Google Cloud Platform에서 Centos 7로 멀티 마스터 Kubernetes 클러스터 생성

목차

소개

이 문서에서는 Google Cloud Platform(GCP)에서 로드 밸런서 역할을 하는 베이션 호스트가 포함된 3개의 마스터 노드와 4개의 작업자 노드를 포함하는 Kubernetes 클러스터의 구현에 대해 설명합니다.

사전 요구 사항

요구 사항

다음 주제에 대한 지식을 보유하고 있으면 유용합니다.

- Linux
- 부두 및 쿠베르네테
- Google 클라우드 플랫폼

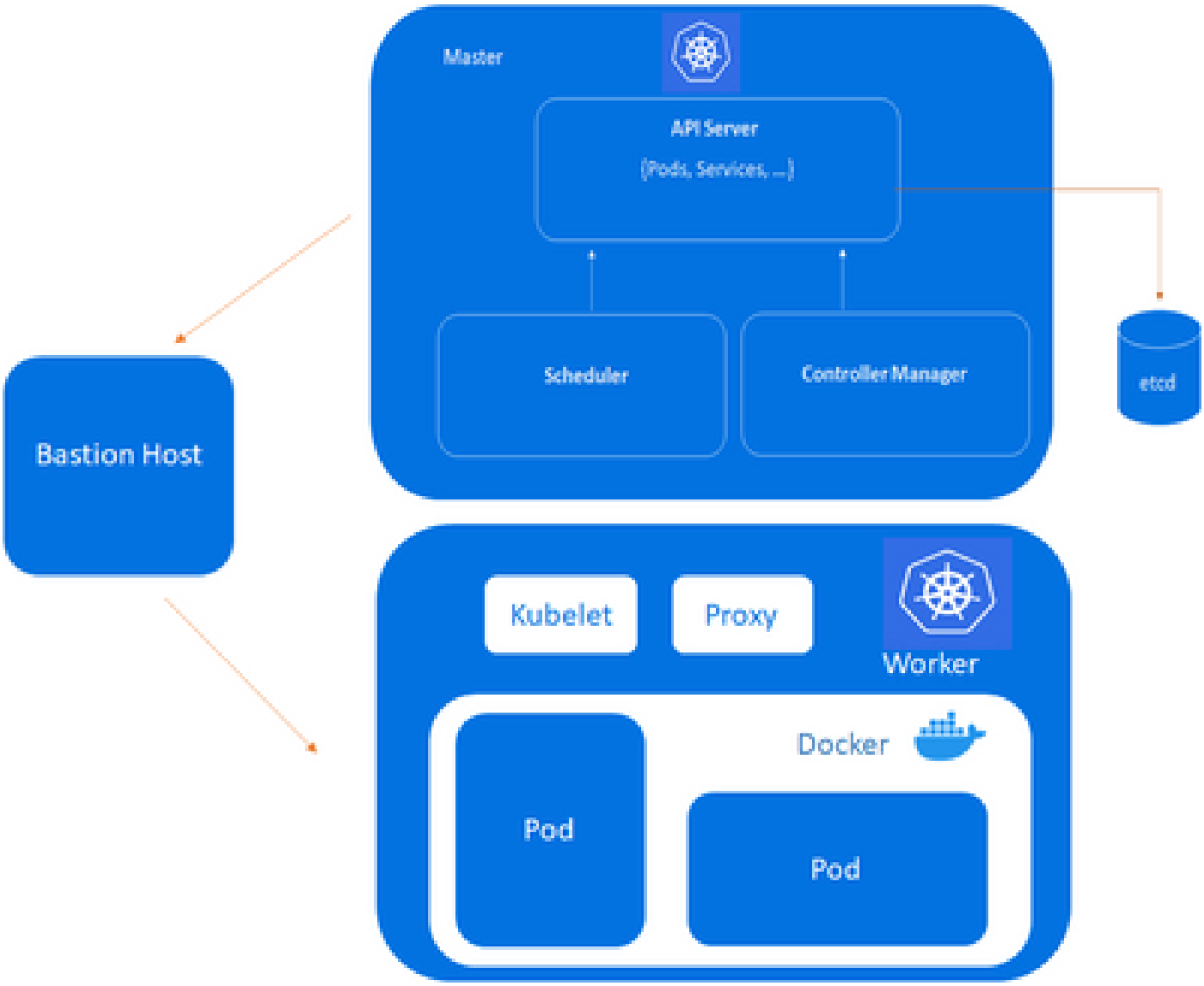
사용되는 구성 요소

이 문서의 정보는 다음을 기반으로 합니다.

- OS - Centos 7 가상 머신
- 시스템 제품군(e2-standard-16):
  - vCPU - 16개 vCPU
  - RAM - 64GB

이 문서의 정보는 특정 랩 환경의 디바이스를 토대로 작성되었습니다. 이 문서에 사용된 모든 디바이스는 초기화된(기본) 컨피그레이션으로 시작되었습니다. 현재 네트워크가 작동 중인 경우 모든 명령의 잠재적인 영향을 미리 숙지하시기 바랍니다.

네트워크 다이어그램



Bastion 호스트, Kube-Master, Kube-node

Kubernetes 마스터 노드 구성 요소

Kube-apiserver:
i. Kubernetes 컨트롤 플레인의 프런트 엔드 역할을 하는 API를 제공합니다.
나. 요청이 유효한지 확인한 다음 처리하는 외부 및 내부 요청을 처리합니다.
iii. API는 CLI(Command Line Interface) 또는 **kubectl** 기타 툴(예: REST 호출)을 통해 **kubeadm**액세스할 수 있습니다.

Kube 스케줄러:
i. 이 구성 요소는 자동화된 워크플로 및 사용자 정의 조건에 따라 특정 노드의 포드를 예약합니다.

Kube 컨트롤러 관리자:
i. Kubernetes 컨트롤러 관리자는 Kubernetes 클러스터의 상태를 모니터링하고 제어하는 제어 루프입니다.
나. 클러스터의 현재 상태 및 클러스터 내의 객체에 대한 정보를 수신하고 클러스터 운영자가 원하는 상태로 클러스터를 이동하라는 명령을 전송합니다.

etcd:
i. 클러스터 상태 및 컨피그레이션에 대한 데이터를 포함하는 키 값 데이터베이스입니다.
나. Etcd는 내결함성이며 배포됩니다.

