Create Multi-Master Kubernetes Cluster with Centos 7 on Google Cloud Platform

Contents

Introduction

This document describes the implementation of the Kubernetes cluster with 3 master and 4 worker nodes with a bastion host that acts as a load balancer on Google Cloud Platform (GCP).

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Linux
- Dockers and Kubernetes
- Google Cloud Platform

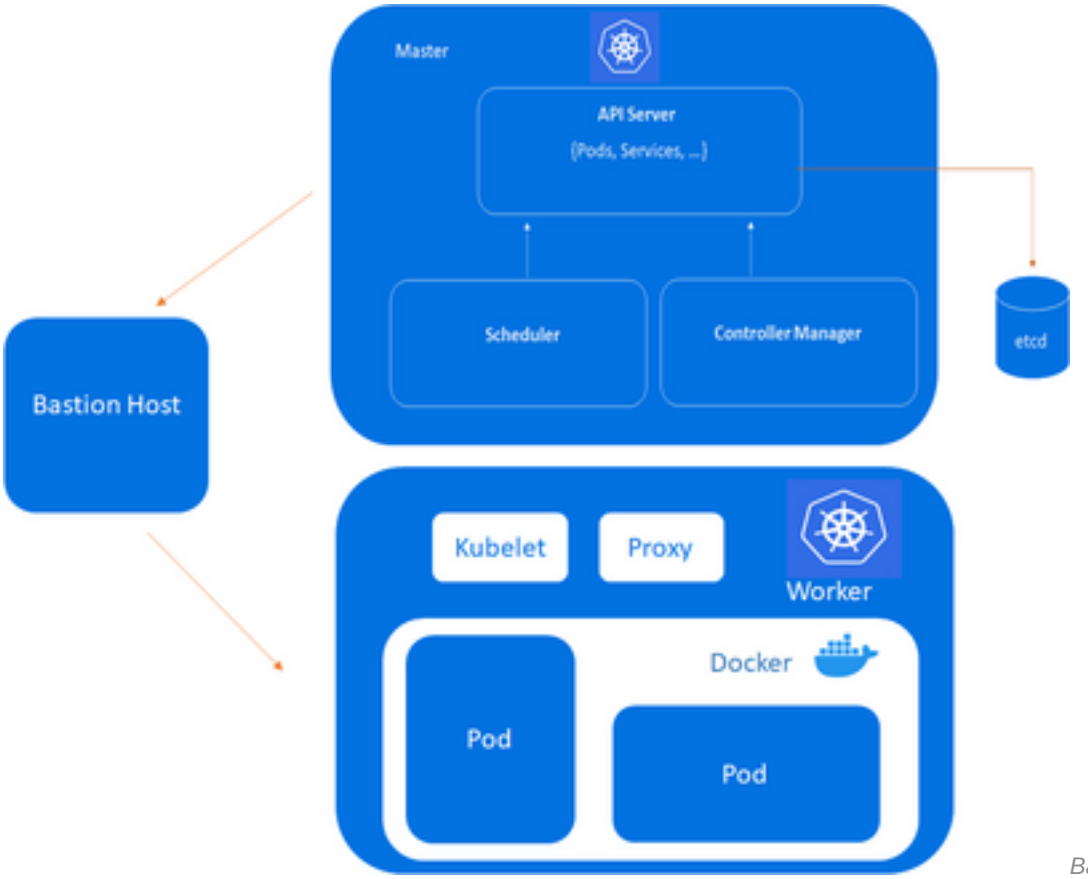
Components Used

The information in this document is based on:

- OS - Centos 7 virtual machine
- Machine Family (e2-standard-16): vCPUs - 16 vCPURAM - 64 GB

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

Network Diagram



Bastion host, Kube-Master, Kube-node

Kubernetes Master Node Components

Kube-apiserver:
i. Provides an API that serves as the front end of a Kubernetes control plane.
ii. It handles external and internal requests that determine whether a request is valid and then processes it.
iii. The API can be accessed via the **kubectl** command-line interface or other tools like **kubeadm**, and via REST calls.

Kube-scheduler:
i. This component schedules pods on specific nodes as per automated workflows and user-defined conditions.

Kube-controller-manager:
i. The Kubernetes controller manager is a control loop that monitors and regulates the state of a Kubernetes cluster.
ii. It receives information about the current state of the cluster and objects within it and sends instructions to move the cluster towards the cluster operator’s desired state.

etcd:
i. A key-value database that contains data about your cluster state and configuration.
ii. Etcd is fault-tolerant and distributed.

Kubernetes Worker Node Components

Kubelet:
i. Each node contains a **kubelet**, which is a small application that can communicate with the Kubernetes control plane.
ii. The **kubelet** ensures that containers specified in pod configuration run on a specific node, and manage their lifecycle.
iii. It executes the actions commanded by your control plane.

Kube-proxy:
i. All compute nodes contain **kube-proxy**, a network proxy that facilitates Kubernetes networking services.
ii. It handles all network communications outside and inside the cluster, and forwards traffic or replies on the packet filtering layer of the operating system.

