**https://docs.openshift.com/container-platform/4.11/installing/installing\_bare\_metal/installing-bare-metal.html**

**Machine requirements for a cluster for user-provisioned infrastructure**

Required machines:

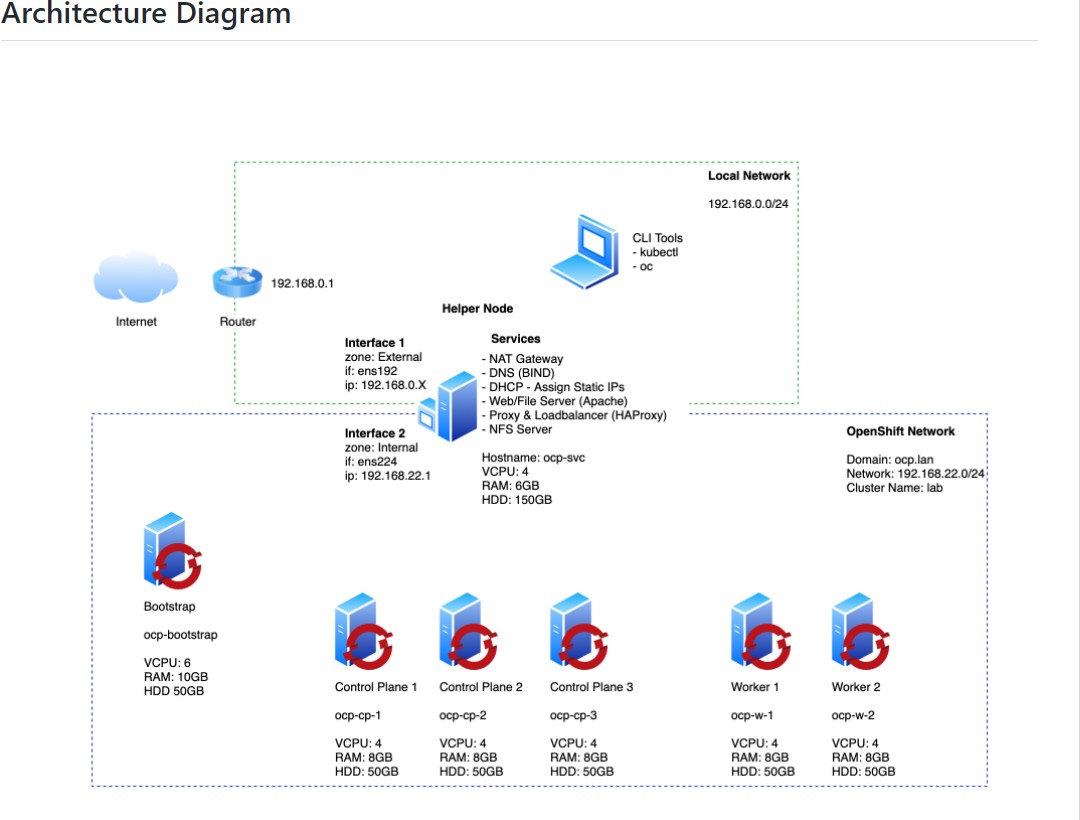
The smallest OpenShift Container Platform clusters require the following hosts:

* One temporary bootstrap machine
* Three control plane, or master machines
* At least two compute machines, which are also known as worker machines.
* One helper node, what we call a bastion in our environment.

**Hardware requirements:**

OCP-Hardware Prerequisites:

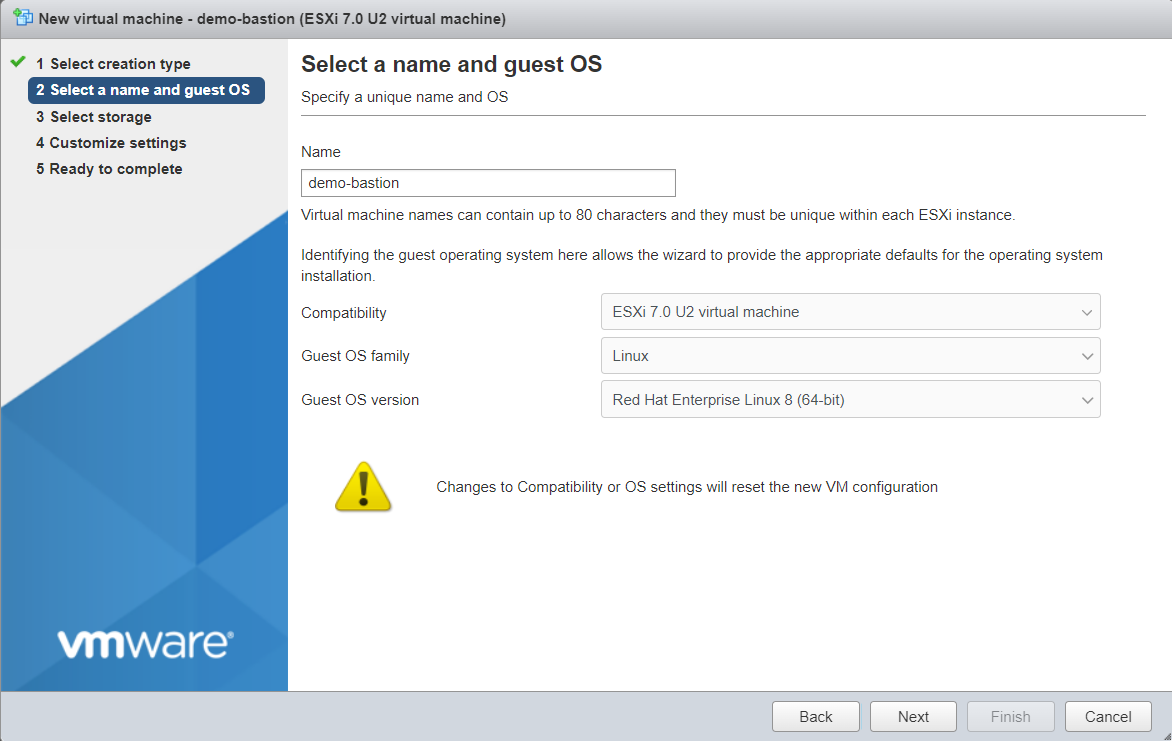
| Node Type | Operating System | RAM | Storage | CPU |
| --- | --- | --- | --- | --- |
| Master1 | RHCOS | 16GB | 120GB | 4 |
| Master2 | RHCOS | 16GB | 120GB | 4 |
| Master3 | RHCOS | 16GB | 120GB | 4 |
| Worker1 | RHCOS | 8GB | 120GB | 2 |
| Worker2 | RHCOS | 8GB | 120GB | 2 |
| Bootstrap | RHCOS | 16GB | 120GB | 4 |
| Bastion | RHEL/Centos | 8GB | 60GB | 2 |
| Total | | 88GB | 780GB | 22 |

