Multi-Master Kubernetes Cluster maken met Centos 7 op Google Cloud Platform

Inhoud

Inleiding

Dit document beschrijft de implementatie van het Kubernetes-cluster met 3 master- en 4 werkersknooppunten met een bastion-host die fungeert als een load balancer op het Google Cloud Platform (GCP).

Voorwaarden

Vereisten

Cisco raadt kennis van de volgende onderwerpen aan:

- Linux
- Dockers en Kubernetes
- Google Cloud Platform

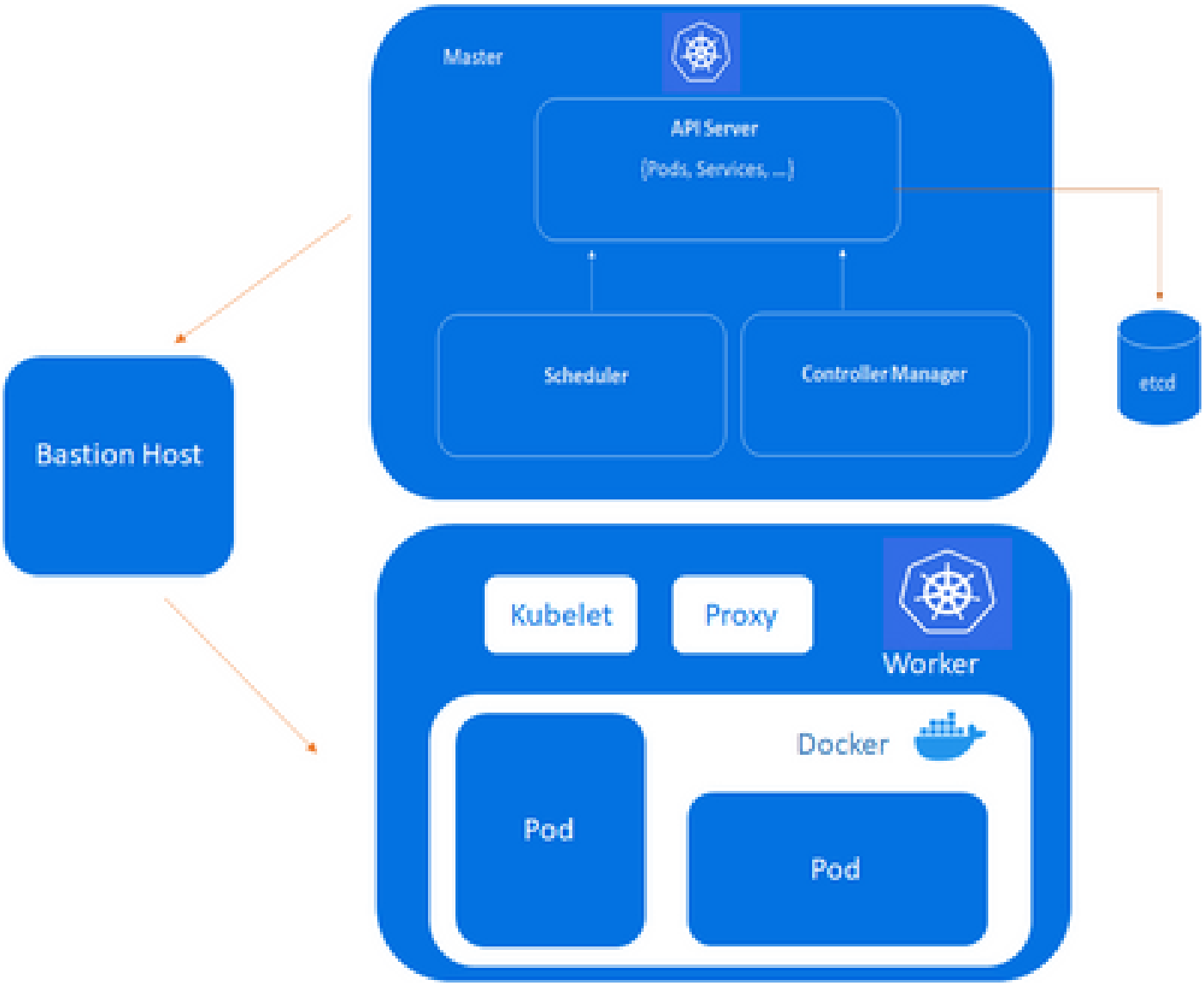
Gebruikte componenten

De informatie in dit document is gebaseerd op:

- Besturingssysteem - Centos 7 virtuele machine
- Machinefamilie (e2-standaard-16):
  - vCPU's - 16 vCPU
  - RAM - 64 GB

De informatie in dit document is gebaseerd op de apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde (standaard)configuratie. Als uw netwerk live is, moet u zorgen dat u de potentiële impact van elke opdracht begrijpt.

Netwerkdiagram



Bastion host, Kube-Master, Kube-node

Kubernetes-hoofdknooppuntcomponenten

Kube-apiserver:
i. Biedt een API die dient als de voorkant van een Kubernetes-besturingsvlak.
ii. Het behandelt externe en interne verzoeken die bepalen of een verzoek geldig is en verwerkt het vervolgens.
iii. De API is toegankelijk via de **kubectl** opdrachtregelinterface of andere hulpmiddelen zoals **kubeadm**en via REST-oproepen.

Kube-scheduler:
i. Deze component plant pods op specifieke knooppunten volgens geautomatiseerde workflows en door de gebruiker gedefinieerde voorwaarden.

Kube-controller-manager:
i. De Kubernetes-controllermanager is een controlelus die de status van een Kubernetes-cluster bewaakt en reguleert.
ii. Het ontvangt informatie over de huidige status van het cluster en de objecten daarin en verzendt

instructies om het cluster naar de gewenste status van de clusteroperator te verplaatsen.

ETCD:
i. Een database met essentiële waarden die gegevens bevat over de status en configuratie van uw cluster.
ii. Etcd is fouttolerant en gedistribueerd.

