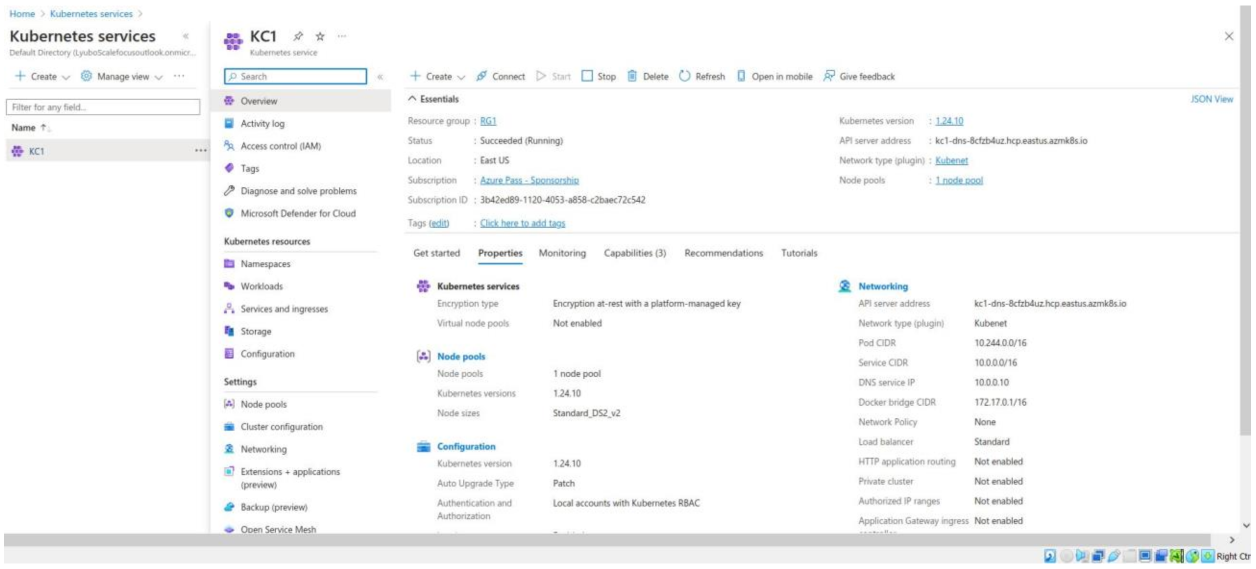
Kubernetes Pods lab

Practice1: Simple pods operations

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



Resource group	: RG1
Status	: Succeeded (Running)
Location	: East US
Subscription	: Azure Pass - Sponsorship
Subscription ID	: 3b42ed89-1120-4053-a858-c2baec72c542
Tags (edit)	: Click here to add tags
Kubernetes version	: 1.24.10
API server address	: kc1-dns-8cfzb4uz.hcp.eastus.azmk8s.io
Network type (plugin)	: Kubenet
Node pools	: 1 node pool

Kubernetes services

Encryption type	Encryption at-rest with a platform-managed key
Virtual node pools	Not enabled

Node pools

Node pools	1 node pool
Kubernetes versions	1.24.10
Node sizes	Standard_DS2_v2

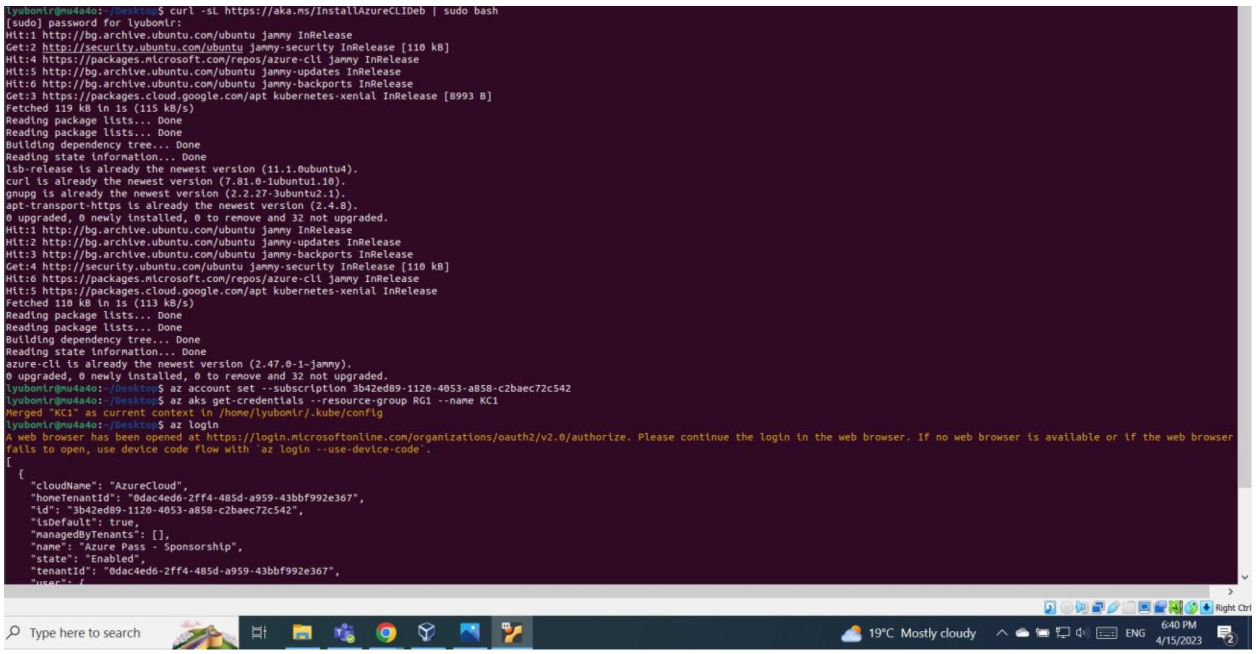
Configuration

Kubernetes version	1.24.10
Auto Upgrade Type	Patch
Authentication and Authorization	Local accounts with Kubernetes RBAC

Networking

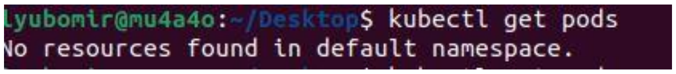
API server address	kc1-dns-8cfzb4uz.hcp.eastus.azmk8s.io
Network type (plugin)	Kubenet
Pod CIDR	10.244.0.0/16
Service CIDR	10.0.0.0/16
DNS service IP	10.0.0.10
Docker bridge CIDR	172.17.0.1/16
Network Policy	None
Load balancer	Standard
HTTP application routing	Not enabled
Private cluster	Not enabled
Authorized IP ranges	Not enabled
Application Gateway ingress	Not enabled

Now you need to connect to your cluster using command line tooling to interact directly with cluster using kubectl, the command line tool for Kubernetes.



```
lyubomir@mu4a4o:~/Desktop$ curl -sL https://aka.ms/InstallAzureCLIDeb | sudo bash
[sudo] password for lyubomir:
Hit:1 http://bg.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:4 https://packages.microsoft.com/repos/azure-cli jammy InRelease
Hit:5 http://bg.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://bg.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:3 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8993 B]
Fetched 119 kB in 1s (115 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
lsb-release is already the newest version (11.1.0ubuntu4).
curl is already the newest version (7.81.0-1ubuntu1.10).
gnupg is already the newest version (2.2.27-3ubuntu2.1).
apt-transport-https is already the newest version (2.4.8).
0 upgraded, 0 newly installed, 0 to remove and 32 not upgraded.
Hit:1 http://bg.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://bg.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://bg.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:6 https://packages.microsoft.com/repos/azure-cli jammy InRelease
Hit:5 https://packages.cloud.google.com/apt kubernetes-xenial InRelease
Fetched 110 kB in 1s (113 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
azure-cli is already the newest version (2.47.0-1~jammy).
0 upgraded, 0 newly installed, 0 to remove and 32 not upgraded.
lyubomir@mu4a4o:~/Desktop$ az account set --subscription 3b42ed89-1120-4053-a858-c2baec72c542
lyubomir@mu4a4o:~/Desktop$ az aks get-credentials --resource-group RG1 --name KC1
Merged "KC1" as current context in /home/lyubomir/.kube/config
lyubomir@mu4a4o:~/Desktop$ az login
A web browser has been opened at https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web browser. If no web browser is available or if the web browser
fails to open, use device code flow with `az login --use-device-code`.
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "0dac4ed6-2ff4-485d-a959-43bbf992e367",
    "id": "3b42ed89-1120-4053-a858-c2baec72c542",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure Pass - Sponsorship",
    "state": "Enabled",
    "tenantId": "0dac4ed6-2ff4-485d-a959-43bbf992e367",
```