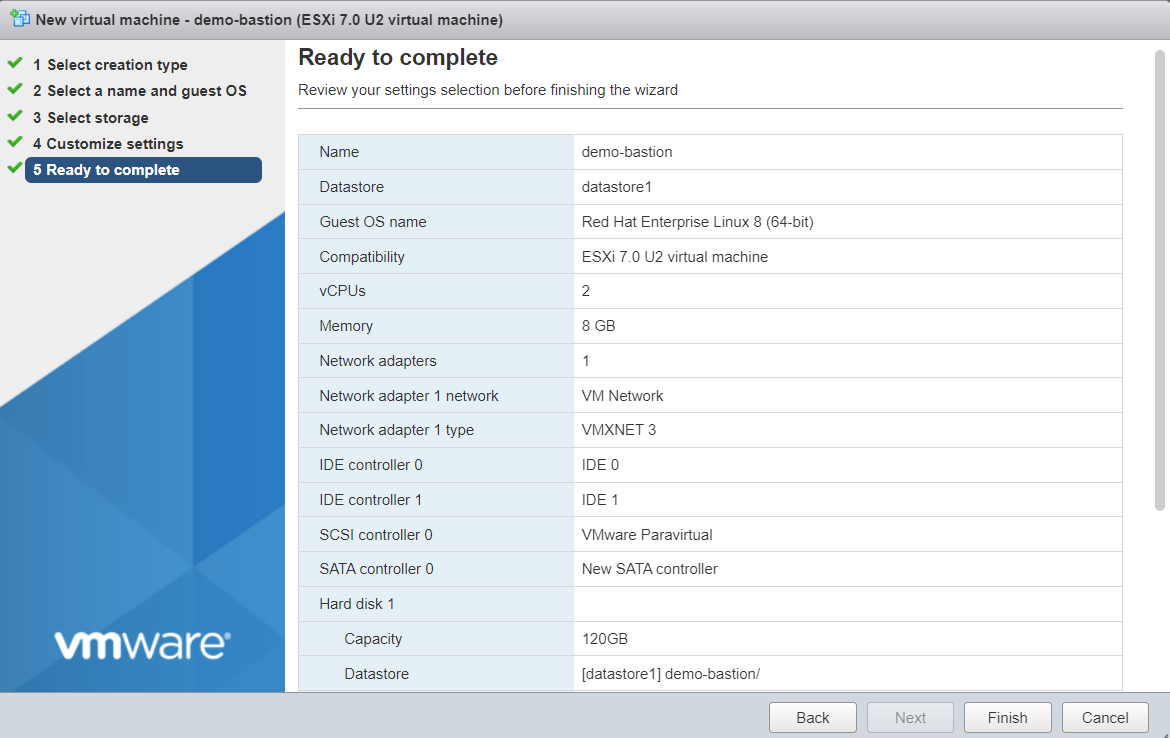


**Let us start with the deployment:**

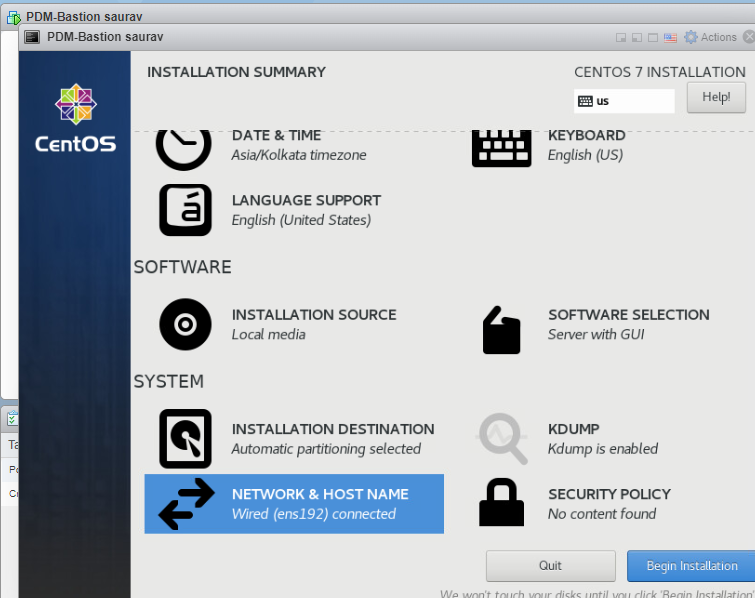
**Creating bare metal environment:**

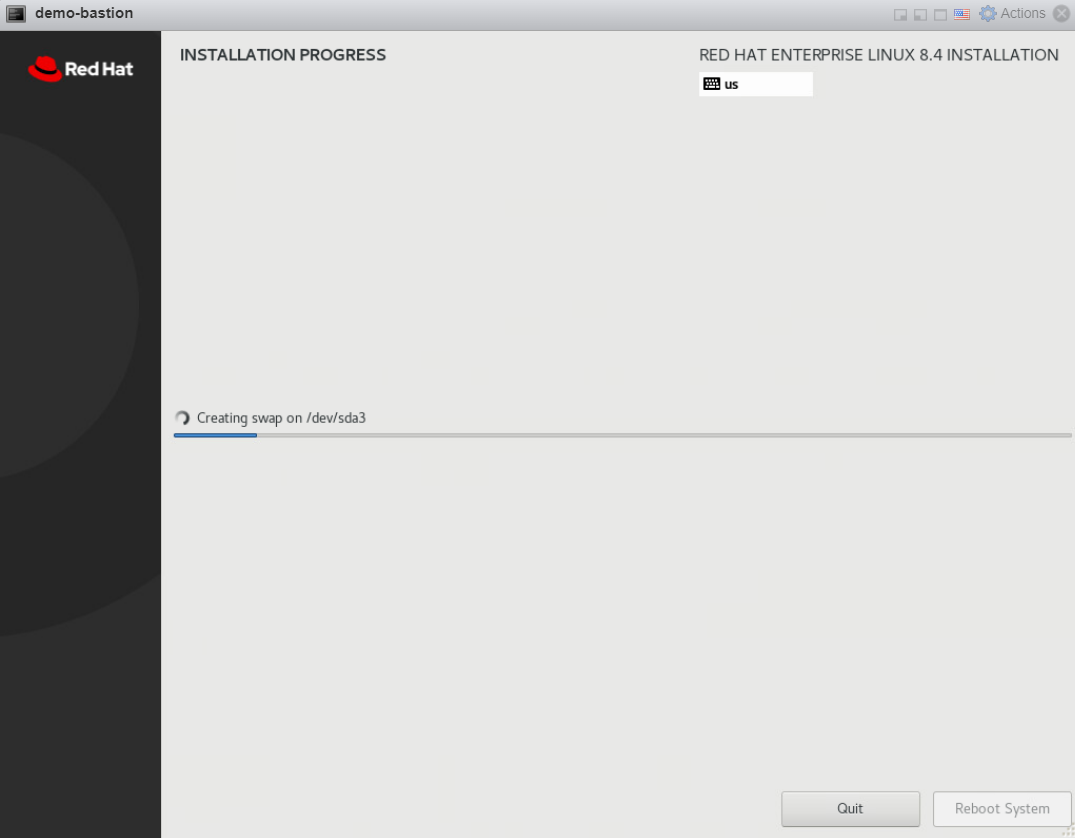
Create the bastion/helper node with the mentioned hardware specification





Then complete the basic installation process:

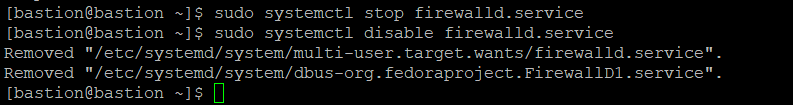




**Then disable firewalld and selinux using this command:**

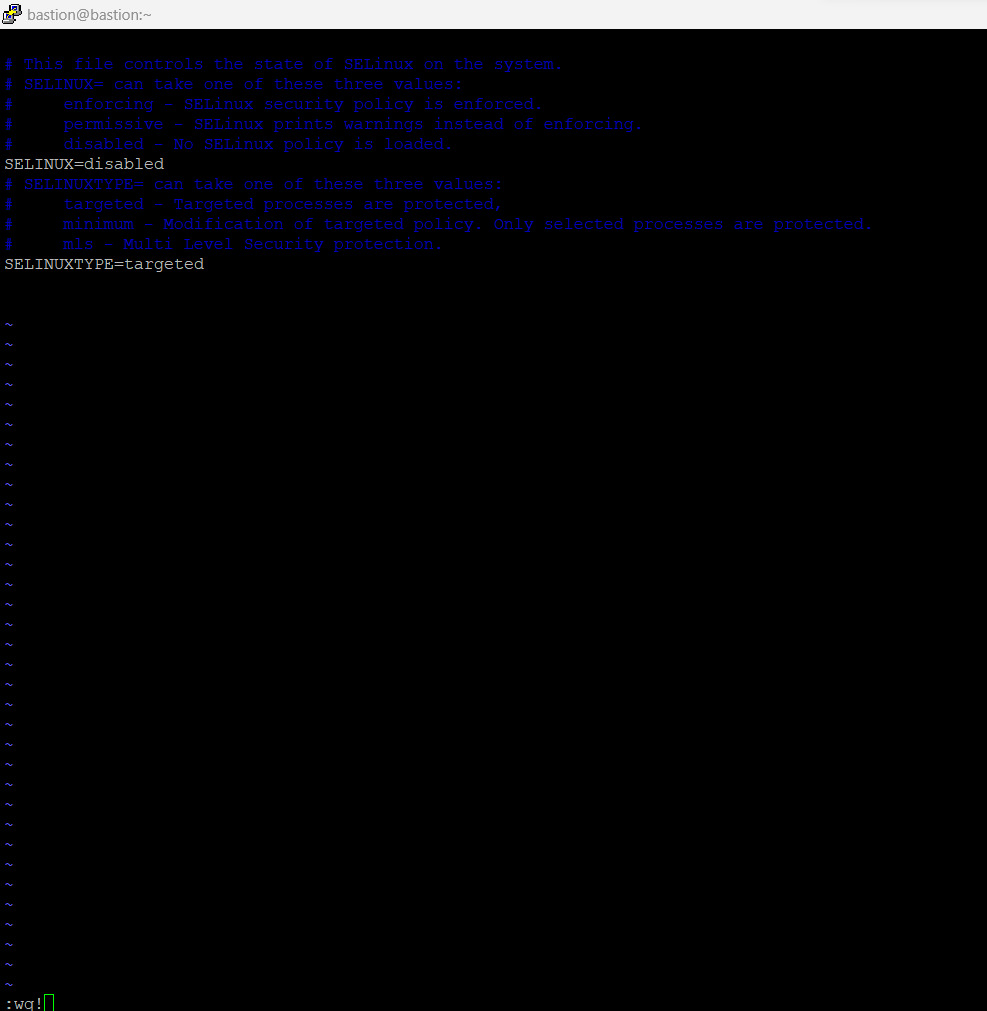
**#sudo systemctl stop firewalld.service**

