Cree un clúster de Kubernetes multi-master con Centos 7 en la plataforma de nube de Google

Contenido

Introducción

Este documento describe la implementación del clúster de Kubernetes con 3 nodos maestros y 4 nodos de trabajo con un host bastión que actúa como equilibrador de carga en la Plataforma de nube de Google (GCP).

Prerequisites

Requirements

Cisco recomienda que tenga conocimiento sobre estos temas:

- Linux
- Dockers y Kubernetes
- Plataforma de nube de Google

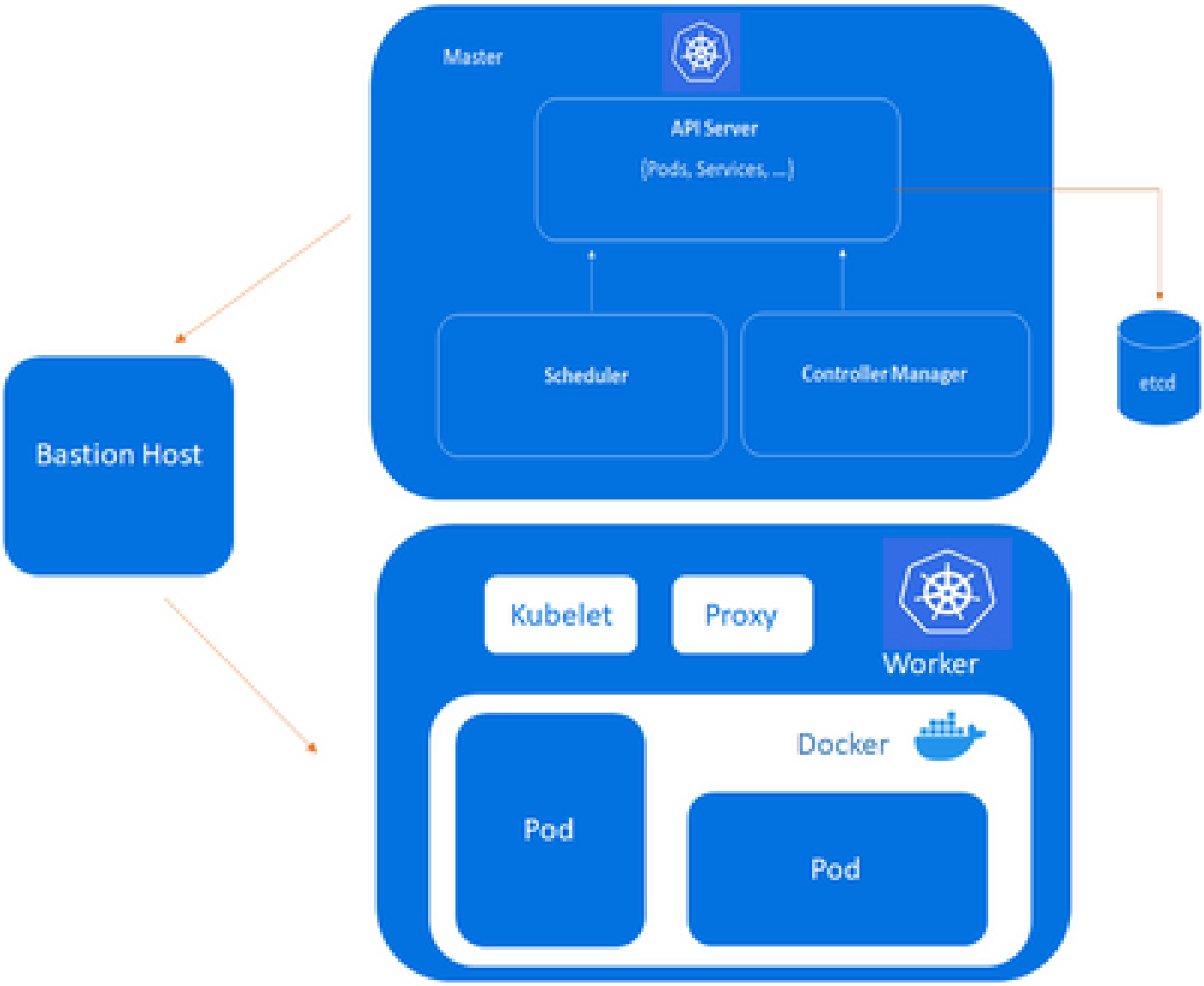
Componentes Utilizados

La información de este documento se basa en:

- Sistema operativo: máquina virtual Centos 7
- Familia de máquinas (e2-standard-16):
  - vCPU: 16 vCPU
  - RAM: 64 GB

La información que contiene este documento se creó a partir de los dispositivos en un ambiente de laboratorio específico. Todos los dispositivos que se utilizan en este documento se pusieron en funcionamiento con una configuración verificada (predeterminada). Si tiene una red en vivo, asegúrese de entender el posible impacto de cualquier comando.

Diagrama de la red



Host bastión, Kube-Master, Kube-node

Componentes del nodo maestro de Kubernetes

Servidor Kube:
i. Proporciona una API que sirve como el extremo frontal de un plano de control de Kubernetes.
ii. Administra las solicitudes externas e internas que determinan si una solicitud es válida y, a continuación, la procesa.
iii. Se puede acceder a la API a través de la interfaz de línea de **kubectl** comandos u otras herramientas como **kubeadm**, y a través de llamadas REST.

Programador Kube:
i. Este componente programa grupos de dispositivos en nodos específicos según flujos de trabajo automatizados y condiciones definidas por el usuario.

Kube-controller-manager:
i. El administrador de controladores de Kubernetes es un loop de control que monitorea y regula el estado de un clúster de Kubernetes.
ii. Recibe información sobre el estado actual del clúster y los objetos que contiene y envía

instrucciones para mover el clúster hacia el estado deseado por el operador del clúster.

etcd:
i. Una base de datos de valor de clave que contiene datos sobre el estado y la configuración del clúster.
ii. Etcd es tolerante a fallas y está distribuido.

Componentes del nodo de Kubernetes Worker

Kubelet:
i. Cada nodo contiene un **kubelet**, que es una pequeña aplicación que puede comunicarse con el plano de control de Kubernetes.
ii. El **kubelet** garantiza que los contenedores especificados en la configuración de grupo de dispositivos se ejecuten en un nodo específico y administren su ciclo de vida.
iii. Ejecuta las acciones comandadas por el plano de control.

