DevOps Sesión 8 (2022-03-07)

Documentación relacionada

- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorios Kubernetes 2020.pdf
- 2-Despliegue de Aplicaciones Kubernetes/2-Laboratorios basicos kubernetes .pdf
- 2-Despliegue de Aplicaciones Kubernetes/Laboratorio kubernetes Curso-DevOps.txt línea 481

Clase

secretos

- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorios Kubernetes 2020.pdf pag 91
- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/mariadb-deployment-secret.yaml
- `create secret…`
  - docker-registry
  - generic
  - tls
- desde directorio, fichero o literal
- no son seguros, no están encriptados, estan en base64 → `kubectl get secret <nombre> -o yaml` y `echo «….» | base64 –decode`
  - si tienes privilegios administrativos, puedes verlos (no será lo normal)
  - `kubectl config get-contexts`

- ```
  kubectl create secret generic mariadb --from-literal=password=00000000
  kubectl describe pod ...
  kubectl delete secret mariadb
  kubectl delete pod ...
  kubectl describe pod ...
  ```

en volumenes

- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/deployment/helloworld-secrets-volumes.yml
- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/deployment/helloworld-secrets.yml
- desplegar secretos usando volúmenes, al estilo SWARM
  - creamos secreto
  - creamos volumen asociando al secreto
  - montamos el volumen en el pod

  - ```
    ##El secreto:
    kubectl apply -f /vagrant/kubernetes-curso/deployment/helloworld-secrets.yml
    kubectl apply -f /vagrant/kubernetes-curso/deployment/helloworld-secrets-volumes.yml
    kubectl describe pod helloworld-deployment-...
    kubectl exec -ti helloworld-deployment-... -- /bin/bash
    #root@helloworld-deployment-78457f7dfc-zn2bf:/app# cat /etc/creds/password
    #root@helloworld-deployment-78457f7dfc-zn2bf:/app# cat /etc/creds/username
    ```

configmap

- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorios Kubernetes 2020.pdf pag 106
- cambios de configuración en caliente (si la aplicación lo soporta)
- no encriptación, todo en texto plano
- `kubectl create configmap`

lab

- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/mariadb-deployment-configmap.yaml

```
kubectl create cm mariadb --from-literal=root_password=00000000 --from-
literal=mysql_usuario=usuario --from-literal=mysql_password=00000000 --from-
literal=basededatos=test
kubectl get cm
kubectl describe cm mariadb
kubectl apply -f /vagrant/kubernetes-curso/mariadb-deployment-configmap.yaml
kubectl get pod
kubectl exec -it mariadb-deploy-cm-... -- mysql -u usuario -p
# show databases;
```

lab

- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/configmap/*
- pasar un fichero de configuración a un POD
- creamos el CM
- creamos el volumen asociado al CM
- asociamos el POD al volumen
    - este POD tiene 2 contenedores (se hablan como **localhost**) y las peticiones al :80 van a :3000 (del otro contenedor). NGINX en proxypass
- asociamos el servicio al pod (en este caso, están en ficheros separados)

```
kubectl create configmap nginx-config --from-file=/vagrant/kubernetes-
curso/configmap/reverseproxy.conf
kubectl get cm
kubectl describe cm nginx-config
kubectl get configmap nginx-config -o yaml
kubectl apply -f /vagrant/kubernetes-curso/configmap/ # desplegamos los 2 ficheros
.yml
```

volúmenes

- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorios Kubernetes 2020.pdf pag 144
- muchos tipos:
    - aws
    - iscsi
    - azure..
    - emptyDir:
        - no persiste (en memoria). Desaparece si cae el POD
        - me permite compartir un directorio entre 2 contenedores de un POD
    - hostPath:
        - asociamos el volumen a un nodo

- nfs:
  - Para cualquiera de ellas en Kubernetes tendremos que crear:
    - PersistentVolume: Donde especificamos el volumen persistente
    - PersistentVolumeClaim: Donde reclamamos espacio en el volumen
  - Modos de acceso:
    - ReadWriteOnce: read-write solo para un nodo (RWO)
    - ReadOnlyMany: read-only para muchos nodos (ROX)
    - ReadWriteMany: read-write para muchos nodos (RWX)
  - Políticas de reciclaje de volúmenes son son:
    - Retain: Reclamación manual
    - Recycle: Reutilizar contenido
    - Delete: Borrar contenido
  - Estados de un volumen:
    - Available: disponible para reclamación
    - Bound: No disponible, se esta utilizando por una reclamación.
    - Released: La reclamación del volumen se a eliminado y esta esperando otra petición del cluster.
    - Failed: En estado de fallo.

lab

- 2-Despliegue de Aplicaciones Kubernetes/Laboratorio kubernetes Curso-DevOps.txt línea 685
- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/webtest-volum-emptydir.yml