Kubernetes 작업자 노드 구성 요소

쿠벨렛:
i. 각 노드에는 Kubernetes **kubelet**컨트롤 플레인과 통신할 수 있는 소형 애플리케이션인 가 포함되어 있습니다.
나. 는 **kubelet** Pod 컨피그레이션에 지정된 컨테이너가 특정 노드에서 실행되도록 보장하며, 라이프 사이클을 관리합니다.
iii. 컨트롤 플레인에 의해 명령된 작업을 실행합니다.

Kube-proxy:
i. 모든 컴퓨팅 노드에는 Kubernetes 네트워킹 서비스**kube-proxy**를 지원하는 네트워크 프록시가 포함되어 있습니다.
나. 클러스터 외부 및 내부의 모든 네트워크 통신을 처리하고 운영 체제의 패킷 필터링 레이어에 트래픽 또는 응답을 전달합니다.

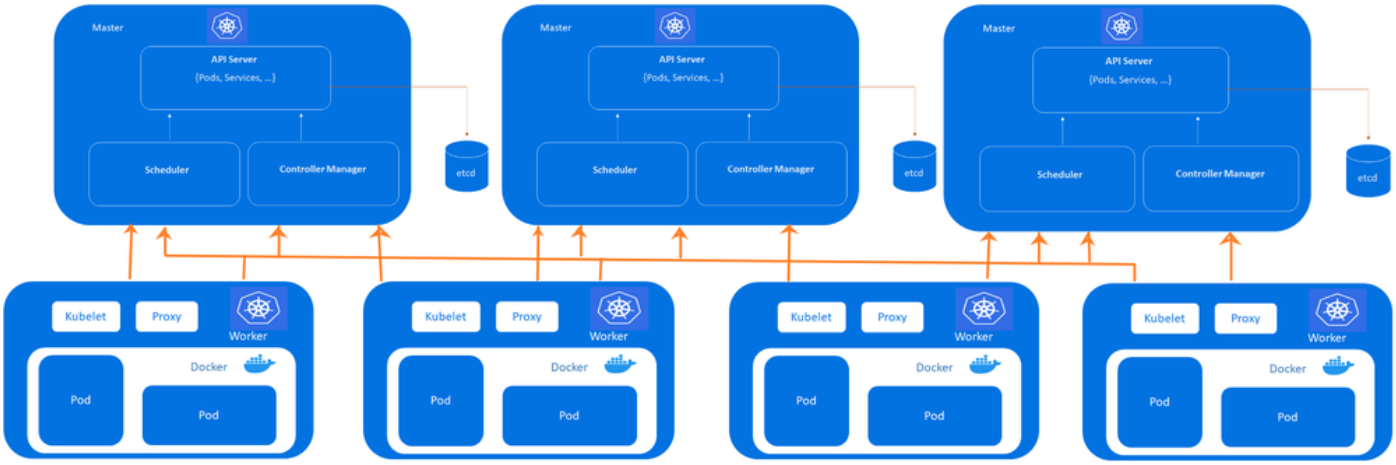
포드:
i. Pod는 단일 애플리케이션 인스턴스 역할을 하며 Kubernetes의 개체 모델에서 가장 작은 단위로 간주됩니다.

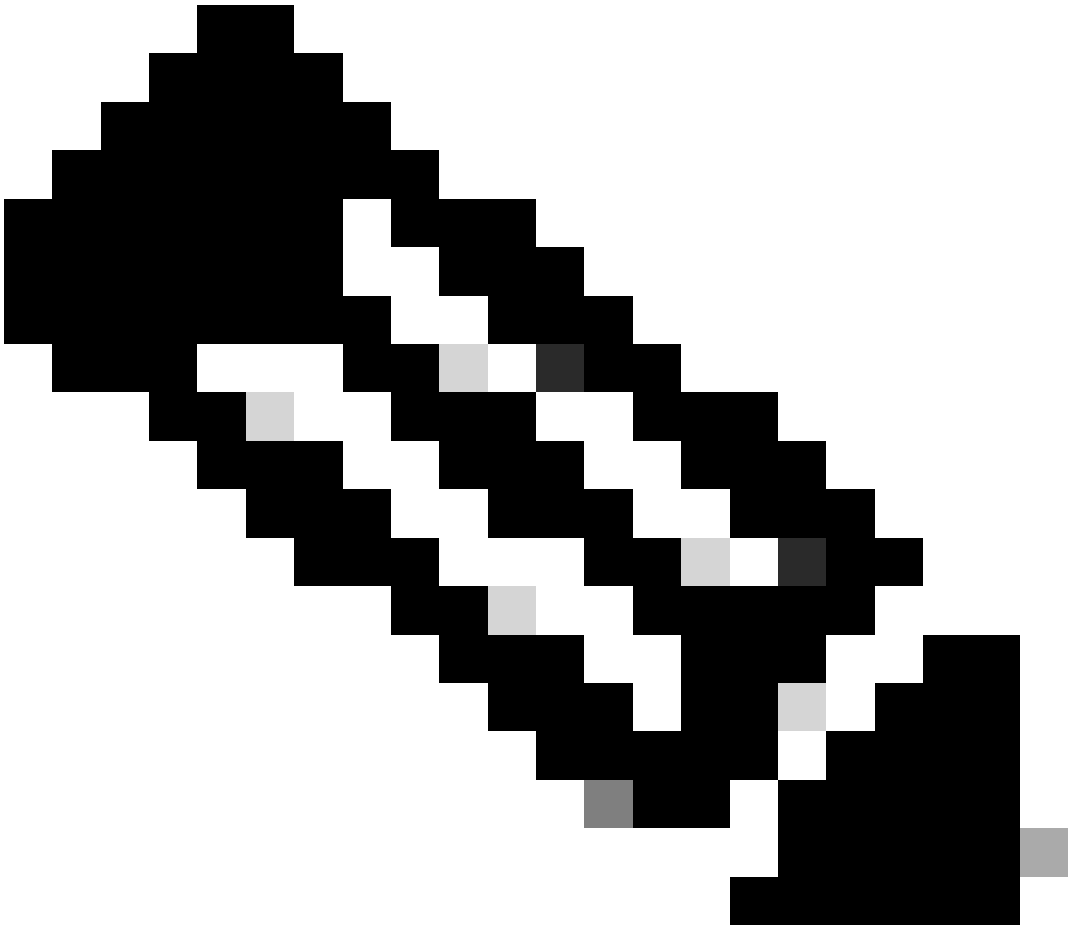
요새 호스트:
i. 일반적으로 컴퓨터는 프록시 서버 또는 로드 밸런서와 같은 단일 애플리케이션 또는 프로세스를 호스팅하며, 컴퓨터에 대한 위협을 줄이기 위해 다른 모든 서비스를 제거하거나 제한합니다.

