Kubernetes\_8\_Probes

Liveness probe

Readiness probe

Startup probe

Readiness probe: If your application is ready to receive traffic or not. If it fails, Kubernetes stops sending new connections to that particular pod and does not restart the Pod. (When to allow and when to stop the traffic to the Pods)

Liveness probe: If your app is alive and healthy. If it fails, Kubernetes will restart the container. But Readiness will not restart the Pod (When to restart the container when issue is not resolved)

ubuntu@ip-172-31-9-165:~/Probes$ cat readiness-live-manifest.yml

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: javawebdeploy

labels:

app: javawebapp

spec:

replicas: 3

selector:

matchLabels:

app: javawebapp

template:

metadata:

name: javawebpod

labels:

app: javawebapp

spec:

containers:

- name: javawebapp

image: hacker123shiva/springbt-in-docker:latest

ports:

- containerPort: 8080 # Your Spring Boot app listens on 8080

readinessProbe:

initialDelaySeconds: 30

periodSeconds: 5

timeoutSeconds: 10

successThreshold: 1

failureThreshold: 3

httpGet:

path: /

port: 8080 # Corrected: Probe must check the port your app is listening on

livenessProbe: # <--- ADD THIS SECTION

httpGet:

path: /

port: 8080

initialDelaySeconds: 45 # Give the app ample time to fully start before health checks begin

periodSeconds: 10 # Check every 10 seconds

timeoutSeconds: 5 # Consider a failure if no response within 5 seconds

successThreshold: 1

failureThreshold: 3

---

apiVersion: v1

kind: Service

metadata:

name: javawebapp-service # Name for your LoadBalancer Service

labels:

app: javawebapp

spec:

type: LoadBalancer # This exposes your service externally

selector:

app: javawebapp # This must match the 'app' label in your Deployment's template

ports:

- protocol: TCP

port: 80 # The port the LoadBalancer will listen on (e.g., standard HTTP port)

targetPort: 8080 # The port your Spring Boot application is running on inside the pod

# You can optionally specify a nodePort if needed, e.g., nodePort: 30080

...

initialDelaySeconds: 30 --> Initial 30 seconds the Pods will not start

periodSeconds: 5 --> Every 5 seconds Kubernetes will perform the readiness check

Suppose continuously if the pod is not ready then Kubernetes will only come to a conclusion that the pod is not ready and I should not allow traffic to this particular pod. Liveness will recreate the container while also redirecting the traffic.

