_manager": "nova.virt.baremetal.virtual_power_driver.VirtualPowerManager",
  "host-ip": "$hypervisor",
  "arch": "x86_64",
  "nodes": [
    {
      "name": "overcloud-node1",
      "pm_addr": "$hypervisor",
      "pm_password": "$(cat ~/.ssh/id_rsa)",
      "pm_type": "pxe_ssh",
      "mac": [
        "$(sed -n 1p ~/hwinstackfiles/nodemacs.txt)"
      ],
      "cpu": "4",
      "memory": "8192",
      "disk": "60",
      "arch": "x86_64",
      "pm_user": "root"
    },
    {
      "name": "overcloud-node2",
      "pm_addr": "$hypervisor",
      "pm_password": "$(cat ~/.ssh/id_rsa)",
```

```
    "pm_type": "pxe_ssh",
    "mac": [
      "$(sed -n 2p ~/hwinstackfiles/nodemacs.txt)"
    ],
    "cpu": "4",
    "memory": "8192",
    "disk": "60",
    "arch": "x86_64",
    "pm_user": "root"
  },
  {
    "name": "overcloud-node3",
    "pm_addr": "$hypervisor",
    "pm_password": "$(cat ~/.ssh/id_rsa)",
    "pm_type": "pxe_ssh",
    "mac": [
      "$(sed -n 3p ~/hwinstackfiles/nodemacs.txt)"
    ],
    "cpu": "4",
    "memory": "8192",
    "disk": "60",
    "arch": "x86_64",
    "pm_user": "root"
  },
  {
    "name": "overcloud-node4",
    "pm_addr": "$hypervisor",
    "pm_password": "$(cat ~/.ssh/id_rsa)",
    "pm_type": "pxe_ssh",
    "mac": [
      "$(sed -n 4p ~/hwinstackfiles/nodemacs.txt)"
    ],
    "cpu": "4",
    "memory": "8192",
    "disk": "60",
    "arch": "x86_64",
    "pm_user": "root"
  },
  {
    "name": "overcloud-node5",
    "pm_addr": "$hypervisor",
    "pm_password": "$(cat ~/.ssh/id_rsa)",
    "pm_type": "pxe_ssh",
    "mac": [
      "$(sed -n 5p ~/hwinstackfiles/nodemacs.txt)"
    ],
    "cpu": "4",
    "memory": "8192",
    "disk": "60",
    "arch": "x86_64",
    "pm_user": "root"
  }
]
```

```
}
EOF
cp ~/hwinstackfiles/instackenv.json ~/instackenv.json
```

Import node definitions

```
[stack@undercloud] # source /home/stack/stackrc
[stack@undercloud] # openstack baremetal import --json /home/stack/instackenv.json
```

Notes: (Applicable only if in specific cases)

Case 1:

During the introspect (Section: Introspect overcloud nodes with ironic in "RHT/JNPR - OSP8 Contrail 3.0.2 Implementation - Full Lifecycle" document), all the VMs added vi instack.json will be spawned and evaluated. It's possible that VMs can timeout while PXE booting, use below fix to resolve it. This step should be executed when VM boot fails

```
vi ipxe-timeout-fix
#!/usr/bin/env bash

while true;do
        find /httpboot/ -type f -iname "*.ipxe" -o -iname config | while read
filename;do
          if `file $filename | grep -q "ASCII text"`;then
            #echo $filename
            sed -i 's/--timeout [0-9]\+//g' $filename
          fi
        done
        sleep 1
done

vi ipxe-timeout-fix.service
[Unit]
Description=Automated fix for incorrect iPXE timeout

[Service]
Type=simple
ExecStart=/usr/bin/ipxe-timeout-fix

[Install]
WantedBy=multi-user.target
```

Run above scriptlet as a service

```
[stack@undercloud] # sudo cp ipxe-timeout-fix /usr/bin/ipxe-timeout-fix
[stack@undercloud] # sudo cp ipxe-timeout-fix.service
/etc/systemd/system/ipxe-timeout-fix.service
[stack@undercloud] # sudo chmod a+x /usr/bin/ipxe-timeout-fix
[stack@undercloud] # sudo systemctl daemon-reload
[stack@undercloud] # sudo systemctl enable ipxe-timeout-fix
```

```
[stack@undercloud] # sudo systemctl start ipxe-timeout-fix
```

Case 2:

If user prefers the VMs to boot with nic 0 Create below script which sets the boot interface as net0

```
[stack@undercloud ~]$ cat /usr/bin/bootif-fix
#!/usr/bin/env bash

while true;do
        find /httpboot/ -type f ! -iname "kernel" ! -iname "ramdisk" ! -iname
"*kernel" ! -iname "*ramdisk" | while read filename;do
          if `file $filename | grep -q "ASCII text"`;then
            #echo $filename
            sed -i 's|{mac|{net0/mac|g' $filename
          fi
        done
        sleep 5
done
[stack@undercloud ~]$
```

Run this script as a service

```
[stack@undercloud ~]$ cat /etc/systemd/system/bootif-fix.service
[Unit]
Description=Automated fix for incorrect iPXE BOOFIF

[Service]
Type=simple
ExecStart=/usr/bin/bootif-fix

[Install]
WantedBy=multi-user.target
```

And enable this servie

```
[stack@undercloud] # sudo chmod 755 /usr/bin/bootif-fix
[stack@undercloud] # sudo systemctl daemon-reload
[stack@undercloud] # sudo systemctl enable bootif-fix
[stack@undercloud] # sudo systemctl start bootif-fix
```

Contrail VM Preparation

Contrail packages are not pre installed in the vm images and fabric-utils from contrail provides necessary scripts to install and provision contrail components. Below preparation would configure basic VM images for contrail and could be launched from the hypervisor

From the hypervisor, Resize the contrail-vm image to the desired size. At least 80G of disk is recommended

```
[root@hypervisor ~] # mkdir /tmp/contrail-prep
[root@hypervisor ~] # cd /tmp/contrail-prep
[root@hypervisor] # cp
/usr/share/rhel-guest-image-7/rhel-guest-image-*.x86_64.qcow2
/tmp/contrail-prep/contrail-vm.qcow2
```

Note that the fdisk command will return an error but it actually completes. The error is a warning that it is editing a mounted device and the change will not be visible until it is remounted.

