Exercise: Pods

Pods are the smallest, most basic deployable objects in Kubernetes. A Pod represents a single instance of a running process in your cluster. Pods contain one or more containers, such as Docker containers. Although you want deploy pods directly (static pods), knowledge for defining pods manifest files will be used for defining more complex Kubernetes resources like Controllers.

Practice1: Simple pods operations

Note: Try not to do a copy/paste on commands requests unless you are instructed to do so. Copy/paste will not help you to learn Kubernetes!

1. Login to Azure and connect to your AKS cluster.

We created an Azure Kubernetes Service cluster by navigating to 'Kubernetes Services' and selecting 'Create'. The process for creating the cluster is similar to that for a virtual machine. After creating the cluster, we used the following commands to connect to it from Azure CLI.

```
PS /home/vladimir> az account set --subscription c031718c-8f31-4c8d-ae56-634904ec50f9
PS /home/vladimir> az aks get-credentials --resource-group kuber --name Test
Merged "Test" as current context in /home/vladimir/.kube/config
PS /home/vladimir>
```

2. Check how many pods run under the default namespace. Run **kubectl get pods**.

```
PS /home/vladimir> kubectl get pods
No resources found in default namespace.
```

3. You should not see any pod under the default namespace. Now check all namespaces. Run **kubectl get pods –all-namespace**.

```
PS /home/vladimir> kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   azure-ip-masq-agent-lvdvx             1/1     Running   0          13m
kube-system   azure-ip-masq-agent-vzkwj             1/1     Running   0          13m
kube-system   cloud-node-manager-85pxp              1/1     Running   0          13m
kube-system   cloud-node-manager-ffklf              1/1     Running   0          13m
kube-system   coredns-59b6bf8b4f-4rlln              1/1     Running   0          14m
kube-system   coredns-59b6bf8b4f-bvncj              1/1     Running   0          12m
kube-system   coredns-autoscaler-5655d66f64-z7ts2   1/1     Running   0          14m
kube-system   csi-azuredisk-node-6sssk              3/3     Running   0          13m
kube-system   csi-azuredisk-node-c9f2d              3/3     Running   0          13m
kube-system   csi-azurefile-node-gb2cf              3/3     Running   0          13m
kube-system   csi-azurefile-node-z8ssh              3/3     Running   0          13m
kube-system   konnectivity-agent-7f75c5f96f-6xxf4   1/1     Running   0          14m
kube-system   konnectivity-agent-7f75c5f96f-p9ktc   1/1     Running   0          12m
kube-system   kube-proxy-gwd78                      1/1     Running   0          13m
kube-system   kube-proxy-wpvnp                      1/1     Running   0          13m
kube-system   metrics-server-8655f897d8-lf67v       2/2     Running   0          12m
kube-system   metrics-server-8655f897d8-rtnlz       2/2     Running   0          12m
```

4. How many pods do you see? Who deployed these pods? Why are they deployed?

There are 17 pods that were deployed automatically by the Azure Kubernetes Service (AKS) when we created our cluster. They are part of the Kubernetes infrastructure and perform various tasks that are essential for the proper functioning of our cluster. Here is a brief description of each of the pods:

- azure-ip-masq-agent: This pod is responsible for configuring IP masquerading rules to enable network connectivity between our Kubernetes pods and the external network.
- cloud-node-manager: This pod manages the lifecycle of the worker nodes in our cluster and ensures that the correct number of nodes is available at all times.
- coredns: This is a DNS server that provides name resolution for our Kubernetes services and pods.
- coredns-autoscaler: This pod scales the number of coredns replicas based on the number of nodes in our cluster.
- csi-azuredisk-node: These pods provide support for the Azure Disk storage class in our cluster.
- csi-azurefile-node: These pods provide support for the Azure File storage class in our cluster.
- konnectivity-agent: This pod provides secure connectivity between our cluster and other Kubernetes clusters or services running on different networks.
- kube-proxy: This is a network proxy that runs on each worker node and enables communication between the pods and services in our cluster.
- metrics-server: This pod collects resource utilization data (CPU, memory, etc.) for our cluster and provides it to other Kubernetes components and tools.

