Creazione di un cluster multi-master Kubernetes con Centos 7 sulla piattaforma Google Cloud

Sommario

Introduzione

Questo documento descrive l'implementazione del cluster Kubernetes con 3 nodi master e 4 nodi di lavoro con un host bastion che funge da bilanciatore del carico su Google Cloud Platform (GCP).

Prerequisiti

Requisiti

Cisco raccomanda la conoscenza dei seguenti argomenti:

- Linux
- Docker e Kubernetes
- Piattaforma Google Cloud

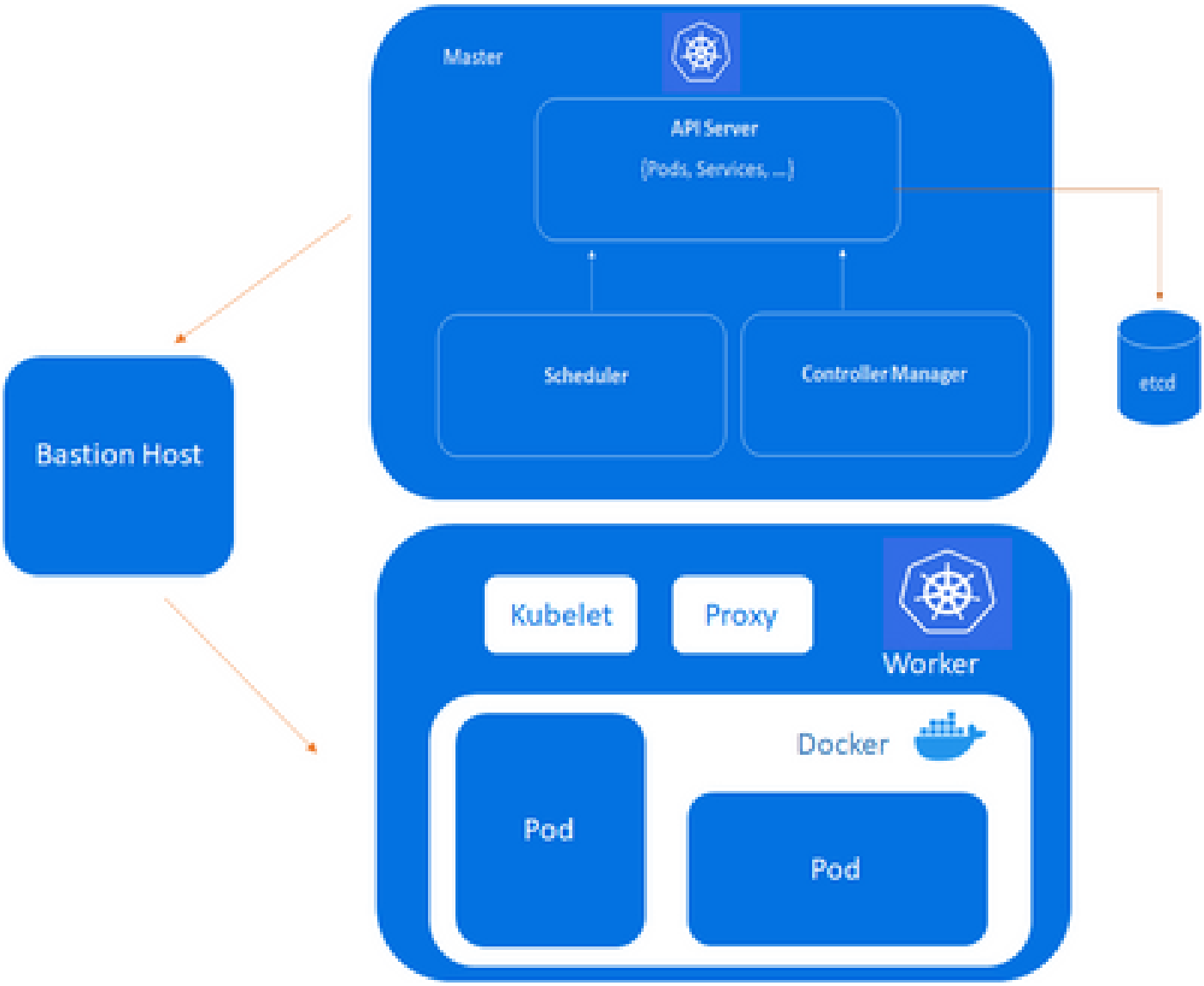
Componenti usati

Le informazioni fornite in questo documento si basano su:

- Sistema operativo: macchina virtuale Centos 7
- Famiglia di macchine (e2-standard-16):
  - vCPU - 16 vCPU
  - RAM - 64 GB

Le informazioni discusse in questo documento fanno riferimento a dispositivi usati in uno specifico ambiente di emulazione. Su tutti i dispositivi menzionati nel documento la configurazione è stata ripristinata ai valori predefiniti. Se la rete è operativa, valutare attentamente eventuali conseguenze derivanti dall'uso dei comandi.

Esempio di rete



Preside del bastione, Kube-Master, Kube-node

Componenti nodo principale Kubernetes

Kube-apiserver:
i. Fornisce un'API che funge da front-end di un control plane Kubernetes.
ii. Gestisce le richieste esterne e interne che determinano se una richiesta è valida e quindi la elabora.
iii. È possibile accedere all'API tramite l'interfaccia della riga di **kubectl** comando o altri strumenti come **kubeadm**e tramite chiamate REST.

Utilità di pianificazione Kube:
i. Questo componente pianifica pod su nodi specifici in base a workflow automatizzati e condizioni definite dall'utente.

Kube-controller-manager:
i. Il controller manager Kubernetes è un loop di controllo che controlla e regola lo stato di un cluster Kubernetes.
ii. Riceve informazioni sullo stato corrente del cluster e degli oggetti al suo interno e invia istruzioni

per spostare il cluster verso lo stato desiderato dall'operatore del cluster.

etcd:
i. Database key-value contenente i dati relativi allo stato e alla configurazione del cluster.
ii. Etcd è a tolleranza d'errore e distribuito.