Kubernetes 멀티 마스터 아키텍처

클러스터:

- 8 - 총 노드
- 3 - 마스터 노드
- 4 - 작업자 노드
- 1 - Basation Server(로드 밸런서)

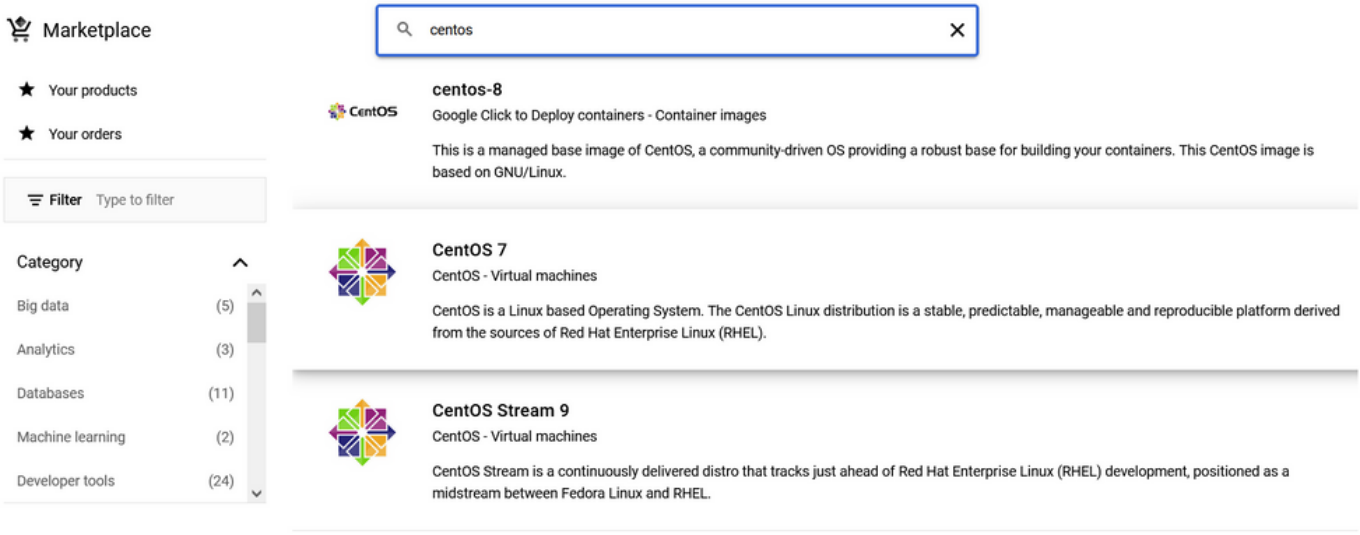




참고: VM을 프로비저닝하기 전에 GCP에 VPC를 구성합니다. GCP의 VPC를 참조합니다.