ubuntu@ip-172-31-9-165:~/Probes$ kubectl apply -f readiness-live-manifest.yml

deployment.apps/javawebdeploy created

service/javawebapp-service created

ubuntu@ip-172-31-9-165:~/Probes$

ubuntu@ip-172-31-9-165:~/Probes$

ubuntu@ip-172-31-9-165:~/Probes$ kubectl get pods

NAME READY STATUS RESTARTS AGE

javawebdeploy-b8fd9679b-6wbdd 0/1 Running 0 40s

javawebdeploy-b8fd9679b-6xdbl 0/1 Running 0 40s

javawebdeploy-b8fd9679b-qht2b 0/1 Running 0 40s

ubuntu@ip-172-31-9-165:~/Probes$ kubectl delete all --all

pod "javawebdeploy-b8fd9679b-6wbdd" deleted

pod "javawebdeploy-b8fd9679b-6xdbl" deleted

pod "javawebdeploy-b8fd9679b-qht2b" deleted

service "javawebapp-service" deleted

service "kubernetes" deleted

deployment.apps "javawebdeploy" deleted

replicaset.apps "javawebdeploy-b8fd9679b" deleted

ubuntu@ip-172-31-9-165:~/Probes$ kubectl apply -f readiness-live-manifest.yml

deployment.apps/javawebdeploy created

service/javawebapp-service created

ubuntu@ip-172-31-9-165:~/Probes$ kubectl get pods -l app=javawebapp -w

NAME READY STATUS RESTARTS AGE

javawebdeploy-b8fd9679b-2c7t2 1/1 Running 0 64s

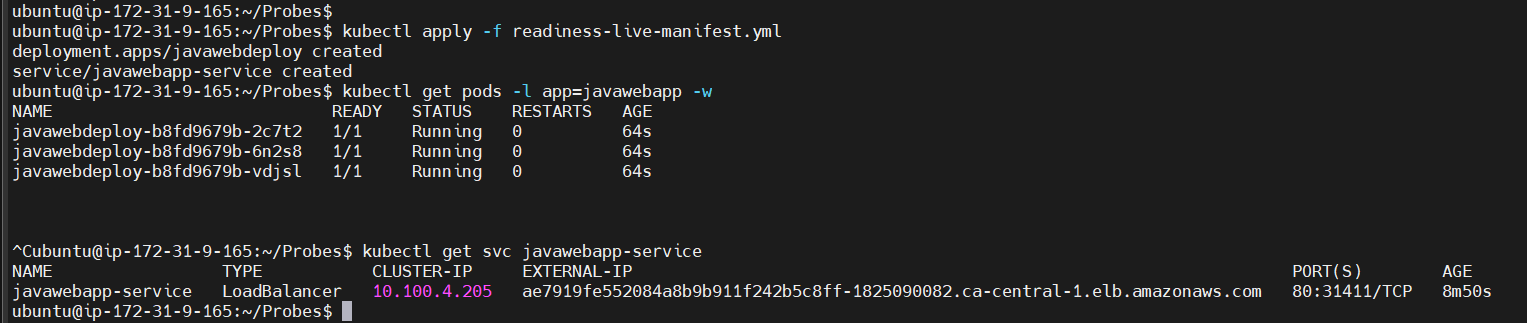
javawebdeploy-b8fd9679b-6n2s8 1/1 Running 0 64s

javawebdeploy-b8fd9679b-vdjsl 1/1 Running 0 64s

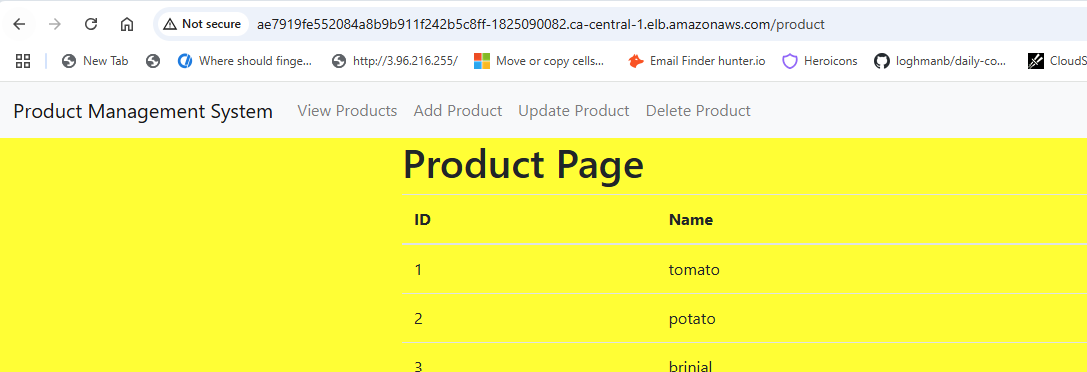
^Cubuntu@ip-172-31-9-165:~/Probes$ kubectl get svc javawebapp-service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

javawebapp-service LoadBalancer 10.100.4.205 ae7919fe552084a8b9b911f242b5c8ff-1825090082.ca-central-1.elb.amazonaws.com 80:31411/TCP 8m50s



ae7919fe552084a8b9b911f242b5c8ff-1825090082.ca-central-1.elb.amazonaws.com



Application is live without a problem

How to demonstrate Readiness probe?