5. Now deploy you first pod using the imperative approach. Run **kubectl run nginx --image=nginx**.

```
PS /home/vladimir> kubectl run nginx --image=nginx
pod/nginx created
```

6. Validate if the pods has been created. What is the status of your pod?

```
PS /home/vladimir> kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          24s
```

7. Check the logs coming out of your pod. Run **kubectl logs nginx**.

```
PS /home/vladimir> kubectl logs nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2023/04/03 21:46:31 [notice] 1#1: using the "epoll" event method
2023/04/03 21:46:31 [notice] 1#1: nginx/1.23.4
2023/04/03 21:46:31 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/04/03 21:46:31 [notice] 1#1: OS: Linux 5.4.0-1104-azure
2023/04/03 21:46:31 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/04/03 21:46:31 [notice] 1#1: start worker processes
2023/04/03 21:46:31 [notice] 1#1: start worker process 29
2023/04/03 21:46:31 [notice] 1#1: start worker process 30
2023/04/03 21:46:31 [notice] 1#1: start worker process 31
2023/04/03 21:46:31 [notice] 1#1: start worker process 32
```

8. Run following command to check current resource consumption of your pod: **kubectl top pod nginx**.

```
PS /home/vladimir> kubectl top pod nginx
NAME    CPU(cores)   MEMORY(bytes)
nginx   0m           5Mi
```

9. Check on which Node your pods has been scheduled. Run **kubectl get pods –o wide**.

```
PS /home/vladimir> kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE    IP           NODE                                NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          6m5s   10.244.0.8   aks-agentpool-24052243-vmss000001   <none>           <none>
```

10. Try to find the same information but this time running **kubectl describe pod nginx.**

```
PS /home/vladimir> kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             aks-agentpool-24052243-vmss000001/10.224.0.5
Start Time:       Mon, 03 Apr 2023 21:46:28 +0000
Labels:           run=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.8
```

11. Delete your pod using **kubectl delete pod nginx.**

```
PS /home/vladimir> kubectl delete pod nginx
pod "nginx" deleted
PS /home/vladimir>
```

12. Let’s find the image used on one of the coredns pods under the kube-system namespace.

13. Once again list all pods under all namespaces.

```
PS /home/vladimir> kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   azure-ip-masq-agent-lvdvx             1/1     Running   0          46m
kube-system   azure-ip-masq-agent-vzkwj             1/1     Running   0          46m
kube-system   cloud-node-manager-85pxp              1/1     Running   0          46m
kube-system   cloud-node-manager-ffklf              1/1     Running   0          46m
kube-system   coredns-59b6bf8b4f-4rlln              1/1     Running   0          46m
kube-system   coredns-59b6bf8b4f-bvncj              1/1     Running   0          45m
kube-system   coredns-autoscaler-5655d66f64-z7ts2   1/1     Running   0          46m
kube-system   csi-azuredisk-node-6sssk              3/3     Running   0          46m
kube-system   csi-azuredisk-node-c9f2d              3/3     Running   0          46m
kube-system   csi-azurefile-node-gb2cf              3/3     Running   0          46m
kube-system   csi-azurefile-node-z8ssh              3/3     Running   0          46m
kube-system   konnectivity-agent-7f75c5f96f-6xxf4   1/1     Running   0          46m
kube-system   konnectivity-agent-7f75c5f96f-p9ktc   1/1     Running   0          45m
kube-system   kube-proxy-gwd78                      1/1     Running   0          46m
kube-system   kube-proxy-wpvnp                      1/1     Running   0          46m
kube-system   metrics-server-8655f897d8-lf67v       2/2     Running   0          44m
kube-system   metrics-server-8655f897d8-rtnlz       2/2     Running   0          44m
```

14. Note one of the coredns pods. Now run **kubectl describe pod <coredns-name> -n kube-system**. Replace the <coredns-name> place holder with noted name.

```
PS /home/vladimir> kubectl describe pod coredns-59b6bf8b4f-bvncj -n kube-system
```

