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

```
controlplane:~$ kubectl get configmaps --all-namespaces
NAMESPACE            NAME                                                 DATA   AGE
default              kube-root-ca.crt                                     1      36d
kube-node-lease      kube-root-ca.crt                                     1      36d
kube-public          cluster-info                                         2      36d
kube-public          kube-root-ca.crt                                     1      36d
kube-system          canal-config                                         6      36d
kube-system          coredns                                              1      36d
kube-system          extension-apiserver-authentication                   6      36d
kube-system          kube-apiserver-legacy-service-account-token-tracking 1      36d
kube-system          kube-proxy                                           2      36d
kube-system          kube-root-ca.crt                                     1      36d
kube-system          kubeadm-config                                       1      36d
kube-system          kubelet-config                                       1      36d
local-path-storage   kube-root-ca.crt                                     1      36d
local-path-storage   local-path-config                                    4      36d
```

2- Create a new ConfigMap Use the spec given below.
ConfigName Name: webapp-config-map
Data: APP_COLOR=darkblue

```
controlplane:~$ kubectl create configmap webapp-config-map --from-literal=APP_COLOR=darkblue
configmap/webapp-config-map created
```

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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp-color
spec:
  containers:
  - name: nginx
    image: nginx
    envFrom:
    - configMapRef:
        name: webapp-config-map
```

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

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

```
controlplane:~$ kubectl get secrets --all-namespaces
NAMESPACE     NAME                     TYPE                                  DATA   AGE
kube-system   bootstrap-token-fa18uz   bootstrap.kubernetes.io/token         5      36d
```

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

```
controlplane:~$ kubectl describe secret bootstrap-token-fa18uz --namespace=kube-system
Name:         bootstrap-token-fa18uz
Namespace:    kube-system
Labels:       <none>
Annotations:  <none>

Type:  bootstrap.kubernetes.io/token

Data
====
auth-extra-groups:               47 bytes
token-id:                        6 bytes
token-secret:                    16 bytes
usage-bootstrap-authentication:  4 bytes
usage-bootstrap-signing:         4 bytes
```

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

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

```
controlplane:~$ k apply -f pod2.yaml
pod/db-pod created
```

```
controlplane:~$ kubectl get pods
NAME           READY   STATUS    RESTARTS      AGE
db-pod         0/1     Error     3 (31s ago)   59s
webapp-color   1/1     Running   0             7m35s
```

7- why the db-pod status not ready

```
State:          Waiting
  Reason:       CrashLoopBackOff
Last State:     Terminated
  Reason:       Error
  Exit Code:    1
  Started:      Sun, 27 Apr 2025 21:07:57 +0000
  Finished:     Sun, 27 Apr 2025 21:07:57 +0000
```

likely because MySQL needs environment variables MYSQL_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

```
controlplane:~$ kubectl create secret generic db-secret \
--from-literal=MYSQL_DATABASE=sql01 \
--from-literal=MYSQL_USER=user1 \
--from-literal=MYSQL_PASSWORD=password \
--from-literal=MYSQL_ROOT_PASSWORD=password123
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
```

```
controlplane:~$ kubectl delete pod db-pod
pod "db-pod" deleted
controlplane:~$ kubectl apply -f pod2.yaml
pod/db-pod created
controlplane:~$ k get pods --all-namespaces
NAMESPACE           NAME                                      READY   STATUS    RESTARTS      AGE
default             db-pod                                    1/1     Running   0             15s
```

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

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

```
controlplane:~$ vi pod3.yaml
controlplane:~$ k apply -f pod3.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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: red
spec:
  initContainers:
  - name: init-myservice
    image: busybox
    command: ['sh', '-c', 'sleep 20']
  containers:
  - name: redis
    image: redis
```

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: [ "sh", "-c", 'echo "$(GREETING) $(COMPANY) $(GROUP)" && sleep 3600' ]
```

```
controlplane:~$ k apply -f pod5.yaml
pod/print-envars-greeting created
controlplane:~$ kubectl logs -f print-envars-greeting
Welcome to DevOps Industries
```

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

~/.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
```

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

```
controlplane:~$ kubectl config view --minify -o jsonpath='{.contexts[0].context.user}'
kubernetes-admincontrolplane:~$
```

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

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

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

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

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

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

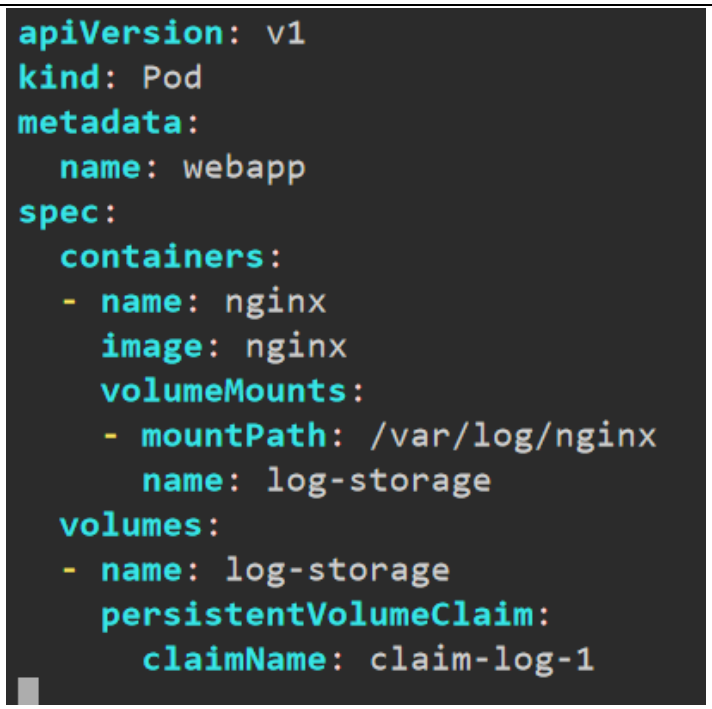
18- Create a `webapp` pod to use the persistent volume claim as its storage.

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
    image: nginx
    volumeMounts:
    - mountPath: /var/log/nginx
      name: log-storage
  volumes:
  - name: log-storage
    persistentVolumeClaim:
      claimName: claim-log-1
```