Kubernetes-werkknooppuntcomponenten

Kubelet:
i. Elk knooppunt bevat een **kubelet**element, dat een kleine toepassing is die kan communiceren met het Kubernetes-besturingsvlak.
ii. De service **kubelet** zorgt ervoor dat containers die in pod-configuratie zijn opgegeven, op een specifieke node worden uitgevoerd en hun levenscyclus beheren.
iii. Het voert de acties uit die door uw controlevliegtuig worden bevolen.

Kube-proxy:
i. Alle compute nodes bevatten **kube-proxy**een netwerkproxy die Kubernetes-netwerkdiensten faciliteert.
ii. Het behandelt alle netwerkcommunicatie buiten en binnen het cluster en stuurt verkeer door of antwoordt op de pakketfilterlaag van het besturingssysteem.

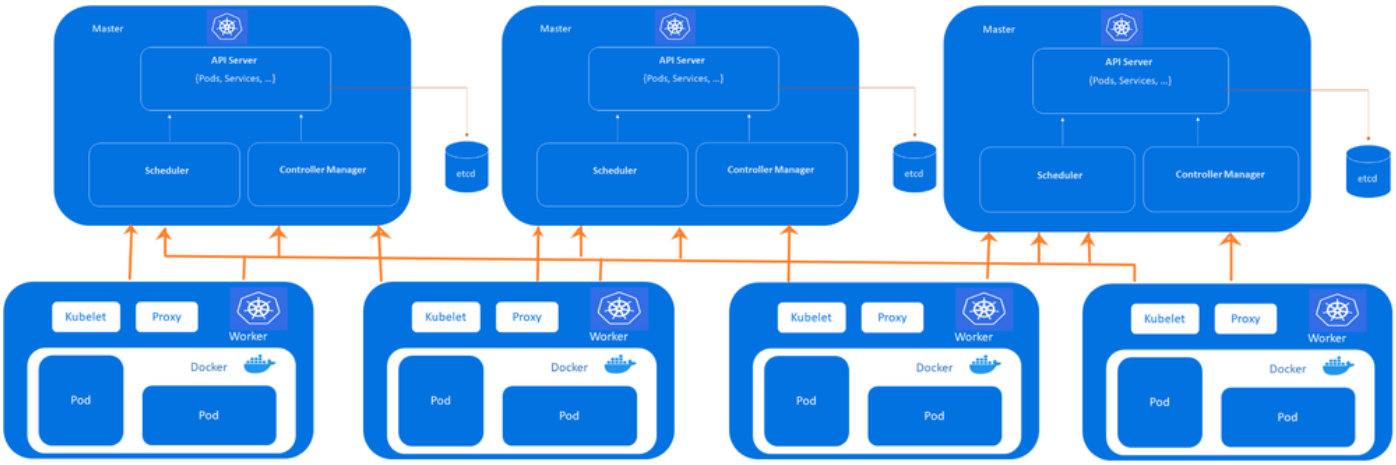
Peul:
i. Een pod dient als een enkele applicatie-instantie en wordt beschouwd als de kleinste eenheid in het objectmodel van Kubernetes.

Bastion Host:
i. De computer host over het algemeen één toepassing of proces, bijvoorbeeld een proxyserver of taakverdeling, en alle andere services worden verwijderd of beperkt om de dreiging voor de computer te verminderen.

Kubernetes Multi-Master Architecture

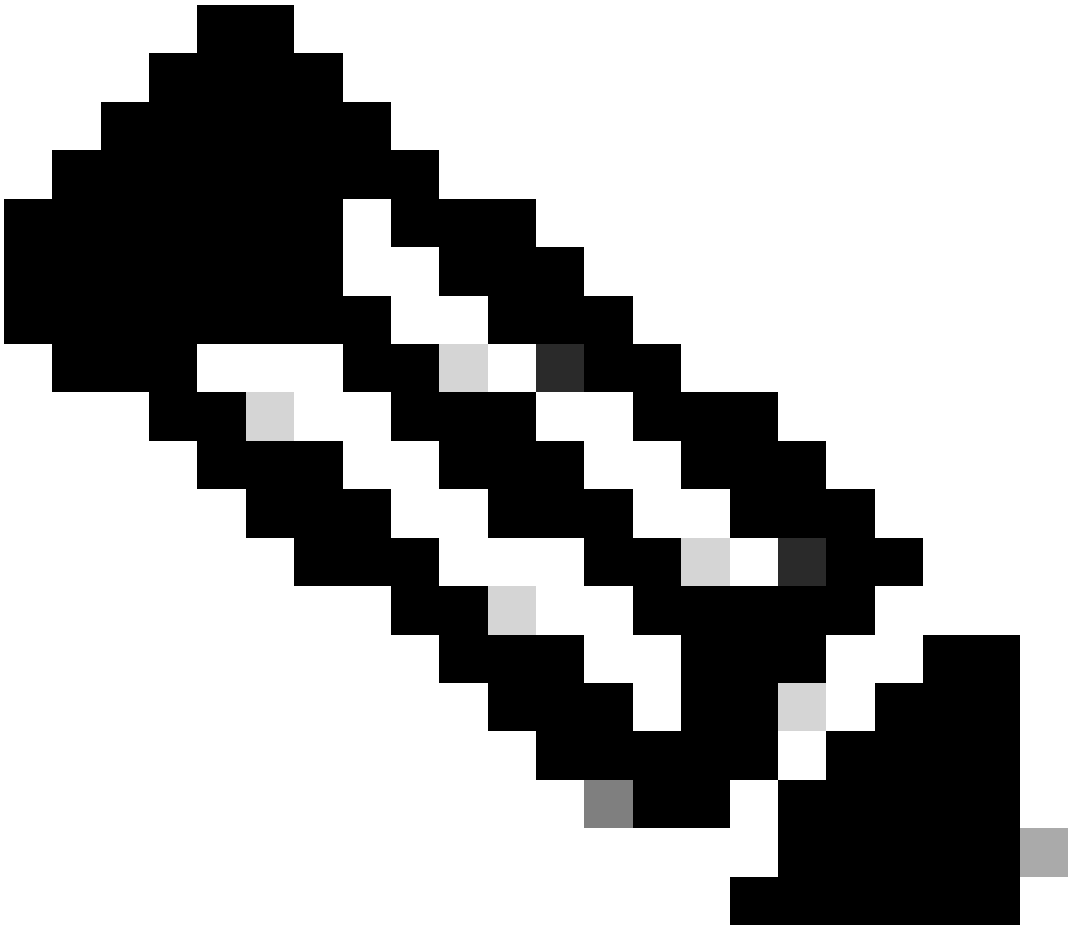
Cluster:

- 8 - Totaal aantal knooppunten
- 3 - Hoofdknooppunten
- 4 - werkknooppunten
- 1 - Bastion-server (taakverdeling)



Kubernetes-cluster met hoge beschikbaarheid en drie besturingsvlakken

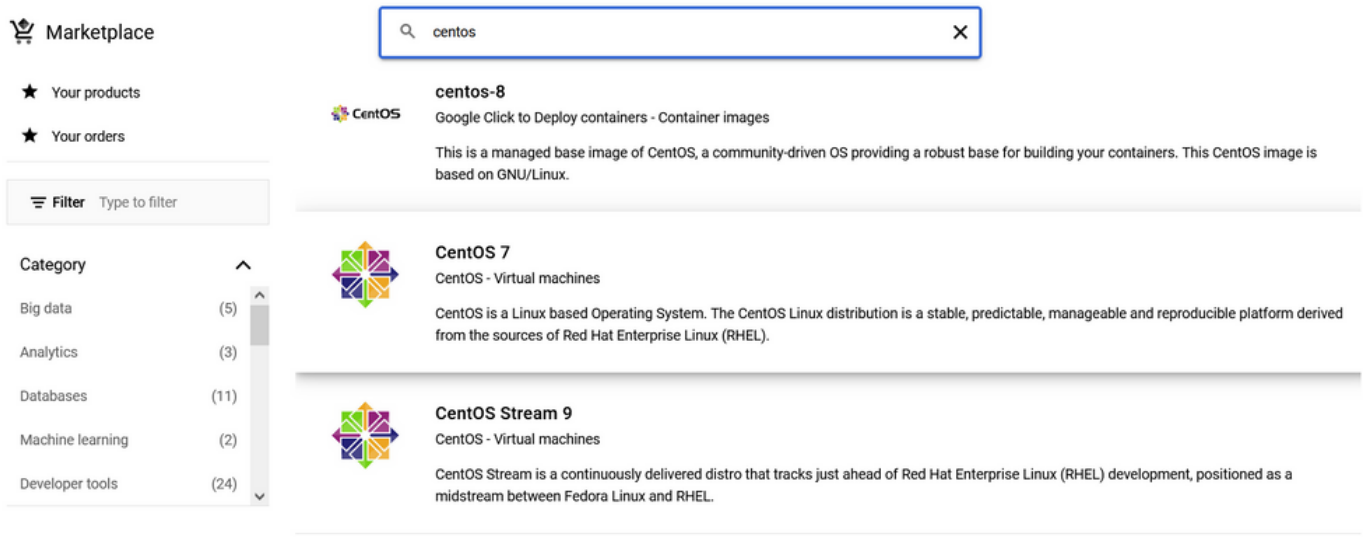
---



Opmerking: VPC configureren op GCP voordat de VM's worden geleverd. Referentie VPC op GCP.

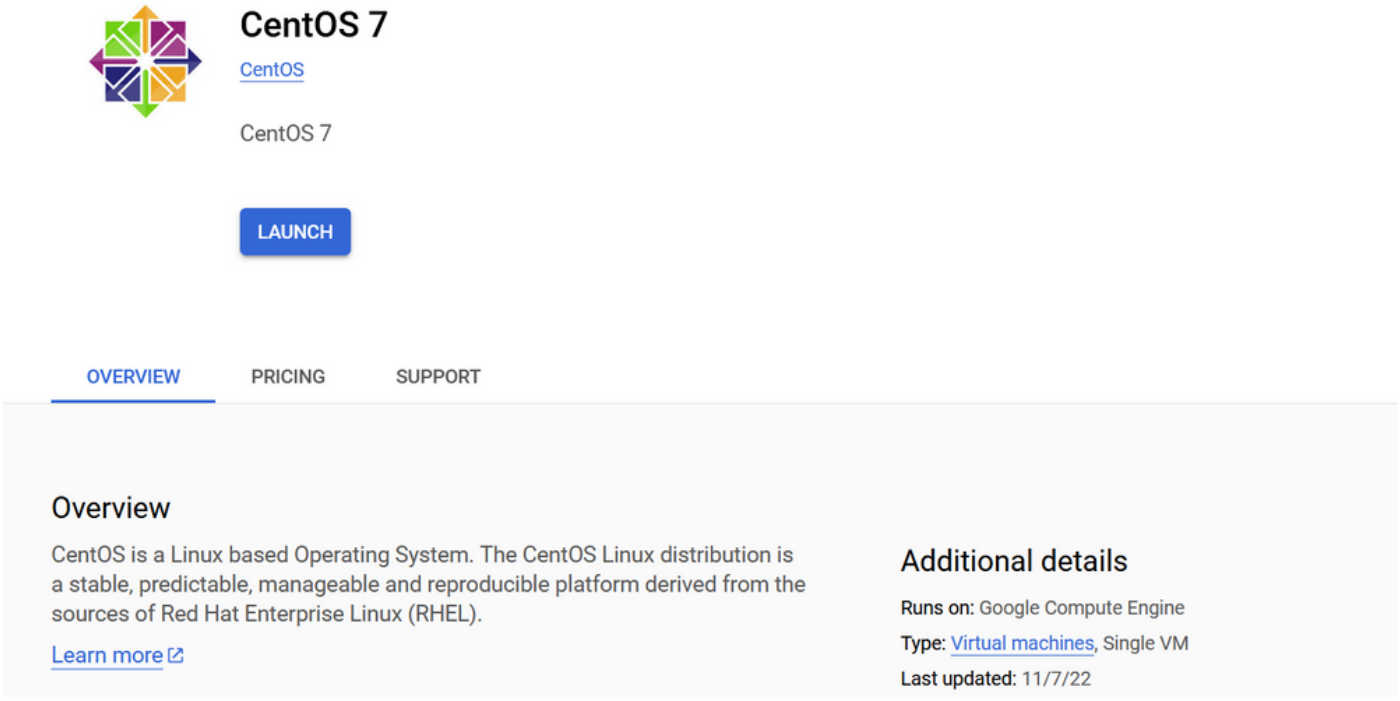
Beschikbaarstelling van virtuele machines op GCP

