Lab4

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 $ vim redis.yaml
controlplane $ k apply -f redis.yaml
pod/redis created
```

```
controlplane $ k get po
NAME    READY   STATUS    RESTARTS   AGE
redis   1/1     Running   0          28s
```

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 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: "welcom to"
    - name: COMPANY
      value: "DevOps"
    - name: GROUP
      value: "Industries"
    command: ['sh','-c','echo "$GREETNG $COMPANY $GROUP "']
```

```
controlplane $ vim greet.yaml
controlplane $ k apply -f greet.yaml
pod/print-envars-greeting created
controlplane $ k logs -f print-envars-greeting
welcom 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 $ vim pv-log.yaml
controlplane $ k apply -f pv-log.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 claim.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/lpg/nginx
  volumes:
  - name: vol
    persistentVolumeClaim:
        claimName: claim-log-1
```

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

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

```
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   4d7h
kube-system   kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   4d7h
```

7- what DaemonSets exist on the kube-system namespace?

```
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   4d7h
kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   4d7h
controlplane $
```

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

```
controlplane $ kubectl describe daemonset kube-proxy -n kube-system | grep Image
    Image:      registry.k8s.io/kube-proxy:v1.26.0
```

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

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluend-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 $ vim daemon.yaml
controlplane $ k apply -f daemon.yaml
daemonset.apps/elasticsearch created
```

```
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   4d7h
elasticsearch   2         2         2       2            2           <none>                   21s
kube-proxy      2         2         2       2            2           kubernetes.io/os=linux   4d7h
```

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: Pod
metadata:
  name: yellow
spec:
  containers:
  - name: lemon
    image: busybox
    tty: true
  - name: gold
    image: redis
```

```
controlplane $ vim multi-container.yaml
controlplane $ k apply -f multi-container.yaml
pod/yellow created
controlplane $ k get po
NAME     READY   STATUS    RESTARTS   AGE
yellow   2/2     Running   0          19s
```

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
```

```
controlplane $ vim db-pod.yaml
controlplane $ k apply -f db-pod.yaml
pod/db-pod created
controlplane $ k get po
NAME     READY   STATUS             RESTARTS      AGE
db-pod   0/1     CrashLoopBackOff   1 (6s ago)    18s
yellow   2/2     Running            0             10m
controlplane $ k get po
NAME     READY   STATUS    RESTARTS      AGE
db-pod   0/1     Error     2 (22s ago)   34s
yellow   2/2     Running   0             10m
controlplane $
```

12- why the db-pod status not ready

we didn't assign database env variables

```
controlplane $ k logs db-pod
2023-01-30 22:00:11+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.41-1.el7 started.
2023-01-30 22:00:11+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2023-01-30 22:00:11+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.41-1.el7 started.
2023-01-30 22:00:11+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
```

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: sql01
  MYSQL_USER: user1
  MYSQL_PASSWORD: password
  MYSQL_ROOT_PASSWORD: password123
```

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
db-pod   1/1     Running
```