

使用Google雲平台上的Centos 7建立多主Kubernetes集群

目錄

[簡介](#)

[必要條件](#)

[需求](#)

[採用元件](#)

[網路圖表](#)

[Kubernetes主節點元件](#)

[Kubernetes工作節點元件](#)

[Kubernetes多主架構](#)

[在GCP上提供虛擬機器](#)

[高級概述](#)

[低級配置](#)

[網路設定](#)

[堡壘伺服器](#)

[可能的錯誤](#)

[解析](#)

[在主節點和工作節點上安裝Docker](#)

[在主節點和工作節點上安裝Kubernetes](#)

[主節點](#)

[生成令牌時可能遇到的錯誤](#)

[錯誤CRI](#)

[錯誤CRI解決方案](#)

[錯誤檔案內容 — proc-sys-net-ipv4-ip_forward](#)

[錯誤檔案內容 — proc-sys-net-ipv4-ip_forward解析度](#)

[核心DNS服務未運行](#)

[解析](#)

[工作人員節點](#)

[最終輸出](#)

簡介

本文檔介紹使用3個主節點和4個輔助節點的Kubernetes群集的實施，該群集帶有在Google雲平台(GCP)上充當負載均衡器的輔助主機。

必要條件

需求

思科建議您瞭解以下主題：

- Linux
- 道克斯和庫伯內特
- Google雲端平台

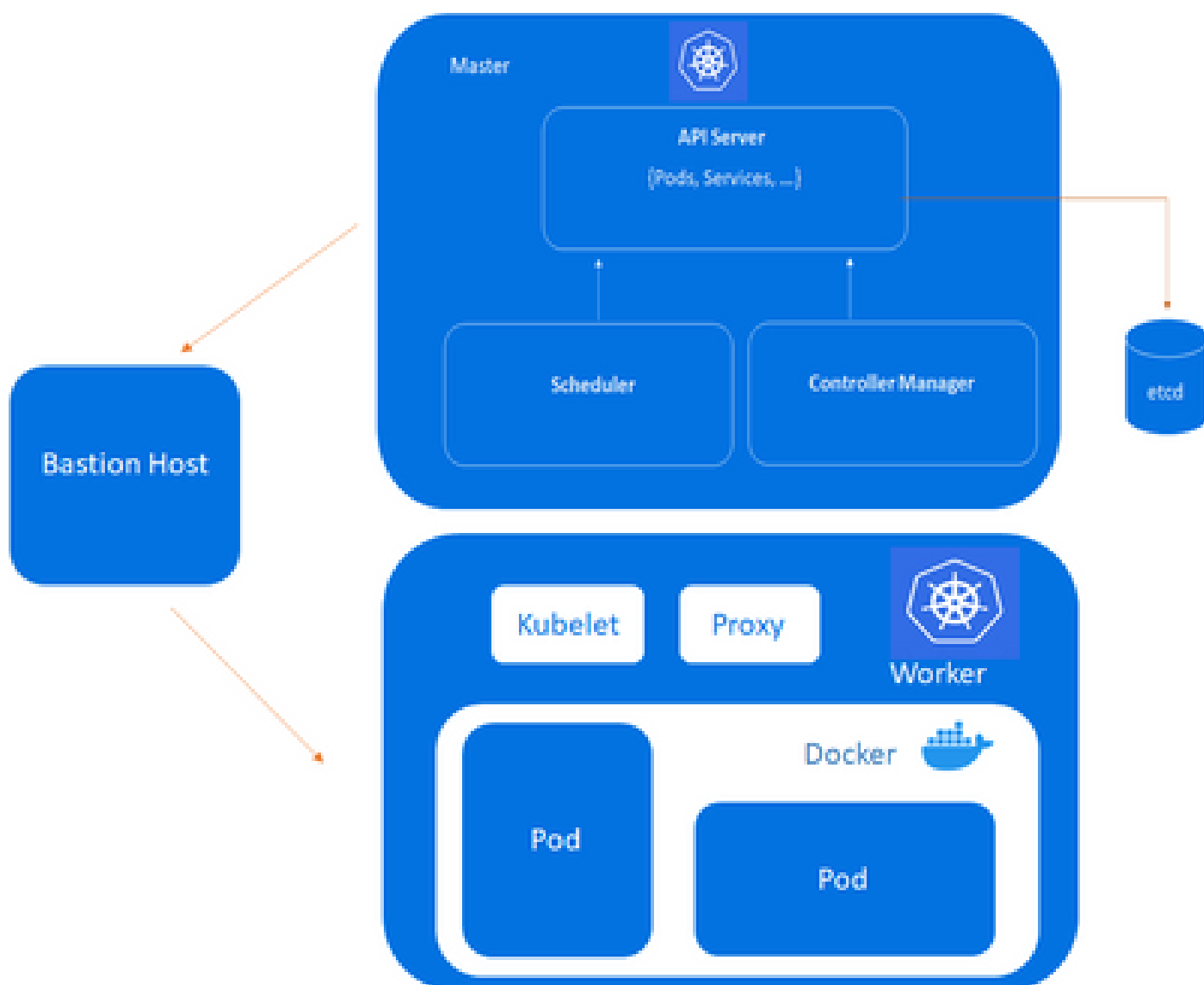
採用元件

本檔案中的資訊是根據：

- OS - Centos 7 虛擬機器
- 機器系列(e2-standard-16):
 - vCPU - 16 vCPU
 - RAM - 64 GB

本文中的資訊是根據特定實驗室環境內的裝置所建立。文中使用到的所有裝置皆從已清除（預設）的組態來啟動。如果您的網路運作中，請確保您瞭解任何指令可能造成的影響。

網路圖表



Kubernetes主節點元件

Kube-apiserver:

- i.提供一個充當Kubernetes控制平面前端的API。
- 二。它處理用於確定請求是否有效的外部 and 內部請求，然後處理該請求。
- 三。API可以通過命令列界kubecti面或其他工具（如）kubeadm和REST呼叫訪問。

Kube-scheduler:

- i.此元件根據自動工作流程和使用使用者定義的條件在特定的節點上安排Pod。

Kube-controller-manager:

- i.Kubernetes控制器管理器是一個控制環路，用於監視和調整Kubernetes群集的狀態。
- 二。它接收有關群集的當前狀態及其中對象的資訊，並傳送指令以將群集移動到群集操作員的期望狀態。

etcd:

- i.一個鍵值資料庫，其中包含有關群集狀態和配置的資料。
- 二。Etcd具有容錯性和分散式性。

Kubernetes工作節點元件

庫貝萊：

- i.每個節點都包含kubelelet一個，這是一個可以與Kubernetes控制平面通訊的小型應用程式。
- 二。確保kubelelet在Pod配置中指定的容器在特定節點上運行，並管理其生命週期。
- 三。它執行由控制平面命令執行的操作。

Kube-proxy:

- i.所有計算節點都包kubeproxy含一個網路代理，可促進Kubernetes網路服務。
- 二。它處理群集內外的所有網路通訊，並在作業系統的資料包過濾層轉發流量或應答。

Pod:

- i.Pod作為單個應用例項，被認為是庫伯內特斯的對象模型的最小單元。

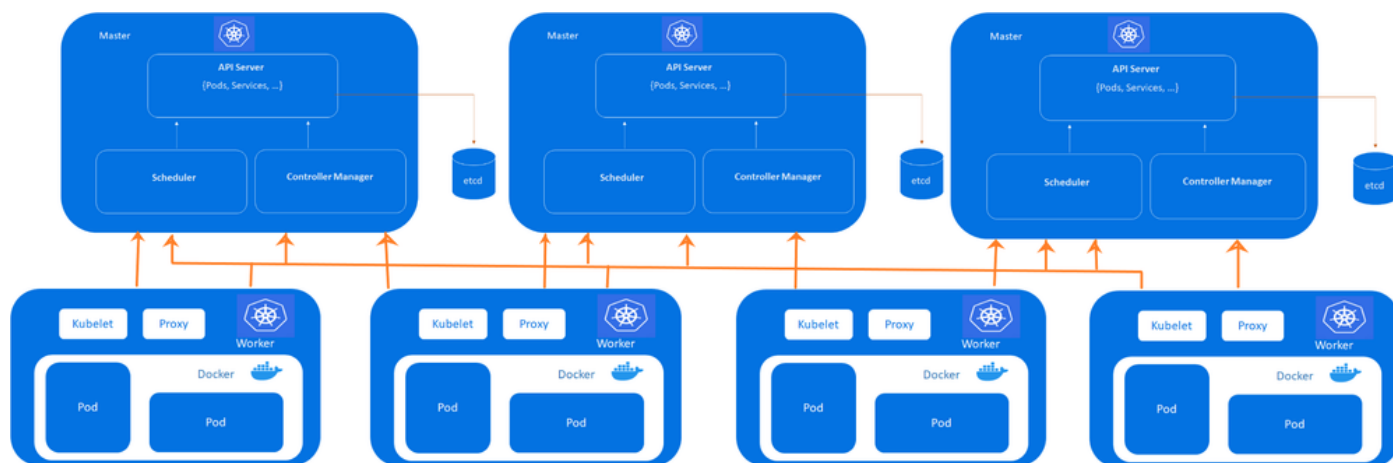
Bastion主機：

- i.電腦通常託管單個應用程式或進程，例如代理伺服器或負載均衡器，並且刪除或限制所有其他服務以減少對電腦的威脅。

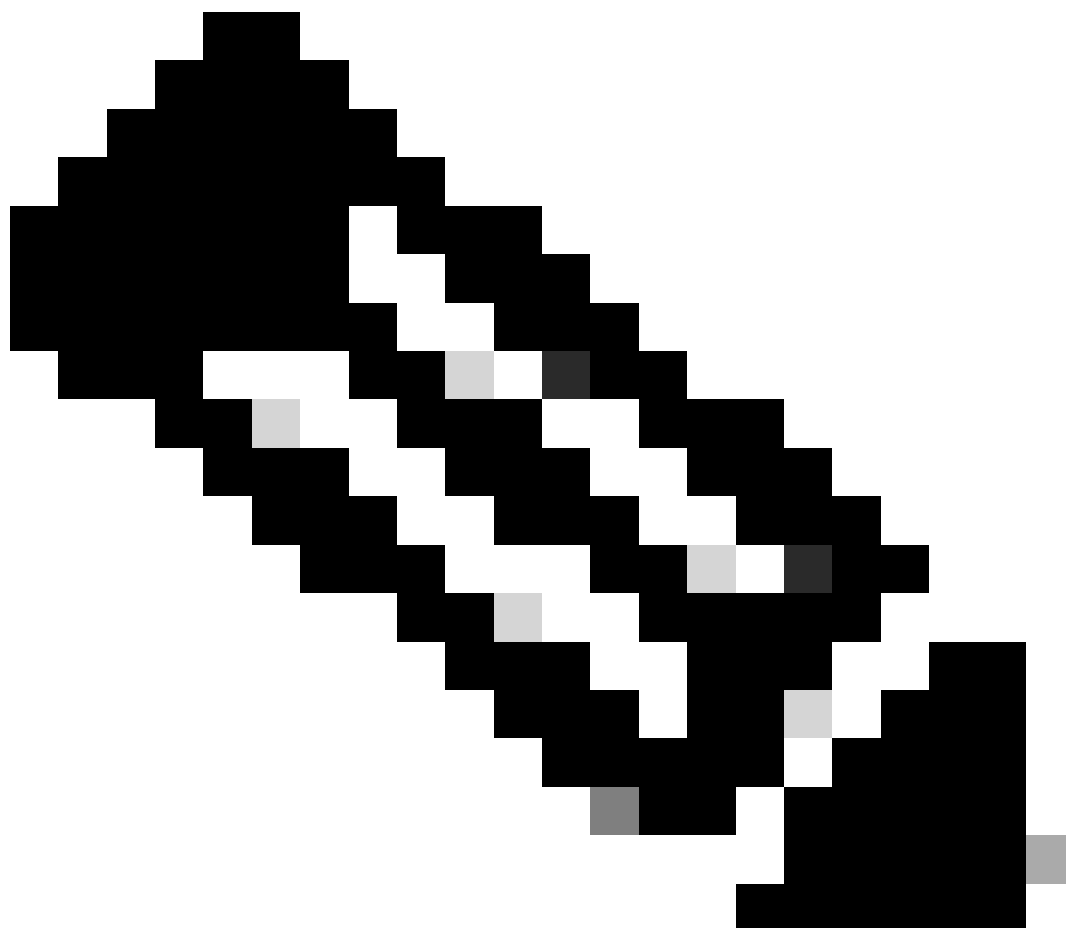
Kubernetes多主架構

叢集:

- 8 — 節點總數
- 3 — 主節點
- 4 — 工作節點
- 1 — 基準伺服器（負載均衡器）



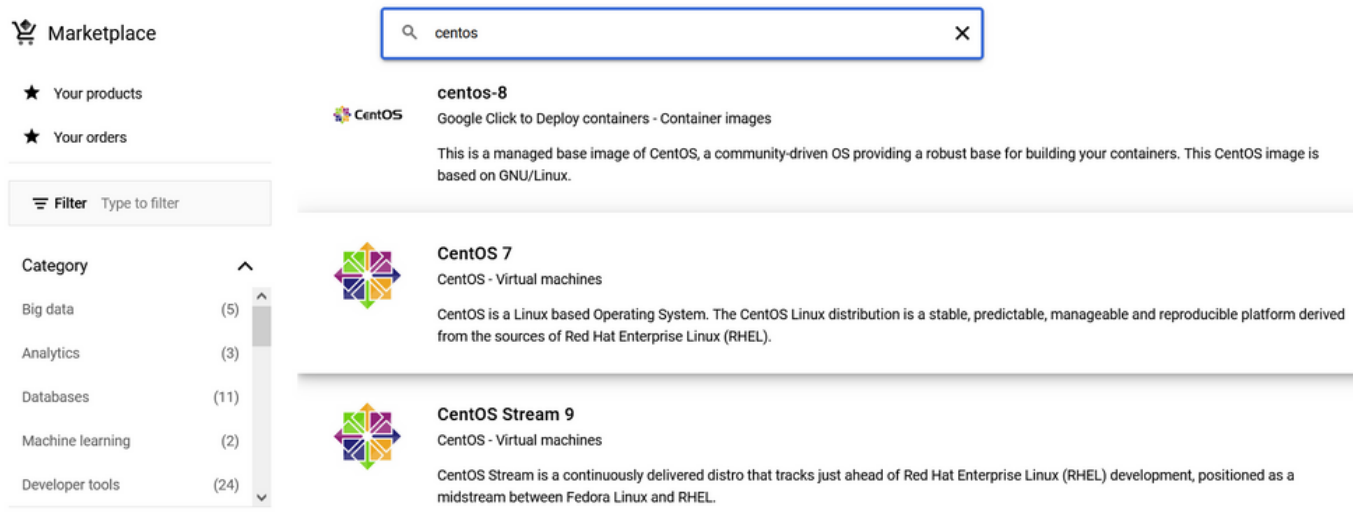
具有三個控制平面的高可用Kubernetes集群



附註：在調配VM之前，在GCP上配置VPC。參考[GCP上的VPC](#)。

在GCP上提供虛擬機器

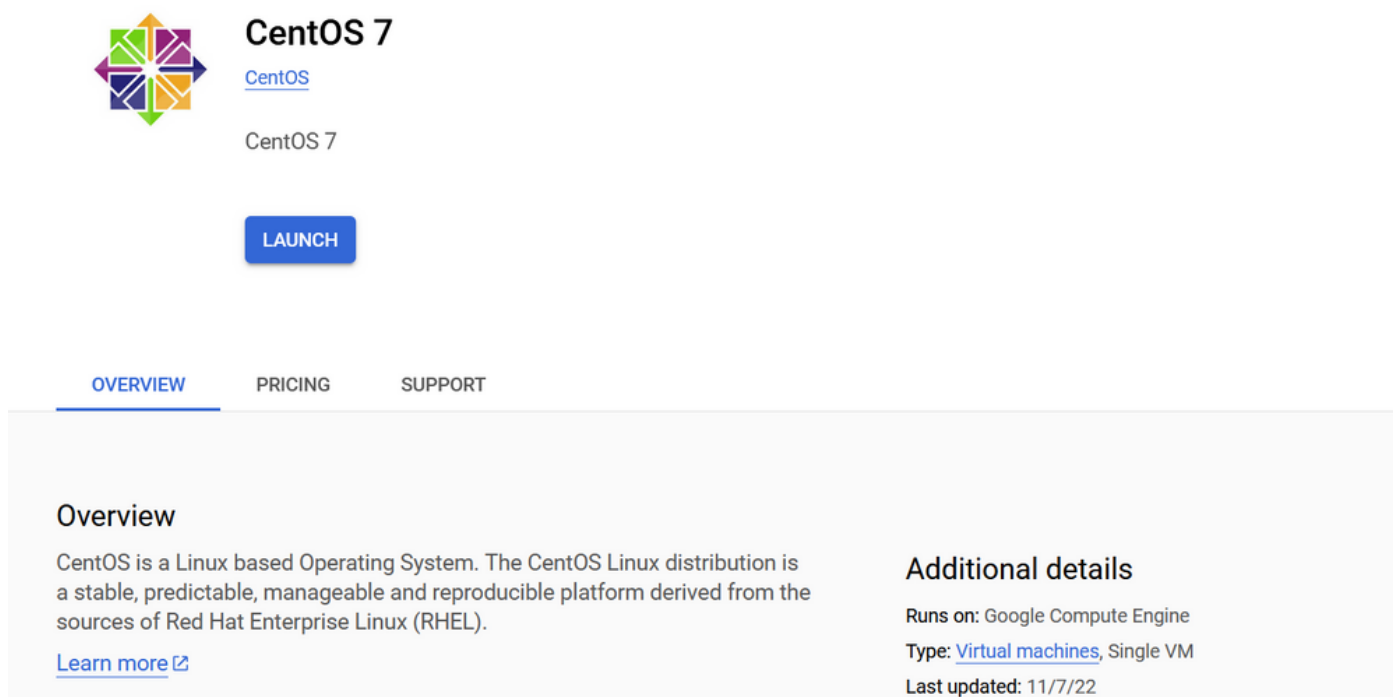
在GCP調配時，從GCP市場獲取Centos7。



The screenshot shows the Google Cloud Marketplace interface. At the top, there's a search bar with 'centos' entered. On the left, there's a sidebar with 'Marketplace' and 'Filter' options. The main area displays three results: 'centos-8' (Google Click to Deploy containers - Container images), 'CentOS 7' (CentOS - Virtual machines), and 'CentOS Stream 9' (CentOS - Virtual machines). Each result includes a brief description and the CentOS logo.

GCP上的Centos市場

按一Launch下。



The screenshot shows the detailed product page for 'CentOS 7'. It features the CentOS logo, the product name 'CentOS 7', and a 'LAUNCH' button. Below the product name, there are tabs for 'OVERVIEW', 'PRICING', and 'SUPPORT'. The 'OVERVIEW' tab is selected, showing a description of CentOS as a Linux-based operating system. To the right, under 'Additional details', it specifies 'Runs on: Google Compute Engine', 'Type: Virtual machines, Single VM', and 'Last updated: 11/7/22'.

Centos 7虛擬機器

根據最近的連通性選擇區域。在本實驗中，該區域被配置為Mumbai。

Virtual machines

Storage

Disks

Snapshots

Images

Instance groups

Instance groups

Health checks

VM Manager

Marketplace

Release Notes

Create an instance

HELP ASSISTANT

Name *
master

Labels
+ ADD LABELS

Region *
asia-south1 (Mumbai)
Region is permanent

Zone *
asia-south1-c
Zone is permanent

Machine configuration

Machine family

GENERAL-PURPOSE

COMPUTE-OPTIMIZED

MEMORY-OPTIMIZED

GPU

Machine types for common workloads, optimized for cost and flexibility

Series
E2

Monthly estimate

\$472.43

That's about \$0.65 hourly

Pay for what you use: No upfront costs and per second billing

Item	Monthly estimate
16 vCPU + 64 GB memory	\$470.03
20 GB balanced persistent disk	\$2.40
Sustained use discount	-\$0.00
Total	\$472.43

[Compute Engine pricing](#)

LESS

Centos7計算引擎配置

機器配置是通用E2系列和機器類e2-standard-16 (16 vCPU, 64 GB memory)型。

Virtual machines

Storage

Disks

Snapshots

Images

Instance groups

Instance groups

Health checks

VM Manager

Marketplace

Release Notes

Create an instance

HELP ASSISTANT

Series
E2
CPU platform selection based on availability

Machine type
e2-standard-16 (16 vCPU, 64 GB memory)

vCPU
16

Memory
64 GB

CPU PLATFORM AND GPU

Display device

Enable to use screen capturing and recording tools.

