

# Kubernetes probes: liveness, readiness and startup

One page to review your probes before you deploy, and to diagnose a `CrashLoopBackOff` before you open the code.

## 1 What each probe does when it fails

| PROBE                       | IF IT FAILS                                                                                                  | USE IT TO                                                                                               |
|-----------------------------|--------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| <code>livenessProbe</code>  | The kubelet restarts the container.                                                                          | Catch a hung process that will not recover on its own.                                                  |
| <code>readinessProbe</code> | The pod is removed from the Service endpoints and stops receiving traffic. It is not restarted.              | Keep traffic away from a pod that cannot serve right now: starting, overloaded or missing a dependency. |
| <code>startupProbe</code>   | Until it succeeds, the other two do not run. If it runs out of attempts, the kubelet restarts the container. | Apps with a slow or variable startup.                                                                   |

Defaults: `initialDelaySeconds: 0, periodSeconds: 10, timeoutSeconds: 1, successThreshold: 1, failureThreshold: 3`. A liveness probe restarts the container after `failureThreshold` consecutive failures, counted from `initialDelaySeconds` every `periodSeconds`.

## 2 Before you deploy

- ☐ Apps with a slow or variable startup have a `startupProbe`, with `failureThreshold` × `periodSeconds` above the slowest startup you have measured.
- ☐ The liveness probe checks only the process itself, not the database or other services. If a dependency goes down, restarting every pod does not fix it.
- ☐ The readiness probe reflects whether the pod can serve traffic right now.
- ☐ `timeoutSeconds` leaves room for CPU spikes, throttling from limits or garbage collection pauses.
- ☐ Probe endpoints answer fast and do no heavy work: no queries and no calls to other services.
- ☐ After a change that makes startup longer (migrations, cache warm-up, a new dependency), the probe timings are reviewed.
- ☐ Values are tuned per container, not copied from a generic template.

## 4 Starting template

```
startupProbe:
  httpGet: { path: /healthz, port: 8080 }
  periodSeconds: 5
  failureThreshold: 30 # up to 150 s to start
livenessProbe:
  httpGet: { path: /healthz, port: 8080 }
  periodSeconds: 10
  timeoutSeconds: 2
  failureThreshold: 3
```

## 3 If you hit CrashLoopBackOff

```
# Events: look for "Liveness probe failed"
# and "will be restarted"
kubectl describe pod <pod>

# Probe failures only
kubectl get events \
  --field-selector reason=Unhealthy

# Logs from the previous start
kubectl logs <pod> --previous
```

How to read it: if startup is cut short with no application error and the events mention the probe, the probe is the cause. If the logs show an application error, the cause is in the code or the configuration. A container restarted by the liveness probe usually shows `Last State` with exit code 137 or 143.

```
readinessProbe:
  httpGet: { path: /ready, port: 8080 }
  periodSeconds: 5
  timeoutSeconds: 2
  failureThreshold: 3
# /healthz: the process responds
# /ready: it can serve traffic now
```

Tune the numbers to your app's real timings. Do not raise the liveness `initialDelaySeconds` to cover a slow startup: it delays catching real hangs for the whole life of the pod.