Crie um cluster Kubernetes multi-master com Centos 7 na plataforma de nuvem do Google

Contents

- Introdução
- Pré-requisitos
  - Requisitos
  - Componentes Utilizados
- Diagrama de Rede
- Componentes do nó mestre do Kubernetes
- Componentes do nó de trabalho do Kubernetes
- Arquitetura multi-master do Kubernetes
- Fornecimento de máquinas virtuais em GCP
- Visão geral de alto nível
- Configurações de baixo nível
  - Configuração de rede
  - Servidor Bastion
    - Possíveis erros
    - Resolução
  - Instalar Docker nos Nós Mestre e de Trabalho
  - Instalar o Kubernetes nos nós mestre e de trabalho
  - Nó mestre
  - Possíveis erros encontrados no momento da geração do token
    - Erro de CRI
    - Erro de resolução de CRI
    - Erro FileContent—proc-sys-net-ipv4-ip_forward
    - Erro na resolução FileContent—proc-sys-net-ipv4-ip_forward
    - O serviço DNS principal não é executado
    - Resolução
  - Nó do trabalhador
- Saída final

Introdução

Este documento descreve a implementação do cluster Kubernetes com 3 nós mestre e 4 nós de trabalho com um bastion host que atua como um balanceador de carga no Google Cloud Platform (GCP).

Pré-requisitos

Requisitos

A Cisco recomenda que você tenha conhecimento destes tópicos:

- Linux
- Dockers e Kubernetes
- Plataforma de nuvem do Google

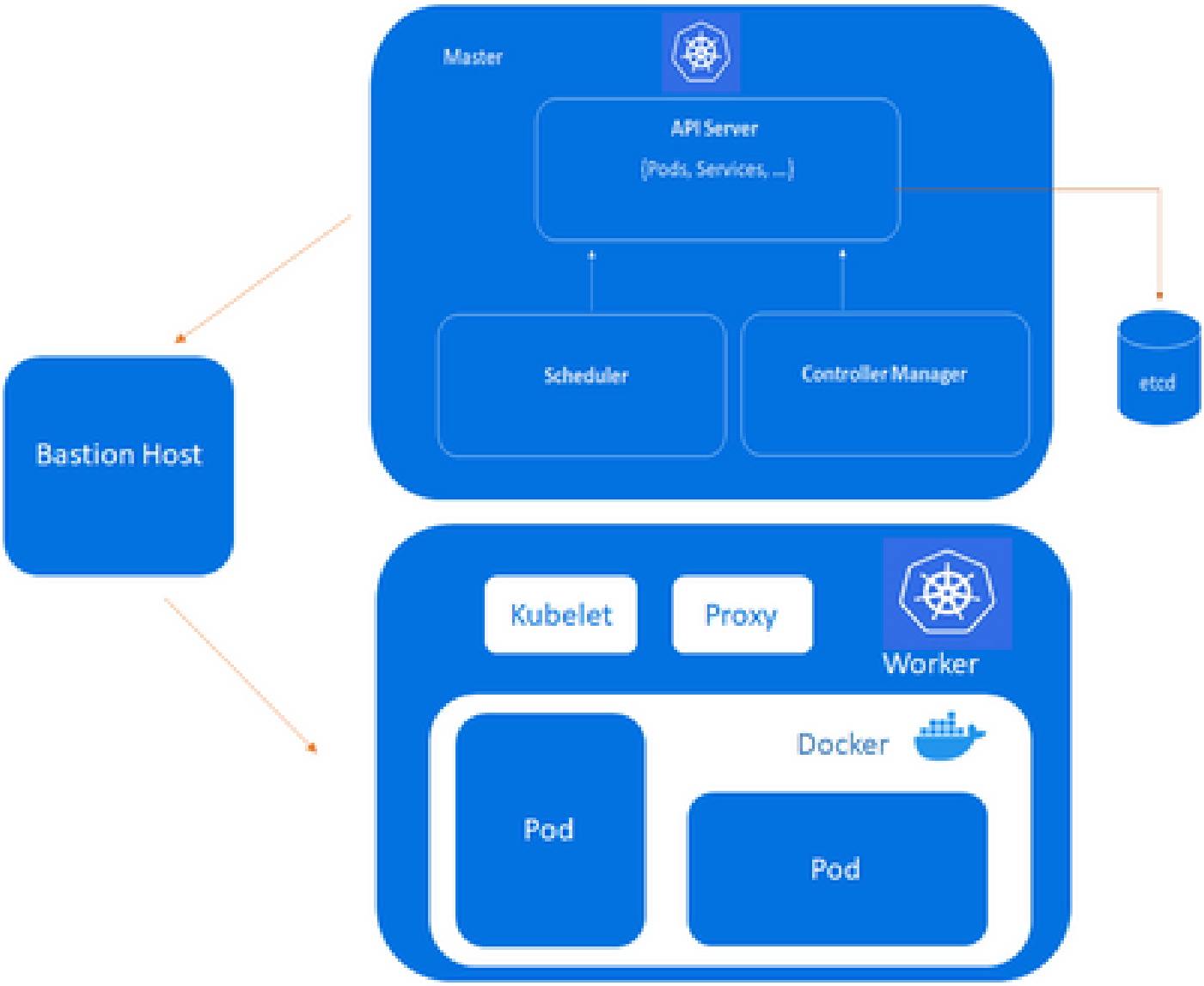
Componentes Utilizados

As informações neste documento são baseadas em:

- SO - Máquina virtual Centos 7
- Família de máquinas (e2-standard-16):
  - vCPUs - 16 vCPUs
  - RAM - 64 GB

As informações neste documento foram criadas a partir de dispositivos em um ambiente de laboratório específico. Todos os dispositivos utilizados neste documento foram iniciados com uma configuração (padrão) inicial. Se a rede estiver ativa, certifique-se de que você entenda o impacto potencial de qualquer comando.