**#sudo systemctl disable firewalld.service**



**#sudo vim /etc/selinux/config**





**Then reboot #reboot**

**# subscription-manager register**

**#subscription-manager attach**

**update packages on bastion using this command:**

**# sudo yum update && sudo yum upgrade -y**

**After full upgrade, install the following packages in Bastion node.using this command:**

**#sudo yum install httpd -y**

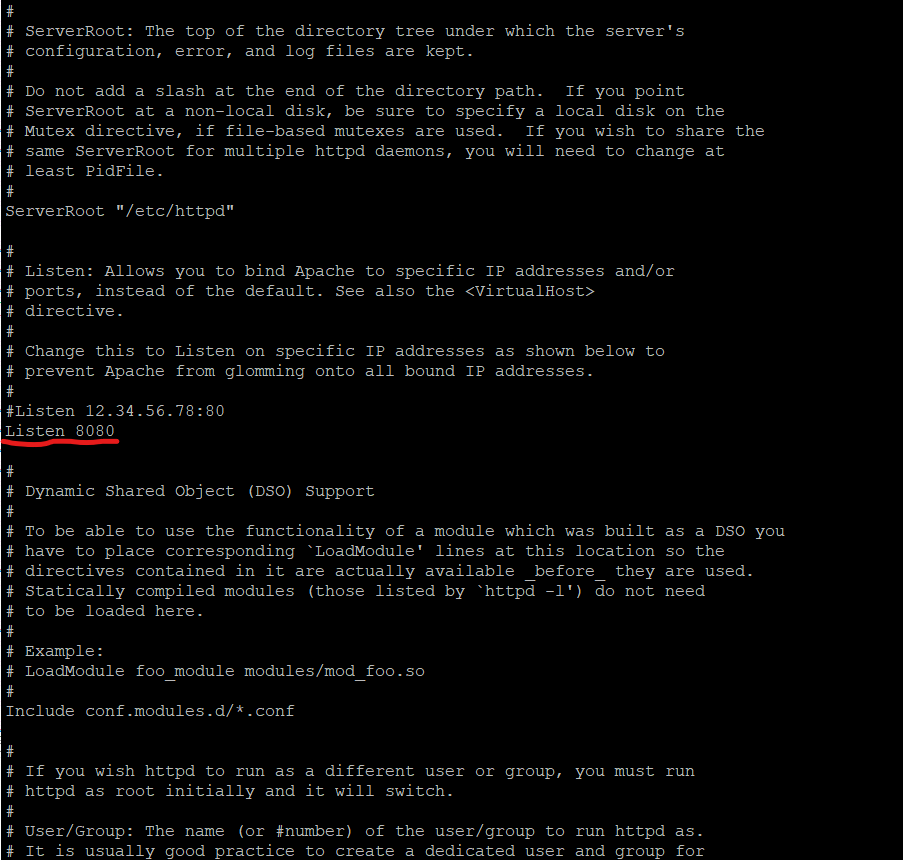
**# sudo yum install haproxy -y**

**# sudo yum install bind -y**

**now start configure DNS**

**change httpd listen port 80 to 8080 using this command:**

**# vim /etc/httpd/conf/httpd.conf**

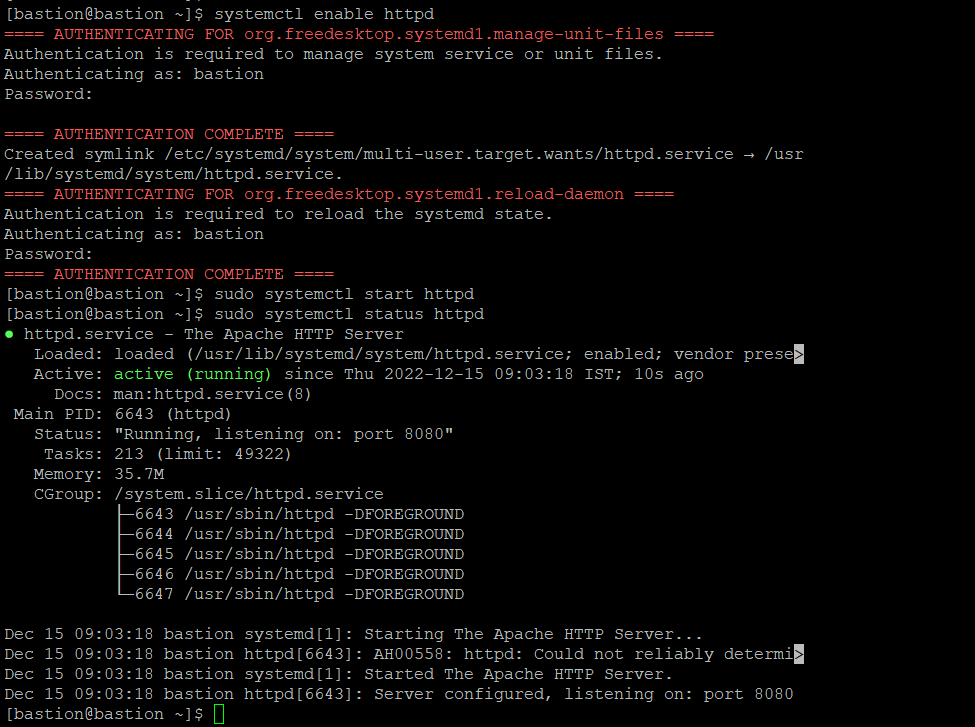


**Then enable ,start, and check status by using this command:**

**# systemctl enable httpd**

**# systemctl start httpd**

**# systemctl status httpd**



**Configure forward dns:**

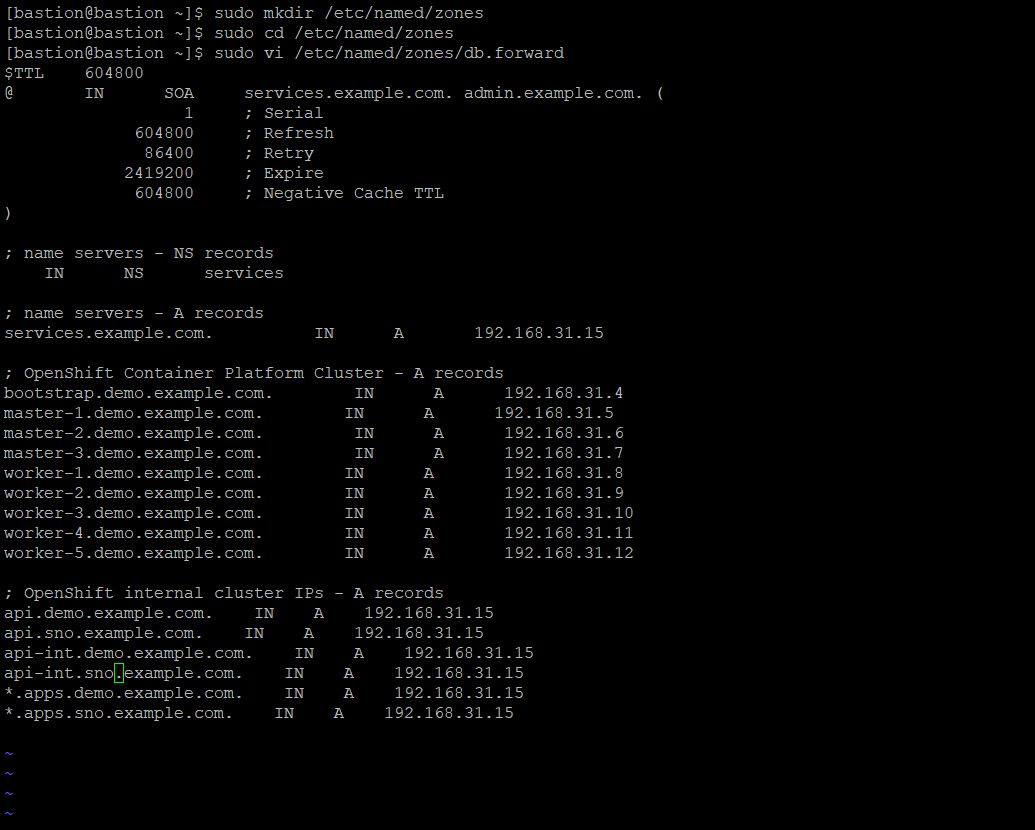
**Create a folder zones inside /etc/named/**

**#sudo mkdir /etc/named/zones**

**#sudo vim /etc/named/zones/db.forward**



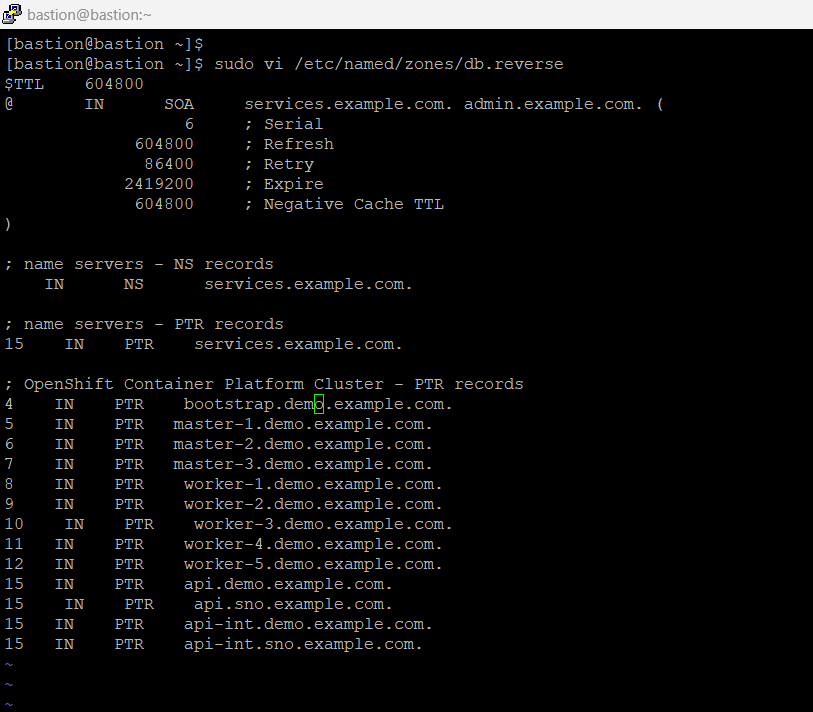
Copy the content from saved file forward-DNS in your folder and paste here

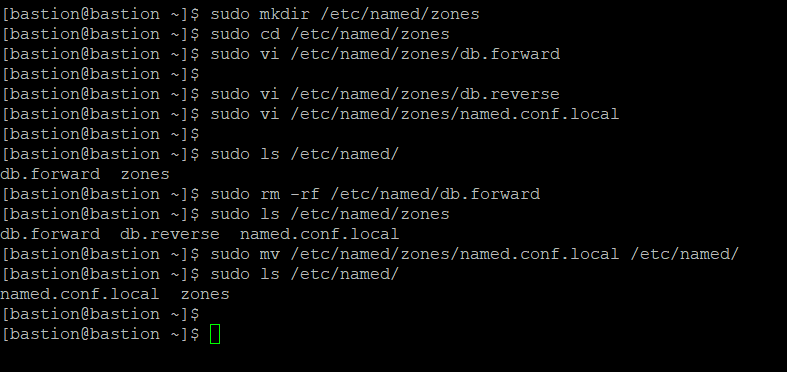


**Configure reverse dns:**

**# sudo vim /etc/named/zones/db.reverse**

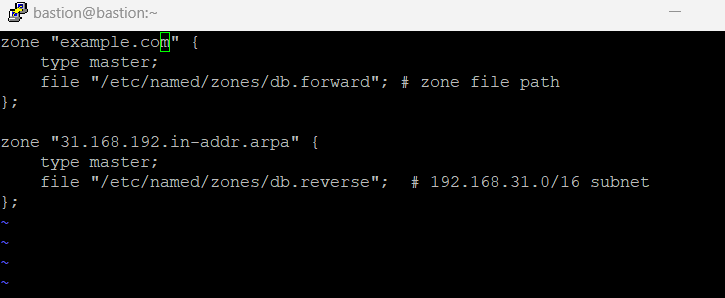
Copy the content from saved file reverse-DNS in your folder and paste here





