

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

```
! pod.yaml x
Lab4 > ! pod.yaml > {} spec > [ ] initContainers > {} 0 > image
all.json
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: red
5  spec:
6    containers:
7      - name: redis-container
8        image: redis
9    initContainers:
10     - name: init-myservice
11       image: busybox:1.28
12       command: ['sh', '-c', 'sleep 20']

hawila@hawila-pc: ~/test/Lab4
hawila@hawila-pc:~/test/Lab4$ kubectl apply -f pod.yaml
pod/red created
hawila@hawila-pc:~/test/Lab4$ kubectl describe pod red
Name: red
Namespace: default
Priority: 0
Service Account: default
Node: minikube/192.168.49.2
Start Time: Thu, 02 Feb 2023 05:19:10 +0200
Labels: <none>
Annotations: <none>
Status: Running
IP: 10.244.0.19

! pod.yaml x
Lab4 > ! pod.yaml > {} spec > [ ] initContainers > {} 0 > image
3  metadata:
4    name: red
5  spec:
6    containers:
7      - name: redis-container
8        image: redis
9    initContainers:
10     - name: init-myservice
11       image: busybox:1.28
12       command: ['sh', '-c', 'sleep 20']

hawila@hawila-pc: ~/test/Lab4
Init Containers:
init-myservice:
  Container ID: docker://2c14023c44de8faa454e0a354676d0dd934cb5871ce0b781907d5f33564981f0
  Image: busybox:1.28
  Image ID: docker-pullable://busybox@sha256:141c253bc4c3fd0a201d32dc1f493bcf3fff003b6df416dea4f41046e0f37d47
  Port: <none>
  Host Port: <none>
  Command:
  sh
  -c
  sleep 20
State: Terminated
Reason: Completed
Exit Code: 0
Started: Thu, 02 Feb 2023 05:19:14 +0200
```

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

```
Lab4 > ! pod.yaml > {} spec > [ ] containers > {} 0 > [ ] args
all.json
1
2  apiVersion: v1
3  kind: Pod
4  metadata:
5    name: print-envvars-greeting
6  spec:
7    containers:
8      - image: bash
9        name: print-env-container
10       env:
11         - name: GREETING
12           value: "Welcome to"
13         - name: COMPANY
14           value: "DevOps"
15         - name: GROUP
16           value: "Industries"
17       command: ["echo"]
18       args: ["$(GREETING) $(COMPANY) $(GROUP)"]
```

3- Create a Persistent Volume with the given specification.

```
... ! pod.yaml ! pv-claim.yaml ! prs-vol.yaml x
Lab4 > ! prs-vol.yaml > {} spec > {} hostPath > path
alljson
1  apiVersion: v1
2  kind: PersistentVolume
3  metadata:
4    name: pv-log
5  spec:
6    storageClassName: manual
7    capacity:
8      storage: 100Mi
9    accessModes:
10     - ReadWriteMany
11    hostPath:
12      path: "/pv/log"
13

hawila@hawila-pc: ~/test/Lab4
hawila@hawila-pc:~/test/Lab4$ kubectl apply -f prs-vol.yaml
persistentvolume/pv-log created
hawila@hawila-pc:~/test/Lab4$
```

4- Create a Persistent Volume Claim with the given specification.

```
... ! pod.yaml ! pv-claim.yaml x ! prs-vol.yaml
Lab4 > ! pv-claim.yaml > {} metadata > name
alljson
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: claim-log-1
5  spec:
6    storageClassName: manual
7    accessModes:
8     - ReadWriteMany
9    resources:
10     requests:
11       storage: 50Mi
12

hawila@hawila-pc: ~/test/Lab4
hawila@hawila-pc:~/test/Lab4$ kubectl apply -f pv-claim.yaml
persistentvolumeclaim/claim-log-1 created
```

```
hawila@hawila-pc: ~/test/Lab4
hawila@hawila-pc:~/test/Lab4$ kubectl get pvc claim-log-1
NAME      STATUS   VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
claim-log-1 Bound    pv-log   100Mi      RWM           manual         95s
hawila@hawila-pc:~/test/Lab4$ kubectl get pv pv-log
NAME      CAPACITY   ACCESS MODES   RECLAIM POLICY   STATUS   CLAIM                STORAGECLASS   REASON   AGE
pv-log    100Mi      RWM           Retain           Bound    default/claim-log-1  manual                2m45s
hawila@hawila-pc:~/test/Lab4$
```