Diagrama de Rede



Hospedeiro Bastion, Kube-Master, Kube-node

Componentes do nó mestre do Kubernetes

Kube-apiserver:
i. Fornece uma API que serve como front-end de um plano de controle Kubernetes.
ii) Ele processa solicitações externas e internas que determinam se uma solicitação é válida e, em seguida, a processa.
iii) **kubectl** A API pode ser acessada por meio da interface de linha de comando ou outras ferramentas como o **kubeadm**e por meio de chamadas REST.

Agendador do Kube:
i. Este componente agenda pods em nós específicos de acordo com fluxos de trabalho automatizados e condições definidas pelo usuário.

Controlador-gerenciador-Kube:
i. O gerenciador do controlador Kubernetes é um loop de controle que monitora e regula o estado de um cluster Kubernetes.
ii) Ele recebe informações sobre o estado atual do cluster e objetos dentro dele e envia instruções

para mover o cluster para o estado desejado pelo operador do cluster.

etcd:
i. Um banco de dados de valor-chave que contém dados sobre o estado e a configuração do cluster.
ii) O Etcd é tolerante a falhas e distribuído.

Componentes do nó de trabalho do Kubernetes

Kubelet
i. Cada nó contém um **kubelet**, que é uma pequena aplicação que pode se comunicar com o plano de controle do Kubernetes.
ii) O **kubelet** garante que os contêineres especificados na configuração do pod sejam executados em um nó específico e gerenciem seu ciclo de vida.
iii) Ele executa as ações comandadas pelo seu plano de controle.

Proxy do Kube:
i. Todos os nós de computação contêm **kube-proxy**, um proxy de rede que facilita os serviços de rede Kubernetes.
ii) Ele lida com todas as comunicações de rede fora e dentro do cluster e encaminha o tráfego ou as respostas na camada de filtragem de pacotes do sistema operacional.

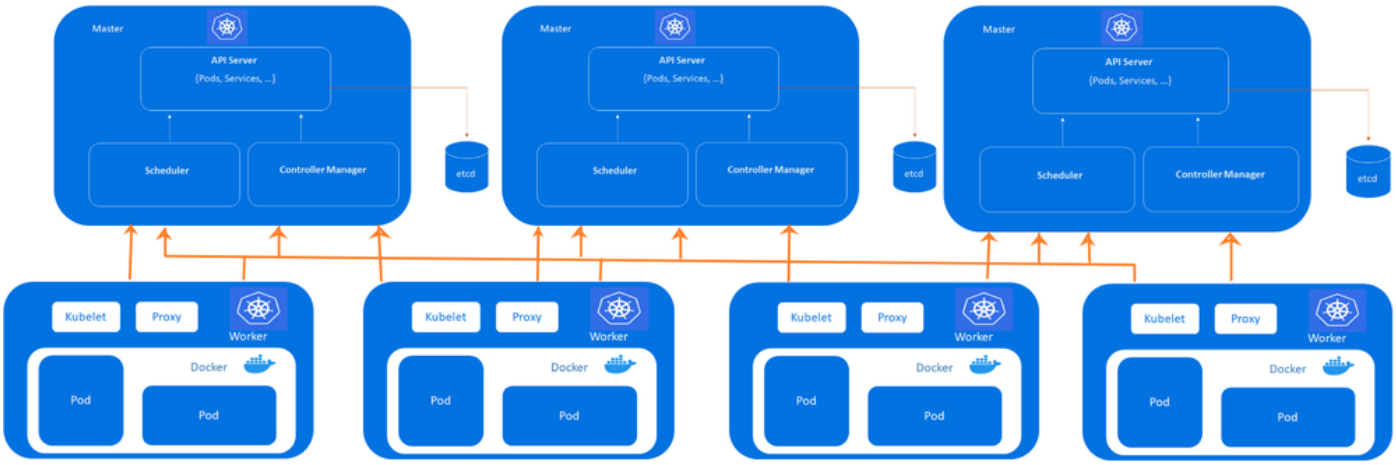
Vagens:
i. Um pod serve como uma única instância de aplicação e é considerado a menor unidade no modelo de objeto do Kubernetes.

Host Bastion:
i. O computador geralmente hospeda um único aplicativo ou processo, por exemplo, um servidor proxy ou balanceador de carga, e todos os outros serviços são removidos ou limitados para reduzir a ameaça ao computador.

Arquitetura multi-master do Kubernetes

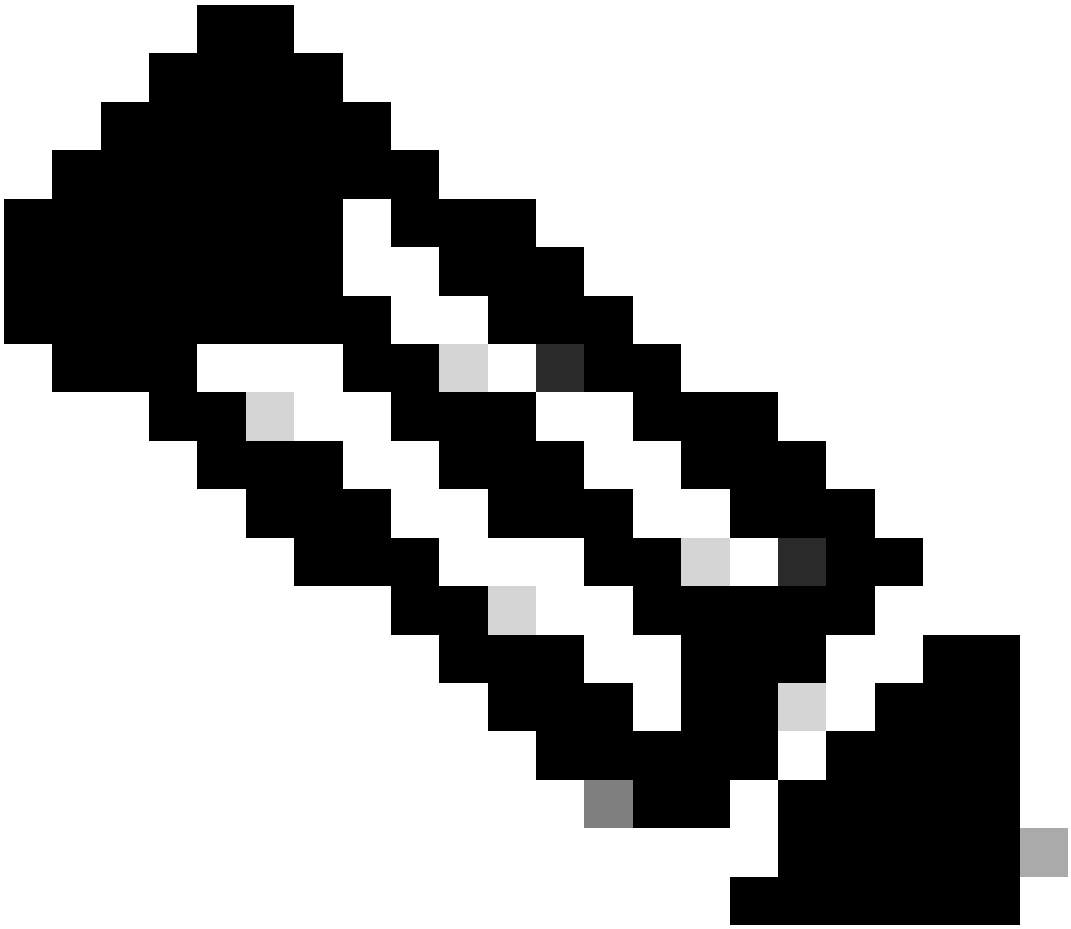
Cluster:

- 8 - Total de nós
- 3 - Nós mestres
- 4 - Nós de trabalho
- 1 - Servidor Bastion (Balanceador de Carga)



Cluster Kubernetes altamente disponível com três planos de controle

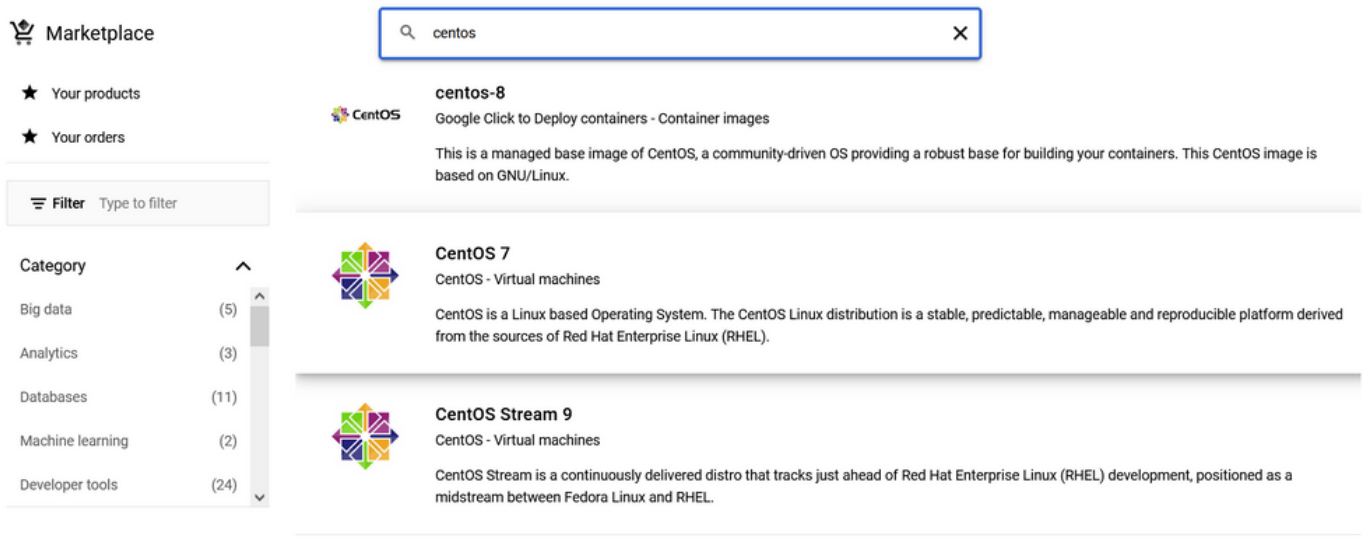
---



Note: Configure o VPC no GCP antes de provisionar as VMs. Consulte [VPC em GCP](#).

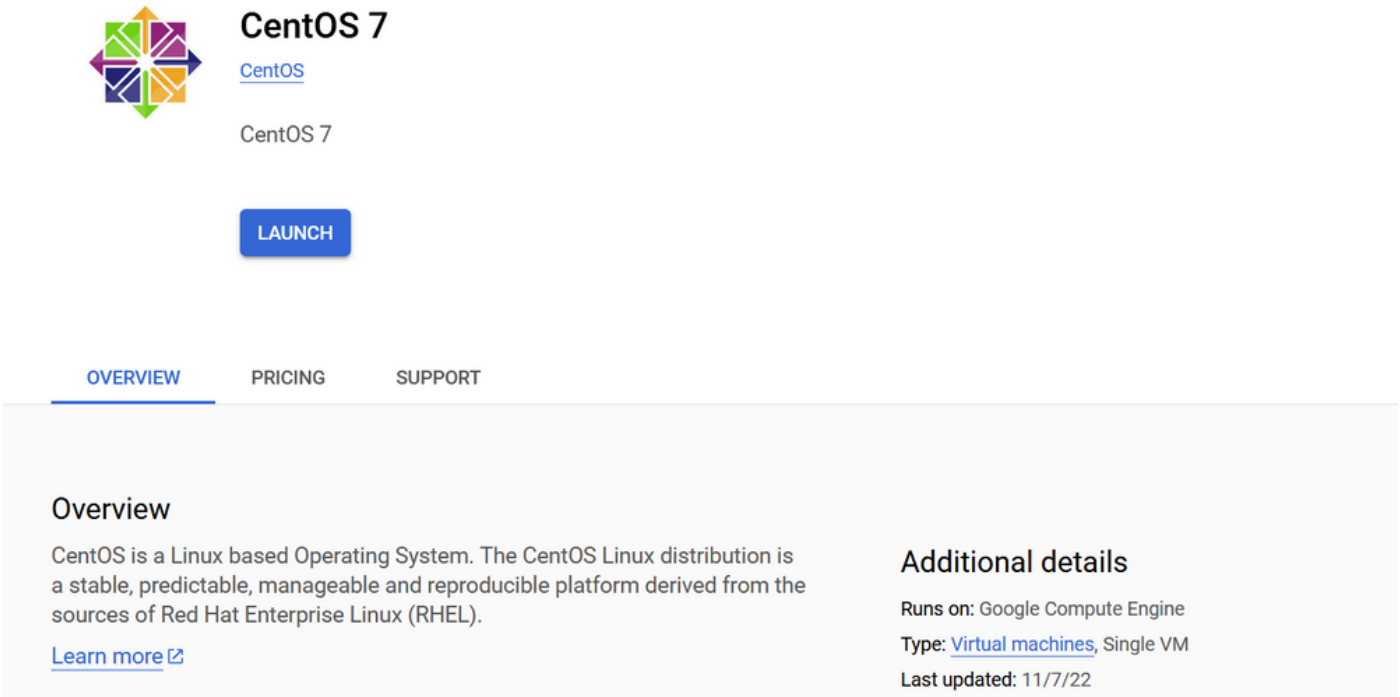
Fornecimento de máquinas virtuais em GCP

