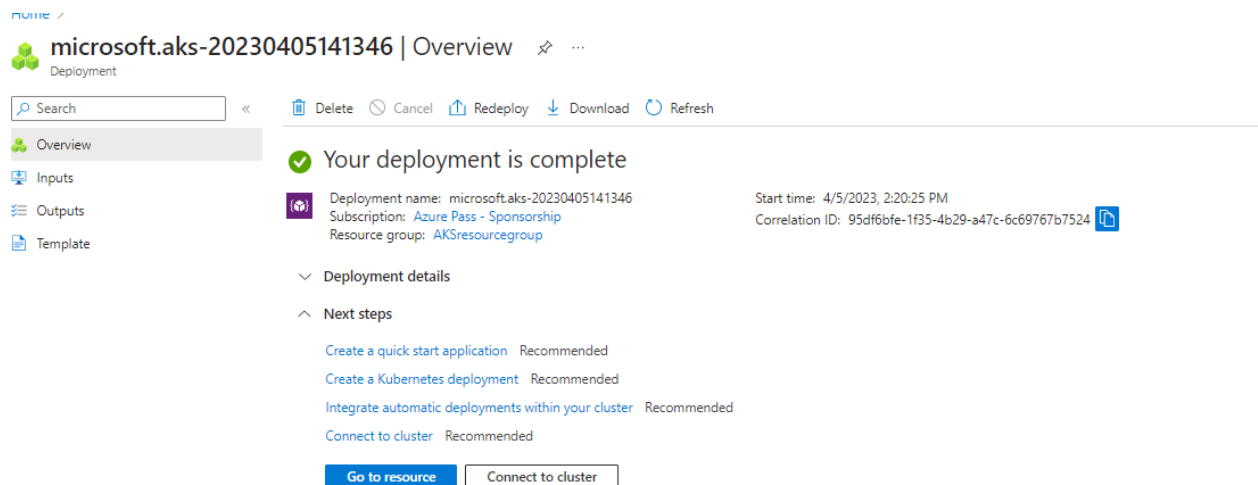


Kubernetes Pods Lab

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.



The screenshot shows the Azure portal interface for a deployment named 'microsoft.aks-20230405141346'. The deployment is in a 'complete' state, indicated by a green checkmark. The page displays deployment details such as the name, subscription (Azure Pass - Sponsorship), and resource group (AKSresourcegroup). It also shows the start time (4/5/2023, 2:20:25 PM) and a correlation ID. Under the 'Next steps' section, there are four recommended actions: 'Create a quick start application', 'Create a Kubernetes deployment', 'Integrate automatic deployments within your cluster', and 'Connect to cluster'. The 'Connect to cluster' button is highlighted.

- First, we need to create a Kubernetes cluster and connect to it via the PowerShell built-in plugin on Azure or Azure CLI. I chose the first option.

```
VERBOSE: Building your Azure drive ...
PS /home/andrijana> Import-AzAksCredential -ResourceGroupName AKSresourcegroup -Name myCluster

Confirm
Do you want to import the Kubernetes config?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): y
PS /home/andrijana> kubectl get nodes

NAME                                STATUS    ROLES    AGE     VERSION
aks-agentpool-35156179-vmss000000  Ready    agent    4m5s    v1.24.10
aks-agentpool-35156179-vmss000001  Ready    agent    3m53s    v1.24.10
aks-agentpool-35156179-vmss000002  Ready    agent    3m59s    v1.24.10
```

- I've logged in with the credentials of the resourceGroup name and the name of the cluster

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

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

- No pods are found in the default namespace, and this is expected behavior since we haven't deployed any.

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

```
PS /home/andrijana> kubectl get pods --all-namespaces
NAMESPACE   NAME                                     READY   STATUS    RESTARTS   AGE
kube-system  ama-logs-b2n8k                         2/2     Running   0           7m22s
kube-system  ama-logs-hdf7r                         2/2     Running   0           7m28s
kube-system  ama-logs-rs-7df6dd884-xkx94           1/1     Running   0           7m41s
kube-system  ama-logs-xrcwf                         2/2     Running   0           7m16s
kube-system  azure-ip-masq-agent-hkrtz             1/1     Running   0           7m28s
kube-system  azure-ip-masq-agent-j9ncm             1/1     Running   0           7m16s
kube-system  azure-ip-masq-agent-qg6rz             1/1     Running   0           7m22s
kube-system  cloud-node-manager-6jq4d              1/1     Running   0           7m16s
kube-system  cloud-node-manager-bqggv              1/1     Running   0           7m22s
kube-system  cloud-node-manager-jrhsm              1/1     Running   0           7m28s
kube-system  coredns-59b6bf8b4f-ntrz8              1/1     Running   0           7m40s
kube-system  coredns-59b6bf8b4f-zq2tr              1/1     Running   0           6m20s
kube-system  coredns-autoscaler-5f9cb57949-clzcb    1/1     Running   0           7m40s
kube-system  csi-azuredisk-node-4xtx5              3/3     Running   0           7m16s
kube-system  csi-azuredisk-node-v2zsv              3/3     Running   0           7m28s
kube-system  csi-azuredisk-node-wqshs              3/3     Running   0           7m22s
kube-system  csi-azurefile-node-bjllv              3/3     Running   0           7m16s
kube-system  csi-azurefile-node-g4v4s              3/3     Running   0           7m22s
kube-system  csi-azurefile-node-md5dx              3/3     Running   0           7m28s
kube-system  konnectivity-agent-c88dd5cff-t6hsb    1/1     Running   0           7m40s
kube-system  konnectivity-agent-c88dd5cff-x9hj7    1/1     Running   0           7m40s
kube-system  kube-proxy-5xqzd                      1/1     Running   0           7m16s
kube-system  kube-proxy-h2nn2                      1/1     Running   0           7m22s
kube-system  kube-proxy-hzk5n                      1/1     Running   0           7m28s
kube-system  metrics-server-5f8d84558d-2nnzd       2/2     Running   0           6m18s
kube-system  metrics-server-5f8d84558d-kqx6s       2/2     Running   0           6m18s
PS /home/andrijana>
```