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

javawebapp-service LoadBalancer 10.100.4.205 ae7919fe552084a8b9b911f242b5c8ff-1825090082.ca-central-1.elb.amazonaws.com 80:31411/TCP 8m50s

ubuntu@ip-172-31-9-165:~/Probes$

ubuntu@ip-172-31-9-165:~/Probes$

ubuntu@ip-172-31-9-165:~/Probes$

ubuntu@ip-172-31-9-165:~/Probes$ kubectl exec -it javawebdeploy-b8fd9679b-vdjsl -- /bin/sh

sh-4.4# kill 1

sh-4.4# command terminated with exit code 137

Pod is restarting

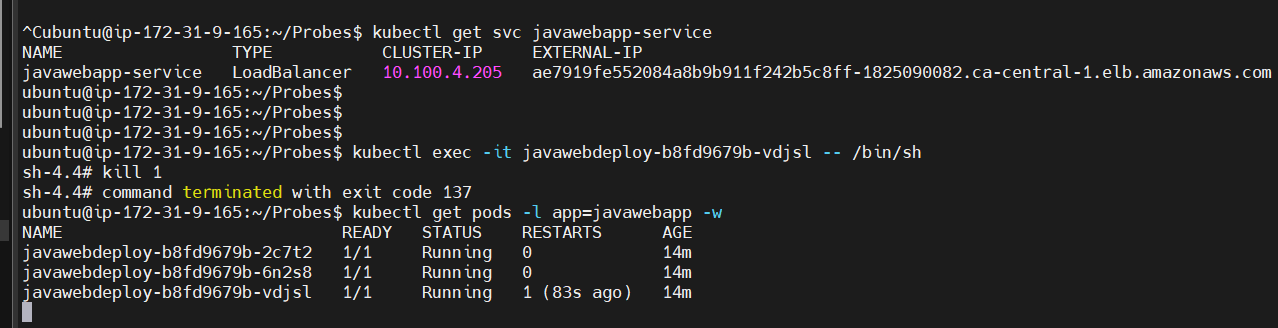
ubuntu@ip-172-31-9-165:~/Probes$ kubectl get pods -l app=javawebapp -w

NAME READY STATUS RESTARTS AGE

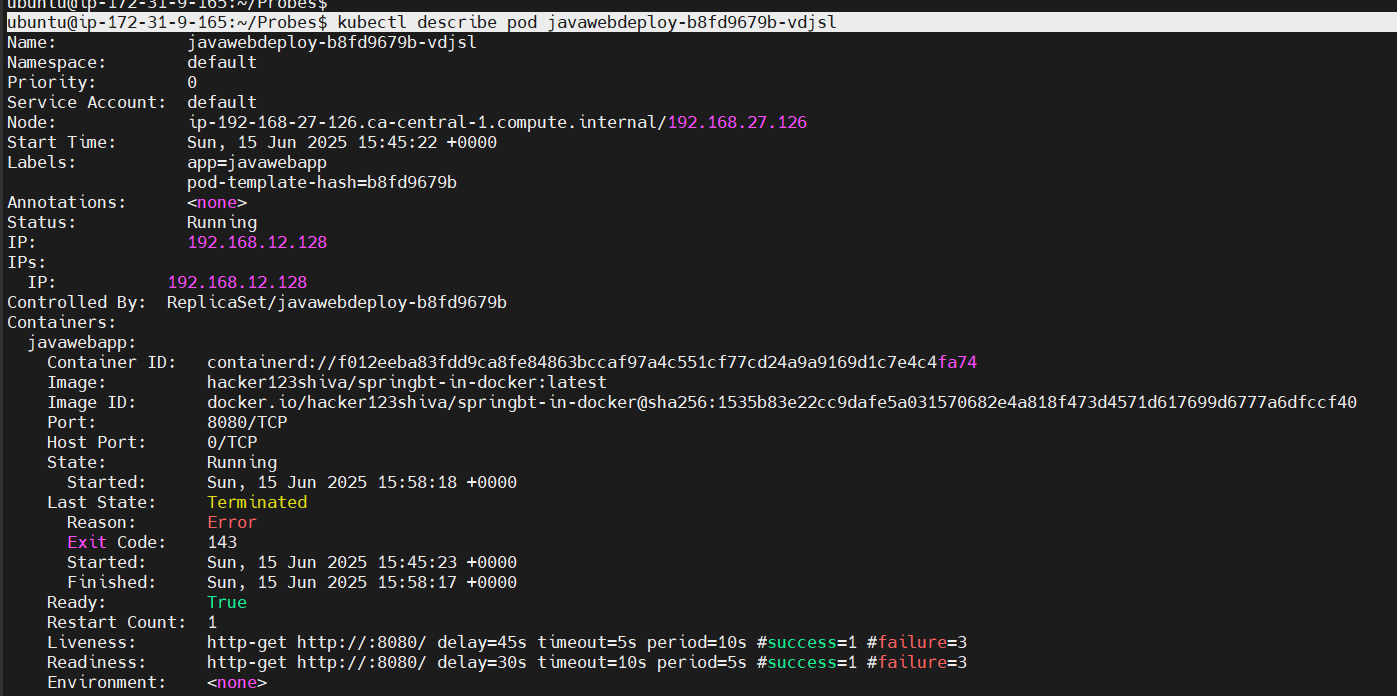
javawebdeploy-b8fd9679b-2c7t2 1/1 Running 0 14m