Na provisão GCP, um Centos7 do mercado GCP.



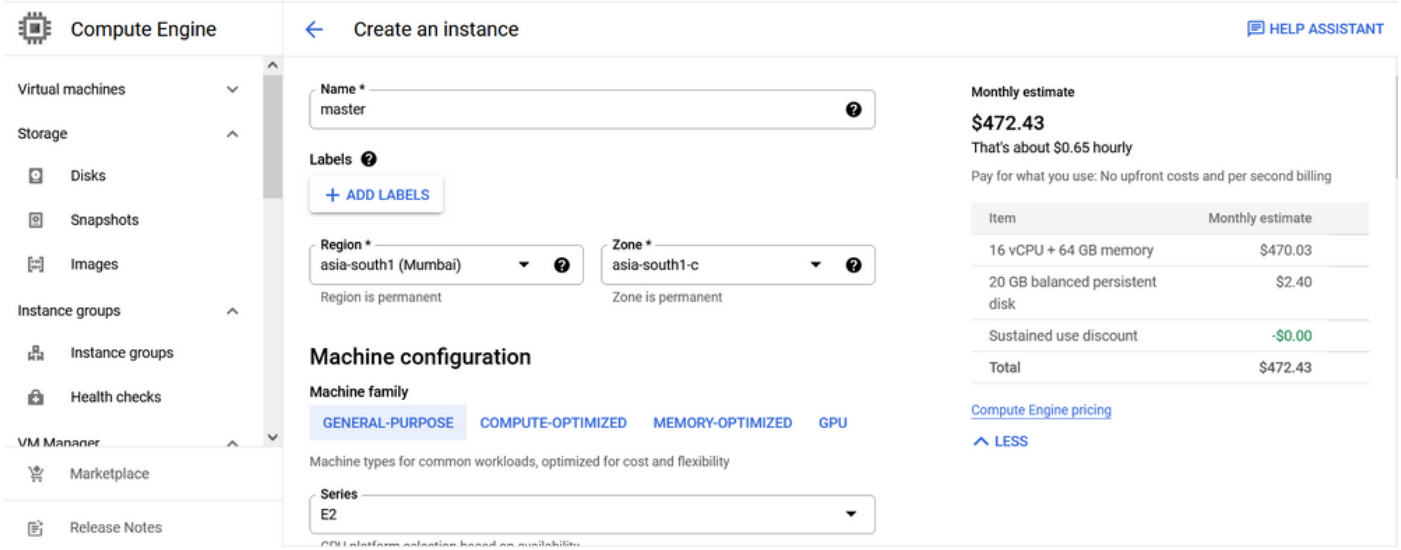
Mercado de Centos em GCP

Clique em **.Launch**



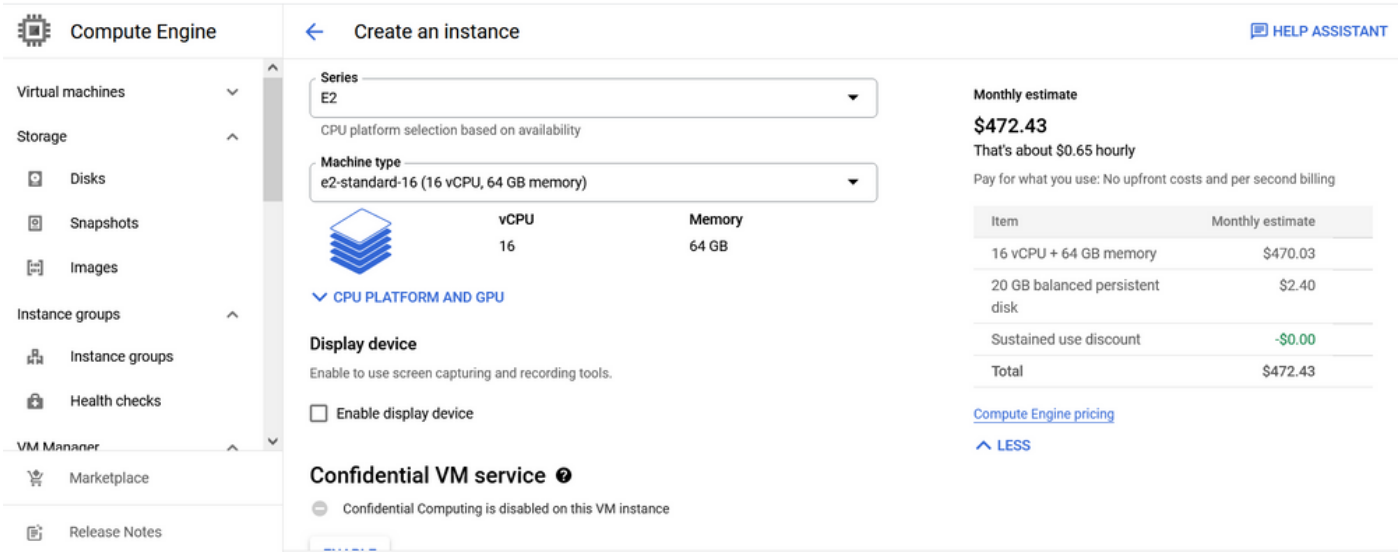
Máquina virtual Centos 7

Escolha a região de acordo com a acessibilidade mais próxima. Neste laboratório, a região está configurada como Mumbai.