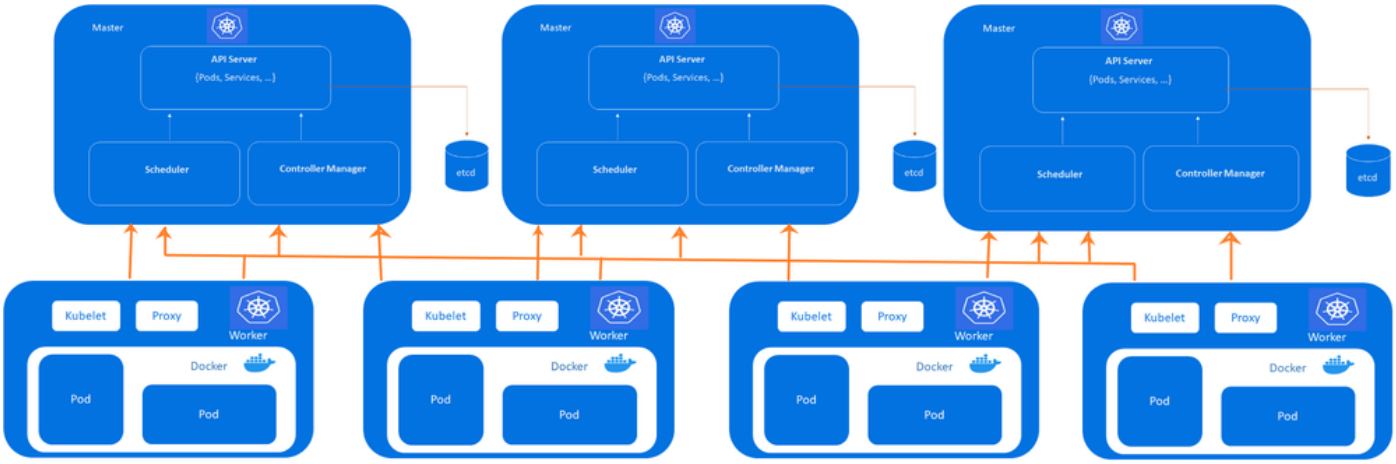
Pods:
i. A pod serves as a single application instance and is considered the smallest unit in the object model of Kubernetes.

Bastion Host:
i. The computer generally hosts a single application or process, for example, a proxy server or load balancer, and all other services are removed or limited to reduce the threat to the computer.

Kubernetes Multi-Master Architecture

Cluster:

- 8 - Total Nodes
- 3 - Master Nodes
- 4 - Worker Nodes
- 1 - Bastion Server (Load Balancer)

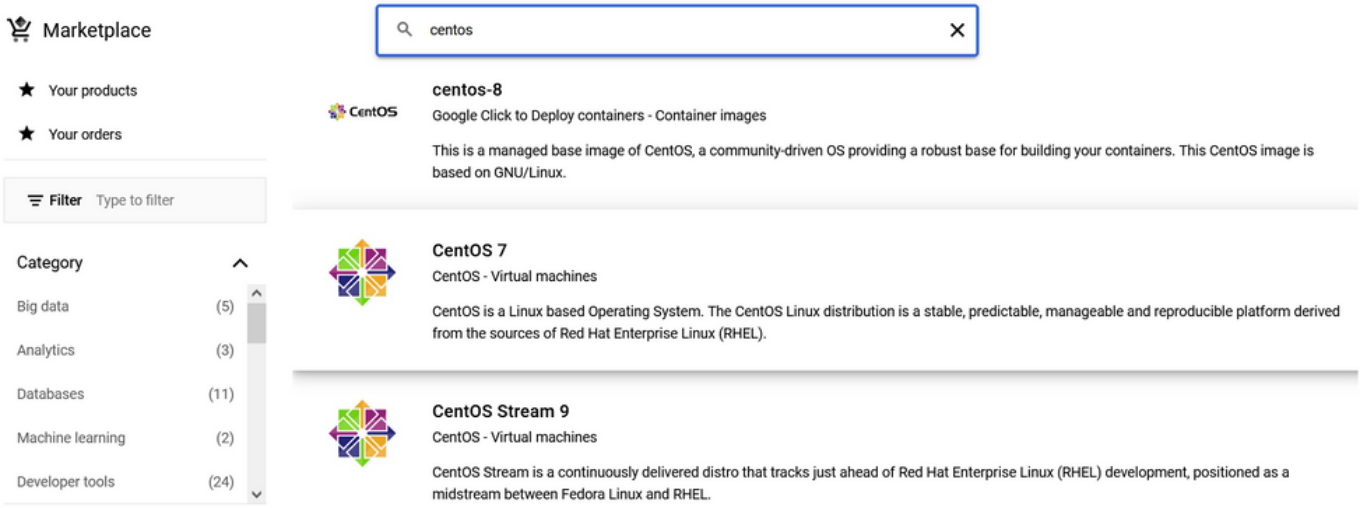


Highly available Kubernetes cluster with three control-planes

Note: Configure VPC on GCP prior to provisioning the VMs. Reference VPC on GCP.

