Actividad 8 Kubernetes (v. 2111028) Pablo González Nalda
Depto. de *Lenguajes y Sistemas Informáticos* lsi.vc.ehu.eus/pablogn
Universidad del País Vasco Euskal Herriko Unibertsitatea
GASTEIZKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE VITORIA-GASTEIZ

Kubernetes

En esta práctica daremos los primeros pasos en *Kubernetes* (K8S, siendo el 8 las letras que se omiten) y observaremos sus principales características.

8.1. Instalación de `minikube`, (K8S mono-nodo)

`minikube` es una instalación flexible de **K8S** de un único nodo. Para ello tenemos que instalar `kubectl` por separado por uno de los dos posibles caminos, por `wget` o por `apt`

Por `wget` sin usar privilegios de `root`:

```
# https://www.gitbook.com/book/ramitsurana/awesome-kubernetes/details
# https://kubernetes.io/docs/tasks/tools/install-kubectl/
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(curl -s https://storage.
    googleapis.com/kubernetes-release/release/stable.txt)/bin/linux/amd64/kubectl
chmod +x kubectl
alias kubectl='./kubectl'
```

Por `apt` con usar privilegios de `root` a través de `sudo`:

```
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
sudo bash -c "echo >/etc/apt/sources.list.d/kubernetes.list \
    'deb http://apt.kubernetes.io/ kubernetes-xenial main'"
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubernetes-cni
```

Y ahora instalamos y arrancamos `minikube`:

```
# https://github.com/kubernetes/minikube
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
chmod +x minikube
alias minikube='./minikube'         # alternativa 1 crear un alias
export PATH=$(pwd):$PATH            # alternativa 2 incorporar el dir actual al PATH
sudo cp minikube /usr/local/bin/    # alternativa 3 ponerlo en un PATH del sistema
source <(kubectl completion bash)

# https://www.linux.com/learn/getting-started-kubernetes-easy-minikube
# https://minikube.sigs.k8s.io/docs/drivers/
minikube start --memory=4096 --cpus=4    # con KVM
minikube start --driver=docker           # con Docker
minikube status
```

Accedemos en local al entorno web de gestión de `minikube`:

```
minikube dashboard --url
minikube dashboard #
```

Se puede usar `minikube` como se hace con Vagrant, y realizar las operaciones de conectar, parar y terminar el K8S creado con `minikube`.

```
minikube ssh
minikube stop
minikube delete
```

8.2. Instalación de Kubernetes con `kubeadm init`

Para la instalación hay muchas variantes. Se puede hacer de muchas formas, algunas serían las siguientes (ver `esta web`):

- En Windows se puede activar el soporte a K8S en las opciones de Docker:

  https://enmilocalfunciona.io/instalando-y-probando-kubernetes-windows-10/

- En GNU/Linux:

  Primero instalamos Docker, en el nodo maestro y en los esclavos:

```
sudo apt install docker.io
sudo groupadd docker
sudo usermod -aG docker $USER
```

  La forma estándar es usar `kubeadm init` y `kubeadm join` (ver [1])

  Gracias a la documentación del trabajo de un alumno de matrícula de honor, hay muchos pasos que se me han facilitado.

  Tenemos que eliminar el *swap* tanto en el *máster* como en los *workers* (llamados `as1` el máster y `as2` el *worker*):

```
sudo su
swapon -s # miramos si hay sistema de paginación en disco
swapoff -a
vi /etc/fstab # y comentamos con # la línea de swap
```

  Si es en Ubuntu hay que cambiar el `cgroup` de `docker` modificando la línea de arranque (que contiene la palabra `ExecStart`) y reiniciando[2]

```
vi /lib/systemd/system/docker.service

ExecStart=/usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --exec-opt
    native.cgroupdriver=systemd

systemctl daemon-reload
systemctl restart docker
systemctl restart kubelet # si ya está instalado kubelet
systemctl status kubelet
```

  Instalamos como en el caso de `minikube`:

```
sudo su
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
bash -c "echo >/etc/apt/sources.list.d/kubernetes.list \
'deb http://apt.kubernetes.io/ kubernetes-xenial main'"
apt-get update
apt-get install -y kubelet kubeadm kubernetes-cni
```

  Limpiamos si ha habido arranques fallidos o ejecuciones anteriores:

```
kubeadm reset # si ya se ha arrancado antes
rm -rf /etc/kubernetes
rm -rf /var/lib/etcd
```

[1] https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/ y https://devopscube.com/setup-kubernetes-cluster-kubeadm/

[2] https://stackoverflow.com/questions/69085180/how-to-install-kubernetes-cluster-on-azure-ubuntu-virtual-machine-20-04-lts/69128645#69128645.