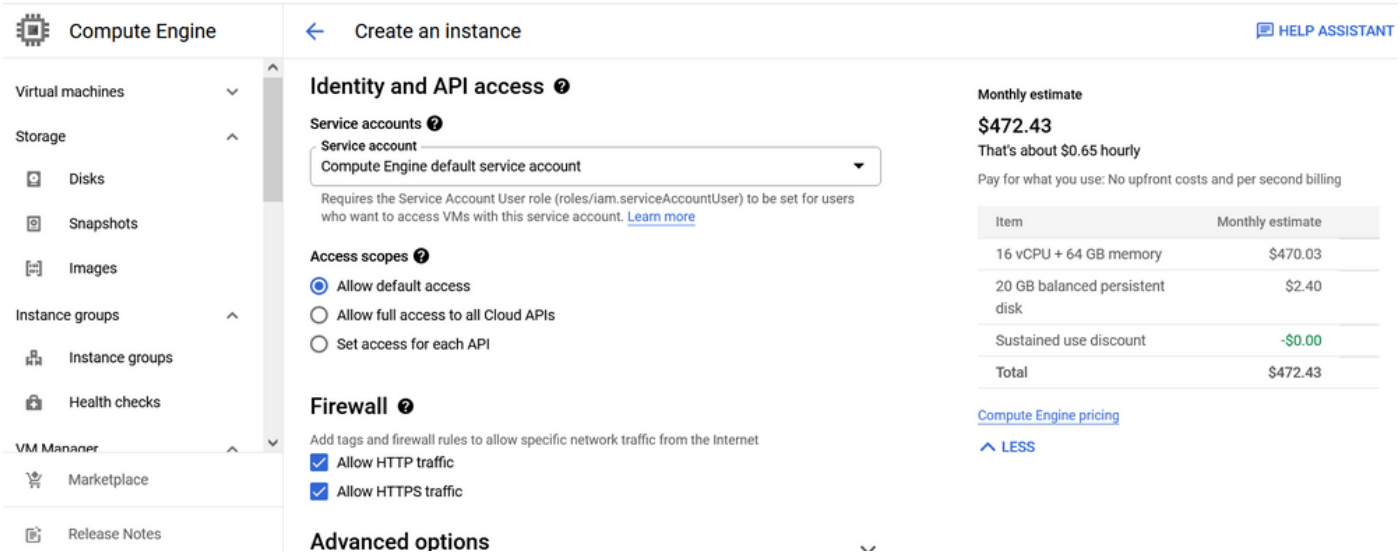
Configuração do mecanismo de computação Centos7

A configuração da máquina é da série E2 de uso geral e do tipo de máquina **e2-standard-16 (16 vCPU, 64 GB memory).**



Configuração de recursos do Centos 7

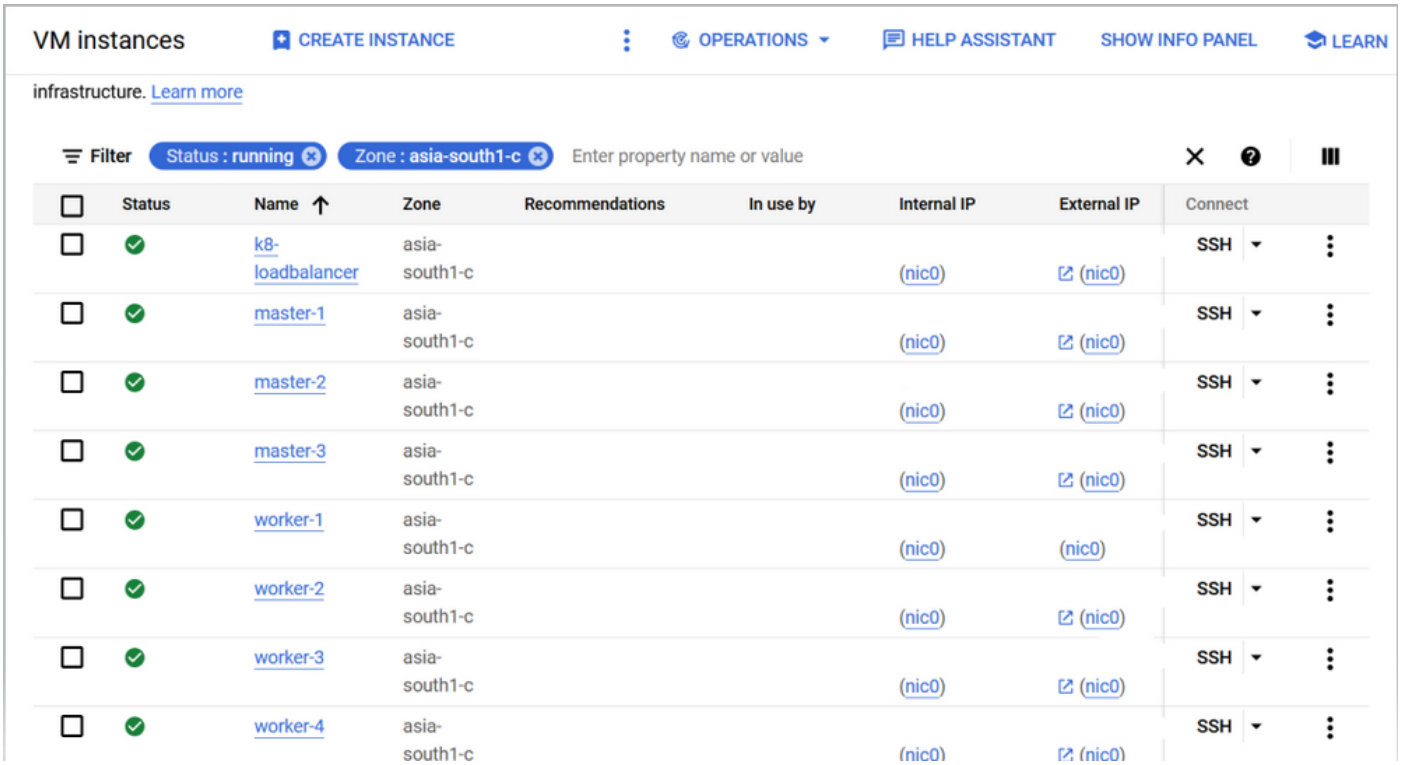
Selecione **Allow default access** e para firewall **Allow HTTP traffic** e **Allow HTTPS traffic.**



