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

```
apiVersion: v1
kind: Pod
metadata:
  name: red
spec:
  containers:
  - name: redis
    image: redis:5.0.4
  initContainers:
  - name: init-myservice
    image: busybox:1.28
    command: ['sh','-c',"sleep 20"]
```

```
controlplane $ kubectl get po
NAME   READY   STATUS    RESTARTS   AGE
red    1/1     Running   0          4m39s
```

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.

```
apiVersion: v1
kind: Pod
metadata:
  name: print-envars-greeting
spec:
  containers:
  - name: bash
    image: bash:4.4
    env:
    - name: GREETING
      value: "Welcome to"
    - name: COMPANY
      value: "DevOps"
    - name: GROUP
      value:  "Industries"
    command: ['sh','-c',"echo $GREETING$COMPANY$GROUP"]
```

4. You can check the output using command

```
controlplane $ k logs -f print-envars-greeting
Welcome toDevOpsIndustries
```

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

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-log
spec:
  capacity:
    storage: 100Mi
  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

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

5- Create a webapp pod to use the persistent volume claim as its storage. Name: webapp Image Name: nginx

```
apiVersion: v1
kind: Pod
metadata:
  name: webapp
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
    volumeMounts:
      - mountPath: /var/log/nginx
        name: data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: claim-log-1
~
```

```
controlplane $ k create -f fifth
pod/webapp created
```

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

2 Daemons

```
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   15d
kube-system   kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   15d
```

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

```
NAMESPACE     NAME         DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR            AGE
kube-system   canal        2         2         2       2            2           kubernetes.io/os=linux   15d
kube-system   kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   15d
controlplane $
```

Canal & kube-proxy

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

```
 Containers:
  kube-proxy:
   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
spec:
  selector:
    matchLabels:
      tier: elasticsearch
  template:
    metadata:
      labels:
        tier: elasticsearch
    spec:
      containers:
      - name: elasticsearch
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
```

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:1.28
  - name: gold
    image: redis:5.0.4
~
```

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:
  name:  db-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.7
```

```
NAME     READY   STATUS             RESTARTS        AGE
db-pod   0/1     Error              5 (102s ago)    3m24s
```

12- why the db-pod status not ready

Because we didn't assign any of the environmental variables.

13- Create a new secret named db-secret with the data given below.

```
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
data:
   MYSQL_DATABASE: c3FsMDE=
   MYSQL_USER: dXNlcjE=
   MYSQL_PASSWORD: cGFzc3dvcmQ=
   MYSQL_ROOT_PASSWORD: cGFzc3dvcmQxMjM=
```

14- Configure db-pod to load environment variables from the newly created secret.

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