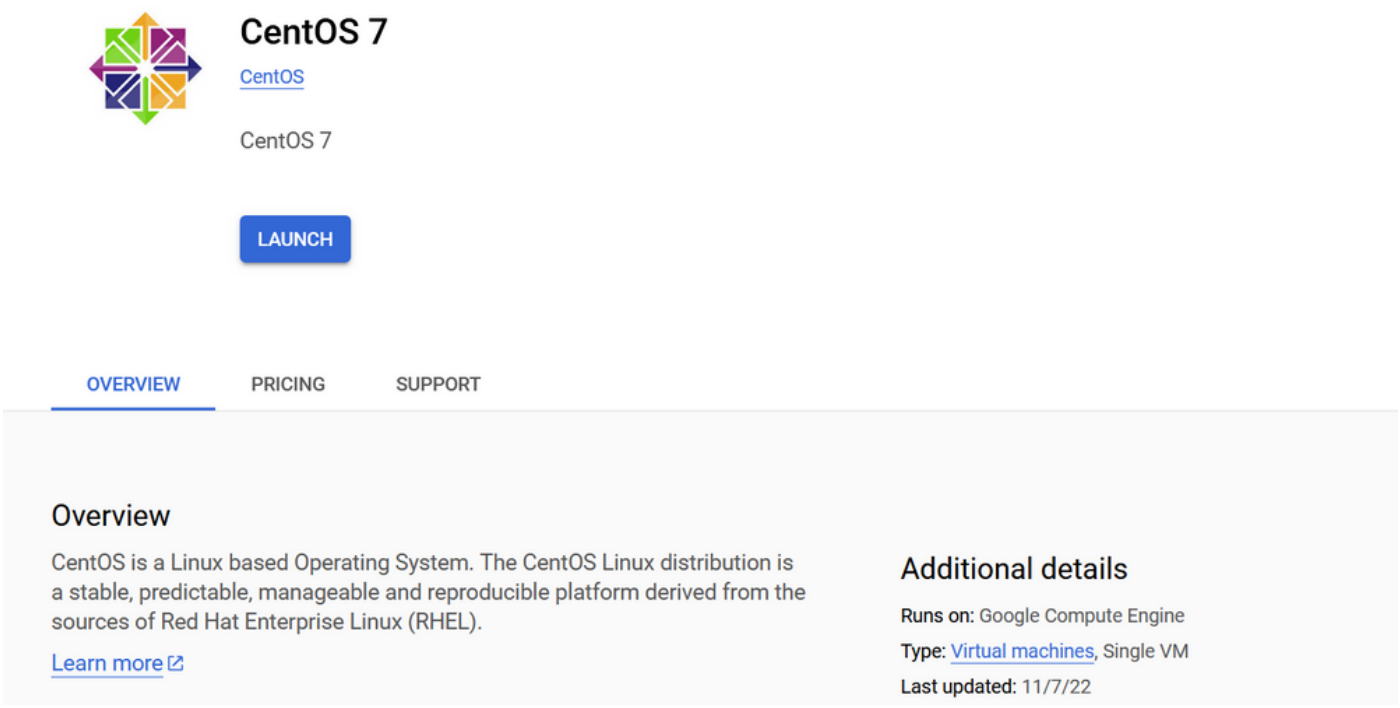
GCP에서 가상 머신 프로비저닝

GCP에서 GCP 마켓플레이스에서 Centos7을 프로비저닝합니다.



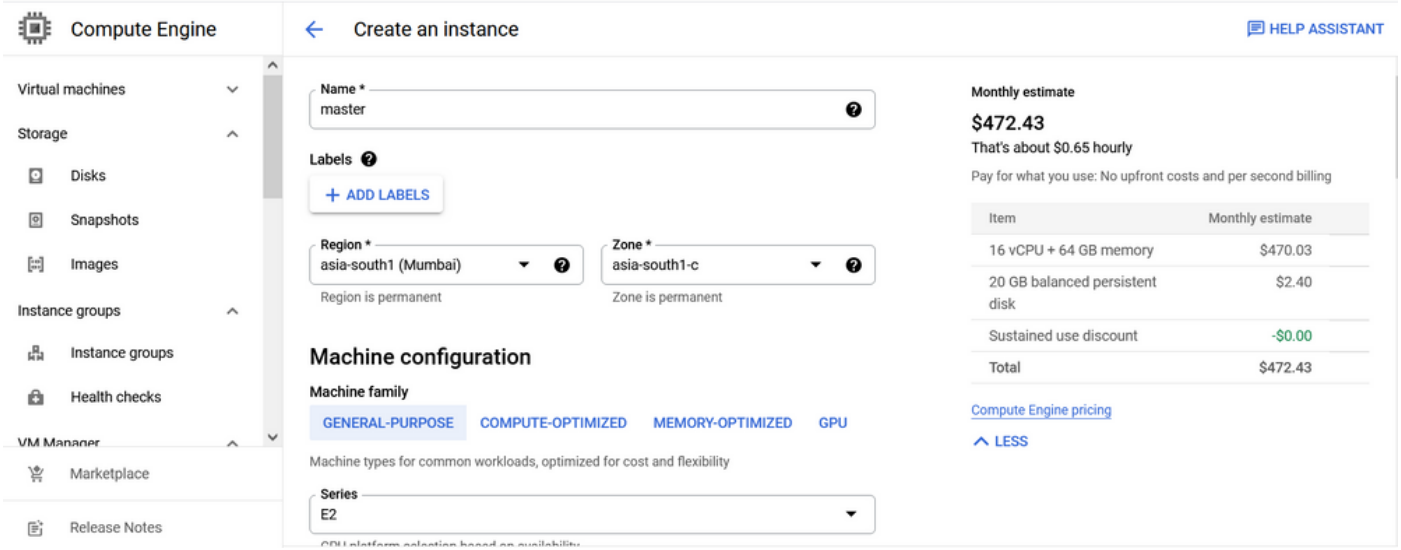
GCP의 Centos 마켓플레이스

을 클릭합니다.**Launch**



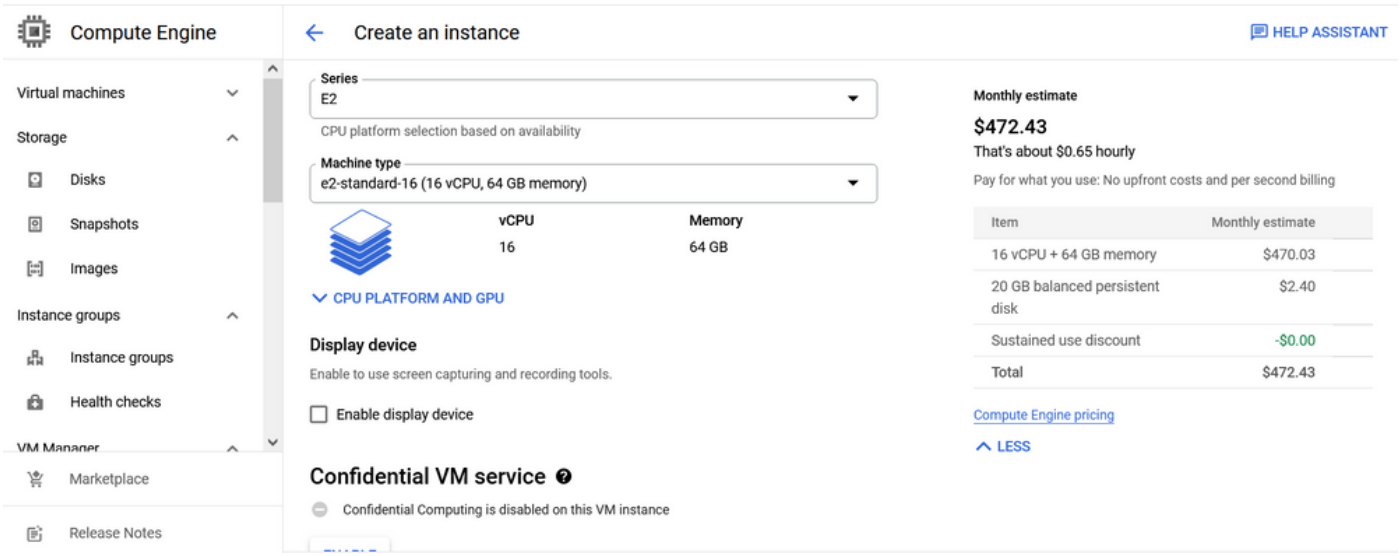
Centos 7 가상 머신

가장 가까운 연결성에 따라 지역을 선택합니다. 이 Lab에서는 해당 지역이 뭄바이로 구성됩니다.