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

- I can see 26 pods, deployed by the Azure Kubernetes Service automatically.

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

```
PS /home/andrijana> 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/andrijana> kubectl describe pods
Name:          nginx
Namespace:     default
Priority:       0
Service Account: default
Node:          aks-agentpool-35156179-vmss000000/10.224.0.6
Start Time:    Wed, 05 Apr 2023 12:33:48 +0000
Labels:        run=nginx
Annotations:    <none>
Status:        Running
IP:            10.244.0.13
IPs:
  IP: 10.244.0.13
```

- The pod **has been created** and the status of the pod is **Running**

```
Normal Started 23s kubelet Started container nginx
PS /home/andrijana> kubectl describe pods nginx
Name:          nginx
Namespace:     default
Priority:       0
Service Account: default
Node:          aks-agentpool-35156179-vmss000000/10.224.0.6
Start Time:    Wed, 05 Apr 2023 12:33:48 +0000
Labels:        run=nginx
Annotations:    <none>
Status:        Running
IP:            10.244.0.13
IPs:
  IP: 10.244.0.13
Containers:
  nginx:
    Container ID:  containerd://5807fdfcebfe6f70680754414ae192849eb5725589cef7b729be20d62d2499fe
    Image:         nginx
    Image ID:      docker.io/library/nginx@sha256:2ab30d6ac53580a6db8b657abf0f68d75360ff5cc1670a85acb5bd85ba1b19c0
    Port:          <none>
    Host Port:     <none>
    State:         Running
      Started:     Wed, 05 Apr 2023 12:33:51 +0000
    Ready:         True
    Restart Count: 0
    Environment:   <none>
```

- I can also use the command `kubectl describe pods nginx` to get the same result

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

```
PS /home/andrijana> 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
2023/04/05 12:33:51 [notice] 1#1: using the "epoll" event method
2023/04/05 12:33:51 [notice] 1#1: nginx/1.23.4
2023/04/05 12:33:51 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/04/05 12:33:51 [notice] 1#1: OS: Linux 5.4.0-1104-azure
2023/04/05 12:33:51 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/04/05 12:33:51 [notice] 1#1: start worker processes
2023/04/05 12:33:51 [notice] 1#1: start worker process 29
2023/04/05 12:33:51 [notice] 1#1: start worker process 30
PS /home/andrijana>
```

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

```
PS /home/andrijana> kubectl top pod nginx
NAME      CPU(cores)   MEMORY(bytes)
nginx     0m           3Mi
```

9. Check on which Node your pods have been scheduled. Run `kubectl get pods -o wide`.

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	NOMINATED NODE	READINESS GATES
nginx	1/1	Running	0	4m27s	10.244.0.13	aks-agentpool-35156179-vmss000000	<none>	<none>

- The pod is scheduled on the **aks-agentpool-35156179-vmss000000** node

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

```
PS /home/andrijana> kubectl describe pod nginx
Name:          nginx
Namespace:     default
Priority:       0
Service Account: default
Node:          aks-agentpool-35156179-vmss000000/10.244.0.6
```

- The last one is the name of the Node

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

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

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

```
Container ID:   containerd://4a66b5c789c6bba79c315ee5e18814b66654aec54181c65afb567b/e3a9876
Image:         mcr.microsoft.com/oss/kubernetes/coredns:v1.9.3
Image ID:      sha256:c38f956b642366c8eeb0babfda6b0bb2aa92f27a968589804cadb445f6df72d6
```

13. Once again list all pods under all namespaces.

```
PS /home/andrijana> kubectl get pods --all-namespaces
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	ama-logs-hdf7r	2/2	Running	0	10m
kube-system	ama-logs-rs-7df6dd884-xkx94	1/1	Running	0	10m
kube-system	azure-ip-masq-agent-hkrtz	1/1	Running	0	17m
kube-system	cloud-node-manager-jrhsm	1/1	Running	0	17m
kube-system	coredns-59b6bf8b4f-ntrz8	1/1	Running	0	18m
kube-system	coredns-59b6bf8b4f-zq2tr	1/1	Running	0	16m
kube-system	coredns-autoscaler-5f9cb57949-clzcb	1/1	Running	0	18m
kube-system	csi-azuredisk-node-v2zsv	3/3	Running	0	17m
kube-system	csi-azurefile-node-md5dx	3/3	Running	0	17m
kube-system	konnektivity-agent-c88dd5cff-t6hsb	1/1	Running	0	18m
kube-system	konnektivity-agent-c88dd5cff-x9hj7	1/1	Running	0	18m
kube-system	kube-proxy-hzk5n	1/1	Running	0	17m
kube-system	metrics-server-7dd74d8758-pxdqp	2/2	Running	0	4m40s
kube-system	metrics-server-7dd74d8758-v5xwq	2/2	Running	0	4m41s

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