Op GCP-voorziening een Centos7 van de GCP-marktplaats.



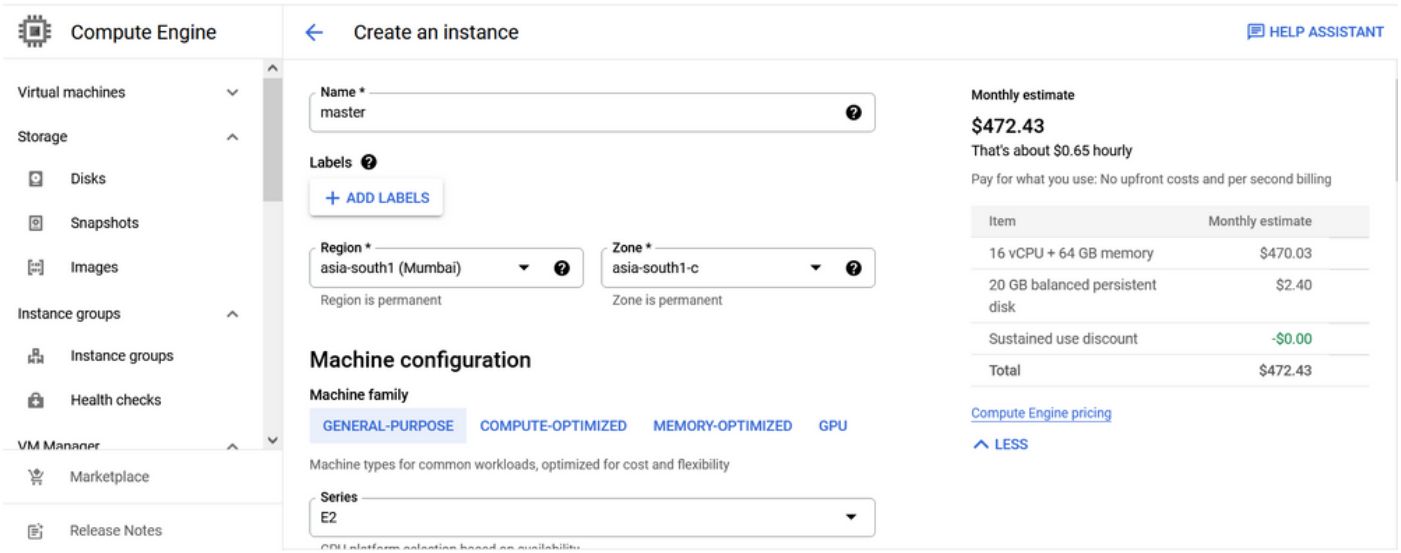
Centos marktplaats op GCP

Klik op de knop **.Launch**



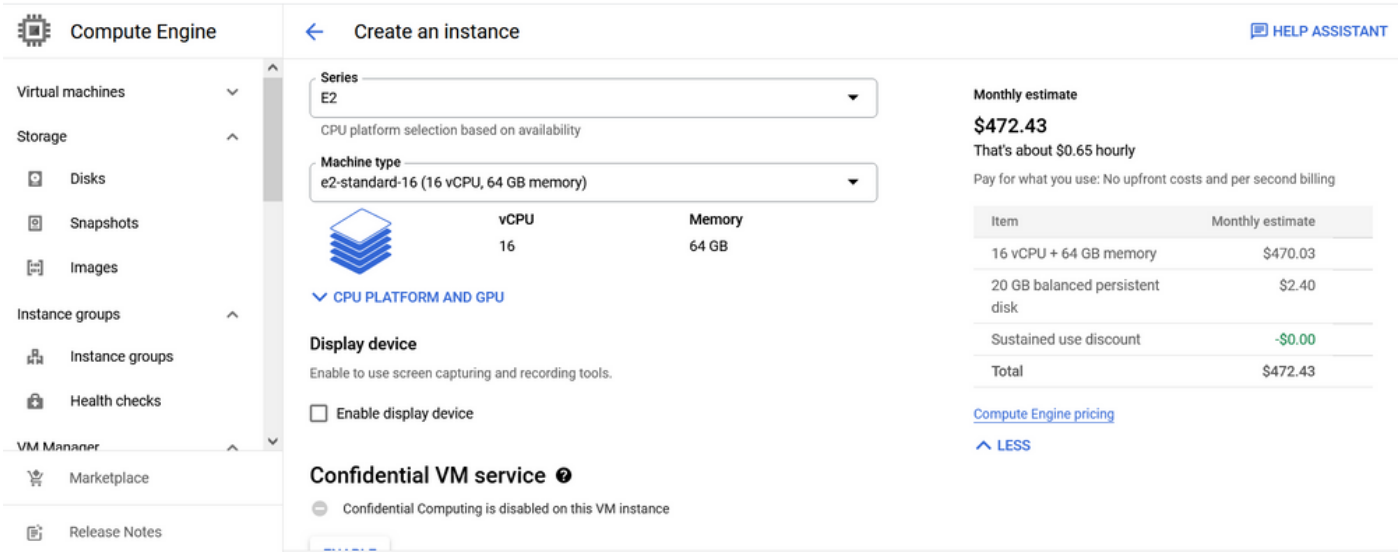
Centos 7 virtuele machine

Kies de regio volgens de dichtstbijzijnde bereikbaarheid. In dit lab is de regio geconfigureerd als Mumbai.



