

DevOps Sesión 6 (2022-02-28) k8s

Documentación relacionada

lab1

- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorio kubernetes Curso-DevOps.txt
- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorios Kubernetes 2020.pdf, pág 20
- 2-Despliegue de Aplicaciones Kubernetes/2-Laboratorios basicos kubernetes .pdf
- 2-Despliegue de Aplicaciones Kubernetes/Seminario kubernetes.pdf
- lab-docker/mv-kubernetes-Vagrant-2020
 - cluster 3 nodos k8s (1 master, 2 workers)

Clase

k8s

2-Despliegue de Aplicaciones Kubernetes/Seminario kubernetes.pdf

- masters no ejecutan contenedores por defecto (al contrario que swarm)
- orquestados que soporta diferentes runtimes
 - no solo docker. de hecho, desde la versión 1.20 no soporta docker si no containerd
- control plane - panel de control
 - <https://kubernetes.io/es/docs/concepts/overview/components/>
 - servicios en master
 - servicios en worker
 - kubelet
 - kube-proxy
- microk8s.io - minikube

objetos k8s

- POD - unidad mínima (compuesta por 1 o varios contenedores)
 - no escalable
 - no alive
 - → replication controller
- SERVICE
 - conceptualmente diferente a la idea de servicio de SWARM
 - para llegar a un pod:
 - **nodePort**
 - **clusterIP**
 - **loadBalancer**

- se gestiona por **labels** (de hecho, todos los objetos de k8s)
- REPLICATION CONTROLLER (RC)
 - si escala
 - si se mira si falla
- DEPLOYMENT
 - REPLICA SET (RS) ← «avanzado» de RC
 - permite el cambio de versiones en el POD, pero no lo permite RC
 - POD
 - SERVICE

kubectl

- ficheros de trabajo, copiar a mv-kubernetes-Vagrant-2020 (accesible en la máquina virtual a través de /vagrant)
 - 2-Despliegue de Aplicaciones Kubernetes/k8-for-devs-master
 - 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso
 - 2-Despliegue de Aplicaciones Kubernetes/kubernetes-Helm3-API-Metrics-Server
 - 2-Despliegue de Aplicaciones Kubernetes/kubernetes-labs2
 - 2-Despliegue de Aplicaciones Kubernetes/Laboratorio-deployment-strategies

lab

- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorios Kubernetes 2020.pdf, pág 20
- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorio kubernetes Curso-DevOps.txt pag 52-100

- <https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#run>
- kubectl run
- deplegamos un pod y miramos:

```
kubectl apply -f /vagrant/kubernetes-curso/first-app/helloworld.yml
kubectl get pod -o wide
kubectl describe pod nodehelloworld.example.com
```

- montamos el servicio al POD (que permite relacionarse con el exterior)

```
kubectl expose pod nodehelloworld.example.com --type NodePort --name nodehelloworld-service
kubectl get service ' ' # información del servicio
kubectl describe service nodehelloworld-service
```

- acceder a un contenedor:

```
kubectl exec -ti nodehelloworld.example.com -- bash
```

- logs:

```
kubectl logs nodehelloworld.example.com
```

- eventos del cluster: kubectl get events
- eliminar (2 alternativas):

```
kubectl delete pod nodehelloworld.example.com
kubectl delete pod -f /vagrant/kubernetes-curso/first-app/helloworld.yml
kubectl delete service nodehelloworld-service
```

- ... `--wait=false`: devuelve el control al prompt sin esperar a ejecutar la orden
- Visual Studio Code, extensiones para facilitar los yaml
 - kubernetes templates
 - u otros... ¿?

lab

- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorios Kubernetes 2020.pdf, pág 29
- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-curso/replication-controller/
- 2-Despliegue de Aplicaciones Kubernetes/1-Laboratorio kubernetes Curso-DevOps.txt pag. 100-142

- `kubectl scale rc <controller> --replicas=3`
- mala práctica: `kubectl edit rc <controller>`
- desplegar:

```
kubectl apply -f /vagrant/kubernetes-curso/replication-controller/helloworld-repl-controller.yml
```

