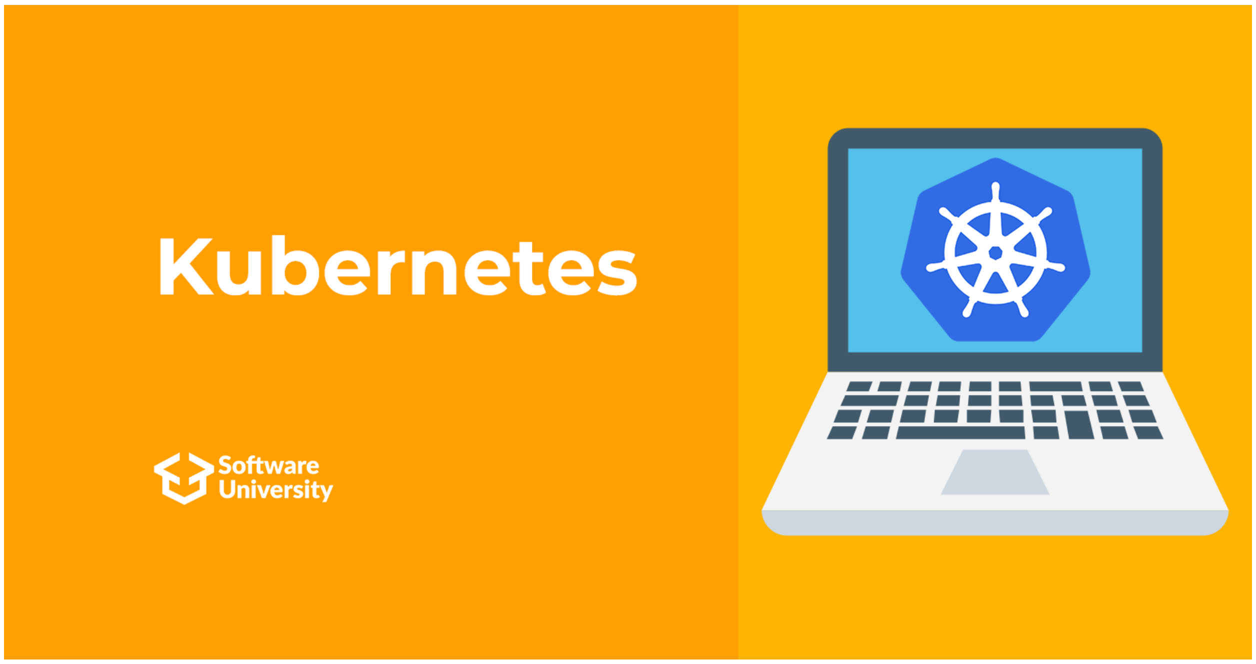


Kubernetes

June 2022

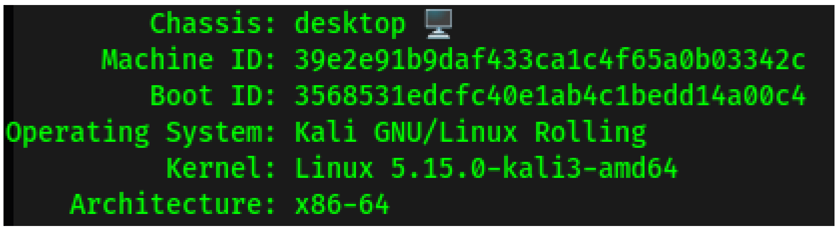
Observability and Troubleshooting

Home Work

Stefan Veselinov

Environment:

Host



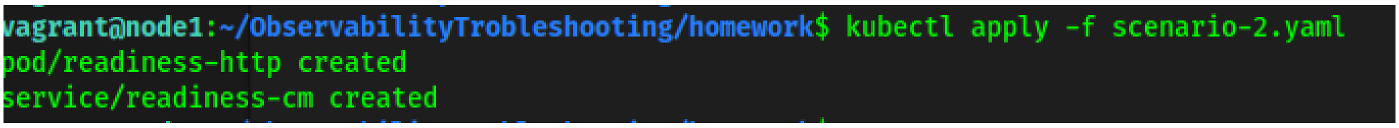
```
         Chassis: desktop 🖥️
      Machine ID: 39e2e91b9daf433ca1c4f65a0b03342c
         Boot ID: 3568531edcfc40e1ab4c1bedd14a00c4
Operating System: Kali GNU/Linux Rolling
          Kernel: Linux 5.15.0-kali3-amd64
    Architecture: x86-64
```