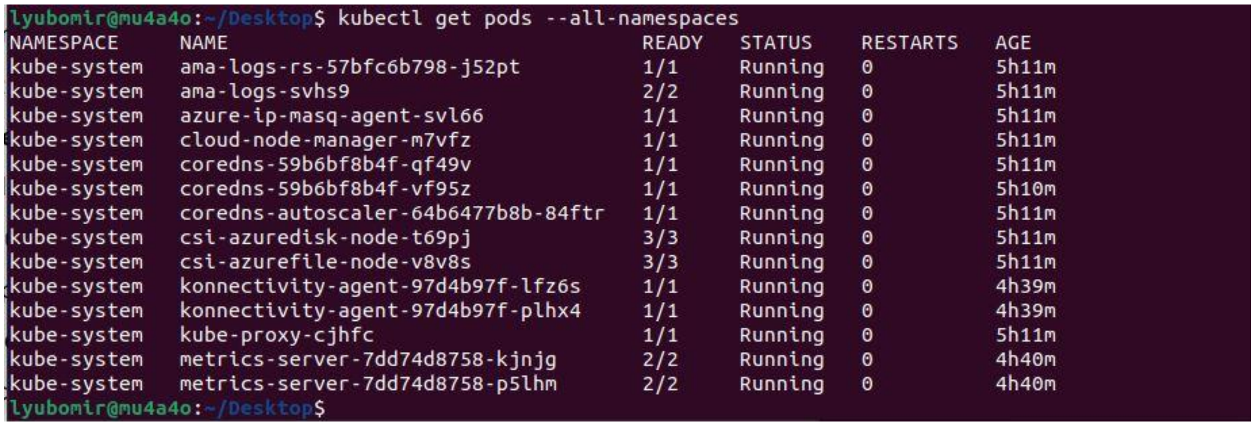
2. Check how many pods run under the default namespace. Run kubectl get pods.



```
lyubomir@mu4a4o:~/Desktop$ 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.



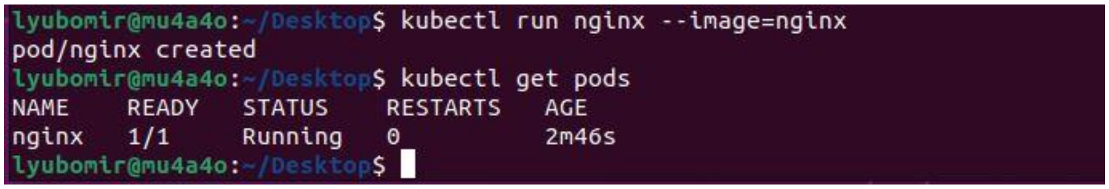
```
lyubomir@mu4a4o:~/Desktop$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                  READY   STATUS    RESTARTS   AGE
kube-system   ama-logs-rs-57bfc6b798-j52pt          1/1     Running   0          5h11m
kube-system   ama-logs-svhs9                        2/2     Running   0          5h11m
kube-system   azure-ip-masq-agent-svl66             1/1     Running   0          5h11m
kube-system   cloud-node-manager-m7vfz              1/1     Running   0          5h11m
kube-system   coredns-59b6bf8b4f-qf49v              1/1     Running   0          5h11m
kube-system   coredns-59b6bf8b4f-vf95z              1/1     Running   0          5h10m
kube-system   coredns-autoscaler-64b6477b8b-84ftr   1/1     Running   0          5h11m
kube-system   csi-azuredisk-node-t69pj              3/3     Running   0          5h11m
kube-system   csi-azurefile-node-v8v8s              3/3     Running   0          5h11m
kube-system   konnectivity-agent-97d4b97f-lfz6s     1/1     Running   0          4h39m
kube-system   konnectivity-agent-97d4b97f-plhx4     1/1     Running   0          4h39m
kube-system   kube-proxy-cjhfc                      1/1     Running   0          5h11m
kube-system   metrics-server-7dd74d8758-kjnjg       2/2     Running   0          4h40m
kube-system   metrics-server-7dd74d8758-p5lhm       2/2     Running   0          4h40m
lyubomir@mu4a4o:~/Desktop$
```

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

I see fourteen pods. The “kube-system” namespace is a special namespace in Kubernetes that contains system-level components,kubelet and other components that are critical for the operation of the Kubernetes cluster.

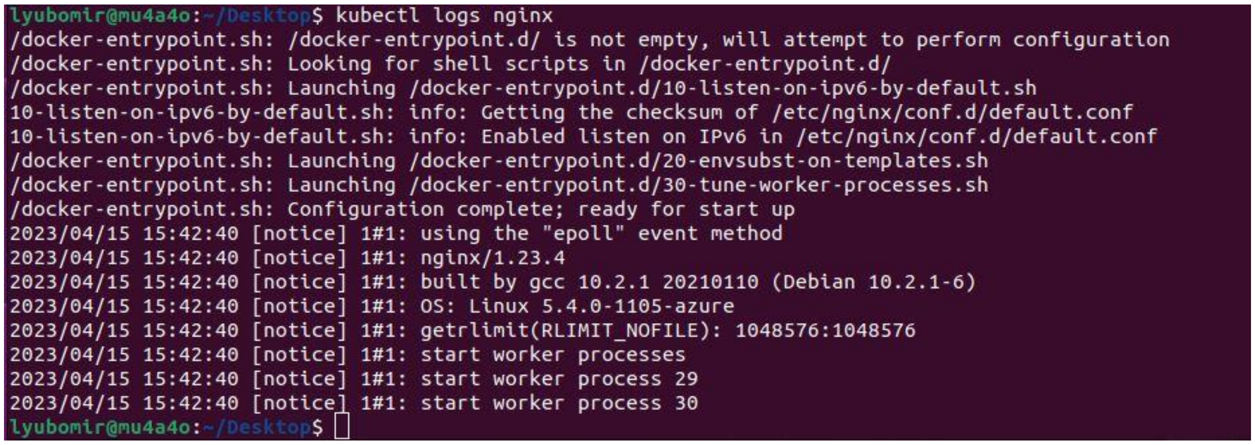
5. Now deploy you first pod using the imperative approach. Run kubectl run nginx --image=nginx

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



```
lyubomir@mu4a4o:~/Desktop$ kubectl run nginx --image=nginx
pod/nginx created
lyubomir@mu4a4o:~/Desktop$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          2m46s
lyubomir@mu4a4o:~/Desktop$
```

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

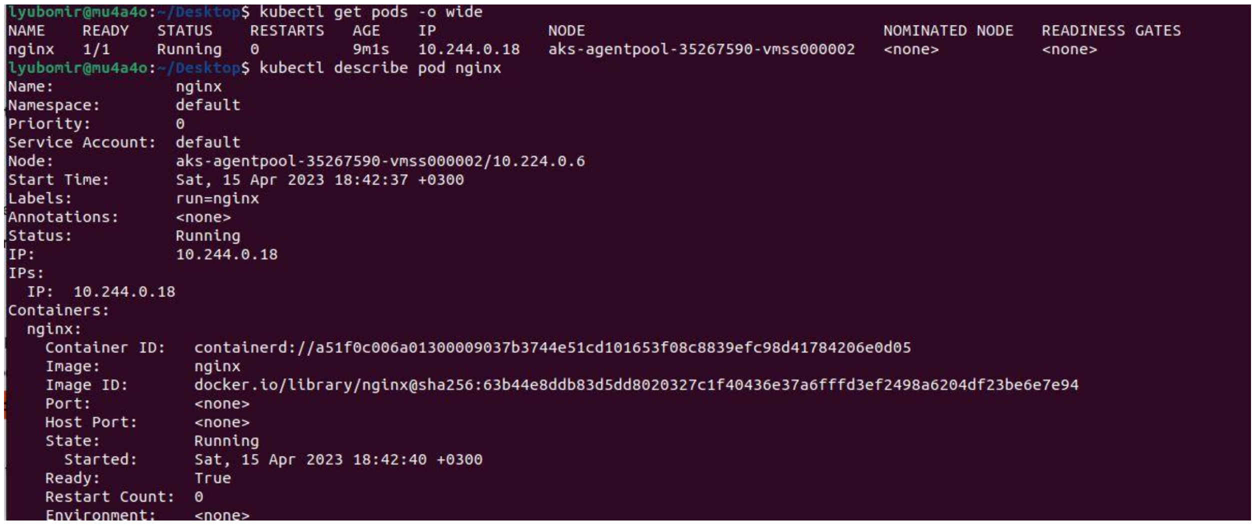