Proxy de Kube:
i. Todos los nodos informáticos contienen **kube-proxy**, un proxy de red que facilita los servicios de red de Kubernetes.
ii. Gestiona todas las comunicaciones de red fuera y dentro del clúster, y reenvía el tráfico o las respuestas en la capa de filtrado de paquetes del sistema operativo.

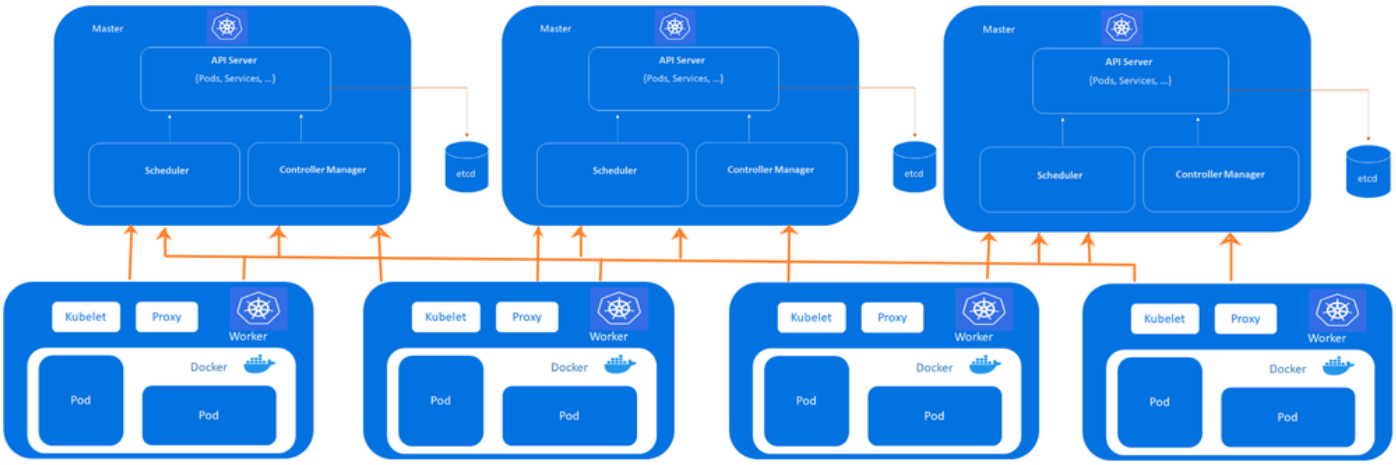
Grupos:
i. Una vaina sirve como una instancia de aplicación única y se considera la unidad más pequeña en el modelo de objetos de Kubernetes.

Host bastión:
i. El equipo suele alojar una única aplicación o proceso, por ejemplo, un servidor proxy o un equilibrador de carga, y todos los demás servicios se eliminan o limitan para reducir la amenaza al equipo.

Arquitectura Multi-Master de Kubernetes

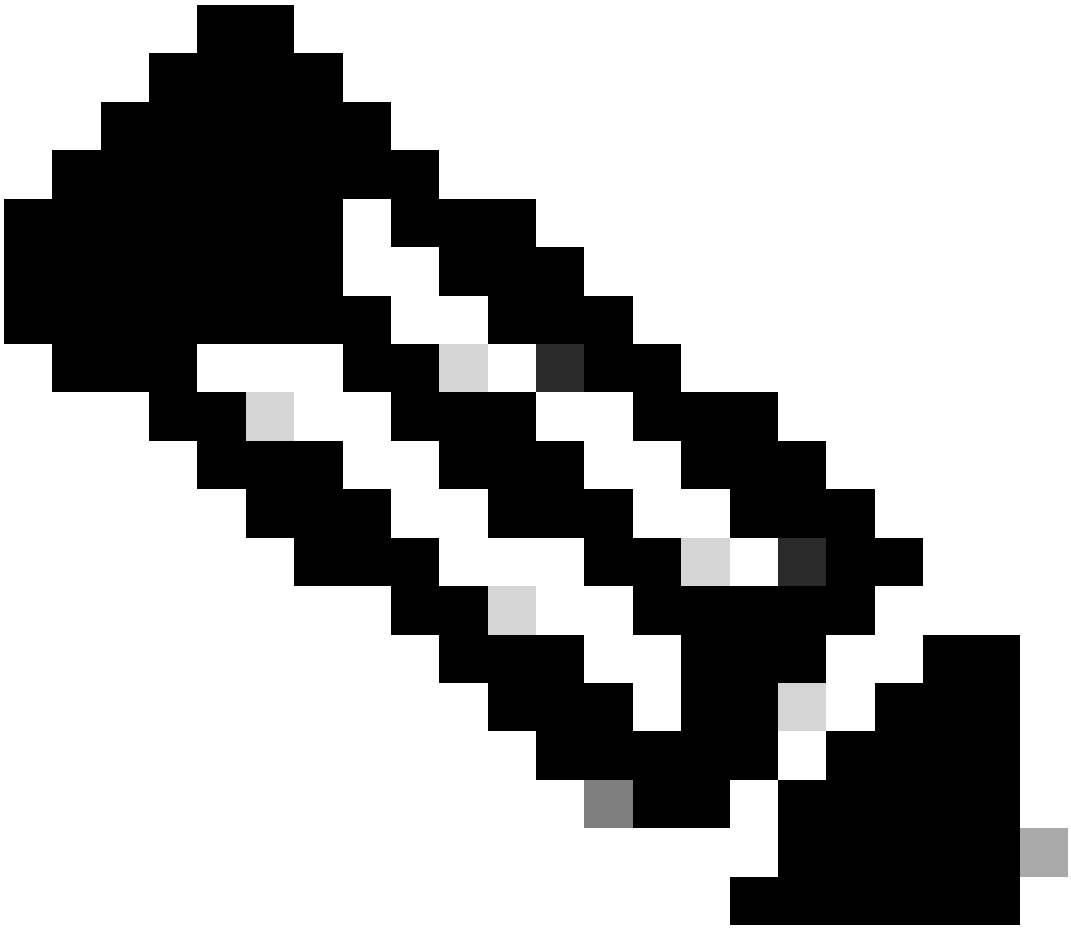
Clúster:

- 8 - Nodos totales
- 3 - Nodos maestros
- 4 - Nodos de trabajador
- 1 - Servidor Bastion (equilibrador de carga)



Clúster de Kubernetes de alta disponibilidad con tres planos de control

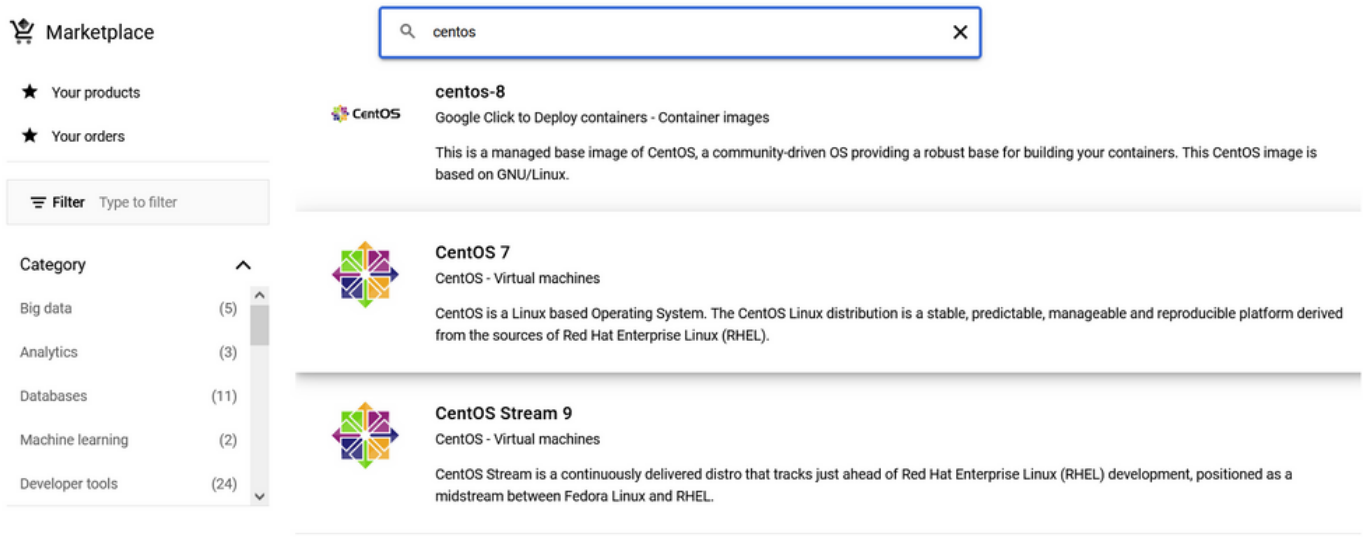
---



Nota: Configure VPC en GCP antes de aprovisionar las VM. Haga referencia a VPC en GCP.

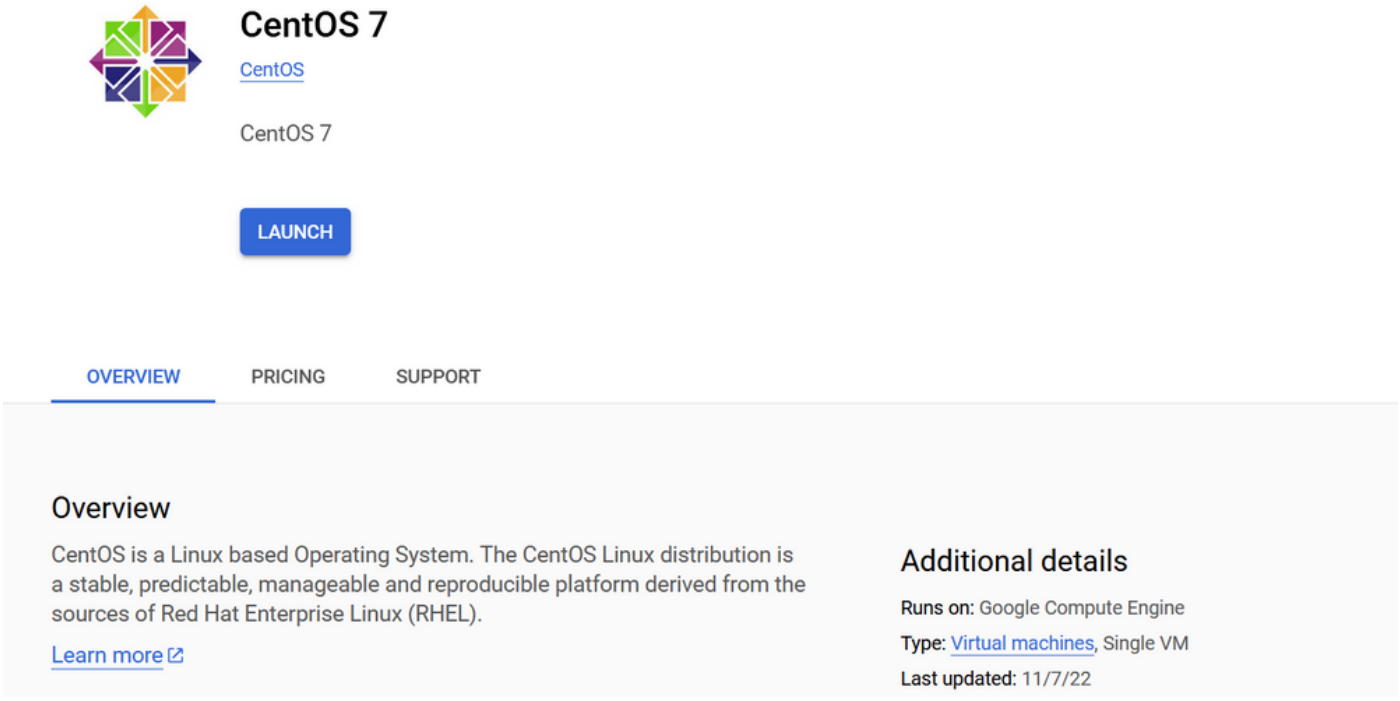
Suministro de máquinas virtuales en GCP

Sobre la provisión de GCP a Centos7 del mercado de GCP.



