## Prerequisites: <u>You must have external network connectivity.</u>

## DHCP ( neutron)

- When we create the subnet the DHCP get created automatically with the first IP of the subnet (192.168.122.100 here)

$ neutron net-list *// to check the network list (public/external network)*


**Create subnet for OCP cluster and provide range for OCP hosts**

# source overcloudrc

# neutron subnet-create --dns-nameserver 8.8.8.8,8.8.4.4 --allocation-pool start=192.168.122.100,end=192.168.122.150
*// We will change the DNS server later as we are creating it for our OCP cluster.*

# neutron subnet-list
neutron CLI is deprecated and will be removed in the future. Use openstack CLI instead.

**To get a fixed IP address, create ports with static IP for the OCP hosts.**

# for bootstrap
neutron port-create --fixed-ip
subnet_id=29f61c6a-3025-4b73-9238-ef81c41267e2,ip_address=192.168.122.101
public_network

# for Master
neutron port-create --fixed-ip
subnet_id=29f61c6a-3025-4b73-9238-ef81c41267e2,ip_address=192.168.122.102
public_network

# for worker
neutron port-create --fixed-ip
subnet_id=29f61c6a-3025-4b73-9238-ef81c41267e2,ip_address=192.168.122.103
public_network

# for LB
neutron port-create --fixed-ip
subnet_id=29f61c6a-3025-4b73-9238-ef81c41267e2,ip_address=192.168.122.105
public_network

## DNS:

Now we create the DNS server on 192.168.122.149 (again this is a fixed port we created it)

> Create RHEL DNS host ( using dnsmasq)

- Adding a records in /etc/addnhost

```
################################
/etc/addnhost
192.168.122.101 bootstrap.ocplab.example.com
192.168.122.102 master.ocplab.example.com etcd-0.ocplab.example.com
192.168.122.103 worker.ocplab.example.com
192.168.122.149 lb.ocplab.example.com api.ocplab.example.com api-int.ocplab.example.com
lb
192.168.122.149 dns-server-ocp-lab.ocplab.example.com
################################
```

- Config of dnsmasq:

```
~~~
/etc/dnsmasq.d/abc
local=/ocplab.example.com/
address=/apps.ocplab.example.com/192.168.122.149
srv-host=_etcd-server-ssl._tcp.ocplab.example.com,master.ocplab.example.com,2380,0,10
no-hosts
server=192.168.122.1
addn-hosts=/etc/addnhosts
~~~
```

**[IMP]** Update DNS server in the subnet that we have created.**

```
# neutron subnet-update --dns-nameserver 192.168.122.149
29f61c6a-3025-4b73-9238-ef81c41267e2
```

## LB (host haproxy)

- Sample haproxy config having 1 master configuration

```
#################################################################################
# Global settings
#---------------------------------------------------------------------
        log     127.0.0.1:514 local0 debug
        chroot          /var/lib/haproxy
        pidfile  /var/run/haproxy.pid
        maxconn         10000
        user    haproxy
        group           haproxy
        daemon

        # turn on stats unix socket
        stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#---------------------------------------------------------------------
defaults
        mode                    http
        log             global
        option                  httplog
        option                  dontlognull
        option                  redispatch
        retries         3
        timeout http-request    10s
        timeout queue           1m
        timeout connect         10s
        timeout client          300s
        timeout server          300s
```

```
        timeout http-keep-alive 10s
        timeout check           10s
        maxconn                 10000

#---------------------------------------OCP API-------------------------------------------
frontend  atomic-openshift-api
        bind *:6443
        default_backend atomic-openshift-api
        mode tcp
        option tcplog

backend atomic-openshift-api
        balance source
        mode tcp
        server          bootstrap 10.74.252.114:6443 check
        server          master 10.74.255.187:6443 check



#---------------------------------MACHINE CONFIGS-------------------------------------------
frontend  machine-config-server
        bind *:22623
        default_backend machine-config-server
        mode tcp
        option tcplog

backend machine-config-server
        balance source
        mode tcp
        server          bootstrap 10.74.252.114:22623 check
        server          master 10.74.255.187:22623 check


#---------------------------------------HTTP traffic---------------------------------------------

frontend http-forward
        bind *:80
        use_backend http-backend
        mode http

backend http-backend
        balance leastconn
        mode http
```

```
        server          worker 10.74.255.187:80 check


#---------------------------------------HTTPS traffic---------------------------------------------------
frontend https-forward
        bind *:443
        default_backend https-backend
        mode tcp
        option tcplog

backend https-backend
        balance leastconn
        mode tcp
        server          worker 10.74.255.187:443 check

#############################################################################
```

