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: red
spec:
  containers:
  - image: redis
    name: red
  initContainers:
  - name: init-busybox
    image: busybox:1.28
    command: ["sleep", "20"]
~
~
~
```

```
amr@amrgomaa:~/Documents/kubernetes-sprints/lab4$ kubectl apply -f redpod.yml
pod/red created
amr@amrgomaa:~/Documents/kubernetes-sprints/lab4$
```

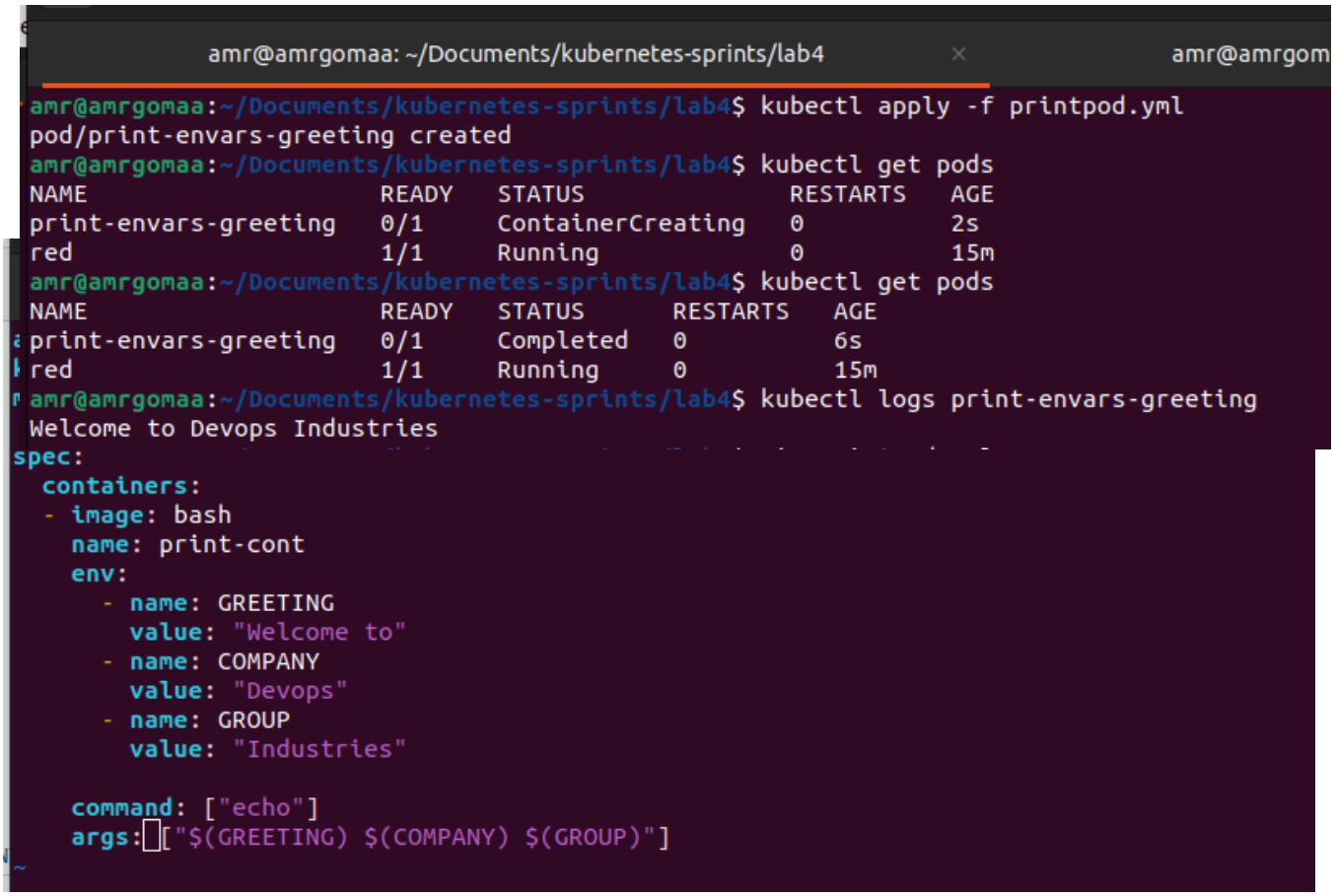
2- Create a pod named print-envars-greeting.

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



3- Create a Persistent Volume with the given specification.
Volume Name: pv-log
Storage: 100Mi
Access Modes: ReadWriteMany
Host Path: /pv/log

```
Editor   Tab 1   +
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-log
spec:
  capacity:
    storage: 100Mi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /pv/log
~
```

4- Create a Persistent Volume Claim with the given specification.
Volume Name: claim-log-1
Storage Request: 50Mi
Access Modes: ReadWriteMany

```
Editor   Tab 1   +

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

```
controlplane $ kubectl get pvc
NAME          STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
claim-log-1   Bound    pv-log   100Mi      RWX                           68s
controlplane $
```

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

```
Editor   Tab 1   +
apiVersion: v1
kind: Pod
metadata:
  name: webapp
spec:
  containers:
    - name: webapp
      image: nginx
      ports:
        - containerPort: 80
      volumeMounts:
       - mountPath: "/var/log/nginx"
         name: podv
  volumes:
    - name: podv
      persistentVolumeClaim:
        claimName: claim-log-1
~
~
```

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

```
amr@amrgomaa:~/Documents/kubernetes-sprints/lab4$ kubectl get ds --all-namespaces --no-headers | wc -l
1
```

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

```
amr@amrgomaa:~/Documents/kubernetes-sprints/lab4$ kubectl get ds --all-namespaces --no-headers
kube-system   kube-proxy   1     1     1     1     1     kubernetes.io/os=linux   22d
```

Or use kubectl get ds -n kubesystem

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

```
amr@amrgomaa:~/Documents/kubernetes-sprints/lab4$ kubectl describe pod kube-proxy-xzqx5 -n kube-system | grep Image
    Image:         registry.k8s.io/kube-proxy:v1.25.3
```

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
spec:
  selector:
    matchLabels:
      name: elasticsearch
  template:
    metadata:
      labels:
        name: elasticsearch
    spec:
      containers:
      - name: elasticsearch
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
~
~
~
~
~
                                                                17,52         All
```

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
  - name: gold
    image: redis
~
~
~
```

########## Bonus Question OR if you couldn't Pull MongoDB image yesterday ;) ########
11- create a POD called db-pod with the image mysql:5.7 then check the
POD status

```
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: db-pod
  name: db-pod
spec:
  containers:
  - image: mysql:5.7
    name: db-pod
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
```

12- why the db-pod status not ready

```
   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

```
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
data:
  MYSQL_DATABASE: c3FsMDEK
  MYSQL_USER: dXNlcjEK
  MYSQL_PASSWORD: cGFzc3dvcmQK
  MYSQL_ROOT_PASSWORD: cGFzc3dvcmQxMjMK
```

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:
  creationTimestamp: null
  labels:
    run: db-pod
  name: db-pod
spec:
  containers:
  - image: mysql:5.7
    name: db-pod
    envFrom:
    - secretRef:
        name: db-secret

    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Always
status: {}
~
~
~
```

```
amr@amrgomaa:~/Documents/kubernetes-sprints/lab4$ kubectl get pods
NAME                    READY   STATUS             RESTARTS       AGE
db-pod                  1/1     Running            0              4s
```