Componenti del nodo di lavoro Kubernetes

Kubelet
i. Ogni nodo contiene un **kubelet**, che è una piccola applicazione in grado di comunicare con il control plane Kubernetes.
ii. garantisce **kubelet** che i contenitori specificati nella configurazione pod vengano eseguiti su un nodo specifico e ne gestiscono il ciclo di vita.
iii. Esegue le azioni comandate dal piano di controllo.

Kube-proxy:
i. Tutti i nodi di elaborazione contengono **kube-proxy**, un proxy di rete che facilita i servizi di rete Kubernetes.
ii. Gestisce tutte le comunicazioni di rete all'esterno e all'interno del cluster e inoltra il traffico o le risposte sul livello di filtro dei pacchetti del sistema operativo.

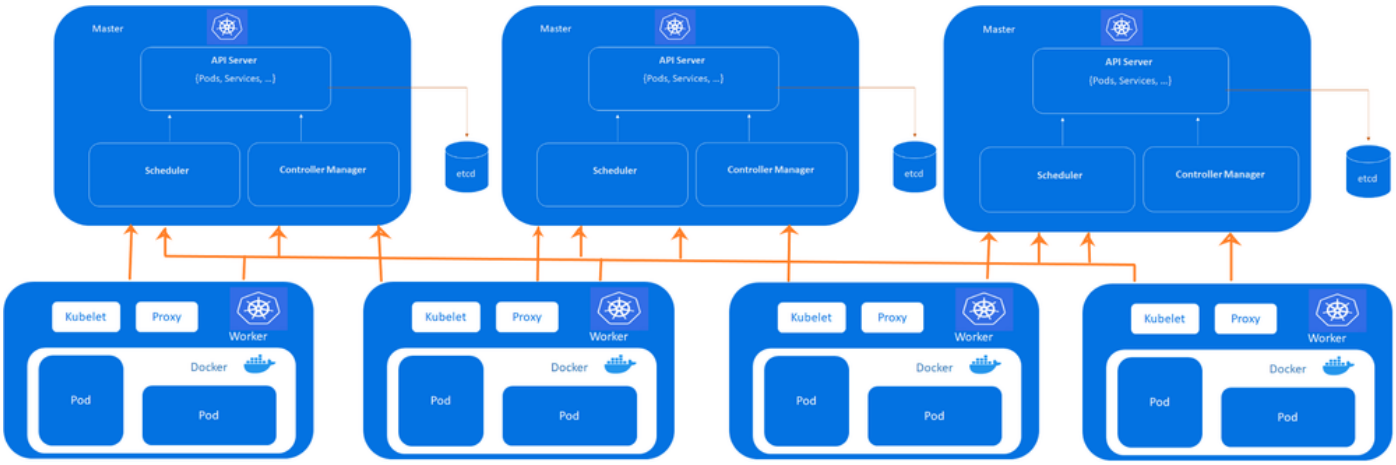
Baccelli:
i. Un baccello serve come singola istanza di applicazione ed è considerato l'unità più piccola nel modello a oggetti di Kubernetes.

Ospitante del bastione:
i. Il computer in genere ospita una singola applicazione o un singolo processo, ad esempio un server proxy o un servizio di bilanciamento del carico, mentre tutti gli altri servizi vengono rimossi o limitati per ridurre la minaccia per il computer.

Architettura multi-master Kubernetes

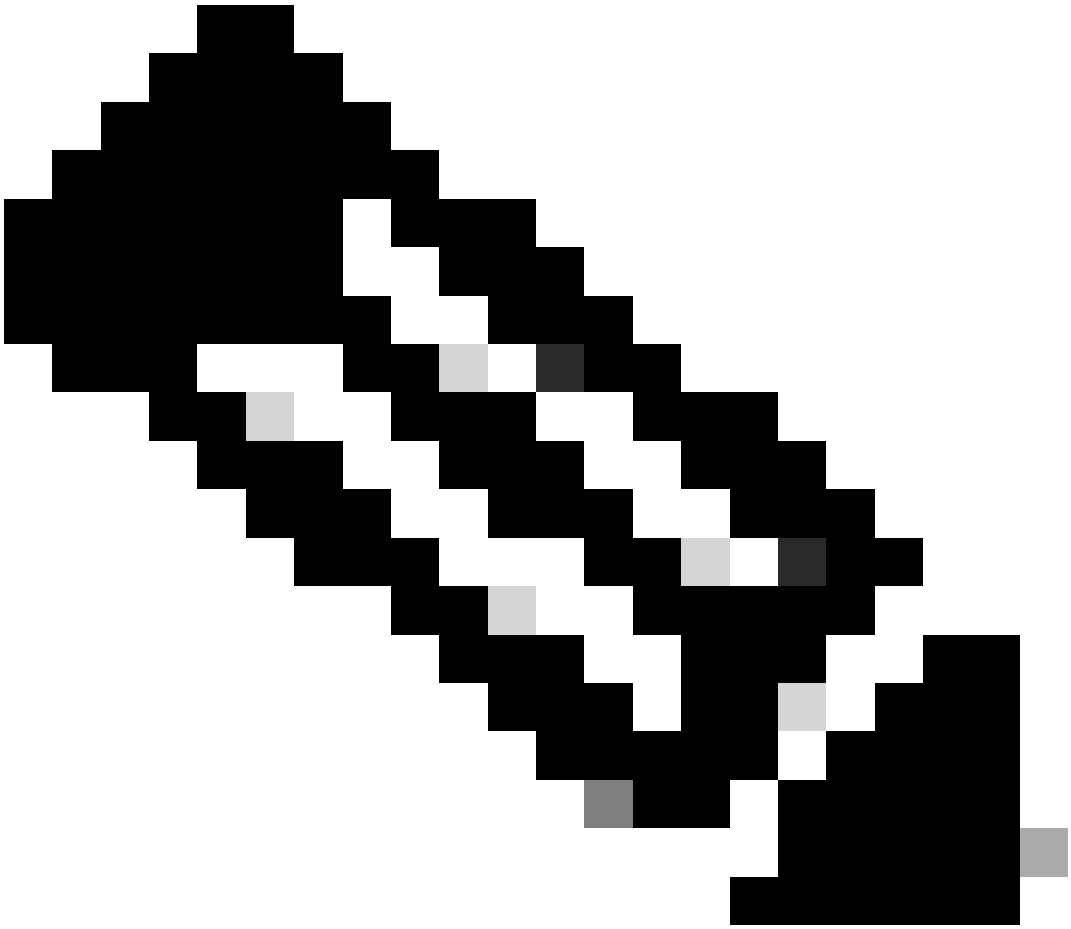
Cluster:

- 8 - Nodi totali
- 3 - Nodi principali
- 4 - Nodi di lavoro
- 1 - Server Bastion (servizio di bilanciamento del carico)



Cluster Kubernetes ad alta disponibilità con tre control-plane

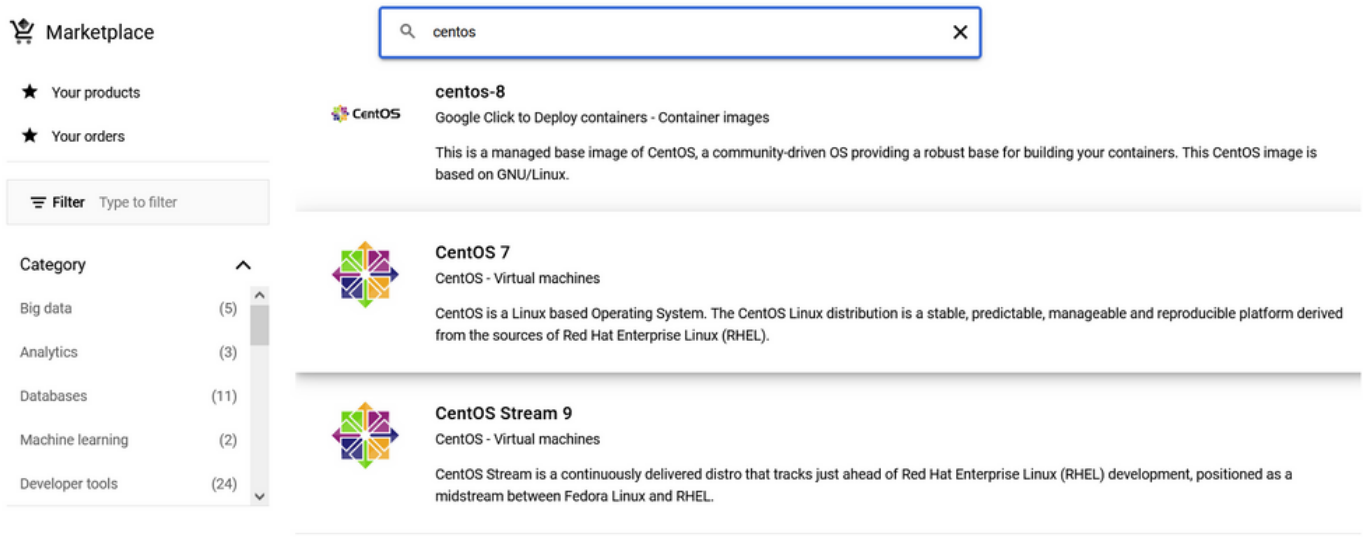
---



Nota: Configurare VPC su GCP prima di eseguire il provisioning delle VM. Fare riferimento [a VPC su GCP](#).