## BASTION:

- Download openshift-install-linux-4.2.12.tar.gz and openshift-client-linux-4.2.12.tar.gz and extract it.
- Create installer-config Read https://docs.openshift.com/container-platform/4.2/installing/installing_bare_metal/installing-bare-metal.html#installation-initializing-manual_installing-bare-metal

- Create manifiests and ignition files Read https://docs.openshift.com/container-platform/4.2/installing/installing_bare_metal/installing-bare-metal.html#installation-user-infra-generate-k8s-manifest-ignition_installing-bare-metal

## HTTP host (RHEL)

Download the rhcos-4.2.0-x86_64-installer.iso and rhcos-4.2.0-x86_64-metal-bios.raw.gz and place it in http server. Also, place the ignition configs in it.

## Boot the OCP Instances

### Create bootstrap host on OSP

- Download rhcos iso and make it available on openstack ( on over-cloud):

  # openstack image create --container-format=bare --disk-format=iso --file=rhcos-4.2.0-x86_64-installer.iso --public rhcos

  # openstack image list

- Create temporary instance using the newly created image:

  # nova boot --flavor <flavor-name> --image rhcos <instance-name>

- Create a volume and add it to the instance:

  # openstack volume create --size <size> --bootable <vol-name>

  # openstack server add volume <instance-name> <vol-name>t --device /dev/vda

- On the OSP console, open up the temporary instance and check if the volume is attached to the instance. After that install the coreos on the attached volume:

  # /usr/libexec/coreos-installer -d vda -i <path/to/ign/file> -b <path/to/raw.gz>

- After the installation is complete, delete the temporary instance and create the qcow2 image using the volume from that instance:

  # openstack server delete <temporary-instance>

  # openstack image create --volume <vol-name> <img-name>

- Create the node from newly created image:

  # nova boot --flavor <flavor-name> --image <img-name> <name-of-node> --nic port-id=20d55d3-e018-4421-95dc-700ca7ac449 // which we created earlier

**NOTE:** Apply the aforementioned process for the master and worker nodes(hosts), just change the ignition configs as per the hosts

**IMP: Why have we followed the above approach for creating coreOS bootable image?**

- As OSP has constraint that it accepts only a single bootable disk image(iso or qcow2 or raw or etc..) to boot the instance.
- So we initially mount the ISO at the time of boot and it takes as Read Only partition, so for that we attached a separate volume for instance form cinder and mark that volume as "bootable".
- After that we will pass the aforementioned volume for coreos installation along with the other OCP coreos parameters the configs and data will be written in the Cinder Volume.
- Now using this volume we create a bootable qcow2 OSP compatible image and use this to boot the CoreOs instance for installation.

**After installation:**

Monitor the logs on master and workers

- Check the status of nodes

```
# ./oc get nodes
NAME                          STATUS     ROLES   AGE    VERSION
master.ocplab.example.com     Ready      master  156m   v1.14.6+cebabbf4a
worker.ocplab.example.com     Ready      worker  156m   v1.14.6+cebabbf4a
```

**KNOWN ISSUE:**

> After the installation the instances boot with the random names. Reset the names as we set in DNS and restart the kubelet service on all hosts.

```
# systemctl restart kubelet
```

> After that just approve the pending CSR's

```
# oc get csr -o name | xargs oc adm certificate approve
```