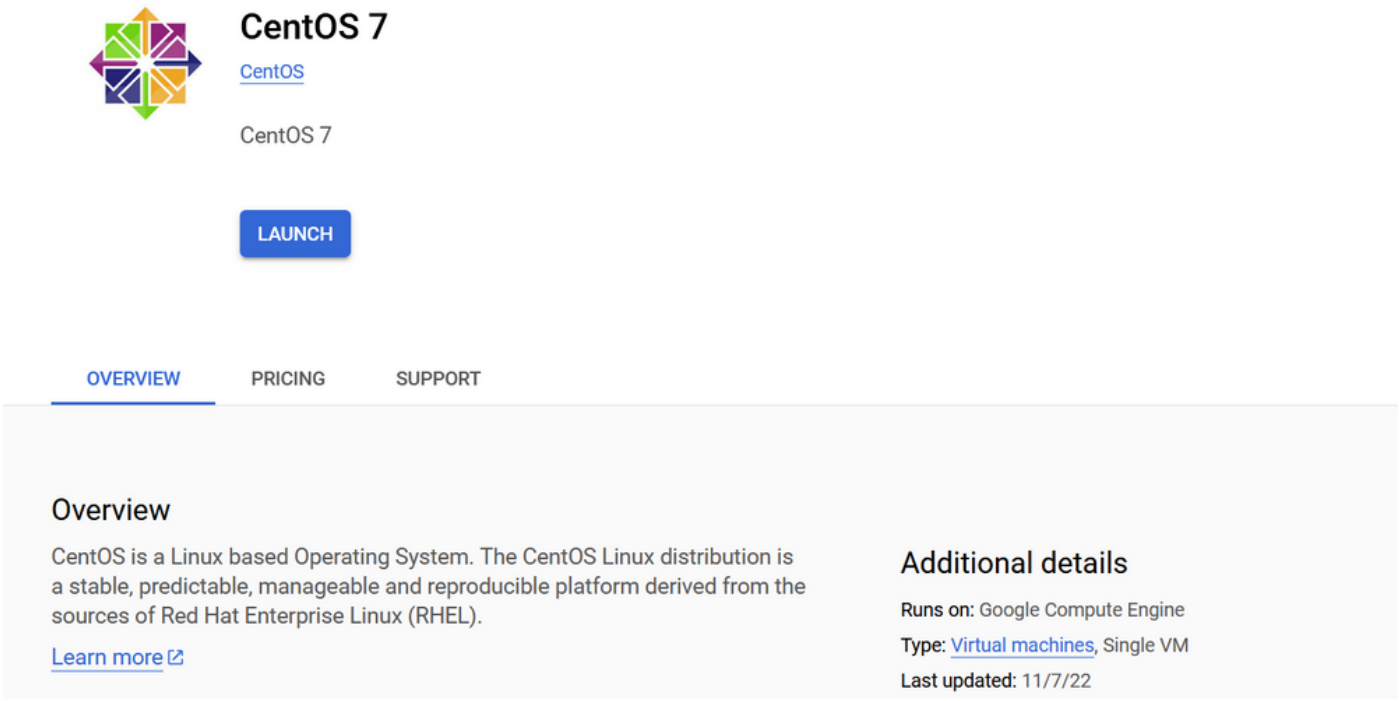
Provision of Virtual Machines on GCP

On GCP provision a Centos7 from the GCP marketplace.



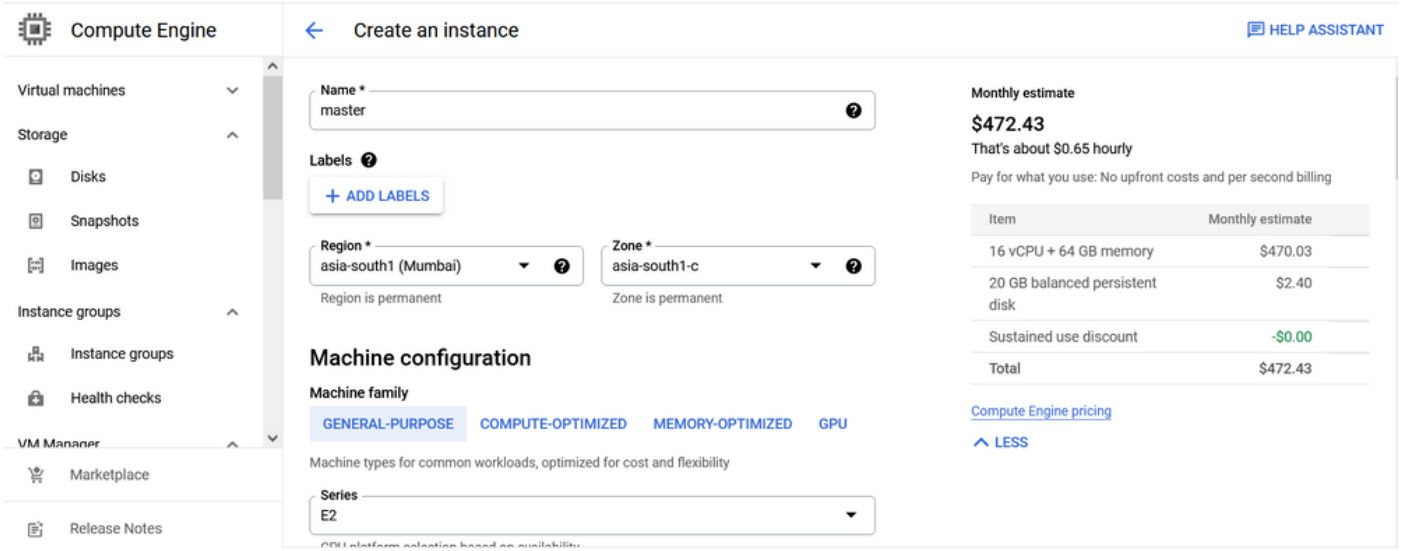
Centos marketplace on GCP

Click **Launch**.



