LAB 4

1- 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: redis
  labels:
    app: redis
spec:
  containers:
  - name: redis
    image: redis
  initContainers:
  - name: init-myservice
    image: busybox:1.28
    command: ['sh', '-c', "sleep 20"]
```

```
controlplane $ k apply -f  redis-pod.yaml
pod/redis created
controlplane $ k get po
NAME    READY   STATUS    RESTARTS   AGE
redis   1/1     Running   0          48s
```

2- 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”
3. Use command to echo ["$(GREETING) $(COMPANY) $(GROUP)"] message.
4. You can check the output using <kubctl logs -f [pod-name]>command

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: print-envars-greeting
  labels:
    app: 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" ']
```

```
controlplane $ vim greet.yaml
controlplane $ k apply -f  greet.yaml
pod/print-envars-greeting created
controlplane $ k logs -f print-envars-greeting
welcome to DevOps Industries
```

3- 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
  labels:
    app: pv-log
spec:
  accessModes:
    - ReadWriteMany
  capacity:
    storage:  100Mi
  hostPath:
    path: "/pv/log"
  claimRef:
    name: claim-log-1
```

```
controlplane $ k apply -f pv.yaml
persistentvolume/pv-log created
```

4- 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
  namespace: default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: "50Mi"
  selector:
    matchLabels:
      app: pv-log
```

```
controlplane $ k apply -f pvc.yaml
persistentvolumeclaim/claim-log-1 created
```

5- 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
  labels:
    app: nginx
spec:
  containers:
  - name: webapp-pod
    image: nginx
    volumeMounts:
      - name: vol
        mountPath: /var/log/nginx
  volumes:
  - name: vol
    persistentVolumeClaim:
        claimName: claim-log-1
```

```
controlplane $ k apply -f pv-pod.yaml
pod/webapp created
```

6- How many DaemonSets are created in the cluster in all namespaces? *2 Daemonset*

```
controlplane $ k get DaemonSets --all-namespaces
NAMESPACE     NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   canal        2         2         2       2            2           kubernetes.io/os=linux   35d
kube-system   kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   35d
```

7- what DaemonSets exist on the kube-system namespace*?* *Canal and kube-proxy*

```
controlplane $ k get DaemonSets -n kube-system
NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
canal        2         2         2       2            2           kubernetes.io/os=linux   35d
kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   35d
```

8- What is the image used by the POD deployed by the kube-proxy DaemonSet ?

registry.k8s.io/kube-proxy:v1.26.0

```
Containers:
  kube-proxy:
    Container ID:  containerd://dd8c2ed37033d2084faa0068ade8225691c0ae5aed4c36e8d641323a84889c3f
    Image:         registry.k8s.io/kube-proxy:v1.26.0
    Image ID:      registry.k8s.io/kube-proxy@sha256:1e9bbe429e4e2b2ad32681c91deb98a334f1bf4135137df5f84f9d03689060fe
    Port:          <none>
    Host Port:     <none>
```

9- Deploy a DaemonSet for FluentD Logging. Use the given specifications.
Name: elasticsearch -- Namespace: kube-system -- Image: k8s.gcr.io/fluentd-elasticsearch:1.20

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
```

```
controlplane $ k apply -f daemon.yaml
daemonset.apps/elasticsearch created
controlplane $ k get DaemonSet -n kube-system
NAME            DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
canal           2         2         2       2            2           kubernetes.io/os=linux   35d
elasticsearch   2         2         2       2            2           <none>                   35s
kube-proxy      2         2         2       2            2           kubernetes.io/os=linux   35d
```

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
    tty: true
  - name: gold
    image: redis
```

```
controlplane $ k apply -f m-pod.yaml
pod/yellow created
controlplane $ k get po
NAME     READY   STATUS              RESTARTS   AGE
yellow   0/2     ContainerCreating   0          5s
controlplane $ k get po
NAME     READY   STATUS    RESTARTS   AGE
yellow   2/2     Running   0          8s
```

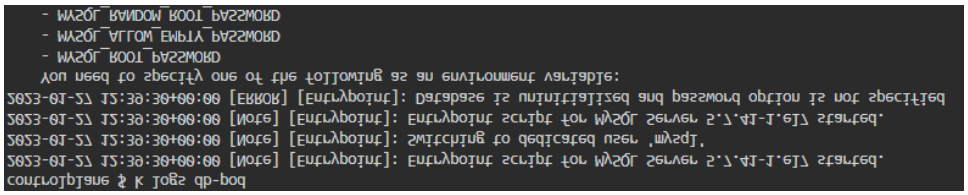
11- create a POD called db-pod with the image mysql:5.7 then check the POD status

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

```
NAME     READY   STATUS    RESTARTS   AGE
db-pod   0/1     Error     0          10s
```

12- why the db-pod status not ready

because we didn't assign database env variables



13- 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

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
data:
  MYSQL_DATABASE: c3FsMDEg
  MYSQL_USER: dXNlcjEg
  MYSQL_PASSWORD: cGFzc3dvcmQgIA==
  MYSQL_ROOT_PASSWORD: cGFzc3dvcmQxMjMgIA==
```

14 - 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: db-container
    image: mysql:5.7
    envFrom:
      - secretRef:
        name: db-secret
```

```
controlplane $ k get po
NAME     READY   STATUS    RESTARTS      AGE
db-pod   1/1     Running   1 (5s ago)   10s
```