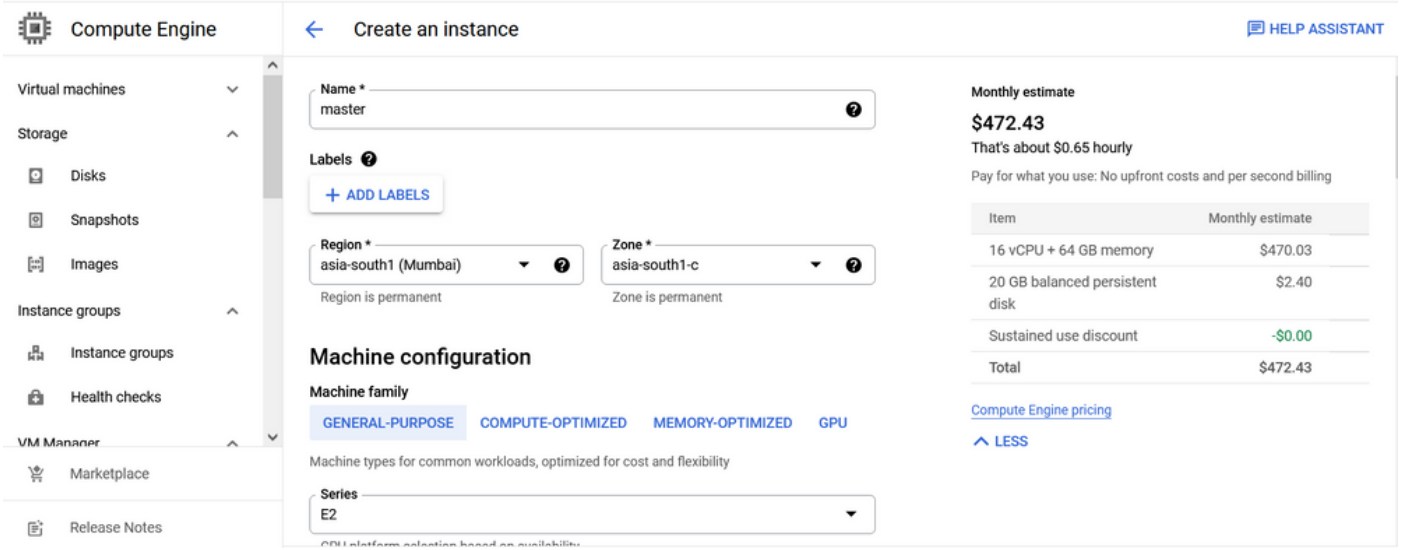
Centos Marketplace en GCP

Haga clic en **Launch**.



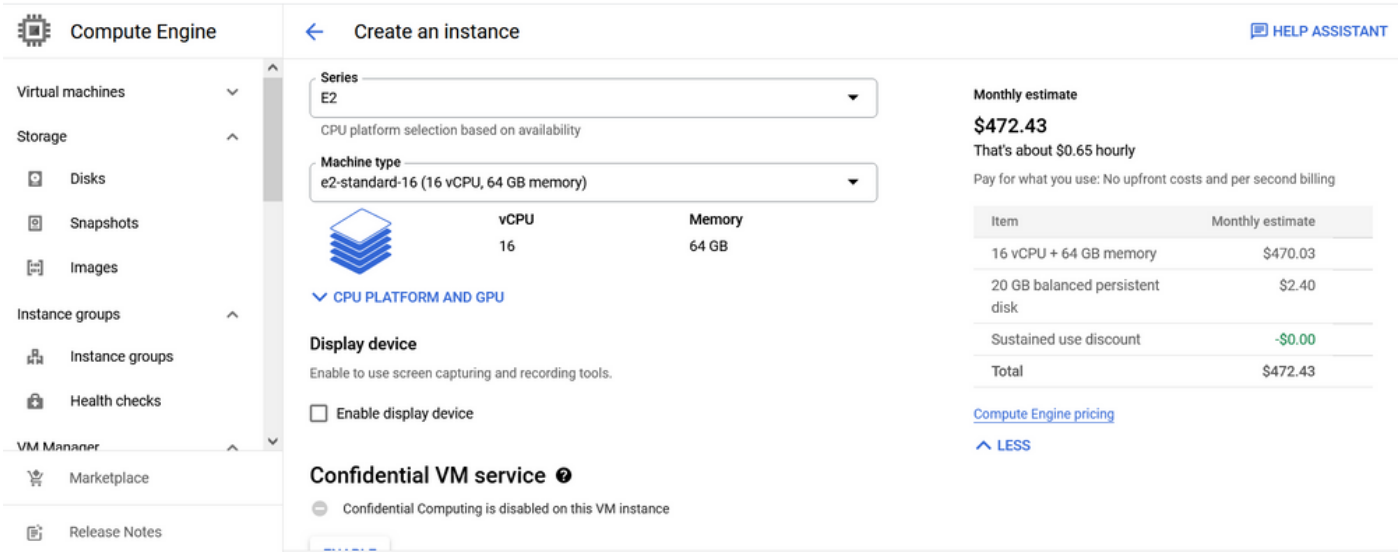
Máquina virtual Centos 7

Elija la región según el alcance más cercano. En este laboratorio, la región se configura como Mumbai.



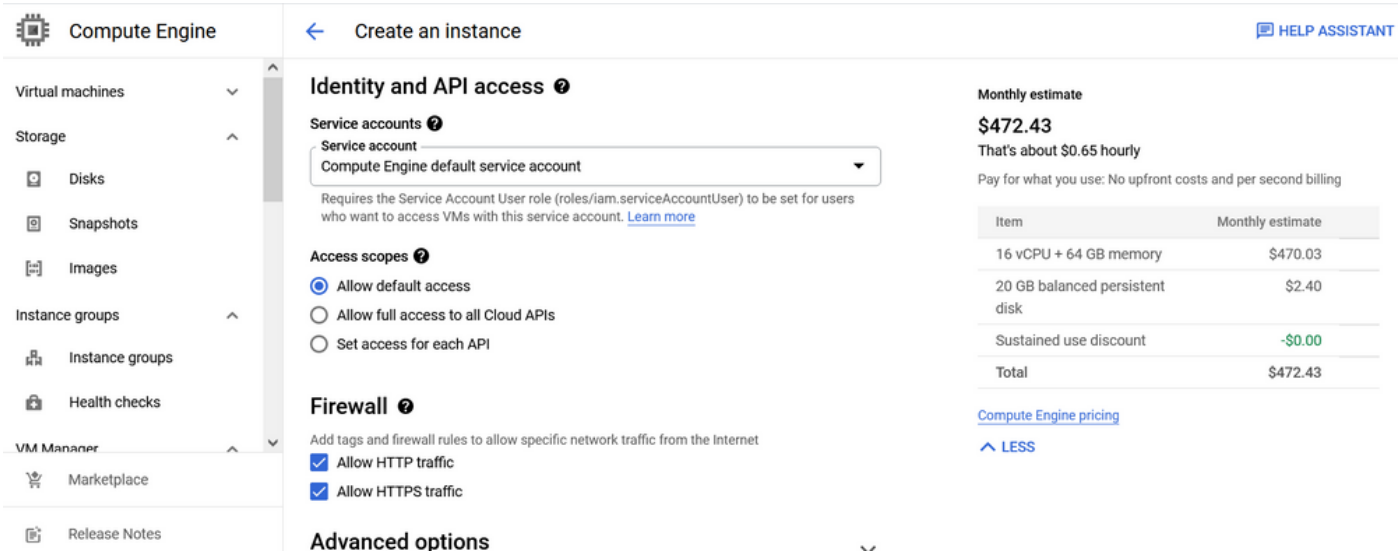
Configuración del motor informático Centos7