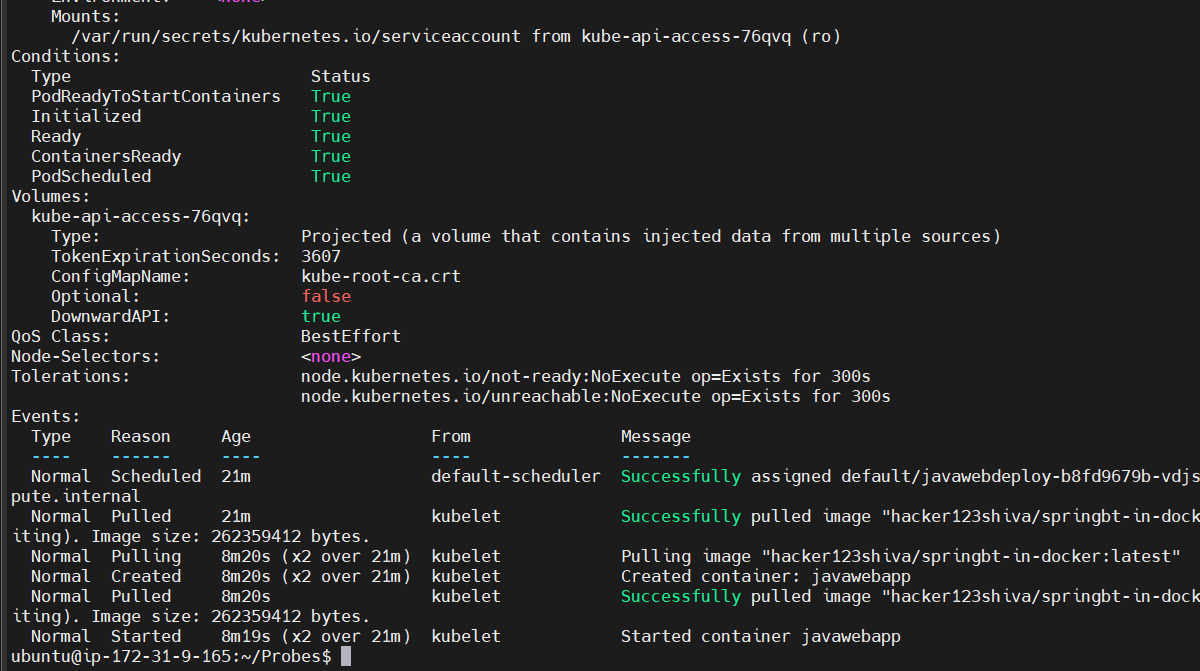
javawebdeploy-b8fd9679b-6n2s8 1/1 Running 0 14m