15. Inspect the output and locate the image information.

```
PS /home/vladimir> kubectl describe pod coredns-59b6bf8b4f-bvncj -n kube-system
Name:                 coredns-59b6bf8b4f-bvncj
Namespace:            kube-system
Priority:             2000001000
Priority Class Name:  system-node-critical
Service Account:      coredns
Node:                 aks-agentpool-24052243-vmss000000/10.224.0.4
Start Time:           Mon, 03 Apr 2023 21:12:40 +0000
Labels:               k8s-app=kube-dns
                      kubernetes.io/cluster-service=true
                      pod-template-hash=59b6bf8b4f
                      version=v20
Annotations:          prometheus.io/port: 9153
Status:               Running
IP:                   10.244.1.3
IPs:
  IP:           10.244.1.3
Controlled By:  ReplicaSet/coredns-59b6bf8b4f
Containers:
  coredns:
    Container ID:  containerd://50a15d64ca6b5b4add8f808b2120ea4c601d673ee8c93566586f22aa60de7bcb
    Image:         mcr.microsoft.com/oss/kubernetes/coredns:v1.9.3
    Image ID:      sha256:c38f956b642366c8eeb0babfda6b0bb2aa92f27a968589804cadb445f6df72d6
```

16. Now let us check the logs of the metrics-server pod. Run the same command as in step 7 but don't forget to add the namespace in which this pod is created.

"kubectl logs metrics-server-8655f897d8-rtnlz -n kube-system -c metrics-server"

-**c flag** is used to specify the container name

-n flag is used to specify the namespace of the pod

This will retrieve the logs only for the **metrics-server** container running in the **metrics-server-8655f897d8-rtnlz** pod in the **kube-system** namespace.

```
PS /home/vladimir> kubectl logs metrics-server-8655f897d8-rtnlz -n kube-system -c metrics-server
I0403 21:13:05.603746       1 serving.go:342] Generated self-signed cert (/tmp/apiserver.crt, /tmp/apiserver.key)
I0403 21:13:11.404567       1 requestheader_controller.go:169] Starting RequestHeaderAuthRequestController
I0403 21:13:11.404590       1 shared_informer.go:240] Waiting for caches to sync for RequestHeaderAuthRequestController
I0403 21:13:11.404598       1 secure_serving.go:266] Serving securely on [::]:4443
I0403 21:13:11.404616       1 configmap_cafile_content.go:201] "Starting controller" name="client-ca::kube-system::extension-a
I0403 21:13:11.404637       1 dynamic_serving_content.go:131] "Starting controller" name="serving-cert::/tmp/apiserver.crt::/t
I0403 21:13:11.404643       1 shared_informer.go:240] Waiting for caches to sync for client-ca::kube-system::extension-apiserv
I0403 21:13:11.404658       1 configmap_cafile_content.go:201] "Starting controller" name="client-ca::kube-system::extension-a
I0403 21:13:11.404662       1 shared_informer.go:240] Waiting for caches to sync for client-ca::kube-system::extension-apiserv
I0403 21:13:11.404992       1 tlsconfig.go:240] "Starting DynamicServingCertificateController"
W0403 21:13:11.405106       1 shared_informer.go:372] The sharedIndexInformer has started, run more than once is not allowed
I0403 21:13:11.603781       1 shared_informer.go:247] Caches are synced for RequestHeaderAuthRequestController
I0403 21:13:11.603902       1 shared_informer.go:247] Caches are synced for client-ca::kube-system::extension-apiserver-auther
I0403 21:13:11.705205       1 shared_informer.go:247] Caches are synced for client-ca::kube-system::extension-apiserver-auther
```

Practice2: Working with pod manifest files

Note: Try not to do a copy/paste on commands requests unless you are instructed to do so. Copy/paste will not help you to learn Kubernetes!

1. Now it is time to deploy pod using manifest file (declarative approach). Copy the following code block on your local computer in a file called redis.yaml:

```
apiVersion: v11
kind: pod
metadata:
name: static-web
labels:
role: myrole
specs:
containers:
- name: redis
image: redis123
```

2. Try to deploy the pod defined in redis.yaml. Run **kubectl create –f redis.yaml**.

```
PS C:\Users\V&M\Desktop\lab> kubectl create -f redis.yaml
error: resource mapping not found for name: "" namespace: "" from "redis.yaml": no matches for kind "pod" in version "v11"
ensure CRDs are installed first
PS C:\Users\V&M\Desktop\lab>
```

3. You will receive errors on your screen. Your next task will be to correct the syntax of the code you just copied. You can use the online Kubernetes documentation or you can search the internet in general.