La configuración de la máquina es de uso general serie E2 y tipo de máquina **e2-standard-16 (16 vCPU, 64 GB memory)**.



Configuración de recursos de Centos 7

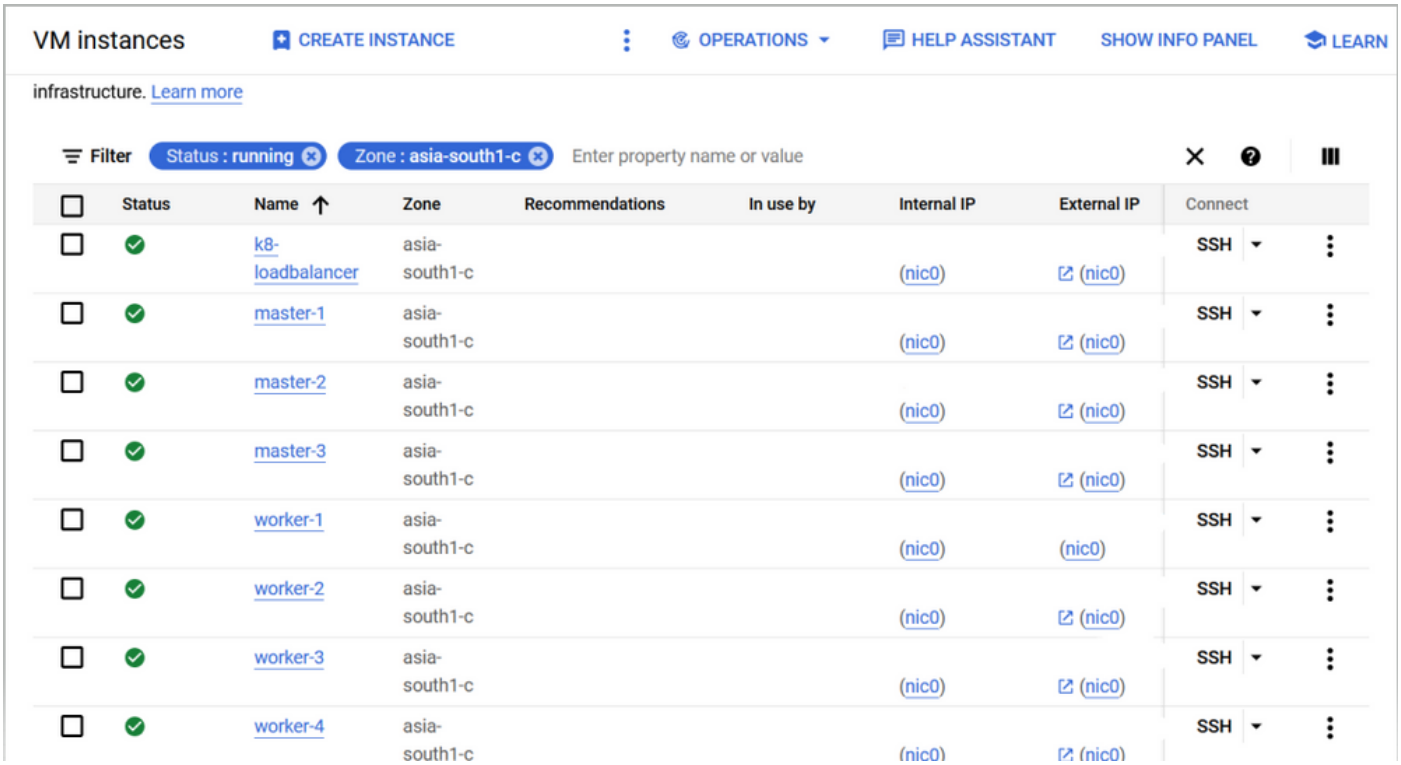
Seleccione **Allow default access** y para firewall, **Allow HTTP traffic** y **Allow HTTPS traffic**.



Configuración de red de Centos 7

Haga clic en **Create**.

De forma similar, cree 8 nodos como se muestra aquí.



Implementación multi-master en la plataforma de nube de Google

IP privadas:

En GCP, las IP privadas y públicas se asignan automáticamente.

```
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
```