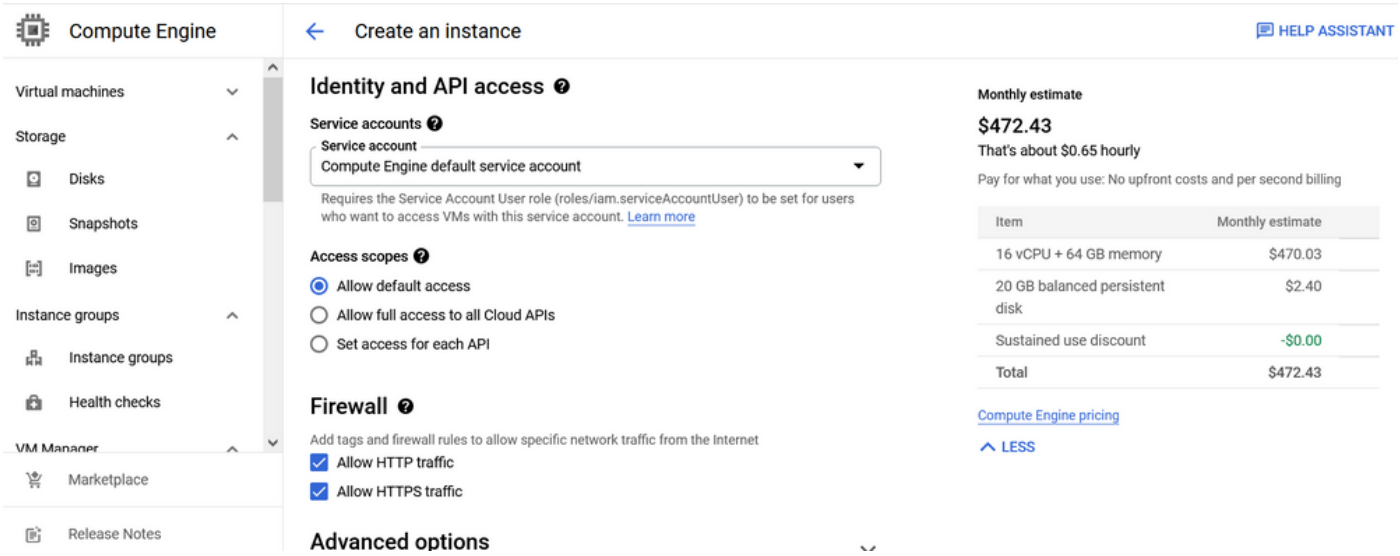
Centos7 컴퓨팅 엔진 컨피그레이션

시스템 컨피그레이션은 범용 E2 시리즈 및 시스템 유형입니다**e2-standard-16 (16 vCPU, 64 GB memory).**



Centos 7 리소스 컨피그레이션

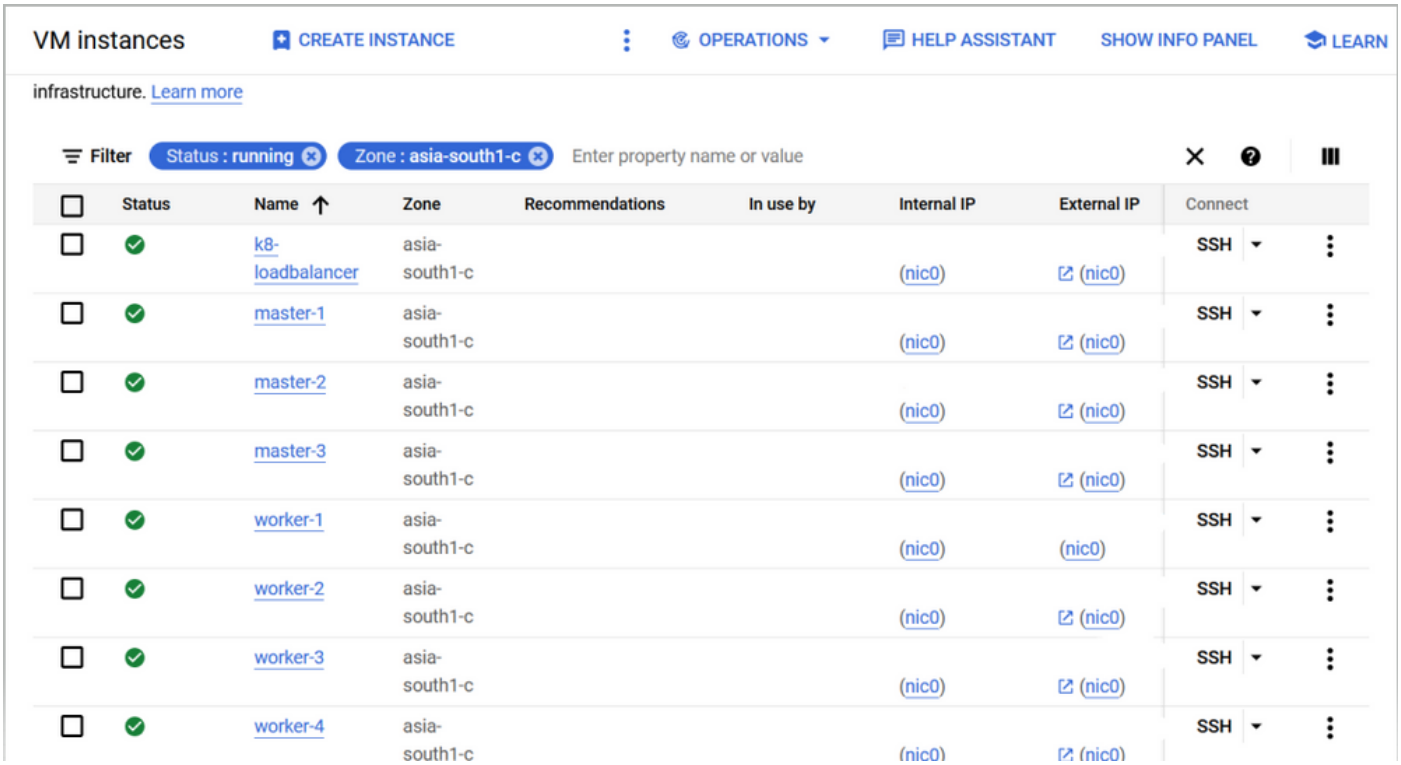
방화벽**Allow default access**에 대해 및 **Allow HTTP traffic** 을 선택합니다**Allow HTTPS traffic.**



Centos 7 네트워크 컨피그레이션

을 클릭합니다.**Create**

마찬가지로, 여기에 표시된 대로 8개의 노드를 생성합니다.



