Although Kubernetes is able to automatically restart containers by default if their main processes crash, sometimes more sophisticated error detection is needed in order to take full advantage of self-healing functionality. In this lesson, you will learn how to implement a basic liveness probe in Kubernetes that will periodically verify that the application is responding correctly to requests. After completing this lesson, you will have a basic understanding of what liveness probes can do and how to implement them.

For more information on Kubernetes liveness probes, check out their documentation: https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-probes/.

Here is the final Kubernetes template file used in the demo (train-schedule-kube.yml):

```
kind: Service
apiVersion: v1
metadata:
  name: train-schedule-service
  annotations:
    prometheus.io/scrape: 'true'
spec:
  type: NodePort
  selector:
    app: train-schedule
  ports:
  - protocol: TCP
    port: 8080
    nodePort: 8080

---

apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: train-schedule-deployment
  labels:
    app: train-schedule
spec:
  replicas: 2
  selector:
    matchLabels:
      app: train-schedule
  template:
    metadata:
      labels:
        app: train-schedule
    spec:
      containers:
      - name: train-schedule
        image: linuxacademycontent/train-schedule:selfhealing
        ports:
        - containerPort: 8080
        livenessProbe:
          httpGet:
            path: /
            port: 8080
          initialDelaySeconds: 15
          timeoutSeconds: 1
          periodSeconds: 10
```

You can find the sample source code here: https://github.com/linuxacademy/cicd-pipeline-train-schedule-selfhealing