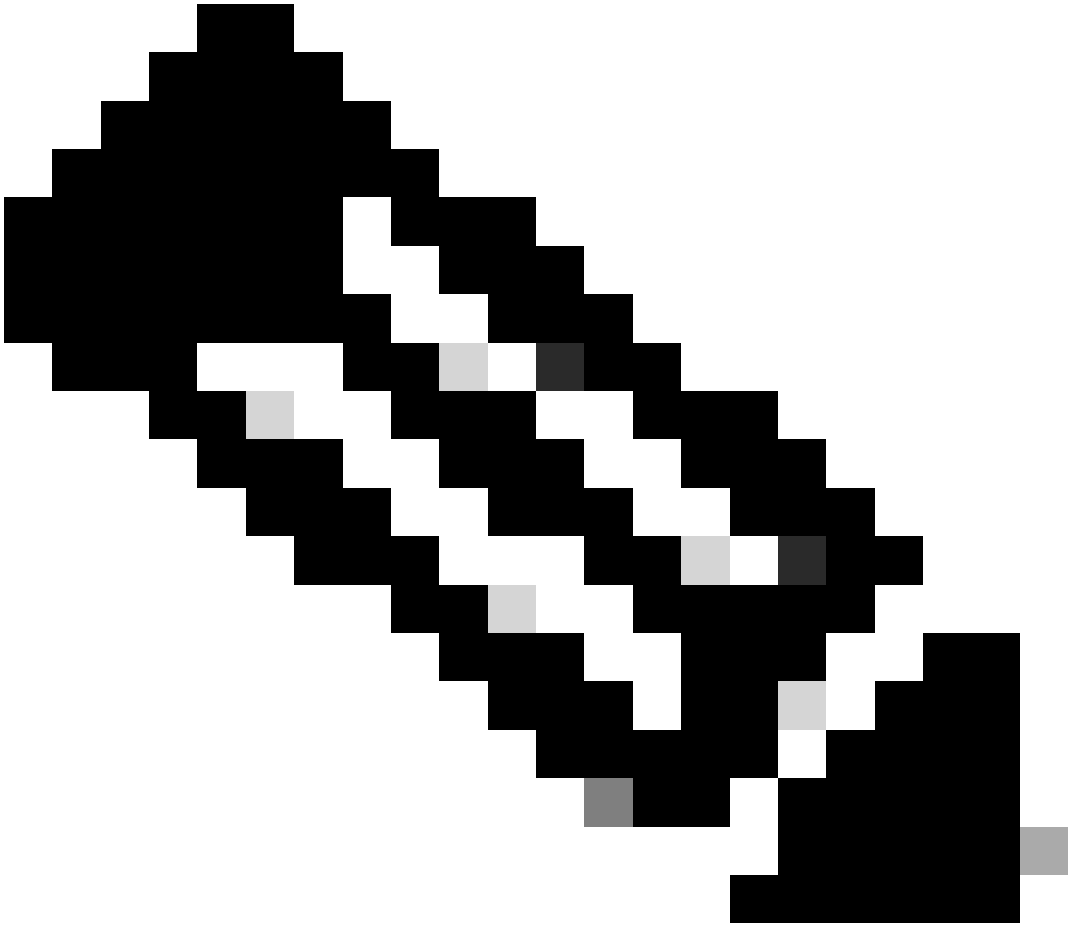
Descripción general de alto nivel

En todos los nodos (maestro, trabajador, bastión):

1. Abra los puertos de firewall requeridos en el servidor Linux y configure las configuraciones de seguridad.

En los nodos maestros de las implementaciones de varios maestros:

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



Nota: Además, asegúrese de que los puertos del firewall GCP estén permitidos.

2. Configure los parámetros de red necesarios (DNS local, nombres de host, NTP).

Servidor Bastion:
1. Configure un proxy HA.
2. Agregue la configuración del servidor front-end y backend en el **haproxy.conf** archivo.
3. Reinicie el **haproxy** servicio.

Pasos comunes para los nodos maestros y de trabajo:
1. Instale y configure el acoplador.
2. Instalar y configurar Kubernetes.

Sólo en nodos maestros:
1. Inicialice el nuevo clúster de Kubernetes (**kubeadm init**).
2. Instale el **Calico** complemento de red (utilizado específicamente para el servicio DNS principal en los nodos maestros).

3. Utilice este comando para unir los nodos maestros con un nodo maestro.

```
kubeadm join

        :6443 --token

          \
        --discovery-token-ca-cert-hash

           \
        --control-plane --certificate-key
```

4. Valide la información del clúster con el **kubectl get nodes**comando.

Sólo en nodos de trabajo:

1. Utilice este comando para unir el nodo de trabajo al nodo maestro.

```
kubeadm join

        :6443 --token

          \
        --discovery-token-ca-cert-hash
```

2. Una vez que se haya unido correctamente, valide la información de agrupamiento en los nodos

maestros con este comando **kubectl get nodes**.

Configuraciones de bajo nivel

Configuración de red

1. Cambie la contraseña raíz si es desconocida con este comando:

```
passwd
```

2. Cambie los nombres de host si es necesario con este comando.

```
hostnamectl set-hostname
```

3. Configure el DNS local.

```
cat > /etc/hosts <
```

```
4. Habilite chrony para los servicios NTP con este comando.
```

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

```
2. Verifique el estado con este comando.
```

```
systemctl status chronyd
```

```
3. Verifique los orígenes NTP con este comando.
```

```
chronyc sources -v
```

Servidor Bastion

Paso 1. Compruebe las actualizaciones.

```
sudo yum check-update
```

Paso 2. Instalar actualizaciones si no están actualizadas.

```
yum update -y
```

Paso 3. Instale yum utilities.

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

Paso 4. Instale el paquete haproxy.

```
yum install haproxy -y
```

Paso 5. Agregue esta configuración en valores predeterminados en **/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 /
```

Paso 6. Verifique el archivo de configuración para que se vea como este comando **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 /
```

Paso 7. Verifique el estado de 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/hapr
Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: [WARNING] 298/083317 (30986) : config : 'option f
Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: [WARNING] 298/083317 (30986) : config : 'option f
Hint: Some lines were ellipsized, use -l to show in full.
[root@k8-loadbalancer vapadala]#
```

Posibles errores

1. El servicio HAProxy se encuentra en estado de error después de realizar cambios de configuración en **haproxy.cfg**. Por ejemplo;

```
[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 $OPT
 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/hapr
Oct 26 08:29:23 k8-loadbalancer haproxy-systemd-wrapper[30951]: [WARNING] 298/082923 (30952) : config : 'option f
Oct 26 08:29:23 k8-loadbalancer haproxy-systemd-wrapper[30951]: [WARNING] 298/082923 (30952) : config : 'option f
Oct 26 08:29:23 k8-loadbalancer haproxy-systemd-wrapper[30951]: [ALERT] 298/082923 (30952) : Starting frontend ku
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.
```

Resolución

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

Ejecución:

```
[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/hapr
Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: [WARNING] 298/083317 (30986) : config : 'option f
Oct 26 08:33:17 k8-loadbalancer haproxy-systemd-wrapper[30985]: [WARNING] 298/083317 (30986) : config : 'option f
Hint: Some lines were ellipsized, use -l to show in full.
[root@k8-loadbalancer vapadala]#
```

Instalar Docker en nodos maestros y de trabajo

Paso 1. Compruebe las actualizaciones.