Google 클라우드 플랫폼에 멀티 마스터 구축

개인 IP:

GCP에서는 사설 및 공용 IP가 자동으로 할당됩니다.

```
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
```

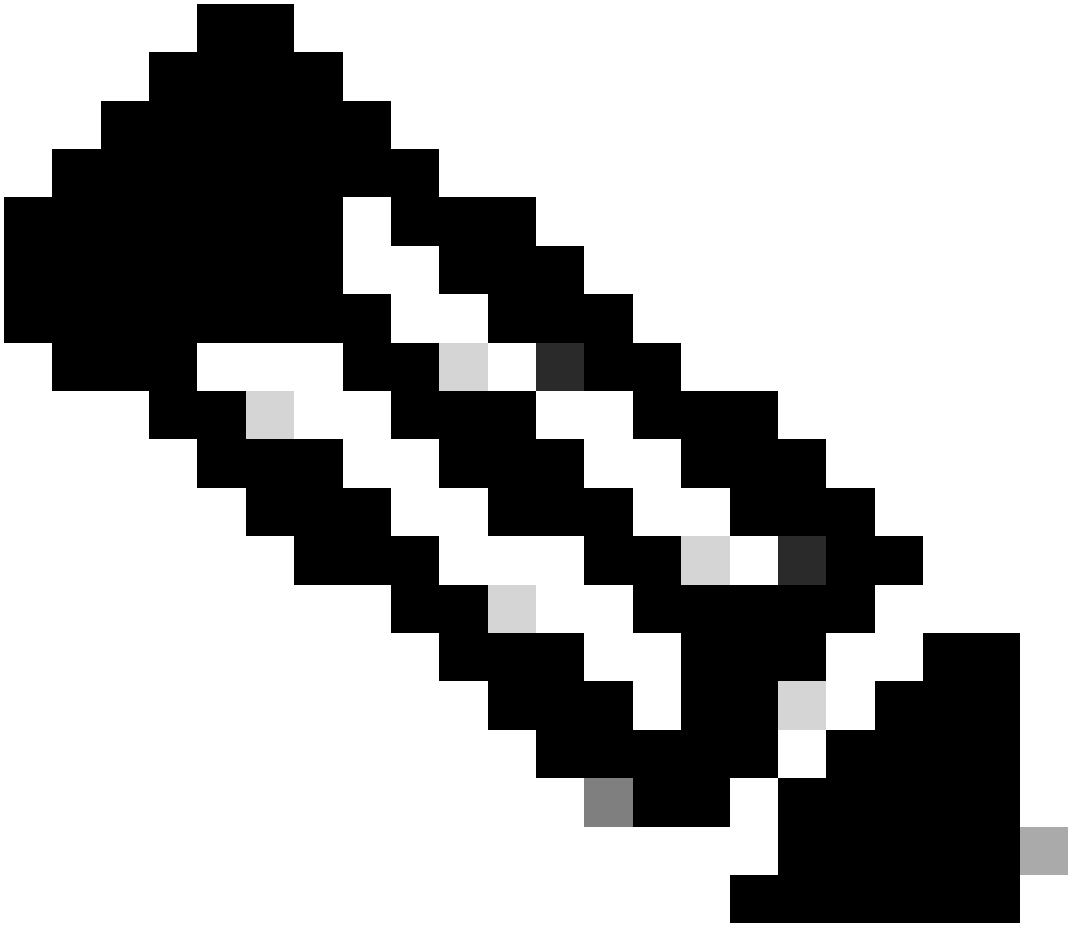
개괄적 개요

모든 노드(마스터, 작업자, 보스턴)에서 다음을 수행합니다.

1. Linux 서버에서 필요한 방화벽 포트를 열고 보안 구성을 구성합니다.

멀티 마스터 구축의 마스터 노드에서 다음을 수행합니다.

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



참고: 또한 GCP 방화벽의 포트가 허용되는지 확인합니다.

---

2. 필요한 네트워크 설정(로컬 DNS, 호스트 이름, NTP)을 구성합니다.

요새 서버:
1. HA 프록시를 설정합니다.
2. 파일에 프런트 엔드 및 백 엔드 서버 구성을 **haproxy.conf** 추가합니다.
3. 서비스를 다시 **haproxy** 시작합니다.

마스터 및 작업자 노드에 대한 일반적인 단계:
1. docker를 설치하고 구성합니다.
2. Kubernetes 설치 및 구성

마스터 노드에서만:
1. 새 Kubernetes 클러스터를 초기화합니다(**kubeadm init**).
2. 네트워크 **Calico** 플러그인(특히 마스터 노드의 코어 DNS 서비스에 사용됨)을 설치합니다.
3. 마스터 노드를 마스터 노드와 조인하려면 이 명령을 사용합니다.

```
kubeadm join

        :6443 --token

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

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

4. 명령을 사용하여 클러스터 정보를 **kubectl get nodes**확인합니다.

작업자 노드에서만:

1. 이 명령을 사용하여 작업자 노드를 마스터 노드에 조인합니다.

```
kubeadm join

        :6443 --token

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

2. 가입에 성공하면 이 명령으로 마스터 노드의 클러스터 정보를 **kubectl get nodes**확인합니다.

하위 레벨 컨피그레이션

네트워크 설정

1. 이 명령으로 알 수 없는 경우 루트 비밀번호를 변경합니다.

```
passwd
```

2. 필요한 경우 이 명령을 사용하여 호스트 이름을 변경합니다.

```
hostnamectl set-hostname
```

3. 로컬 DNS를 구성합니다.

