

El contingut d'aquest document, és facilitar o guiar en el desplegament d'un Clúster de Kubernetes amb alta disponibilitat (Kubernetes HA Cluster). Semblant a aquest contingut, el podem trobar per internet, però així és com ho faig jo i el resultat és 100% productiu. Per fer-ho, no utilitzarem cap eina de tercers, de fet quan provo un producte nou, m'agrada fer-ho amb les eines que aquest ofereix d'inici. Posteriorment, esbrino si hi ha algun producte que pot afegir alguna cosa interessant amb un cost/resultat adequat (el cost es pot mesurar amb dedicació i preu, per exemple).

Si ens llegim la documentació oficial, veurem que recomana com a mínim 3 màsters. Per explicar com fer el desplegament, utilitzarem 2 màsters i 2 nodes. Amb això tenim suficient per disposar de HA. Afegir més màsters i nodes, si s'entén el que explico, és una tasca molt automàtica. Els nodes seran Linux (i els màsters també, ja que només poden ser desplegats sobre Linux).

El primer que farem és una mica d'arquitectura de xarxa, definirem el nom de les 4 màquines, les IP's i un nom i **IP virtual** que serà la IP de administració del clúster. Per entendre-ho, **és la IP que agafa el màster que està disponible amb prioritat més alta.**

Servidor	IP
cm1 (màster) 2cpu + 2Gb RAM	192.168.21.11
cm2 (màster) 2cpu + 2Gb RAM	192.168.21.12
cn1 (node) 2cpu + 4Gb RAM	192.168.21.13
cn2 (node) 2cpu + 4Gb RAM	192.168.21.14
vip-cm (ip virtual)	192.168.21.10

** Podem posar el segment de xarxa que més ens convingui.

** El dimensionament dels nodes (Cpu i Ram) ha de ser l'adequat a les necessitats. Si treballem amb virtualització, ho podem anar escalant segons necessitats.

PAS1 (**tots els servidors**).

- Instal·lar SO. Jo utilitzo **Ubuntu Server**. L'últim clúster que he desplegat **Ubuntu Server 20.04 LTS**. A part del SO, només cal instal·lar el servidor **OpenSSH Server** per poder treballar en remot (no és indispensable). Creem un usuari no root amb permisos sudo per fer de tot (la instal·lació ja ho fa). A cada servidor li posarem la seva IP estàtica, la porta d'enllaç, la xarxa i els DNS, que correspongui. Aquesta instal·lació la farem com si no tinguéssim DNS a la xara local (només els de internet)
- Tot seguit a cada servidor li posarem el nom:
 - **hostnamectl set-hostname "cm1"** (cm2,cn1,cn2)
- Editarem /etc/hosts i posarem:
 - **sudo nano /etc/hosts**

```
...
# IP virtual
192.168.21.10 vip-cm
# Masters
192.168.21.11 cm1
192.168.21.12 cm2
# Nodes o Workers
192.168.21.13 cn1
192.168.21.14 cn2
...
```

- Ara instal·larem el sistema de contenidors que vulguem. Jo personalment utilitzo Docker, però podríem fer servir un altre.
- **sudo apt-get update**
- **sudo apt-get install ca-certificates curl gnupg lsb-release**
- **curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg**
- **sudo echo "deb [arch=\$(dpkg --print-architecture) signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \$(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null**
- **sudo apt-get update**
- **sudo apt-get install docker-ce docker-ce-cli containerd.io**
- **** Posem systemd com a cgroup driver:**
 - **sudo nano /etc/docker/daemon.json**

```
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
```

- **sudo systemctl enable docker**
- **sudo systemctl daemon-reload**
- **sudo systemctl restart docker**

- Ara instal·larem Kubernetes. Abans però habilitarem el trànsit bridge perquè sigui visible per iptables.

```
sudo cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf
br_netfilter
EOF
sudo cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

- Des habilitem swap : **sudo swapoff -a** (comentem també la línia de swap a **/etc/fstab**)
- Instal·lem
 - **sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg https://packages.cloud.google.com/apt/doc/apt-key.gpg**
 - **sudo echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list**
 - **sudo apt-get update**
 - **sudo apt-get install -y kubelet kubeadm kubectl**

PAS2 (màsters).

Un cop tenim instal·lats els serveis Docker i Kubernetes, començarem a configurar el clúster per HA. Per fer-ho utilitzarem dos serveis més. El primer ens serveix perquè un servidor dels màsters allotgi la IP virtual i sigui el que respon a aquesta, i si aquest "cau" la IP l'agafi un altre màster (el següent amb més prioritat disponible). El segon servei, ens dona el servei de Reverse Proxy, que utilitzarem per distribuir les peticions al panel de control de Kubernetes, a tots els màsters, amb la tècnica round-robin. Aquest serveis són **KeepAlived i HAProxy**.

- Instal·lem els dos serveis (als dos màsters només).
 - **sudo apt-get install haproxy keepalived -y**
- Ara creem un script que comprova que el API Server (Api d'administració) de Kubernetes funcioni (port 6443).
 - **sudo nano /etc/keepalived/check_apiserver.sh**