```
sudo yum check-update
```

Paso 2. Instalar actualizaciones si no están actualizadas.

```
yum update -y
```

Paso 3. Instale yum utilities.

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

Paso 4. Instale Docker.

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

Paso 5. Habilite docker.

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

Paso 6. Inicie el servicio de acoplamiento.

```
sudo systemctl start docker
```

Paso 7. Compruebe el estado del acoplador.

```
sudo systemctl status docker
```

Salida:

```
[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\"
Oct 25 10:44:26 kube-master1 dockerd[4275]: time="2022-10-25T10:44:26.128251910Z" level=info msg="ccResolverWrapp
Oct 25 10:44:26 kube-master1 dockerd[4275]: time="2022-10-25T10:44:26.128260953Z" level=info msg="ClientConn swit
Oct 25 10:44:27 kube-master1 dockerd[4275]: time="2022-10-25T10:44:27.920336293Z" level=info msg="Loading contain
Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.104357517Z" level=info msg="Default bridge
Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.163830549Z" level=info msg="Loading contain
Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.182833703Z" level=info msg="Docker daemon"
Oct 25 10:44:28 kube-master1 dockerd[4275]: time="2022-10-25T10:44:28.182939545Z" level=info msg="Daemon has comp
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 /
Hint: Some lines were ellipsized, use -l to show in full.
[root@kube-master1 vapadala]#
```

Instalar Kubernetes en nodos maestros y de trabajo

Paso 1. Agregar repositorio de Kubernetes.

```
cat <
```

```
"gpgcheck = 0" will not verify the authenticity of the package if unsigned. Production environment it i
```

Paso 2. Desactivar **SELinux**.

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

Paso 3. Instalación **Kubernetes**.

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

Paso 4. Activar **Kubelet**.

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

Paso 5. Configuración **Pod Network**.

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

Paso 6. Generación de token:

Salida para el maestro (plano de control) y para los nodos de trabajo.

Nodo maestro

Token: El plano de control de Kubernetes se ha inicializado correctamente.

Para utilizar su cluster, ejecute esto como un usuario regular:

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

Alternativamente, si usted es el usuario root, puede ejecutar.

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

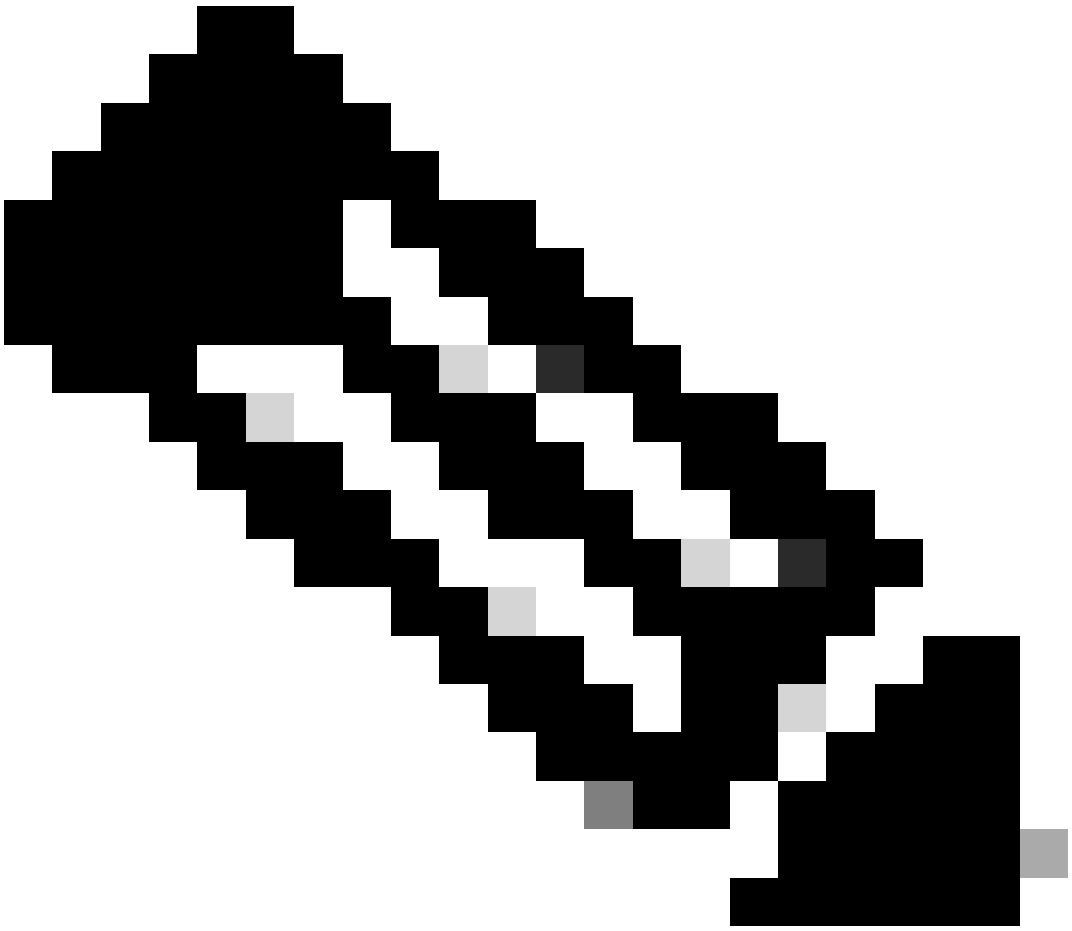
Ahora puede implementar una red de grupo de dispositivos en el clúster.

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

Ahora puede unir cualquier número de nodos del plano de control o nodos maestros que ejecuten este comando en cada uno como raíz.

Referencia: [kubeadm join workflow]

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.hO7kyelronuyOmw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
        --control-plane --certificate-key 66773b960199ef4530461ef4014e1432066902d4a3dee01669d8622579731
```



Nota: Tenga en cuenta que la clave del certificado da acceso a los datos confidenciales del clúster, ¡manténgala en secreto!

Como medida de seguridad, los certificados cargados se eliminan en dos horas; si es necesario, puede usarlos.

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

A continuación, puede unirse a cualquier número de nodos de trabajo y ejecutarlo en cada uno como raíz.