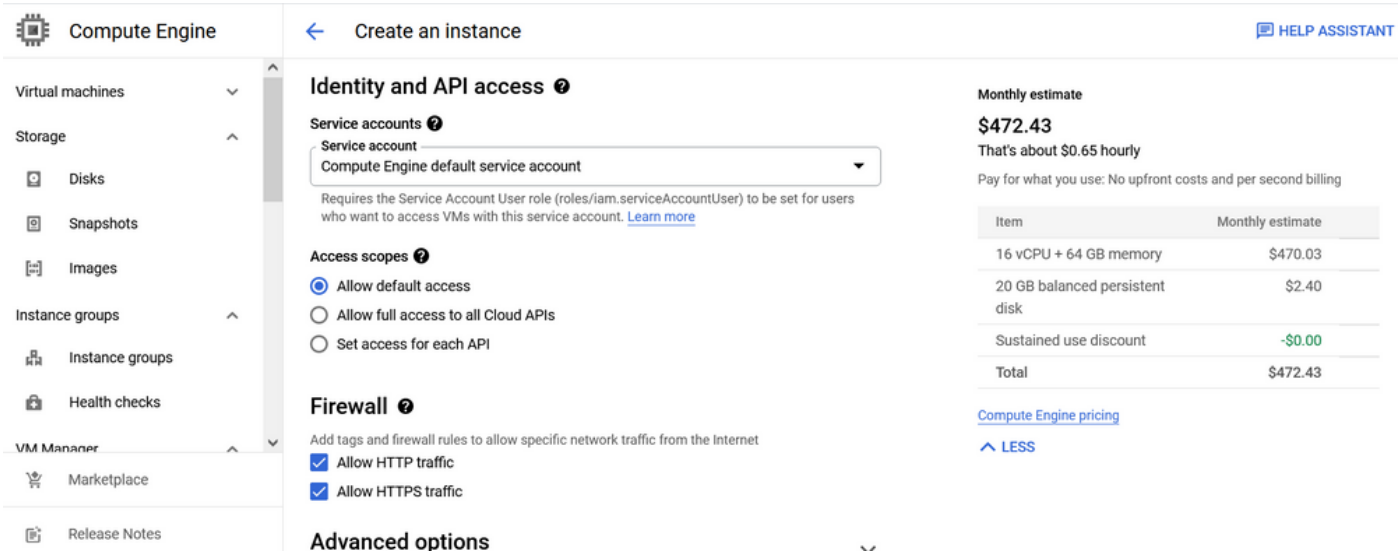
Configuratie van de Centos7-rekenengine

De machineconfiguratie is de E2-serie voor algemene doeleinden en het **e2-standard-16 (16 vCPU, 64 GB memory)**machinetype.



Centos 7-bronconfiguratie

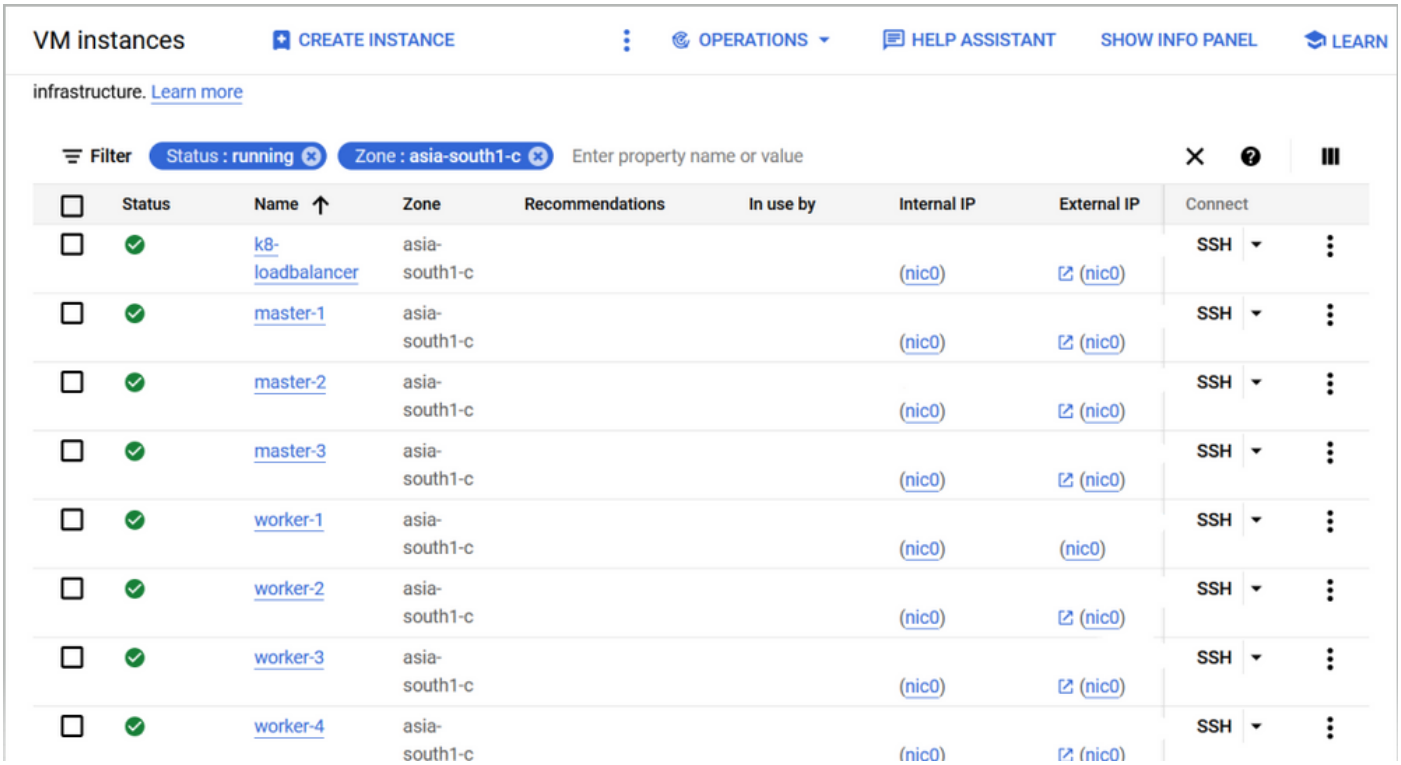
Selecteer **Allow default access** en voor firewall, **Allow HTTP traffic** en **Allow HTTPS traffic**.



