Kurulum

ilk önce loadbalancer vm'inde paketleri güncelleyelim. Ardından haproxy paketini kuralım.

> Kurulum yaparken her makinede `sudo su` ile birlikte **root** kullanıcı olmamız gerekmektedir. SSH, konfigürasyon ve kontroller root kullanıcı ile yapılacaktır.

```
apt update
apt install -y haproxy
```

Şimdi loadbalancer'imizin konfigürasyonunu yapabiliriz. Master ve worker node'ların ip'lerini burada kullanacağız. Şuan kullanacağımız ip konfigürasyonları aşağıdaki tabloda verilmiştir.

	IP
Master-1	10.20.0.27
Master-2	10.20.0.215
Master-3	10.20.0.214
Worker-1	10.20.0.119
Worker-2	10.20.0.146
Worker-3	10.20.0.206
LoadBalancer	10.20.0.251

Aşağıdaki kısımda ETCD, Kubernetes ve Ingress için konfigürasyonlar mevcuttur. Bizim deploy edeceğimiz cluster'de bu şekilde devam edeceğiz. Bu kısmı `/etc/haproxy/haproxy.cfg` konfigürasyon dosyasının en altına ekleyeceğiz. Dosyayı kaydedip çıkabiliriz.

```
# ==============================
# Kubernetes API (6443)
# ==============================
frontend k8s_api
    bind 10.20.0.251:6443
    mode tcp
    default_backend k8s_masters

backend k8s_masters
    mode tcp
    balance roundrobin
    option tcp-check
```

```
    default-server inter 3s fall 3 rise 2
    server master1 10.20.0.27:6443 check
    server master2 10.20.0.215:6443 check
    server master3 10.20.0.214:6443 check


# ============================
# RKE2 Server Join (9345)
# ============================
frontend rke2_server
    bind 10.20.0.251:9345
    mode tcp
    default_backend rke2_masters

backend rke2_masters
    mode tcp
    balance roundrobin
    option tcp-check
    default-server inter 3s fall 3 rise 2
    server master1 10.20.0.27:9345 check
    server master2 10.20.0.215:9345 check
    server master3 10.20.0.214:9345 check


# ============================
# Ingress HTTP/HTTPS (Rancher UI)
# SSL Passthrough to Workers
# ============================
frontend ingress_http
    bind *:80
    mode tcp
    default_backend ingress_http

frontend ingress_https
    bind *:443
    mode tcp
    default_backend ingress_https

backend ingress_http
    mode tcp
    balance roundrobin
    option tcp-check
    server worker1 10.20.0.119:80 check
    server worker2 10.20.0.146:80 check
    server worker3 10.20.0.206:80 check
```

```
backend ingress_https
    mode tcp
    balance roundrobin
    option tcp-check
    server worker1 10.20.0.119:443 check
    server worker2 10.20.0.146:443 check
    server worker3 10.20.0.206:443 check
```

Şimdi haproxy servisini `systemctl restart haproxy` diyerek yeniden başlatalım ve rke2 node kurulumlarına başlayalım.

Bastion Loadbalancer sanal makinemizden sırasıyla bütün node'lara SSH atarak `ssh ubuntu@<PRIVATE_IP>` aşağıdaki şekilde swap memory ve kernel parametrelerine yönlendirmeyi etkinleştirelim.

> Her bir node'a bağlanarak aşağıdaki komutları girmeyi unutmayınız.

```
swapoff -a
sed -i '/ swap / s/^/#/' /etc/fstab
modprobe br_netfilter
sysctl -w net.ipv4.ip_forward=1
```

Master 1 Cluster Bootstrap Yapılması

Bütün işlemler tamamlandıktan sonra master 1 sunucumuza bağlanıp ardından aşağıdaki şekilde RKE2 nin kurulum konfigürasyon dosyasını oluşturalım.

```
mkdir -p /etc/rancher/rke2
vim /etc/rancher/rke2/config.yaml
```

Dosyanın içerisine aşağıdaki şekilde konfigürasyon dosyamızı yazıp kaydedip çıkalım.

> İlk master LB arkasında başlatılmayacak düzgün bootstrap için server kısmı yorum satırlı kalsın.

```
token: super-secret-token

#server: https://10.20.0.251:6443

tls-san:
  - 10.20.0.251   # HAProxy IP
```

```
  - rancher.dc01.local

node-name: dc01-master-1

node-taint:
  - "node-role.kubernetes.io/control-plane=true:NoSchedule"
```

Sonrasında kurulumu gerçekleştirebiliriz.

```
curl -sfL https://get.rke2.io | sh -
systemctl enable rke2-server
systemctl start rke2-server
```

> Servis durumunu `journalctl -u rke2-server -f` ile takip edebiliriz.