- v11 is not a valid API version. The correct API version for creating Pods is v1
- specs should be spec
- wrong identation

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: static-web
5    labels:
6      role: myrole
7  spec:
8    containers:
9    - name: redis
10     image: redis123
```

4. When you solve all the syntax errors your pod should be deployed but is it running? What is the status of your pod? The status is ImagePullBackOff.

```
PS C:\Users\V&M\Desktop\lab> kubectl create -f redis.yaml
pod/static-web created
PS C:\Users\V&M\Desktop\lab> kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
static-web    0/1     ImagePullBackOff   0          27s
PS C:\Users\V&M\Desktop\lab>
```

5. Check the events associated with this pod. Run the **kubectl describe pod static-web** command. What are the events showing? Why your pod is not running?

The error messages indicates that the image redis123 could not be pulled from the specified container registry.

```
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  93s                default-scheduler  Successfully assigned default/static-web to docker-desktop

  Warning  Failed     45s (x3 over 91s)  kubelet            Failed to pull image "redis123": rpc error: code = Unknown
 desc = Error response from daemon: pull access denied for redis123, repository does not exist or may require 'docker
login': denied: requested access to the resource is denied
  Warning  Failed     45s (x3 over 91s)  kubelet            Error: ErrImagePull
  Normal   BackOff    15s (x4 over 90s)  kubelet            Back-off pulling image "redis123"
  Warning  Failed     15s (x4 over 90s)  kubelet            Error: ImagePullBackOff
  Normal   Pulling    2s (x4 over 93s)   kubelet            Pulling image "redis123"
```

6. Find the correct image (check the Docker hub page) and correct it in the manifest.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: static-web
  labels:
    role: myrole
spec:
  containers:
  - name: redis
    image: asterixlegaulois/redis123
```

7. Locate the image information and put the correct image name. Redeploy the pod (fist run **kubectl delete pod static-web** to delete the pod, then run kubectl create once again).

```
PS C:\Users\V&M\Desktop\lab> kubectl delete pod static-web
pod "static-web" deleted
PS C:\Users\V&M\Desktop\lab> kubectl create -f redis.yaml
pod/static-web created
PS C:\Users\V&M\Desktop\lab> kubectl get pods
NAME          READY   STATUS    RESTARTS   AGE
static-web    1/1     Running   0          7s
PS C:\Users\V&M\Desktop\lab>
```

8. Check the status of your pod. It should be running now. Running

9. Now you can delete the pod. Try to delete it using the **kubectl delete –f redis.yaml**.

```
PS C:\Users\V&M\Desktop\lab> kubectl delete -f redis.yaml
pod "static-web" deleted
PS C:\Users\V&M\Desktop\lab>
```

10. Your next task is to create and test nginx pod definition. Your definition should use the nginx official image, should use label named app with value frontend and should publish port 80. Make sure you complete this task because we will use this template in our next Labs. Your nginx pod should be running without any issues.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: frontend
spec:
  containers:
  - name: nginx-container
    image: nginx
    ports:
    - containerPort: 80
```

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  77s   default-scheduler  Successfully assigned default/nginx-pod to docker-desktop
  Normal  Pulling    77s   kubelet            Pulling image "nginx"
  Normal  Pulled     63s   kubelet            Successfully pulled image "nginx" in 13.57627594s
  Normal  Created    63s   kubelet            Created container nginx-container
  Normal  Started    63s   kubelet            Started container nginx-container
```

11. Final task of this practice will be to define pod definition with following details:

- Image=memcached
- Port= 11211
- Label app=web
- CPU request=0.35 cores
- RAM request=0.15 GB
- CPU limit=0.5 cores
- Ram limit=0.25 GB
- Restart policy=Never

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: memcached-web
  labels:
    app: web
spec:
  containers:
  - name: memcached
    image: memcached
    ports:
    - containerPort: 11211
    resources:
      requests:
        cpu: "0.35"
        memory: "0.15Gi"
      limits:
        cpu: "0.5"
        memory: "0.25Gi"
    imagePullPolicy: IfNotPresent
  restartPolicy: Never
```

12. Don’t forget to try your pod definition.

