在Google云平台上使用Centos 7创建多主Kubernetes集群

目录

简介

本文档介绍在Google Cloud Platform(GCP)上实施具有3个主节点和4个辅助节点的Kubernetes集群，该集群具有用作负载均衡器的堡垒主机。

先决条件

要求

Cisco 建议您了解以下主题：

- Linux
- 多克和库伯内特
- Google云平台

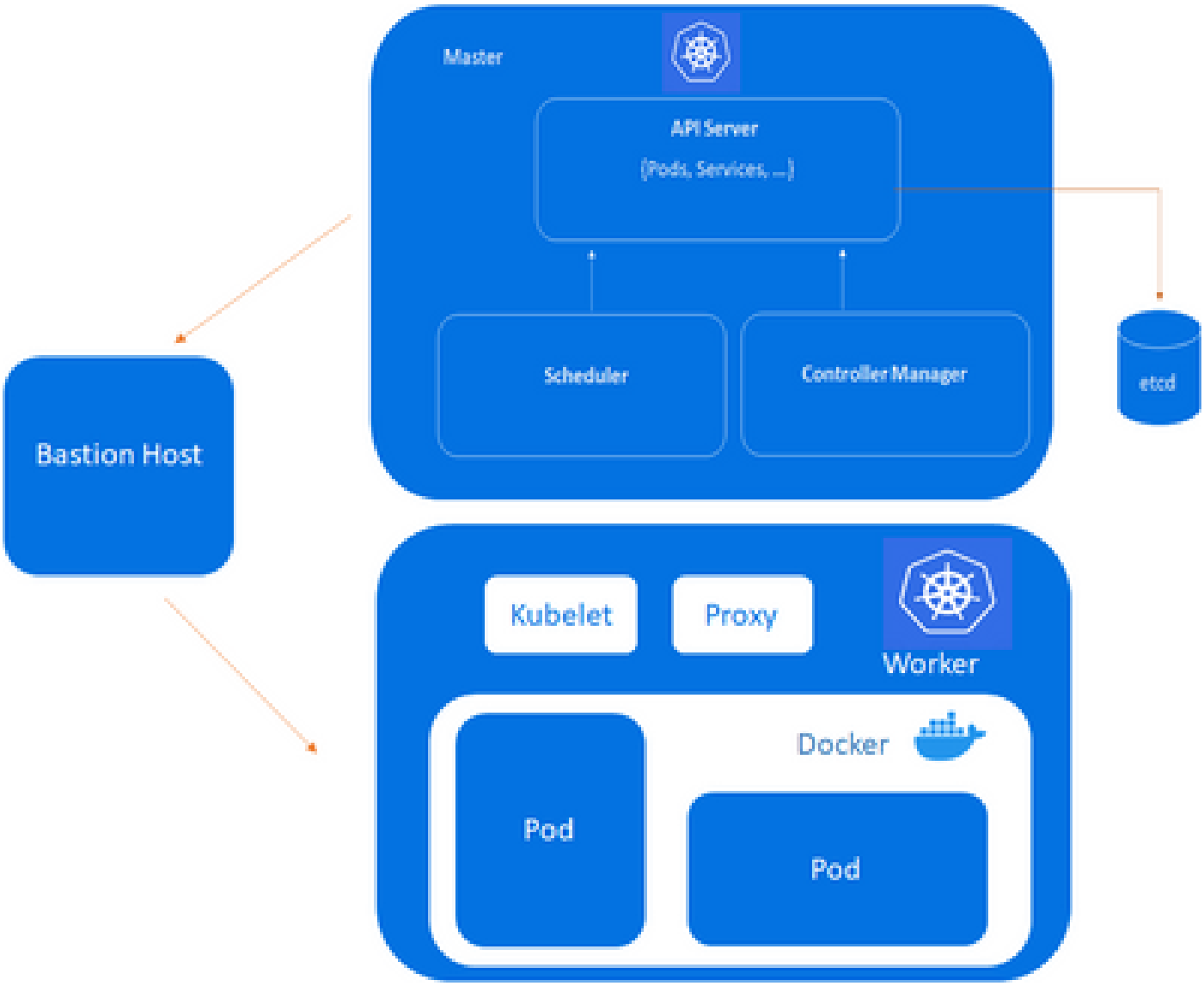
使用的组件

本文档中的信息基于：

- OS - Centos 7虚拟机
- 机器系列(e2-standard-16):
  - vCPU - 16个vCPU
  - RAM - 64 GB

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

网络图



Kubernetes主节点组件

Kube-apiserver:
i.提供充当Kubernetes控制平面前端的API。
二、它处理用于确定请求是否有效的外部和内部请求，然后处理该请求。
三。API可通过命令行界kubectl面或其他工具（如）以及kubeadmREST调用访问。

Kube-scheduler:
i.此组件根据自动化工作流程和用户定义的条件在特定节点上安排Pod。

Kube-controller-manager:
i.Kubernetes控制器管理器是一个控制环路，用于监控和调整Kubernetes集群的状态。
二、它接收有关集群的当前状态及其内部对象的信息，并发送指令以将集群移动到集群运营商的期望状态。

etcd:
i.一个键值数据库，其中包含有关集群状态和配置的数据。
二、Etcd是容错和分布式的。

Kubernetes工作节点组件

库贝莱：
i.每个节点都包含kubelet一个，这是一个可以与Kubernetes控制平面通信的小型应用程序。
二、确保kubeletPod配置中指定的容器在特定节点上运行，并管理其生命周期。
三。它执行由控制平面命令执行的操作。