Ayağa kalktığını anlamak için **rke2-server** servisinin log'larına bakabilirsiniz. Hızlı bir şekilde node ayakta mı test etmek için. Aşağıdaki şekilde bakınız.

```
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml
/var/lib/rancher/rke2/bin/kubectl get nodes
```

Master 2 ve 3'ün HA Katılımı

Sırada ise master2 ve master3 var bu kısımlarda ise artık cluster bootstrap olduğu için server verip etcd peer'leri discover ederek sertifikalarını paylaşabilir. Konfigürasyon yolunu ve dosyasını yine aynı şekilde oluşturalım.

```
mkdir -p /etc/rancher/rke2
vim /etc/rancher/rke2/config.yaml
```

Dosya içeriğine ise bu şekilde yazıp kaydediniz.

```
server: https://10.20.0.251:9345
token: super-secret-token

node-name: dc01-master-2

tls-san:
  - 10.20.0.251
  - rancher.dc01.local
```

```
write-kubeconfig-mode: "0644"
```

```
curl -sfL https://get.rke2.io | sh -
systemctl enable rke2-server
systemctl start rke2-server
```

```
export KUBECONFIG=/etc/rancher/rke2/rke2.yaml
/var/lib/rancher/rke2/bin/kubectl get nodes
```

Yukarıda yapılan işlemleri master 3 içinde tekrarlayalım.

> **node-name** alanında ise dc01-master-3 olarak değiştirmeyi unutmayınız.

Worker Node'ların Cluster'a Katılması

Burada diğer kısımlara göre kurulumda birkaç konfigürasyon farkı mevcuttur. Sırasıyla aşağıdaki komutları diğer worker-node'lere bağlanarak çalıştıralım. İlk önce konfigürasyon dosyalarımızı oluşturalım.

```
mkdir -p /etc/rancher/rke2
vim /etc/rancher/rke2/config.yaml
```

İçeriğini ise aşağıdaki şekilde yapıp kaydedelim.

```
server: https://10.20.0.251:9345
token: super-secret-token
node-name: dc01-worker-1
```

Ardından kuruluma başlayabiliriz. Aşağıdaki şekilde node'mizi agent olarak env verip worker olarak cluster'a join işlemini tamamlayabiliriz.

```
curl -sfL https://get.rke2.io | INSTALL_RKE2_TYPE="agent" sh -
systemctl enable rke2-agent
systemctl start rke2-agent
```

BastionLB üzerinden Kubernetes Cluster'a Erişimin Sağlanması

LoadBalancer sanal makinemize gelip kendi tarafımızda kubeconfig dosyasını kopyalayacağız. Bu şekilde server'a erişimimiz olacak.

`scp ubuntu@10.20.0.215:/etc/rancher/rke2/rke2.yaml /root/kubeconfig`

Ardından LoadBalancer ip'sini kubeconfig içerisinde server kısmına vererek KUBECONFIG dosyamızı tamamlayalım

`sed -i 's/127.0.0.1/10.20.0.251/' /root/kubeconfig`

Erişim için loadbalancer sanal makinemiz üzerinde kubectl binary'si mevcut değil bunun için en güncel ve stabil versiyonu çekerek makinemize kubectl kurulumunu gerçekleştirelim.

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

`kubectl get node` diyerek cluster'e erişimimiz sağlandı mı kontrol edelim. Eğer erişimimizi teyit edip doğrulayabiliyorsak şuan tam anlamıyla çalışır bir rke2 kubernetes cluster'ine sahibiz demektir. Fakat kubectl ile birlikte kubernetes cluster'i yönetmek çoğu zaman yorucu gelebilir. Bundan dolayı Cluster'imize Rancher UI kurarak tam anlamıyla Cluster kontrolünü sağlayabiliriz. Bunun için kubernetes tarafında bize yardımcı olacak tool Helm'i kullanacağız. Helm ,Kubernetes üzerinde uygulamaları kolayca yönetmenizi yarayan bir araç olarak karşımıza çıkıyor. Helm ile kolayca deploy edebilir,upgrade edebilir,sürümleri kontrol edebiliriz. Helm'i kolay bir şekilde kurabiliriz.

```
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-4
chmod 700 get_helm.sh
./get_helm.sh
```

RancherUI, CertManager kullandığı için bazı CRD'lere ihtiyaç duyuyor bundan dolayı ilk önce CertManager kurulumunu yapalım.

`kubectl apply -f https://github.com/cert-manager/cert-manager/releases/download/v1.14.4/cert-manager.yaml`

Ardından RancherUI kurulumuna başlayabiliriz.

```
helm repo add rancher-latest https://releases.rancher.com/server-
charts/latest
helm repo update
```

```
kubectl create namespace cattle-system

helm install rancher rancher-latest/rancher --namespace cattle-system --set
hostname=rancher.dc01.com
```

Kendi lokal bilgisayarımıza geri döndükten sonra `hosts` dosyasını düzenleyelim buraya girdi olarak `<BASTION_LB_IP> rancher.dc01.local` olarak ekleyelim sonrasında tarayıcı üzerinden bu adrese gidelim. İlk defa kurulum olduğu için şifre'yi cluster'dan almak gerekecek bu noktada `kubectl get secret --namespace cattle-system bootstrap-secret -o go-template='{{.data.bootstrapPassword|base64decode}}{{ "\n" }}'` ile birlikte bu şifreyi alıp sonrasında şifremizi değiştirip UI kısmına erişebiliriz.