Liveness Probe

What happen if the application in the POD is running but it can’t serve its main purpose for whatever reason? also applications that runs for long time might transition to broken states. In all cases the last thing you want have is a call reporting a problem in an application that could be easily fixed with restarting the POD  
liveness probes is a Kubernetes features made specially for that.   
liveness probes sent a pre-defined request to the POD on a regular basis then restart the POD if this request failed.  
The most commonly used liveness probe is HTTP GET request, but it could also be opening TCP socket or issuing a command

this is a TCP socket probe example where the “initialDelaySeconds” is the waiting time before the first try to open a TCP socket to port 80 then it will run the probe every 20 second as specified in “periodSeconds” If that failed the POD would be restarted automatically

apiVersion: v1

kind: Pod

metadata:

name: liveness-pod

labels:

app: tcpsocket-test

spec:

containers:

- name: liveness-pod

image: virtualhops/ato-ubuntu:latest

ports:

containerPort: 80

securityContext:

privileged: true

capabilities:

add:

- NET\_ADMIN

livenessProbe:

tcpSocket:

port: 80

initialDelaySeconds: 15

periodSeconds: 20

HTTP GET request probe is similar to the TCP socket probes but it will will sent HTTP GET request and you have the option to specify the path which in here just the main website also you can send the probe with customized header

apiVersion: v1

kind: Pod

metadata:

name: liveness-pod

labels:

app: tcpsocket-test

spec:

containers:

- name: liveness-pod

image: virtualhops/ato-ubuntu:latest

ports:

containerPort: 80

securityContext:

privileged: true

capabilities:

add:

- NET\_ADMIN

livenessProbe:

httpGet:

path: /

port: 80

httpHeaders:

- name: some-header

value: Running

initialDelaySeconds: 15

periodSeconds: 20

Readiness Probe

Liveness probe make sure that your POD is able to good health but for some application Liveness alone isn’t enough. some application need to load large files before it start.  
you might think if we set a higher “initialDelaySeconds” value then problem solve but this not an efficient way.  
Readiness probe is solution in here specially with Kubernetes services as the POD will not receive a traffic until it report ready   
Readiness Probe is configured the same way as liveness prob

apiVersion: v1

kind: Pod

metadata:

name: liveness-pod

labels:

app: tcpsocket-test

spec:

containers:

- name: liveness-pod

image: virtualhops/ato-ubuntu:latest

ports:

containerPort: 80

securityContext:

privileged: true

capabilities:

add:

- NET\_ADMIN

livenessProbe:

tcpSocket:

port: 80

initialDelaySeconds: 15  
 periodSeconds: 20

readinessProbe:

tcpSocket:

port: 80

initialDelaySeconds: 5

periodSeconds: 10

TIP : its recommended to use both Readiness Probe and Liveness Probe where Liveness probe restart the POD if it failed and Readiness Probe make sure the POD is ready before it gets the traffic