```
PS C:\Users\V&M\Desktop\lab> kubectl create -f memcached.yaml
pod/memcached-web created
PS C:\Users\V&M\Desktop\lab> kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
memcached-web   1/1     Running   0          15s
nginx-pod       1/1     Running   0          7m7s
PS C:\Users\V&M\Desktop\lab>
```

Practice3: Multi-container pods

Note: Try not to do a copy/paste on commands requests unless you are instructed to do so. Copy/paste will not help you to learn Kubernetes!

1. Once finished you can try to create multi-container pod definition. Your multi-container pod should use redis and nginx containers with port 6379 and 80 published respectively. Label name should be app with value web.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp
  labels:
    app: web
spec:
  containers:
  - name: redis
    image: asterixlegaulois/redis123
    ports:
    - containerPort: 6379
  - name: nginx-container
    image: nginx
    ports:
    - containerPort: 80
```

2. Note that in reality there is no sense to put the redis and nginx under the same pod but it can be done for the purpose of learning.

3. Deploy your multi-container pod. It should have running status. What is written under Ready column when you kubectl get the pods? Why your pod displays different values for ready? Under Ready column is written 2/2 because in this pod we have two containers.

```
PS C:\Users\V&M\Desktop\lab> kubectl get pods
NAME            READY   STATUS    RESTARTS   AGE
memcached-web   1/1     Running   0          16m
nginx-pod       1/1     Running   0          28m
webapp          2/2     Running   0          66s
PS C:\Users\V&M\Desktop\lab>
```

4. Kubectl describe your new pod, and locate the containers section. How many containers are listed? Two.

```
Containers:
  redis:
    Container ID:   docker://e2630b2c8b28c2339972f5dfbbddf6d77e9cbbcc36e908e59bc1e4238589e58a
    Image:          asterixlegaulois/redis123
    Image ID:       docker-pullable://asterixlegaulois/redis123@sha256:cb2ddf11373c66e8c66ea7cb4
ddd82bd69849d4838ae41ee0e71a0bc5c6cc4c4
    Port:           6379/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 04 Apr 2023 13:08:44 +0200
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-8zgfw (ro)
  nginx-container:
    Container ID:   docker://12a72152b265220b56f0bc81f61f78265c0c97a76a53c04a94aea93ee4833f5a
    Image:          nginx
    Image ID:       docker-pullable://nginx@sha256:2ab30d6ac53580a6db8b657abf0f68d75360ff5cc1670
a85acb5bd85ba1b19c0
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Tue, 04 Apr 2023 13:08:45 +0200
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-8zgfw (ro)
```

5. Delete all the pods under the default namespace.

```
PS C:\Users\V&M\Desktop\lab> kubectl delete pods --all
>>
pod "memcached-web" deleted
pod "nginx-pod" deleted
pod "webapp" deleted
PS C:\Users\V&M\Desktop\lab> _
```

6. Don’t delete any of the manifest files you have created so far.

Practice4: Probes

Note: Try not to do a copy/paste on commands requests unless you are instructed to do so. Copy/paste will not help you to learn Kubernetes!

1. First we will create and test liveness probe with exec test. Create a file named probes_exec.yaml with following content:

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
name: liveness-exec
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/busybox
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    livenessProbe:
      exec:
      command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
```

2. Examine the containers args commands especially the line that start with touch. This bash pipeline will help us to test the liveness probes.

The container runs a shell command that touches a file named /tmp/healthy, sleeps for 30 seconds, removes the file and then sleeps for 600 seconds.

3. Run **kubectl create –f probes_exec.yaml**.

