Escalado y replicación

Nos puede interesar tener más de una instancia de un POD, pero por necesidades de demanda o por disponibilidad, nos puede interesar tener réplicas de un POD.

Vamos a ver cómo utilizar esto en aplicaciones stateless. La forma más habitual es especificar el parámetro replica en nuestro despliegue.

Escalado modificando el despliegue

Podemos hacer varias cosas, entre ellas podemos cambiar el despliegue para que nos cree 4 réplicas del POD.

```
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: tomcat-deployment
spec:
  selector:
    matchLabels:1
      app: tomcat
  replicas: 4
 ...
 ...
```

Escalado por comando

Otra opción es no modificar el despliegue (dejarlo a replicas=1 y ejecutar el comando scale para decirle cuantas réplicas quiero

```
kubectl scale --replicas=4 deployment/tomcat-deployment
deployment.extensions/tomcat-deployment scaled
```

Veremos que ahora los despliegues nos mostrarán el número de réplicas:

```
kubectl get deployments
NAME                READY   UP-TO-DATE   AVAILABLE   AGE
hazelcast           0/1     1            0           29m
hello-minikube      1/1     1            1           6d20h
tomcat-deployment   4/4     4            4           53m
```

Ahora tenemos un problema de red, puesto que cuando únicamente teníamos un POD, mapeábamos un puerto suyo con uno accesible externamente.

Para que los 4 puedan escuchar en el mismo puerto y se repartan las peticiones de servicio entre todos, necesitaremos un **balanceador de carga**.

Crear servicio de balanceo de carga

Crearemos un servicio para utilizar un **load balancer** que exponga un único puerto externo y balancear la carga a los diferentes pods.

Ojo que está en varias líneas pero es una sola.

```
kubectl expose deployment tomcat-deployment
--type=LoadBalancer
--port=8080
--target-port 8080
--name=tomcat-load-balancer
```

A lo que nos contesta:

```
service/tomcat-load-balancer exposed
```

Como podemos ver, hemos asignado el balanceador al despliegue **tomcat-deployment**, para que actúe sobre él.

Mostrar información del balanceador

Vamos a ver cómo ha quedado la cosa, mirando la descripción del balanceador que hemos creado:

```
kubectl describe service tomcat-load-balancer
```

La respuesta nos indica que hemos mapeado el puerto 8080 al 8080 de cada uno de los 4 PODs que tenemos en ejecución.

Se encargará de redirigir las peticiones a uno de ellos, repartiendo la carga.

```
Name:                     tomcat-load-balancer
Namespace:                default
Labels:                   <none>
Annotations:              <none>
Selector:                 app=tomcat
Type:                     LoadBalancer
IP:                       10.96.43.114
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30854/TCP
Endpoints:                172.17.0.5:8080,172.17.0.7:8080,172.17.0.8:8080 + 1
```

```
more...
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>
```

Como podemos ver, las peticiones al puerto `8080` de la IP `10.96.43.114` se distribuirán a los endpoints:

- 172.17.0.5:8080
- 172.17.0.7:8080
- etc.