Tasks Solution

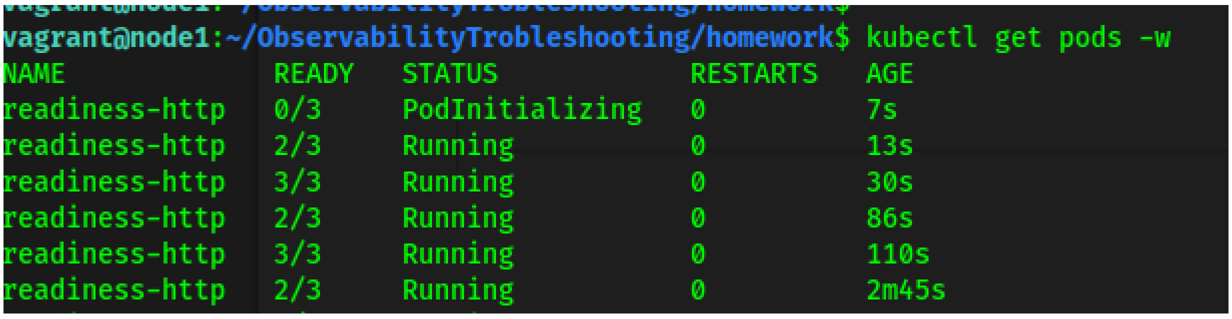
1. Try to solve **scenario 2** and make the application working again

scenario-2.yaml

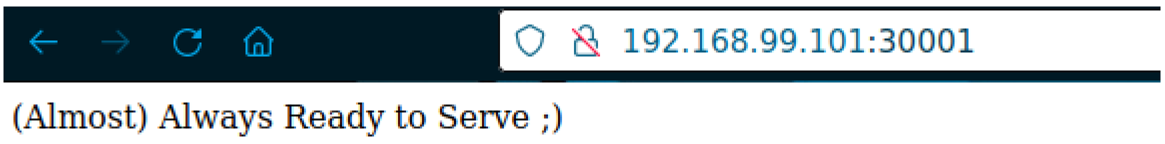
```
apiVersion: v1
kind: Pod
metadata:
  labels:
    app: readiness-http
  name: readiness-http
spec:
  initContainers:
  - name: init-data
    image: alpine
    command: ["/bin/bash", "-c"] # must be "/bin/sh"
    args:
      - echo '(Almost) Always Ready to Serve' ;) > /data/index.html # must be – quota after serve must be
after ;)
    volumeMounts:
    - name: data
      mountPath: /data
  containers:
  - name: cont-main
    image: nginx
    volumeMounts:
    - name: data
      mountPath: /usr/share/nginx/html
    readinessProbe:
      httpGet:
        path: /healthy.html
        port: 80
      initialDelaySeconds: 5
      periodSeconds: 5
  - name: cont-sidecar-postpone
    image: alpine
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          sleep 20;
          echo 'WORKING' > /check/healthy.html;
        sleep 60;
      done
    volumeMounts:
    - name: html # must be data
      mountPath: /check
  - name: cont-sidecar-break
    image: alpine
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
        sleep 60;
        rm /check/healthy.html;
        sleep 20;
      done
    volumeMounts:
    - name: data
      mountPath: /check
  volumes:
  - name: data
    emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: readiness-cm
  labels:
    app: readiness-cmd
spec:
  type: NodePort
  ports:
  - port: 80
    nodePort: 30001
    protocol: TCP
  selector:
    app: readiness-cmd # must be readiness-http
```



```
vagrant@node1:~/ObservabilityTrobleshooting/homework$ kubectl apply -f scenario-2.yaml
pod/readiness-http created
service/readiness-cm created
```



```
vagrant@node1:~/ObservabilityTrobleshooting/homework$ kubectl get pods -w
NAME             READY   STATUS            RESTARTS   AGE
readiness-http   0/3     PodInitializing   0          7s
readiness-http   2/3     Running           0          13s
readiness-http   3/3     Running           0          30s
readiness-http   2/3     Running           0          86s
readiness-http   3/3     Running           0          110s
readiness-http   2/3     Running           0          2m45s
```

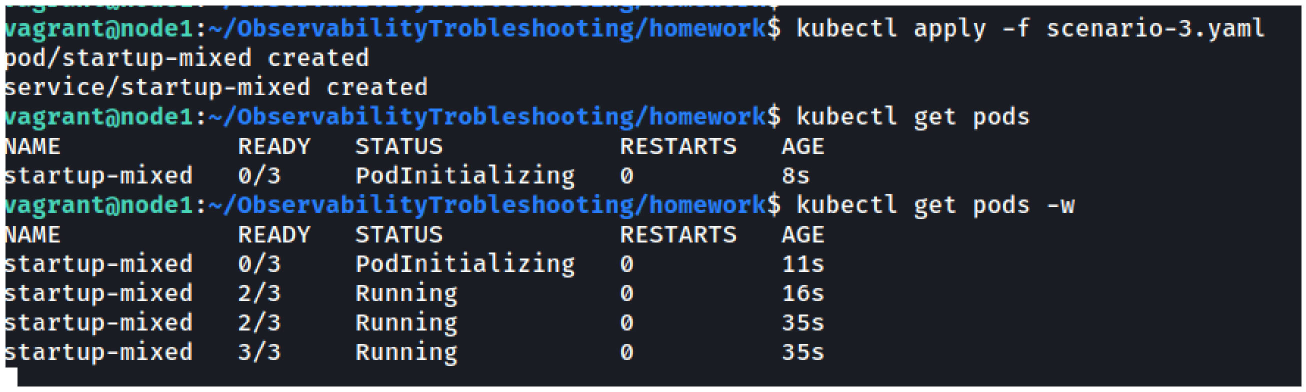

192.168.99.101:30001

(Almost) Always Ready to Serve ;)

