Kubernetes lab

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

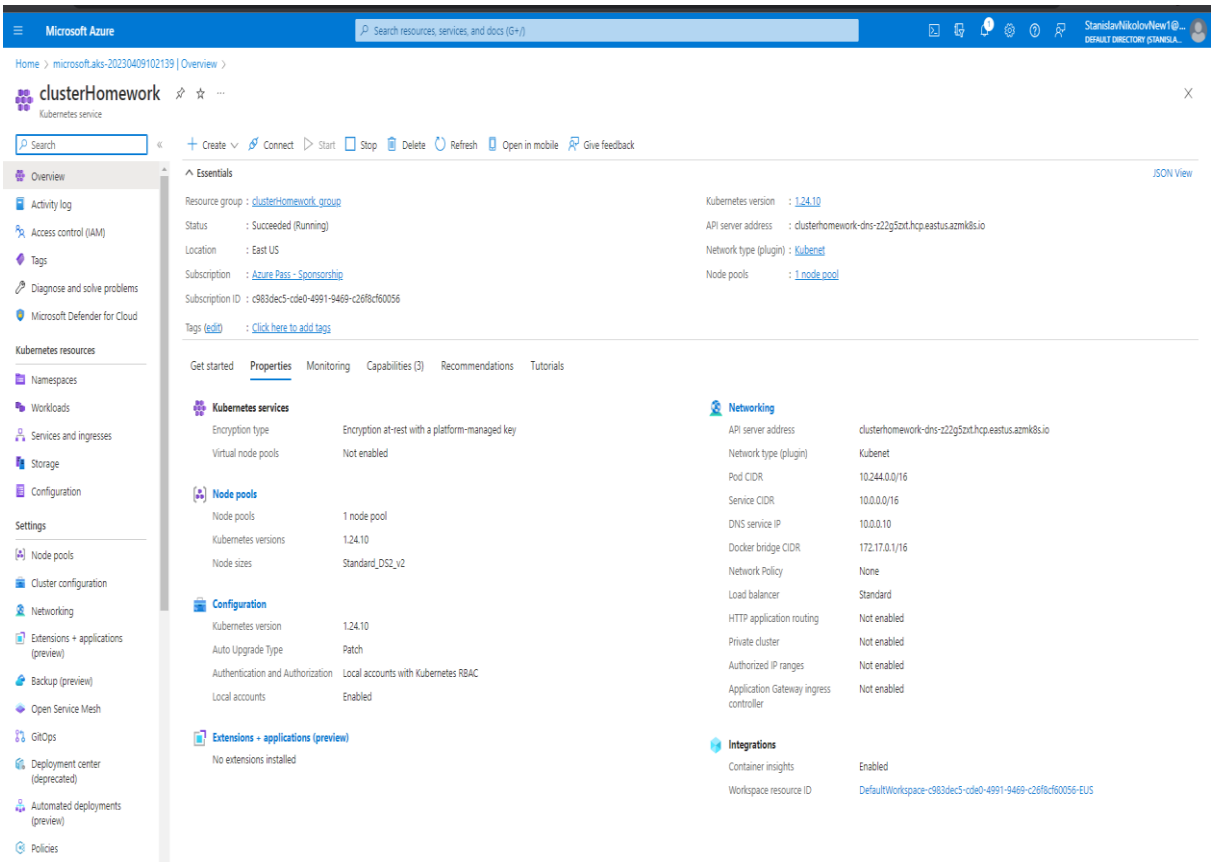
manifest files will be used for defining more complex Kubernetes resources like Controllers.

Practice1: Simple pods operations

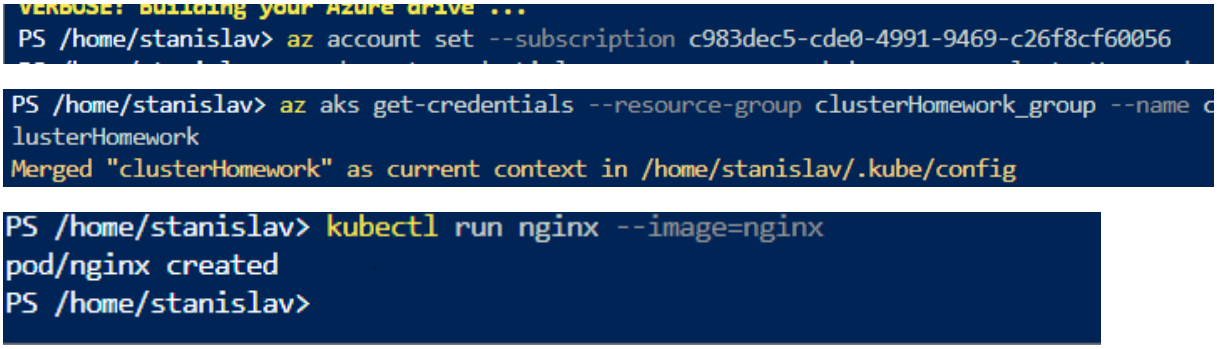
Note: Try not to do a copy/paste on commands requests unless you are instructed to do so.

