Health Checks

Introduction:

Kubernetes uses health check probes to monitor the health of the application or containers or pods.

Kubelet plays an important role in checking the health which is using the below given methods.

1. **Liveness Probe:** if there is a deadlock in the Kubernetes cluster and container is not able to restart automatically then Kubelet uses this probe to restart the pod. This probe is responsible for container. If it fails then pod will be restated.
   Best practices: Only include basic checks in the liveness probe. Never include checks on connections to other services (e.g. database).
2. **Readiness Probe:** We can consider it a doorkeeper for the incoming traffic. Kubelet uses this probe if the pod is not ready to receive the traffic, then delete it from the service. This probe is responsible for the pod. It does not restart the Pod; it will remove the pod from the service only.
   Best practices: Include all necessary checks including connections to other services. Nevertheless, the check shouldn't take too long to complete. Always specify a Readiness Probe to make sure that the pod only gets traffic, if the pod can properly handle incoming requests.
3. **Startup Probe**: Kubelet uses this probe to check the health of the application. it is responsible for the application running in the docker container.if the application is not working then there is no point of keeping a container. so keep liveness and readiness disabled.if application start working then start liveness and readiness probe. Startup Probes check, when the pod is available after startup.
   Best practices: Specify a Startup Probe, if the pod takes a long time to start. The Startup and Liveness Probe can use the same endpoint, but the Startup Probe will have a less strict failure threshhold which prevents a failure on startup

   All the above probes are configured using **HTTP**, **TCP** and **Command**.

   Objectives:

   1. Create deployments using a probe

1. Create deployments using a probe

We are going to create a deployment with a probe configured. Use the below Yaml file.

```
vi health-deploy.yaml
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-frontend-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: myapp-container
        image: nginx
        livenessProbe:
          httpGet:
            path: /
            port: 80
          initialDelaySeconds: 10
          timeoutSeconds: 5
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-frontend-app2
spec:
  replicas: 1
  selector:
    matchLabels:
      app: frontend
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
      - name: myapp-container
        image: nginx
        livenessProbe:
          httpGet:
            path: /abc
            port: 80
          initialDelaySeconds: 10
          timeoutSeconds: 5
```

As you can see in above Yaml file, we are going to create two deployments with liveness probe configured. In first deployment, the probe is sending HTTPGet request to / which would be a success whereas in the second deployment, the probe is sending HTTPGet request to /abc which is likely to fail and pod will be restated in this case.

Apply the above Yaml and check the output below.

```
root@master:~/manifest# vi health-deploy.yaml
root@master:~/manifest#
root@master:~/manifest# kubectl apply -f health-deploy.yaml
deployment.apps/my-frontend-app created
deployment.apps/my-frontend-app2 created
root@master:~/manifest#
root@master:~/manifest# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
my-frontend-app-64675f7f99-9n2qv    1/1     Running   0          8s
my-frontend-app2-85cfd5f7d9-qmhzw   1/1     Running   0          8s
root@master:~/manifest#
root@master:~/manifest# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
my-frontend-app-64675f7f99-9n2qv    1/1     Running   0          50s
my-frontend-app2-85cfd5f7d9-qmhzw   1/1     Running   1          50s
root@master:~/manifest# kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
my-frontend-app-64675f7f99-9n2qv    1/1     Running   0          74s
my-frontend-app2-85cfd5f7d9-qmhzw   1/1     Running   2          74s
root@master:~/manifest#
```

In above output, we can see the second pod is getting restated over and over again.