Enable display device

Confidential VM service

Confidential Computing is disabled on this VM instance

Monthly estimate

\$472.43

That's about \$0.65 hourly

Pay for what you use: No upfront costs and per second billing

Item	Monthly estimate
16 vCPU + 64 GB memory	\$470.03
20 GB balanced persistent disk	\$2.40
Sustained use discount	-\$0.00
Total	\$472.43

[Compute Engine pricing](#)

LESS

Centos 7資源配置

為防Allow default access火牆選擇and , Allow HTTP traffic和Allow HTTPS traffic。

Compute Engine

Virtual machines

Storage

Disks

Snapshots

Images

Instance groups

Instance groups

Health checks

VM Manager

Marketplace

Release Notes

←

Create an instance

HELP ASSISTANT

Identity and API access

Service accounts

Service account

Compute Engine default service account

Requires the Service Account User role (roles/iam.serviceAccountUser) to be set for users who want to access VMs with this service account. [Learn more](#)

Access scopes

☒ Allow default access

☐ Allow full access to all Cloud APIs

☐ Set access for each API

Firewall

Add tags and firewall rules to allow specific network traffic from the Internet

☒ Allow HTTP traffic

☒ Allow HTTPS traffic

Advanced options

Monthly estimate

\$472.43

That's about \$0.65 hourly

Pay for what you use: No upfront costs and per second billing

Item	Monthly estimate
16 vCPU + 64 GB memory	\$470.03
20 GB balanced persistent disk	\$2.40
Sustained use discount	-\$0.00
Total	\$472.43

[Compute Engine pricing](#)
[^ LESS](#)

Centos 7網路配置

按一Create下。

同樣地，建立8個節點，如下圖所示。

VM instances

CREATE INSTANCE

OPERATIONS

HELP ASSISTANT

SHOW INFO PANEL

LEARN

infrastructure. [Learn more](#)

Filter

Status : running

Zone : asia-south1-c

Enter property name or value

☐	Status	Name ↑	Zone	Recommendations	In use by	Internal IP	External IP	Connect
☐	✓	k8-loadbalancer	asia-south1-c			(nic0)	(nic0)	SSH ▾ ⋮
☐	✓	master-1	asia-south1-c			(nic0)	(nic0)	SSH ▾ ⋮
☐	✓	master-2	asia-south1-c			(nic0)	(nic0)	SSH ▾ ⋮
☐	✓	master-3	asia-south1-c			(nic0)	(nic0)	SSH ▾ ⋮
☐	✓	worker-1	asia-south1-c			(nic0)	(nic0)	SSH ▾ ⋮
☐	✓	worker-2	asia-south1-c			(nic0)	(nic0)	SSH ▾ ⋮
☐	✓	worker-3	asia-south1-c			(nic0)	(nic0)	SSH ▾ ⋮
☐	✓	worker-4	asia-south1-c			(nic0)	(nic0)	SSH ▾ ⋮

Google雲平台上的多主機部署

專用IP:

在GCP上，私有IP和公共IP自動分配。

```

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
```