```
lyubomir@mu4a4o:~/Desktop$ kubectl logs nginx
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
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
2023/04/15 15:42:40 [notice] 1#1: using the "epoll" event method
2023/04/15 15:42:40 [notice] 1#1: nginx/1.23.4
2023/04/15 15:42:40 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
2023/04/15 15:42:40 [notice] 1#1: OS: Linux 5.4.0-1105-azure
2023/04/15 15:42:40 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2023/04/15 15:42:40 [notice] 1#1: start worker processes
2023/04/15 15:42:40 [notice] 1#1: start worker process 29
2023/04/15 15:42:40 [notice] 1#1: start worker process 30
lyubomir@mu4a4o:~/Desktop$
```

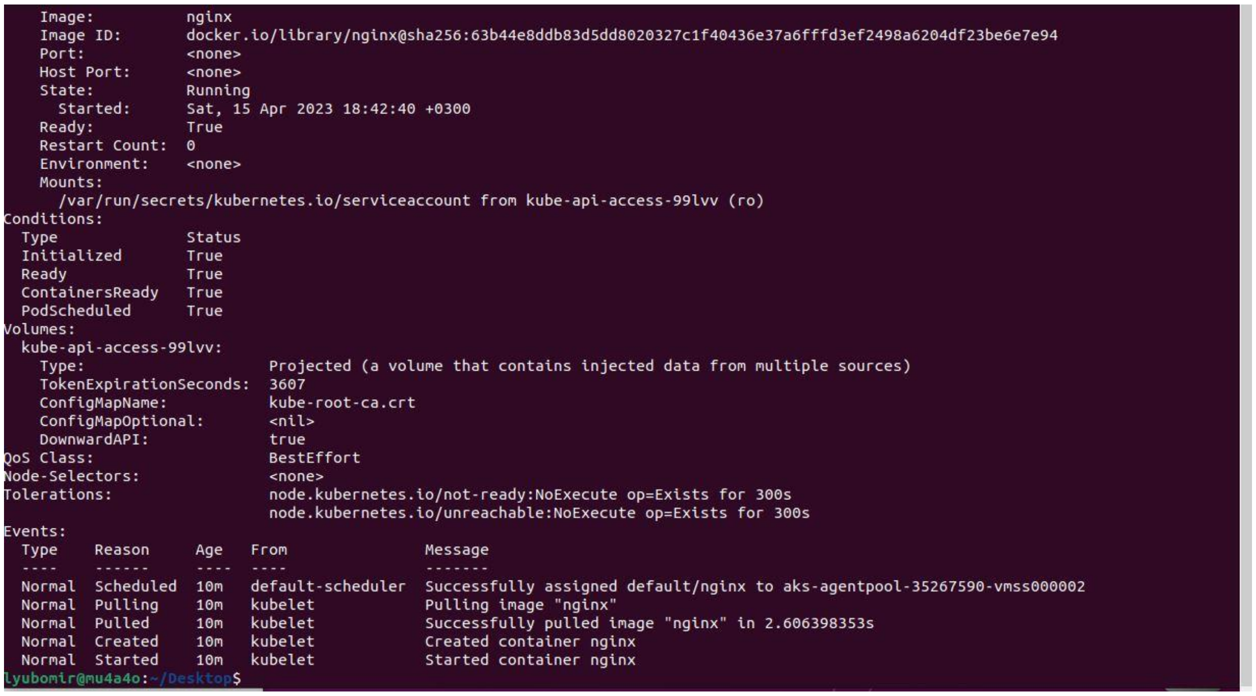
8. Run following command to check current resource consumption of your pod: kubectl top pod nginx.

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

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

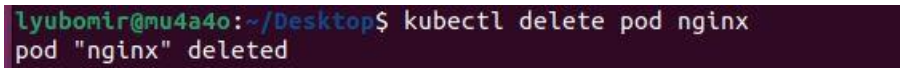


```
lyubomir@mu4a4o:~/Desktop$ kubectl get pods -o wide
NAME    READY   STATUS    RESTARTS   AGE    IP            NODE                                NOMINATED NODE   READINESS GATES
nginx   1/1     Running   0          9m1s   10.244.0.18   aks-agentpool-35267590-vmss000002   <none>           <none>
lyubomir@mu4a4o:~/Desktop$ kubectl describe pod nginx
Name:             nginx
Namespace:        default
Priority:         0
Service Account:  default
Node:             aks-agentpool-35267590-vmss000002/10.224.0.6
Start Time:       Sat, 15 Apr 2023 18:42:37 +0300
Labels:           run=nginx
Annotations:      <none>
Status:           Running
IP:               10.244.0.18
IPs:
  IP:  10.244.0.18
Containers:
  nginx:
    Container ID:   containerd://a51f0c006a01300009037b3744e51cd101653f08c8839efc98d41784206e0d05
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:63b44e8ddb83d5dd8020327c1f40436e37a6fffd3ef2498a6204df23be6e7e94
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sat, 15 Apr 2023 18:42:40 +0300
    Ready:          True
    Restart Count:  0
    Environment:    <none>
```



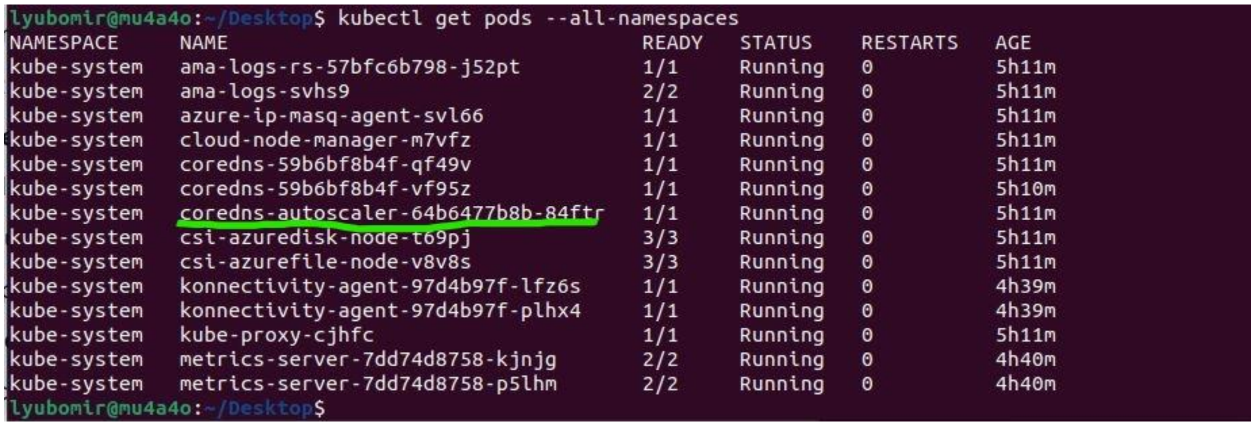
```
    Image:          nginx
    Image ID:       docker.io/library/nginx@sha256:63b44e8ddb83d5dd8020327c1f40436e37a6fffd3ef2498a6204df23be6e7e94
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sat, 15 Apr 2023 18:42:40 +0300
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-99lvv (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  kube-api-access-99lvv:
    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  10m   default-scheduler  Successfully assigned default/nginx to aks-agentpool-35267590-vmss000002
  Normal  Pulling    10m   kubelet            Pulling image "nginx"
  Normal  Pulled     10m   kubelet            Successfully pulled image "nginx" in 2.606398353s
  Normal  Created    10m   kubelet            Created container nginx
  Normal  Started    10m   kubelet            Started container nginx
lyubomir@mu4a4o:~/Desktop$
```

11. Delete your pod using kubectl delete pod nginx.



```
lyubomir@mu4a4o:~/Desktop$ 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.