**Then create file vim /etc/named/named.conf.local and place both the forward** **and reverse DNS entries in it**

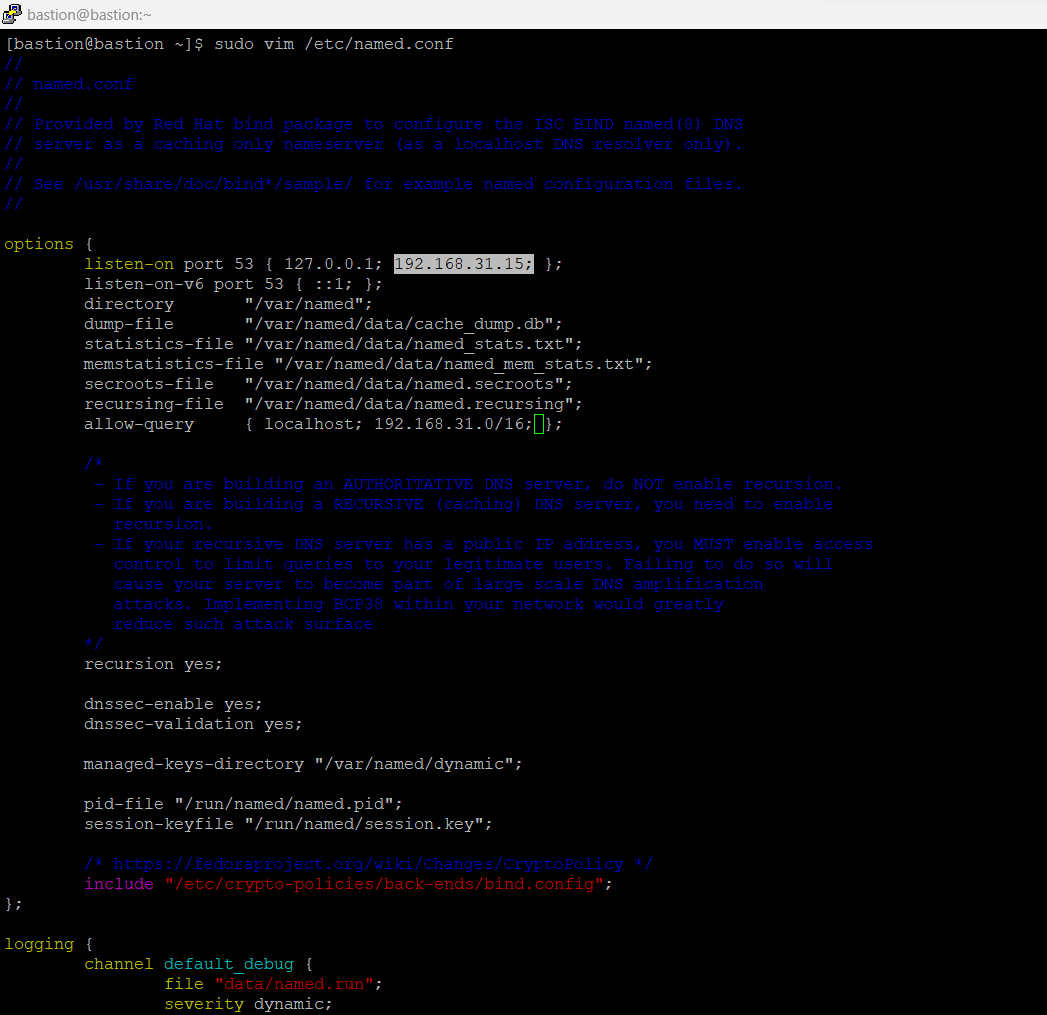
Copy the content from saved file named.conf.local in your folder and paste here



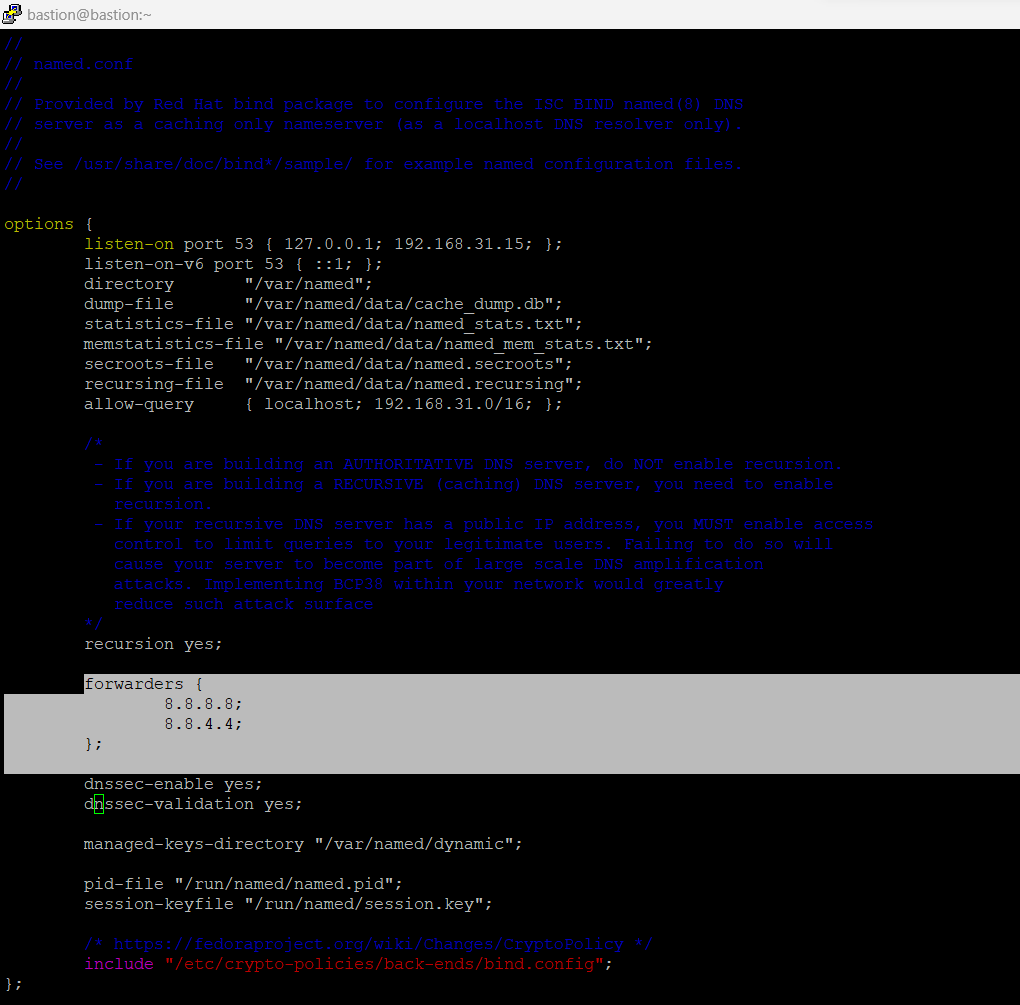
**Then Configure the named.config file using like this :**

**# vim /etc/named.conf**

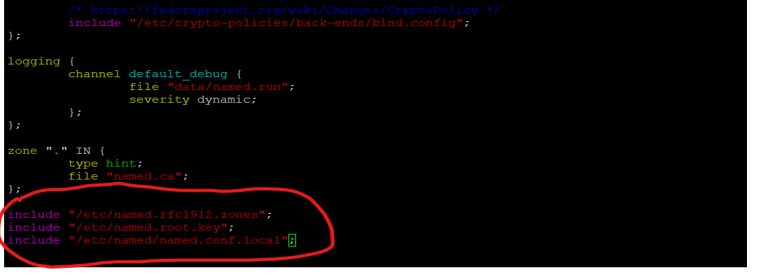
Copy the content from saved file named.conf in your folder and paste here



**Add forwarders 8.8.8.8 ,8.8.4.4**

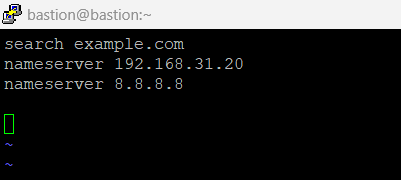


**Add one more line at the end :**



**After complete this configuration.**

**Add the local ip address or ur bastion ip in vim /etc/resolve.conf using this command :**

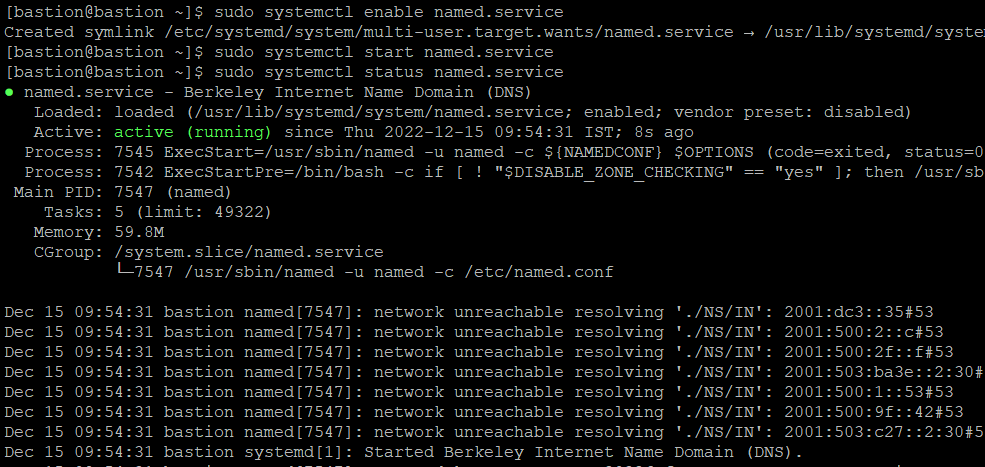


**Then enable, start, and check status of named service by using this command:**

**# sudo systemctl enable named.service**

**# sudo systemctl start named.service**

**# sudo systemctl status named.service**



Verify DNS:

#nslookup 192.168.31.90

#nslookup <bootstrap ip>

Note: if there is no output your DNS configuration is wrong, so don’t proceed to next step

Also,

**Then use these commands to verify DNS configuration setup**

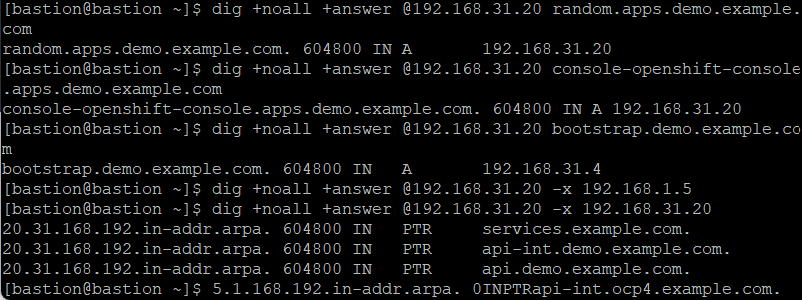
*Procedure*

1. Perform a lookup against the Kubernetes API record name. Check that the result points to the IP address of the API load balancer:

**$ dig +noall +answer @<nameserver\_ip> api.<cluster\_name>.<base\_domain>**

*Example output*

**api.ocp4.example.com. 0 IN A 192.168.1.5**



1. Perform a lookup against the Kubernetes internal API record name. Check that the result points to the IP address of the API load balancer:

**$ dig +noall +answer @<nameserver\_ip> api-int.<cluster\_name>.<base\_domain>**

*Example output*

**api-int.ocp4.example.com. 0 IN A 192.168.1.5**

1. Test an example \*.apps.<cluster\_name>.<base\_domain> DNS wildcard lookup. All of the application wildcard lookups must resolve to the IP address of the application ingress load balancer:

**$ dig +noall +answer @<nameserver\_ip> console-openshift-console.apps.<cluster\_name>.<base\_domain>**

*Example output*

**random.apps.ocp4.example.com. 0 IN A 192.168.1.5**

1. Run a lookup against the bootstrap DNS record name. Check that the result points to the IP address of the bootstrap node:

**$ dig +noall +answer @<nameserver\_ip> bootstrap.<cluster\_name>.<base\_domain>**

*Example output*

**bootstrap.ocp4.example.com. 0 IN A 192.168.1.96**

1. Use this method to perform lookups against the DNS record names for the control plane and compute nodes. Check that the results correspond to the IP addresses of each node.
2. Perform a reverse lookup against the IP address of the API load balancer. Check that the response includes the record names for the Kubernetes API and the Kubernetes internal API:

**$ dig +noall +answer @<nameserver\_ip> -x 192.168.1.5**

*Example output*

**5.1.168.192.in-addr.arpa. 0 IN PTR api-int.ocp4.example.com.**

**5.1.168.192.in-addr.arpa. 0 IN PTR api.ocp4.example.com.**

1. Perform a reverse lookup against the IP address of the bootstrap node. Check that the result points to the DNS record name of the bootstrap node:

**$ dig +noall +answer @<nameserver\_ip> -x 192.168.1.96**

*Example output*

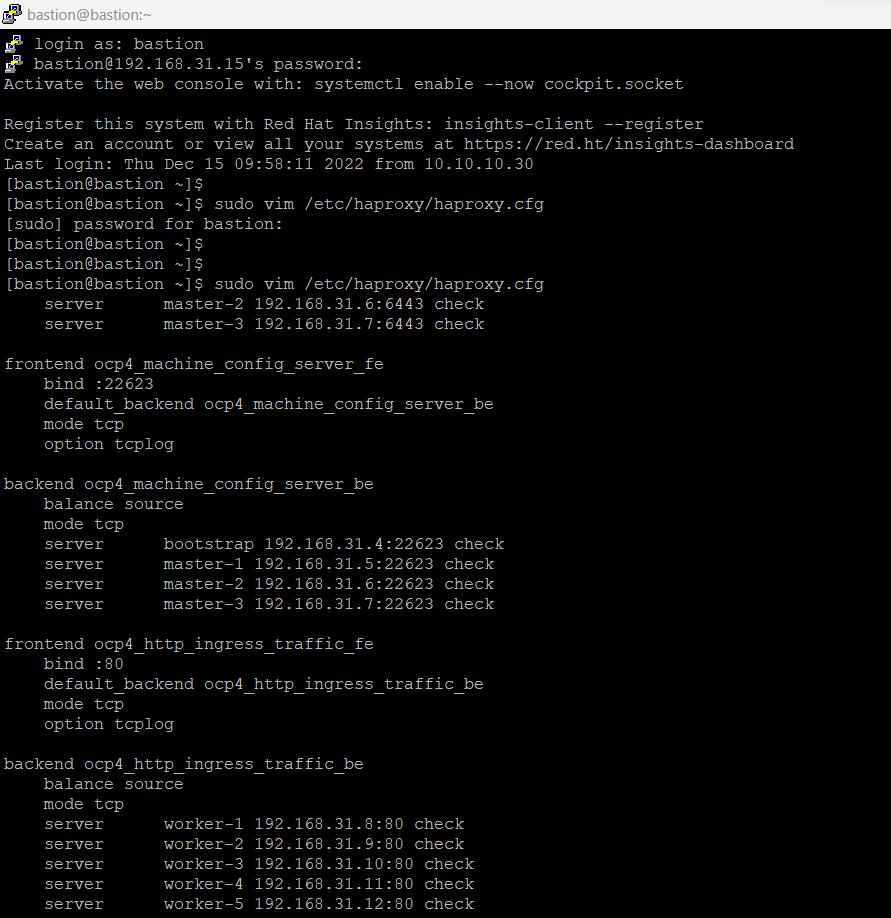
**96.1.168.192.in-addr.arpa. 0 IN PTR bootstrap.ocp4.example.com.**

