Mohanad Mahmoud Sayed

Lab4

1- How many ConfigMaps exist in the environment?

```
controlplane:~$ kubectl get configmaps --all-namespaces
NAMESPACE            NAME                                                   DATA   AGE
default              kube-root-ca.crt                                       1      33d
kube-node-lease      kube-root-ca.crt                                       1      33d
kube-public          cluster-info                                           2      33d
kube-public          kube-root-ca.crt                                       1      33d
kube-system          canal-config                                           6      33d
kube-system          coredns                                                1      33d
kube-system          extension-apiserver-authentication                     6      33d
kube-system          kube-apiserver-legacy-service-account-token-tracking   1      33d
kube-system          kube-proxy                                             2      33d
kube-system          kube-root-ca.crt                                       1      33d
kube-system          kubeadm-config                                         1      33d
kube-system          kubelet-config                                         1      33d
local-path-storage   kube-root-ca.crt                                       1      33d
local-path-storage   local-path-config                                      4      33d
controlplane:~$
```

2- Create a new ConfigMap Use the spec given below.

ConfigName Name: webapp-config-map

Data: APP_COLOR=darkblue

Firstly, create webapp-config-map.yml:

```
Editor   Tab 1   +
 GNU nano 7.2                                        webapp-config-map.yml *
apiVersion: v1
kind: ConfigMap
metadata:
  name: webapp-config-map
data:
  APP_COLOR: darkblue
```

Then, apply and check:

```
controlplane:~$ kubectl apply -f webapp-config-map.yml
configmap/webapp-config-map created
controlplane:~$ kubectl get configmap
NAME                 DATA   AGE
kube-root-ca.crt     1      33d
webapp-config-map    1      20s
controlplane:~$
```

3- Create a webapp-color POD with nginx image and use the created ConfigMap

Firstly, create webapp-config-map.yml:

```
Editor   Tab 1   +
  GNU nano 7.2                                          webapp-color.yml *
apiVersion: v1
kind: Pod
metadata:
  name: webapp-color
spec:
  containers:
    - name: nginx-container
      image: nginx
      env:
        - name: APP_COLOR
          valueFrom:
            configMapKeyRef:
              name: webapp-config-map
              key: APP_COLOR
```

Then, apply and check:

```
controlplane:~$ kubectl apply -f webapp-color.yml
pod/webapp-color created
controlplane:~$ k get pods
NAME           READY   STATUS    RESTARTS   AGE
webapp-color   1/1     Running   0          13s
```

Finally, using command (kubectl describe pod webapp-color)

```
Environment:
  APP_COLOR:  <set to the key 'APP_COLOR' of config map 'webapp-config-map'>
```

4-How many Secrets exist on the system?

```
Editor   Tab 1   +
controlplane:~$ kubectl get secrets --all-namespaces
NAMESPACE     NAME                  TYPE                            DATA   AGE
kube-system   bootstrap-token-fa18uz   bootstrap.kubernetes.io/token   5      33d
controlplane:~$
```

5- How many secrets are defined in the default-token secret?

```
controlplane:~$ kubectl get secrets | grep default-token
No resources found in default namespace.
controlplane:~$
```

6- create a POD called db-pod with the image mysql:5.7 then check the

POD status

```
Editor   Tab 1   +
  GNU nano 7.2                                        db-pod-incomplete.yml *
apiVersion: v1
kind: Pod
metadata:
  name: db-pod
spec:
  containers:
    - name: mysql
      image: mysql:5.7
      env:
        - name: MYSQL_ROOT_PASSWORD
          value: "password"
```

Note: This is an **incomplete configuration**, which leads to a crash.

```
controlplane:~$ kubectl apply -f db-pod-incomplete.yml
pod/db-pod created
controlplane:~$ kubectl get pod db-pod
NAME      READY   STATUS              RESTARTS   AGE
db-pod    0/1     ContainerCreating   0          12s
```

→ The pod fails to become Ready

7- Why the db-pod status is not ready

Because it is missing required environment variables like:

- MYSQL_DATABASE
- MYSQL_USER
- MYSQL_PASSWORD

8- Create a new secret named db-secret with the data given below.

Secret Name: db-secret

Secret 1: MYSQL_DATABASE=sql01

Secret 2: MYSQL_USER=user1

Secret3: MYSQL_PASSWORD=password

Secret 4: MYSQL_ROOT_PASSWORD=password123

Firstly, create the secret file:

```
Editor   Tab 1   +
  GNU nano 7.2                                          db-secret.yml *
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
stringData:
  MYSQL_DATABASE: sql01
  MYSQL_USER: user1
  MYSQL_PASSWORD: password
  MYSQL_ROOT_PASSWORD: password123
```

Then, apply and check:

```
controlplane:~$ kubectl apply -f db-secret.yml
secret/db-secret created
controlplane:~$ k get secrets
NAME        TYPE     DATA   AGE
db-secret   Opaque   4      3s
controlplane:~$
```

9- Configure db-pod to load environment variables from the newly created secret.