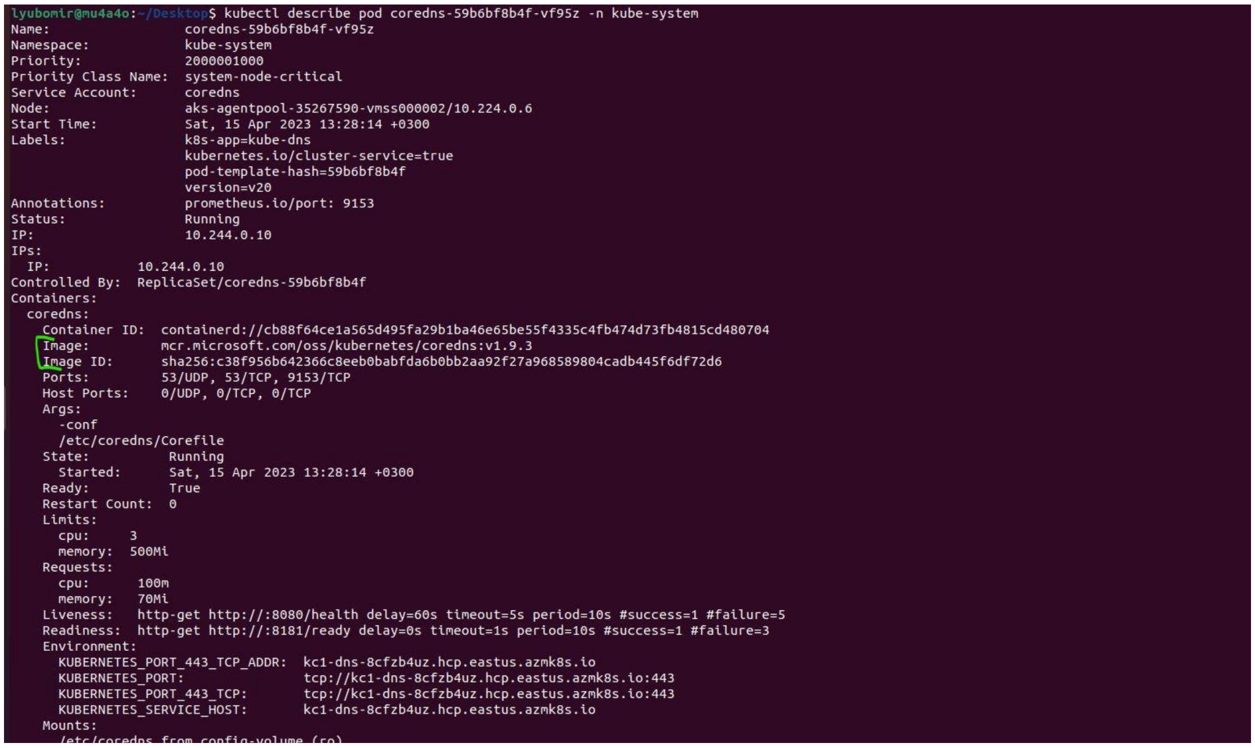


```
lyubomir@mu4a4o:~/Desktop$ kubectl get pods --all-namespaces
NAMESPACE     NAME                                   READY   STATUS    RESTARTS   AGE
kube-system   ama-logs-rs-57bfc6b798-j52pt           1/1     Running   0          5h11m
kube-system   ama-logs-svhs9                         2/2     Running   0          5h11m
kube-system   azure-ip-masq-agent-svl66              1/1     Running   0          5h11m
kube-system   cloud-node-manager-m7vfz               1/1     Running   0          5h11m
kube-system   coredns-59b6bf8b4f-qf49v               1/1     Running   0          5h11m
kube-system   coredns-59b6bf8b4f-vf95z               1/1     Running   0          5h10m
kube-system   coredns-autoscaler-64b6477b8b-84ftr    1/1     Running   0          5h11m
kube-system   csi-azuredisk-node-t69pj               3/3     Running   0          5h11m
kube-system   csi-azurefile-node-v8v8s               3/3     Running   0          5h11m
kube-system   konnectivity-agent-97d4b97f-lfz6s      1/1     Running   0          4h39m
kube-system   konnectivity-agent-97d4b97f-plhx4      1/1     Running   0          4h39m
kube-system   kube-proxy-cjhfc                       1/1     Running   0          5h11m
kube-system   metrics-server-7dd74d8758-kjnjg        2/2     Running   0          4h40m
kube-system   metrics-server-7dd74d8758-p5lhm        2/2     Running   0          4h40m
lyubomir@mu4a4o:~/Desktop$
```

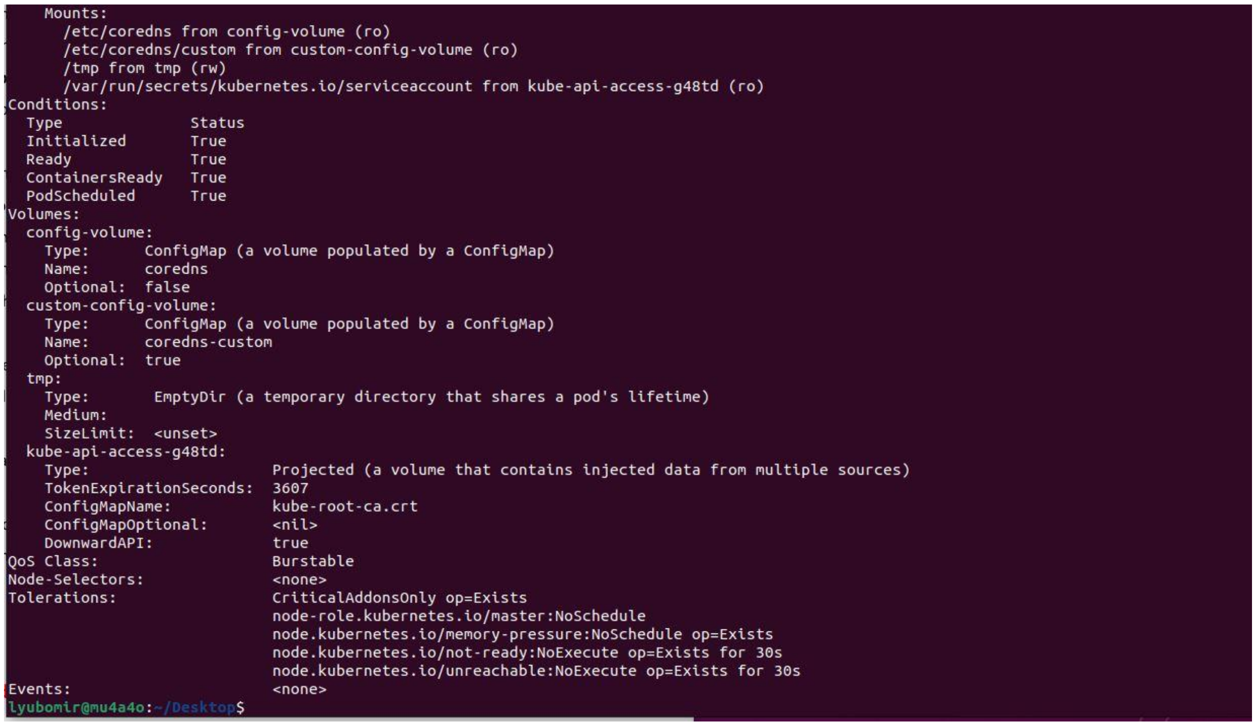
13. Once again list all pods under all namespaces.

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.