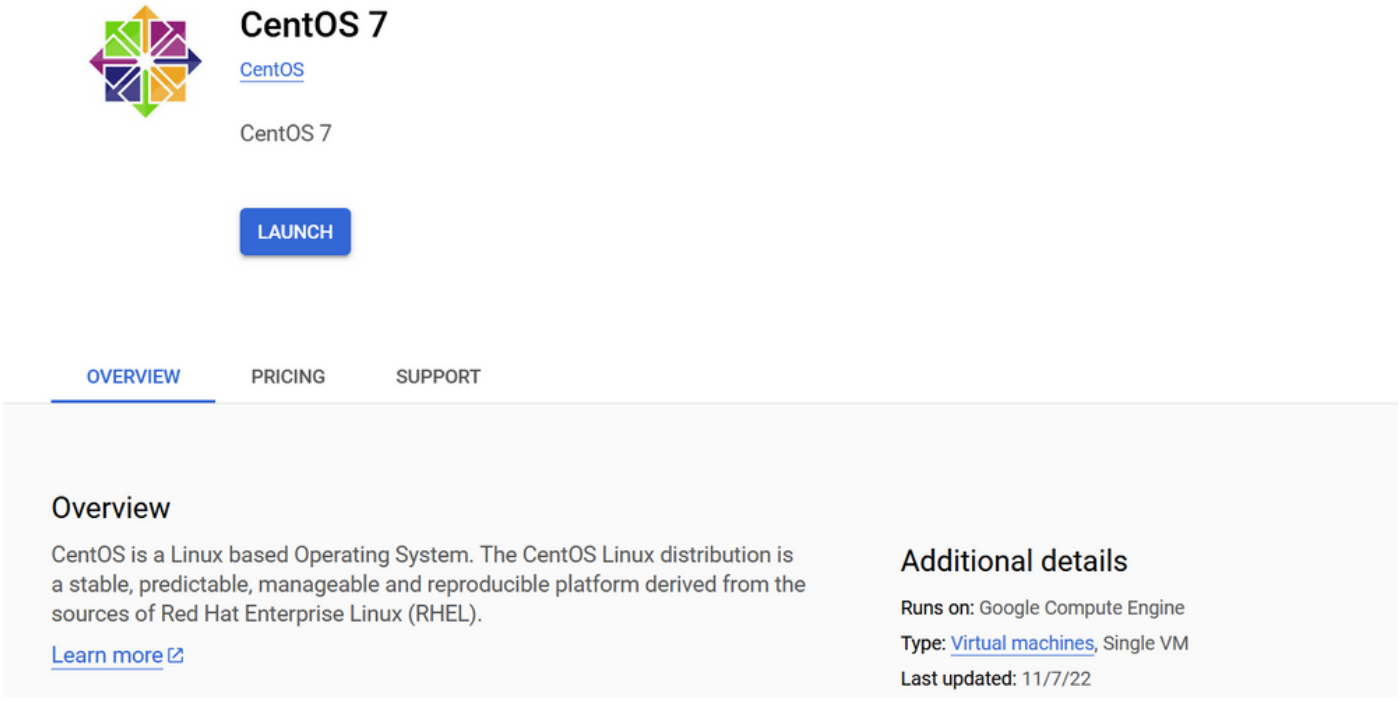
Fornitura di macchine virtuali su GCP

Per quanto riguarda la fornitura di GCP, un Centos7 del mercato GCP.



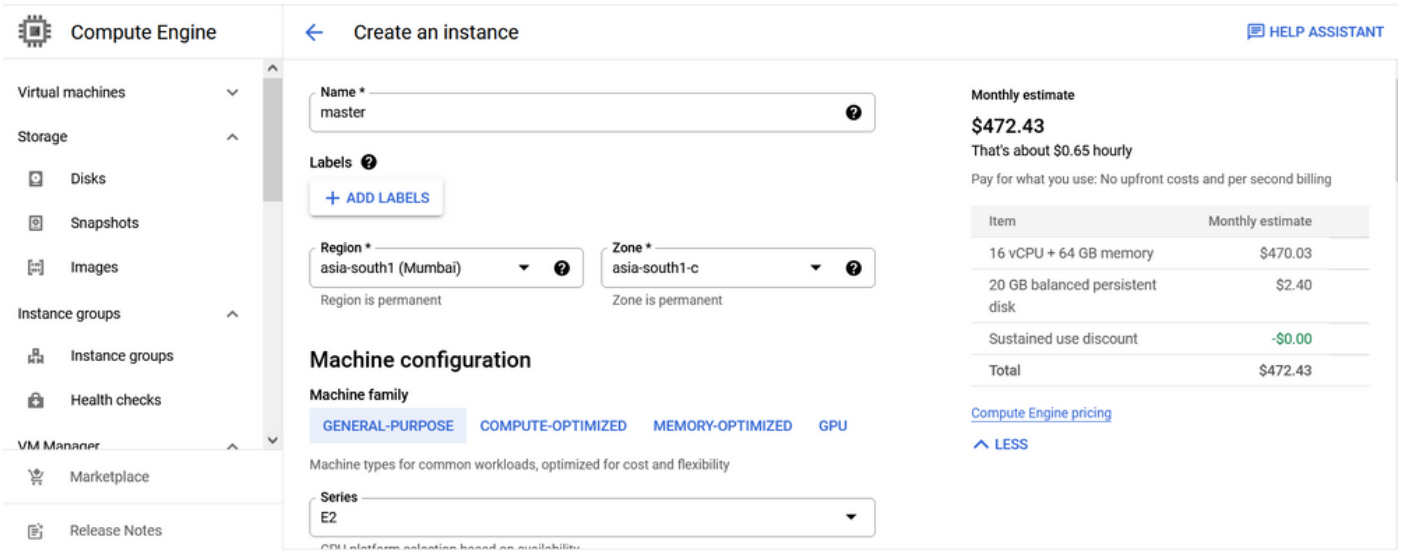
Centos marketplace su GCP

Fare clic su **.Launch**



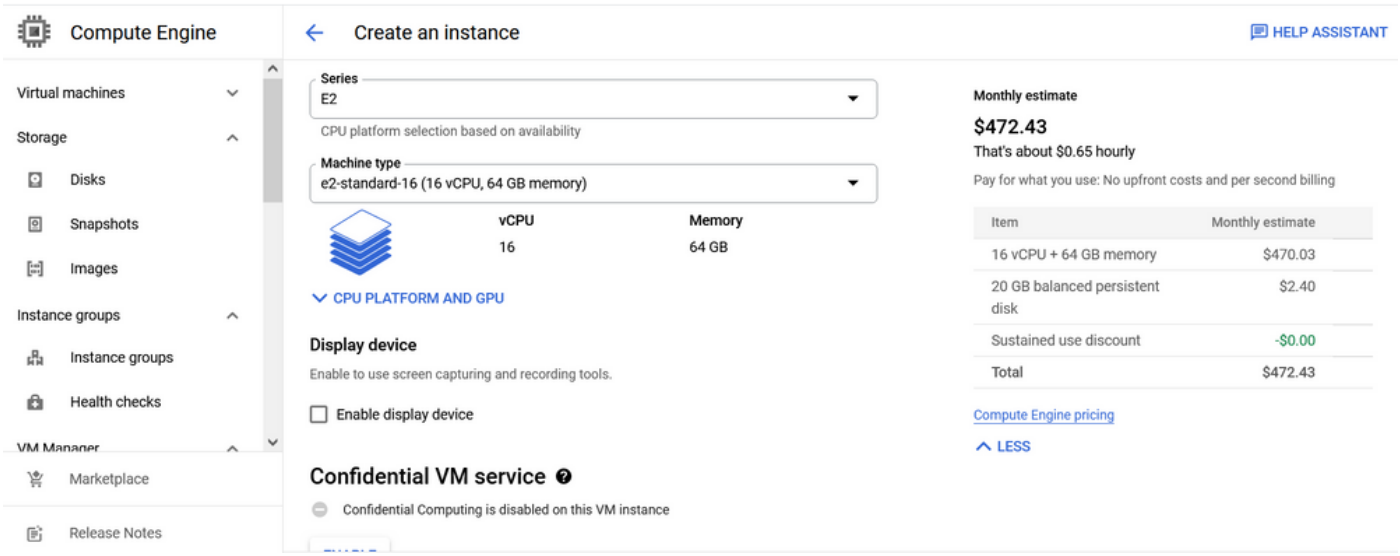
Macchina virtuale Centos 7

Scegliere la regione in base alla raggiungibilità più vicina. In questo laboratorio, la regione è configurata come Mumbai.