Copy/paste will not help you to learn Kubernetes!

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



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



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

```
PS /home/stanislav> kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   ama-logs-7bnjj                        2/2     Running   0          9m47s
kube-system   ama-logs-c75mm                        2/2     Running   0          9m51s
kube-system   ama-logs-r77cl                        2/2     Running   0          9m52s
kube-system   ama-logs-rs-79b79975d5-5dgmr          1/1     Running   0          9m58s
kube-system   azure-ip-masq-agent-6nbj8             1/1     Running   0          9m51s
kube-system   azure-ip-masq-agent-m7pjm             1/1     Running   0          9m52s
kube-system   azure-ip-masq-agent-nlzh4             1/1     Running   0          9m47s
kube-system   cloud-node-manager-5qbwg              1/1     Running   0          9m51s
kube-system   cloud-node-manager-rtrnm              1/1     Running   0          9m47s
kube-system   cloud-node-manager-zbh2z              1/1     Running   0          9m52s
kube-system   coredns-59b6bf8b4f-dcxbv              1/1     Running   0          9m58s
kube-system   coredns-59b6bf8b4f-vrnp2              1/1     Running   0          8m37s
kube-system   coredns-autoscaler-64b6477b8b-njsnl   1/1     Running   0          9m58s
kube-system   csi-azuredisk-node-dnf6r              3/3     Running   0          9m52s
kube-system   csi-azuredisk-node-qz4ks              3/3     Running   0          9m47s
kube-system   csi-azuredisk-node-xl5rx              3/3     Running   0          9m51s
kube-system   csi-azurefile-node-5dmrq              3/3     Running   0          9m51s
kube-system   csi-azurefile-node-br9w4              3/3     Running   0          9m52s
kube-system   csi-azurefile-node-nwgdz              3/3     Running   0          9m47s
kube-system   konnectivity-agent-7cf8bd6556-b7828   1/1     Running   0          9m58s
kube-system   konnectivity-agent-7cf8bd6556-t6qqh   1/1     Running   0          9m58s
kube-system   kube-proxy-n8q64                      1/1     Running   0          9m47s
kube-system   kube-proxy-tg9j6                      1/1     Running   0          9m51s
kube-system   kube-proxy-v52qk                      1/1     Running   0          9m52s
kube-system   metrics-server-5f8d84558d-d7cbb       2/2     Running   0          8m30s
kube-system   metrics-server-5f8d84558d-xnpnd       2/2     Running   0          8m30s
PS /home/stanislav>
```

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

- There are kube-system pods mostly. Around 28

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

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

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

```
PS /home/stanislav> kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          83s
PS /home/stanislav>
```

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

```
PS /home/stanislav> 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
2023/04/09 07:38:08 [notice] 1#1: using the "epoll" event method
2023/04/09 07:38:08 [notice] 1#1: nginx/1.23.4
2023/04/09 07:38:08 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/04/09 07:38:08 [notice] 1#1: OS: Linux 5.4.0-1104-azure
2023/04/09 07:38:08 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/04/09 07:38:08 [notice] 1#1: start worker processes
2023/04/09 07:38:08 [notice] 1#1: start worker process 29
2023/04/09 07:38:08 [notice] 1#1: start worker process 30
PS /home/stanislav>
```

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

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

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

```
PS /home/stanislav> kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE    IP           NODE                                NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          5m9s   10.244.0.8   aks-agentpool-35627940-vmss000000   <none>           <none>
PS /home/stanislav>
```