15. Inspect the output and locate the image information.

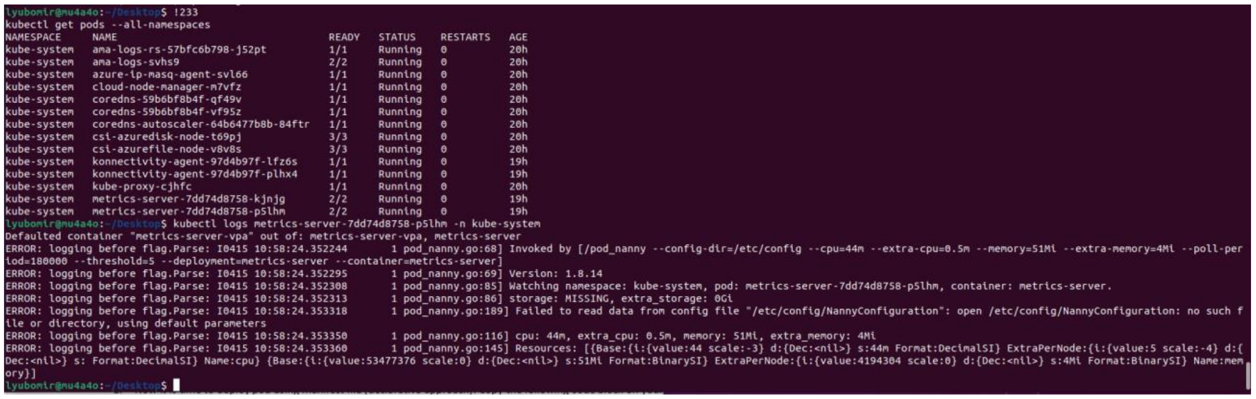


```
lyubomir@mu4a4o:~/Desktop$ kubectl describe pod coredns-59b6bf8b4f-vf95z -n kube-system
Name:                 coredns-59b6bf8b4f-vf95z
Namespace:            kube-system
Priority:             2000001000
Priority Class Name:  system-node-critical
Service Account:      coredns
Node:                 aks-agentpool-35267590-vmss000002/10.224.0.6
Start Time:           Sat, 15 Apr 2023 13:28:14 +0300
Labels:               k8s-app=kube-dns
                      kubernetes.io/cluster-service=true
                      pod-template-hash=59b6bf8b4f
                      version=v20
Annotations:          prometheus.io/port: 9153
Status:               Running
IP:                   10.244.0.10
IPs:
  IP:           10.244.0.10
Controlled By:  ReplicaSet/coredns-59b6bf8b4f
Containers:
  coredns:
    Container ID:  containerd://cb88f64ce1a565d495fa29b1ba46e65be55f4335c4fb474d73fb4815cd480704
    Image:         mcr.microsoft.com/oss/kubernetes/coredns:v1.9.3
    Image ID:      sha256:c38f956b642366c8eeb0babfda6b0bb2aa92f27a968589804cadb445f6df72d6
    Ports:         53/UDP, 53/TCP, 9153/TCP
    Host Ports:    0/UDP, 0/TCP, 0/TCP
    Args:
      -conf
      /etc/coredns/Corefile
    State:          Running
      Started:      Sat, 15 Apr 2023 13:28:14 +0300
    Ready:          True
    Restart Count:  0
    Limits:
      cpu:     3
      memory:  500Mi
    Requests:
      cpu:      100m
      memory:   70Mi
    Liveness:   http-get http://:8080/health delay=60s timeout=5s period=10s #success=1 #failure=5
    Readiness:  http-get http://:8181/ready delay=0s timeout=1s period=10s #success=1 #failure=3
    Environment:
      KUBERNETES_PORT_443_TCP_ADDR:  kc1-dns-8cfzb4uz.hcp.eastus.azmk8s.io
      KUBERNETES_PORT:               tcp://kc1-dns-8cfzb4uz.hcp.eastus.azmk8s.io:443
      KUBERNETES_PORT_443_TCP:       tcp://kc1-dns-8cfzb4uz.hcp.eastus.azmk8s.io:443
      KUBERNETES_SERVICE_HOST:       kc1-dns-8cfzb4uz.hcp.eastus.azmk8s.io
    Mounts:
```



```
    Mounts:
      /etc/coredns from config-volume (ro)
      /etc/coredns/custom from custom-config-volume (ro)
      /tmp from tmp (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-g48td (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  config-volume:
    Type:      ConfigMap (a volume populated by a ConfigMap)
    Name:      coredns
    Optional:  false
  custom-config-volume:
    Type:      ConfigMap (a volume populated by a ConfigMap)
    Name:      coredns-custom
    Optional:  true
  tmp:
    Type:       EmptyDir (a temporary directory that shares a pod's lifetime)
    Medium:
    SizeLimit:  <unset>
  kube-api-access-g48td:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 CriticalAddonsOnly op=Exists
                             node-role.kubernetes.io/master:NoSchedule
                             node.kubernetes.io/memory-pressure:NoSchedule op=Exists
                             node.kubernetes.io/not-ready:NoExecute op=Exists for 30s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 30s
Events:                      <none>
lyubomir@mu4a4o:~/Desktop$
```

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.



```
lyubomir@mu4a4o:~/Desktop$ !233
kubectl get pods --all-namespaces
NAMESPACE     NAME                                   READY   STATUS    RESTARTS   AGE
kube-system   ama-logs-rs-57bfc6b798-j52pt           1/1     Running   0          20h
kube-system   ama-logs-svhs9                         2/2     Running   0          20h
kube-system   azure-ip-masq-agent-svl66              1/1     Running   0          20h
kube-system   cloud-node-manager-m7vfz               1/1     Running   0          20h
kube-system   coredns-59b6bf8b4f-qf49v               1/1     Running   0          20h
kube-system   coredns-59b6bf8b4f-vf95z               1/1     Running   0          20h
kube-system   coredns-autoscaler-64b6477b8b-84ftr    1/1     Running   0          20h
kube-system   csi-azuredisk-node-t69pj               3/3     Running   0          20h
kube-system   csi-azurefile-node-v8v8s               3/3     Running   0          20h
kube-system   konnectivity-agent-97d4b97f-lfz6s      1/1     Running   0          19h
kube-system   konnectivity-agent-97d4b97f-plhx4      1/1     Running   0          19h
kube-system   kube-proxy-cjhfc                       1/1     Running   0          20h
kube-system   metrics-server-7dd74d8758-kjnjg        2/2     Running   0          19h
kube-system   metrics-server-7dd74d8758-p5lhm        2/2     Running   0          19h
lyubomir@mu4a4o:~/Desktop$ kubectl logs metrics-server-7dd74d8758-p5lhm -n kube-system
Defaulted container "metrics-server-vpa" out of: metrics-server-vpa, metrics-server
ERROR: logging before flag.Parse: I0415 10:58:24.352244       1 pod_nanny.go:68] Invoked by [/pod_nanny --config-dir=/etc/config --cpu=44m --extra-cpu=0.5m --memory=51Mi --extra-memory=4Mi --poll-per
iod=180000 --threshold=5 --deployment=metrics-server --container=metrics-server]
ERROR: logging before flag.Parse: I0415 10:58:24.352295       1 pod_nanny.go:69] Version: 1.8.14
ERROR: logging before flag.Parse: I0415 10:58:24.352308       1 pod_nanny.go:85] Watching namespace: kube-system, pod: metrics-server-7dd74d8758-p5lhm, container: metrics-server.
ERROR: logging before flag.Parse: I0415 10:58:24.352313       1 pod_nanny.go:86] storage: MISSING, extra_storage: 0Gi
ERROR: logging before flag.Parse: I0415 10:58:24.353318       1 pod_nanny.go:189] Failed to read data from config file "/etc/config/NannyConfiguration": open /etc/config/NannyConfiguration: no such f
ile or directory, using default parameters
ERROR: logging before flag.Parse: I0415 10:58:24.353350       1 pod_nanny.go:116] cpu: 44m, extra_cpu: 0.5m, memory: 51Mi, extra_memory: 4Mi
ERROR: logging before flag.Parse: I0415 10:58:24.353360       1 pod_nanny.go:145] Resources: [{Base:{i:{value:44 scale:-3} d:{Dec:<nil>} s:44m Format:DecimalSI} ExtraPerNode:{i:{value:5 scale:-4} d:{
Dec:<nil>} s: Format:DecimalSI} Name:cpu} {Base:{i:{value:53477376 scale:0} d:{Dec:<nil>} s:51Mi Format:BinarySI} ExtraPerNode:{i:{value:4194304 scale:0} d:{Dec:<nil>} s:4Mi Format:BinarySI} Name:mem
ory}]
lyubomir@mu4a4o:~/Desktop$
```

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!