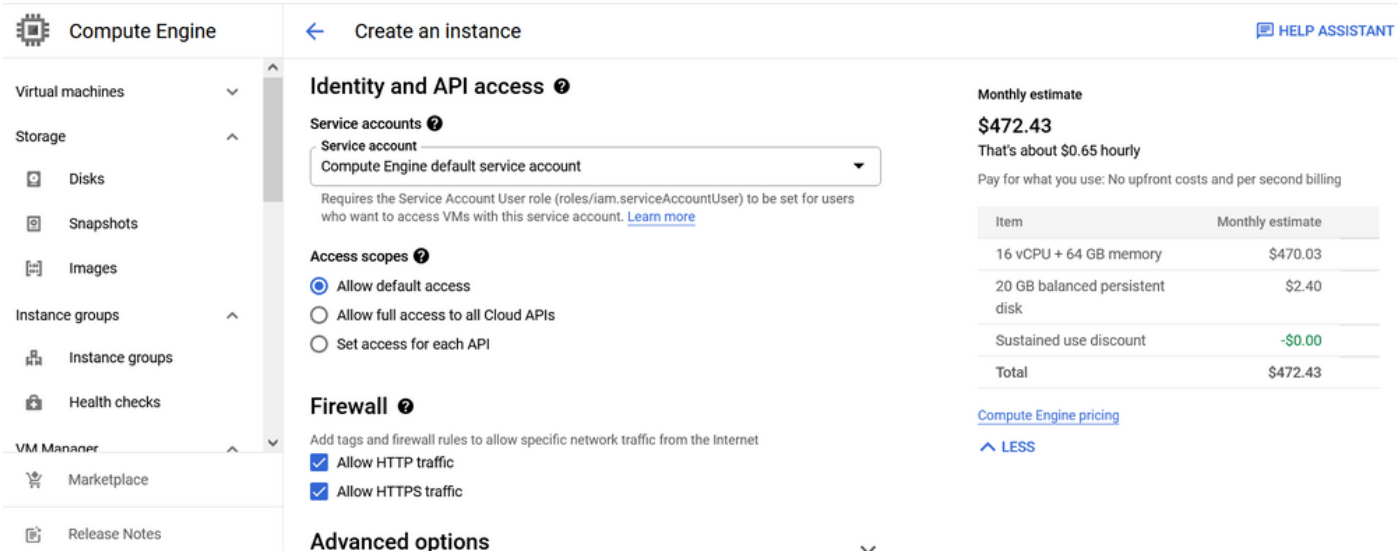
Configurazione del motore di elaborazione Centos7

La configurazione della macchina è di tipo generico e **e2-standard-16 (16 vCPU, 64 GB memory)**tipo E2.



Configurazione risorse Centos 7

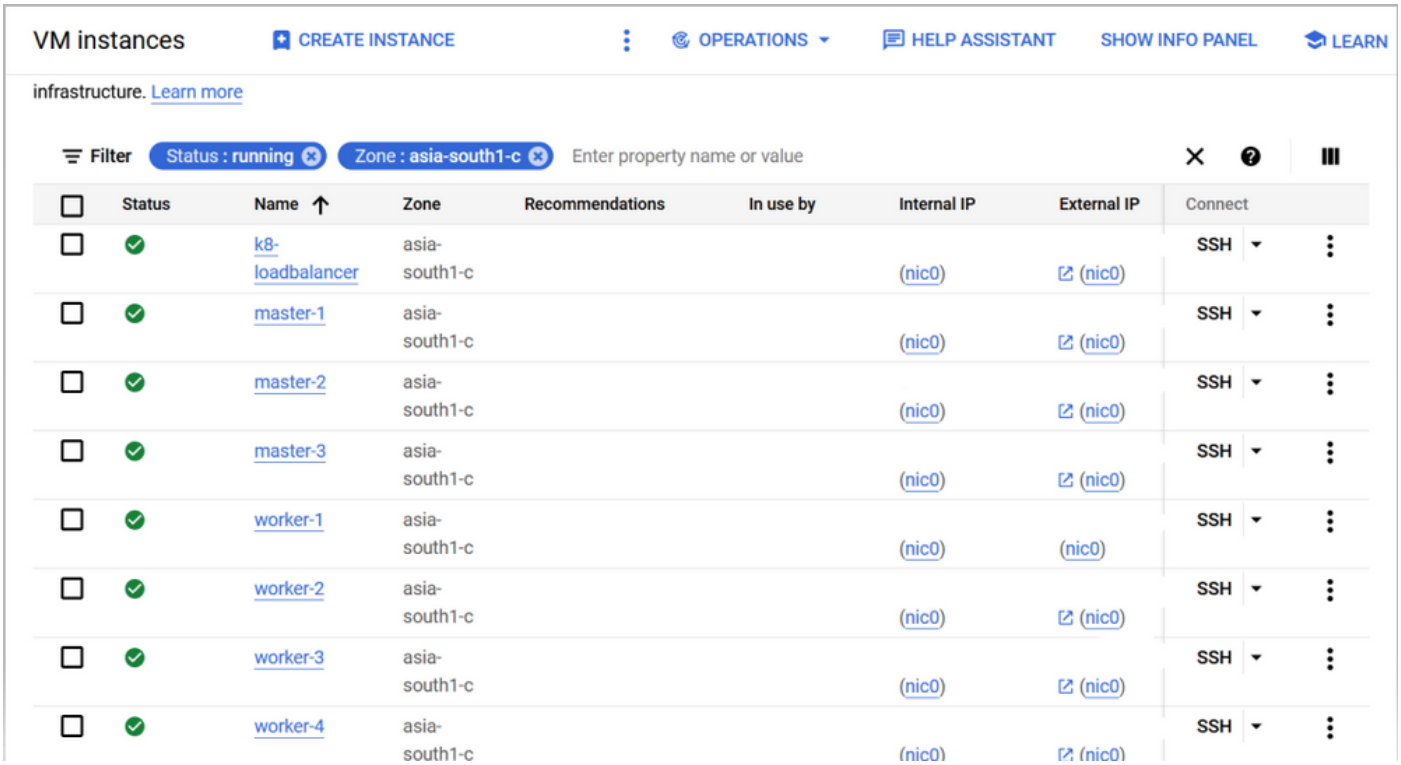
Selezionare **Allow default access** e per il firewall, **Allow HTTP traffic** e **Allow HTTPS traffic.**



Configurazione di rete Centos 7

Fare clic su .**Create**

Analogamente, create 8 nodi come mostrato di seguito.



