1- How manyConfigMaps exist in the environment?

```
menna@docker:~$ kubectl get configmaps --all-namespaces
NAMESPACE              NAME                                                 DATA   AGE
default                kube-root-ca.crt                                     1      11d
kube-node-lease        kube-root-ca.crt                                     1      11d
kube-public            cluster-info                                         2      11d
kube-public            kube-root-ca.crt                                     1      11d
kube-system            coredns                                              1      11d
kube-system            extension-apiserver-authentication                   6      11d
kube-system            kube-apiserver-legacy-service-account-token-tracking 1      11d
kube-system            kube-proxy                                           2      11d
kube-system            kube-root-ca.crt                                     1      11d
kube-system            kubeadm-config                                       1      11d
kube-system            kubelet-config                                       1      11d
kubernetes-dashboard   kube-root-ca.crt                                     1      11d
kubernetes-dashboard   kubernetes-dashboard-settings                        0      11d
```

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

ConfigName Name: webapp-config-map

Data: APP_COLOR=darkblue

```
menna@docker:~$ vim webapp-config-map.yaml
```

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: webapp-config-map
data:
  APP_COLOR: darkblue
~
```

```
menna@docker:~$ kubectl apply -f webapp-config-map.yaml
configmap/webapp-config-map created
```

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

```
menna@docker:~$ vim webapp-color-pod.yaml
```

```
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
```

```
menna@docker:~$ kubectl apply -f webapp-color-pod.yaml
pod/webapp-color created
```

4- How many Secrets exist on the system?

```
menna@docker:~$ kubectl get secrets --all-namespaces
NAMESPACE              NAME                                TYPE                                  DATA   AGE
kube-system            bootstrap-token-6u7co9              bootstrap.kubernetes.io/token         6      11d
kubernetes-dashboard   kubernetes-dashboard-certs          Opaque                                0      11d
kubernetes-dashboard   kubernetes-dashboard-csrf           Opaque                                1      11d
kubernetes-dashboard   kubernetes-dashboard-key-holder     Opaque                                2      11d
```

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

```
menna@docker:~$ kubectl describe secret
No resources found in default namespace.
```

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

```
menna@docker:~$ vim db-pod.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  name: db-pod
spec:
  containers:
  - name: mysql-container
    image: mysql:5.7
```

```
menna@docker:~$ kubectl apply -f db-pod.yaml
pod/db-pod created
```

```
menna@docker:~$ kubectl get pod db-pod
NAME     READY   STATUS             RESTARTS      AGE
db-pod   0/1     CrashLoopBackOff   1 (7s ago)   80s
```

7- why the db-pod status not ready

Nothing to create or access MYSQL database

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

```
menna@docker:~$ vim db-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
stringData:
  MYSQL_DATABASE: sql01
  MYSQL_USER: user1
  MYSQL_PASSWORD: password
  MYSQL_ROOT_PASSWORD: password123
```

```
menna@docker:~$ kubectl apply -f db-secret.yaml
secret/db-secret created
```

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

Delete and recreate the pod if required.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-pod
spec:
  containers:
  - name: mysql-container
    image: mysql:5.7
    env:
    - name: MYSQL_DATABASE
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: MYSQL_DATABASE
    - name: MYSQL_USER
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: MYSQL_USER
    - name: MYSQL_PASSWORD
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: MYSQL_PASSWORD
    - name: MYSQL_ROOT_PASSWORD
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: MYSQL_ROOT_PASSWORD
```

```
menna@docker:~$ vim db-pod.yaml
menna@docker:~$ kubectl delete pod db-pod
pod "db-pod" deleted
menna@docker:~$ kubectl apply -f db-pod.yaml
pod/db-pod created
```

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

```
apiVersion: v1
kind: Pod
metadata:
  name: yellow
spec:
  containers:
  - name: lemon
    image: busybox
    command: ["sleep", "3600"]
  - name: gold
    image: redis
```

```
menna@docker:~$ vim yellow-pod.yaml
menna@docker:~$ kubectl apply -f yellow-pod.yaml
pod/yellow created
```

11- Create a pod red with redis image and use an initContainer that

uses the busybox image and sleeps for 20 seconds

```
apiVersion: v1
kind: Pod
metadata:
  name: red
spec:
  containers:
  - name: redis-container
    image: redis
  initContainers:
  - name: init-busybox
    image: busybox
    command: ["sleep", "20"]
```

```
menna@docker:~$ vim red-pod.yaml
menna@docker:~$ kubectl apply -f red-pod.yaml
pod/red created
```

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.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: print-envars-greeting
spec:
  containers:
  - name: print-env-container
    image: bash
    env:
    - name: GREETING
      value: "Welcome to"
    - name: COMPANY
      value: "DevOps"
    - name: GROUP
      value: "Industries"
    command: ["/bin/sh", "-c", "echo $(GREETING) $(COMPANY) $(GROUP)"]
```

```
menna@docker:~$ vim print-envars-greeting.yaml
menna@docker:~$ kubectl apply -f print-envars-greeting.yaml
pod/print-envars-greeting created
```

```
menna@docker:~$ kubectl logs -f print-envars-greeting
Welcome to DevOps Industries
```

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

```
menna@docker:~$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /home/menna/.minikube/ca.crt
    extensions:
    - extension:
        last-update: Mon, 24 Feb 2025 18:22:46 EET
        provider: minikube.sigs.k8s.io
        version: v1.35.0
      name: cluster_info
    server: https://192.168.49.2:8443
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Mon, 24 Feb 2025 18:22:46 EET
        provider: minikube.sigs.k8s.io
        version: v1.35.0
      name: context_info
    namespace: default
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate: /home/menna/.minikube/profiles/minikube/client.crt
    client-key: /home/menna/.minikube/profiles/minikube/client.key
```

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

```
menna@docker:~$ kubectl config view -o jsonpath='{.clusters[*].name}' | wc -w
1
```

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

```
menna@docker:~$ kubectl config view | grep -A 3 "$(kubectl config current-context)" | grep "user:" | awk '{print $2}'
minikube
```

16- Create a Persistent Volume with the given specification.

Volume Name: pv-log

Storage: 100Mi

Access Modes: ReadWriteMany

Host Path: /pv/log

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-log
spec:
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /pv/log
~
```

v

```
menna@docker:~$ vim pv-log.yaml
menna@docker:~$ kubectl apply -f pv-log.yaml
persistentvolume/pv-log created
```

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

Volume Name: claim-log-1

Storage Request: 50Mi

Access Modes: ReadWriteMany

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: claim-log-1
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi
~
```

```
menna@docker:~$ vim claim-log-1.yaml
menna@docker:~$ kubectl apply -f claim-log-1.yaml
persistentvolumeclaim/claim-log-1 created
```

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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp
spec:
  containers:
  - name: nginx-container
    image: nginx
    volumeMounts:
    - mountPath: /var/log/nginx
      name: log-volume
  volumes:
  - name: log-volume
    persistentVolumeClaim:
      claimName: claim-log-1
```

```
menna@docker:~$ vim webapp-pod.yaml
menna@docker:~$ kubectl apply -f webapp-pod.yaml
pod/webapp created
```