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

Paso 7. Verifique el servicio DNS principal.

```
[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]#
```

Paso 8. Compruebe el estado del nodo maestro.

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

Para unir varios nodos maestros, este comando join se ejecuta en los nodos maestros.

```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
        --control-plane --certificate-key 66773b960199ef4530461ef4014e1432066902d4a3dee01669d8622579731
```

Posibles errores encontrados en el momento de la generación del token

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 clu
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
        [ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are n
[preflight] If you know what you are doing, you can make a check non-fatal with `--ignore-preflight-err
To see the stack trace of this error execute with --v=5 or higher
```

Resolución CRI de error

Paso 1. Quite el archivo **config.toml** y reinicie el **contenedor.**

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

Paso 2. Instalar contenido:
Instale el paquete containerd.io desde los repositorios Docker oficiales con estos comandos.

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

Paso 3. Configuración de contenido:

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

Paso 4. Reiniciar contenido:

```
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 t
```

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

Establezca el valor ip_forward en 1.

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

El servicio DNS principal no se ejecuta

```
[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   A
kube-system   coredns-565d847f94-cwgj8                      0/1         Pending   0                12m
kube-system   coredns-565d847f94-tttpq                       0/1         Pending   0                12
```

```
kube-system   etcd-kube-master1                                1/1         Running   0
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
kube-system   kube-scheduler-kube-master1                  1/1         Running   0               12m
[root@kube-master1 vapadala]#
```

Resolución

El DNS principal está pendiente, lo que indica un problema con la red. Por lo tanto, Calico necesita ser instalado.

Referencia: Calico

Ejecute estos dos comandos:

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

Nodo de trabajo

En el nodo de trabajo cuando el token se obtiene del maestro:

Paso 1. Habilite el servicio de kubelet.

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

Paso 2. Unir todos los nodos de trabajo con el nodo maestro con este comando de unión.

```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

Paso 3. Si se encuentran errores relacionados con archivos o puertos, restablezca el kubeadm con este comando.

```
kubeadm reset
```

Después de restablecer, introduzca el token del nodo maestro.

```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \
```

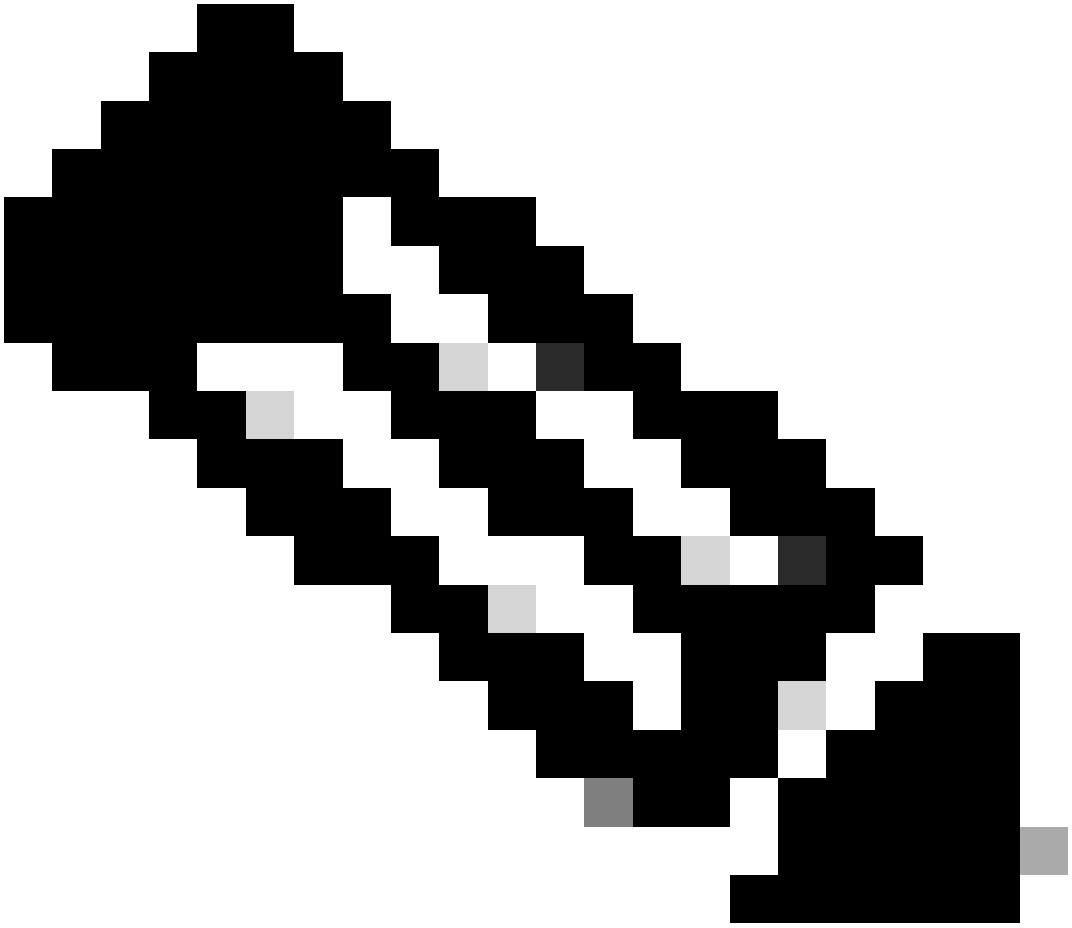
```
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

Resultado final

El clúster está ahora formado, verifíquelo con el comando **kubectl get nodes**.

```
[root@master-1 vapadala]# kubectl get nodes
NAME       STATUS   ROLES           AGE     VERSION
master-1   Ready    control-plane   57m     v1.25.3
master-2   Ready    control-plane   40m     v1.25.3
master-3   Ready    control-plane   2m3s    v1.25.3
worker-1   Ready    kube-node1      17m     v1.25.3
worker-2   Ready    kube-node2      6m14s   v1.25.3
worker-3   Ready    kube-node3      12m     v1.25.3
worker-4   Ready    kube-node4      9m21s   v1.25.3
[root@master-1 vapadala]#
```

resultado de clúster de kubernetes de alta disponibilidad



Nota: En el momento de la formación del clúster, sólo se configura el control desde los nodos maestros. El host bastión no está configurado como servidor centralizado para monitorear todos los Kube del clúster.

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).