Probes in Kubernetes are **health checks** used by the kubelet to determine the **status of a container**. They help Kubernetes decide whether to:

* Start sending traffic to a pod
* Restart a failing container
* Remove a pod from a service

**🚦 2. Why Are Probes Important?**

Without probes:

* **Services** may send traffic to a pod that's **still starting** or **stuck**.
* **Ingress or LoadBalancer** may forward traffic to a **non-working pod**, leading to failures.
* Kubernetes can’t **automatically recover** a broken pod unless it knows it’s broken.

With probes:

* Kubernetes routes traffic **only to ready pods**.
* Kubernetes restarts pods if they **hang or crash silently**.
* Users experience **better reliability and uptime**.

**🔧 3. How Probes Work Internally**

Each probe is defined in the Pod spec, and Kubernetes uses the **kubelet** (agent running on the node) to:

1. **Periodically call the probe** (every X seconds).
2. Based on the response:
   * **200 OK** (HTTP), or **exit code 0** (exec), or \*\*successful TCP connection): ✅ Success.
   * Otherwise: ❌ Failure.

A screenshot of a questionnaire

AI-generated content may be incorrect.

**There are three types of probes:**

**🚦 1. Liveness Probe**

Determines **if a container is still running**.

* If it fails, Kubernetes **kills the container** and restarts it.
* Useful for **deadlock detection**.

livenessProbe:

httpGet:

path: /actuator/health/liveness

port: 8881

initialDelaySeconds: 15

periodSeconds: 10

failureThreshold: 2

🔁 K8s starts checking after initialDelaySeconds, and probes every periodSeconds.

**🚥 2. Readiness Probe**

Checks if the container is **ready to accept traffic**.

* If it fails, the pod is **removed from the Service's endpoints**.
* The pod **is not restarted**.

readinessProbe:

httpGet:

path: /actuator/health/readiness

port: 8881

initialDelaySeconds: 10

periodSeconds: 5

failureThreshold: 2

✅ Used during graceful shutdown to mark pod as not ready (stop receiving traffic).

**🔄 3. Startup Probe**

Used for **slow-starting apps** (e.g., large Java apps).

* Runs **before liveness** or **readiness** probes.
* Once it **succeeds**, other probes begin.
* Prevents premature restarts.

startupProbe:

httpGet:

path: /actuator/health

port: 8080

failureThreshold: 30

periodSeconds: 5

⏱ This allows up to **5 minutes (30×10s)** for the app to start.

**🔧 Probe Methods**

Kubernetes supports **three methods** for probes:

| **Method** | **Description** | **Use Case** |
| --- | --- | --- |
| httpGet | Makes HTTP request to a path/port | Web servers, APIs |
| exec | Runs a command inside the container | CLI-based apps |
| tcpSocket | Opens a TCP socket to the container | Databases, message brokers |

**🧪 5. Real Example – Spring Boot App with Probes**

livenessProbe:

httpGet:

path: /actuator/health/liveness

port: 8080

initialDelaySeconds: 10

periodSeconds: 10

readinessProbe:

httpGet:

path: /actuator/health/readiness

port: 8080

initialDelaySeconds: 5

periodSeconds: 5

This configuration:

* **Waits 10s** before checking if the app is **alive**.
* **Checks every 5s** whether app is **ready for traffic**.

**🌐 6. Probes and Load Balancing (Service / Ingress)**

**➕ Kubernetes Service:**

* **readinessProbe** is key.
* If a pod **fails readiness**, it’s **removed** from the **Service Endpoint list** → **no traffic** routed to it by:
  + ClusterIP
  + NodePort
  + LoadBalancer

**➕ Ingress Controllers (e.g., NGINX, AGIC):**

* They **respect Kubernetes Services** → If **readinessProbe fails**, ingress won’t send traffic to the pod.
* Some ingress controllers can be configured with **custom health checks**, but that’s not mandatory if you use K8s readinessProbe

**🔐 8. How to Configure with Ingress LoadBalancer**

**✅ You don’t configure probes in the ingress directly.**

Instead:

1. Use a Service with selector targeting pods.
2. Ensure pods have correct **readinessProbe**.
3. Ingress routes traffic to the **Service**, which internally **routes only to "ready" pods**.

A screenshot of a computer

AI-generated content may be incorrect.

🌐 Step 3: Access the App from Browser or Curl

curl http://<EXTERNAL-IP>:8881/actuator/health

curl <http://20.193.45.22:8881/actuator/health>

**📌 Notes for Azure AKS**

* You must have a working **Standard SKU Load Balancer** (Basic is limited).
* Ensure **egress rules** and **NSGs** allow external access on port 8881.