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

```
Service Account:  default
Node:             aks-agentpool-35627940-vmss000000/10.224.0.6
Start Time:       Sun, 09 Apr 2023 07:38:04 +0000
Labels:           run=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.8
IPs:
  IP:  10.244.0.8
Containers:
  nginx:
    Container ID:   containerd://12dea4222105c5e3005ba5d167c0f58555ef343d604ae7ec7e10561226741831
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:2ab30d6ac53580a6db8b657abf0f68d75360ff5cc1670a85acb5bd85ba1b19c0
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sun, 09 Apr 2023 07:38:08 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-2g7hz (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-2g7hz:
    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  6m36s  default-scheduler  Successfully assigned default/nginx to aks-agentpool-35627940-vmss000000
  Normal  Pulling    6m36s  kubelet            Pulling image "nginx"
  Normal  Pulled     6m33s  kubelet            Successfully pulled image "nginx" in 2.65517193s
  Normal  Created    6m33s  kubelet            Created container nginx
  Normal  Started    6m33s  kubelet            Started container nginx
```

11. Delete your pod using kubectl delete pod nginx.

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

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/stanislav> kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   ama-logs-7bnjj                        2/2     Running   0          20m
kube-system   ama-logs-r77cl                        2/2     Running   0          20m
kube-system   ama-logs-rs-79b79975d5-5dgmr          1/1     Running   0          21m
kube-system   azure-ip-masq-agent-m7pjm             1/1     Running   0          20m
kube-system   azure-ip-masq-agent-nlzh4             1/1     Running   0          20m
kube-system   cloud-node-manager-rtrnm              1/1     Running   0          20m
kube-system   cloud-node-manager-zbh2z              1/1     Running   0          20m
kube-system   coredns-59b6bf8b4f-dcxbv              1/1     Running   0          21m
kube-system   coredns-59b6bf8b4f-vrnp2              1/1     Running   0          19m
kube-system   coredns-autoscaler-64b6477b8b-njsnl   1/1     Running   0          21m
kube-system   csi-azuredisk-node-dnf6r              3/3     Running   0          20m
kube-system   csi-azuredisk-node-qz4ks              3/3     Running   0          20m
kube-system   csi-azurefile-node-br9w4              3/3     Running   0          20m
kube-system   csi-azurefile-node-nwgdz              3/3     Running   0          20m
kube-system   konnectivity-agent-7cf8bd6556-b7828   1/1     Running   0          21m
kube-system   konnectivity-agent-7cf8bd6556-t6qqh   1/1     Running   0          21m
kube-system   kube-proxy-n8q64                      1/1     Running   0          20m
kube-system   kube-proxy-v52qk                      1/1     Running   0          20m
kube-system   metrics-server-8655f897d8-glgrl       2/2     Running   0          7m28s
kube-system   metrics-server-8655f897d8-swgjh       2/2     Running   0          7m28s
PS /home/stanislav>
```

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/stanislav> kubectl describe pod kube-proxy-v52qk -n kube-system
Name:                 kube-proxy-v52qk
Namespace:            kube-system
Priority:             2000001000
Priority Class Name:  system-node-critical
Service Account:      kube-proxy
Node:                 aks-agentpool-35627940-vmss000000/10.224.0.6
Start Time:           Sun, 09 Apr 2023 07:26:25 +0000
Labels:               component=kube-proxy
                      controller-revision-hash=6985689589
                      pod-template-generation=1
                      tier=node
Annotations:          <none>
Status:               Running
IP:                   10.224.0.6
IPs:
  IP:           10.224.0.6
Controlled By:  DaemonSet/kube-proxy
Init Containers:
  kube-proxy-bootstrap:
    Container ID:  containerd://4c89a79a19cb1222ff8d1291c66df263439fb00a10f35b25c796792c2a5e2635
    Image:         mcr.microsoft.com/oss/kubernetes/kube-proxy:v1.24.10.3
    Image ID:      sha256:a03c20def8004c7cca3003be78610ab1b6155fd26b91163459b0ef49d08e3b9a
    Port:          <none>
    Host Port:     <none>
    Command:
      /bin/sh
      -c
      SYSCTL=/proc/sys/net/netfilter/nf_conntrack_max
      echo "Current net.netfilter.nf_conntrack_max: $(cat $SYSCTL)"
      DESIRED=$(awk -F= '/net.netfilter.nf_conntrack_max/ {print $2}' /etc/sysctl.d/999-sysctl-aks.conf)
      if [ -z "$DESIRED" ]; then
        DESIRED=$((32768*$(nproc)))
        if [ $DESIRED -lt 131072 ]; then
          DESIRED=131072
        fi

        echo "AKS custom config for net.netfilter.nf_conntrack_max not set."
        echo "Setting nf_conntrack_max to $DESIRED (32768 * $(nproc) cores, minimum 131072)."
        echo $DESIRED > $SYSCTL
      else
        echo "AKS custom config for net.netfilter.nf_conntrack_max set to $DESIRED."
        echo "Setting nf_conntrack_max to $DESIRED."
        echo $DESIRED > $SYSCTL
      fi

    State:          Terminated
```