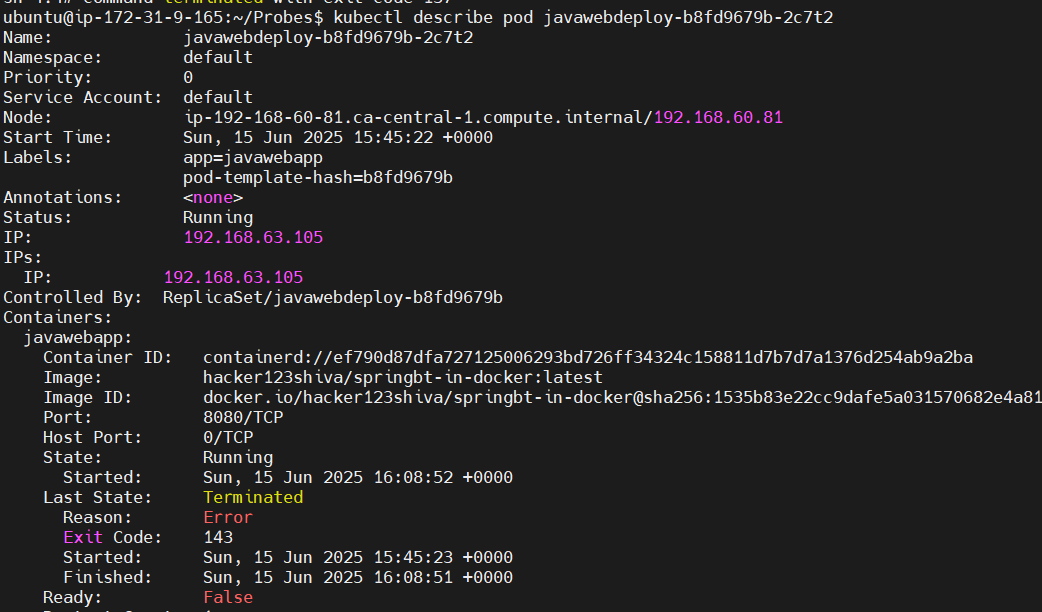
javawebdeploy-b8fd9679b-vdjsl 1/1 Running 1 (83s ago) 14m



ubuntu@ip-172-31-9-165:~/Probes$ kubectl describe pod javawebdeploy-b8fd9679b-vdjsl







ubuntu@ip-172-31-9-165:~/Probes$ kubectl exec -it javawebdeploy-b8fd9679b-2c7t2 -- /bin/sh