Kube-proxy:
i.所有计算节点都包kube-proxy含一个网络代理，可促进Kubernetes网络服务。
二、它处理集群外部和内部的所有网络通信，并在操作系统的数据包过滤层转发流量或应答。

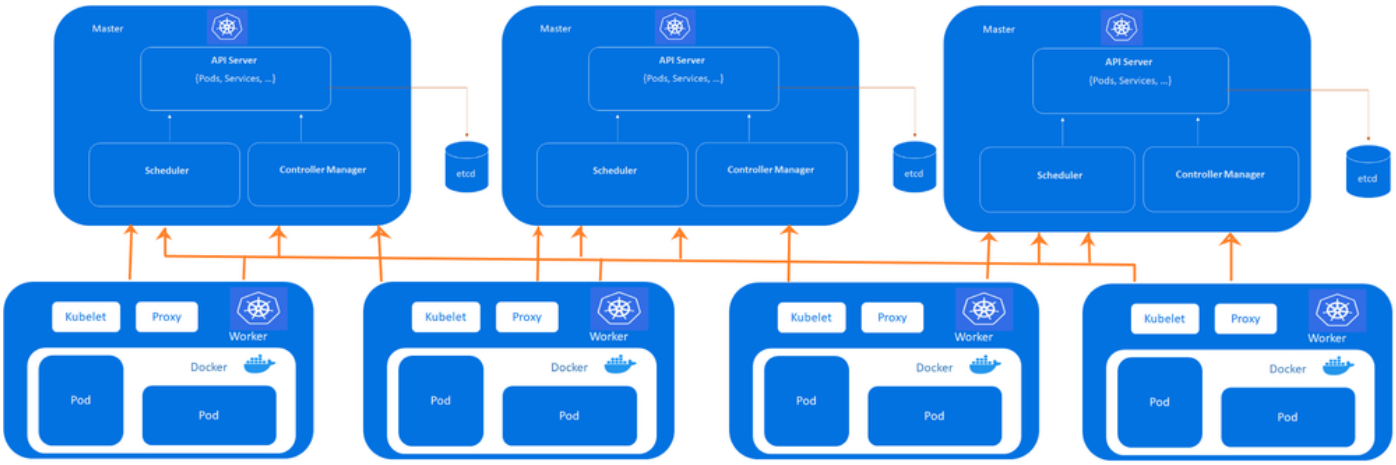
Pod:
i.Pod用作单个应用实例，并且被视为库伯内特斯的对象模型的最小单元。