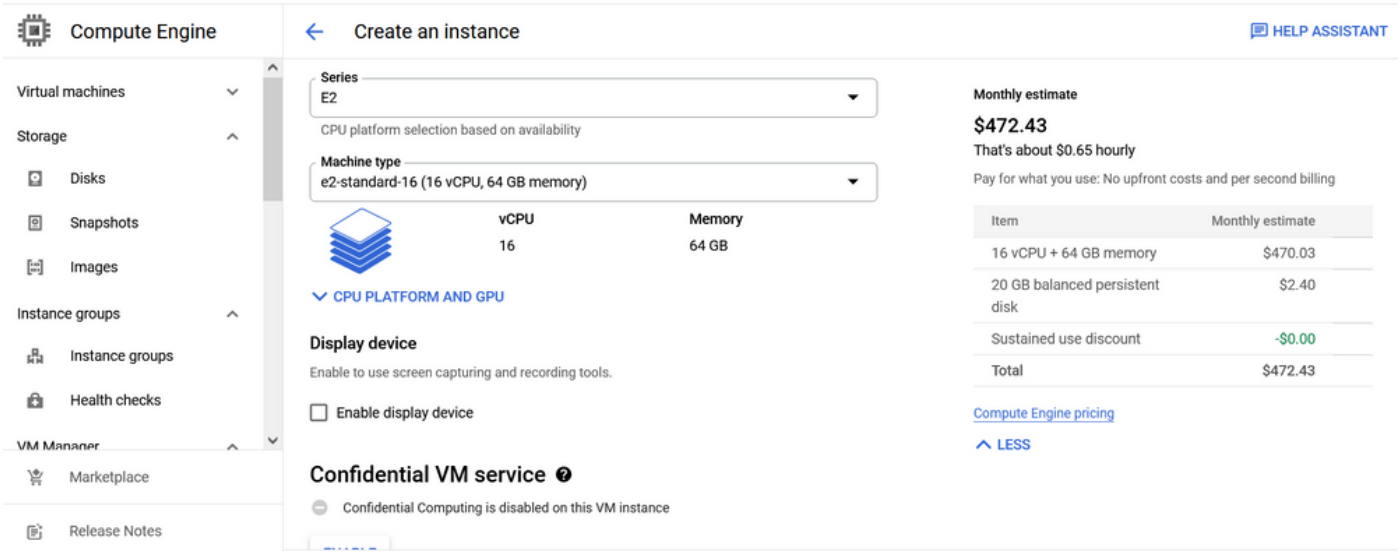
Centos 7 virtual machine

Choose the region as per the nearest reachability. In this lab, the region is configured as Mumbai.



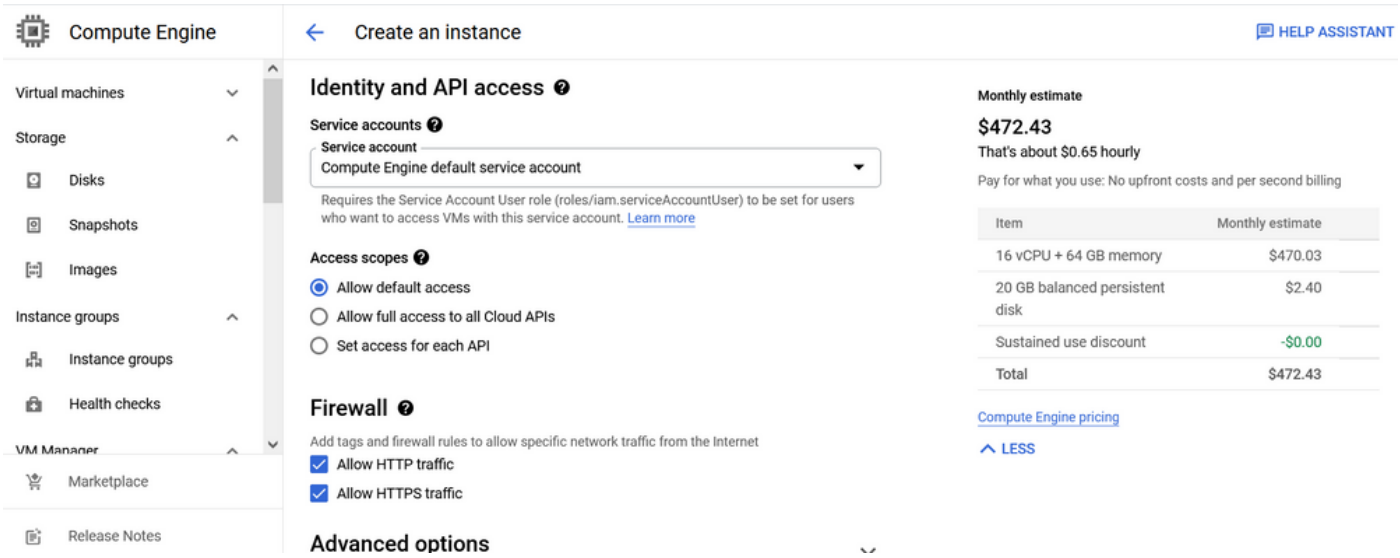
Centos7 compute engine configuration

The machine configuration is general-purpose E2 series and machine-type **e2-standard-16 (16 vCPU, 64 GB memory).**



Centos 7 resource configuration

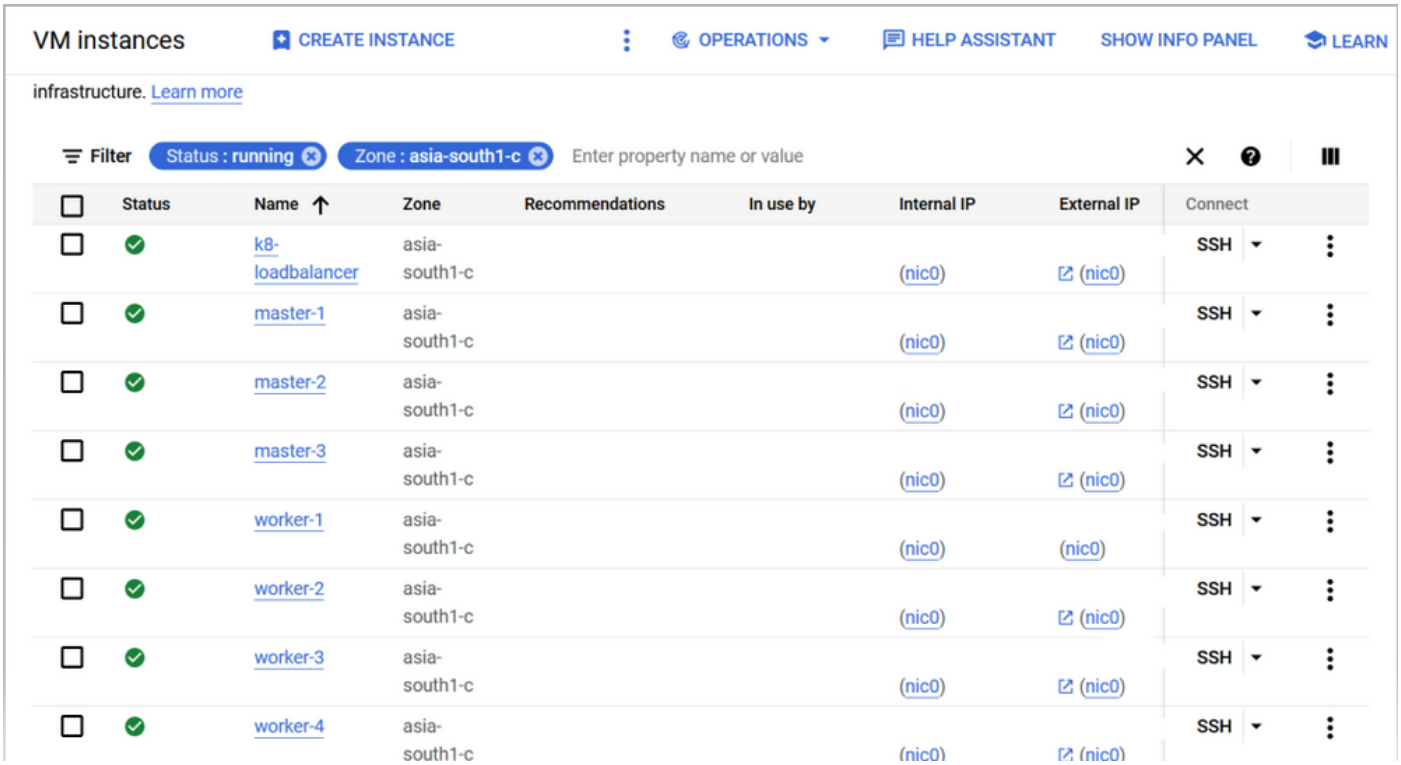
Select **Allow default access** and for firewall, **Allow HTTP traffic** and **Allow HTTPS traffic**.