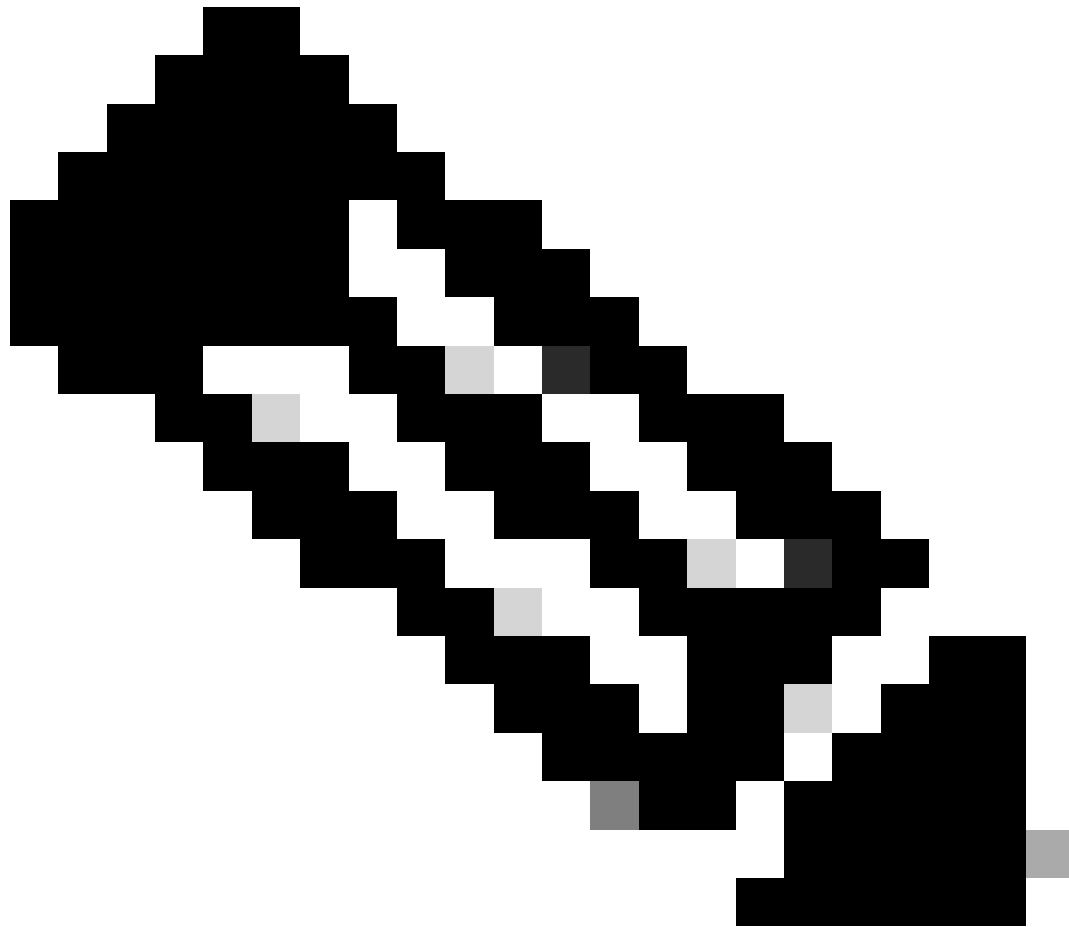
高級概述

在所有節點 (主節點、輔助節點、輔助節點) 上：

1.在Linux伺服器上開啟所需的防火牆埠並配置安全配置。

在多主部署的主節點上：

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



附註：此外，請確保允許GCP防火牆上的埠。

2.配置所需的網路設定（本地DNS、主機名、NTP）。

Bastion伺服器：

- 1.設定HA代理。
- 2.在檔案中新增前端和後端伺服器haproxy.conf 配置。
- 3.重新啟動服haproxy務。

主節點和輔助節點的一般步驟：

- 1.安裝和配置docker。
- 2.安裝和配置Kubernetes。

僅在主節點上：

- 1.初始化新的Kubernetes群集(kubeadm init)。
- 2.安裝Calico網路外掛（專門用於主節點上的核心DNS服務）。
- 3.使用此命令將主節點與主節點連線。

```
kubeadm join
```

```
:6443 --token
```

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

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

4. 使用命令驗證群集信息。`kubectl get nodes`

僅在工作節點上：

1. 使用此命令將工作節點加入主節點。

```
kubeadm join
```

```
:6443 --token
```

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

2. 成功加入後，使用此命令驗證主節點上的群集信息。`kubectl get nodes`

低級配置

網路設定

1.使用以下命令更改未知的root密碼：

```
passwd
```

2.如有必要，使用此命令更改主機名。

```
hostnamectl set-hostname
```

3.配置本地DNS。

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

4.使用此命令啟用NTP服務時序。

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

2.使用此命令驗證狀態。

```
systemctl status chronyd
```

3.使用此命令檢查NTP源。

```
chronyc sources -v
```

堡壘伺服器

步驟1. 檢查更新。

```
sudo yum check-update
```

步驟2. 安裝更新（如果不是最新的）。

```
yum update -y
```

步驟3. 安裝yum實用程式。

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

步驟4. 安裝haproxy程式包。

```
yum install haproxy -y
```

步驟5. 將此配置新增到預設配置/etc/haproxy/haproxy.cfg 中：

```
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 /
```

步驟6. 驗證組態檔，使其看起來像以下命令 `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 /

```

步驟7. 驗證haproxy的狀態：

```
[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/hap
```

```
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 vapidala]#
```

可能的錯誤

1. 在中更改配置後，HAProxy服務處於故障狀態**haproxy.cfg**。例如：

```
[root@k8-loadbalancer vapidala]# 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/hap
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.
```

解析

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

執行：

```
[root@k8-loadbalancer vapidala]# setsebool -P haproxy_connect_any=1
```

```
[root@k8-loadbalancer vapidala]# systemctl restart haproxy
```

```
[root@k8-loadbalancer vapidala]# 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/hap
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]#
```

在主節點和工作節點上安裝Docker

步驟1. 檢查更新。

```
sudo yum check-update
```

步驟2. 安裝更新（如果不是最新的）。

```
yum update -y
```

步驟3. 安裝yum實用程式。

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

步驟4. 安裝Docker。

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

步驟5. 啟用docker。

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

步驟6. 啟動docker服務。

```
sudo systemctl start docker
```

步驟7. 檢查塢站狀態。

```
sudo systemctl status docker
```

輸出：

```
[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]#
```

在主節點和工作節點上安裝Kubernetes

步驟1. 新增Kubernetes儲存庫。

```
cat <
```

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

步驟2. 禁SELinux用。

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

步驟3. 安Kubernetes裝。

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

步驟4. 啟Kubelet用。

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

步驟5. 設**Pod Network**定。

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

步驟6. 令牌生成：

主（控制平面）和工作節點的輸出。

主節點

令牌：您的Kubernetes控制平面已成功初始化！

若要使用集群，請以常規使用者身份運行該程式：