```
PS /home/andrijana> kubectl describe pod coredns-59b6bf8b4f-ntrz8 -n kube-system
```

Name: coredns-59b6bf8b4f-ntrz8
Namespace: kube-system
Priority: 2000001000
Priority Class Name: system-node-critical
Service Account: coredns
Node: aks-agentpool-35156179-vmss000000/10.244.0.6
Start Time: Wed, 05 Apr 2023 12:24:30 +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: 10.244.0.8
IPs: IP: 10.244.0.8
Controlled By: ReplicaSet/coredns-59b6bf8b4f
Containers:

- Inspecting the pod with name **coredns-59b6bf8b4f-ntrz8**

15. Inspect the output and locate the image information.

```
Container ID:   containerd://74a66b5c789c6bba79c315ee5e18814b66654aec54181c65afb567b7e3a9876
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.

```
PS /home/andrijana> kubectl logs metrics-server-7dd74d8758-pxdqp -n kube-system -c metrics-server
I0405 12:37:02.397696   1 serving.go:342] Generated self-signed cert (/tmp/apiserver.crt, /tmp/apiserver.key)
I0405 12:37:13.306536   1 secure_serving.go:266] Serving securely on [::]:4443
I0405 12:37:13.306586   1 requestheader_controller.go:169] Starting RequestHeaderAuthRequestController
I0405 12:37:13.394685   1 shared_informer.go:240] Waiting for caches to sync for RequestHeaderAuthRequestController
I0405 12:37:13.394681   1 dynamic_serving_content.go:131] "Starting controller" name="serving-cert::/tmp/apiserver.crt::/tmp/apiserver.key"
I0405 12:37:13.395517   1 tlsconfig.go:240] "Starting DynamicServingCertificateController"
W0405 12:37:13.395673   1 shared_informer.go:372] The sharedIndexInformer has started, run more than once is not allowed
I0405 12:37:13.395757   1 configmap_cafile_content.go:201] "Starting controller" name="client-ca::kube-system::extension-apiserver-authentication::requestheader-client-ca-file"
I0405 12:37:13.395792   1 shared_informer.go:240] Waiting for caches to sync for client-ca::kube-system::extension-apiserver-authentication::requestheader-client-ca-file
I0405 12:37:13.395836   1 configmap_cafile_content.go:201] "Starting controller" name="client-ca::kube-system::extension-apiserver-authentication::client-ca-file"
I0405 12:37:13.395856   1 shared_informer.go:240] Waiting for caches to sync for client-ca::kube-system::extension-apiserver-authentication::client-ca-file
I0405 12:37:13.497147   1 shared_informer.go:247] Caches are synced for client-ca::kube-system::extension-apiserver-authentication::client-ca-file
I0405 12:37:13.497193   1 shared_informer.go:247] Caches are synced for client-ca::kube-system::extension-apiserver-authentication::requestheader-client-ca-file
I0405 12:37:13.594753   1 shared_informer.go:247] Caches are synced for RequestHeaderAuthRequestController
```

- `kubectl logs metrics-server-7dd74d8758-pxdqp -n kube-system -c metrics-server`

This command will print the logs of the metrics-server pod under the name 7dd74d8758-pxdqp, created in the kube-system namespace

-n flag -> pod namespace

-c flag -> container name

Practice2: Working with pod manifest files

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
```

- I copied this code in Visual Studio yaml file and saved it locally.

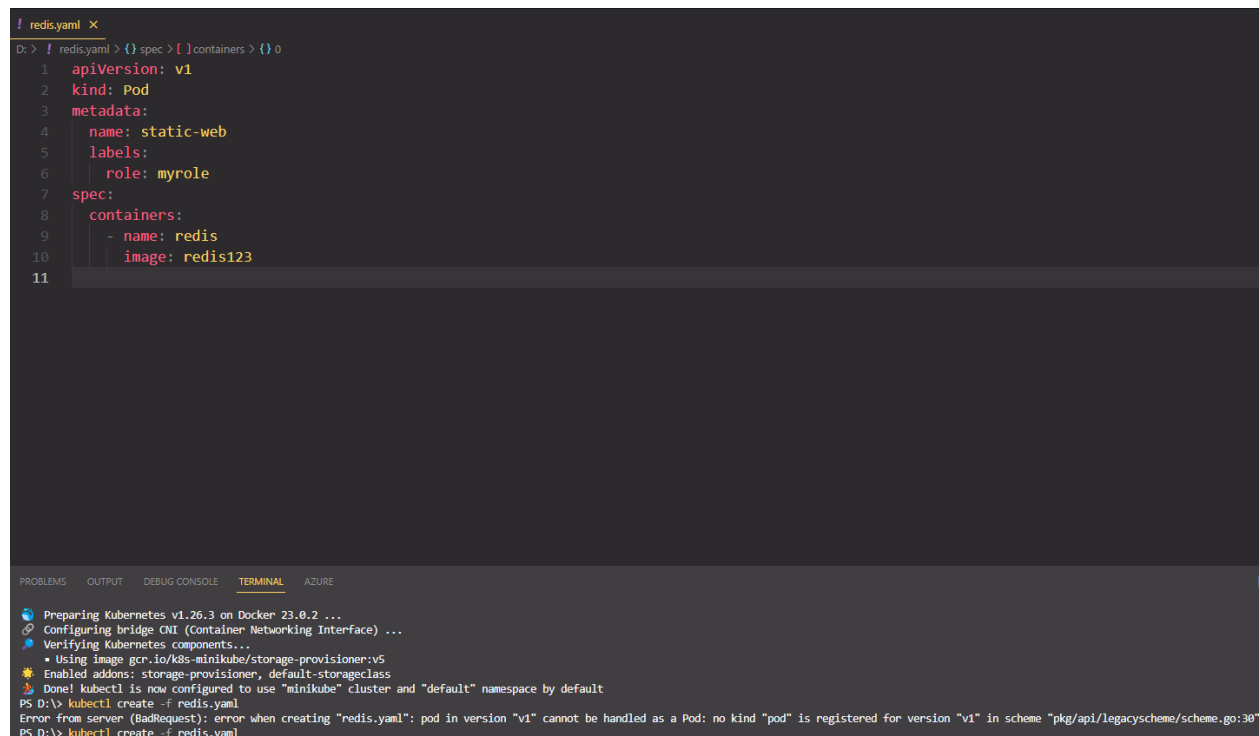
2. Try to deploy the pod defined in redis.yaml. Run `kubectl create -f redis.yaml`.