```
kubectl apply -f /vagrant/kubernetes-curso/webtest-volum-emptydir.yml
kubectl get pod
kubectl describe pod webtest
kubectl exec -ti webtest -c contenedor2 -- bash
# touch /web2/contenedor2
kubectl exec -ti webtest -c contenedor1 -- ash
# touch /web1/contenedor1
```

lab

- 2-Despliegue de Aplicaciones Kubernetes/Laboratorio kubernetes Curso-DevOps.txt línea 717
- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/mysql-affinity.yml
- en este caso se usa **affinity** (más sencillo con **nodeSelector** o **nodeName**)

```
kubectl get nodes
kubectl describe nodes worker1 |grep -i labels -A 10
kubectl label nodes worker1 env=bd # Etiquetamos nuestro nodo con la variable env con el valor bd
kubectl describe nodes worker1 |grep -i labels -A 10 # Describimos el nodo para visualizar sus labels
kubectl apply -f /vagrant/kubernetes-curso/mysql-affinity.yml
kubectl exec -it mysql-... -- mysql -u root -p
mysql> create database kubernetes;
mysql> show databases;
kubectl delete deployment mysql
kubectl apply -f /vagrant/kubernetes-curso/mysql-affinity.yml
kubectl exec -it mysql-... -- mysql -u root -p
mysql> show databases;
kubectl label node worker1 env- # elimina la etiqueta (el guión final)
```

lab

- 2-Despliegue de Aplicaciones Kubernetes/Laboratorio kubernetes Curso-DevOps.txt línea 790
- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorios Kubernetes 2020.pdf pag 151
- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/nfs-pv-pvc.yaml
- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/mysql-storage-nfs.yaml
- instalar NFS en master:

```
sudo apt install nfs-kernel-server
sudo mkdir /bd-nfs
sudo mkdir /var/shared
sudo chmod 777  /bd-nfs/
sudo chown nobody:nogroup  /bd-nfs/
sudo vi /etc/exports
# /bd-nfs *(rw,no_root_squash)
# /var/shared 10.0.0.0/24(rw,sync,no_root_squash,no_all_squash)
sudo systemctl restart nfs-kernel-server
sudo systemctl status nfs-kernel-server
sudo systemctl enable nfs-kernel-server
```

- ```
  kubectl apply -f /vagrant/kubernetes-curso/nfs-pv-pvc.yaml
  kubectl get pv,pvc
  kubectl apply -f /vagrant/kubernetes-curso/mysql-storage-nfs.yml
  kubectl describe pod mysql-storage-nfs-...
  kubectl exec -ti mysql-storage-nfs-... -- bash
  # df -h
  ```

- para controlar realmente el espacio usado, hay que usar un storageClass
  - los PersistentVolume se crean dinámicamente
- `kubectl drain worker1 --delete-local-data --ignore-daemonsets`
- `kubectl uncordon worker1`

k0sctl

- 2-Despliegue de Aplicaciones Kubernetes/Install kubernetes k0sctl
- 2-Despliegue de Aplicaciones Kubernetes/Laboratorio Instalar Cluster Kubernetes Alumnos (kubeadm)
- https://k0sproject.io
- Lens (portainer para k8s)

extra

- `kubectl get pod -A -o wide`
- `kubectl -n kube-system describe pod kube-flannel…` → ConfigMap ← 1 pod en cada nodo (servicio global en Swarm)
- HELM:
  - yum o apt en k8s
  - kubeapps hub
- RBAC - reglas de acceso
- https://ngrok.com/
  ```

TODO