```
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，如果您是根使用者，則可以運行。

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

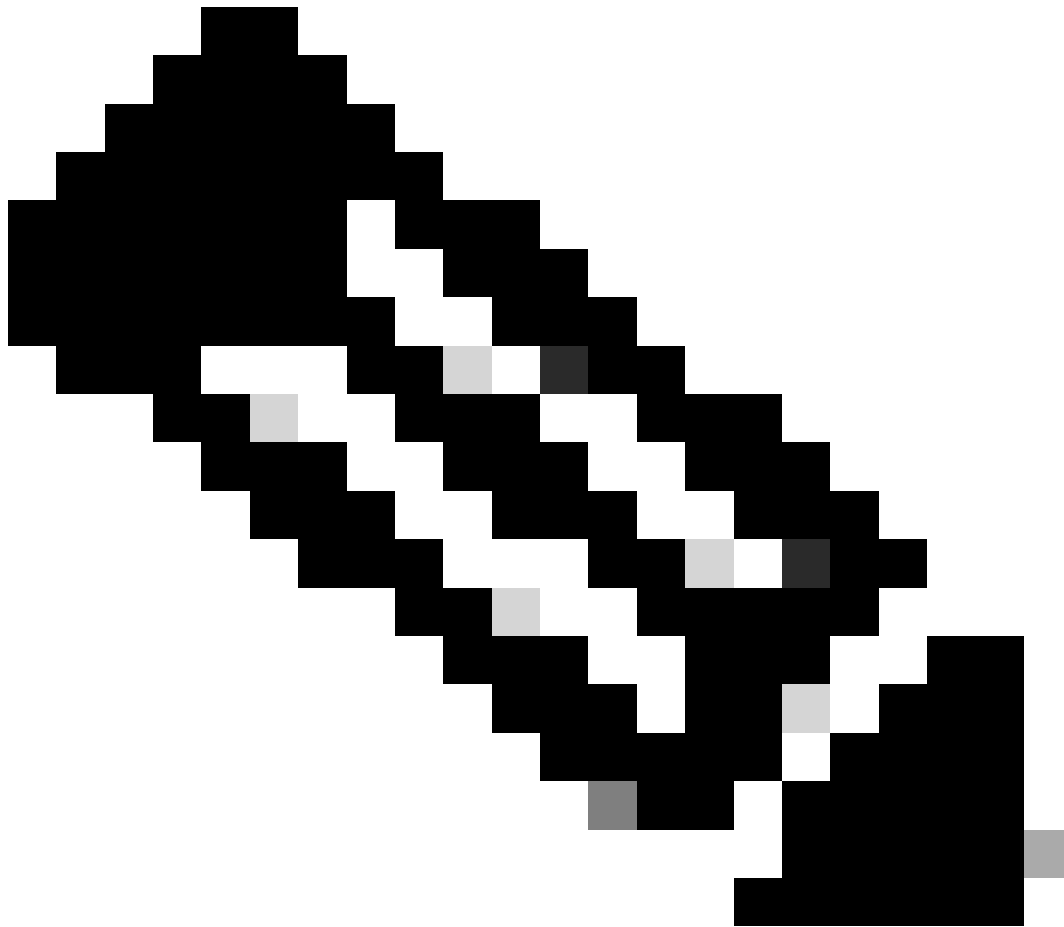
現在，您可以將**Pod**網路部署到集群。

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

現在，您可以加入任意數量的控制平面節點或主節點，這些節點作為根節點在每個節點上運行該命令。

請參閱：[kubeadm加入工作流](#)

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.h07kye1ronuy0mw2 \
--discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
--control-plane --certificate-key 66773b960199ef4530461ef4014e1432066902d4a3dee01669d8622579731
```



附註：請注意，證書金鑰提供對群集敏感資料的訪問，請將其保密！

作為保護措施，上傳的憑證會在兩個小時內刪除；如有必要，可以使用它們。

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

然後，您可以加入任意數量的工作節點，並在每個工作節點上作為根節點運行此任務。

```
kubeadm join 10.160.0.19:6443 --token 5fv6ce.h07kye1ronuy0mw2 \  
--discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
```

步驟7. 驗證核心DNS服務。

```
[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-ttttpq	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]#

步驟8. 檢驗主節點的狀態。

```
[root@kube-master1 vapadala]# kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
kube-master1	Ready	control-plane	11m	v1.25.3

要加入多個主節點，在主節點上執行此加入命令。

```
kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kylronuy0mw2 \
--discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61
--control-plane --certificate-key 66773b960199ef4530461ef4014e1432066902d4a3dee01669d8622579731
```

生成令牌時可能遇到的錯誤

錯誤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 Firewall]: firewall is active, please ensure ports [6443 10250] are open or your cluster
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 m
[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
```

錯誤CRI解決方案

步驟1. 移除config.toml檔案，然後重新啟動containerd。

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

步驟2. 安裝容器：

使用以下命令從官方Docker儲存庫安裝containerd.io軟體包。

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

步驟3. 配置容器：

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

步驟4. 重新啟動容器：

```
systemctl restart containerd
```

錯誤檔案內容 — proc-sys-net-ipv4-ip_forward

```
[ERROR FileContent--proc-sys-net-ipv4-ip_forward]: /proc/sys/net/ipv4/ip_forward contents are not set to 1
```