```
PS D:\> kubectl create -f redis.yaml
Error from server (BadRequest): error when creating "redis.yaml": pod in version "v1" cannot be handled as a Pod: no kind "pod" is registered for version "v1" in scheme "pkg/api/legacyscheme/scheme.go:30"
```

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.

```
PS D:\> kubectl create -f redis.yaml
Error from server (BadRequest): error when creating "redis.yaml": pod in version "v1" cannot be handled as a Pod: no kind "pod" is registered for version "v1" in scheme "pkg/api/legacyscheme/scheme.go:30"
```

- I was getting an error that pod cannot be handled as Pod, and it has to start with the capital letter P.
- apiVersion should be v1 instead of v11



```
! redis.yaml x
D:\> ! redis.yaml > {} spec > {} containers > {} 0
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
11

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL AZURE
Preparing Kubernetes v1.26.3 on Docker 23.0.2 ...
Configuring bridge CNI (Container Networking Interface) ...
Verifying Kubernetes components...
  Using image gcr.io/k8s-minikube/storage-provisioner:v5
  Enabled addons: storage-provisioner, default-storageclass
Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
PS D:\> kubectl create -f redis.yaml
Error from server (BadRequest): error when creating "redis.yaml": pod in version "v1" cannot be handled as a Pod: no kind "pod" is registered for version "v1" in scheme "pkg/api/legacyscheme/scheme.go:30"
PS D:\> kubectl create -f redis.yaml
```

- Correct syntax

```
Error from server (BadRequest): error
PS D:\> kubectl create -f redis.yaml
pod/static-web created
```

- The pod is deployed

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


```
PS D:\> kubectl get pods
NAME          READY   STATUS             RESTARTS   AGE
static-web    0/1     ImagePullBackOff    0           10m
PS D:\>
```

- The status it states ImagePullBackOff
- The status **ImagePullBackOff** means that a Pod couldn't start, because Kubernetes couldn't pull a container image. The 'BackOff' part means that Kubernetes will keep trying to pull the image, with an increasing delay ('back-off').

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?

```
PS D:\> kubectl describe pod static-web
Name:          static-web
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Wed, 05 Apr 2023 16:03:28 +0200
Labels:        role=myrole
Annotations:    <none>
Status:        Pending
IP:            10.244.0.4
IPs:
  IP: 10.244.0.4
Containers:
  ConfigMapOptional: <nil>
  DownwardAPI:      true
QoS Class:          BestEffort
Node-Selectors:     <none>
Tolerations:        node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                    node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age      From          Message
  ----     -
Normal    Scheduled   12m      default-scheduler Successfully assigned default/static-web to minikube
Normal    Pulling     10m (x4 over 12m) kubelet        Pulling image "redis123"
Warning   Failed      10m (x4 over 12m) kubelet        Failed to pull image "redis123": rpc error: code = Unknown desc = Error r
login': denied: requested access to the resource is denied
Warning   Failed      10m (x4 over 12m) kubelet        Error: ErrImagePull
Warning   Failed      10m (x6 over 12m) kubelet        Error: ImagePullBackOff
Normal    BackOff     2m7s (x40 over 12m) kubelet        Back-off pulling image "redis123"
PS D:\>
```

- In the Events section, I get a warning that the in the deployment process of the pod, it failed to pull the image with image name as "**redis123**". The access was denied since either the repository doesn't exist or it may require 'docker login'.
- According to the official documentation, the name of the image should be "**redis**".

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

```
! redis.yaml X
D: > ! redis.yaml > {} spec > [ ] containers > {} 0 > image
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: redis
11
```

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

```
PS D:\> kubectl delete pod static-web
pod "static-web" deleted
PS D:\> kubectl create -f redis.yaml
pod/static-web created
```

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