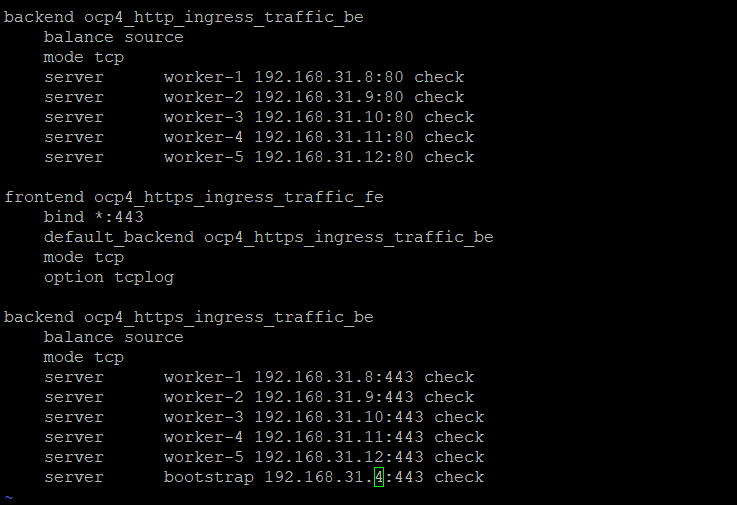
**Then edit & configure the haproxy**

**# sudo vim /etc/haproxy/haproxy.cfg**



Copy the content from saved file Haproxy.cfg in your folder and paste here



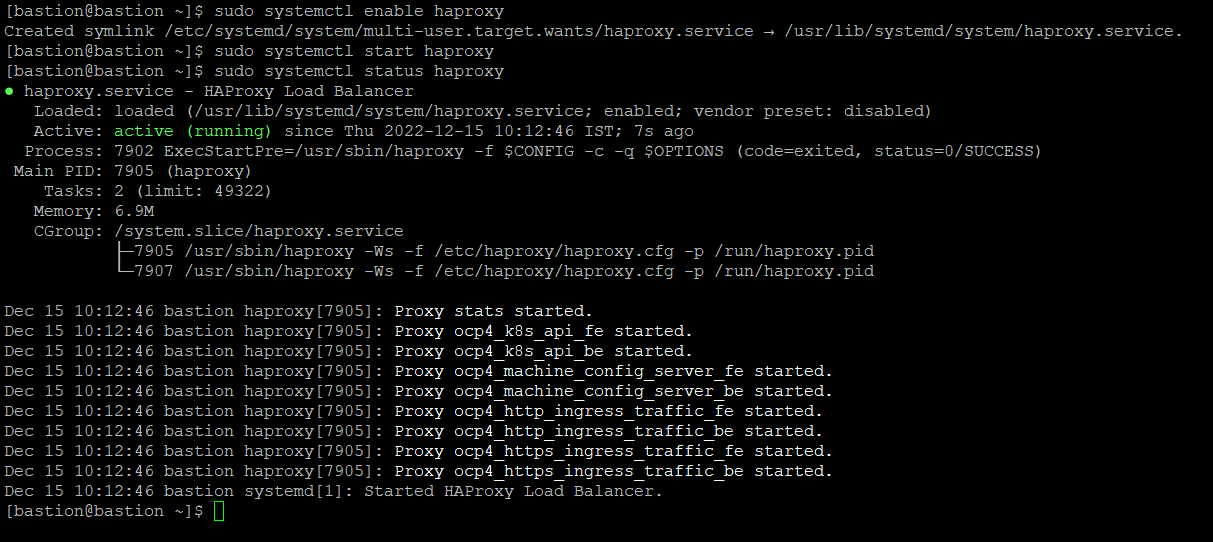


**Then start the haproxy service and enable it and check its status**

# sudo **systemctl enable haproxy**

# sudo **systemctl start haproxy**

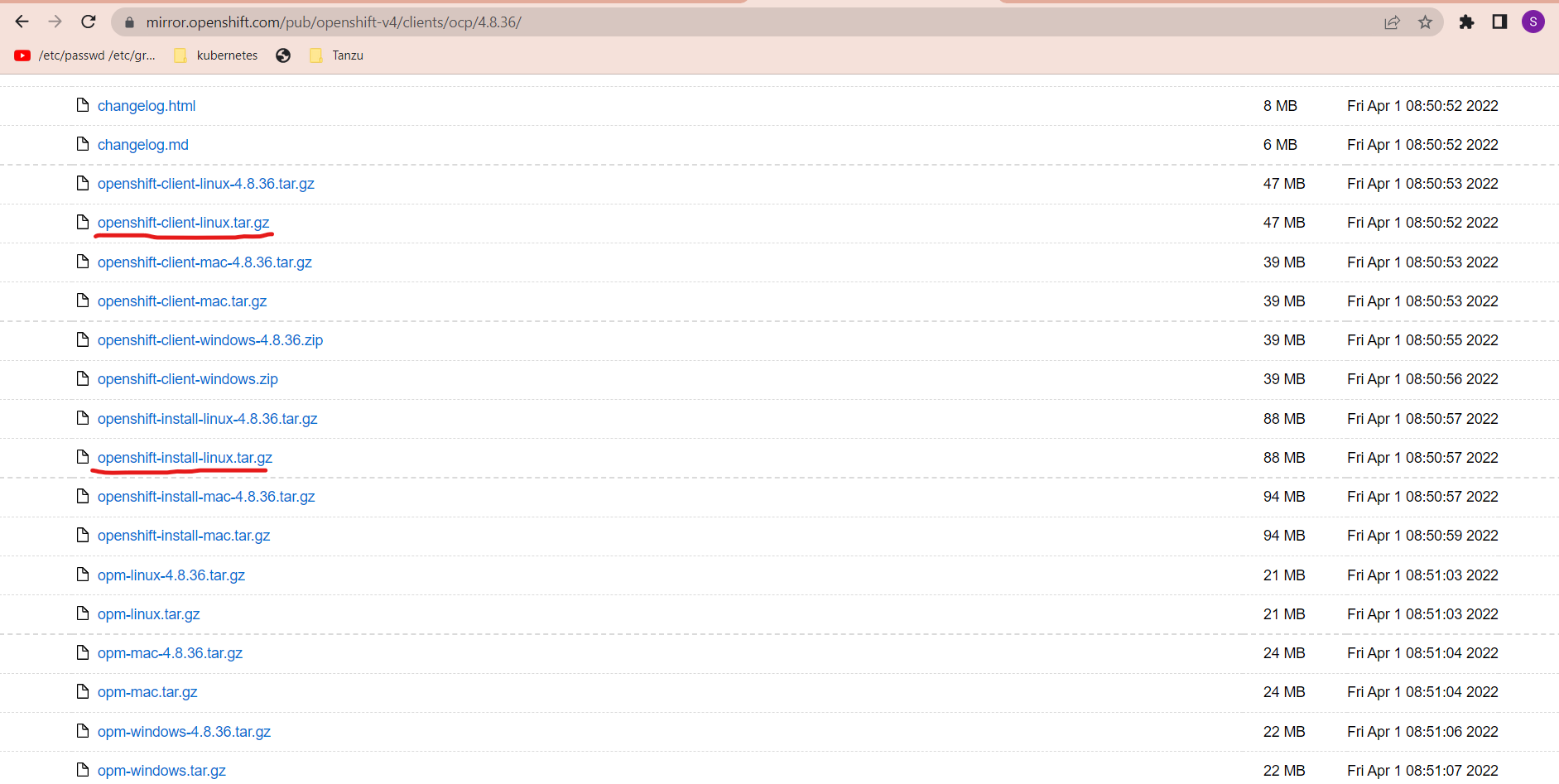
# sudo **systemctl status haproxy**

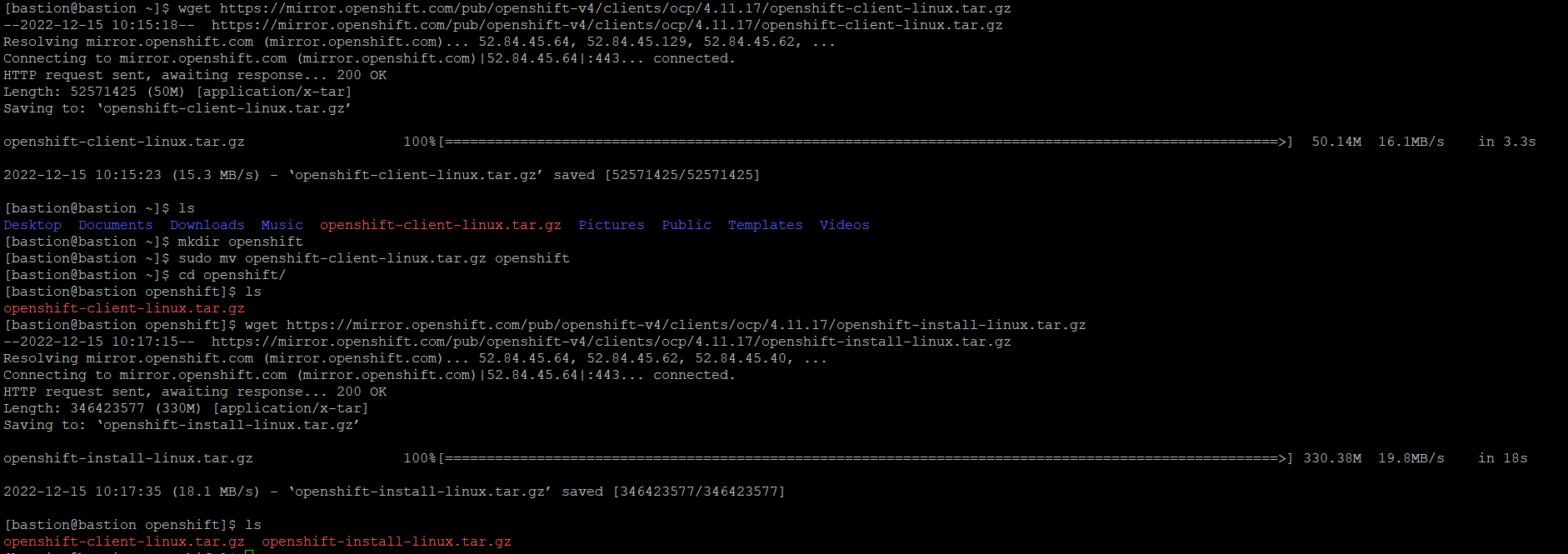


**Download the open shift-install & open shift-client-Linux from below site:**

[**https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.8.36/**](https://mirror.openshift.com/pub/openshift-v4/clients/ocp/4.8.36/)

**#wget <download link>**

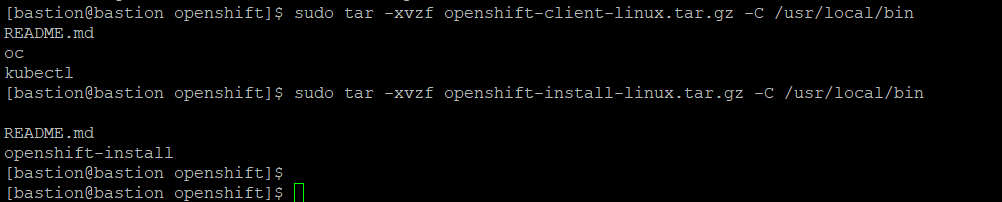




**Extract both the file and place it in /usr/local/bin directory using this command:**

**#sudo tar –xzvf openshift-client-linux.tar.gz -C /usr/local/bin**

**#sudo tar –xzvf openshift-install-linux.tar.gz -C /usr/local/bin**

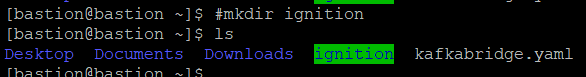


**If you want to check then follow these command :**

**#ls /usr/local/bin | grep oc**

**After creating all the configuration files, create one installation directory Ignition and create install-config.yaml file in it.(with the help of github)**

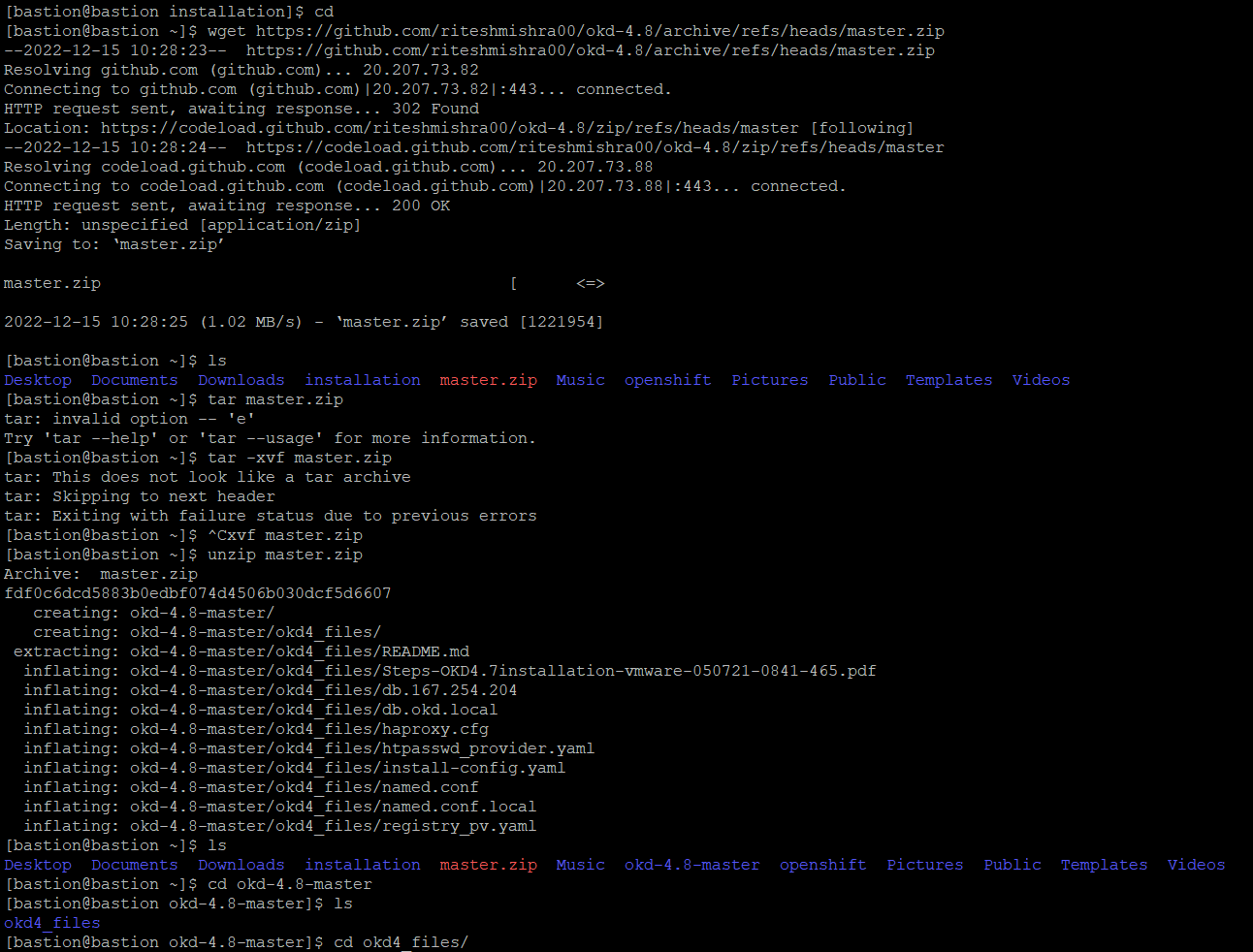
**#mkdir ignition**



<https://github.com/riteshmishra00/okd-4.8.git>

**#wget** [**https://github.com/riteshmishra00/okd-4.8**](https://github.com/riteshmishra00/okd-4.8)**/archieve/refs/heads/master.zip**