錯誤檔案內容 — proc-sys-net-ipv4-ip_forward解析度

將ip_forward值設定為1。

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

核心DNS服務未運行

```
[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   AGE
kube-system    coredns-565d847f94-cwgj8           0/1     Pending   0           12m
```

```

kube-system    coredns-565d847f94-ttttpq          0/1          Pending    0          12m
kube-system    etcd-kube-master1                  1/1          Running    0          12m
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          12m
kube-system    kube-scheduler-kube-master1        1/1          Running    0          12m
[root@kube-master1 vapadala]#

```

解析

核心DNS處於掛起狀態，這表示存在網路問題。因此，需要安裝Calico。

參考：[Calico](#)

執行以下兩個命令：

```

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

```

工作人員節點

從主節點獲取令牌時在Worker Node上：

步驟1. 啟用kubelet服務。

```

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

```

步驟2. 使用此join命令將所有worker節點與主節點連線。

```

kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kye1ronuy0mw2 \
--discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61

```

步驟3. 如果遇到與檔案或埠相關的錯誤，請使用此命令重置kubeadm。

```
kubeadm reset
```

重置後，從主節點輸入令牌。

```

kubeadm join 10.160.x.19:6443 --token 5fv6ce.h07kye1ronuy0mw2 \
--discovery-token-ca-cert-hash sha256:b5509b6fe784561f3435bf6b909dc8877e567c70921b21e35c464eb61

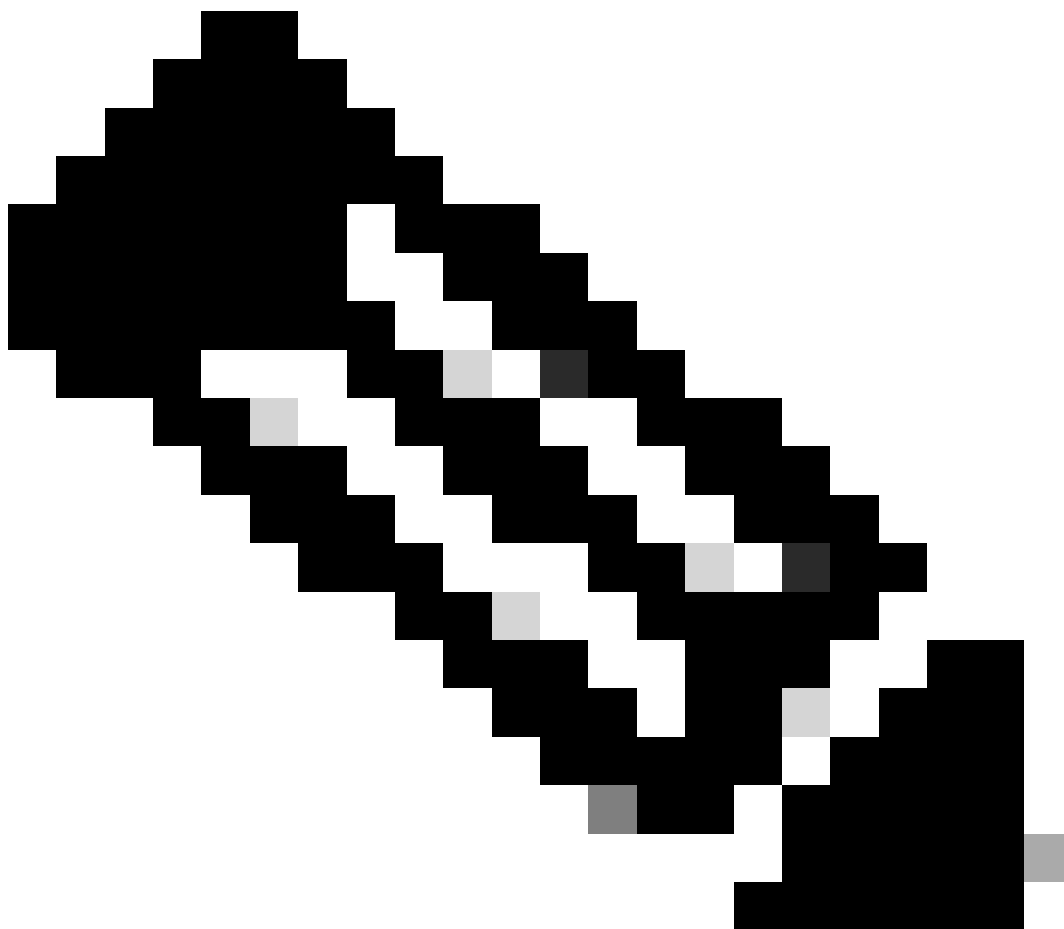
```

最終輸出

現在已形成群集，請使用 `kubect1 get nodes` 命令進行驗證。

```
node/worker 4 labeled
[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* 群集輸出



附註：在形成集群時，只配置來自主節點的控制。未將堡壘主機配置為集中式伺服器以監視群集中的所有庫貝。

關於此翻譯

思科已使用電腦和人工技術翻譯本文件，讓全世界的使用者能夠以自己的語言理解支援內容。請注意，即使是最佳機器翻譯，也不如專業譯者翻譯的內容準確。Cisco Systems, Inc. 對這些翻譯的準確度概不負責，並建議一律查看原始英文文件（提供連結）。