```
PS D:\> kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
static-web  1/1     Running   0           2m2s
```

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

```
PS D:\> kubectl delete -f redis.yaml
pod "static-web" deleted
```

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.

This is the nginx pod definition file:

```
nginx.yaml ✕
omeworks > ! nginx.yaml > {} spec > [ ] containers > {} 0 > [ ] ports > {} 0
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: frontend
5    labels:
6      role: myrole
7  spec:
8    containers:
9      - name: web
10        image: nginx
11        ports:
12          - name: web
13            containerPort: 80
14            protocol: TCP
15
```

- Here we can see the pod is running

```
PS D:\> kubectl create -f nginx.yaml
pod/frontend created
PS D:\> kubectl get pods
NAME      READY   STATUS             RESTARTS   AGE
frontend  0/1     ContainerCreating  0          33s
PS D:\> kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
frontend  1/1     Running   0          90s
```

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

```
! memcached.yaml ●
D: > ! memcached.yaml > restartPolicy
1  apiVersion: v1
2  kind: Pod
3  restartPolicy: Never
4  metadata:
5    name: web
6    labels:
7      app: web
8  spec:
9    containers:
10     - name: memcached
11       image: memcached
12       ports:
13         - containerPort: 11211
14           protocol: TCP
15       resources:
16         limits:
17           cpu: "0.5"
18           memory: "0.25Gi"
19         requests:
20           cpu: "0.35"
21           memory: "0.15Gi"
22
```

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

```
pod/web created
```

Practice3: Multi-container pods

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.

```
! multi-container.yaml X
D: > ! multi-container.yaml > {} spec > [ ] containers > {} 1 > [ ] ports
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: web2
5    labels:
6      app: web
7  spec:
8    containers:
9      - name: redis
10        image: redis:latest
11        ports:
12          - containerPort: 6379
13      - name: nginx
14        image: nginx
15        ports:
16          - containerPort: 80
17
```

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?

```
PS D:\> kubectl create -f multi-container.yaml
pod/web2 created
```

NAME	READY	STATUS	RESTARTS	AGE
web2	0/1	ContainerCreating	0	18m
web2	2/2	Running	0	33s

- Here we can see our web2 pod is running

```
PS D:\> kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
web2    2/2     Running   0           7s
```

- Our web2 pod has value 2/2 under Ready column. It shows this way because we have deployed 2 containers in the same pod - nginx and redis.

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

```
web2      2/2      Running      0          33s
PS D:\> kubectl describe pod web2
Name:      web2
Namespace: default
Priority:   0
Service Account: default
Node:      minikube/192.168.49.2
Start Time: Wed, 05 Apr 2023 17:46:40 +0200
Labels:    app=web
Annotations: <none>
Status:    Running
IP:        10.244.0.9
IPs:
  IP: 10.244.0.9
Containers:
  redis:
    Container ID:  docker://7c49da9ed615954bfc3fc543803b1e8c481d09a0301842eae671543ea45f528c
    Image:         redis
    Image ID:      docker-pullable://redis@sha256:7b83a0167532d4320a87246a815a134e19e31504d85e8e55f0bb5bb9edf70448
    Port:         6379/TCP
    Host Port:    0/TCP
    State:        Running
      Started:    Wed, 05 Apr 2023 17:46:45 +0200
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                  node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age   From          Message
  ----     -
Normal    Scheduled   2m32s default-scheduler Successfully assigned default/web2 to minikube
Normal    Pulling     2m31s kubelet       Pulling image "redis"
Normal    Pulled      2m29s kubelet       Successfully pulled image "redis" in 1.856134923s (1.856147345s including waiting)
Normal    Created     2m29s kubelet       Created container redis
Normal    Started     2m28s kubelet       Started container redis
Normal    Pulling     2m28s kubelet       Pulling image "nginx"
Normal    Pulled      2m26s kubelet       Successfully pulled image "nginx" in 1.965341326s (1.96536559s including waiting)
Normal    Created     2m26s kubelet       Created container nginx
Normal    Started     2m26s kubelet       Started container nginx
```

There are 2 containers:

```
Containers:
  redis:
    Container ID:  docker://7c49da9ed615954bfc3fc543803b1e8c481d09a0301842eae671543ea45f528c
    Image:         redis
    Image ID:      docker-pullable://redis@sha256:7b83a0167532d4320a87246a815a134e19e31504d85e8e55f0bb5bb9edf70448
    Port:         6379/TCP
    Host Port:    0/TCP
    State:        Running
      Started:    Wed, 05 Apr 2023 17:46:45 +0200
    Ready:        True
    Restart Count: 0
    Environment:  <none>
    Mounts:
```

5. Delete all the pods under the default namespace.

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

Practice4: Probes

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

```
probes_exec.yaml > \ spec > \ containers > \ 0 > \ livenessProbe > \ exec
apiVersion: v1
kind: Pod
metadata:
  name: liveness-exec
  labels:
    app: web
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.