Delete and recreate the pod if required.

Firstly, lets edit the pod file to take its env variables from the newly created secret file

```
Editor   Tab 1   +
  GNU nano 7.2                                   db-pod-incomplete.yml *
apiVersion: v1
kind: Pod
metadata:
  name: db-pod
spec:
  containers:
    - name: mysql
      image: mysql:5.7
      envFrom:
        - secretRef:
            name: db-secret
```

Then, let’s delete the pod and apply and check:

```
controlplane:~$ kubectl delete pod db-pod
pod "db-pod" deleted
controlplane:~$ k get pods
NAME           READY   STATUS    RESTARTS   AGE
webapp-color   1/1     Running   0          21m
controlplane:~$ kubectl apply -f db-pod-incomplete.yml
pod/db-pod created
controlplane:~$ k get pods
NAME           READY   STATUS    RESTARTS   AGE
db-pod         1/1     Running   0          5s
webapp-color   1/1     Running   0          21m
controlplane:~$
```

→The db-pod is now running, and its status has been changed to ready and running

10- Create a multi-container pod with 2 containers.

Name: yellow

Container 1 Name: lemon

Container 1 Image: busybox

Container 2 Name: gold

Container 2 Image: redis

Firstly, create yellow-pod.yml:

```
Editor   Tab 1   +
  GNU nano 7.2                                          yellow-pod.yml
apiVersion: v1
kind: Pod
metadata:
  name: yellow
spec:
  containers:
    - name: lemon
      image: busybox
      command: ["sleep", "3600"]
    - name:  gold
      image: redis
```

Then, apply and check:

```
controlplane:~$ kubectl apply -f yellow-pod.yml
pod/yellow created
controlplane:~$ k get pods
NAME           READY   STATUS    RESTARTS   AGE
db-pod         1/1     Running   0          6m45s
webapp-color   1/1     Running   0          28m
yellow         2/2     Running   0          4s
controlplane:~$
```

11- Create a pod red with redis image and use an initContainer that uses the busybox image and sleeps for 20 seconds

Firstly, create red-pod.yml:

```
Editor   Tab 1   +
  GNU nano 7.2                                                    red-pod.yml *
apiVersion: v1
kind: Pod
metadata:
  name: red
spec:
  initContainers:
    - name: init-busybox
      image: busybox
      command: ["sh", "-c", "sleep 20"]
  containers:
    - name: redis
      image: redis
```

Then, apply and check:

```
controlplane:~$ k apply -f red-pod.yml
pod/red created
```

Before 20 seconds:

```
controlplane:~$ k get pod red
NAME    READY   STATUS     RESTARTS   AGE
red     0/1     Init:0/1   0          13s
controlplane:~$
```

```
Init Containers:
  init-busybox:
    Container ID:  containerd://27d2223d5cc9a9ce023cf204787344827ae764173ab3f097a0a357f8b38028e1
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:37f7b378a29ceb4c551b1b5582e27747b855bbfaa73fa11914fe0df028dc581f
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 20
    State:          Running
      Started:      Fri, 25 Apr 2025 19:23:11 +0000
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vntf8 (ro)
Containers:
  redis:
    Container ID:
    Image:          redis
    Image ID:
    Port:           <none>
    Host Port:      <none>
    State:          Waiting
      Reason:       PodInitializing
    Ready:          False
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vntf8 (ro)
```

➔ The state init container is running while the state of redis container is waiting, because:

1-**Kubelet runs the initContainers** first:

- Only init-busybox runs.
- It executes: sh -c "sleep 20"

2-Main container redis is **not created yet**.

- No container image is pulled or started.
- Resources are held until init-busybox finishes.

After 20 seconds finishes:

```
controlplane:~$ k get pod red
NAME   READY   STATUS    RESTARTS   AGE
red    1/1     Running   0          4m21s
controlplane:~$
```

```
Init Containers:
  init-busybox:
    Container ID:  containerd://27d2223d5cc9a9ce023cf204787344827ae764173ab3f097a0a357f8b38028e1
    Image:         busybox
    Image ID:      docker.io/library/busybox@sha256:37f7b378a29ceb4c551b1b5582e27747b855bbfaa73fa11914fe0df028dc581f
    Port:          <none>
    Host Port:     <none>
    Command:
      sh
      -c
      sleep 20
    State:          Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Fri, 25 Apr 2025 19:23:11 +0000
      Finished:     Fri, 25 Apr 2025 19:23:31 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vntf8 (ro)
Containers:
  redis:
    Container ID:   containerd://2925527aedeb3b74045456dd0e07402cb7037d8e76b909c00586808097726636
    Image:          redis
    Image ID:       docker.io/library/redis@sha256:8bc666424ef252009ed34b0432564cabbd4094cd2ce7829306cb1f5ee69170be
    Port:           <none>
    Host Port:      <none>
    State:          Running
      Started:      Fri, 25 Apr 2025 19:23:32 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-vntf8 (ro)
```

➔ The state init container is completed while the state of redis container is running, because:

The pod controller proceeds to start the main container:

- redis image is pulled
- redis container is started

12- Create a pod named print-envars-greeting.