- si elimino uno de los pods, el RC se encarga de levantar otro

```
kubectl get rc -o wide
kubectl describe pod helloworld-controller-xxxxx
kubectl delete pod helloworld-controller-xxxxx
kubectl get pods --show-labels
```

- Para escalar nuestro rc, podemos realizarlo mediante las dos comandos:

```
kubectl scale --replicas=4 -f /vagrant/kubernetes-curso/replication-controller/helloworld-repl-controller.yml
kubectl scale rc helloworld-controller --replicas=4
```

- `kubectl expose rc helloworld-controller --type=NodePort --name helloworld-controller-service`
`kubectl get service`
`kubectl describe service helloworld-controller-service`

- <http://10.0.0.11:nodeport>

- Para finalizar el laboratorio eliminamos el rc, y veremos que se eliminan los pods, asiados a este rc:

```
kubectl get rc
kubectl delete rc helloworld-controller
kubectl get pod,rc
kubectl delete service helloworld-controller-service
```

labels

- `kubectl get pods --show-labels`
- `kubectl get all -l app=helloworld (o --selector)`
- `kubectl get pods -l 'env in (production,development)' --wait=false`
- todo se relaciona con etiquetas
- restricciones (documentación)

lab (labels)

- 2-Despliegue de Aplicaciones Kubernetes/2-Laboratorios básicos kubernetes.pdf, pág 4
- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-labs2/labels/

- desplegar un pod:

```
kubectl apply -f /vagrant/kubernetes-labs2/labels/pod.yaml
kubectl get pods --show-labels
```

- cambiar etiquetas:

```
kubectl label pods label=owner=miempresa
kubectl get pods --show-labels
kubectl get pods --selector owner=miempresa
kubectl get pods -l env=development
```

- lanzamos otro pod:

```
kubectl apply -f /vagrant/kubernetes-labs2/labels/anotherpod.yaml
kubectl get pods -l 'env in (production, development)'
kubectl delete pods -l 'env in (production, development)'
```

lab

- <https://kubernetes.io/docs/concepts/workloads/pods>

- ```
apiVersion: v1
kind: Pod
metadata:
 name: nginx
spec:
 containers:
 - name: nginx
 image: nginx:1.14.2
 ports:
 - containerPort: 80
```

- `kubectl apply -f pod.yaml`  
`kube expose pod nginx --type=NodePort nginx-service # no funciona, falta una etiqueta`

```
kube label pods nginx app=miapp
kubectl expose pod nginx --type=NodePort nginx-service # el servicio se
engancha a través de la etiqueta
kubectl describe service nginx-service
```

- mirar **Endpoints** para ver que el servicio está relacionado con POD
- si hay varias etiquetas, relaciona con todas
- el **kubectl expose** hace el enganche entre el pod y el service que crea a través de **labels(parte pod)-selector(parte service)** y del **containerPort(parte pod)-targetPort(parte service)**. Estos valores los has de setear tu si pasas toda la información en el .yaml

## lab

- 2-Despliegue de Aplicaciones Kubernetes/kubernetes-labs2/pods
- 2-Despliegue de Aplicaciones Kubernetes/2-Laboratorios básicos kubernetes.pdf, pág 1

- despliegue POD desde CLI (demo):

```
kubectl run sise --image=mhausenblas/simpleservice:0.5.0 --port=9876
```

- desplegamos 2 contenedores:

```
kubectl apply -f /vagrant/kubernetes-labs2/pods/pod.yaml
kubectl exec twocontainers -c shell -i -t -- bash # muestra la información del
POD (y los dos contenedores)
```

- para acceder a uno de los contenedores de ese pod, hay que usar el parámetro **-c**:

```
kubectl exec twocontainers -c shell -i -t -- bash
```

- los contenedores no tienen IP propia, se hablan a través de localhost
- " curl -s localhost:9876/info" (desde shell)
- uso: concepto de sidecars container
- un pod no está **RUNNING** si no están todos los contenedores arriba

From:  
<https://matebcn.eu/> - miguel angel torres egea

Permanent link:  
<https://matebcn.eu/info:cursos:pue:devops2022:s6>

Last update: **09/03/2022 16:43**

