UNIDAD 5
PRUEBAS DE DISPONIBILIDAD

Liveness probe

- Cada nodo de Kubernetes trae consigo un agente llamado kubelet, que tiene como cometido asegurarse de que tus pods están sanos.

- Por cada pod podemos además configurar una prueba/s que indican que estos se están comportando de manera correcta, más allá de que el contenedor se esté ejecutando.

- Estas pruebas pueden ser de tres tipos: **a través de un comando que se ejecuta dentro del contenedor, una llamada HTTP o bien TCP.**

- Si la prueba que hemos configurado no se cumple, el componente kubelet, **reiniciará el pod,** para intentar solucionar el problema.

Liveness probe

```
...
 spec:
  containers:
  - name: liveness
    image: nginx
    livenessProbe:
      httpGet:
        path: /healthz
        port: 80
      initialDelaySeconds: 3
      timeoutSeconds: 3
      periodSeconds: 10
      successThreshold: 1 #Must be 1 for liveness
      failureThreshold: 5
```

- En este ejemplo he configurado una prueba del tipo httpGet, donde indicamos un path y un puerto en el que cada cierto tiempo se lanzará una llamada.
- Si el resultado es satisfactorio (cualquier código mayor o igual a 200 y menos que 400) **significara que tu aplicación está funcionando correctamente.**
- **delay:** retardo con el cual se lanza la prueba cuando el contenedor es iniciado. Esperamos a que la aplicación esté lista.
- **timeout**: se trata del tiempo en el que se espera que el contenedor responda a la petición.
- **period**: si no especificamos ningún valor, la prueba se lanza cada 10 segundos.
- **success**: indica cuántas veces seguidas tiene que ser satisfactoria la prueba. En este tipo de pruebas el valor solo puede ser 1.
- **failure**: por defecto, el pod no se reinicia cada vez que la prueba falla, sino que se espera a que se realicen 3 intentos de cada prueba.

Liveness probe. Ejemplo

```
kubectl apply -f pod.yaml
```

Comprobamos que ha tenido reinicios, porque la prueba está fallando:

```
kubectl get all
NAME                READY   STATUS    RESTARTS   AGE
pod/liveness-http   1/1     Running   2          45s
```

Podemos comprobar el estado del pod:

```
kubectl describe pod/liveness-http
...

Liveness:       http-get http://:80/healthz delay=3s timeout=3s period=3s #success=1 #failure=5
…
Warning  Unhealthy  16s (x11 over 52s)  kubelet, minikube  Liveness probe failed: HTTP probe failed with statuscode: 404
```

Creamos un fichero, para que la prueba no falle:

```
kubectl exec pod/liveness-http -- sh -c 'mkdir /usr/share/nginx/html/healthz;echo "Ok" >
/usr/share/nginx/html/healthz/index.html'
```

Comprobamos el funcionamiento del pod:

```
kubectl port-forward pod/liveness-http 8081:80
```

Readlines probe

- Otro tipo de pruebas que están pensadas para comprobar si una aplicación está lista para atender peticiones por parte de los usuarios u otros pods.

- En lugar de reiniciar tu contenedor, con el objetivo de recuperar tu aplicación, lo que hace es que **saca al mismo de su balanceador para que no le lleguen peticiones.**

- Este mecanismo es útil cuando tu aplicación se está iniciando y no puede estar disponible de inmediato para recibir peticiones.

- Estas pruebas pueden ser de tres tipos: **a través de un comando que se ejecuta dentro del contenedor, una llamada HTTP o bien TCP.**

Readliness probe

```
...
 sspec:
      containers:
        - name: nginx
          image: nginx
          ports:
            - name: http
              containerPort: 80
          readinessProbe:
            exec:
              command:
                - cat
                - /tmp/iamready
```

- En este ejemplo he configurado una prueba del tipo exec, que lanzará el comando cat /tmp/iamready esperando que dicho archivo al menos exista.
- Si existe el fichero, la prueba será OK y se incluirá el pod en el servicio NodePort que hemos creado.
- En caso contrarío, se sacará del servicio el pod.
- Los parámetros que vimos en la prueba anterior son válidos en esta clase de prueba.

Readlines probe. Ejemplo

```
kubectl apply -f depoly_service.yaml
```

Como no existe el fichero en ninguna replica, comprobamos que el servicio no tiene asociado ningún pod:

```
kubectl describe service/nginx-service
...
Endpoints:
```

Creamos en el fichero en una replica, para que la prueba sea ok:

```
kubectl exec nginx-deployment-fcf86974b-62622 -- sh -c "touch /tmp/iamready”
```

Y volvemos a comprobar el servicio:

```
kubectl describe service/nginx-service
...
Endpoints:   172.17.0.2:80
```

Startup probe

- Este apareció en la versión 1.16 de Kubernetes y actualmente está en versión alpha.

- El uso ideal del mismo es aplicarlo cuando nuestro pod tarda algún tiempo antes de arrancar por completo.

- Sai nuestro pod tarda en arrancar, podría ser problemático si no especificamos correctamente nuestro liveness probe ya que podríamos acabar en un reinicio constante del pod,

- Cuando definimos el startup probe, Kubernetes no habilita el liveness probe para que le indique si el pod ya está arrancado. Si este probe fallara, Kubernetes reiniciaría el pod.