Arrancamos el *máster*, y guardamos los token que nos da para unir los *workers*:

```
NODENAME=$(hostname -s)
IPADDR="10.10.10.101"
kubeadm init --apiserver-advertise-address=$IPADDR  --apiserver-cert-extra-sans=$IPADDR \
     --pod-network-cidr=192.168.0.0/16 --node-name $NODENAME
export KUBECONFIG=/etc/kubernetes/admin.conf
```

En el *máster* desplegamos Weave, una red para los contenedores del clúster:

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

Y en el segundo nodo, el *worker* con los token que nos ha dado en el arranque del maestro:

```
sudo su
kubeadm join 10.10.10.101:6443 --token h5rxct.v6krsdnzbb04a447 \
     --discovery-token-ca-cert-hash \
     sha256:c23c5ee438e7ddf6bf5fcff5ca6817700e5b9ef63db0927eff8252a4a8f001a9
```

Y podemos ver los nodos en el *máster* con `kubectl get nodes`

Si hacemos `kubectl label node as2 node-role.kubernetes.io/worker=worker` haremos que el nodo esclavo, perdón, *worker*, pase a estado `worker`.

Si accedemos por otra línea de comandos deberemos, para controlar el clúster, hacer de nuevo desde `root` la línea `export KUBECONFIG=/etc/kubernetes/admin.conf`

8.3. Acceso al *Dashboard* desde el exterior de *localhost*

En las webs siguientes se explica cómo hacer visible el Panel desde fuera.

https://www.thegeekdiary.com/how-to-access-kubernetes-dashboard-externally/

https://adamtheautomator.com/kubernetes-dashboard/

Primero desplegamos el Panel o *Dashboard*:

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.4.0/aio/deploy/
    recommended.yaml
kubectl proxy
```

Y podemos acceder desde local con el siguiente enlace `Panel en localhost` o en la línea de comandos con:

```
curl \
 'http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/
    proxy/'
curl localhost:8001/version
```

En la segunda línea cambiamos abajo con `vi` el tipo ClusterIP a NodePort:

```
kubectl get all -n kubernetes-dashboard
kubectl edit service/kubernetes-dashboard -n kubernetes-dashboard
kubectl -n kubernetes-dashboard get services
kubectl get pods --all-namespaces
```

Borramos el pod que nos aparece con un nombre similar a éste:

```
kubectl delete pod kubernetes-dashboard-78c79f97b4-gjr2l -n kubernetes-dashboard
kubectl get svc --all-namespaces # vemos el nuevo NodePort y su puerto 31491
lsof -i tcp:31491 # comprobamos que escucha

kubectl create serviceaccount dashboard -n kubernetes-dashboard

curl -k https://10.10.10.141:31491
kubectl version --output=json
```

Creamos el *token* y lo guardamos:

```
vi dashboard-adminuser.yaml
kubectl apply -f dashboard-adminuser.yaml
kubectl -n kubernetes-dashboard create token admin-user
vi token.txt
history > historiaK8S.txt
```

dashboard-adminuser.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

En apache deberemos configurar para no tener problemas con https (con SSL):

```
<VirtualHost *:443>
  ProxyPreserveHost On
  ServerName k8s.ehu.eus

  SSLProxyEngine on
  SSLProxyVerify none
  SSLProxyCheckPeerCN off
  SSLProxyCheckPeerName off
  SSLProxyCheckPeerExpire off

  ProxyPass               "/"  https://10.10.10.101:32414/
  ProxyPassReverse        "/"  https://k8s.ehu.eus/

  ServerAdmin pablo.gonzalez@ehu.eus
</VirtualHost>
```

Con lo que ya podemos entrar (si tuviéramos el DNS apuntando con ese subdominio) a https://k8s.ehu.eus/ e introducir el *token* como contraseña.

En minikube, sin terminar de comprobar:

```
kubectl -n kubernetes-dashboard edit service kubernetes-dashboard
kubectl -n kubernetes-dashboard get services
lsof -i tcp:32414
kubectl -n kubernetes-dashboard describe $(kubectl -n kubernetes-dashboard get secret -n kubernetes-
    dashboard -o name | \grep dashboard-token) |\grep token:
```

Historia de instrucciones:

```
sudo apt install docker.io
sudo groupadd docker
sudo usermod -aG docker $USER
sudo su
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -
bash -c "echo >/etc/apt/sources.list.d/kubernetes.list \
'deb http://apt.kubernetes.io/ kubernetes-xenial main'"
apt update
apt-get install -y kubelet kubeadm kubernetes-cni
NODENAME=$(hostname -s)
IPADDR="10.10.10.141"
kubeadm init --apiserver-advertise-address=$IPADDR --apiserver-cert-extra-sans=$IPADDR --pod-network
    -cidr=192.168.0.0/16 --node-name $NODENAME
export KUBECONFIG=/etc/kubernetes/admin.conf
kubectl apply -f https://github.com/weaveworks/weave/releases/download/v2.8.1/weave-daemonset-k8s.
    yaml
kubectl label node k2 node-role.kubernetes.io/worker=worker
kubectl get nodes
kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.6.1/aio/deploy/
    recommended.yaml
kubectl proxy &
curl 'http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-
    dashboard:/proxy/'
curl localhost:8001/version
kubectl edit service/kubernetes-dashboard -n kubernetes-dashboard
kubectl get all -n kubernetes-dashboard
kubectl delete pod kubernetes-dashboard-6db586b496-z6ftb -n kubernetes-dashboard
kubectl create serviceaccount dashboard -n kubernetes-dashboard
kubectl -n kubernetes-dashboard get services
kubectl get pods --all-namespaces
kubectl get pods -n kubernetes-dashboard
lsof -i tcp:31491
kubectl get svc --all-namespaces
curl -k https://10.10.10.141:31491

# En default-ssl.conf
# ProxyPass "/" https://10.10.10.141:31491/
# ProxyPassReverse "/" https://lgux61-lsi.ehu.eus/

kubectl version --output=json
vi dashboard-adminuser.yaml
kubectl apply -f dashboard-adminuser.yaml
kubectl -n kubernetes-dashboard create token admin-user
history > historiaK8S.txt
```

dashboard-adminuser.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

8.4. Instalando Træfik con `Helm`

```
export KUBECONFIG=/etc/kubernetes/admin.conf # para poder usar en una sesión nueva
kubectl -n kubernetes-dashboard create token admin-user # para usar el dashboard o panel hay que renovar el token
systemd-resolve --set-dns=10.20.13.6 --interface=ens18 # cambiar el DNS porque da problemas
apt update
curl https://baltocdn.com/helm/signing.asc | sudo apt-key add -
apt-get install apt-transport-https --yes
echo "deb https://baltocdn.com/helm/stable/debian/ all main" | sudo tee /etc/apt/sources.list.d/helm-stable-debian.list
apt update
apt install helm
kubectl create namespace metallb-system
helm repo add traefik https://helm.traefik.io/traefik
helm repo update
helm install metallb --namespace=metallb-system  metallb/metallb

helm install traefik traefik/traefik --set experimental.kubernetesGateway.enabled=true,dashboard.enabled=true,serviceType=LoadBalancer,rbac.enabled=true,dashboard.auth.basic.admin='$apr1$ZywpxeoS$6U80kYPG116slxBceEsVz0',dashboard.domain=lgux61-lsi.ehu.eus --namespace=kube-system
helm list -n kube-system
helm status traefik2 -n kube-system
helm uninstall traefik2 -n kube-system
kubectl get pods -n kube-system| grep '^traefik-' | awk '{print $1}'
kubectl port-forward -n kube-system "$(kubectl get pods -n kube-system| grep '^traefik-' | awk '{print $1}')" 9000:9000
bg
curl localhost:9000/dashboard/
helm search repo
kubectl get crd | grep traefik
# https://blog.zachinachshon.com/traefik-ingress/
```

8.5. Manejo básico de Kubernetes

Para comprobar el sistema usamos (en `root` con ese `export` si hemos instalado sin `minikube`):

```
export KUBECONFIG=/etc/kubernetes/admin.conf
kubectl cluster-info
kubectl get nodes
kubectl get pods
kubectl get services
```

Para acceder desde el exterior ya hemos arrancado el *proxy* al desplegar el *dashboard* en la sección 8.3.

Si estamos en local, podemos ahora arrancar un *proxy*, como dice en `esta web`.