Distribuzione multi-master su piattaforma cloud Google

IP privati:

Su GCP gli IP pubblico e privato vengono assegnati 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
```

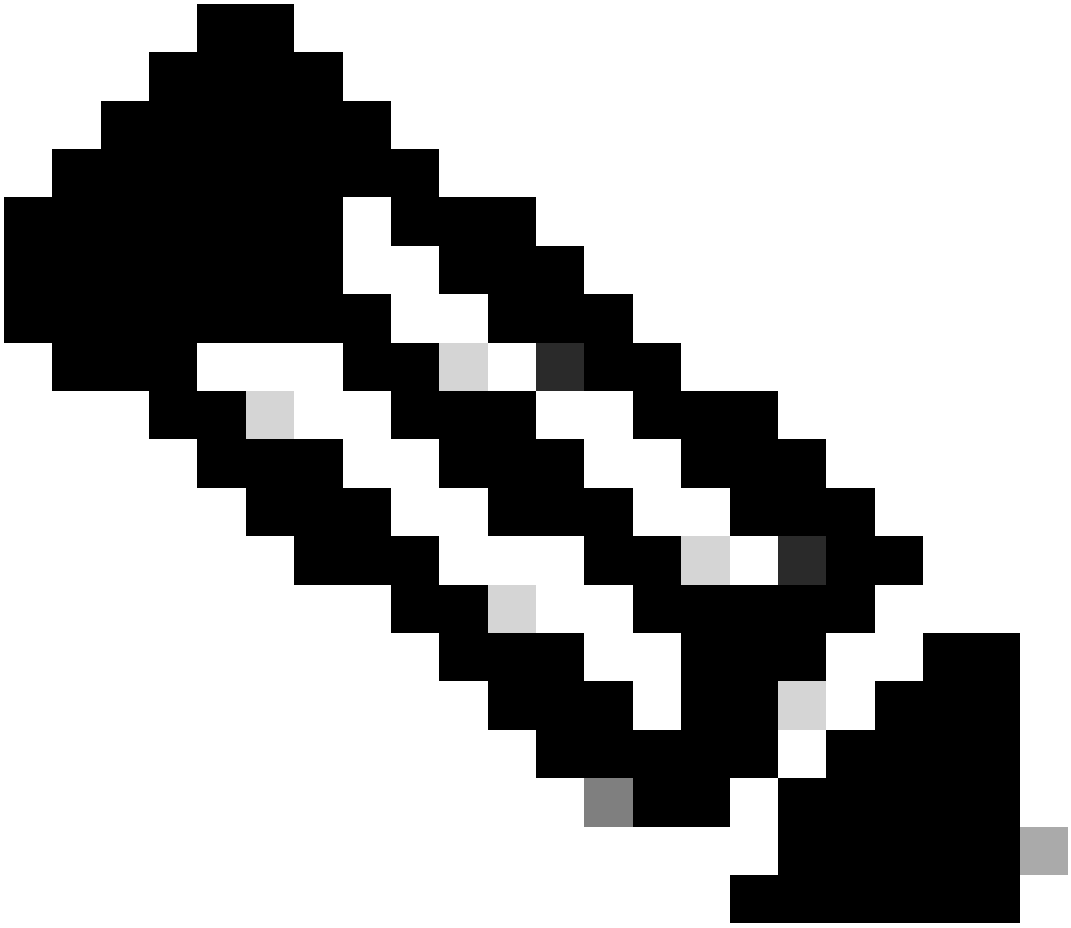
Panoramica di alto livello

Su tutti i nodi (master, lavoratore, bastione):

1. Aprire le porte del firewall richieste sul server Linux e configurare le configurazioni di sicurezza.

Sui nodi master nelle distribuzioni con più master:

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



Nota: Verificare inoltre che le porte sul firewall GCP siano consentite.

2. Configurare le impostazioni di rete necessarie (DNS locale, nomi host, NTP).

Bastion Server:
1. Impostare un proxy HA.
2. Aggiungere la configurazione del server front-end e back-end nel **haproxy.conf** file.
3. Riavviare il **haproxy** servizio.

Passaggi comuni per i nodi master e di lavoro:
1. Installare e configurare il docker.
2. Installare e configurare Kubernetes.

Solo nei nodi principali:
1. Inizializzare il nuovo cluster Kubernetes (**kubeadm init**).
2. Installare **Calico** il plug-in di rete (utilizzato in modo specifico per il servizio DNS di base sui nodi master).

3. Utilizzare questo comando per unire i nodi principali con un nodo principale.

```
kubeadm join

        :6443 --token

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

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

4. Convalidare le informazioni sul cluster con il **kubectl get nodes**comando.

Solo nei nodi di lavoro:

1. Utilizzare questo comando per unire il nodo di lavoro al nodo principale.

```
kubeadm join

        :6443 --token

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

2. Una volta completata l'aggiunta, convalidare le informazioni sul cluster sui nodi master con

questo comando **kubectl get nodes.**

Configurazioni di basso livello

Configurazione della rete

1. Modificare la password di root se sconosciuta con questo comando:

```
passwd
```

2. Se necessario, modificare i nomi host con questo comando.