Bastion主机：
i.计算机通常托管单个应用或进程，例如代理服务器或负载均衡器，并删除或限制所有其他服务以减少对计算机的威胁。

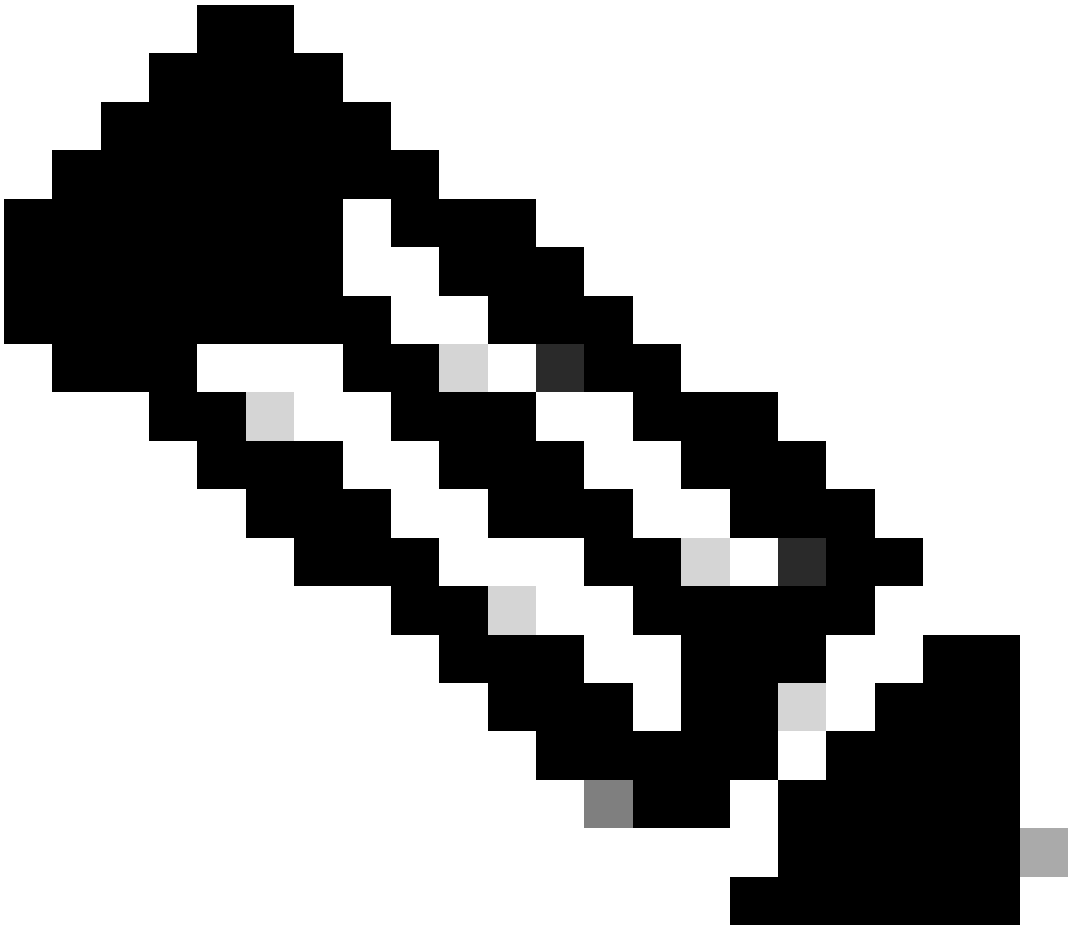
Kubernetes多主架构

集群：

- 8 — 节点总数
- 3 — 主节点
- 4 — 工作节点
- 1 — 基准服务器（负载均衡器）



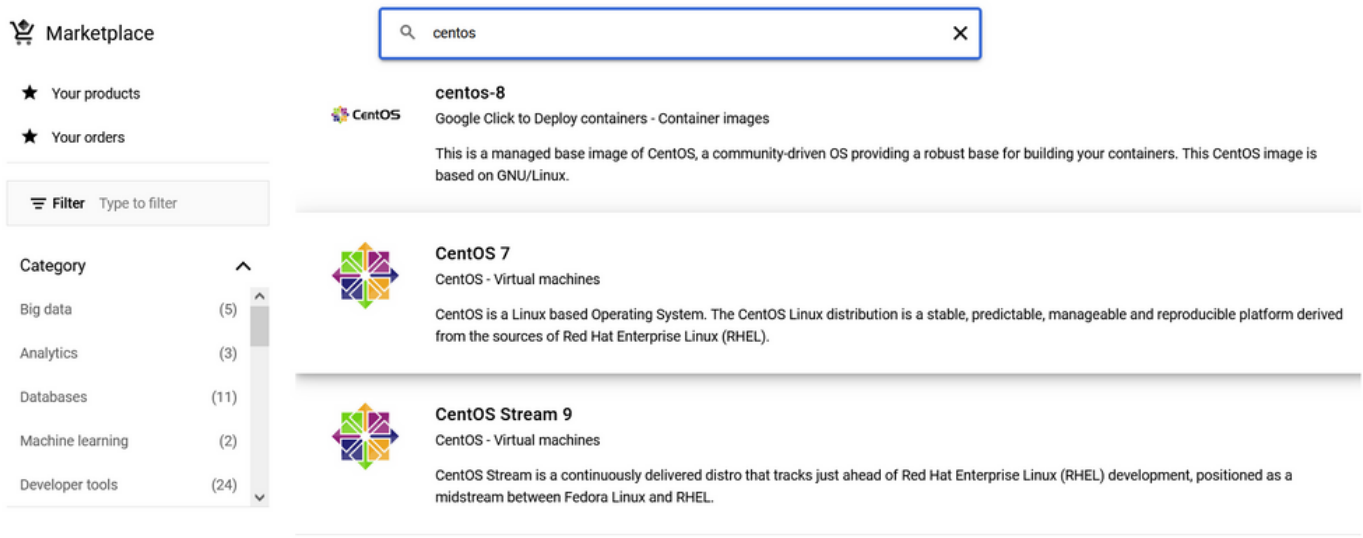
具有三个控制平面的高可用性Kubernetes集群



