

Explain Readiness & Liveness Probe with analogy and real example.

Analogy: Doctor's Clinic

Concept Analogy

Container A doctor's clinic

Readiness Probe "Is the clinic ready for patients?"

"Is clinic in working

Liveness Probe condition?"

• If clinic is not ready, don't send patients.

• If power cut, call backup power or start generator service.

Real-World Example: Food Delivery App

Imagine you're running a Swiggy/Zomato-like app in Kubernetes.

Components:

- API server (manages orders)
- Search service
- Delivery assignment service

Now, let's say:

- The API server is starting up, loading database connections.
- It takes 10–15 seconds to become usable, but the container is already "Running".

If there's **no readiness probe**, Kubernetes **will start sending user traffic** to it during startup — leading to **errors or failed orders**.

That's where a **Readiness Probe** helps:

Don't send traffic until I'm ready.

Later, suppose the API server gets stuck (e.g., infinite loop, deadlock), even though the container is running.

Now, Liveness Probe helps:

If I'm broken, restart me.

DEMO: Simulating Liveness & Readiness Probes

Step 1: Create a basic Flask app with /health and /ready

```
python
CopyEdit
# app.py
from flask import Flask
import time
app = Flask(__name__)
start time = time.time()
@app.route('/ready')
def ready():
    # Simulates readiness only after 10 seconds
    if time.time() - start time > 10:
       return "Ready", 200
    else:
       return "Not Ready", 500
@app.route('/health')
def health():
    # Simulate failure after 30 seconds (crash/hang)
    if time.time() - start time < 30:
        return "Alive", 200
    else:
       return "Crashed", 500
@app.route('/')
def hello():
    return "Welcome to the Demo App", 200
if name == ' main
    app.run(host='0.0.0.0')
```

Step 2: Dockerfile

```
Dockerfile
CopyEdit
FROM python:3.9
WORKDIR /app
COPY app.py .
RUN pip install flask
CMD ["python", "app.py"]
```

☑ Build the Docker Image Locally

Assuming your files are:

- app.py (Flask app)
- Dockerfile

Run:

bash

docker build -t probe-demo: latest .

This creates a local image named probe-demo: latest.

V Load Image into kind Cluster

By default, kind has its own internal Docker registry, so we must load the image into it:

```
bash
CopyEdit
kind load docker-image probe-demo:latest -name <cluster name>
```

This tells kind: "Hey, here's a local image. Use it directly inside your cluster."

☑ Use the Same Image in Your YAML

In your probe-demo. yaml, reference this image exactly as you built it:

yaml

image: probe-demo:latest

Step 3: Kubernetes YAML with probes

```
yaml
CopyEdit
apiVersion: v1
kind: Pod
metadata:
 name: probe-demo
spec:
 containers:
  - name: demo-container
    image: your-dockerhub-user/probe-demo:latest
    - containerPort: 5000
    readinessProbe:
      httpGet:
        path: /ready
        port: 5000
      initialDelaySeconds: 2
      periodSeconds: 5
    livenessProbe:
      httpGet:
```

path: /health
 port: 5000
initialDelaySeconds: 5

periodSeconds: 10

Step 4: Observe Behavior

- Initially, pod will be in "Not Ready" state for 10 sec (no traffic).
- After 10 sec, it becomes "Ready".
- After 30 sec, /health fails → Kubernetes restarts the pod due to liveness failure.
- Then it repeats the cycle again.

To Test:

bash
CopyEdit
kubectl apply -f probe-demo.yaml
kubectl describe pod probe-demo
kubectl get pods
kubectl get pods -w

Once ready

2

Now traffic would be allowed if behind a service.

Watch for Liveness Failures and Restarts

After 30 seconds (from /health logic), the liveness probe starts failing, and Kubernetes **restarts** the container.

You'll see:

sql
CopyEdit
probe-demo 0/1 Running 1 31s

- The pod restarts (RESTARTS count increases to 1)
- It will again go to 0/1, wait for readiness, and then become 1/1
- This cycle repeats every ~30–40 seconds

3. Detailed Logs: Describe the Pod

bash

kubectl describe pod probe-demo

Look for:

- Readiness probe failed: in the Events
- Liveness probe failed: followed by Killing container
- Restart logs

4. View App Logs

See what your Flask app is logging:

bash

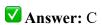
kubectl logs probe-demo

You'll notice:

- Initially the app saying "Not Ready"
- Then "Ready"
- Then "Alive"
- Finally, "Crashed" (after 30 sec) → followed by container restart

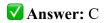
What is the purpose of a Readiness Probe?

- A. To check if the container is running
- **B.** To decide if a container should be restarted
- C. To check if the container is ready to receive traffic
- **D.** To check disk usage of the pod



What happens when a Liveness Probe fails?

- A. Pod is scaled down
- **B.** Pod is marked as Not Ready
- C. Kubernetes restarts the container
- **D.** Kubernetes stops scheduling other pods



Which of the following statements is true about readiness probes?

- **A.** They are only checked once when the pod starts
- **B.** If a readiness probe fails, the container is killed
- C. If a readiness probe fails, the pod is removed from service endpoints
- **D.** Readiness probes have no effect on service routing

Answer: C			