Configuração de rede do Centos 7

Clique em .**Create**

Da mesma forma, crie 8 nós conforme mostrado aqui.



Implantação multi-master na plataforma de nuvem do google

IPs privados:

No GCP, os IPs públicos e privados são atribuídos automaticamente.

```
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
```

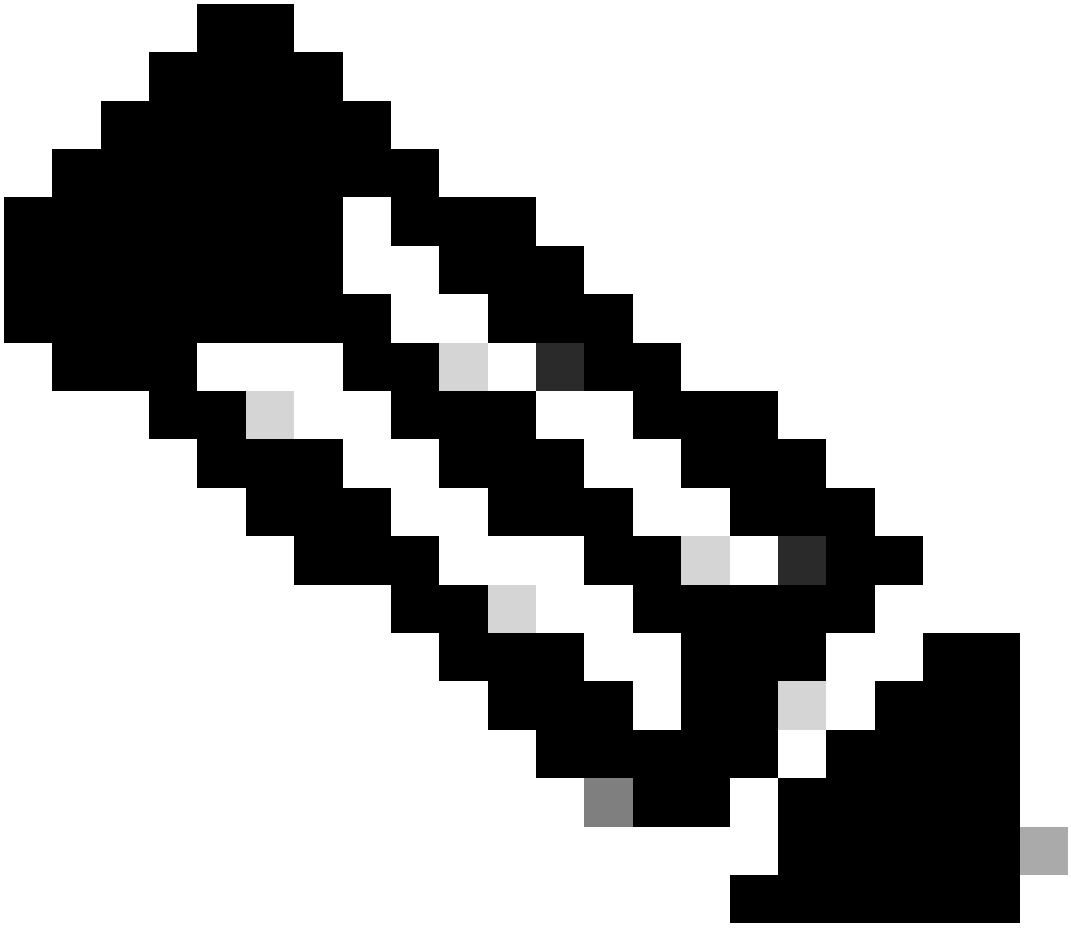
Visão geral de alto nível

Em todos os nós (mestre, trabalhador, bastião):

1. Abra as portas de firewall necessárias no servidor Linux e configure as configurações de segurança.

Em nós mestres em implantações multimestre:

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



Note: Além disso, verifique se as portas no firewall GCP são permitidas.

2. Defina as configurações de rede necessárias (DNS local, nomes de host, NTP).

Servidor Bastion:
1. Configure um proxy de alta disponibilidade.
2. Adicione a configuração de servidor front-end e back-end ao **haproxy.conf** arquivo.
3. Reinicie o **haproxy** serviço.

Etapas comuns para nós mestre e de trabalho:
1. Instale e configure o encaixe.
2. Instale e configure o Kubernetes.

Somente nos nós mestres:
1. Inicialize o novo cluster do Kubernetes (**kubeadm init**).
2. Instale **Calico** um plug-in de rede (usado especificamente para o serviço DNS básico nos nós principais).