```
hostnamectl set-hostname
```

3. Configurare il DNS locale.

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

```
4. Abilitare la sincronizzazione per i servizi NTP con questo comando.
```

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

```
2. Verificare lo stato con questo comando.
```

```
systemctl status chronyd
```

```
3. Controllare le origini NTP con questo comando.
```

```
chronyc sources -v
```

Bastion Server

Passaggio 1. Controllare gli aggiornamenti.

```
sudo yum check-update
```

Passaggio 2. Installare gli aggiornamenti se non sono aggiornati.

```
yum update -y
```

Passaggio 3. Installare le utilità yum.

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

Passaggio 4. Installare il pacchetto haproxy.

```
yum install haproxy -y
```

Passaggio 5. Aggiungere questa configurazione ai valori predefiniti 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 /
```

Passaggio 6. Verificare il file di configurazione in modo che appaia come questo 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 /
```

Passaggio 7. Verificare lo stato di 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]#
```

Errori possibili

1. Il servizio HAProxy si trova in stato di errore dopo aver apportato modifiche alla configurazione in **haproxy.cfg**. Ad esempio;

```
[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.
```

Risoluzione

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

Esecuzione:

```
[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]#
```

Installa Docker nei nodi master e di lavoro

Passaggio 1. Controllare gli aggiornamenti.

```
sudo yum check-update
```

Passaggio 2. Installare gli aggiornamenti se non sono aggiornati.

```
yum update -y
```

Passaggio 3. Installare le utilità yum.

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

Passaggio 4. Installare Docker.

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

Passaggio 5. Abilitare il docker.

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

Passaggio 6. Avviare il servizio docker.

```
sudo systemctl start docker
```

Passaggio 7. Controllare lo stato dell'alloggiamento di espansione.

```
sudo systemctl status docker
```

Uscita:

```
[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]#
```

Installazione di Kubernetes sui nodi master e di lavoro

Passaggio 1. Aggiungere il repository Kubernetes.

```
cat <
```

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

Passaggio 2. Disabilitare **SELinux**.

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

Passaggio 3. Installare **Kubernetes**.

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

Passaggio 4. Abilitare **Kubelet**.

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

Passaggio 5. Configurare **Pod Network**.

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

Passaggio 6. Generazione token:

Output per il master (control plane) e per i nodi di lavoro.

Nodo principale

Token: Inizializzazione del control plane Kubernetes completata.

Per utilizzare il cluster, eseguire questo comando come utente normale:

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

In alternativa, l'utente root può eseguire.

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

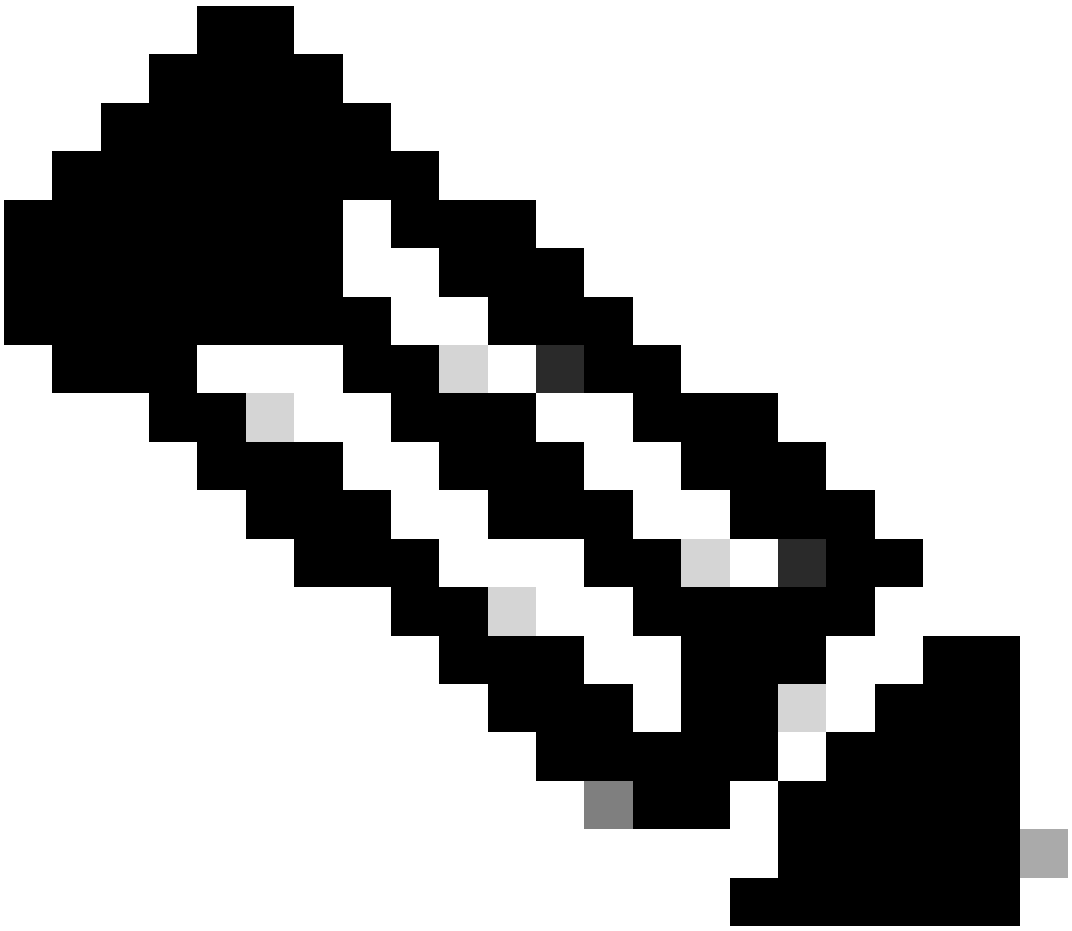
È ora possibile distribuire una rete di pod al cluster.

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

È ora possibile unire come root qualsiasi numero di nodi del control plane o di nodi master che eseguono questo comando su ciascuno di essi.

Riferimento: [flusso di lavoro kubeadm join]

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



Nota: La chiave del certificato consente di accedere ai dati sensibili al cluster e di mantenerla segreta.