15. Inspect the output and locate the image information.

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/stanislav> kubectl logs kube-proxy-v52qk -n kube-system -c kube-proxy
I0409 07:26:39.772028       1 flags.go:64] FLAG: --add-dir-header="false"
I0409 07:26:39.772082       1 flags.go:64] FLAG: --alsologtostderr="false"
I0409 07:26:39.772085       1 flags.go:64] FLAG: --bind-address="0.0.0.0"
I0409 07:26:39.772090       1 flags.go:64] FLAG: --bind-address-hard-fail="false"
I0409 07:26:39.772100       1 flags.go:64] FLAG: --boot-id-file="/proc/sys/kernel/random/boot_id"
I0409 07:26:39.772104       1 flags.go:64] FLAG: --cleanup="false"
I0409 07:26:39.772106       1 flags.go:64] FLAG: --cluster-cidr="10.244.0.0/16"
I0409 07:26:39.772110       1 flags.go:64] FLAG: --config=""
I0409 07:26:39.772113       1 flags.go:64] FLAG: --config-sync-period="15m0s"
I0409 07:26:39.772117       1 flags.go:64] FLAG: --conntrack-max-per-core="0"
I0409 07:26:39.772121       1 flags.go:64] FLAG: --conntrack-min="131072"
I0409 07:26:39.772124       1 flags.go:64] FLAG: --conntrack-tcp-timeout-close-wait="1h0m0s"
I0409 07:26:39.772127       1 flags.go:64] FLAG: --conntrack-tcp-timeout-established="24h0m0s"
I0409 07:26:39.772130       1 flags.go:64] FLAG: --detect-local-mode="ClusterCIDR"
I0409 07:26:39.772134       1 flags.go:64] FLAG: --feature-gates=""
I0409 07:26:39.772138       1 flags.go:64] FLAG: --healthz-bind-address="0.0.0.0:10256"
I0409 07:26:39.772141       1 flags.go:64] FLAG: --healthz-port="10256"
I0409 07:26:39.772144       1 flags.go:64] FLAG: --help="false"
I0409 07:26:39.772147       1 flags.go:64] FLAG: --hostname-override=""
I0409 07:26:39.772150       1 flags.go:64] FLAG: --iptables-masquerade-bit="14"
I0409 07:26:39.772153       1 flags.go:64] FLAG: --iptables-min-sync-period="1s"
I0409 07:26:39.772155       1 flags.go:64] FLAG: --iptables-sync-period="30s"
I0409 07:26:39.772158       1 flags.go:64] FLAG: --ipvs-exclude-cidrs="[]"
I0409 07:26:39.772165       1 flags.go:64] FLAG: --ipvs-min-sync-period="0s"
I0409 07:26:39.772169       1 flags.go:64] FLAG: --ipvs-scheduler=""
I0409 07:26:39.772172       1 flags.go:64] FLAG: --ipvs-strict-arp="false"
I0409 07:26:39.772175       1 flags.go:64] FLAG: --ipvs-sync-period="30s"
I0409 07:26:39.772177       1 flags.go:64] FLAG: --ipvs-tcp-timeout="0s"
I0409 07:26:39.772180       1 flags.go:64] FLAG: --ipvs-tcpfin-timeout="0s"
I0409 07:26:39.772183       1 flags.go:64] FLAG: --ipvs-udp-timeout="0s"
I0409 07:26:39.772185       1 flags.go:64] FLAG: --kube-api-burst="10"
I0409 07:26:39.772188       1 flags.go:64] FLAG: --kube-api-content-type="application/vnd.kubernetes.protobuf"
I0409 07:26:39.772192       1 flags.go:64] FLAG: --kube-api-qps="5"
I0409 07:26:39.772196       1 flags.go:64] FLAG: --kubeconfig="/var/lib/kubelet/kubeconfig"
I0409 07:26:39.772199       1 flags.go:64] FLAG: --log-backtrace-at=":0"
I0409 07:26:39.772204       1 flags.go:64] FLAG: --log-dir=""
I0409 07:26:39.772208       1 flags.go:64] FLAG: --log-file=""
I0409 07:26:39.772211       1 flags.go:64] FLAG: --log-file-max-size="1800"
I0409 07:26:39.772213       1 flags.go:64] FLAG: --log-flush-frequency="5s"
I0409 07:26:39.772216       1 flags.go:64] FLAG: --logtostderr="true"
I0409 07:26:39.772219       1 flags.go:64] FLAG: --machine-id-file="/etc/machine-id,/var/lib/dbus/machine-id"
I0409 07:26:39.772222       1 flags.go:64] FLAG: --masquerade-all="false"
I0409 07:26:39.772225       1 flags.go:64] FLAG: --master=""
I0409 07:26:39.772228       1 flags.go:64] FLAG: --metrics-bind-address="0.0.0.0:10249"
I0409 07:26:39.772230       1 flags.go:64] FLAG: --metrics-port="10249"
I0409 07:26:39.772233       1 flags.go:64] FLAG: --nodeport-addresses="[]"
```

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

- This is the redis.yaml, which I will upload in GitHub

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

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 C:\Users\stani> kubectl create -f redis.yaml
Unable to connect to the server: dial tcp 127.0.0.1:56306: connectex: No connection could be made because the target mac
hine actively refused it.
PS C:\Users\stani>
```