3. Run `kubectl create -f probes_exec.yaml`.

```
PS D:\> 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.

```
PS D:\> kubectl describe pod liveness-exec
Name:          liveness-exec
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Wed, 05 Apr 2023 19:43:54 +0200
Labels:        app=web
Annotations:   <none>
Status:        Running
IP:            10.244.0.16
IPs:           IP: 10.244.0.16
Containers:
  liveness:
    Container ID:  docker://7c592f26c658fca0ddad9018375a32769721dbbe6b28fd2f966941a05441452e
    Image:         k8s.gcr.io/busybox
    kube-api-access-7cwp7:
      Type:          Projected (a volume that contains injected data from multiple sources)
      TokenExpirationSeconds: 3607
      ConfigMapName:  kube-root-ca.crt
      ConfigMapOptional: <nil>
      DownwardAPI:    true
    QoS Class:       BestEffort
    Node-Selectors:  <none>
    Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                     node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age      From          Message
  ----     -
  Normal   Scheduled   2m35s    default-scheduler   Successfully assigned default/liveness-exec to minikube
  Normal   Pulled      2m32s    kubelet        Successfully pulled image "k8s.gcr.io/busybox" in 2.446740694s (2.446761285s including waiting)
  Normal   Pulled      78s      kubelet        Successfully pulled image "k8s.gcr.io/busybox" in 1.397228603s (1.397249921s including waiting)
  Warning  Unhealthy   35s (x6 over 2m) kubelet        Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory
  Normal   Killing     35s (x2 over 110s) kubelet        Container liveness failed liveness probe, will be restarted
  Normal   Pulling     5s (x3 over 2m34s) kubelet        Pulling image "k8s.gcr.io/busybox"
  Normal   Created     3s (x3 over 2m32s) kubelet        Created container liveness
  Normal   Started     3s (x3 over 2m31s) kubelet        Started container liveness
  Normal   Pulled      3s      kubelet        Successfully pulled image "k8s.gcr.io/busybox" in 1.390842806s (1.390862173s including waiting)
PS D:\>
```

- We can see in the Events section that the liveness container was started successfully:

```
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason      Age      From          Message
  ----     -
  Normal   Scheduled   2m35s    default-scheduler   Successfully assigned default/liveness-exec to minikube
  Normal   Pulled      2m32s    kubelet        Successfully pulled image "k8s.gcr.io/busybox" in 2.446740694s (2.446761285s including waiting)
  Normal   Pulled      78s      kubelet        Successfully pulled image "k8s.gcr.io/busybox" in 1.397228603s (1.397249921s including waiting)
  Warning  Unhealthy   35s (x6 over 2m) kubelet        Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory
  Normal   Killing     35s (x2 over 110s) kubelet        Container liveness failed liveness probe, will be restarted
  Normal   Pulling     5s (x3 over 2m34s) kubelet        Pulling image "k8s.gcr.io/busybox"
  Normal   Created     3s (x3 over 2m32s) kubelet        Created container liveness
  Normal   Started     3s (x3 over 2m31s) kubelet        Started container liveness
  Normal   Pulled      3s      kubelet        Successfully pulled image "k8s.gcr.io/busybox" in 1.390842806s (1.390862173s including waiting)
PS D:\>
```


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

```
PS D:\> kubectl describe pod liveness-exec
Name:          liveness-exec
Namespace:     default
Priority:       0
Service Account: default
Node:          minikube/192.168.49.2
Start Time:    Wed, 05 Apr 2023 19:43:54 +0200
Labels:        app=web
Annotations:   <none>
Status:        Running
IP:            10.244.0.16
IPs:
  IP: 10.244.0.16
Containers:
  liveness:
    Container ID:  docker://869f3c3d3bbcd3356278fae19dec400dbf92b15ce987c8c8c2b5ca3a275a1d6
    Image:          k8s.gcr.io/busybox
    Image ID:       docker-pullable://k8s.gcr.io/busybox@sha256:d8d3bc2c183ed2f9f10e7258f84971202325ee6011ba137112e01e30f206de67
    Port:           <none>
    Host Port:      <none>
    Args:
      /bin/sh
      -c
      touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    State:          Running
      Started:       Wed, 05 Apr 2023 19:47:41 +0200
    Last State:     Terminated
      Reason:        Error
      Exit Code:     137
      Started:       Wed, 05 Apr 2023 19:46:26 +0200
      Finished:      Wed, 05 Apr 2023 19:47:39 +0200
    Ready:          True
    Restart Count:   3
```

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.

```
Warning   Unhealthy   34s (x9 over 3m14s)   kubelet      Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory
Normal    Killing     34s (x3 over 3m4s)    kubelet      Container liveness failed liveness probe, will be restarted
Normal    Pulling     4s (x4 over 3m48s)    kubelet      Pulling image "k8s.gcr.io/busybox"
```

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

```
PS D:\> kubectl get pod liveness-exec
NAME          READY   STATUS    RESTARTS   AGE
liveness-exec 1/1     Running   3 (74s ago) 4m59s
PS D:\>
```

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

```
RESTARTS
4 (31s ago)
```

- Now the number of restarts has increased to 4, compared to the previous case when it was 3.

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

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    labels:
5      test: liveness
6      name: liveness-http
7  spec:
8    containers:
9      - name: liveness
10        image: k8s.gcr.io/liveness
11        args:
12          - /server
13        livenessProbe:
14          httpGet:
15            path: /healthz
16            port: 8080
17          httpHeaders:
18            - name: Custom-Header
19              value: Awesome
20          initialDelaySeconds: 3
21          periodSeconds: 3
```