```
[root@hypervisor ~] # cd /tmp/contrail-prep/
[root@hypervisor ~] # qemu-img resize contrail-vm.qcow2 +80G
[root@hypervisor ~] # virt-customize -a contrail-vm.qcow2 --run-command 'echo -e
"d\nn\n\n\n\n\nw\n" | fdisk /dev/sda'
[root@hypervisor ~] # virt-customize -a contrail-vm.qcow2 --run-command 'xfs_growfs
/'
[root@hypervisor ~] # virt-customize -a contrail-vm.qcow2 --run-command 'yum remove
cloud-init* -y'
[root@hypervisor ~] # virt-customize -a contrail-vm.qcow2 --root-password
password:<password>
```

Copy prepared contrail image to the standard image path of libvirt in the hypervisor.

```
[root@hypervisor ~] # cp /tmp/contrail-prep/contrail-vm.qcow2
/var/lib/libvirt/images/contrail-controller1.qcow2
[root@hypervisor ~] # cp /tmp/contrail-prep/contrail-vm.qcow2
/var/lib/libvirt/images/contrail-controller2.qcow2
[root@hypervisor ~] # cp /tmp/contrail-prep/contrail-vm.qcow2
/var/lib/libvirt/images/contrail-controller3.qcow2
[root@hypervisor ~] # cp /tmp/contrail-prep/contrail-vm.qcow2
/var/lib/libvirt/images/contrail-control1.qcow2
[root@hypervisor ~] # cp /tmp/contrail-prep/contrail-vm.qcow2
/var/lib/libvirt/images/contrail-control2.qcow2
[root@hypervisor ~] # cp /tmp/contrail-prep/contrail-vm.qcow2
/var/lib/libvirt/images/contrail-control3.qcow2
[root@hypervisor ~] # cp /tmp/contrail-prep/contrail-vm.qcow2
```

```
/var/lib/libvirt/images/contrail-webui1.qcow2
```

Create Static eth1/eth0 interfaces in each of the contrail vm image. Replace image-name with the qcow2 name of the contrail image.

```
[root@hypervisor] # cd /var/lib/libvirt/images/
[root@hypervisor] # virt-customize -a <image-name>.qcow2 --run-command 'cp
/etc/sysconfig/network-scripts/ifcfg-eth{0,1} && sed -i s/DEVICE=.*/DEVICE=eth1/g
/etc/sysconfig/network-scripts/ifcfg-eth1'

# For external network interface
Update IP address of eth0, eth1, eth2 with External subnet, Ctlplane subnet and
Internal API network respectively.

[root@hypervisor ~] # virt-customize -a <image-name>.qcow2 --run-command 'cat <<
EOF > /etc/sysconfig/network-scripts/ifcfg-eth0
DEVICE="eth0"
IPADDR=<IP-ADDRESS-FROM-EXTERNAL-SUBNET>
NETMASK=<NET-MASK>
ONBOOT="yes"
TYPE="Ethernet"
PEERDNS="yes"
IPV6INIT="no"
BOOTPROTO=none
EOF'

# For Control plane (ctlplane) network interface
[root@hypervisor ~] # virt-customize -a <image-name>.qcow2 --run-command 'cat <<
EOF > /etc/sysconfig/network-scripts/ifcfg-eth1
DEVICE="eth1"
IPADDR=<IP-ADDRESS-FROM-CTLPLANE-SUBNET>
NETMASK=<NET-MASK>
ONBOOT="yes"
TYPE="Ethernet"
PEERDNS="yes"
IPV6INIT="no"
BOOTPROTO=none
EOF'

# For Internal API network interface
[root@hypervisor ~]#
virt-customize -a contrail-controller1.qcow2 --run-command 'cat << EOF >
/etc/sysconfig/network-scripts/ifcfg-<vlan name of internal_api network>
DEVICE="<vlan name of internal_api network>"
ONBOOT="yes"
HOTPLUG="no"
NM_CONTROLLED="no"
PEERDNS="no"
IPADDR=<IP-ADDRESS-FROM-INTERNAL-API-SUBNET>
NETMASK=<NETMASK>
```

```
BOOTPROTO=none
VLAN=yes
PHYSDEV="<CTRLPLANE INTERFACE NAME ex: eth1>"
EOF'
```

Boot all the Contrail image using below command. Replace image-name with the qcow2 name of the contrail image.

```
[root@hypervisor ~] # virt-install --ram 16384 --vcpus 8 \
--os-variant rhel7 \
--disk path=/var/lib/libvirt/images/<image-name>.qcow2 \
--import --noautoconsole --vnc \
--bridge br-ex \
--bridge br-ctrlplane \
--name <image-name>
```

Conclusion

Setup is now prepared and is ready for overcloud installation.
Note: Overcloud images added during this doc is vanilla overcloud-full image. However Contrail's neutron-plugin would need to be installed in openstack-controller and contrail vrouter needs to be installed in openstack-compute image.