注意：在调配VM之前，在GCP上配置VPC。参考GCP上的VPC。

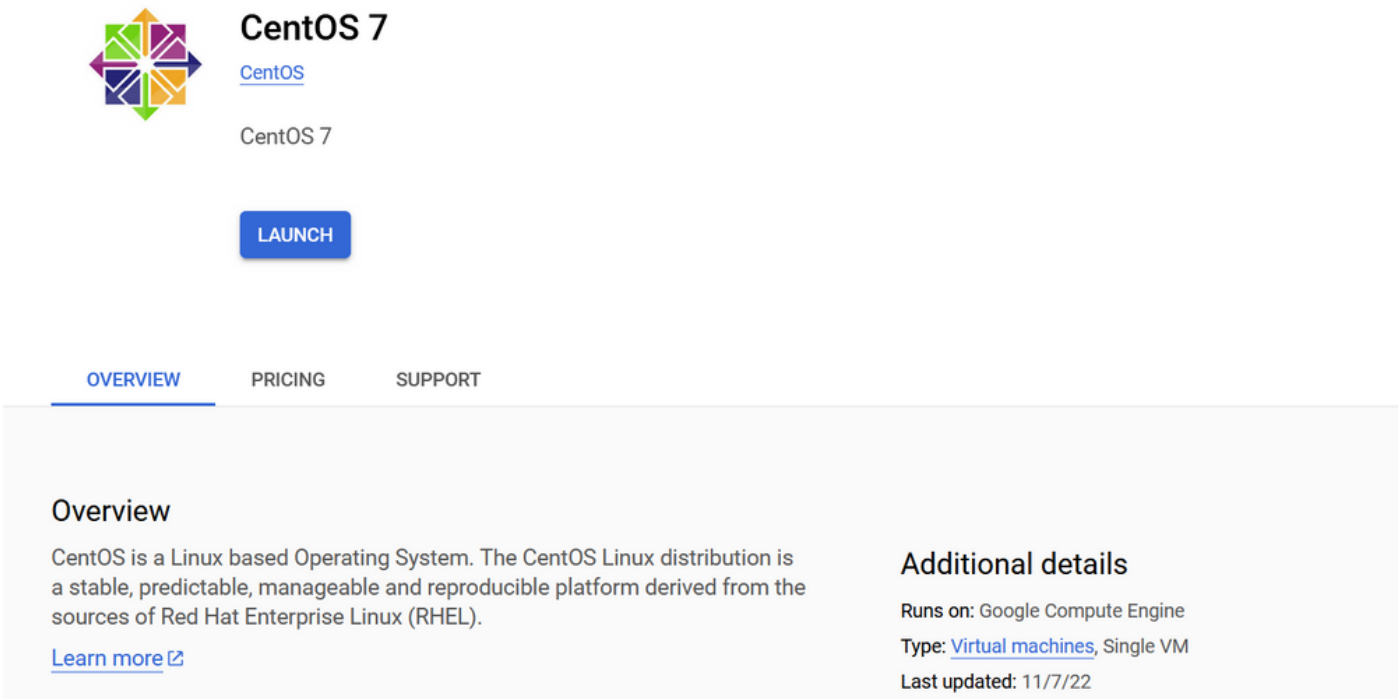
在GCP上调配虚拟机

在GCP上，从GCP市场调配Centos7。



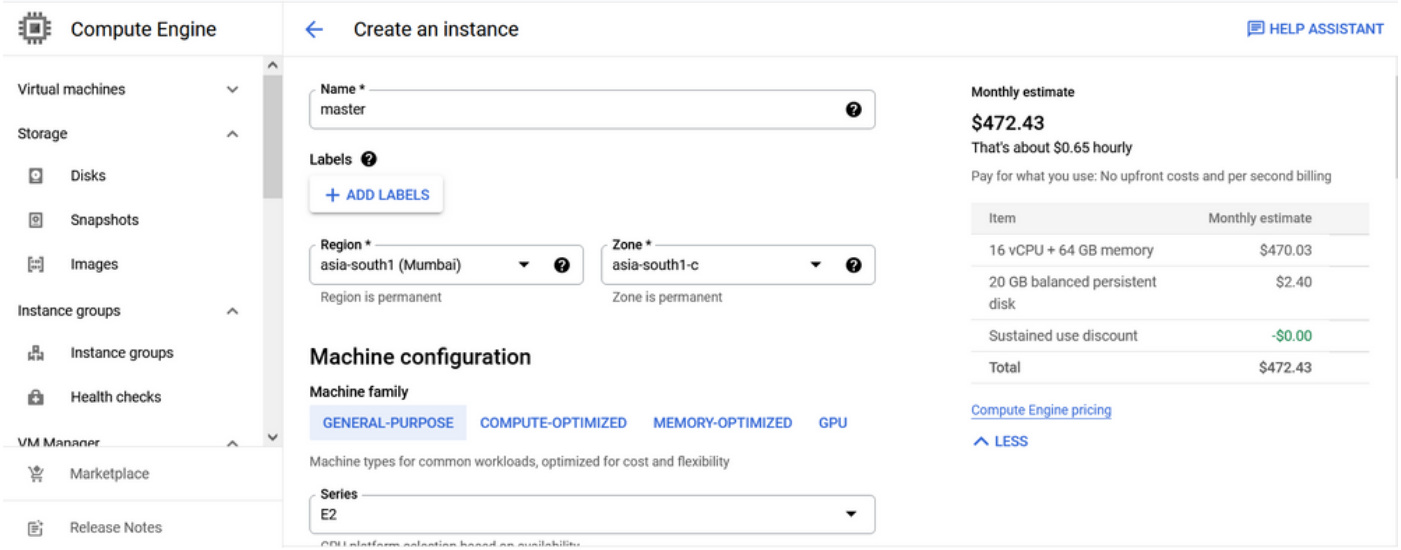
GCP上的Centos市场

单击。**Launch**



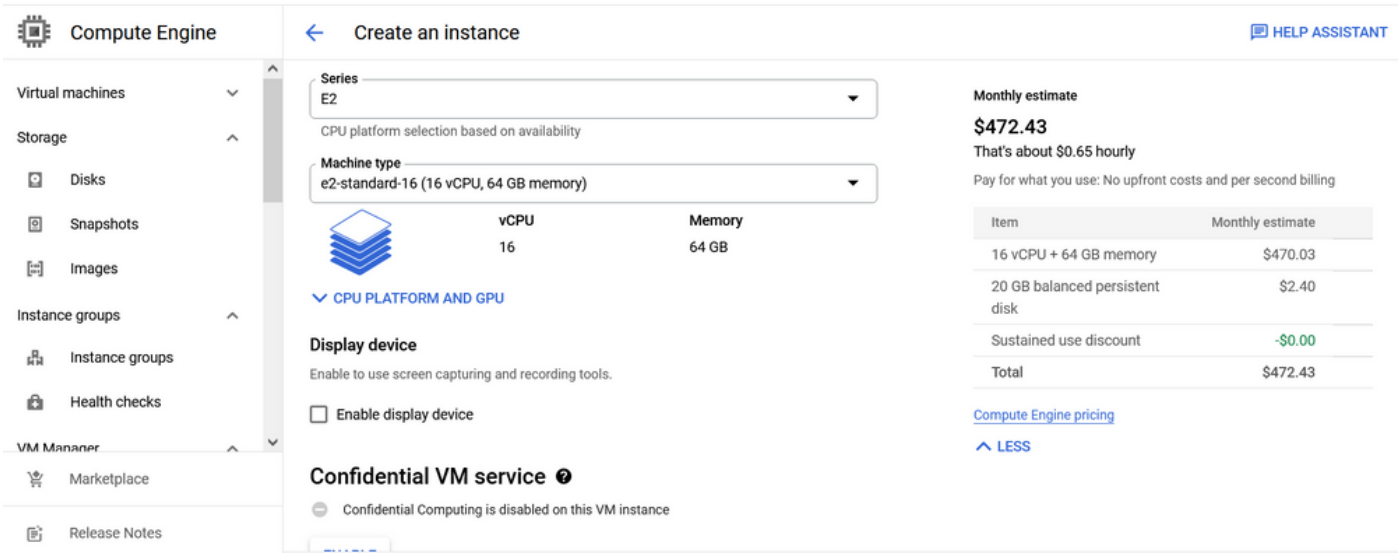
Centos 7虚拟机

根据最近的连通性选择区域。在本实验中，该区域被配置为Mumbai。