A titolo di salvaguardia, i certificati caricati vengono eliminati in due ore; se necessario, è possibile utilizzarli.

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

È quindi possibile unire qualsiasi numero di nodi di lavoro ed eseguirli su ciascuno come nodi radice.

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

Passaggio 7. Verificare il servizio DNS di base.

```
[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]#
```

Passaggio 8. Verificare lo stato del nodo principale.

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

Per unire in join più nodi master, questo comando di join viene eseguito sui nodi master.

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

Possibili errori rilevati al momento della generazione del token

Errore 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
```

Risoluzione errore CRI

Passaggio 1. Rimuovere il file **config.toml** e riavviare il **contenitore**.

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

Passaggio 2. Installare il contenitore:
Installare il pacchetto containerd.io dai repository Docker ufficiali con questi comandi.

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

Passaggio 3. Configurare il contenitore:

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

Passaggio 4. Riavviare il contenitore:

```
systemctl restart containerd
```

Errore 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
```

FileContent di errore—proc-sys-net-ipv4-ip_forward Resolution

Impostare il valore ip_forward su 1.

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

Il servizio DNS di base non viene eseguito

```
[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]#
```

Risoluzione

Il DNS di base è in sospeso, il che indica un problema di rete. Per questo motivo, è necessario installare Calico.

Riferimento: Calico

Eseguire i due comandi seguenti:

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

Nodo Worker

Nel nodo di lavoro quando il token viene ottenuto dal master:

Passaggio 1. Abilitare il servizio Kubelet.

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

Passaggio 2. Unire tutti i nodi di lavoro con il nodo master con questo comando di join.

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

Passaggio 3. Se vengono rilevati errori relativi a file o porte, ripristinare il kubeadm con questo comando.

```
kubeadm reset
```

Dopo la reimpostazione, immettere il token dal nodo master.

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

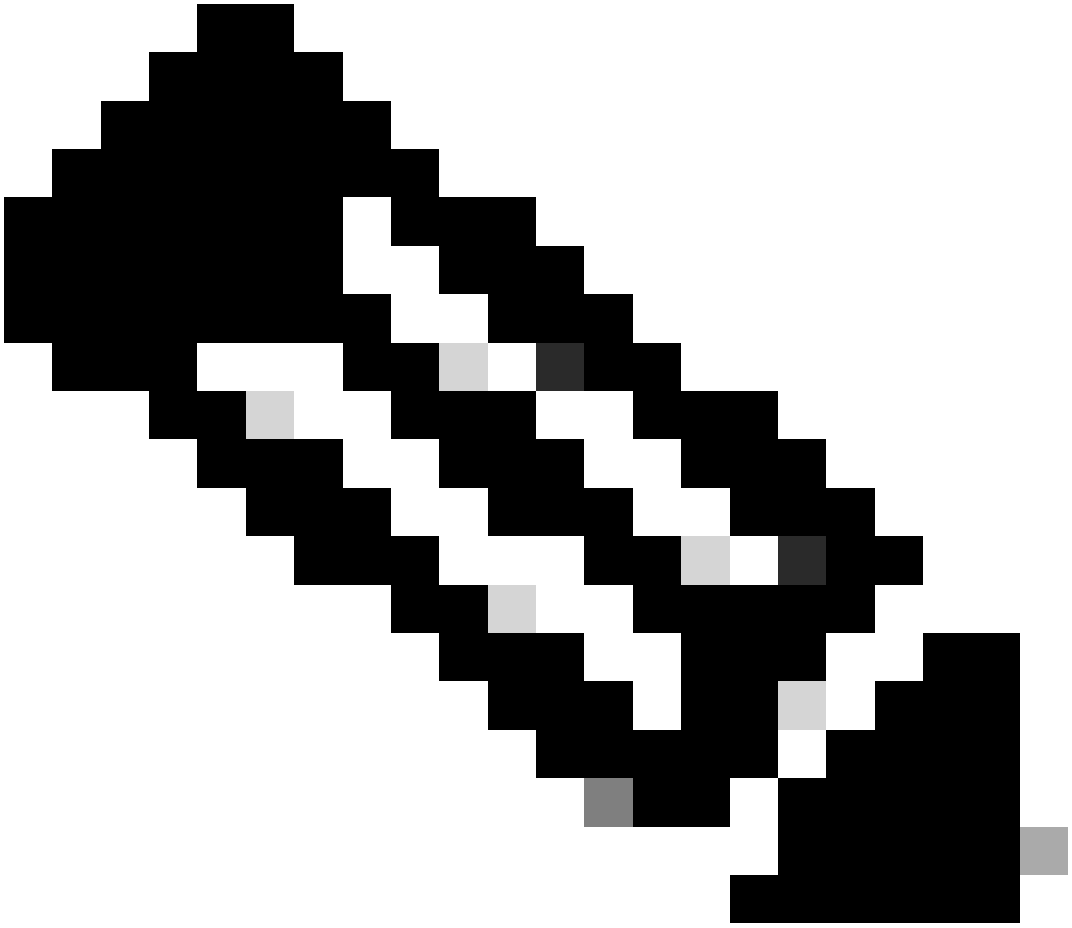
```
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

Output finale

Il cluster è ora formato. Verificarlo con il 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]#
```

output cluster kubernetes ad alta disponibilità



Nota: Al momento della formazione del cluster, viene configurato solo il controllo dai nodi master. L'host bastione non è configurato come server centralizzato per monitorare tutti i Kube nel cluster.

Informazioni su questa traduzione

Cisco ha tradotto questo documento utilizzando una combinazione di tecnologie automatiche e umane per offrire ai nostri utenti in tutto il mondo contenuti di supporto nella propria lingua. Si noti che anche la migliore traduzione automatica non sarà mai accurata come quella fornita da un traduttore professionista. Cisco Systems, Inc. non si assume alcuna responsabilità per l’accuratezza di queste traduzioni e consiglia di consultare sempre il documento originale in inglese (disponibile al link fornito).