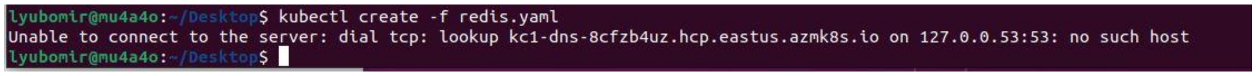
1. Now it is time to deploy pod using manifest file (declarative approach). Copy the following code block on your local computer in a file called redis.yaml:

```
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.

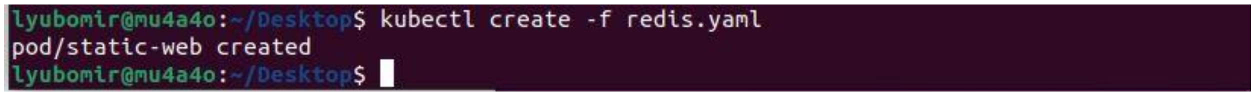
3. You will receive errors on your screen. Your next task will be to correct the syntax of the code you just

copied. You can use the online Kubernetes documentation or you can search the internet in general.

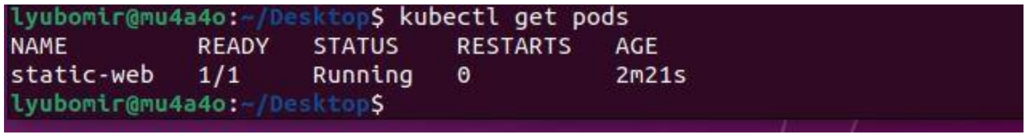


```
lyubomir@mu4a4o:~/Desktop$ kubectl create -f redis.yaml
Unable to connect to the server: dial tcp: lookup kc1-dns-8cfzb4uz.hcp.eastus.azmk8s.io on 127.0.0.53:53: no such host
lyubomir@mu4a4o:~/Desktop$
```

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



```
lyubomir@mu4a4o:~/Desktop$ kubectl create -f redis.yaml
pod/static-web created
lyubomir@mu4a4o:~/Desktop$
```



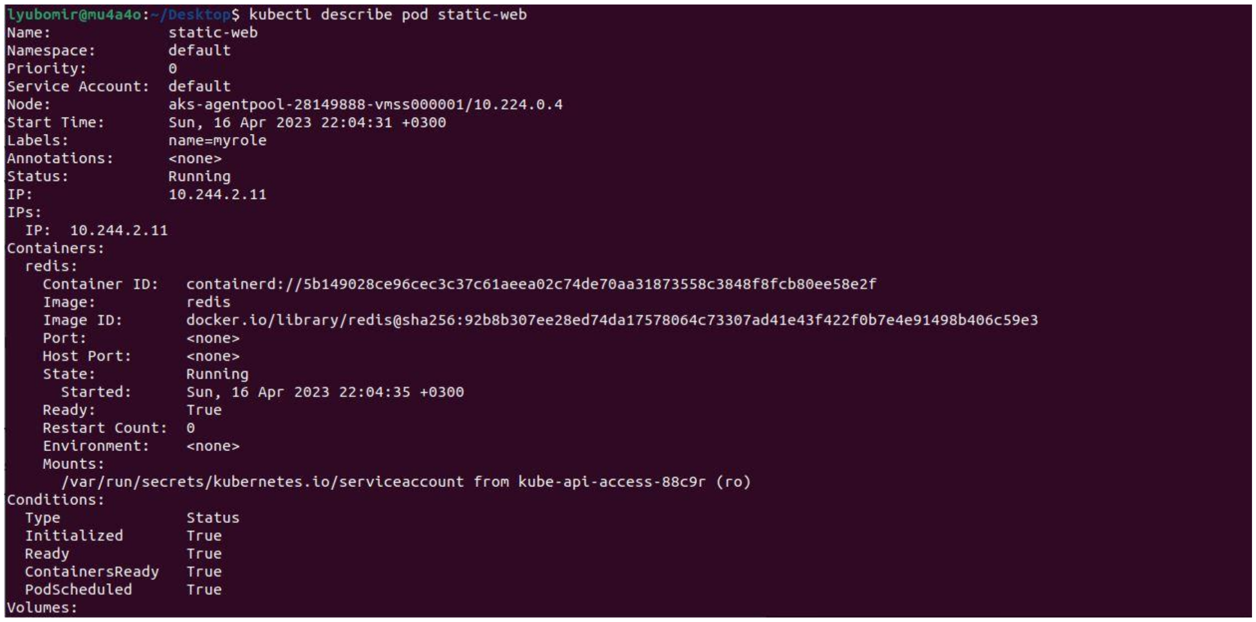
```
lyubomir@mu4a4o:~/Desktop$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
static-web   1/1     Running   0          2m21s
lyubomir@mu4a4o:~/Desktop$
```

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?

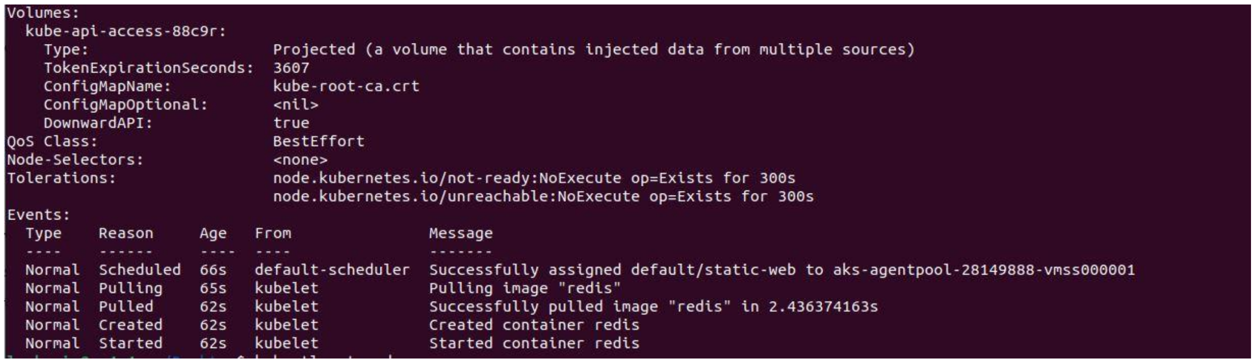
6. Find the correct image (check the Docker hub page) and correct it in the manifest.

7. Locate the image information and put the correct image name. Redeploy the pod (first 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.

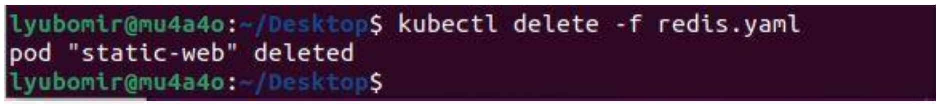


```
lyubomir@mu4a4o:~/Desktop$ kubectl describe pod static-web
Name:             static-web
Namespace:        default
Priority:         0
Service Account:  default
Node:             aks-agentpool-28149888-vmss000001/10.224.0.4
Start Time:       Sun, 16 Apr 2023 22:04:31 +0300
Labels:           name=myrole
Annotations:      <none>
Status:           Running
IP:               10.244.2.11
IPs:
  IP:  10.244.2.11
Containers:
  redis:
    Container ID:   containerd://5b149028ce96cec3c37c61aeea02c74de70aa31873558c3848f8fcb80ee58e2f
    Image:          redis
    Image ID:       docker.io/library/redis@sha256:92b8b307ee28ed74da17578064c73307ad41e43f422f0b7e4e91498b406c59e3
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Sun, 16 Apr 2023 22:04:35 +0300
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-88c9r (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
```



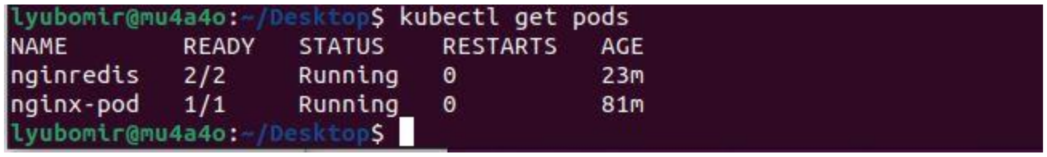
```
Volumes:
  kube-api-access-88c9r:
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   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  66s   default-scheduler  Successfully assigned default/static-web to aks-agentpool-28149888-vmss000001
  Normal  Pulling    65s   kubelet            Pulling image "redis"
  Normal  Pulled     62s   kubelet            Successfully pulled image "redis" in 2.436374163s
  Normal  Created    62s   kubelet            Created container redis
  Normal  Started    62s   kubelet            Started container redis
```

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



```
lyubomir@mu4a4o:~/Desktop$ kubectl delete -f redis.yaml
pod "static-web" deleted
lyubomir@mu4a4o:~/Desktop$
```

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.