10. Just for your info, /healthz 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.

```
Normal      Pulled      12s      kubelet      Successfully pulled image "k8s.gcr.io/liveness" in 1.231878415s (1.231884872s including waiting)
Normal      Started     11s (x3 over 47s) kubelet      Started container liveness
Warning     Unhealthy   1s (x7 over 37s) kubelet      Liveness probe failed: HTTP probe failed with statuscode: 500
```

12. Run `kubectl create -f probes_http.yaml`.

```
PS D:\> kubectl create -f probes_http.yaml
pod/liveness-http created
```

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

```
Normal      Pulling      14s      kubelet      Pulling image "k8s.gcr.io/liveness"
Normal      Pulled       13s      kubelet      Successfully pulled image "k8s.gcr.io/liveness" in 1.466386708s (1.466410333s including waiting)
Normal      Created      13s      kubelet      Created container liveness
Normal      Started      13s      kubelet      Started container liveness
```

14. Your pod should be live and running.

```
PS D:\> kubectl get pods liveness-http
NAME          READY   STATUS    RESTARTS   AGE
liveness-http 1/1     Running   1 (17s ago) 38s
```

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`.

```
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
Type      Reason      Age          From          Message
----      -
Normal    Scheduled   2m7s        default-scheduler      Successfully assigned default/liveness-http to minikube
Normal    Pulled      2m3s        kubelet        Successfully pulled image "k8s.gcr.io/liveness" in 2.721567373s (2.721586729s including waiting)
Normal    Pulled      105s        kubelet        Successfully pulled image "k8s.gcr.io/liveness" in 1.266623396s (1.26665157s including waiting)
Normal    Created     87s (x3 over 2m2s) kubelet        Created container liveness
Normal    Pulled      87s        kubelet        Successfully pulled image "k8s.gcr.io/liveness" in 1.231878415s (1.231884872s including waiting)
Normal    Started     86s (x3 over 2m2s) kubelet        Started container liveness
Normal    Pulling     70s (x4 over 2m5s) kubelet        Pulling image "k8s.gcr.io/liveness"
Warning   Unhealthy   70s (x9 over 112s) kubelet        Liveness probe failed: HTTP probe failed with statuscode: 500
Normal    Killing     70s (x3 over 106s) kubelet        Container liveness failed liveness probe, will be restarted
```

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

```
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
Type      Reason      Age          From          Message
----      -
Normal    Scheduled   95s         default-scheduler      Successfully assigned default/liveness-http to minikube
Normal    Pulled      93s         kubelet        Successfully pulled image "k8s.gcr.io/liveness" in 1.466386708s (1.466410333s including waiting)
Normal    Pulled      73s         kubelet        Successfully pulled image "k8s.gcr.io/liveness" in 1.246246882s (1.24628027s including waiting)
Normal    Created     55s (x3 over 93s) kubelet        Created container liveness
Normal    Pulled      55s         kubelet        Successfully pulled image "k8s.gcr.io/liveness" in 1.251337911s (1.251358378s including waiting)
Normal    Started     54s (x3 over 93s) kubelet        Started container liveness
Normal    Pulling     38s (x4 over 94s) kubelet        Pulling image "k8s.gcr.io/liveness"
Warning   Unhealthy   38s (x9 over 80s) kubelet        Liveness probe failed: HTTP probe failed with statuscode: 500
Normal    Killing     38s (x3 over 74s) kubelet        Container liveness failed liveness probe, will be restarted
```

- It's giving status code 500 again

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

```
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: 3
        periodSeconds: 3
```

18. Run `kubectl create -f probes_tcp.yaml`.

```
PS D:\> kubectl create -f probes_tcp.yaml
pod/liveness-tcp created
```

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

Normal	Pulling	8s	kubelet	Pulling image "k8s.gcr.io/goproxy:0.1"
Normal	Pulled	5s	kubelet	Successfully pulled image "k8s.gcr.io/goproxy:0.1" in 3.29358175s (3.293632135s including waiting)
Normal	Created	5s	kubelet	Created container goproxy
Normal	Started	4s	kubelet	Started container goproxy

20. Your pod should be live and running.

```
PS D:\> kubectl get pods liveness-tcp
NAME          READY   STATUS    RESTARTS   AGE
liveness-tcp  1/1     Running   1 (7s ago)  22s
```

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`.

Normal	Started	19s (x3 over 38s)	kubelet	Started container goproxy
Normal	Pulled	19s (x2 over 28s)	kubelet	Container image "k8s.gcr.io/goproxy:0.1" already present on machine
Warning	Unhealthy	18s (x9 over 34s)	kubelet	Liveness probe failed: dial tcp 10.244.0.19:9999: connect: connection refused
Normal	Killing	18s (x3 over 28s)	kubelet	Container goproxy failed liveness probe, will be restarted
Warning	BackOff	18s (x2 over 10s)	kubelet	Back-off restarting failed container goproxy in pod liveness-tcp_default(6f4f039b-104c-41b6-b47f-e1641f3badd0)

```
PS D:\>
```

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

```
Warning Unhealthy 69s (x9 over 93s) kubelet Liveness probe failed: dial tcp 10.244.0.19:9999: connect: connection refused
Normal Killing 69s (x3 over 87s) kubelet Container goproxy failed liveness probe, will be restarted
Warning BackOff 69s (x2 over 69s) kubelet Back-off restarting failed container goproxy in pod liveness-tcp_default(6f4f039b-104c-41b6-b47f-e1641f3badd0)
Normal Pulled 54s (x3 over 87s) kubelet Container image "k8s.gcr.io/goproxy:0.1" already present on machine
```

- We get the same failure as in steps 7 and 16.