```
PS C:\Users\V&M\Desktop\lab> kubectl create -f probes_exec.yaml
pod/liveness-exec created
```

4. Run **kubectl describe pod liveness-exec** immediately after you deploy the pod. The output should indicate that no liveness probes have failed yet.

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  15s   default-scheduler  Successfully assigned default/liveness-exec to doc
ker-desktop
  Normal  Pulling    14s   kubelet            Pulling image "k8s.gcr.io/busybox"
  Normal  Pulled     12s   kubelet            Successfully pulled image "k8s.gcr.io/busybox" in
2.11243698s
  Normal  Created    12s   kubelet            Created container liveness
  Normal  Started    12s   kubelet            Started container liveness
```

5. After 35 seconds, view the Pod events again. Run **kubectl describe pod liveness-exec.**

6. At the bottom of the output, there should be a messages indicating that the liveness probes have failed, and the containers have been killed and recreated.

```
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  68s                default-scheduler  Successfully assigned default/live
ness-exec to docker-desktop
  Normal   Pulling    67s                kubelet            Pulling image "k8s.gcr.io/busybox"

  Normal   Pulled     65s                kubelet            Successfully pulled image "k8s.gcr
.io/busybox" in 2.11243698s
  Normal   Created    65s                kubelet            Created container liveness
  Normal   Started    65s                kubelet            Started container liveness
  Warning  Unhealthy  23s (x3 over 33s)  kubelet            Liveness probe failed: cat: can't
open '/tmp/healthy': No such file or directory
  Normal   Killing    23s                kubelet            Container liveness failed liveness
 probe, will be restarted
```

7. Wait another 30 seconds, and verify that the container has been restarted. Run **kubectl get pod liveness- exec.**

8. The output should show that RESTARTS has been incremented.

```
PS C:\Users\V&M\Desktop\lab> kubectl get pod liveness-exec
NAME            READY   STATUS    RESTARTS      AGE
liveness-exec   1/1     Running   1 (55s ago)   2m10s
PS C:\Users\V&M\Desktop\lab>
```

9. We will continue with HTTP probe. Create file named probes_http.yaml with following content:

```yaml
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
name: liveness-http
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
    httpGet:
      path: /healthz
      port: 8080
      httpHeaders:
      - name: Custom-Header
        value: Awesome
    initialDelaySeconds: 3
    periodSeconds: 3
```

10. Just for your info, /healtz handler has following function implemented:

```
http.HandleFunc("/healthz", func(w http.ResponseWriter, r *http.Request) {
   duration := time.Now().Sub(started)
   if duration.Seconds() > 10 {
      w.WriteHeader(500)
      w.Write([]byte(fmt.Sprintf("error: %v", duration.Seconds())))
   } else {
      w.WriteHeader(200)
      w.Write([]byte("ok"))
   }
})
```

11. For the first 10 seconds that the container is alive, the /healthz handler returns a status of 200. After that, the handler returns a status of 500.

12. Run **kubectl create –f probes_http.yaml.**

13. Immediately run (you only have 10 secs to run this command) **kubectl describe pod liveness-http.**

14. Your pod should be live and running.

```
PS C:\Users\V&M\Desktop\lab> kubectl create -f probes_http.yaml
pod/liveness-http created
PS C:\Users\V&M\Desktop\lab> kubectl describe pod liveness-http
Name:             liveness-http
Namespace:        default
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.4
Start Time:       Tue, 04 Apr 2023 13:37:15 +0200
Labels:           test=liveness
Annotations:      <none>
Status:           Running
IP:               10.1.0.18
```

15. After 10 seconds, view Pod events to verify that liveness probes have failed and the container has been restarted. Run again **kubectl describe pod liveness-http.**

```
Events:
  Type     Reason     Age                  From               Message
  ----     ------     ----                 ----               -------
  Normal   Scheduled  80s                  default-scheduler  Successfully assigned defau
lt/liveness-http to docker-desktop
  Normal   Pulled     78s                  kubelet            Successfully pulled image "
k8s.gcr.io/liveness" in 1.216349349s
  Normal   Pulled     60s                  kubelet            Successfully pulled image "
k8s.gcr.io/liveness" in 1.144994584s
  Normal   Created    42s (x3 over 78s)    kubelet            Created container liveness
  Normal   Started    42s (x3 over 78s)    kubelet            Started container liveness
  Normal   Pulled     42s                  kubelet            Successfully pulled image "
k8s.gcr.io/liveness" in 1.182641358s
  Normal   Pulling    26s (x4 over 79s)    kubelet            Pulling image "k8s.gcr.io/l
iveness"
  Warning  Unhealthy  26s (x9 over 68s)    kubelet            Liveness probe failed: HTTP
 probe failed with statuscode: 500
  Normal   Killing    26s (x3 over 62s)    kubelet            Container liveness failed l
iveness probe, will be restarted
```