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

4. 이 명령을 사용하여 NTP 서비스에 대한 시간 기록을 활성화합니다.

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

2. 이 명령으로 상태를 확인합니다.

```
systemctl status chronyd
```

3. 이 명령을 사용하여 NTP 소스를 확인합니다.

```
chronyc sources -v
```

요새 서버

1단계. 업데이트를 확인합니다.

```
sudo yum check-update
```

2단계. 최신 버전이 아닌 경우 업데이트를 설치합니다.

```
yum update -y
```

3단계. yum 유틸리티를 설치합니다.

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

4단계. Haproxy 패키지를 설치합니다.

```
yum install haproxy -y
```

5단계. 이 구성을 다음 위치의 기본값 아래에 **/etc/haproxy/haproxy.cfg** 추가합니다.

```
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 /
```

6단계. 이 명령과 비슷하게 컨피그레이션 파일을 **vi /etc/haproxy/haproxy.cfg**확인합니다.

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

7단계. happroxy 상태를 확인합니다.

```
[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]#
```

가능한 오류.

haproxy.cfg1. 의 컨피그레이션을 변경한 후 HAProxy 서비스가 실패 상태입니다. 예:

```
[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.
```

해결

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

실행:

```
[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]#
```

마스터 및 작업자 노드에 Docker 설치

1단계. 업데이트를 확인합니다.

```
sudo yum check-update
```

2단계. 최신 버전이 아닌 경우 업데이트를 설치합니다.

```
yum update -y
```

3단계. yum 유틸리티를 설치합니다.

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

4단계. Docker 설치

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

5단계. docker를 활성화합니다.

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

6단계. docker 서비스를 시작합니다.

```
sudo systemctl start docker
```

7단계. Docker 상태를 확인합니다.

```
sudo systemctl status docker
```

성과:

```
[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]#
```

마스터 및 작업자 노드에 Kubernetes 설치

1단계. Kubernetes 저장소를 추가합니다.

```
cat <
```

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

2단계. 비활성화합니다**SELinux.**

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

3단계. **Kubernetes**설치합니다.

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

4단계. Enable(활성화)**Kubelet**합니다.

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

5단계. **Pod Network**구성합니다.

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

6단계. 토큰 생성:

마스터(컨트롤 플레인) 및 작업자 노드에 대한 출력입니다.

마스터 노드

토큰: Kubernetes 컨트롤 플레인이 초기화되었습니다!

클러스터를 사용하려면 이 명령을 일반 사용자로 실행합니다.

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

또는 루트 사용자인 경우 를 실행할 수 있습니다.

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

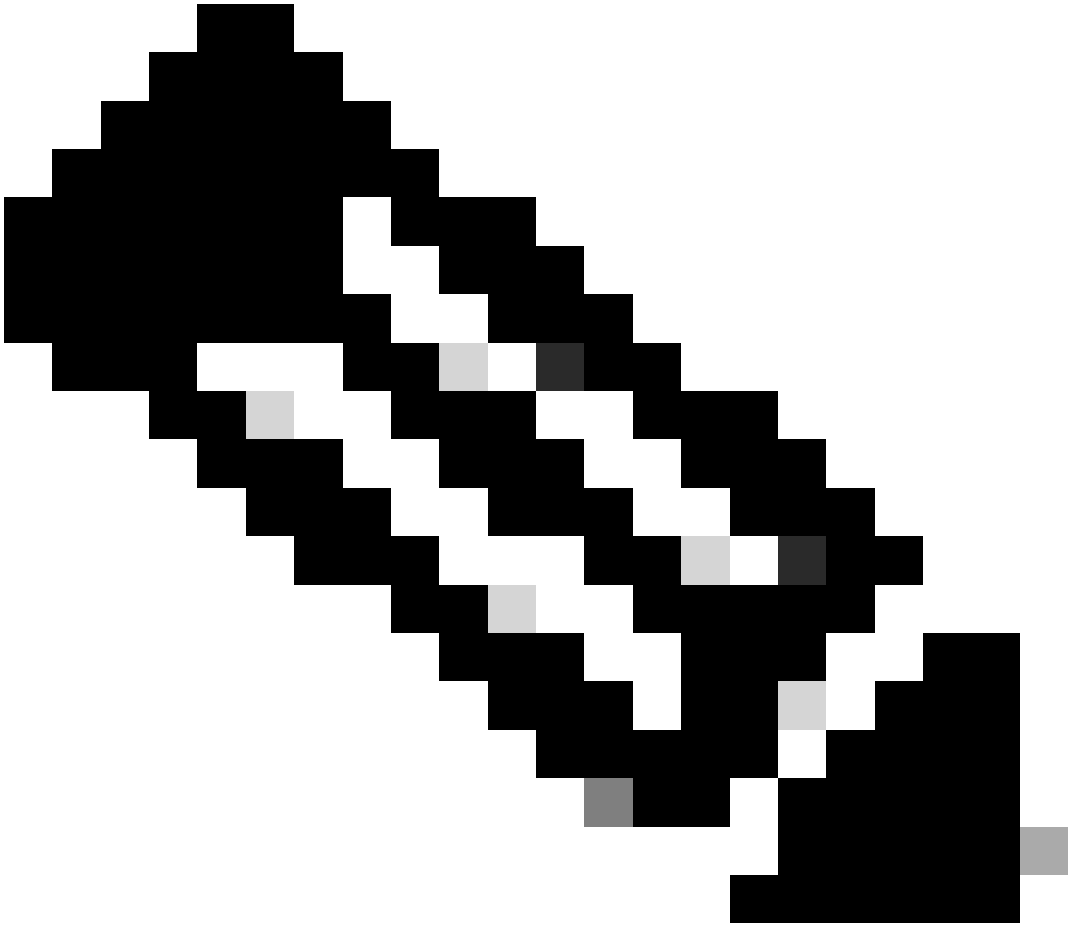
이제 Pod 네트워크를 클러스터에 구축할 수 있습니다.