Centos 7 network configuration

Click **Create**.

Similarly, create 8 nodes as shown here.



Multi-master deployment on google cloud platform

Private IPs:

On GCP the private and public IPs are automatically assigned.

```
master-1 = 10.160.x.9 master-2 = 10.160.x.10 master-3 = 10.160.x.11 worker-1 = 10.160.x.12
worker-2 = 10.160.x.13 worker-3 = 10.160.x.14 worker-4 = 10.160.x.16 bastion = 10.160.x.19
```

High-Level Overview

On all nodes (master, worker, bastion):

1. Open the required firewall ports on the Linux server and configure the security configurations.

On master nodes in multi-master deployments:

```
Kubernetes etcd server client API: 2379/tcp, 2380/tcp Kubernetes API server: 6443/tcp
```

Note: Also, ensure the ports on the GCP firewall are allowed.

2. Configure required network settings (local DNS, hostnames, NTP).

Bastion Server:
1. Set up a HA Proxy.
2. Add frontend and backend server configuration on **haproxy.conf** file.
3. Restart the **haproxy** service.

Common steps for master and worker nodes:
1. Install and configure the docker.
2. Install and configure Kubernetes.

Only on Master nodes:
1. Initialize the new Kubernetes cluster (**kubeadm init**).
2. Install **Calico** network plug-in (specifically used for core DNS service on the master nodes).
3. Use this command to join the master nodes with a master node.

```
kubeadm join <bastion-server-ip>:6443 --token <token-id> \ --discovery-token-ca-cert-hash <hash value> \ --control-plane --certificate-key <certificate-key-value>
```

4. Validate the cluster information with **kubectl get nodes**command.

Only on worker nodes:

1. Use this command to join the worker node to the master node.

```
kubeadm join <bastion-server>:6443 --token <token-id> \ --discovery-token-ca-cert-hash <hash-value>
```

2. Upon successful join, validate the cluster information on master nodes with this command **kubectl get nodes**.

Low-Level Configurations

Network Configuration

1. Change the root password if unknown with this command:

```
passwd
```

2. Change hostnames if necessary with this command.

```
hostnamectl set-hostname <hostname>
```

3. Configure Local DNS.

```
cat > /etc/hosts <<EOF 10.160.x.9 master-1 10.160.x.10 master-2 10.160.x.11 master-3 10.160.x.12 worker-1 10.160.x.13 worker-2 10.160.x.14 worker-3 10.160.x.16 worker-4 10.160.x.19 k8-loadbalancer 127.0.0.1 localhost EOF
```

4. Enable chrony for NTP services with this command.

```
systemctl enable --now chronyd
```

2. Verify the status with this command.

```
systemctl status chronyd
```

3. Check NTP sources with this command.

```
chronyc sources -v
```

Bastion Server

Step 1. Check updates.

```
sudo yum check-update
```

Step 2. Install updates if not up-to-date.

```
yum update -y
```

Step 3. Install yum utilities.

```
yum -y install yum-utils
```

Step 4. Install the haproxy package.

```
yum install haproxy -y
```

Step 5. Add this configuration under defaults in **/etc/haproxy/haproxy.cfg** :

```
frontend kubernetes bind 10.160.x.19:6443 option tcplog mode tcp default_backend kubernetes-
master-nodes frontend http_front mode http bind 10.160.x.19:80 default_backend http_back
frontend https_front mode http bind 10.160.x.19:443 default_backend https_back backend
kubernetes-master-nodes mode tcp balance roundrobin option tcp-check server master-1
10.160.x.9:6443 check fall 3 rise 2 server master-2 10.160.x.10:6443 check fall 3 rise 2 server
master-3 10.160.x.11:6443 check fall 3 rise 2 backend http_back mode http server master-1
10.160.x.9:6443 check fall 3 rise 2 server master-2 10.160.x.10:6443 check fall 3 rise 2 server
master-3 10.160.x.11:6443 check fall 3 rise 2 backend https_back mode http server master-1
10.160.x.9:6443 check fall 3 rise 2 server master-2 10.160.x.10:6443 check fall 3 rise 2 server
master-3 10.160.x.11:6443 check fall 3 rise 2 listen stats bind 10.160.x.19:8080 mode http stats
enable stats uri /
```

