Crear un cluster

Crear un cluster a mano

Para este caso creamos 3 MV diferentes:

- Una hará de nodo máster, que orquestará todo el sistema.
- Las otras 2 harán de nodos worker.

Las 3 MV serán Ubuntu 16.04 para las pruebas.

Paso 1: Agregar repositorio k8s

k8s no viene por defecto en los repositorios de nuestro SO, por lo que deberemos agregar el enlace al repositorio de k8s para que podamos instalarlo a través del gestor **apt**

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"
```

Paso 2: Instalamos Docker y las herramientas de k8s

Deberemos actualizar la lista de paquets para que nos coja los del repositorio nuevo que acabamos de agregar.

```
apt update
apt install -y docker.io
apt install -y kubelet kubeadm kubectl
```

Paso 3: Agregamos permisos

```
usermod -aG docker vagrant
systemctl enable docker
```

Paso 4: Inicializamos el nodo master (tardará algunos minutos)

Iniciamos, pasándole la IP de la interfaz correcta (importantísimo)

```
kubeadm init --apiserver-advertise-address 192.168.205.10
```

Una vez iniciado el cluster, nos devolverá un comando para ejecutar en los nodos worker que queramos agregar al cluster, tal que así:

```
kubeadm join 192.168.205.10:6443
--token gsvzsq.3k76aiuuj12dk1fm
--discovery-token-ca-cert-hash
sha256:8f478c0f3440f8cf6b108f4a947b8c5d3a6185419c5d23033f378e5d9fc3cf66
```

Si la liamos, podemos reiniciar en el maestro y volver a iniciar, reseteando el cluster:

```
kubeadm reset
```

Después de esto, volveríamos a hacer el init en el master y hacer join en los workers.

Paso 5: Comandos posteriores (solo nodo master)

Para empezar a utilizar mi cluster me va apedir ejecutar una serie de comandos

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

Paso 6: Inicializar la red de cluster (nodo master)

Me recomienda desplegar una red de pods en el cluster:

```
kubectl apply -n kube-system -f \
"https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d
'\n')"
```

Tardará un poquito en levantar los pods, unos 3 minutos. Podemos verlo utilizando:

```
kubectl get pods --all-namespaces
```

Al final tendremos un nodo desplegado como master:

```
kubecetl get nodes
```

Deberíamos ver el node1 con estado Ready, si todo va bien.

Paso 7: Agregar nodos a la red (nodos worker)

En cada uno de los nodos worker que queramos agregar al cluster:

```
kubeadm join 192.168.205.10:6443
--token gsvzsq.3k76aiuuj12dk1fm
--discovery-token-ca-cert-hash
sha256:8f478c0f3440f8cf6b108f4a947b8c5d3a6185419c5d23033f378e5d9fc3cf66
```

Si todo va bién se agregará al cluster y contestará algo como:

```
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

Nos vamos al master y comprobamos la lista de nodos:

```
kubectl --insecure-skip-tls-verify get nodes
```

Aquí me ha dado un error debido a falta de certificados