```
lyubomir@mu4a4o:~/Desktop$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
nginredis    2/2     Running   0          23m
nginx-pod    1/1     Running   0          81m
lyubomir@mu4a4o:~/Desktop$
```

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

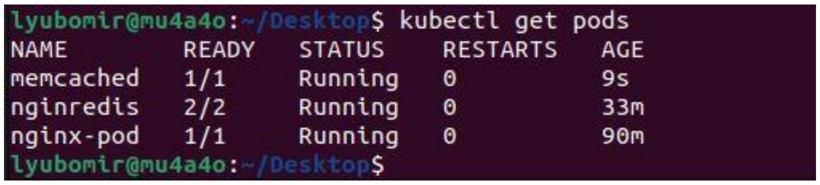
- RAM request=0.15 GB

- CPU limit=0.5 cores

- Ram limit=0.25 GB

- Restart policy=Never

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



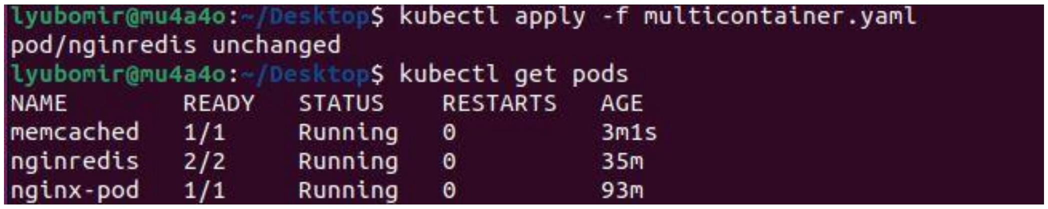
```
lyubomir@mu4a4o:~/Desktop$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
memcached    1/1     Running   0          9s
nginredis    2/2     Running   0          33m
nginx-pod    1/1     Running   0          90m
lyubomir@mu4a4o:~/Desktop$
```

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.

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.

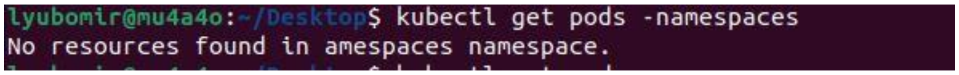
3. Deploy your multi-container pod. It should have running status. What is written under Ready column when you kubectl get the pods? Why your pod displays different values for ready?



```
lyubomir@mu4a4o:~/Desktop$ kubectl apply -f multicontainer.yaml
pod/nginredis unchanged
lyubomir@mu4a4o:~/Desktop$ kubectl get pods
NAME         READY   STATUS    RESTARTS   AGE
memcached    1/1     Running   0          3m1s
nginredis    2/2     Running   0          35m
nginx-pod    1/1     Running   0          93m
```

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

5. Delete all the pods under the default namespace.



```
lyubomir@mu4a4o:~/Desktop$ kubectl get pods -namespaces
No resources found in amespaces 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:

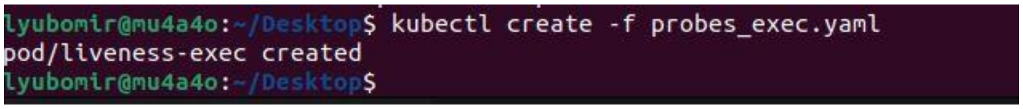
apiVersion: v1

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
```

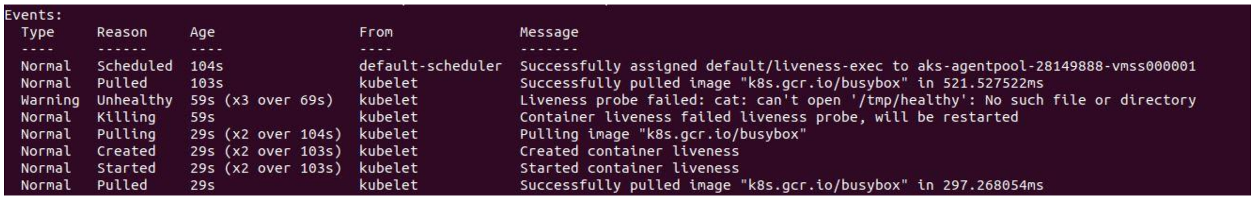
2. Examine the containers args commands especially the line that start with touch. This bash pipeline will help us to test the liveness probes.

3. Run kubectl create –f probes_exec.yaml.



```
lyubomir@mu4a4o:~/Desktop$ kubectl create -f probes_exec.yaml
pod/liveness-exec created
lyubomir@mu4a4o:~/Desktop$
```

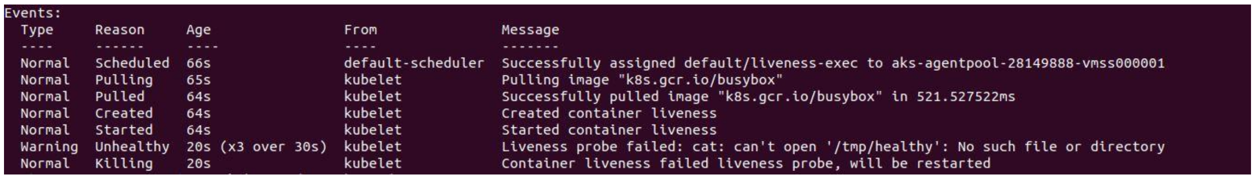
4. Run kubectl describe pod liveness-exec immediately after you deploy the pod. The output should indicate that no liveness probes have failed yet.



```
Events:
  Type     Reason     Age                 From               Message
  ----     ------     ----                ----               -------
  Normal   Scheduled  104s                default-scheduler  Successfully assigned default/liveness-exec to aks-agentpool-28149888-vmss000001
  Normal   Pulled     103s                kubelet            Successfully pulled image "k8s.gcr.io/busybox" in 521.527522ms
  Warning  Unhealthy  59s (x3 over 69s)   kubelet            Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory
  Normal   Killing    59s                 kubelet            Container liveness failed liveness probe, will be restarted
  Normal   Pulling    29s (x2 over 104s)  kubelet            Pulling image "k8s.gcr.io/busybox"
  Normal   Created    29s (x2 over 103s)  kubelet            Created container liveness
  Normal   Started    29s (x2 over 103s)  kubelet            Started container liveness
  Normal   Pulled     29s                 kubelet            Successfully pulled image "k8s.gcr.io/busybox" in 297.268054ms
```

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  66s                default-scheduler  Successfully assigned default/liveness-exec to aks-agentpool-28149888-vmss000001
  Normal   Pulling    65s                kubelet            Pulling image "k8s.gcr.io/busybox"
  Normal   Pulled     64s                kubelet            Successfully pulled image "k8s.gcr.io/busybox" in 521.527522ms
  Normal   Created    64s                kubelet            Created container liveness
  Normal   Started    64s                kubelet            Started container liveness
  Warning  Unhealthy  20s (x3 over 30s)  kubelet            Liveness probe failed: cat: can't open '/tmp/healthy': No such file or directory
  Normal   Killing    20s                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 livenessexec.

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:

```
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)

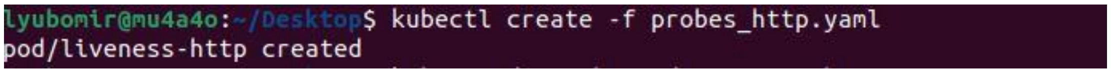
 w.Write([]byte("ok"))

 }

})

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.



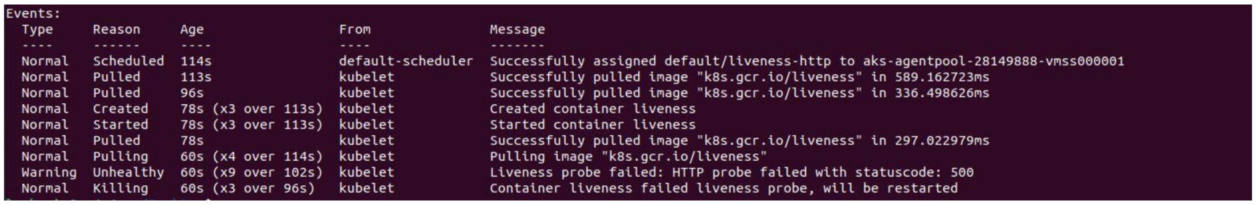
```
lyubomir@mu4a4o:~/Desktop$ kubectl create -f probes_http.yaml
pod/liveness-http created
```

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

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.