**#unzip master.zip**

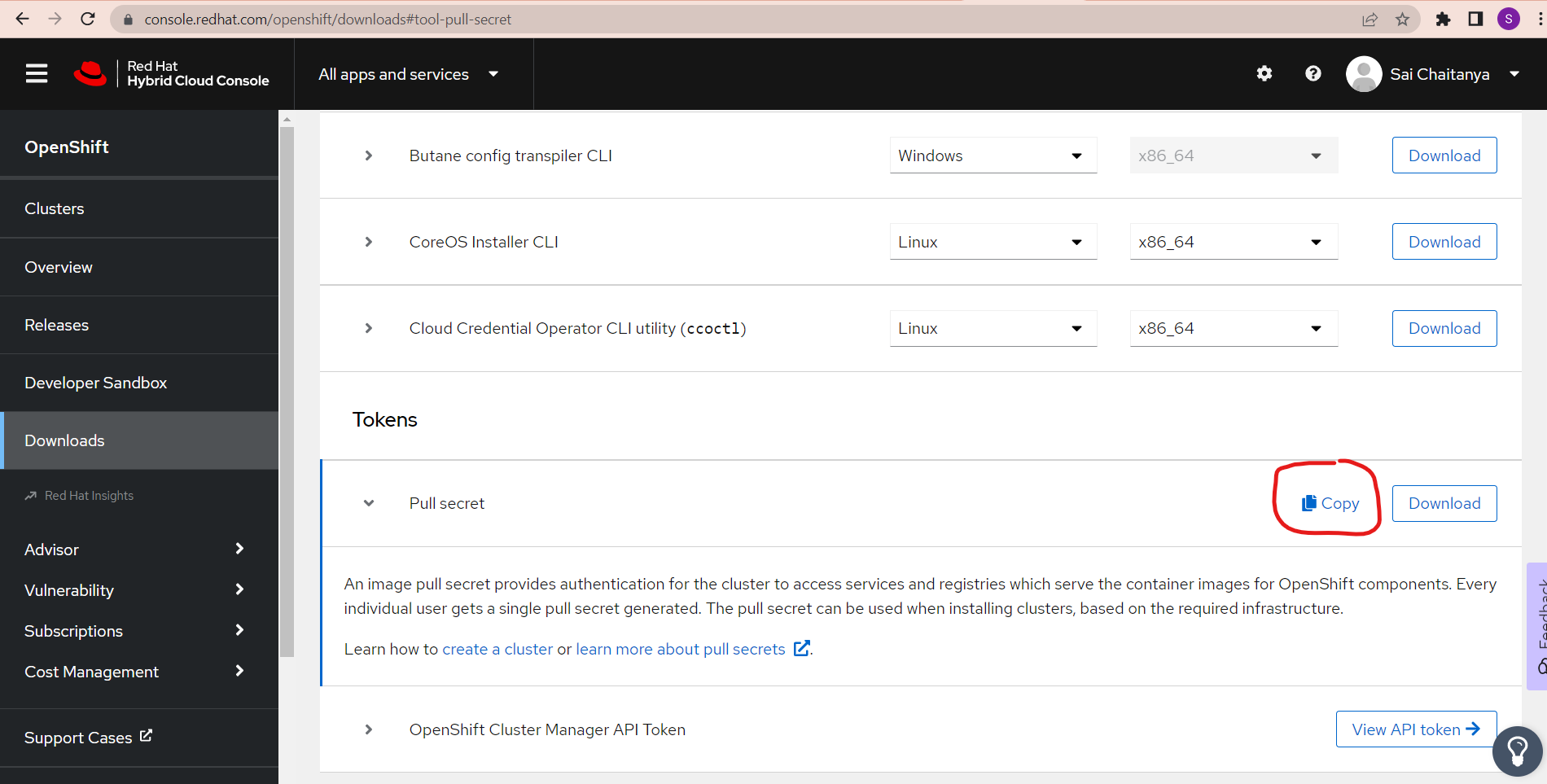


Copy the install-config.yaml file from unzipped folders, to the installation directory **Ignition**

Open the install-config.yaml file

**#sudo vi install-config.yaml**

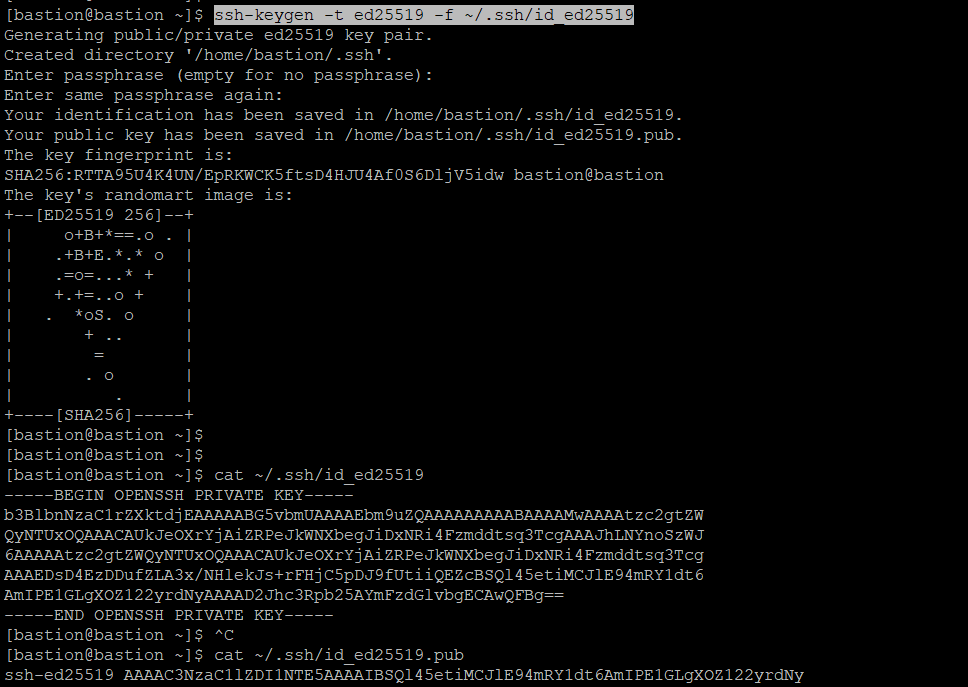
Change the domain name and cluster name, then Go to console.redhat.com 🡪 login 🡪 openshift 🡪 Downloads 🡪scroll down to pull secret 🡪 Copy



Paste this here, pullSecret: 'here'

**Create one ssh key by using the following command**

**# ssh-keygen -t ed25519 -f ~/.ssh/id\_ed25519**



**Copy the public key from the .ssh/id\_rsa.pub file and paste it into the install-config.yaml file**

**# cat .ssh/id\_rsa.pub**



Start

#cd ignition

#openshift-install create manifests

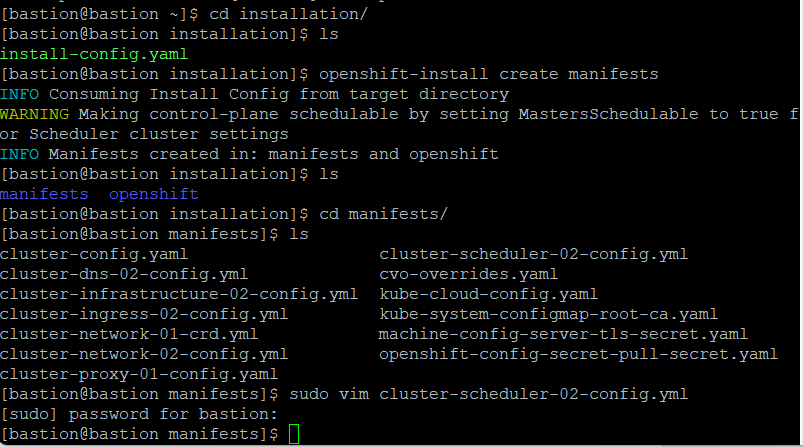
#cd manifests

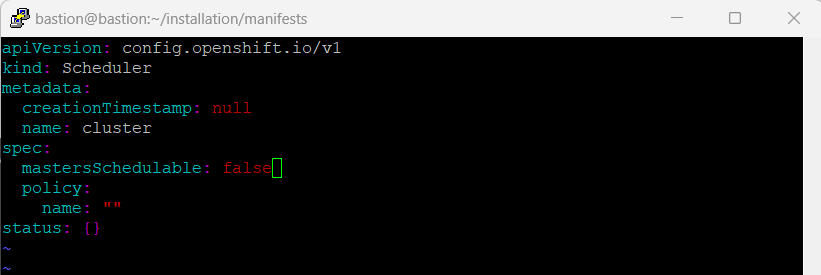
#vi cluster-scheduler-o2-config.yml

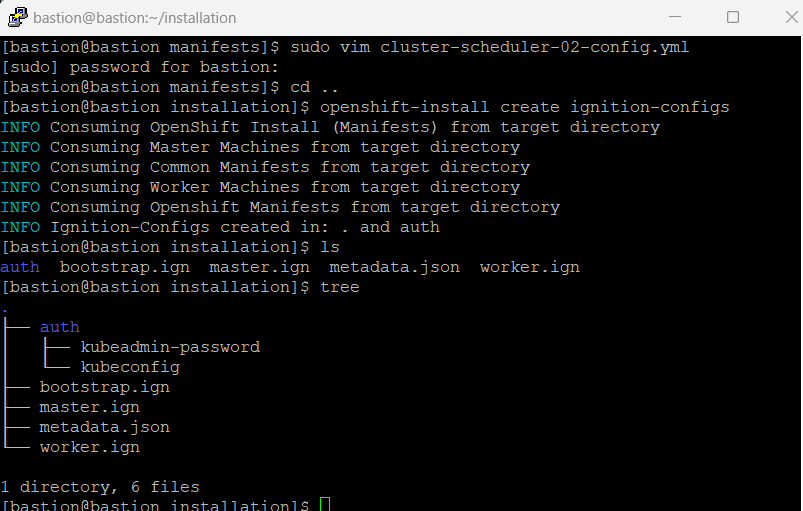
Masterschedulable : false

#cd ..

#openshift-install create ignition-configs







After creating the Ignition files copy them to http webserver folder

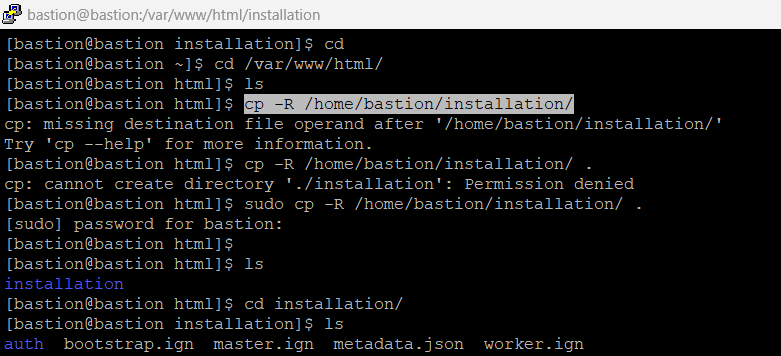
#cd /var/www/html

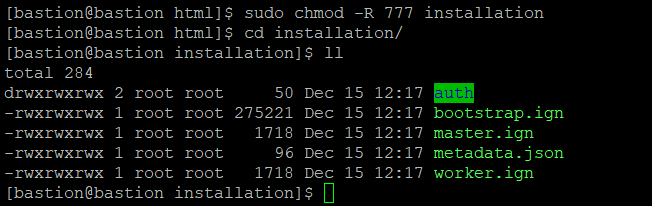
#cp -R /home/bastion/ignition/ .