16. You should see the same output as in step 7. Kubelet will reboot he container.

17. We continue with TCP probes. Create file named probes_tcp.yaml with following content:

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: liveness-tcp
  labels:
    app: goproxy
spec:
  containers:
  - name: goproxy
    image: k8s.gcr.io/goproxy:0.1
    ports:
    - containerPort: 8080
    livenessProbe:
    tcpSocket:
       port: 9999 #8080 is valid port
     initialDelaySeconds: 15
     periodSeconds: 20
```

18. Run **kubectl create –f probes_tcp.yaml.**

19. Immediately run (you only have 10 secs to run this command) **kubectl describe pod liveness-tcp.**

20. Your pod should be live and running.

```
PS C:\Users\V&M\Desktop\lab> kubectl create -f probes_tcp.yaml
pod/liveness-tcp created
PS C:\Users\V&M\Desktop\lab> kubectl describe pod liveness-tcp
Name:             liveness-tcp
Namespace:        default
Priority:         0
Service Account:  default
Node:             docker-desktop/192.168.65.4
Start Time:       Tue, 04 Apr 2023 13:58:20 +0200
Labels:           app=goproxy
Annotations:      <none>
Status:           Running
IP:               10.1.0.19
```

21. After 10 seconds, view Pod events to verify that liveness probes have failed and the container has been restarted. Run again **kubectl describe pod liveness-tcp.**

```
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  74s                default-scheduler  Successfully assigned defa
ult/liveness-tcp to docker-desktop
  Normal   Pulling    73s                kubelet            Pulling image "k8s.gcr.io/
goproxy:0.1"
  Normal   Pulled     71s                kubelet            Successfully pulled image
"k8s.gcr.io/goproxy:0.1" in 2.233846916s
  Normal   Created    13s (x2 over 70s)  kubelet            Created container goproxy
  Normal   Started    13s (x2 over 70s)  kubelet            Started container goproxy
  Warning  Unhealthy  13s (x3 over 53s)  kubelet            Liveness probe failed: dia
l tcp 10.1.0.19:9999: connect: connection refused
  Normal   Killing    13s                kubelet            Container goproxy failed l
iveness probe, will be restarted
```

22. You should see the same output as in step 7 and 16. Kubelet will reboot he container.

23. Our last job will be to define one readiness probe using HTTP test.
24. Create file named readiness_http.yaml with following content:

```
apiVersion: v1
kind: Pod
metadata:
  name: readiness-http
  labels:
  app: test
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
    - containerPort: 80
    readinessProbe:
    initialDelaySeconds: 1
    periodSeconds: 2
    timeoutSeconds: 1
    successThreshold: 1
    failureThreshold: 1
    httpGet:
    host:
    scheme: HTTP
    path: /
    httpHeaders:
    - name: Host
    value: myapplication1.com
    port: 80
```

25. Run **kubectl create –f readiness_http.yaml**.
26. Run **kubectl get pods –A** to see the status of your pod.
27. Pods and their status and ready states will be displayed; our pod should be in running state.

```
PS C:\Users\V&M\Desktop\lab> kubectl create -f readiness_http.yaml
pod/readiness-http created
PS C:\Users\V&M\Desktop\lab> kubectl get pods -A
NAMESPACE     NAME                                     READY   STATUS             RESTARTS         AGE
default       liveness-exec                            0/1     CrashLoopBackOff   13 (56s ago)     37m
default       liveness-http                            0/1     CrashLoopBackOff   13 (3m46s ago)   28m
default       liveness-tcp                             0/1     CrashLoopBackOff   5 (70s ago)      7m11s
default       readiness-http                           1/1     Running            0                7s
kube-system   coredns-565d847f94-f45t7                 1/1     Running            0                120m
kube-system   coredns-565d847f94-sjcxj                 1/1     Running            0                120m
kube-system   etcd-docker-desktop                      1/1     Running            1                120m
kube-system   kube-apiserver-docker-desktop            1/1     Running            1                120m
kube-system   kube-controller-manager-docker-desktop   1/1     Running            1                120m
kube-system   kube-proxy-6h7fw                         1/1     Running            0                120m
kube-system   kube-scheduler-docker-desktop            1/1     Running            1                120m
kube-system   storage-provisioner                      1/1     Running            0                119m
kube-system   vpnkit-controller                        1/1     Running            9 (7m8s ago)     119m
```

28. Run **kubectl describe pod readiness-http**. Examine the events for this pod. Everything should be OK.

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  110s  default-scheduler  Successfully assigned default
/readiness-http to docker-desktop
  Normal  Pulling    109s  kubelet            Pulling image "nginx"
  Normal  Pulled     107s  kubelet            Successfully pulled image "ng
inx" in 1.48232177s
  Normal  Created    107s  kubelet            Created container nginx
  Normal  Started    107s  kubelet            Started container nginx
```