```
#!/bin/sh
APISERVER_VIP=192.168.21.10
```

```
APISERVER_DEST_PORT=6443
```

```
errorExit() {
    echo "*** $" 1>&2
    exit 1
}

curl --silent --max-time 2 --insecure https://localhost:${APISERVER_DEST_PORT}/
-o /dev/null || errorExit "Error GET https://localhost:${APISERVER_DEST_PORT}/"
if ip addr | grep -q ${APISERVER_VIP}; then
    curl --silent --max-time 2 --insecure
https://${APISERVER_VIP}:${APISERVER_DEST_PORT}/ -o /dev/null || errorExit
"Error GET https://${APISERVER_VIP}:${APISERVER_DEST_PORT}/"
fi
```

- **sudo chmod +x /etc/keepalived/check_apiserver.sh**
- Ara configurem la alta disponibilitat a nivell de xarxa
 - **sudo cp /etc/keepalived/keepalived.conf /etc/keepalived/keepalived.conf-org**
 - **sudo sh -c '> /etc/keepalived/keepalived.conf'**
 - **sudo nano /etc/keepalived/keepalived.conf**

```
global_defs {
    router_id LVS_DEVEL
}

vrrp_script check_apiserver {
    script "/etc/keepalived/check_apiserver.sh"
    interval 3
    weight -2
    fall 10
    rise 2
}

vrrp_instance VI_1 {
```

```
state MASTER
interface ens160
virtual_router_id 151
priority 255
authentication {
    auth_type PASS
    auth_pass 12345678
}
virtual_ipaddress {
    192.168.21.10/24
}
track_script {
    check_apiserver
}
}
```

- He marcat en groc les coses importants.
 - Lo primer és posar bé la tarja de xarxa (depèn del servidor) → ens160
 - El priority ha de ser diferent en cada màster. Ex: cm1=255 i cm2=252.
 - El password com a màxim pot tenir 8 caràcters. 12345678
 - I al virtual_ipaddress cal posar la IP virtual i la màscara de xarxa. 192.168.21.10/24
- Configurem el HAProxy.
 - `sudo cp /etc/haproxy/haproxy.cfg /etc/haproxy/haproxy.cfg-org`
 - `sudo nano /etc/haproxy/haproxy.cfg`
 - Esborren totes les línies després de **default settings** i afegim

```
#-----
# Reverse Proxy
#-----

frontend apiserver
    bind *:8443
    mode tcp
    option tcplog
```

```
default_backend apiserver
#-----
# round robin balancing for apiserver
#-----
backend apiserver
    option httpchk GET /healthz
    http-check expect status 200
    mode tcp
    option ssl-hello-chk
    balance roundrobin
        server cm1 192.168.21.11:6443 check
        #server cm2 192.168.21.12:6443 check
```

- Suposant que hem començat pel master cm1, comentem la línia del master cm2. Quan ambdós siguin masters de Kubernetes, descomentem i reiniciem el servei haproxy (**molt important!!!**).
- **sudo groupadd -r keepalived_script**
- **sudo systemctl enable keepalived -now**
- **sudo systemctl enable haproxy -now**
- **I ara si, engegarem el primer màster**
 - **sudo kubeadm init --control-plane-endpoint "vip-cm:8443" --pod-network-cidr=10.244.0.0/22 --upload-certs**
 - La xarxa podem canviar-la (<https://kb.wisc.edu/ns/page.php?id=3493>). El paràmetre **-upload-certs** ens ajuda en el desplegament. Sinó hem de desplegar els certificats de forma manual.
 - **Quan finalitza, si tot ha anat bé**

Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a regular user:

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

Alternatively, if you are the root user, you can run:

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

You should now deploy a pod network to the cluster.

Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:

<https://kubernetes.io/docs/concepts/cluster-administration/addons/>

You can now join any number of control-plane nodes by copying certificate authorities

and service account keys on each node and then running the following as root:

Then you can join any number of worker nodes by running the following on each as root:

```
kubeadm join vip-cm:8443 --token x53s6g.11111nh4hg6s94qi --discovery-token-ca-  
cert-hash sha256:f7f863cab0e74f.....0f12d2bac206b7df451396ba --  
control-plane --certificate-key  
a9e0e92aa2e23574.....861614365e1904a5431bdd2
```

```
kubeadm join vip-cm:8443 --token x53s6g.11111nh4hg6s94qi --discovery-token-ca-  
cert-hash sha256:f7f863cab0e74f.....0f12d2bac206b7df451396ba
```

- Tot seguit a cm1 (on estem) executem:
 - `mkdir -p $HOME/.kube`
 - `sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config`
 - `sudo chown $(id -u):$(id -g) $HOME/.kube/config`
- Ara hem de configurar un controlador de xarxa pel Clúster. Jo he fet servir Calico.
 - `curl https://docs.projectcalico.org/manifests/calico.yaml -O`
 - `kubectl apply -f calico.yaml`
- Anem a cm2 (als masters que volguem configurar) i executem el primer `kubeadm join --control-plane --certificate-key`
- També executem
 - `mkdir -p $HOME/.kube`
 - `sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config`
 - `sudo chown $(id -u):$(id -g) $HOME/.kube/config`
- En els màsters instal·lem també el controlador de xarxa (en aquest calico). En les referències trobarem més.
- Cada cop que instal·lem un màster o un node, anem a qualsevol dels masters operatius i executem per veure si estan operatius (no és instantani):
 - `sudo kubectl get nodes`

PAS3 (workers).

- I ara afegim nodes (workers). Per fer-ho només cal executar el segon join
 - `kubeadm join vip-cm:8443 --token x53s6g.11111nh4hg6s94qi --discovery-token-ca-cert-hash sha256:f7f863cab0e74f.....0f12d2bac206b7df451396ba`
 - Però no aquest sinó el que us surti en pantalla del primer màster.

I anem afegint nodes i màsters, tants com vulguem. Si llegiu veure-ho que poden ser molts dins un sol clúster, amb nodes i màsters distribuïts en localitzacions geogràfiques diferents. Però per fer tota això cal lectura i molta pràctica. Amb aquest clúster però, podem començar a jugar i testejar moltes coses, volums, secrets, namespaces, deployments en general. Ah!! No oblidar posar cada màster a la configuració del HAProxy **de cada màster**.