Centos 7-netwerkconfiguratie

Klik op de knop .**Create**

Maak ook 8 knooppunten zoals hier wordt weergegeven.



Multi-master implementatie op Google Cloud Platform

Privé-IP's:

Op GCP worden de private en publieke IP's automatisch toegewezen.

```
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
```

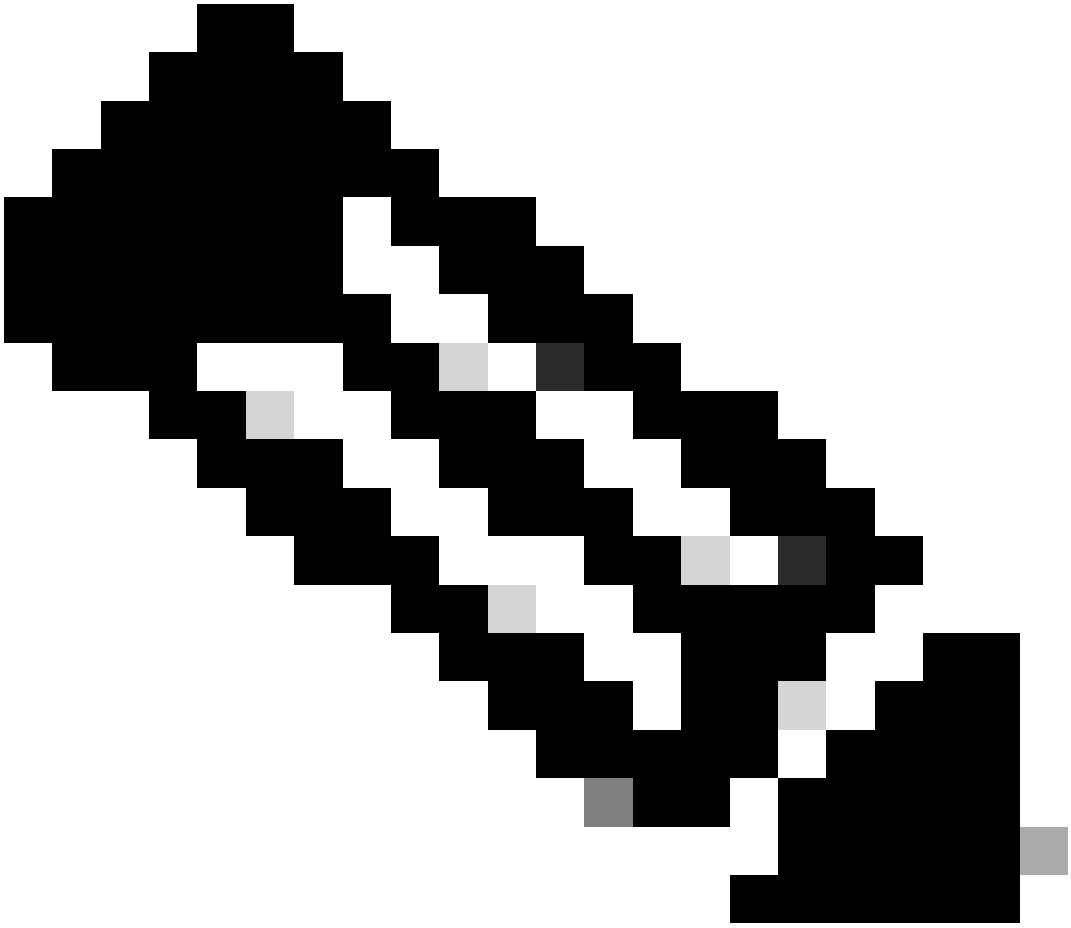
Overzicht op hoog niveau

Op alle knooppunten (meester, werker, bastion):

1. Open de vereiste firewallpoorten op de Linux-server en configureer de beveiligingsconfiguraties.

Op master nodes in multi-master implementaties:

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



Opmerking: Controleer ook of de poorten op de GCP-firewall zijn toegestaan.

---

2. De vereiste netwerkinstellingen configureren (lokale DNS, hostnamen, NTP).

Bastion Server:
1. Stel een HA-proxy in.
2. Voeg front-end- en backend-serverconfiguratie toe aan **haproxy.conf** bestand.
3. Start de **haproxy** service opnieuw.

Gemeenschappelijke stappen voor master- en werkknooppunten:
1. Installeer en configureer de docker.
2. Kubernetes installeren en configureren.

Alleen op masternodes:
1. Initialiseer het nieuwe Kubernetes-cluster (**kubeadm init**KB).
2. Installeer **Calico** een netwerkplug-in (specifiek gebruikt voor de DNS-kernservice op de hoofdknooppunten).

3. Gebruik deze opdracht om de masternodes met een masternode te verbinden.

```
kubeadm join

        :6443 --token

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

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

4. Valideer de clustergegevens met **kubectl get nodes**opdracht.

Alleen op werkknooppunten:

1. Gebruik deze opdracht om de werkersknoop aan de hoofdnode te koppelen.

```
kubeadm join

        :6443 --token

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

2. Als u zich hebt aangemeld, valideert u de clustergegevens op masternodes met deze **kubectl get**

nodesopdracht.