sh-4.4# kill 1

sh-4.4# command terminated with exit code 137

ubuntu@ip-172-31-9-165:~/Probes$ kubectl get pods -l app=javawebapp -w

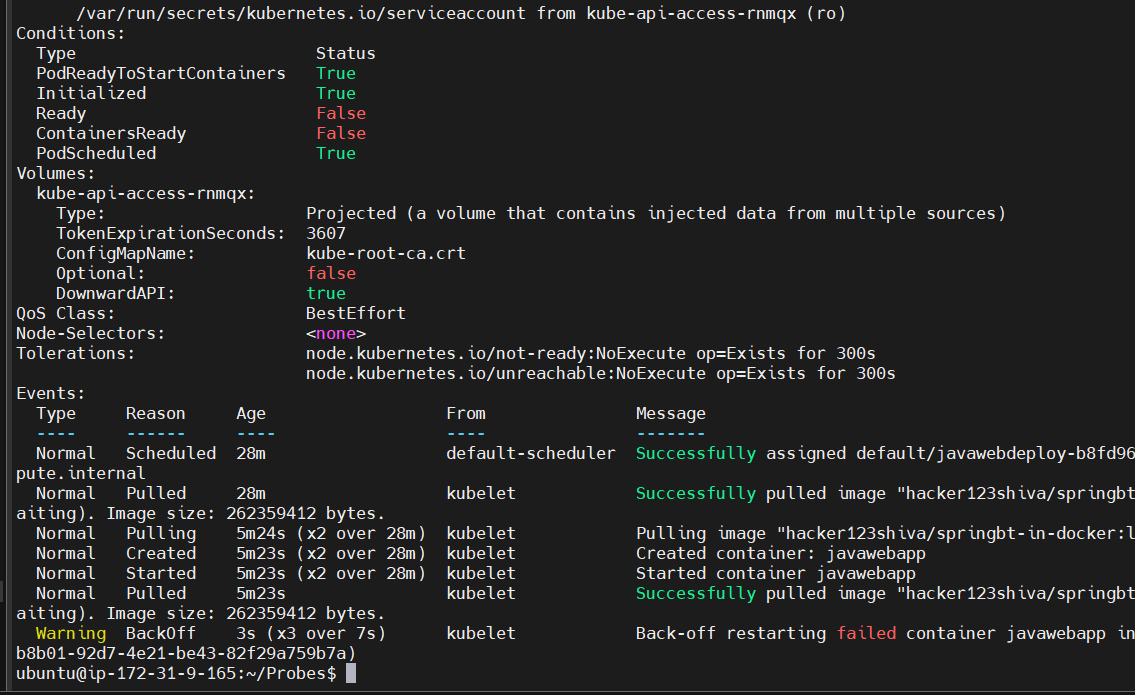
NAME READY STATUS RESTARTS AGE

javawebdeploy-b8fd9679b-2c7t2 1/1 Running 1 (107s ago) 25m

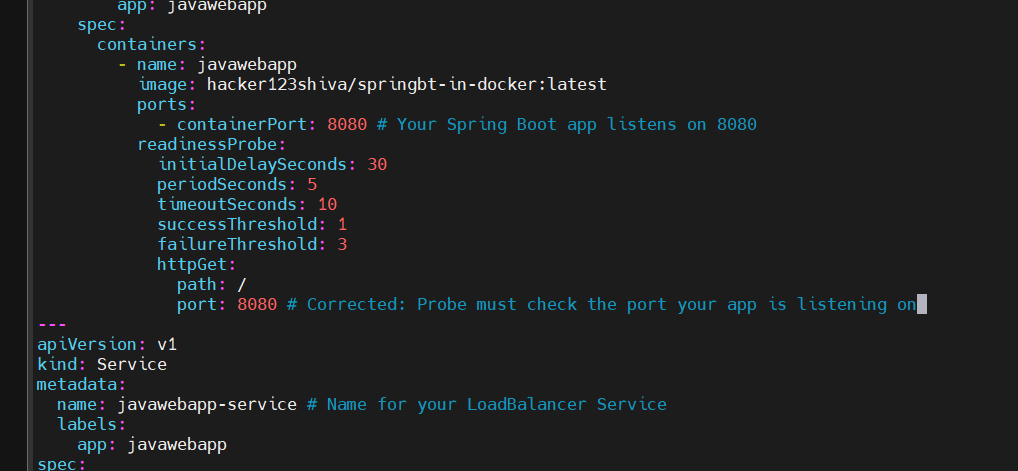
javawebdeploy-b8fd9679b-6n2s8 1/1 Running 0 25m

javawebdeploy-b8fd9679b-vdjsl 1/1 Running 1 (12m ago) 25m

I killed the pod twice, now I see it is restarting because of the Liveness probe



Removed liveness



ubuntu@ip-172-31-9-165:~/Probes$ kubectl apply -f readiness-live-manifest-v1.yml

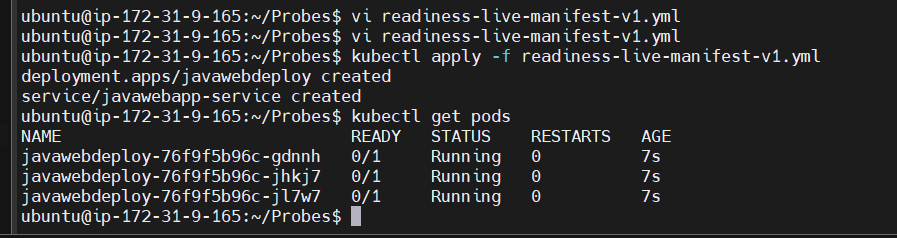
ubuntu@ip-172-31-9-165:~/Probes$ kubectl get pods