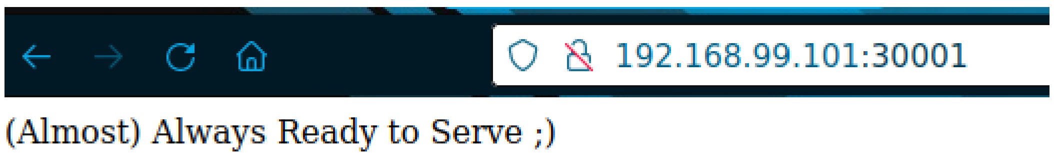
2. Try to solve **scenario 3** and make the application working again
   scenario-3.yaml

```
apiVersion: v1
kind: Pod
metadata:
 labels:
  app: startup-mixed
 name: startup-mixed
spec:
 initContainers:
 - name: init-data
  image: alpine
  command: ["/bin/sh", "-c"]
  args:
   - echo '(Almost) Always Ready to Serve ;)' > /data/index.html
  volumeMounts:
  - name: data
   mountPath: /data
 containers:
 - name: cont-main
  image: nginx
  volumeMounts:
  - name: data
   mountPath: /usr/share/nginx/html
  livenessProbe:
   httpGet:
    path: /check/healthy.html # must be /healthy.html
    port: 80
   initialDelaySeconds: 5
   periodSeconds: 5
  startupProbe:
   exec:
    command:
    - cat
    - /check/healthy.html # must be usrshare/nginx/healthy.html
   failureThreshold: 3 # make it 10
   periodSeconds: 5
 - name: cont-sidecar-postpone
  image: alpine
```

```
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          sleep 20;
          echo 'WORKING' > /check/healthy.html;
          sleep 60;
        done
    volumeMounts:
    - name: data
      mountPath: /check
  - name: cont-sidecar-break
    image: alpine
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          sleep 60;
          rm /check/healthy.html;
          sleep 20;
        done
    volumeMounts:
    - name: data
      mountPath: /check
  volumes:
  - name: data
    emptyDir: {}
---
apiVersion: v1
kind: Service
metadata:
  name: startup-mixed
  labels:
    app: startup-mixed
spec:
  type: NodePort
  ports:
  - port: 8080 # must be 80
    nodePort: 30001
    protocol: TCP
  selector:
    app: startup-nixed # must be startup-mixed
```



```
vagrant@node1:~/ObservabilityTrobleshooting/homework$ kubectl apply -f scenario-3.yaml
pod/startup-mixed created
service/startup-mixed created
vagrant@node1:~/ObservabilityTrobleshooting/homework$ kubectl get pods
NAME            READY   STATUS            RESTARTS   AGE
startup-mixed   0/3     PodInitializing   0          8s
vagrant@node1:~/ObservabilityTrobleshooting/homework$ kubectl get pods -w
NAME            READY   STATUS            RESTARTS   AGE
startup-mixed   0/3     PodInitializing   0          11s
startup-mixed   2/3     Running           0          16s
startup-mixed   2/3     Running           0          35s
startup-mixed   3/3     Running           0          35s
```


192.168.99.101:30001

(Almost) Always Ready to Serve ;)

3. *Try to solve **scenario 4** and make the application working again*

pvss.yaml

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvssa
  labels:
    purpose: ssdemo
spec:
  capacity:
    storage: 10Gi # 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  mountOptions:
    - nfsvers=4.1
  nfs:
    path: /data/nfs/k8spva
    server: nfs-server
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvssb
  labels:
    purpose: ssdemo
spec:
  capacity:
    storage: 1Mi # must be 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadOnly # must be ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  mountOptions:
    - nfsvers=4.1
  nfs:
    path: /data/nfs/k8spvb
    server: nfs-server
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pvssc
  labels:
    purpose: ssdemo
spec:
  capacity:
    storage: 1Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Recycle
  mountOptions:
    - nfsvers=4.1
  nfs:
    path: /bata/nfs/k8spvc # must be /data/nfs/k8spvc
    server: nfs-server
```

svcss.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: facts
spec:
  selector:
    app: factc # must be facts
  clusterIP: None
  ports:
  - port: 5000
    protocol: TCP
```

svcssnp.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: factsnp
spec:
  selector:
    app: fact # must be facts
  type: ClusterIP # must be NodePort
  ports:
  - port: 5000
    nodePort: 30001
    protocol: TCP
```