```
kubectl proxy --port=8080
```

y el *Dashboard* o Panel está en `http://localhost:8080/api/v1/proxy/namespaces/kube-system/services/kubernetes-dashboard`

De esa forma veremos lo que arranquemos en minikube.

8.6. Cómo se exponen servicios en el exterior

Un resumen de `esta web` sería:

- ClusterIP: necesita el proxy y no es claro.
- NodePort: cambian los puertos.
- LoadBalancer: cada elemento tiene su propia IP, y hay uno por servicio.
- Ingress: actúa como un proxy inverso. es el más flexible y complejo.

A través de Træfik:

```
export KUBECONFIG=/etc/kubernetes/admin.conf
kubectl apply -f \
    https://raw.githubusercontent.com/traefik/traefik/v1.7/examples/k8s/traefik-deployment.yaml
netstat -tlpn
kubectl -n kube-system get svc
curl localhost:32454/dashboard/
```

8.7. Servidor web en un pod con `minikube`

Lo más sencillo es crear un contenedor y un servicio para que lo gestione y sea accesible en una dirección web.

```
kubectl create -f single_container_pod.yaml
kubectl expose pod web-server --type=NodePort
minikube service web-server
minikube service web-server --url
kubectl delete pod web-server
kubectl delete svc web-server
```

8.8. 3 réplicas de nginx con `minikube`

```
kubectl create -f pod.yaml
kubectl describe pod nginx-deployment
kubectl create -f service.yaml
kubectl describe service nginxservice
minikube service nginxservice --url
```

Se observa la versión del servidor nginx poniendo una página inexistente:

http://192.168.99.100:30070/ y

http://192.168.99.100:30390/kk

8.9. *shell* dentro de un contenedor de los del pod

```
kubectl get pods
kubectl exec nginx-deployment-2743933351-268sd -it sh
kubectl delete pod nginx-deployment-79d686f8f9-r2g77
kubectl get deploy
```

Y controlamos qué pods hay, y probamos las operaciones de escalado y despliegue de nueva versión problemática

```
kubectl get pods
kubectl scale deployment.v1.apps/nginx-deployment --replicas=10
kubectl get pods
kubectl scale deployment.v1.apps/nginx-deployment --replicas=2
kubectl get pods

kubectl set image deployment.v1.apps/nginx-deployment nginx=nginx:verskk
kubectl describe deployments
kubectl get rs
kubectl rollout undo deployment.v1.apps/nginx-deployment --to-revision=1
kubectl get pod
kubectl get deploy
```

8.10. Stateful Sets

A cada contenedor se le asigna un almacenamiento permanente.

Tomado de https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/

```
kubectl get pods -w -l app=nginx # actualiza en directo
kubectl create -f statefulnginx.yml
kubectl get svc

for i in 0 1; do kubectl exec web-$i -- sh -c 'echo $(hostname) > /usr/share/nginx/html/index.html';
    done
for i in 0 1; do kubectl exec -it web-$i -- curl localhost; done
kubectl delete pod -l app=nginx
for i in 0 1; do kubectl exec -it web-$i -- curl localhost; done
kubectl scale sts web --replicas=5
for i in 0 1 2 3 4; do kubectl exec -it web-$i -- curl localhost; done
for i in 0 1 2 3 4; do kubectl exec web-$i -- sh -c 'echo $(hostname) > /usr/share/nginx/html/index.
    html'; done
for i in 0 1 2 3 4; do kubectl exec -it web-$i -- curl localhost; done
minikube service nginx --url
```

8.11. *Couchbase*

Ejemplo de la Base de Datos *Couchbase* de http://blog.kubernetes.io/2016/08/create-couchbase-cluster-using-kubernetes.html

```
kubectl run couchbase --image=arungupta/couchbase
# Maestro
kubectl create -f https://github.com/arun-gupta/couchbase-kubernetes/blob/master/cluster/cluster-master.yml
kubectl get svc
minikube service couchbase-master-service #  usuario ?Administrator? y la password ?password?
# Esclavo / worker
kubectl create -f https://github.com/arun-gupta/couchbase-kubernetes/blob/master/cluster/cluster-worker.yml
kubectl get rc
kubectl scale rc couchbase-worker-rc --replicas=3

# https://forums.couchbase.com/t/how-to-cbbackup-on-kubernetes-docker-issues/10544/2

# cbbackup
# https://github.com/couchbase/couchbase-cli
```

8.12. Otros puntos de trabajo

https://kubernetes.io/blog/2019/03/15/kubernetes-setup-using-ansible-and-vagrant/

https://github.com/ecomm-integration-ballerina/kubernetes-cluster/pulls

https://github.com/Samueladewole/kubernetes-cluster

https://medium.com/better-programming/build-your-own-multi-node-kubernetes-cluster-with-monitoring

https://kubernetes.io/docs/concepts/storage/persistent-volumes/

https://kubernetes.io/docs/tutorials/

https://kubernetes.io/docs/concepts/workloads/controllers/statefulset/