```
....  
balance      roundrobin  
  
    server cm1 192.168.21.11:6443 check  
  
    server cm2 192.168.21.12:6443 check  
  
....
```

I...

- `sudo kubectl get nodes`

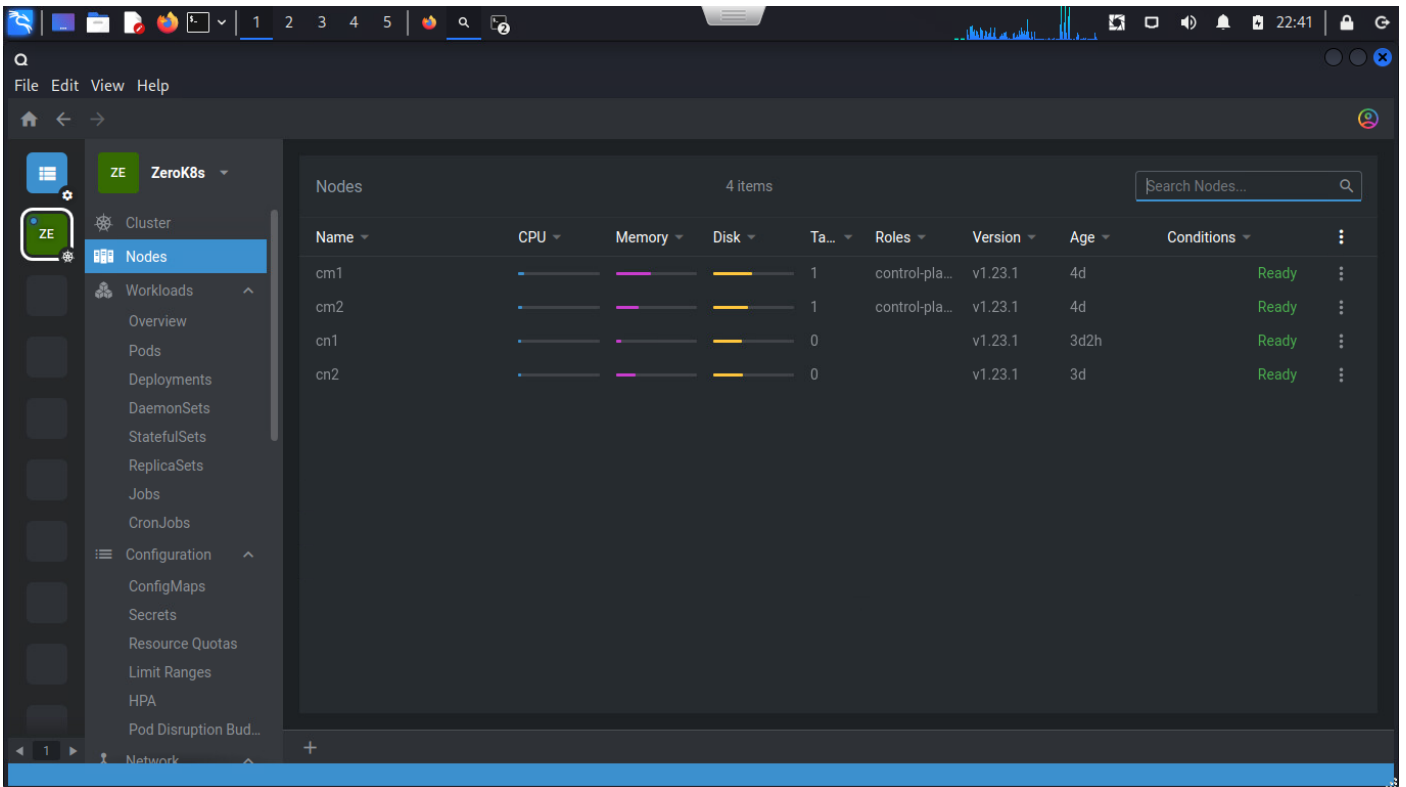
```
zero@cm1:~$ kubectl get nodes  
NAME STATUS ROLES    AGE  VERSION  
cm1  Ready  control-plane,master  3d23h  v1.23.1  
cm2  Ready  control-plane,master  3d23h  v1.23.1  
cn1  Ready  <none>              3d1h  v1.23.1  
cn2  Ready  <none>              2d23h  v1.23.1
```

Ara, si voleu podeu instal·lar un client per gestionar el clúster. Jo personalment utilitzo el Lens, però n'hi ha d'altres. És un client lleuger, gratuït i amb un RoadMap de creixement molt interessant. Hi ha un munt d'extensions. Però no és l'únic.

I ara, a fer **"deployments"**.

kubectl apply -f Deployment.yml

Desplegament d'un Clúster de Kubernetes amb alta disponibilitat



The screenshot shows the Kubernetes dashboard interface. The left sidebar contains a navigation menu with the following items: Cluster, Nodes, Workloads, Overview, Pods, Deployments, DaemonSets, StatefulSets, ReplicaSets, Jobs, CronJobs, Configuration, ConfigMaps, Secrets, Resource Quotas, Limit Ranges, HPA, and Pod Disruption Bud... The 'Nodes' page is selected, displaying a table with 4 items. The table columns are: Name, CPU, Memory, Disk, Ta..., Roles, Version, Age, and Conditions. The data rows are:

Name	CPU	Memory	Disk	Ta...	Roles	Version	Age	Conditions
cm1	1				control-pla...	v1.23.1	4d	Ready
cm2	1				control-pla...	v1.23.1	4d	Ready
cn1	0					v1.23.1	3d2h	Ready
cn2	0					v1.23.1	3d	Ready



The screenshot shows the Kubernetes dashboard interface. The left sidebar contains a navigation menu with the following items: Persistent Volume..., Persistent Volumes, Storage Classes, Namespaces, Events, Apps, Access Control, Service Accounts, Cluster Roles, Roles, Cluster Role Bindin..., Role Bindings, Pod Security Polici..., Custom Reso..., Definitions, and crd.projectcal... The 'Resource Map' page is selected, displaying a graph of resources across different namespaces. The graph shows various components like kube-system, kube-proxy, calico, and calico-kube-controllers. The left sidebar also shows the 'Resource Map' page selected under the 'ZeroK8s' cluster.

Referències

<https://ubuntu.com/download/server>

<https://docs.docker.com/engine/install/#server>

<https://docs.docker.com/engine/install/ubuntu/>

<https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>

<https://kubernetes.io/docs/concepts/cluster-administration/networking/#how-to-implement-the-kubernetes-networking-model>

<https://k8slens.dev/>