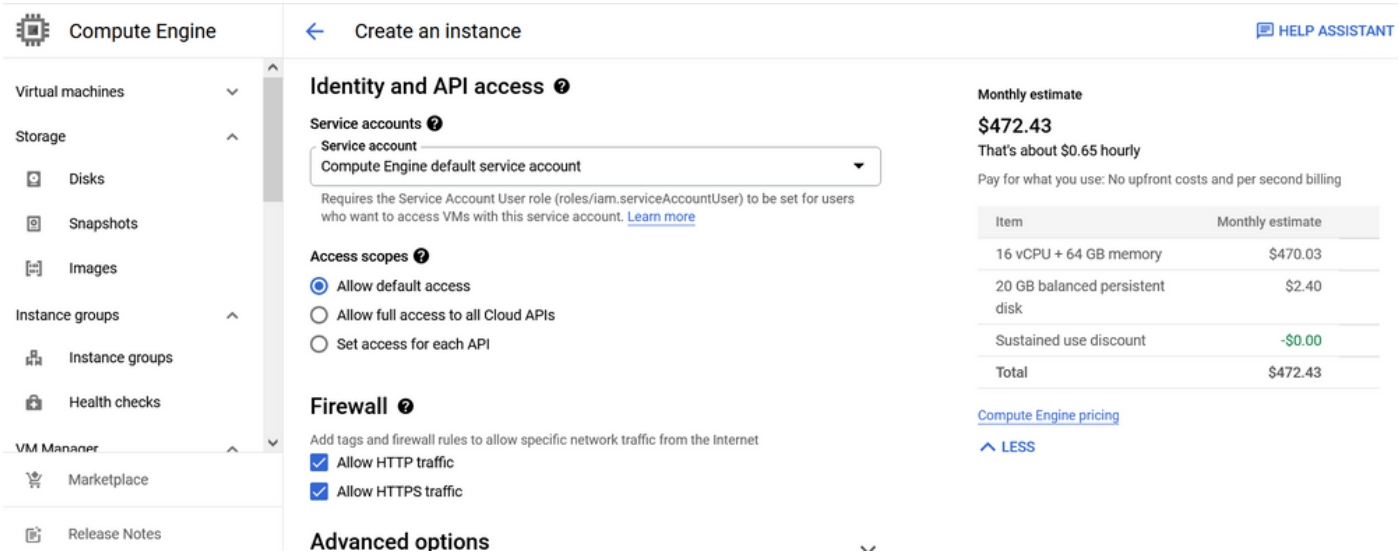
Centos7计算引擎配置

机器配置是通用E2系列和机器类**e2-standard-16 (16 vCPU, 64 GB memory)**型。



Centos 7资源配置

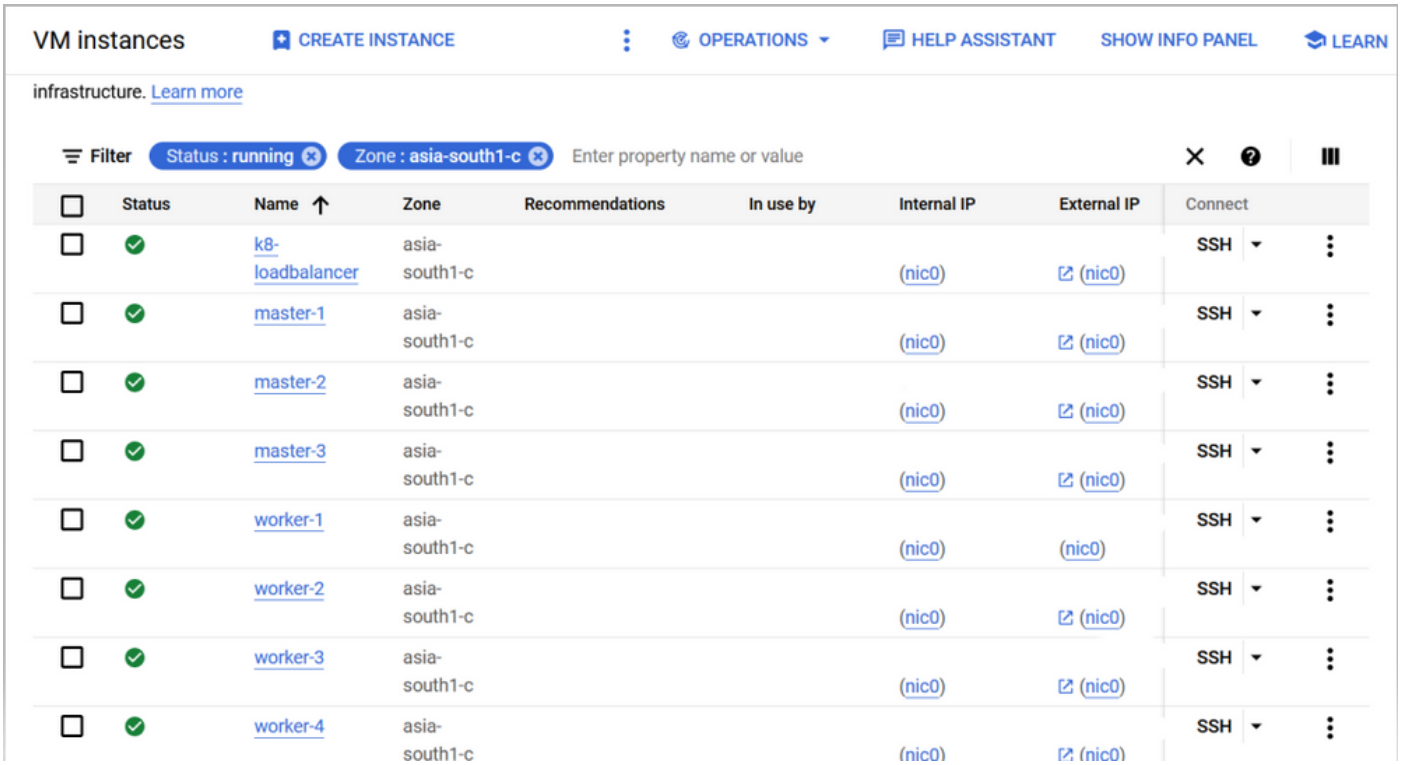
对于**Allow default access**防火墙，选择**Allow HTTP traffic**和，**Allow HTTPS traffic**和。



