1- How many `ConfigMaps` exist in the environment?

```
[root@yara ~]# kubectl get configmaps --all-namespaces
NAMESPACE         NAME                                                 DATA   AGE
default           kube-root-ca.crt                                     1      5d10h
kube-node-lease   kube-root-ca.crt                                     1      5d10h
kube-public       cluster-info                                         1      5d10h
kube-public       kube-root-ca.crt                                     1      5d10h
kube-system       coredns                                              1      5d10h
kube-system       extension-apiserver-authentication                   6      5d10h
kube-system       kube-apiserver-legacy-service-account-token-tracking 1      5d10h
kube-system       kube-proxy                                           2      5d10h
kube-system       kube-root-ca.crt                                     1      5d10h
kube-system       kubeadm-config                                       1      5d10h
kube-system       kubelet-config                                       1      5d10h
[root@yara ~]# kubectl get configmaps --all-namespaces |wc -l
12
```

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

ConfigName Name: webapp-config-map

Data: APP_COLOR=darkblue

```
[root@yara ~]# vim webapp-config-map.yaml
[root@yara ~]# 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

```
[root@yara ~]# vim webapp-color-pod.yaml
[root@yara ~]# kubectl apply -f webapp-color-pod.yaml
pod/webapp-color created
[root@yara ~]# kubectl get pod webapp-color
NAME           READY   STATUS    RESTARTS   AGE
webapp-color   1/1     Running   0          39s
```

4- How many `Secrets` exist on the system?

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

```
[root@yara ~]# kubectl get secrets --all-namespaces
No resources found
```

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

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

```
[root@yara ~]# vim db-pod.yaml
[root@yara ~]# kubectl apply -f db-pod.yaml
pod/db-pod created
[root@yara ~]# kubectl get pod db-pod
NAME     READY   STATUS    RESTARTS   AGE
db-pod   0/1     Error     0          5s
```

7- why the db-pod status not ready

```
[root@yara ~]# kubectl logs db-pod
2025-03-13 15:39:16+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.44-1.el7 started.
2025-03-13 15:39:16+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2025-03-13 15:39:16+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.44-1.el7 started.
2025-03-13 15:39:17+00:00 [ERROR] [Entrypoint]: Database is uninitialized and password option is not specified
    You need to specify one of the following as an environment variable:
    - MYSQL_ROOT_PASSWORD
    - MYSQL_ALLOW_EMPTY_PASSWORD
    - MYSQL_RANDOM_ROOT_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

```
[root@yara ~]# echo -n 'sql01' | base64
c3FsMDE=
[root@yara ~]# echo -n 'user1' | base64
dXNlcjE=
[root@yara ~]# echo -n 'password' | base64
cGFzc3dvcmQ=
[root@yara ~]# echo -n 'password123' | base64
cGFzc3dvcmQxMjM=
[root@yara ~]# vim db-secret.yaml
[root@yara ~]# 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.

```
apiVersion: v1
kind: Pod
metadata:
  name: db-pod
spec:
  containers:
    - name: mysql
      image: mysql:5.7
      envFrom:
        - secretRef:
            name: db-secret
~
~
```

```
[root@yara ~]# vim db-pod.yaml
[root@yara ~]# kubectl delete pod db-pod
pod "db-pod" deleted
[root@yara ~]# kubectl apply -f db-pod.yaml
pod/db-pod created
[root@yara ~]# kubectl get pod db-pod
NAME     READY   STATUS    RESTARTS   AGE
db-pod   1/1     Running   0          7s
```

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

```
[root@yara ~]# vim yellow.yaml
[root@yara ~]# kubectl apply -f yellow.yaml
pod/yellow created
[root@yara ~]# kubectl get pod yellow
NAME     READY   STATUS              RESTARTS   AGE
yellow   0/2     ContainerCreating   0          15s
[root@yara ~]# kubectl get pod yellow
NAME     READY   STATUS              RESTARTS   AGE
yellow   0/2     ContainerCreating   0          27s
[root@yara ~]# kubectl get pod yellow
NAME     READY   STATUS    RESTARTS   AGE
yellow   2/2     Running   0          61s
```

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

```
[root@yara ~]# vim pod-with-initcont.yaml
[root@yara ~]# kubectl apply -f pod-with-initcont.yaml
pod/redis-pod created
[root@yara ~]# kubectl get pod redis-pod
NAME        READY   STATUS            RESTARTS   AGE
redis-pod   0/1     PodInitializing   0          77s
[root@yara ~]# kubectl get pod redis-pod
NAME        READY   STATUS    RESTARTS   AGE
redis-pod   1/1     Running   0          119s
```

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.

```
[root@yara ~]# vim print-envars-greeting.yaml
[root@yara ~]# kubectl apply -f print-envars-greeting.yaml
pod/print-envars-greeting created
[root@yara ~]# kubectl get pod print-envars-greeting
NAME                     READY   STATUS              RESTARTS   AGE
print-envars-greeting    0/1     ContainerCreating   0          8s
[root@yara ~]# kubectl logs print-envars-greeting
[Welcome to DevOps Industries]
```

16- Create a `Persistent Volume` with the given specification.

Volume Name: pv-log
Storage: 100Mi
Access Modes: ReadWriteMany
Host Path: /pv/log

```
[root@yara ~]# vim persistent-volume.yaml
[root@yara ~]# kubectl apply -f persistent-volume.yaml
persistentvolume/pv-log created
[root@yara ~]# kubectl get pv
NAME     CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM   STORAGECLASS   VOLUMEATTRIBUTESCLASS   REASON   AGE
pv-log   100Mi      RWX            Retain           Available                          <unset>                          37s
```

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

Volume Name: claim-log-1
Storage Request: 50Mi
Access Modes: ReadWriteMany

```
[root@yara ~]# vim persistent-volume-claim.yaml
[root@yara ~]# kubectl apply -f persistent-volume-claim.yaml
persistentvolumeclaim/claim-log-1 created
[root@yara ~]# kubectl get pv
NAME                                       CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS      CLAIM                 STORAGECLASS   VOLUMEAT
TRIBUTESCLASS   REASON   AGE
pv-log                                     100Mi      RWX            Retain           Available                                        <unset>
                         3m5s
pvc-a3e8263c-dab2-4bba-969f-085ffca988a9   50Mi       RWX            Delete           Bound       default/claim-log-1   standard       <unset>
                         46s
```

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

```
[root@yara ~]# vim webapp-pod.yaml
[root@yara ~]# kubectl apply -f webapp-pod.yaml
pod/webapp created
[root@yara ~]# kubectl get pod webapp
NAME     READY   STATUS    RESTARTS   AGE
webapp   1/1     Running   0          28s
```