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

```
pod/static-web created
```

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?

```
NAME         READY   STATUS             RESTARTS   AGE
static-web   0/1     ImagePullBackOff   0          27s
```

```
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.

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).

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

```
NAME         READY   STATUS    RESTARTS   AGE
static-web   1/1     Running   0          7s
```

9. Now you can delete the pod. Try to delete it using the 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.

nginx.yaml - the yaml file.

```
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:

- The yaml file for nginx pod.
- 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

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


memcahed-web.yaml - the yaml file for the pod.

```
pod/memcached-web created
```

```
NAME            READY   STATUS    RESTARTS   AGE
memcached-web   1/1     Running   0          15s
nginx-pod       1/1     Running   0          7m7s
```

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.


webapp.yaml - the yaml file for the 2 containers in 1 pod.

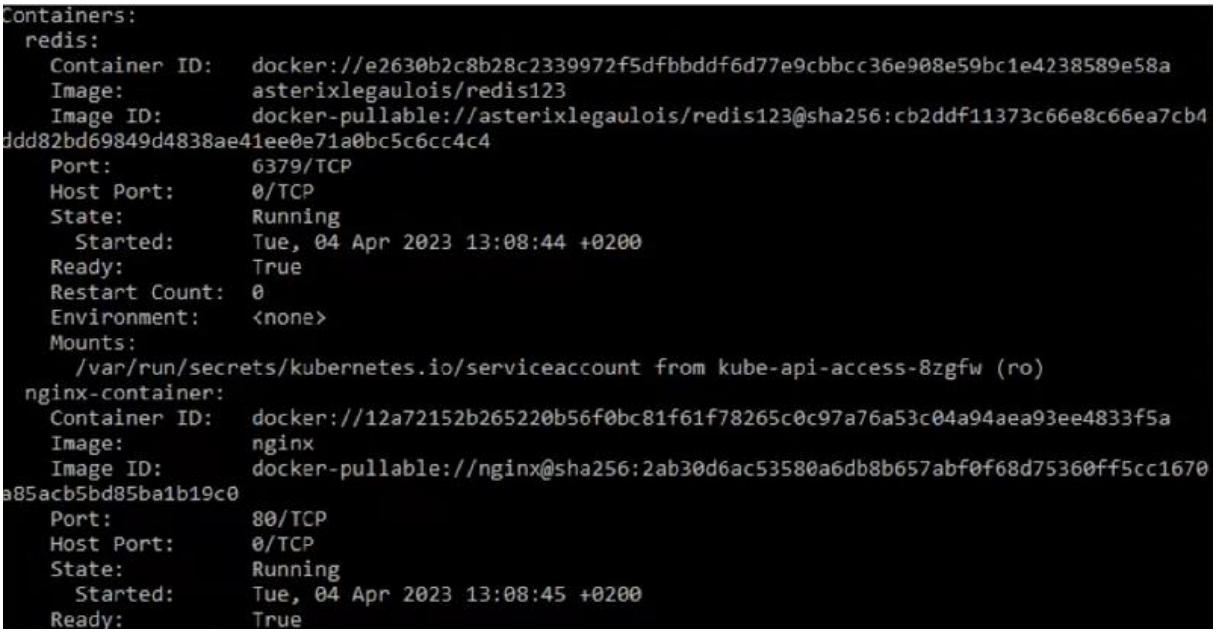
2. Note that in reality there is no sense to put the redis and nginx under the same pod but it can be done for the purpose of learning.