Centos 7网络配置

单击。**Create**

同样，创建8个节点，如下所示。



Google云平台上的多主机部署

私有IP:

在GCP上，私有IP和公有IP自动分配。

```
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
```

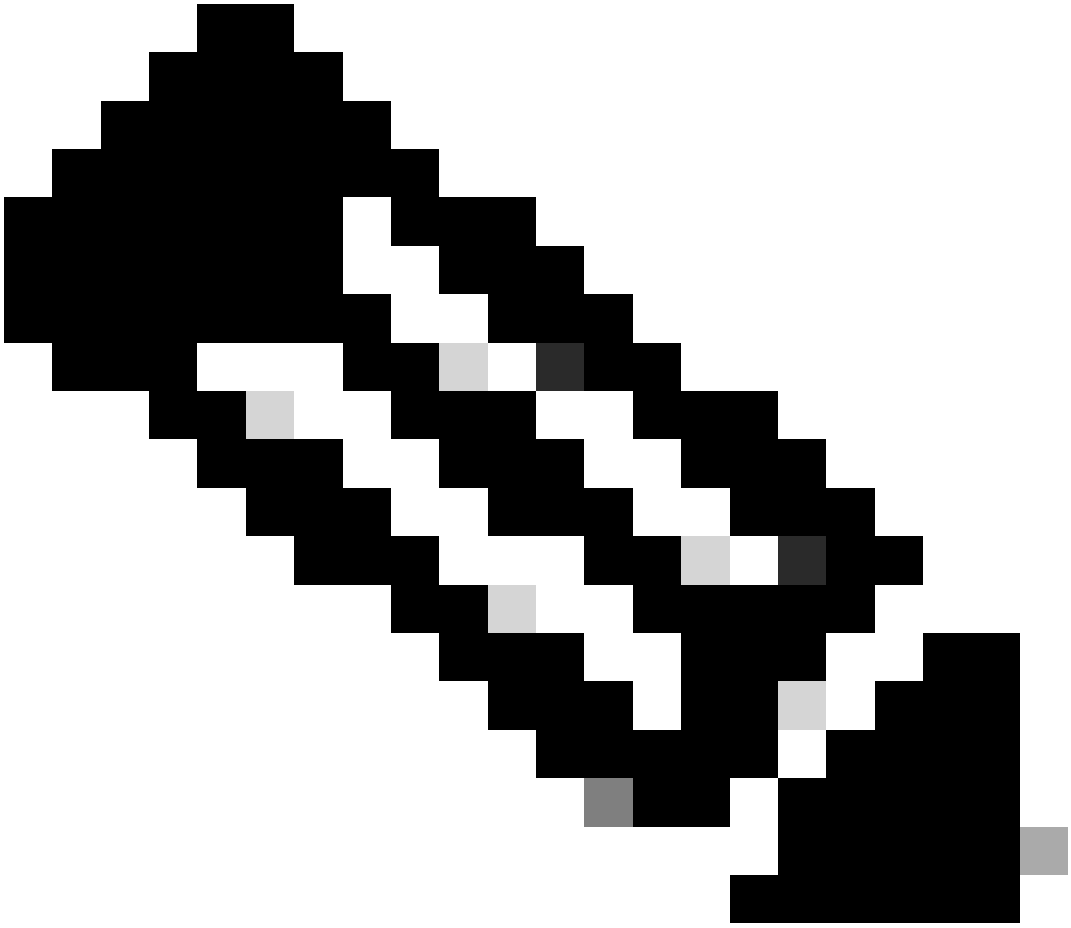
高级概述

在所有节点（主节点、辅助节点、辅助节点）上：

1.在Linux服务器上打开所需的防火墙端口并配置安全配置。

在多主部署的主节点上：

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



注意：此外，请确保允许GCP防火墙上的端口。

2.配置所需的网络设置（本地DNS、主机名、NTP）。

Bastion服务器：
1.设置高可用性代理。
2.在文件中添加前端和后端服务器**haproxy.conf** 配置。
3.重新启动服**haproxy**务。

主节点和辅助节点的一般步骤：
1.安装和配置docker。
2.安装和配置Kubernetes。

仅在主节点上：
1.初始化新的Kubernetes群集(**kubeadm init**)。
2.安装**Calico**网络插件（专门用于主节点上的核心DNS服务）。
3.使用此命令将主节点与主节点联合。

```
kubeadm join

        :6443 --token

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

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

4.使用命令验证集群信**kubectl get nodes**息。

仅在工作节点上：

1.使用此命令将工作节点加入主节点。