3. Use este comando para unir os nós-mestre a um nó-mestre.

```
kubeadm join

        :6443 --token

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

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

4. Valide as informações do cluster com o **kubectl get nodes**comando .

Somente nos nós de trabalho:

1. Use este comando para unir o nó worker ao nó mestre.

```
kubeadm join

        :6443 --token

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

2. Após a junção bem-sucedida, valide as informações do cluster nos nós mestres com este

comando **kubectl get nodes**.

Configurações de baixo nível

Configuração de rede

1. Altere a senha raiz, se desconhecida, com este comando:

```
passwd
```

2. Altere os nomes de host, se necessário, com este comando.

```
hostnamectl set-hostname
```

3. Configure o DNS Local.

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

```
4. Ative crony para serviços NTP com este comando.
```

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

```
2. Verifique o status com este comando.
```

```
systemctl status chronyd
```

```
3. Verifique as fontes NTP com este comando.
```

```
chronyc sources -v
```

Servidor Bastion

Etapa 1. Verifique as atualizações.

```
sudo yum check-update
```

Etapa 2. Instalar atualizações se não estiverem atualizadas.

```
yum update -y
```

Etapa 3. Instalar utilitários yum.

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

Etapa 4. Instale o pacote haproxy.

```
yum install haproxy -y
```

Etapa 5. Adicione essa configuração sob os padrões em **/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 /
```

Etapa 6. Verificar o arquivo de configuração de forma que ele se pareça com 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 /
```

Etapa 7. Verificar o status 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]#
```

Possíveis erros

1. O serviço HAProxy estará em um estado de falha depois que você fizer alterações de configuração no **haproxy.cfg**. Por exemplo;

```
[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.
```

Resolução

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

Execução:

```
[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 nos Nós Mestre e de Trabalho

Etapa 1. Verifique as atualizações.

```
sudo yum check-update
```

Etapa 2. Instalar Atualizações se não estiverem atualizadas.

```
yum update -y
```

Etapa 3. Instalar utilitários yum.

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

Etapa 4. Instale o Docker.

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

Etapa 5. Ative o encaixe.

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

Etapa 6. Inicie o serviço de encaixe.

```
sudo systemctl start docker
```

Etapa 7. Verifique o status do encaixe.

```
sudo systemctl status docker
```

Saída:

```
[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 o Kubernetes nos nós mestre e de trabalho

Etapa 1. Adicione o repositório do Kubernetes.

```
cat <
```

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

Etapa 2. Desativar **SELinux**.

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

Etapa 3. Instalar **Kubernetes**.

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

Etapa 4. Habilitar **Kubelet**.

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

Etapa 5. Configurar **Pod Network**.

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

Etapa 6. Geração de token:

Saída para os nós mestre (plano de controle) e de trabalho.

Nó mestre

Token: Seu plano de controle do Kubernetes foi inicializado com sucesso!

Para usar o cluster, execute-o como um usuário regular:

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

Como alternativa, se você for o usuário root, poderá executar o.

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

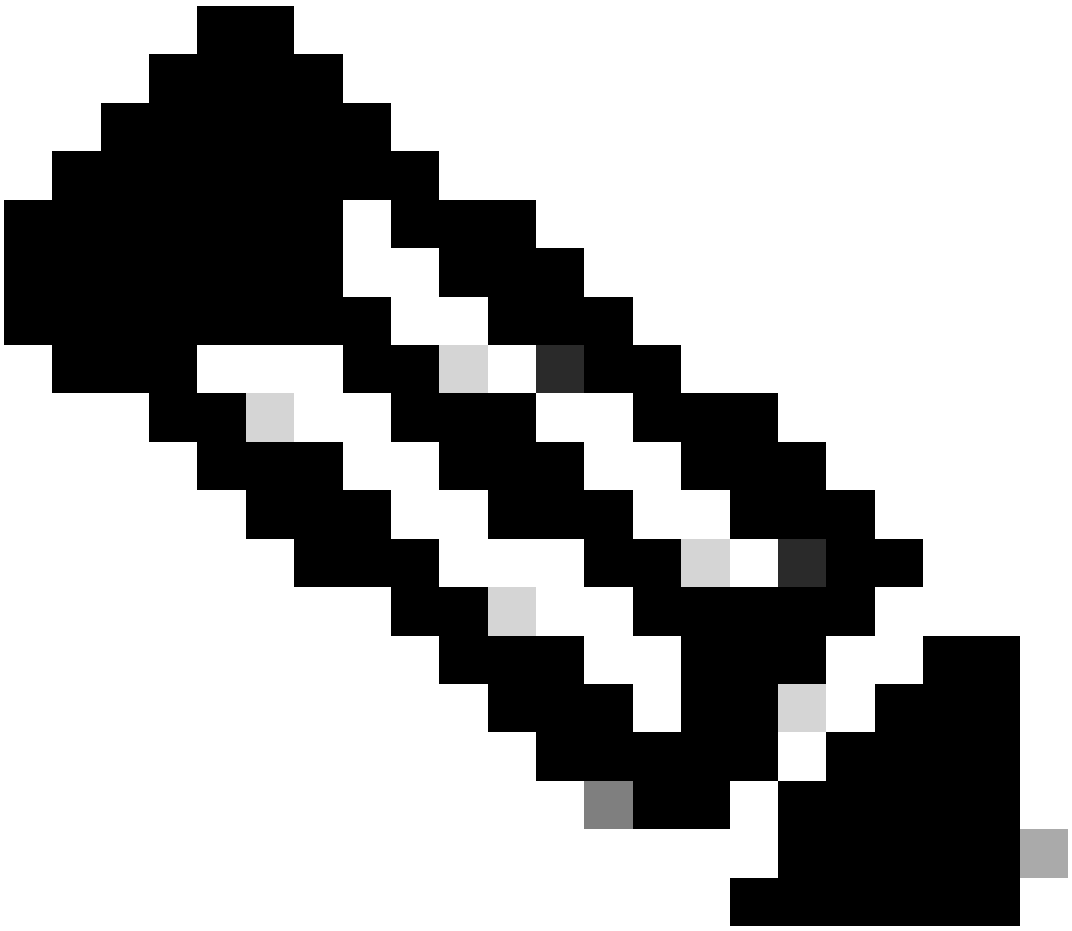
Agora você pode implantar uma rede pod no cluster.

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

Agora você pode unir qualquer número dos nós do plano de controle ou dos nós mestres que executam esse comando em cada um como raiz.

Consulte: [fluxo de trabalho de ingresso no kubeadm]

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.h07kyelronuyOmw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
        --control-plane --certificate-key 66773b960199ef4530461ef4014e1432066902d4a3dee01669d8622579731
```



Note: Observe que a chave de certificado dá acesso a dados sensíveis ao cluster, mantenha-os em segredo!