Step 6. Verify the configuration file so that it looks like this command **vi /etc/haproxy/haproxy.cfg**:

```
---------------------------------------------------------------------- # Example configuration
for a possible web application. See the # full configuration options online. # #
http://haproxy.1wt.eu/download/1.4/doc/configuration.txt # #---------------------------------
---------------------------------- #---------------------------------------------------------------------
-------- # Global settings #---------------------------------------------------------------------
- global # to have these messages end up in /var/log/haproxy.log you will # need to: # # 1)
configure syslog to accept network log events. This is done # by adding the '-r' option to the
SYSLOGD_OPTIONS in # /etc/sysconfig/syslog # # 2) configure local2 events to go to the
/var/log/haproxy.log # file. A line like the following can be added to # /etc/sysconfig/syslog #
# local2.* /var/log/haproxy.log # log 127.0.0.1 local2 chroot /var/lib/haproxy pidfile
/var/run/haproxy.pid maxconn 4000 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 http-server-close option
forwardfor except 127.0.0.0/8 option redispatch retries 3 timeout http-request 10s timeout queue
1m timeout connect 10s timeout client 1m timeout server 1m timeout http-keep-alive 10s timeout
check 10s maxconn 3000 frontend kubernetes bind 10.160.x.19:6443 option tcplog mode tcp
default_backend kubernetes-master-nodes frontend http_front mode http bind 10.160.x.19:80
default_backend http_back frontend https_front mode http bind 10.160.x.19:443 default_backend
https_back backend kubernetes-master-nodes mode tcp balance roundrobin option tcp-check server
master-1 10.160.x.9:6443 check fall 3 rise 2 server master-2 10.160.x.10:6443 check fall 3 rise
2 server master-3 10.160.x.11:6443 check fall 3 rise 2 backend http_back mode http server
master-1 10.160.x.9:6443 check fall 3 rise 2 server master-2 10.160.x.10:6443 check fall 3 rise
2 server master-3 10.160.x.11:6443 check fall 3 rise 2 backend https_back mode http server
master-1 10.160.x.9:6443 check fall 3 rise 2 server master-2 10.160.x.10:6443 check fall 3 rise
```

```
2 server master-3 10.160.x.11:6443 check fall 3 rise 2 listen stats bind 10.160.x.19:8080 mode http stats enable stats uri /
```

Step 7. Verify the status of haproxy:

```
[root@k8-loadbalancer vapadala]# systemctl status haproxy  haproxy.service - HAProxy Load Balancer Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; vendor preset: disabled) Active: active (running) since Wed 2022-10-26 08:33:17 UTC; 6s ago Main PID: 30985 (haproxy-systemd) CGroup: /system.slice/haproxy.service 30985 /usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid 30986 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds 30987 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds Oct 26 08:33:17 k8-loadbalancer systemd[1]: Started HAProxy Load Balancer. Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: haproxy-systemd-wrapper: executing /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /r...pid -Ds Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: [WARNING] 298/083317 (30986) : config : 'option forwardfor' ignored for frontend 'kube...P mode. Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: [WARNING] 298/083317 (30986) : config : 'option forwardfor' ignored for backend 'kuber...P mode. Hint: Some lines were ellipsized, use -l to show in full. [root@k8-loadbalancer vapadala]#
```

Possible Errors

1. The HAProxy service is in a failure state after you make configuration changes in **haproxy.cfg**. For example;

```
[root@k8-loadbalancer vapadala]# systemctl status haproxy  haproxy.service - HAProxy Load Balancer Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; vendor preset: disabled) Active: failed (Result: exit-code) since Wed 2022-10-26 08:29:23 UTC; 3min 44s ago Process: 30951 ExecStart=/usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid $OPTIONS (code=exited, status=1/FAILURE) Main PID: 30951 (code=exited, status=1/FAILURE) Oct 26 08:29:23 k8-loadbalancer systemd[1]: Started HAProxy Load Balancer. Oct 26 08:29:23 k8-loadbalancer haproxy-systemd-wrapper[30951]: haproxy-systemd-wrapper: executing /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /r...pid -Ds Oct 26 08:29:23 k8-loadbalancer haproxy-systemd-wrapper[30951]: [WARNING] 298/082923 (30952) : config : 'option forwardfor' ignored for frontend 'kube...P mode. Oct 26 08:29:23 k8-loadbalancer haproxy-systemd-wrapper[30951]: [WARNING] 298/082923 (30952) : config : 'option forwardfor' ignored for backend 'kuber...P mode. Oct 26 08:29:23 k8-loadbalancer haproxy-systemd-wrapper[30951]: [ALERT] 298/082923 (30952) : Starting frontend kubernetes: cannot bind socket [10.160.x.19:6443]. Oct 26 08:29:23 k8-loadbalancer haproxy-systemd-wrapper[30951]: haproxy-systemd-wrapper: exit, haproxy RC=1. Oct 26 08:29:23 k8-loadbalancer systemd[1]: haproxy.service: main process exited, code=exited, status=1/FAILURE. Oct 26 08:29:23 k8-loadbalancer systemd[1]: Unit haproxy.service entered failed state. Oct 26 08:29:23 k8-loadbalancer systemd[1]: haproxy.service failed. Hint: Some lines were ellipsized, use -l to show in full.
```

Resolution

```
1. Set the boolean value for haproxy_connect_any to true. 2. Restart the haproxy service. 3. Verify the status.
```