Configuraties op laag niveau

Netwerkconfiguratie

1. Wijzig het hoofdwachtwoord indien onbekend met deze opdracht:

```
passwd
```

2. Wijzig de hostnamen indien nodig met deze opdracht.

```
hostnamectl set-hostname
```

3. Configureer lokale DNS.

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

```
4. Schakel chrony voor NTP-services in met deze opdracht.
```

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

```
2. Controleer de status met deze opdracht.
```

```
systemctl status chronyd
```

```
3. Controleer NTP-bronnen met deze opdracht.
```

```
chronyc sources -v
```

Bastion Server

Stap 1. Controleer updates.

```
sudo yum check-update
```

Stap 2. Installeer updates als deze niet up-to-date zijn.

```
yum update -y
```

Stap 3. Installeer yum-hulpprogramma's.

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

Stap 4. Installeer het haproxy-pakket.

```
yum install haproxy -y
```

Stap 5. Voeg deze configuratie toe onder standaardinstellingen 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 /
```

Stap 6. Controleer het configuratiebestand zodat het er als volgt **vi /etc/haproxy/haproxy.cfg**uitziet:

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

Stap 7. Controleer de status van 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]#
```

Mogelijke fouten

1. De HAProxy-service verkeert in een storingsstatus nadat u configuratiewijzigingen hebt **haproxy.cfg**doorgevoerd. Voorbeeld;

```
[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.
```

resolutie

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

Uitvoering:

```
[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]#
```

Docker installeren op hoofd- en werkknooppunten

Stap 1. Controleer updates.

```
sudo yum check-update
```

Stap 2. Updates installeren als deze niet up-to-date zijn.

```
yum update -y
```

Stap 3. Installeer yum-hulpprogramma's.

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

Stap 4. Docker installeren.

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

Stap 5. Docker inschakelen.

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

Stap 6. Dockerservice starten.

```
sudo systemctl start docker
```

Stap 7. Controleer de dockerstatus.

```
sudo systemctl status docker
```

Uitgang:

```
[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]#
```

Kubernetes installeren op hoofd- en werkknooppunten

Stap 1. Voeg Kubernetes repository toe.

```
cat <
```

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

Stap 2. Disable.**SELinux**

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

Stap 3. Installeren.**Kubernetes**

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

Stap 4. Inschakelen.**Kubelet**

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

Stap 5. Configureren.**Pod Network**

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

Stap 6. Tokengeneratie:

Uitvoer voor de master (control plane) en voor de nodes van de werknemer.

master node

Token: Uw Kubernetes control-plane is succesvol geïnitialiseerd!

Om uw cluster te gebruiken, voert u dit als een gewone gebruiker uit:

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

Als alternatief, als u de rootgebruiker bent, kunt u uitvoeren.

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

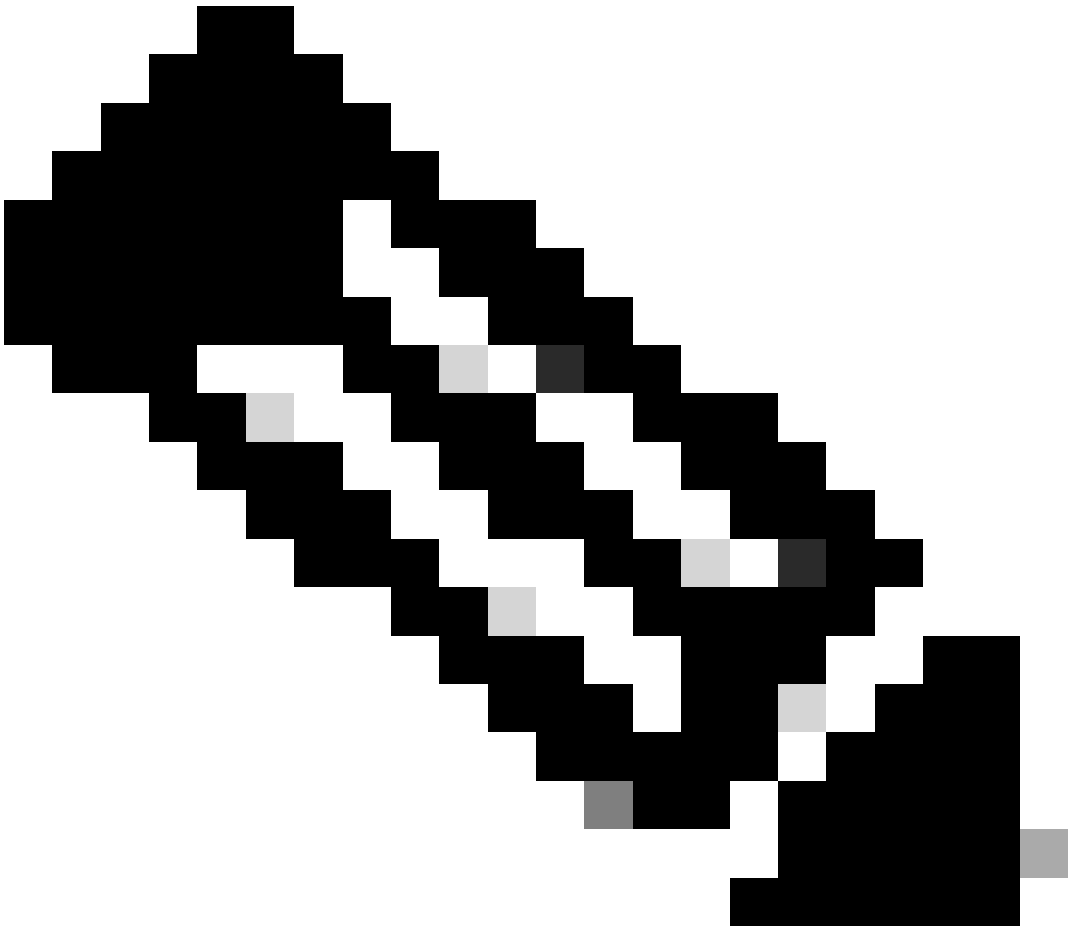
U kunt nu een pod-netwerk implementeren in het cluster.