29. Now delete the pod and edit the readiness_http.yaml so that the port parameter has 81 value.

```
PS C:\Users\V&M\Desktop\lab> kubectl delete pod readiness-http
pod "readiness-http" deleted
```

```yaml
readinessProbe:
  initialDelaySeconds: 1
  periodSeconds: 2
  timeoutSeconds: 1
  successThreshold: 1
  failureThreshold: 1
  httpGet:
    host:
    scheme: HTTP
    path: /
    httpHeaders:
    - name: Host
      value: myapplication1.com
    port: 81
```

30. Run again **kubectl create –f readiness_http.yaml.**

31. Run **kubectl get pods –A** to see the status of your pod. You should see that the pod is running but it is not in ready state.

```
PS C:\Users\V&M\Desktop\lab> kubectl create -f readiness_http.yaml
pod/readiness-http created
PS C:\Users\V&M\Desktop\lab> kubectl get pods -A
NAMESPACE     NAME                                     READY   STATUS             RESTARTS         AGE
default       liveness-exec                            0/1     CrashLoopBackOff   13 (4m46s ago)   41m
default       liveness-http                            0/1     CrashLoopBackOff   15 (111s ago)    32m
default       liveness-tcp                             0/1     CrashLoopBackOff   7 (80s ago)      11m
default       readiness-http                           0/1     Running            0                10s
kube-system   coredns-565d847f94-f45t7                 1/1     Running            0                123m
kube-system   coredns-565d847f94-sjcxj                 1/1     Running            0                123m
kube-system   etcd-docker-desktop                      1/1     Running            1                123m
kube-system   kube-apiserver-docker-desktop            1/1     Running            1                124m
kube-system   kube-controller-manager-docker-desktop   1/1     Running            1                123m
kube-system   kube-proxy-6h7fw                         1/1     Running            0                123m
kube-system   kube-scheduler-docker-desktop            1/1     Running            1                124m
kube-system   storage-provisioner                      1/1     Running            0                123m
kube-system   vpnkit-controller                        1/1     Running            10 (2m55s ago)   123m
```

32. Describe the pod. Run **kubectl describe pod readiness-http.**

33. From the events we can see that readiness probe failed due to the connection being refused therefore pod will not receive any traffic.

```
Events:
  Type     Reason     Age                From               Message
  ----     ------     ----               ----               -------
  Normal   Scheduled  55s                default-scheduler  Successfully assigned de
fault/readiness-http to docker-desktop
  Normal   Pulling    54s                kubelet            Pulling image "nginx"
  Normal   Pulled     53s                kubelet            Successfully pulled imag
e "nginx" in 1.48674243s
  Normal   Created    53s                kubelet            Created container nginx
  Normal   Started    52s                kubelet            Started container nginx
  Warning  Unhealthy  13s (x21 over 51s)  kubelet           Readiness probe failed:
Get "http://10.1.0.21:81/": dial tcp 10.1.0.21:81: connect: connection refused
```

34. Delete all pods under the default namespace.

```
PS C:\Users\V&M\Desktop\lab> kubectl delete pods --all -n default
>>
pod "liveness-exec" deleted
pod "liveness-http" deleted
pod "liveness-tcp" deleted
pod "readiness-http" deleted
PS C:\Users\V&M\Desktop\lab>
```

35. Don’t delete any manifest files created so far.