Execution:

```
[root@k8-loadbalancer vapadala]# setsebool -P haproxy_connect_any=1 [root@k8-loadbalancer vapadala]# systemctl restart haproxy [root@k8-loadbalancer vapadala]# systemctl status haproxy haproxy.service - HAProxy Load Balancer Loaded: loaded (/usr/lib/systemd/system/haproxy.service; enabled; vendor preset: disabled) Active: active (running) since Wed 2022-10-26 08:33:17 UTC; 6s ago Main PID: 30985 (haproxy-systemd) CGroup: /system.slice/haproxy.service 30985 /usr/sbin/haproxy-systemd-wrapper -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid 30986 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds 30987 /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /run/haproxy.pid -Ds Oct 26 08:33:17 k8-loadbalancer systemd[1]:
```

```
Started HAProxy Load Balancer. Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: haproxy-systemd-wrapper: executing /usr/sbin/haproxy -f /etc/haproxy/haproxy.cfg -p /r...pid -Ds. Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: [WARNING] 298/083317 (30986) : config : 'option forwardfor' ignored for frontend 'kube...P mode. Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: [WARNING] 298/083317 (30986) : config : 'option forwardfor' ignored for backend 'kuber...P mode. Hint: Some lines were ellipsized, use -l to show in full. [root@k8-loadbalancer vapadala]#
```

Install Docker on Master and Worker Nodes

Step 1. Check updates.

```
sudo yum check-update
```

Step 2. Install Updates if not up-to-date.

```
yum update -y
```

Step 3. Install yum utilities.

```
yum -y install yum-utils
```

Step 4. Install Docker.

```
curl -fsSL https://get.docker.com/ | sh
```

Step 5. Enable docker.

```
systemctl enable --now docker
```

Step 6. Start docker service.

```
sudo systemctl start docker
```

Step 7. Check the docker status.

```
sudo systemctl status docker
```

Output:

```
[root@kube-master1 vapadala]# sudo systemctl status docker  docker.service - Docker Application Container Engine Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled) Active: active (running) since Tue 2022-10-25 10:44:28 UTC; 40s ago Docs: https://docs.docker.com Main PID: 4275 (dockerd) Tasks: 21 Memory: 35.2M CGroup: /system.slice/docker.service 4275 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock Oct 25 10:44:26 kube-master1 dockerd[4275]: time="2022-10-25T10:44:26.128233809Z" level=info msg="scheme \"unix\" not registered, fallback to defaul...dule=grpc. Oct 25 10:44:26 kube-master1 dockerd[4275]: time="2022-10-25T10:44:26.128251910Z" level=info msg="ccResolverWrapper: sending update to cc: {[{unix:/...dule=grpc. Oct 25 10:44:26 kube-master1 dockerd[4275]: time="2022-10-25T10:44:26.128260953Z" level=info msg="ClientConn switching balancer to \"pick_first\"" module=grpc. Oct 25 10:44:27 kube-master1 dockerd[4275]: time="2022-10-25T10:44:27.920336293Z" level=info msg="Loading containers: start." Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.104357517Z" level=info msg="Default bridge (docker0) is assigned with an IP ad... address". Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.163830549Z" level=info msg="Loading containers: done." Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.182833703Z" level=info msg="Docker daemon"
```

```
commit=03df974 graphdriver(s)=overl...=20.10.20. Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.182939545Z" level=info msg="Daemon has completed initialization". Oct 25 10:44:28 kube-master1 systemd[1]: Started Docker Application Container Engine. Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.208492147Z" level=info msg="API listen on /var/run/docker.sock". Hint: Some lines were ellipsized, use -l to show in full. [root@kube-master1 vapadala]#
```

Install Kubernetes on Master and Worker Nodes

Step 1. Add Kubernetes repository.

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo [kubernetes] name=Kubernetes baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch enabled=1 gpgcheck=0 gpgkey=https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg exclude=kubelet kubeadm kubectl EOF

"gpgcheck = 0" will not verify the authenticity of the package if unsigned. Production environment it is recommended to set "gpgcheck = 1"
```

Step 2. Disable **SELinux**.

```
sudo setenforce 0 sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/selinux/config
```

Step 3. Install **Kubernetes**.

```
sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes
```

Step 4. Enable **Kubelet**.

```
sudo systemctl enable --now kubelet
```

Step 5. Configure **Pod Network**.

```
kubeadm init --control-plane-endpoint "10.160.x.19:6443" --upload-certs
```

Step 6. Token generation:

Output for the master (control plane) and for worker nodes.

Master Node

Token: Your Kubernetes control-plane has initialized successfully!

In order to use your cluster, run this as a regular user:

```
mkdir -p $HOME/.kube sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Alternatively, if you are the root user, you can run.

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

You can now deploy a pod network to the cluster.

```
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-administration/addons/
```

You can now join any number of the control-plane nodes or master nodes that run this command on each as root.

Refer: kubeadm join workflow

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \ --discovery-token-ca-cert-hash
sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61d9527d0 \ --control-plane --
certificate-key 66773b960199ef4530461ef4014e1432066902d4a3dee01669d8622579731701
```

Note: Please note that the certificate key gives access to cluster-sensitive data, keep it secret!

As a safeguard, uploaded certs are deleted in two hours; if necessary, you can use them.

```
"kubeadm init phase upload-certs --upload-certs" to reload certs afterward.
```

Then you can join any number of worker nodes and run this on each as root.

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \ --discovery-token-ca-cert-hash
sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61d9527d0
```

Step 7. Verify the Core DNS service.

```
[root@kube-master1 vapadala]# kubectl get pods --all-namespaces NAMESPACE NAME READY STATUS
RESTARTS AGE kube-system calico-kube-controllers-59697b644f-v2f22 1/1 Running 0 32m kube-system
calico-node-gdwr6 1/1 Running 0 5m54s kube-system calico-node-zszbc 1/1 Running 11 (5m22s ago)
32m kube-system calico-typha-6944f58589-q9jxf 1/1 Running 0 32m kube-system coredns-565d847f94-
cwgj8 1/1 Running 0 58m kube-system coredns-565d847f94-tttpq 1/1 Running 0 58m kube-system etcd-
kube-master1 1/1 Running 0 59m kube-system kube-apiserver-kube-master1 1/1 Running 0 59m kube-
system kube-controller-manager-kube-master1 1/1 Running 0 59m kube-system kube-proxy-flgvq 1/1
Running 0 5m54s kube-system kube-proxy-hf5qv 1/1 Running 0 58m kube-system kube-scheduler-kube-
master1 1/1 Running 0 59m [root@kube-master1 vapadala]#
```