1. Configure spec as, the container name should be print-env-container and use bash image.

2. Create three environment variables:

a. GREETING and its value should be “Welcome to”

b. COMPANY and its value should be “DevOps”

c. GROUP and its value should be “Industries”

4. Use command to echo ["$(GREETING) $(COMPANY) $(GROUP)"] message.

5. You can check the output using <kubctl logs -f [pod-name]> command.

Firstly, create greeting-pod.yml:

```
Editor   Tab 1   +
GNU nano 7.2                                                    greeting-pod.yml *
apiVersion: v1
kind: Pod
metadata:
  name: print-envars-greeting
spec:
  containers:
    - name: print-env-container
      image: bash
      command: ["sh", "-c", "echo [\"$GREETING $COMPANY $GROUP\"] && sleep 30"]
      env:
        - name: GREETING
          value: "Welcome to"
        - name: COMPANY
          value: "DevOps"
        - name: GROUP
          value: "Industries"
```

Then, apply and check:

```
controlplane:~$ kubectl apply -f greeting-pod.yml
pod/print-envars-greeting created
controlplane:~$ kubectl logs print-envars-greeting
[Welcome to DevOps Industries]
controlplane:~$
```

13- Where is the default kubeconfig file located in the current environment?

```
controlplane:~$ ~/.kube/config
bash: /root/.kube/config: Permission denied
controlplane:~$
```

➔ The default kubeconfig file is located at: /root/.kube/config

14- How many clusters are defined in the default kubeconfig file?

```
controlplane:~$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://172.30.1.2:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: DATA+OMITTED
    client-key-data: DATA+OMITTED
controlplane:~$
```

→There is one cluster its name is kubernetes

15- What is the user configured in the current context?

```
controlplane:~$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://172.30.1.2:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate-data: DATA+OMITTED
    client-key-data: DATA+OMITTED
controlplane:~$
```

- Current context name → kubernetes-admin@kubernetes

- User configured in that context → Kubernetes-admin

16- Create a Persistent Volume with the given specification.

Volume Name: pv-log

Storage: 100Mi

Access Modes: ReadWriteMany

Host Path: /pv/log

Firstly, create pv-log.yml:

```
Editor   Tab 1   +
  GNU nano 7.2                                                    pv-log.yml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-log
spec:
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /pv/log
```

Then, apply and check:

```
controlplane:~$ k apply -f pv-log.yml
persistentvolume/pv-log created
controlplane:~$ k get pv
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pv-log   100Mi      RWX            Retain           Available                          <unset>                          5s
controlplane:~$
```

17- Create a Persistent Volume Claim with the given specification.

Volume Name: claim-log-1

Storage Request: 50Mi

Access Modes: ReadWriteMany

Firstly, create pvc-claim.yml:

```
Editor   Tab 1   +
  GNU nano 7.2                                                    pvc-claim.yml *
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: claim-log-1
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi
```

Then, apply and check:

```
controlplane:~$ k apply -f pvc-claim.yml
persistentvolumeclaim/claim-log-1 created
controlplane:~$ k get pvc
NAME          STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   VOLUMEATTRIBUTESCLASS   AGE
claim-log-1   Bound    pv-log   100Mi      RWX                           <unset>                 3s
controlplane:~$
```

18- Create a webapp pod to use the persistent volume claim as its storage.

Name: webapp

Image Name: nginx

Volume: PersistentVolumeClaim=claim-log-1

Volume Mount: /var/log/nginx

Firstly, create webapp-pod.yml:

```
Editor   Tab 1   +
  GNU nano 7.2                                          webapp-pod.yml *
apiVersion: v1
kind: Pod
metadata:
  name: webapp
spec:
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
        - name: log-volume
          mountPath: /var/log/nginx
  volumes:
    - name: log-volume
      persistentVolumeClaim:
        claimName: claim-log-1
```

Then, apply and check:

```
controlplane:~$ k apply -f webapp-pod.yml
pod/webapp created
```

```
controlplane:~$ k get pods
NAME     READY   STATUS    RESTARTS   AGE
webapp   1/1     Running   0          14s
controlplane:~$ kubectl exec -it webapp -- ls /var/log/nginx
access.log  error.log
controlplane:~$
```

→The webapp pod started successfully, and the PVC claim-log-1 is correctly mounted at /var/log/nginx.