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

이제 이 명령을 실행하는 컨트롤 플레인 노드 또는 마스터 노드를 루트로 사용할 수 있습니다.

참조: [kubeadm 가입 워크플로](#)

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



참고: 인증서 키는 클러스터에 민감한 데이터에 대한 액세스를 제공하므로 비밀로 유지하십시오!

보안 수단으로 업로드된 인증서는 2시간 이내에 삭제됩니다. 필요한 경우 사용할 수 있습니다.

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

그런 다음 원하는 수의 작업자 노드에 조인하고 각 노드에서 이를 루트로 실행할 수 있습니다.

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

7단계. 코어 DNS 서비스를 확인합니다.

```
[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]#
```

8단계. 마스터 노드의 상태를 확인합니다.

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

여러 마스터 노드를 조인하려면 이 조인 명령이 마스터 노드에서 실행됩니다.

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

토큰 생성 시 발생 가능한 오류

오차 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
```

오류 CRI 해결

1단계. config.toml **파일을** 제거하고 **containered**를 다시 **시작합니다.**

```
rm -rf /etc/containerd/config.toml
```

```
systemctl restart containerd
kubeadm init
```

2단계. 포함된 설치:
이 명령을 사용하여 공식 Docker 저장소에서 containerd.io 패키지를 설치합니다.

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

3단계. 포함된 항목을 구성합니다.

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

4단계. 재시작 포함됨:

```
systemctl restart containerd
```

오류 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
```

오류 파일 내용 - proc-sys-net-ipv4-ip_forward Resolution

ip_forward 값을 1로 설정합니다.

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

코어 DNS 서비스가 실행되지 않음

```
[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]#
```

해결

코어 DNS가 보류 중입니다. 이는 네트워킹에 문제가 있음을 나타냅니다. 따라서 Calico를 설치해야 합니다.

참조: Calico

다음 두 명령을 실행합니다.

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

작업자 노드

마스터에서 토큰을 가져올 때 작업자 노드에서 다음을 수행합니다.

1단계. kubelet 서비스를 활성화합니다.

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

2단계. 이 join 명령을 사용하여 모든 작업자 노드를 마스터 노드와 조인합니다.

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

3단계. 파일 또는 포트와 관련된 오류가 발생하면 이 명령으로 kubeadm을 재설정합니다.

```
kubeadm reset
```

재설정한 후 마스터 노드의 토큰을 입력합니다.

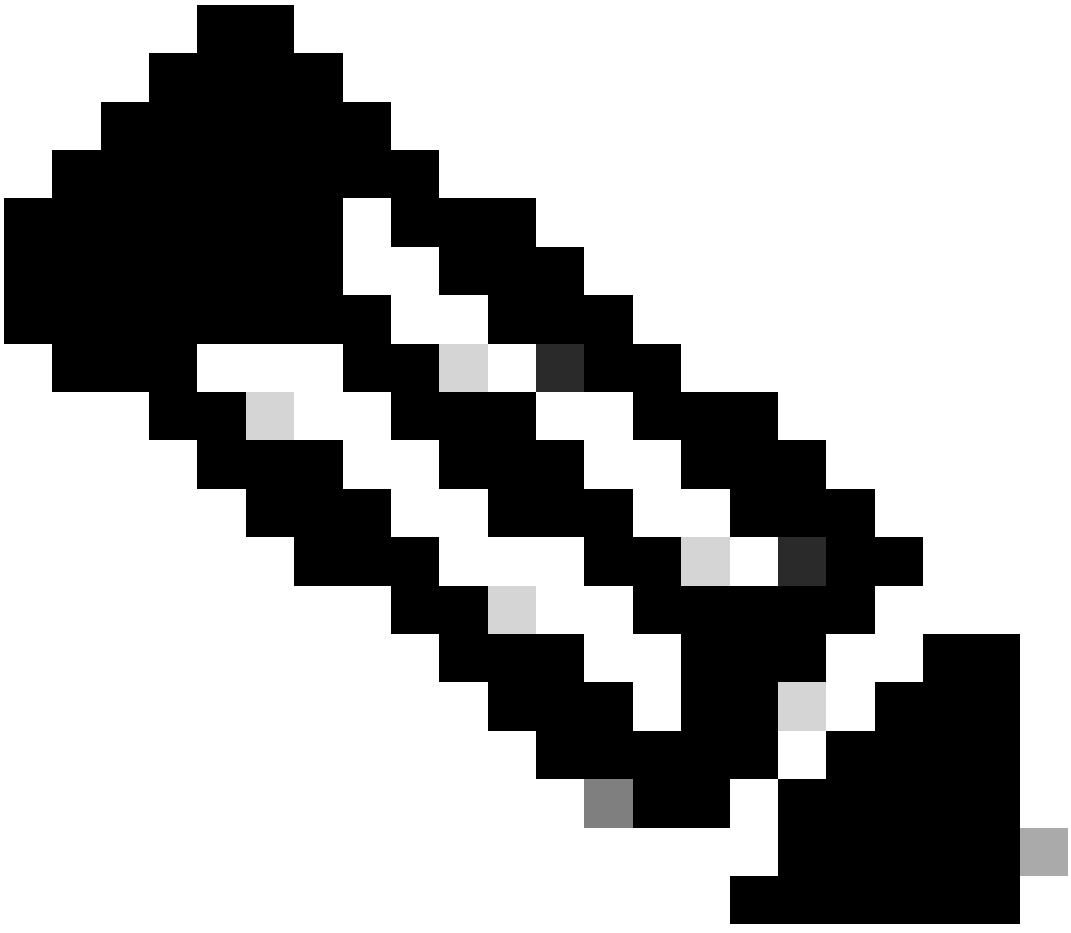
```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kyelronuyOmw2 \
        --discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

최종 출력

이제 클러스터가 구성되었습니다. `kubectl get node` 명령을 **사용하여 확인하십시오.**

```
[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]#
```

고가용성 *kubernetes* 클러스터 출력



참고: 클러스터 형성 시 마스터 노드의 제어만 구성됩니다. 베이션 호스트는 클러스터의 모든 Kube를 모니터링하기 위한 중앙 서버로 구성되지 않았습니다.

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.