Como salvaguarda, os certificados carregados são excluídos em duas horas; se necessário, você pode usá-los.

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

Em seguida, você pode unir qualquer número de nós de trabalho e executar isso em cada um como raiz.

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.h07kyelronuyOmw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

Etapa 7. Verificar o serviço Core DNS.

```
[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]#
```

Etapa 8. Verifique o status do nó mestre.

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

Para unir vários nós mestres, este comando join é executado nos nós mestres.

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

Possíveis erros encontrados no momento da geração do token

Erro de 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
```

Erro de resolução de CRI

Etapa 1. Remova o arquivo **config.toml** e reinicie o **container**.

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

Etapa 2. Instalar em contêiner:
Instale o pacote containerd.io dos repositórios oficiais do Docker com esses 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
```

Etapa 3. Configurar recipiente:

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

Etapa 4. Reiniciar contêiner:

```
systemctl restart containerd
```

Erro 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
```

Erro na resolução FileContent—proc-sys-net-ipv4-ip_forward

Defina o valor ip_forward como 1.

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

O serviço DNS principal não é executado

```
[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]#
```

Resolução

O DNS central está pendente, o que indica um problema com a rede. Portanto, o Calico precisa ser instalado.

Referência: Calico

Execute estes dois comandos:

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

Nó do trabalhador

No Nó de Trabalho quando o token é obtido do mestre:

Etapa 1. Ative o serviço kubelet.

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

Etapa 2. Associe todos os nós de trabalho com o nó mestre com este comando join.

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

Etapa 3. Se forem encontrados erros relacionados a arquivos ou portas, redefina o kubeadm com este comando.

```
kubeadm reset
```

Depois de redefinir, insira o token do nó mestre.

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

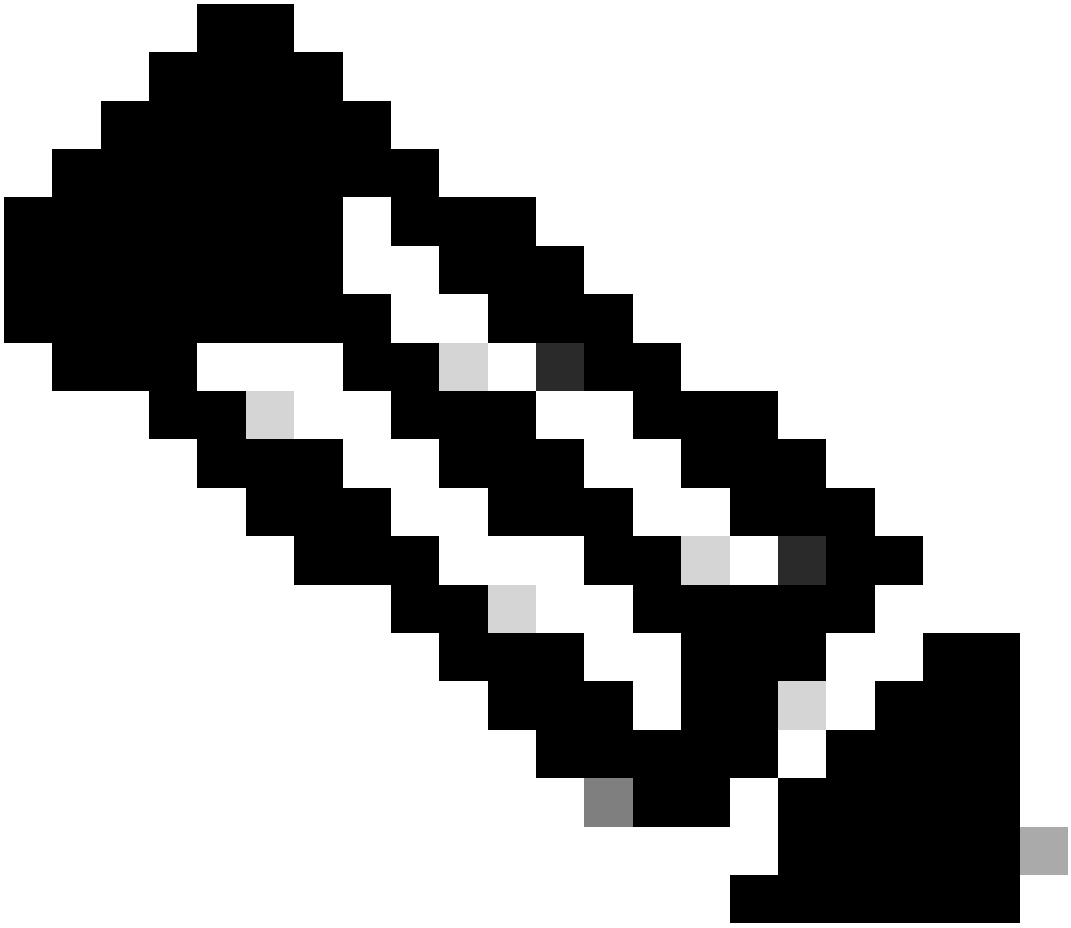
```
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

Saída final

O cluster agora está formado, verifique-o com o 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]#
```

saída de cluster kubernetes de alta disponibilidade



Note: No momento da formação do cluster, somente o controle dos nós principais é configurado. O host bastion não está configurado como um servidor centralizado para monitorar todos os Kubes no cluster.

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.