NAME READY STATUS RESTARTS AGE

javawebdeploy-76f9f5b96c-gdnnh 0/1 Running 0 7s

javawebdeploy-76f9f5b96c-jhkj7 0/1 Running 0 7s

javawebdeploy-76f9f5b96c-jl7w7 0/1 Running 0 7s



ubuntu@ip-172-31-9-165:~/Probes$ kubectl apply -f readiness-live-manifest-v1.yml

deployment.apps/javawebdeploy created

service/javawebapp-service created

ubuntu@ip-172-31-9-165:~/Probes$ kubectl get pods

NAME READY STATUS RESTARTS AGE

javawebdeploy-76f9f5b96c-gdnnh 0/1 Running 0 7s

javawebdeploy-76f9f5b96c-jhkj7 0/1 Running 0 7s

javawebdeploy-76f9f5b96c-jl7w7 0/1 Running 0 7s

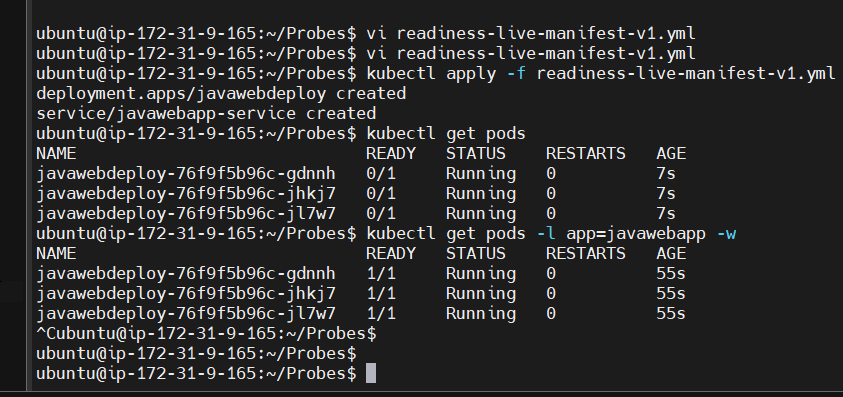
ubuntu@ip-172-31-9-165:~/Probes$ kubectl get pods -l app=javawebapp -w

NAME READY STATUS RESTARTS AGE

javawebdeploy-76f9f5b96c-gdnnh 1/1 Running 0 55s

javawebdeploy-76f9f5b96c-jhkj7 1/1 Running 0 55s

javawebdeploy-76f9f5b96c-jl7w7 1/1 Running 0 55s



ubuntu@ip-172-31-9-165:~/Probes$ kubectl get svc

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

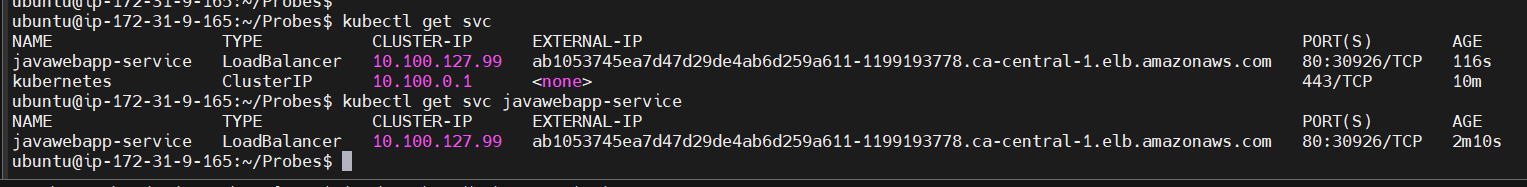
javawebapp-service LoadBalancer 10.100.127.99 ab1053745ea7d47d29de4ab6d259a611-1199193778.ca-central-1.elb.amazonaws.com 80:30926/TCP 116s

kubernetes ClusterIP 10.100.0.1 <none> 443/TCP 10m

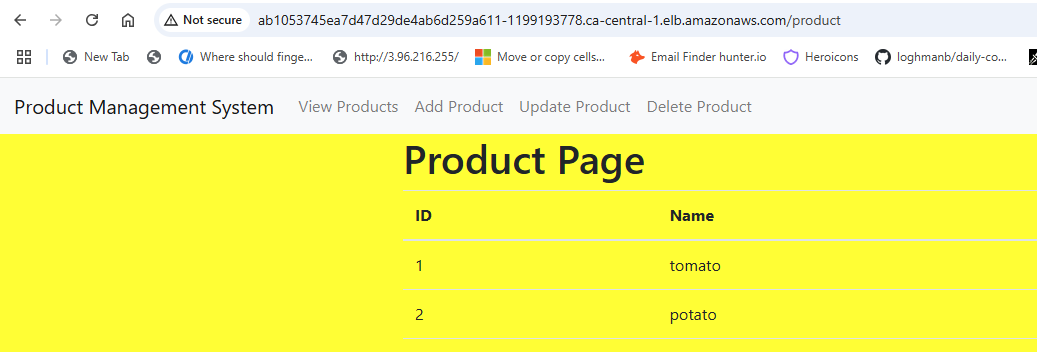
ubuntu@ip-172-31-9-165:~/Probes$ kubectl get svc javawebapp-service

NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

javawebapp-service LoadBalancer 10.100.127.99 ab1053745ea7d47d29de4ab6d259a611-1199193778.ca-central-1.elb.amazonaws.com 80:30926/TCP 2m10s



http://ab1053745ea7d47d29de4ab6d259a611-1199193778.ca-central-1.elb.amazonaws.com/product



Demonstrating readiness probe

In the event section, you will notice about warning and backoff which demonstrate the probes

Traffic reroute -> Readiness probe and Restart container -> Liveness probe

Readiness probe -> Failure action -> Traffic routing control -> stops sending new connections to not ready pod (to tell the service to direct traffic somewhere else)

Liveness probe -> Failure action -> Container restart -> It terminates and restarts container

ubuntu@ip-172-31-9-165:~/Probes$ kubectl apply -f readiness-live-manifest-v1.yml

deployment.apps/javawebdeploy created

service/javawebapp-service created

ubuntu@ip-172-31-9-165:~/Probes$ kubectl get pods -l app=javawebapp -w

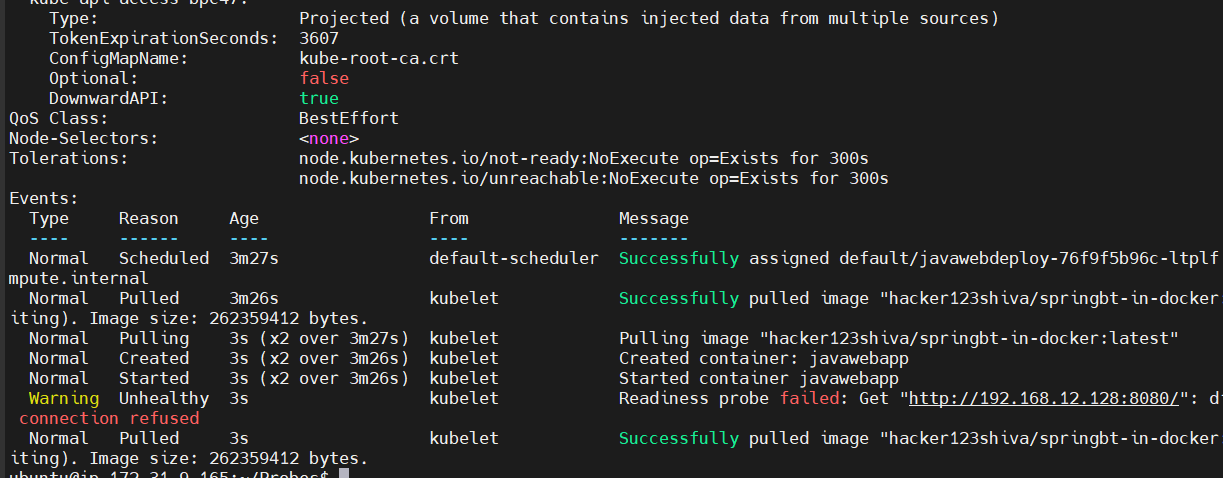
NAME READY STATUS RESTARTS AGE

javawebdeploy-76f9f5b96c-2fqt9 0/1 Running 0 14s

javawebdeploy-76f9f5b96c-b82bg 0/1 Running 0 14s

javawebdeploy-76f9f5b96c-ltplf 0/1 Running 0 14s

^Cubuntu@ip-172-31-9-165:~/Probes$



Normal Started 3s (x2 over 3m26s) kubelet Started container javawebapp

Warning Unhealthy 3s kubelet Readiness probe failed: Get "http://192.168.12.128:8080/": dial tcp 192.168.12.128:8080: connect: connection refused

Readiness probe, connection is refused

eksctl delete cluster --name my-eks-cluster --region ca-central-1