Step 8. Verify the status of the master node.

```
[root@kube-master1 vapadala]# kubectl get nodes NAME STATUS ROLES AGE VERSION kube-master1 Ready
control-plane 11m v1.25.3
```

To join multiple master nodes, this join command is executed on master nodes.

```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \ --discovery-token-ca-cert-hash
sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61d9527d0 \ --control-plane --
certificate-key 66773b960199ef4530461ef4014e1432066902d4a3dee01669d8622579731701
```

Possible Errors Encountered at the Time of Token Generation

Error CRI

```
[root@kube-master1 vapadala]# kubeadm init --control-plane-endpoint "10.160.x.19:6443" --upload-
certs [init] Using Kubernetes version: v1.25.3 [preflight] Running pre-flight checks [WARNING
Firewalld]: firewalld is active, please ensure ports [6443 10250] are open or your cluster may
not function correctly error execution phase preflight: [preflight] Some fatal errors occurred:
[ERROR CRI]: container runtime is not running: output: time="2022-10-25T08:56:42Z" level=fatal
```

```
msg="unable to determine runtime API version: rpc error: code = Unavailable desc = connection error: desc = \"transport: Error while dialing dial unix /var/run/containerd/containerd.sock: connect: no such file or directory\"", error: exit status 1. [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1 [preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-errors=...` To see the stack trace of this error execute with --v=5 or higher
```

Error CRI Resolution

Step 1. Remove the **config.toml** file, and restart **containerd**.

```
rm -rf /etc/containerd/config.toml systemctl restart containerd kubeadm init
```

Step 2. Install containerd:
Install the containerd.io package from the official Docker repositories with these commands.

```
yum install -y yum-utils yum-config-manager --add-repo
https://download.docker.com/linux/centos/docker-ce.repo. yum install -y containerd.io
```

Step 3. Configure containerd:

```
sudo mkdir -p /etc/containerd containerd config default | sudo tee /etc/containerd/config.toml
```

Step 4. Restart containerd:

```
systemctl restart containerd
```

Error FileContent--proc-sys-net-ipv4-ip_forward

```
[ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
```

Error FileContent--proc-sys-net-ipv4-ip_forward Resolution

Set the ip_forward value to 1.

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Core DNS Service Does Not Run

```
[root@kube-master1 vapadala]# kubectl get nodes NAME STATUS ROLES AGE VERSION kube-master1 NotReady control-plane 11m v1.25.3 [root@kube-master1 vapadala]# kubectl get pods --all-namespaces NAMESPACE NAME READY STATUS RESTARTS AGE kube-system coredns-565d847f94-cwgj8 0/1 Pending 0 12m kube-system coredns-565d847f94-tttpq 0/1 Pending 0 12m kube-system etcd-kube-master1 1/1 Running 0 12m kube-system kube-apiserver-kube-master1 1/1 Running 0 12m kube-system kube-controller-manager-kube-master1 1/1 Running 0 12m kube-system kube-proxy-hf5qv 1/1 Running 0 12m kube-system kube-scheduler-kube-master1 1/1 Running 0 12m [root@kube-master1 vapadala]#
```

Resolution

The Core DNS is pending which indicates an issue with networking. Hence, Calico needs to be installed.

Reference: Calico

Execute these two commands:

```
curl https://raw.githubusercontent.com/projectcalico/calico/v3.24.3/manifests/calico-typha.yaml -o calico.yaml kubectl apply -f calico.yaml
```

Worker Node

On Worker Node when token is obtained from master:

Step 1. Enable kubelet service.

```
sudo systemctl daemon-reload sudo systemctl restart docker sudo systemctl restart kubelet systemctl enable kubelet.service
```

Step 2. Join all the worker nodes with master node with this join command.

```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \ --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61d9527d0
```

Step 3. If errors related to files or ports are encountered, reset the kubeadm with this command.

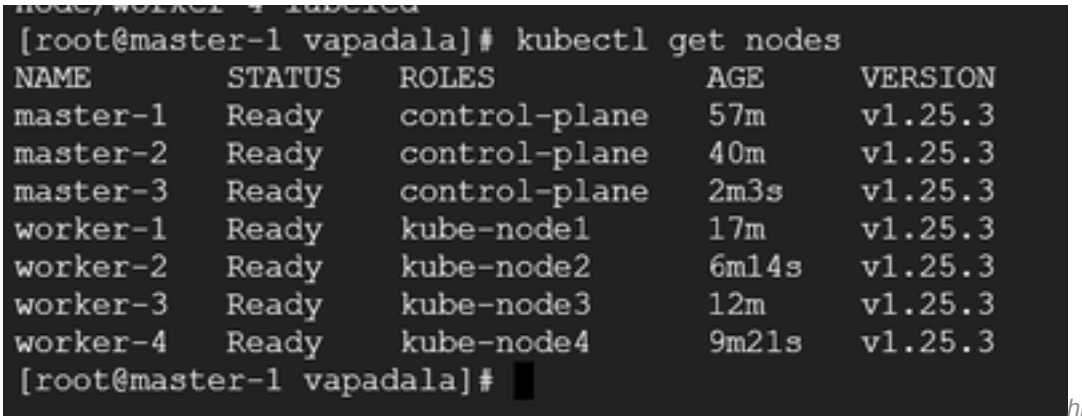
```
kubeadm reset
```

After you reset, enter the token from the master node.

```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \ --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61d9527d0
```

Final Output

The cluster is now formed, verify it with the command **kubectl get nodes**.



high availability kubernetes cluster output

Note: At the time of cluster formation, only control from master nodes is configured. The bastion host is not configured as a centralized server to monitor all the Kube's in the cluster.