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`.

```
PS D:\> kubectl get pods readiness-http
NAME          READY   STATUS    RESTARTS   AGE
readiness-http 1/1     Running   0           79s
```

- I've created the pod and now I'm checking its status

26. Run `kubectl get pods -A` to see the status of your pod.

```
PS D:\> kubectl get pods -A
NAMESPACE     NAME                                READY   STATUS    RESTARTS   AGE
default       liveness-exec                      1/1     Running   16 (5m38s ago)  47m
default       liveness-http                      0/1     CrashLoopBackOff 15 (2m38s ago)  33m
default       liveness-tcp                       0/1     CrashLoopBackOff 13 (4m4s ago)   27m
default       my-release-nginx-6c6dd45bcd-bgxwq 1/1     Running   0           128m
default       my-release2-kafka-0               1/1     Running   5 (96m ago)    103m
default       my-release2-zookeeper-0           1/1     Running   0           103m
default       readiness-http                     1/1     Running   0           112s
kube-system   coredns-787d4945fb-xph2k          1/1     Running   1 (4h43m ago)  4h46m
kube-system   etcd-minikube                     1/1     Running   1 (4h43m ago)  4h46m
kube-system   kube-apiserver-minikube            1/1     Running   1 (4h43m ago)  4h46m
kube-system   kube-controller-manager-minikube   1/1     Running   1 (4h43m ago)  4h46m
kube-system   kube-proxy-qx2ww                   1/1     Running   1 (4h43m ago)  4h46m
kube-system   kube-scheduler-minikube            1/1     Running   1 (4h43m ago)  4h46m
kube-system   storage-provisioner                 1/1     Running   3 (4h41m ago)  4h46m
PS D:\>
```

27. Pods and their status and ready states will be displayed; our pod should be in running state.

- The pod is in running state

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   2m33s default-scheduler Successfully assigned default/readiness-http to minikube
  Normal   Pulling     2m31s kubelet        Pulling image "nginx"
  Normal   Pulled      2m30s kubelet        Successfully pulled image "nginx" in 1.649537021s (1.649543255s including waiting)
  Normal   Created     2m30s kubelet        Created container nginx
  Normal   Started     2m30s kubelet        Started container nginx
```

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

```
PS D:\> kubectl delete pod readiness-http
pod "readiness-http" deleted
PS D:\>
```

```

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 D:\> kubectl get pods -A

```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
default	liveness-exec	0/1	CrashLoopBackOff	17 (3m40s ago)	53m
default	liveness-http	0/1	CrashLoopBackOff	17 (2m27s ago)	38m
default	liveness-tcp	0/1	CrashLoopBackOff	15 (4m14s ago)	32m
default	my-release-nginx-6c6dd45bcd-bgxwq	1/1	Running	0	134m
default	my-release2-kafka-0	1/1	Running	5 (101m ago)	109m
default	my-release2-zookeeper-0	1/1	Running	0	109m
default	readiness-http	0/1	Running	0	51s
kube-system	coredns-787d4945fb-xph2k	1/1	Running	1 (4h49m ago)	4h52m
kube-system	etcd-minikube	1/1	Running	1 (4h49m ago)	4h52m
kube-system	kube-apiserver-minikube	1/1	Running	1 (4h49m ago)	4h52m
kube-system	kube-controller-manager-minikube	1/1	Running	1 (4h49m ago)	4h52m
kube-system	kube-proxy-qx2ww	1/1	Running	1 (4h49m ago)	4h52m
kube-system	kube-scheduler-minikube	1/1	Running	1 (4h49m ago)	4h52m
kube-system	storage-provisioner	1/1	Running	3 (4h47m ago)	4h52m

NAME	READY	STATUS	RESTARTS	AGE
readiness-http	0/1	Running	0	6m55s

- 0/1 showing the pod is not in ready state

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	7m38s	default-scheduler	Successfully assigned default/readiness-http to minikube
Normal	Pulling	7m37s	kubelet	Pulling image "nginx"
Normal	Pulled	7m35s	kubelet	Successfully pulled image "nginx" in 1.621034761s (1.621053745s including waiting)
Normal	Created	7m35s	kubelet	Created container nginx
Normal	Started	7m35s	kubelet	Started container nginx
Warning	Unhealthy	2m35s (x154 over 7m34s)	kubelet	Readiness probe failed: Get "http://10.244.0.21:81/": dial tcp 10.244.0.21:81: connect: connection refused

34. Delete all pods under the default namespace.


```
See "kubectl delete --help" for usage.  
PS D:\> kubectl delete pods --all -n default  
pod "liveness-exec" deleted  
pod "liveness-http" deleted  
pod "liveness-tcp" deleted  
pod "my-release-nginx-6c6dd45bcd-bgxwq" deleted  
pod "my-release2-kafka-0" deleted  
pod "my-release2-zookeeper-0" deleted  
pod "readiness-http" deleted  
PS D:\> 
```

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