Gestionando los Pods

Tenemos un fichero pod.yaml, donde hemos definido un Pod de la siguiente manera:

```
apiVersion: v1
kind: Pod
metadata:
 name: pod-nginx
 labels:
  app: nginx
  service: web
spec:
 containers:
  - image: nginx:1.16
    name: contenedor-nginx
    imagePullPolicy: Always
```

Podemos crear directamente el Pod desde el fichero yaml:

kubectl create -f pod.yaml

Y podemos ver el estado en el que se encuentra y si está o no listo:

kubectl get pods

(Sería equivalente usar po, pod o pods).

Si queremos ver más información sobre los Pods, como por ejemplo, saber en qué nodo del cluster se está ejecutando:

kubectl get pod -o wide

Para obtener información más detallada del Pod (equivalente al inspect de docker):

kubectl describe pod pod-nginx

Podríamos editar el Pod y ver todos los atributos que definen el objeto, la mayoría de ellos con valores asignados automáticamente por el propio Kubernetes y podremos actualizar ciertos valores:

kubectl edit pod pod-nginx

Sin embargo, es una opción compleja para utilizarla a estas alturas del curso y hay que comprender mejor cómo funcionan los objetos de Kubernetes para poder hacer modificaciones de forma apropiada, y además, veremos más adelante otra manera más correcta de actualizar un objeto de Kubernetes.

Normalmente no se interactúa directamente con el Pod a través de una shell, pero sí se obtienen directamente los logs al igual que se hace en docker:

kubectl logs pod-nginx

En el caso poco habitual de que queramos ejecutar alguna orden adicional en el Pod, podemos utilizar el comando exec, por ejemplo, en el caso particular de que queremos abrir una shell de forma interactiva:

kubectl exec -it pod-nginx -- /bin/bash

Podemos acceder a la aplicación, redirigiendo un puerto de localhost al puerto de la aplicación:

kubectl port-forward pod-nginx 8080:80

Y accedemos al servidor web en la url http://localhost:8080.

NOTA: Esta no es la forma con la que accedemos a las aplicaciones en Kubernetes. Para el acceso a las aplicaciones usaremos un recurso llamado Service. Con la anterior instrucción lo que estamos haciendo es una redirección desde localhost el puerto 8080 al puerto 80 del Pod y es útil para pequeñas pruebas de funcionamiento, nunca para acceso real a un servicio. **NOTA2**: El port-forward no es igual a la redirección de puertos de docker, ya que en este caso la redirección de puertos se hace en el equipo que ejecuta kubectl, no en el equipo que ejecuta los Pods o los contenedores.

Para obtener las etiquetas de los Pods que hemos creado:

kubectl get pods --show-labels

Las etiquetas las hemos definido en la sección metadata del fichero yaml, pero también podemos añadirlos a los Pods ya creados:

kubectl label pods pod-nginx service=web --overwrite=true

Las etiquetas son muy útiles, ya que permiten seleccionar un recurso determinado (en un cluster de Kubernetes puede haber cientos o miles de objetos). Por ejemplo para visualizar los Pods que tienen una etiqueta con un determinado valor:

kubectl get pods -l service=web

También podemos visualizar los valores de las etiquetas como una nueva columna:

kubectl get pods -Lservice

Y por último, eliminamos el Pod mediante:

kubectl delete pod pod-nginx