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?

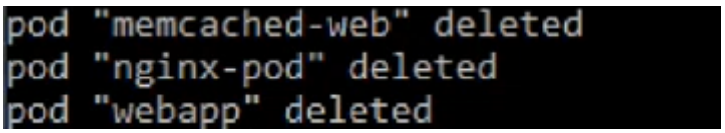
```
webapp          2/2     Running   0          66s
```

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

- There are 2 containers, running in one pod.



5. Delete all the pods under the default namespace.



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

- They all will be in my Github.

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:


liveness-exec.yaml - The yaml file for the container

```
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.

3. Run 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.

```
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>
```

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

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

probes_http.yaml - The yaml file fot this containers.

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.

```
pod/liveness-http created
```

13. Immediately run (you only have 10 secs to run this command) 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
```

14. Your pod should be live and running.

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:

probes_tcp.yaml -The yaml file for this containers

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.

```
pod/liveness-tcp created
```

19. Immediately run (you only have 10 secs to run this command) 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
```

20. Your pod should be live and running.

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
```

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:

readiness.yaml -The last yaml file for this container

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

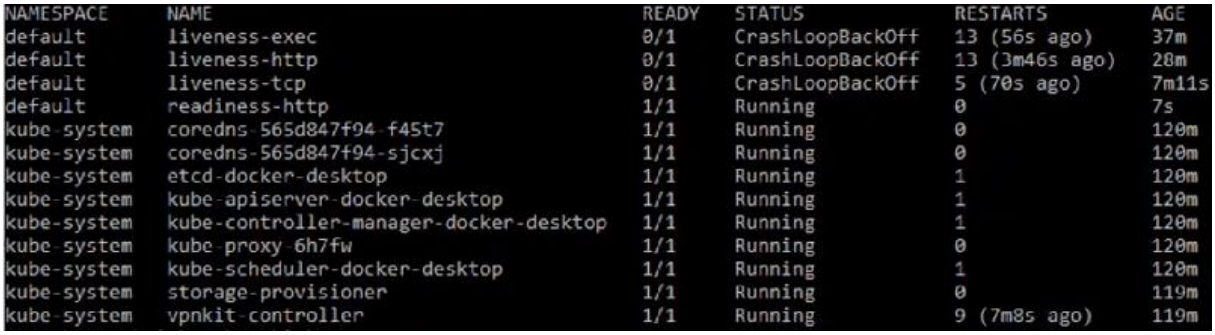
value: myapplication1.com

port: 80

25. Run kubectl create –f readiness_http.yaml.

```
pod/readiness-http created
```

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



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

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

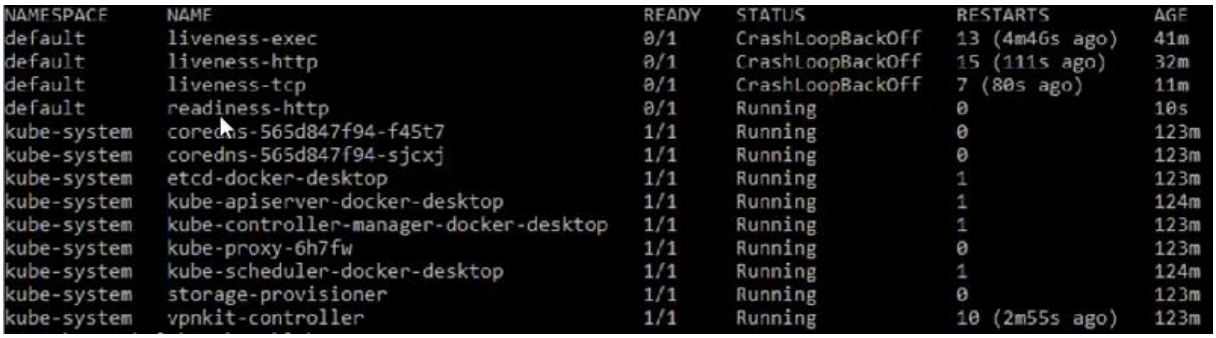
28. Run kubectl describe pod readiness-http. Examine the events for this pod. Everything should be OK.

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

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.



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.

34. Delete all pods under the default namespace.

```
pod "liveness-exec" deleted
pod "liveness-http" deleted
pod "liveness-tcp" deleted
pod "readiness-http" deleted
```

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