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

U kunt nu elk nummer van de knooppunten van het regelvlak of de masternodes die deze opdracht op elk als root uitvoeren, toevoegen.

Refereren: [Kubeadm join workflow]

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



Opmerking: Houd er rekening mee dat de certificaatsleutel toegang geeft tot clustergevoelige gegevens, houd deze geheim!

Als voorzorgsmaatregel worden geüploade certs binnen twee uur verwijderd; indien nodig kunt u ze gebruiken.

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

Vervolgens kunt u zich bij een willekeurig aantal werkknooppunten aansluiten en dit op elk als root uitvoeren.

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

Stap 7. Controleer de DNS-kernservice.

```
[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]#
```

Stap 8. Controleer de status van de masternode.

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

Als u meerdere masternodes wilt samenvoegen, wordt deze opdracht join uitgevoerd op masternodes.

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

Mogelijke fouten die zijn opgetreden op het moment van tokengeneratie

Fout 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
```

Fout CRI-resolutie

Stap 1. Verwijder het bestand **config.toml** en start de **container opnieuw op**.

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

Stap 2. Containers installeren:
Installeer het containerd.io-pakket vanuit de officiële Docker-repositories met deze opdrachten.

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

Stap 3. Containers configureren:

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

Stap 4. Containers opnieuw opstarten:

```
systemctl restart containerd
```

Fout 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
```

Fout FileContent--proc-sys-net-ipv4-ip_forward Resolutie

Stel de waarde ip_forward in op 1.

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

Core DNS-service wordt niet uitgevoerd

```
[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]#
```

resolutie

De DNS-kern is in behandeling, wat wijst op een probleem met netwerken. Daarom moet Calico worden geïnstalleerd.

Referentie: Calico

Voer deze twee opdrachten uit:

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

werkknooppunt

On Worker Node wanneer token wordt verkregen van master:

Stap 1. Kubelet-service inschakelen.

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

Stap 2. Sluit u aan bij alle werkknooppunten met de hoofdknooppunt met deze opdracht join.

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

Stap 3. Als er fouten met betrekking tot bestanden of poorten worden aangetroffen, reset u de kubeadm met deze opdracht.

```
kubeadm reset
```

Nadat u de token opnieuw hebt ingesteld, voert u deze in vanaf de hoofdnode.

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

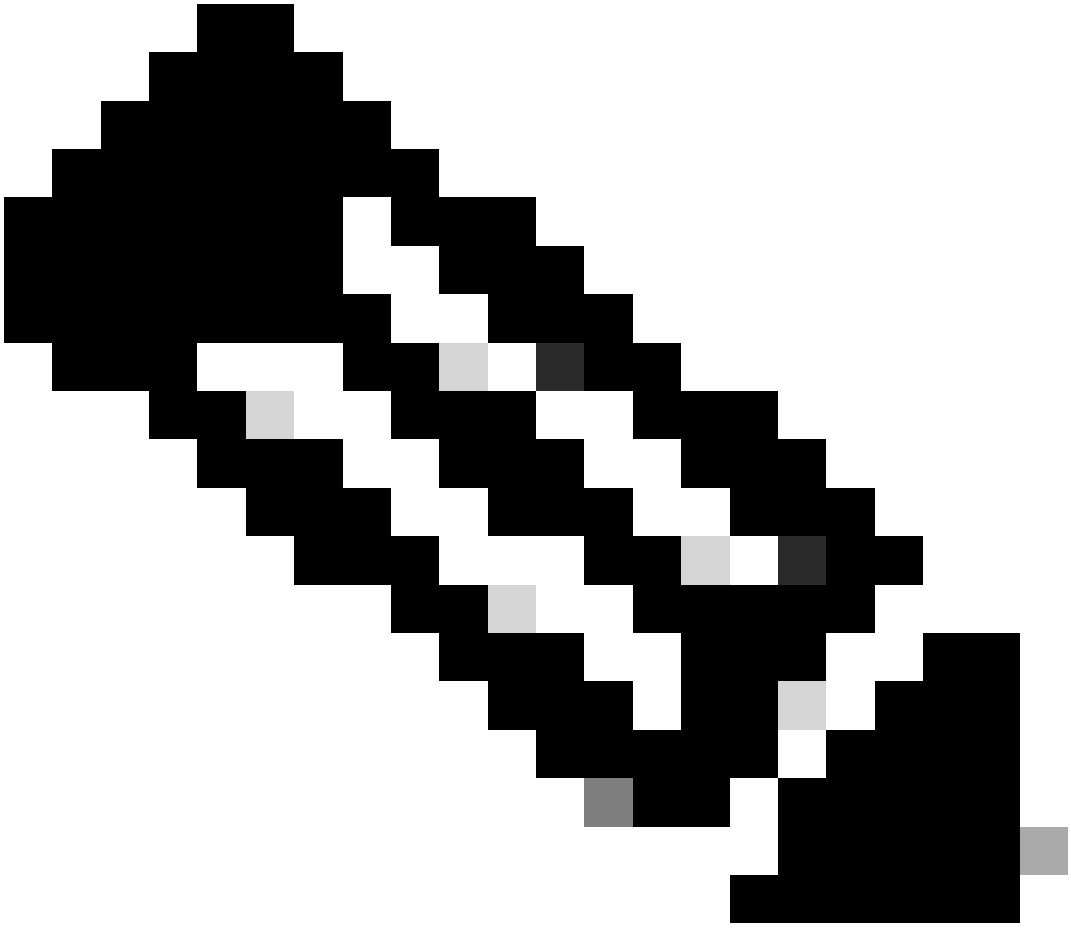
```
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

uiteindelijke productie

Het cluster is nu gevormd, controleer het met de opdracht **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]#
```

Kubernetes-clusteruitvoer met hoge beschikbaarheid



Opmerking: op het moment van clustervorming wordt alleen besturing vanuit masternodes geconfigureerd. De bastion-host is niet geconfigureerd als een gecentraliseerde server om alle Kube's in het cluster te bewaken.

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.