Change the permission

#sudo chmod -R 777 ignition

Try and verify to access these files from webbrowser http://192.168.31.90:8080/ignition/





Then, Prepare the bootstrap, master, worker Vms and download the ignition file respectively

Set static ip

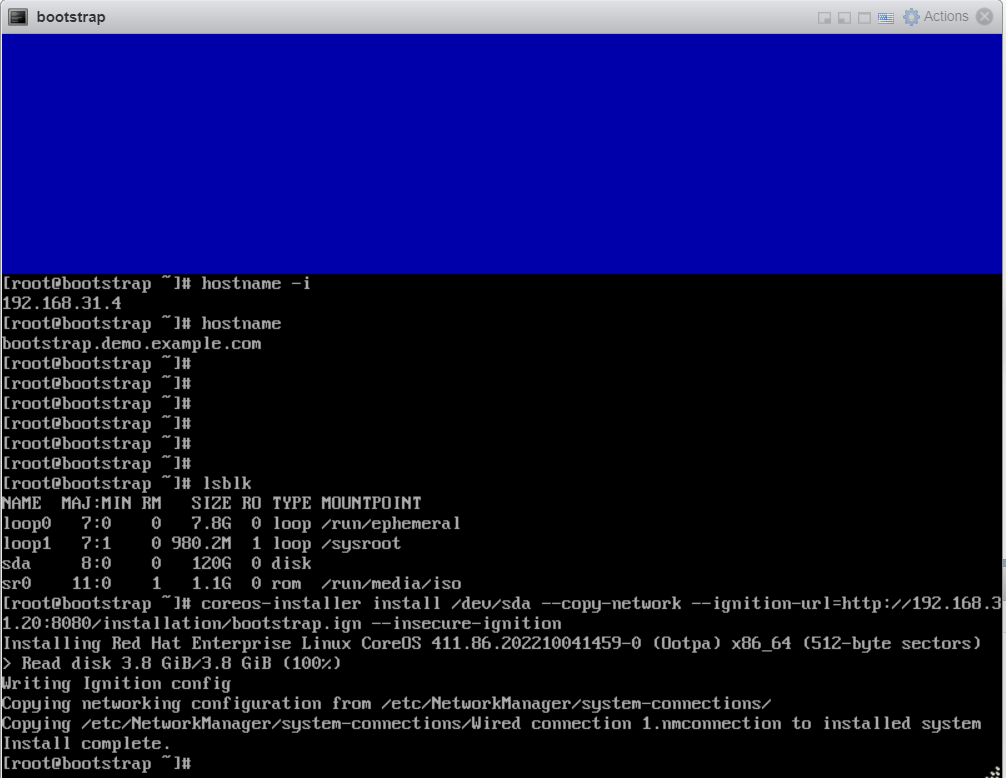
#nmtui

Note: without setting the hostname , Verify hostname of VMs should be assigned from DNS that we configured earlier

#hostname

#sudo -i

#coreos-installer install /dev/sda –copy-network –ignititon-url=http://192.168.31.90:8080/ignition/bootstrap.ign or master.ign or worker.ign –insecure-ignition

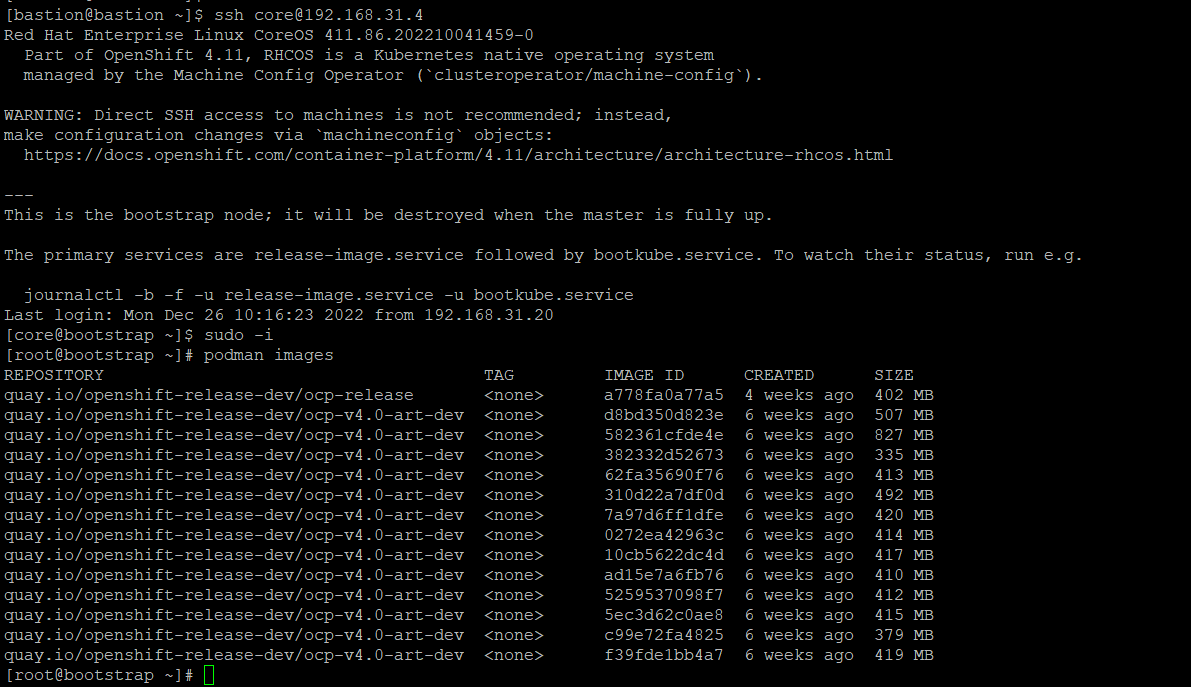


After bootstarp node downloaded ignition file, reboot

Take a putty session of bootstrap by ssh from bastion

#ssh core@<bootstrap-ip>

Run the given journalctl command that appears

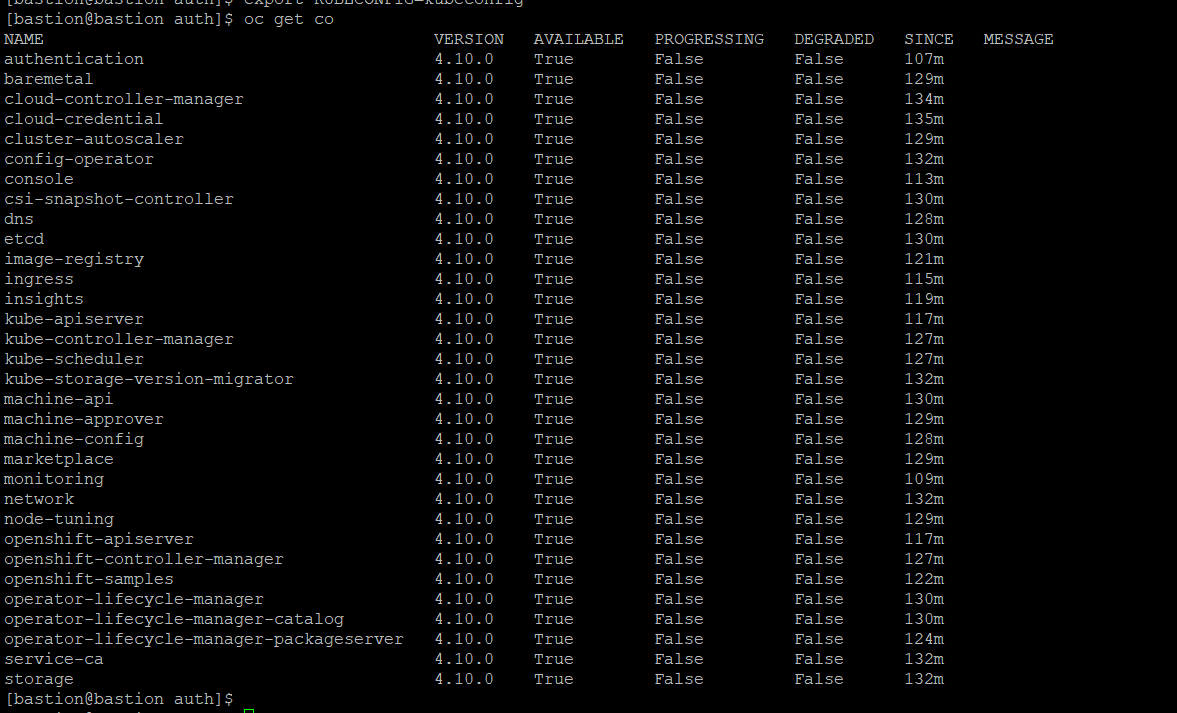


Then restart the master node one by one and wait till they are ready

After that Check the node status in bastion by

#oc get no

#oc get co



Once the master nodes appears, reboot the worker nodes one by one and wait

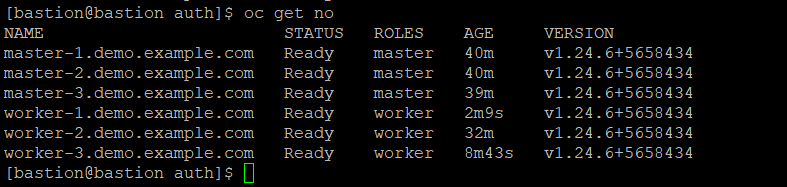
Then check the pending certificates

#oc get csr

# oc adm certificate approve <certificate-name>

#oc get csr -oname | xargs oc adm certificate approve

export KUBECONFIG=kubeconfig



Finally when the cluster is ready finish the bootstrap process

Move to ignition folder and run these commands

#openshift-install wait-for bootstrap-complete –log-level debug

#openshift-install wait-for install-complete!!!