```
kubeadm join

        :6443 --token

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

2.成功加入后，使用此命令验证主节点上的集群信**kubectl get nodes**息。

低级配置

网络配置

1.如果使用此命令未知，请更改根密码：

```
passwd
```

2.如有必要，请使用此命令更改主机名。

```
hostnamectl set-hostname
```

3.配置本地DNS。

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

4.使用此命令启用NTP服务同步。

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

2.使用此命令验证状态。

```
systemctl status chronyd
```

3.使用此命令检查NTP源。

```
chronyc sources -v
```

Bastion服务器

步骤1.检查更新。

```
sudo yum check-update
```

步骤2.安装更新（若不是最新的）。

```
yum update -y
```

步骤3.安装yum实用程序。

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

步骤4.安装haproxy程序包。

```
yum install haproxy -y
```

步骤5.在默认情况下添加以下配**/etc/haproxy/haproxy.cfg** 置：

```
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 /
```

步骤6. 检验配置文件，使其类似于以下命**vi /etc/haproxy/haproxy.cfg**令：

```
---------------------------------------------------------------------
# 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 /
```

步骤7. 检验haproxy的状态：

```
[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]#
```

可能的错误

1.在中更改配置后，HAProxy服务处于故障状态**haproxy.cfg**。例如；

```
[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.
```

分辨率

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

执行：

```
[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]#
```

在主节点和工作节点上安装Docker

步骤1.检查更新。

```
sudo yum check-update
```

步骤2.安装更新（若不是最新的）。

```
yum update -y
```

步骤3.安装yum实用程序。

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

步骤4.安装Docker。

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

步骤5.启用docker。

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

步骤6.启动docker服务。

```
sudo systemctl start docker
```

步骤7.检查docker状态。

```
sudo systemctl status docker
```

输出：

```
[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]#
```

在主节点和工作节点上安装Kubernetes

步骤1. 添加Kubernetes存储库。

```
cat <
```

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

步骤2. 禁SELinux用。

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

步骤3. 安Kubernetes装。

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

步骤4. 启用Kubelet。

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

步骤5. 配**Pod Network**置。

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

步骤6. 令牌生成：

主（控制平面）和工作节点的输出。

主节点

令牌：您的Kubernetes控制平面已成功初始化！

要使用您的集群，请以常规用户身份运行以下命令：

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

或者，如果您是根用户，则可以运行。

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

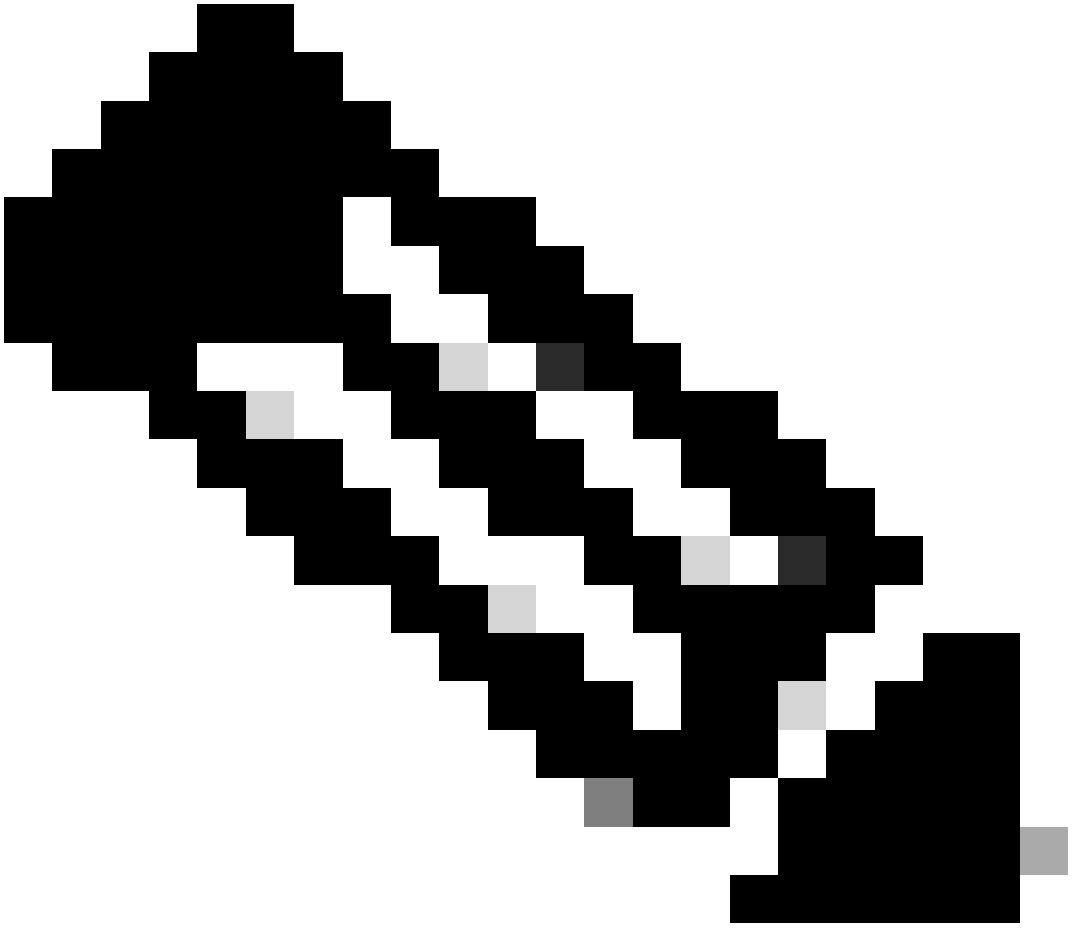
您现在可以将Pod网络部署到集群。

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

现在，您可以加入任意数量的控制平面节点或作为根节点在每个节点上运行此命令的主节点。

请参阅：[kubeadm join workflow]