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



```
Events:
  Type     Reason     Age                 From               Message
  ----     ------     ----                ----               -------
  Normal   Scheduled  114s                default-scheduler  Successfully assigned default/liveness-http to aks-agentpool-28149888-vmss000001
  Normal   Pulled     113s                kubelet            Successfully pulled image "k8s.gcr.io/liveness" in 589.162723ms
  Normal   Pulled     96s                 kubelet            Successfully pulled image "k8s.gcr.io/liveness" in 336.498626ms
  Normal   Created    78s (x3 over 113s)  kubelet            Created container liveness
  Normal   Started    78s (x3 over 113s)  kubelet            Started container liveness
  Normal   Pulled     78s                 kubelet            Successfully pulled image "k8s.gcr.io/liveness" in 297.022979ms
  Normal   Pulling    60s (x4 over 114s)  kubelet            Pulling image "k8s.gcr.io/liveness"
  Warning  Unhealthy  60s (x9 over 102s)  kubelet            Liveness probe failed: HTTP probe failed with statuscode: 500
  Normal   Killing    60s (x3 over 96s)   kubelet            Container liveness failed liveness probe, will be restarted
```

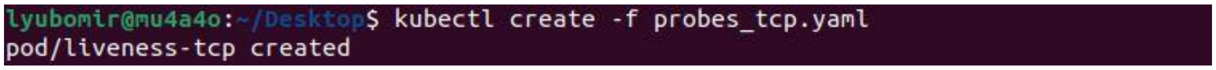
17. We continue with TCP probes. Create file named probes_tcp.yaml with following content:

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: 15
 periodSeconds: 20
```

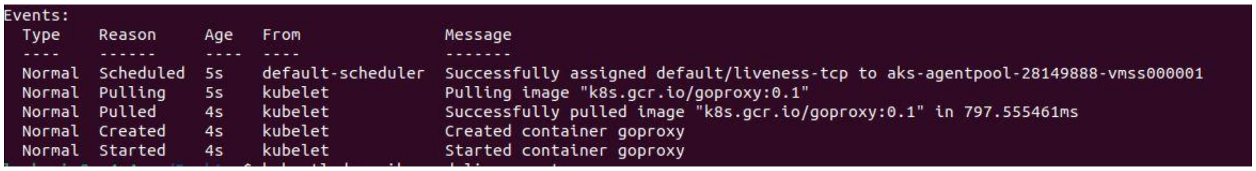
18. Run kubectl create –f probes_tcp.yaml.



```
lyubomir@mu4a4o:~/Desktop$ kubectl create -f probes_tcp.yaml
pod/liveness-tcp created
```

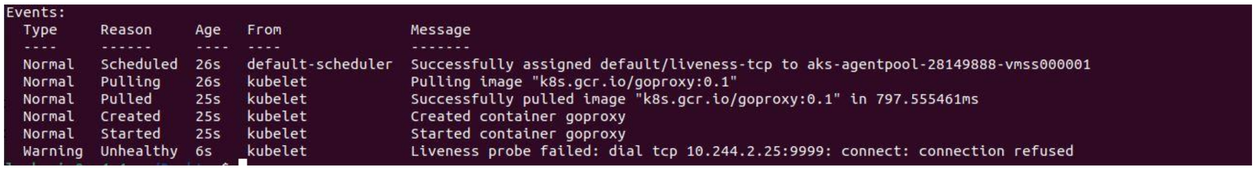
19. Immediately run (you only have 10 secs to run this command) kubectl describe pod liveness-tcp.

20. Your pod should be live and running.



```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  5s    default-scheduler  Successfully assigned default/liveness-tcp to aks-agentpool-28149888-vmss000001
  Normal  Pulling    5s    kubelet            Pulling image "k8s.gcr.io/goproxy:0.1"
  Normal  Pulled     4s    kubelet            Successfully pulled image "k8s.gcr.io/goproxy:0.1" in 797.555461ms
  Normal  Created    4s    kubelet            Created container goproxy
  Normal  Started    4s    kubelet            Started container goproxy
```

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  26s   default-scheduler  Successfully assigned default/liveness-tcp to aks-agentpool-28149888-vmss000001
  Normal   Pulling    26s   kubelet            Pulling image "k8s.gcr.io/goproxy:0.1"
  Normal   Pulled     25s   kubelet            Successfully pulled image "k8s.gcr.io/goproxy:0.1" in 797.555461ms
  Normal   Created    25s   kubelet            Created container goproxy
  Normal   Started    25s   kubelet            Started container goproxy
  Warning  Unhealthy  6s    kubelet            Liveness probe failed: dial tcp 10.244.2.25:9999: connect: connection refused
```

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.

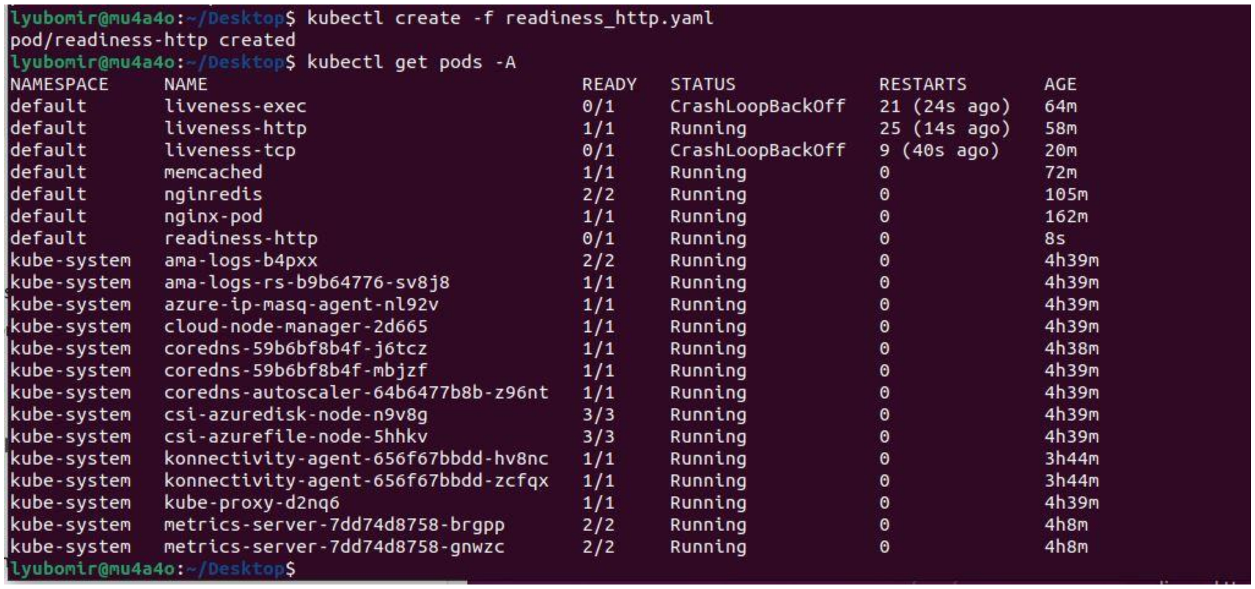
27. Pods and their status and ready states will be displayed; our pod should be in running state.

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.



```
lyubomir@mu4a4o:~/Desktop$ kubectl create -f readiness_http.yaml
pod/readiness-http created
lyubomir@mu4a4o:~/Desktop$ kubectl get pods -A
NAMESPACE     NAME                                  READY   STATUS             RESTARTS       AGE
default       liveness-exec                         0/1     CrashLoopBackOff   21 (24s ago)   64m
default       liveness-http                         1/1     Running            25 (14s ago)   58m
default       liveness-tcp                          0/1     CrashLoopBackOff   9 (40s ago)    20m
default       memcached                             1/1     Running            0              72m
default       nginredis                             2/2     Running            0              105m
default       nginx-pod                             1/1     Running            0              162m
default       readiness-http                        0/1     Running            0              8s
kube-system   ama-logs-b4pxx                        2/2     Running            0              4h39m
kube-system   ama-logs-rs-b9b64776-sv8j8            1/1     Running            0              4h39m
kube-system   azure-ip-masq-agent-nl92v             1/1     Running            0              4h39m
kube-system   cloud-node-manager-2d665              1/1     Running            0              4h39m
kube-system   coredns-59b6bf8b4f-j6tcz              1/1     Running            0              4h38m
kube-system   coredns-59b6bf8b4f-mbjzf              1/1     Running            0              4h39m
kube-system   coredns-autoscaler-64b6477b8b-z96nt   1/1     Running            0              4h39m
kube-system   csi-azuredisk-node-n9v8g              3/3     Running            0              4h39m
kube-system   csi-azurefile-node-5hhkv              3/3     Running            0              4h39m
kube-system   konnectivity-agent-656f67bbdd-hv8nc   1/1     Running            0              3h44m
kube-system   konnectivity-agent-656f67bbdd-zcfqx   1/1     Running            0              3h44m
kube-system   kube-proxy-d2nq6                      1/1     Running            0              4h39m
kube-system   metrics-server-7dd74d8758-brgpp       2/2     Running            0              4h8m
kube-system   metrics-server-7dd74d8758-gnwzc       2/2     Running            0              4h8m
lyubomir@mu4a4o:~/Desktop$
```

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.

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