Pod lifecycle da kaç tan restart policy vardı? Always On Failure ve Never olmak üzere 3 tane.

Kubelet bu policylere göre podu çalıştırır. Ancak bazı durumlarda container çalışıyordur fakat kendinden bekleneni yapmıyordur. Kubelet bunu bilemez, conatinerın ayakta olup olmadığına bakar. Bunun için de probe ları kullanıyoruz.

Policylerden never olursa probe lar işlevsiz kalır.

LivenessProbe; helathcheck conatiner kendinden bekleneni yapıyor mu? Uygulamada bir tıkanma var mı diye bakıyor.

LivenessProbe un bir çeşidi HTTP GET ile HTTP portunda çalışan aplikasyonun hata kodlarını check eder.

LivenessProbe un diğer bir çeşidi ise exec, çünkü her container web aplikasyonu çalıştırmaz bazılarının işi komutlar çalıştırmaktır. Exec de o containerların komutları yerine getirip getirmediğini check eder.

apiVersion: v1

kind: Pod

metadata:

labels:

test: liveness

name: liveness-exec

spec:

containers:

- name: liveness

image: clarusway/probes

args:

- /bin/sh

- -c

- touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600

livenessProbe:

exec:

command:

- cat

- /tmp/healthy

initialDelaySeconds: 5

periodSeconds: 5

Bu pod da args: ile tmp/healthy diye bir dosya oluşturup 30 saniye bekleyip sonra bu dosyayı silip 600 saniye sleep olması istenmiş.

LivenessProbe exec ile de tmp/healthy dosyasının cat ile göterilmesini istiyor. Livenessprobe ilk 5 saniye check yapmayacak (initialDelaySeconds), sonrasında ise 5 saniyede bir check yapacak. Ilk 30 saniye bunu yapabilecek ancak 30 saniye sonra dosya silineceği için cat komutu çalışamayacak ve container hata verecek. Hata verince kubelet container ı restart edecek.

Bir de TCP liveness probe var. Bu da containerın belirli bir port (TCP socket) tan ulaşılabilir olup olmadığına bakıyor.

apiVersion: v1

kind: Pod

metadata:

name: liveness-tcp

spec:

containers:

- name: liveness-tcp

image: mysql

ports:

- containerPort: 3306

env:

- name: MYSQL\_ROOT\_PASSWORD

value: "123456"

livenessProbe:

tcpSocket:

port: 8080

initialDelaySeconds: 15

periodSeconds: 20

Örneğin bu mysql image ile oluşan container a 8080 portundan ulaşılıp ulaşılamadaığını check etsin (çalışmayacak çünkü mysql 3306 tan trafik alıyor) burada initialDelaySeconds 15 sn çünkü mysql image in ayağa kalkması biraz uzun sürer. Sonrasında ise 20 sn de bir check yapacak.

Biz tcpSocket e 3306 portunu yazsaydık hata vermeyecekti.

![Define a Zliveness probe 
FEATURE STATE. 
• Kubernetes VI.24 [beta] 
If your application implements the gRPC Health Checking Protocol, this example shows how to 
configure Kubernetes to use it for application liveness checks. Similarly you can configure 
readiness and startup probes. 
Here is an example manifest: ]()

1. Livenessprobe ise yeni bir yaklaşım. g google ı simgeliyor. **gRPC;** Google’ın geliştirdiği Remote Procedure Call, yani başka bir servis ya da uzak sunucudaki bir metodu sanki kendi servisimizin metoduymuş gibi kullanabilmemizi sağlayan, client-server ilişkisindeki iletişimi kolay ve hızlıca sunan bir frameworktür.

bir diğer probe çeşidi startupProbe dur:

eski büyük uygulamalara legcay uygulamalar deniyor. bunlar kubernetes ortamında çalıştığı zaman başlangıç süreleri değişiyor. bu tür uygulamalar için startupprobe kullanılmış. startupprobe da container içindeki aplikasyonun çalışıp çalışmadığını kontrol eder, çalıştıktan sonra diğer probe lar devreye girer. startup probe devredeyken liveness ve readiness probe lar bekler.

startup liveness ve readiness ile birlikte kullanılır.

eski tip uygulamalarda kullanılıyor çünkü bunlar stabile olmayabiliyor. yoksa livenessprobe daki initialDelaySeconds yeterli olur.

livenessProbe:

httpGet:

path: /healthz

port: 80

initialDelaySeconds: 3

periodSeconds: 3

startupProbe:

httpGet:

path: /healthz

port: 80

failureThreshold: 5

periodSeconds: 15

burada ikisi de httpGET ile kullanılmış ancak farklı da kullanılabilir. startupprobe da failurethreshold da var, 5 kere şans veriyor bu uygulamaya.

startopprobe görevini yaptıktan sonra uzay mekiğinden ayrılan roketler gibi ayrılıyor, diğer probe lar devreye giriyor.

üçüncü probe türü ise readiness probe:

apiVersion: apps/v1

kind: Deployment

metadata:

name: readiness

spec:

replicas: 3

selector:

matchLabels:

test: readiness

template:

metadata:

labels:

test: readiness

spec:

containers:

- name: readiness

image: clarusway/readinessprobe

ports:

- containerPort: 80

readinessProbe:

httpGet:

path: /healthz

port: 80

initialDelaySeconds: 3

periodSeconds: 3

container ın belirtilen porttan trafik alıp alamadığına bakar, yani conatiner ın hazır olup olmadığını check eder.

hazır olmayan pod u service in load balancer yapacağı podlar arasında olmasına engel olur. hazır olmadan podları servisin altına koyma. hazır olmasını bekler. restart yapmaz.