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.h07kyelronuy0mw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
        --control-plane --certificate-key 66773b960199ef4530461ef4014e1432066902d4a3dee01669d8622579731
```



注意：请注意，证书密钥提供对集群敏感数据的访问权限，请保守其秘密！

作为保护措施，上传的证书将在两小时内删除；如有必要，可以使用它们。

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

然后，您可以加入任意数量的工作节点，并在每个工作节点上作为根节点运行此操作。

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

步骤7. 检验核心DNS服务。

```
[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]#
```

步骤8. 检验主节点的状态。

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

要加入多个主节点，在主节点上执行此加入命令。

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

生成令牌时可能遇到的错误

错误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
```

错误CRI解决方案

步骤1. 删除**config.toml**文件，然后重新启动**containerd**。

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

步骤2.安装容器：
使用这些命令从官方Docker存储库安装containerd.io软件包。

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

步骤3.配置容器：

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

步骤4.重新启动容器：

```
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
```

错误文件内容 — proc-sys-net-ipv4-ip_forward解析

将ip_forward值设置为1。

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

核心DNS服务未运行

```
[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]#
```

分辨率

核心DNS处于挂起状态，表明存在网络问题。因此，需要安装Calico。

参考来源：Calico

执行以下两个命令：

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

工作节点

从主节点获取令牌时，在辅助节点上：

步骤1.启用kubelet服务。

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

步骤2.使用此join命令通过主节点加入所有工作节点。

```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuyOmw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

步骤3.如果遇到与文件或端口相关的错误，请使用此命令重置kubeadm。

```
kubeadm reset
```

重置后，从主节点输入令牌。

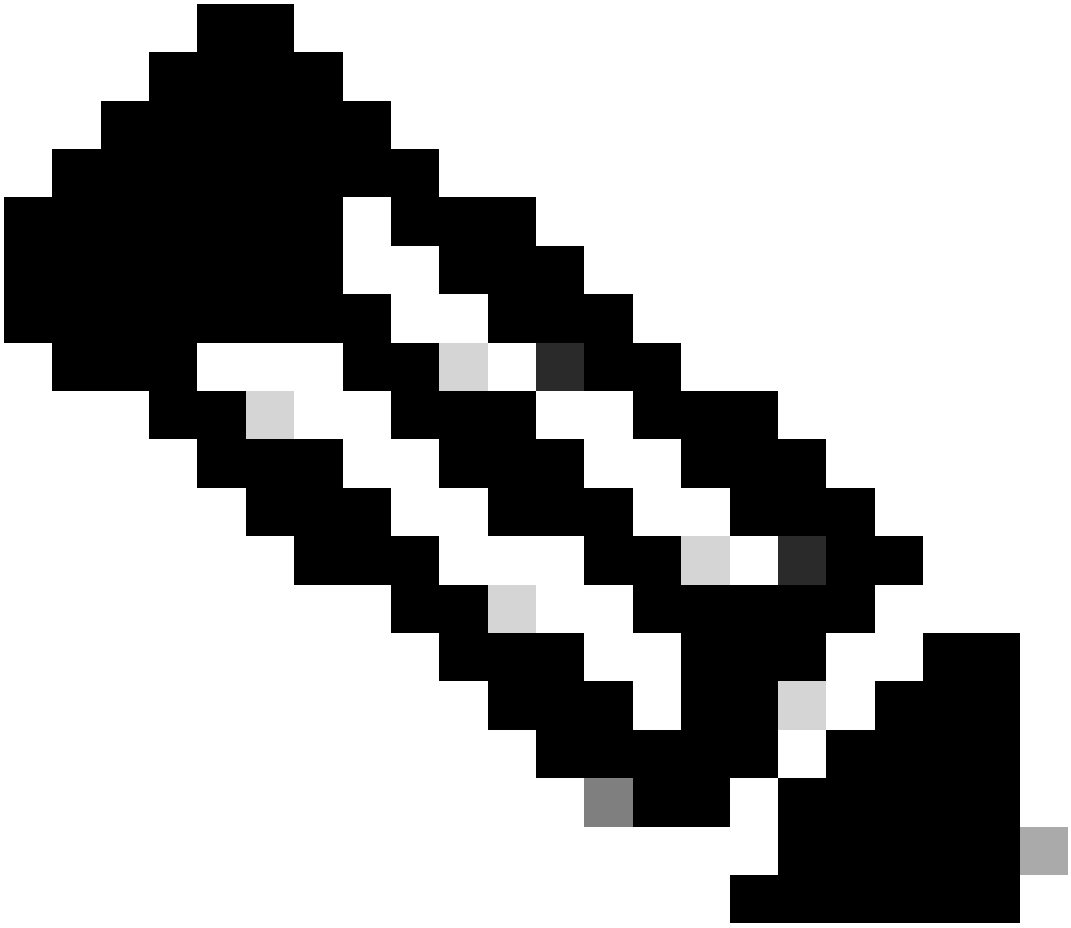
```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuyOmw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

最终输出

集群现已形成，请使用**kubectl get nodes**命令对其进行验证。

```
[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]#
```

高可用性*kubernetes*群集输出



注意：在形成集群时，仅配置来自主节点的控制。堡垒主机未配置为集中式服务器以监控集群中的所有库伯。

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。