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

```
! pod.yaml x ! pv-claim.yaml ! prs-vol.yaml
Lab4 > ! pod.yaml > {} spec > [ ] containers > {} 0 > [ ] volumeMounts > {} 0 > name
alljson
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: webapp
5  spec:
6    containers:
7     - name: nginx-container
8       image: nginx
9       volumeMounts:
10        - mountPath: /var/log/nginx
11          name: pv-claim
12    volumes:
13     - name: pv-claim
14       persistentVolumeClaim:
15         claimName: claim-log-1

hawila@hawila-pc: ~/test/Lab4
hawila@hawila-pc:~/test/Lab4$ kubectl apply -f pod.yaml
pod/webapp created
hawila@hawila-pc:~/test/Lab4$ kubectl get po webapp -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP          NODE   NOMINATED NODE   READINESS GATES
webapp    1/1     Running   0          20s   10.244.0.20 minikube <none>          <none>
hawila@hawila-pc:~/test/Lab4$
```

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

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

```
hawila@hawila-pc: ~/test/Lab4
hawila@hawila-pc:~/test/Lab4$ kubectl get ds
No resources found in default namespace.
hawila@hawila-pc:~/test/Lab4$ kubectl get ds -A
NAMESPACE   NAME           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kube-system  kube-proxy     1         1         1       1            1           kubernetes.io/os=linux   36h
hawila@hawila-pc:~/test/Lab4$ kubectl get ds --namespace kube-proxy
No resources found in kube-proxy namespace.
```

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

DaemonSet

9- Deploy a DaemonSet for FluentD Logging.

```
Lab4 > ! ds.yaml > ...
1  apiVersion: apps/v1
2  kind: DaemonSet
3  metadata:
4    name: fluentd-elasticsearch
5    namespace: kube-system
6    labels:
7      k8s-app: fluentd-logging
8  spec:
9    selector:
10     matchLabels:
11       name: fluentd-elasticsearch
12    template:
13     metadata:
14       labels:
15         name: fluentd-elasticsearch
16     spec:
17       containers:
18         - name: fluentd-elasticsearch
19           image: k8s.gcr.io/fluentd-elasticsearch:1.20
20       resources:
21         limits:

hawila@hawila-pc: ~/test/Lab4
hawila@hawila-pc:~/test/Lab4$ kubectl apply -f ds.yaml
daemonset.apps/fluentd-elasticsearch created
hawila@hawila-pc:~/test/Lab4$ kubectl get daemonsets -A
NAMESPACE   NAME           DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
kube-system  fluentd-elasticsearch  1         1         0       1            0           <none>           10s
kube-system  kube-proxy     1         1         1       1            1           kubernetes.io/os=linux   36h
hawila@hawila-pc:~/test/Lab4$
```

10- Create a multi-container pod with 2 containers.

```
Lab4 > ! mc.yaml > {} spec > [ ] containers > {} 1 > [ ] volumeMounts > {} 0
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: yellow
5  spec:
6    volumes:
7      - name: shared
8        emptyDir: {}
9    containers:
10     - name: lemon
11       image: busybox
12       volumeMounts:
13         - name: shared
14           mountPath: /test
15     - name: gold
16       image: redis
17       volumeMounts:
18         - name: shared
19           mountPath: /redis/data

hawila@hawila-pc: ~/test/Lab4
hawila@hawila-pc:~/test/Lab4$ kubectl apply -f mc.yaml
pod/yellow created
hawila@hawila-pc:~/test/Lab4$ kubectl get pods
NAME     READY   STATUS    RESTARTS   AGE
yellow   1/2     CrashLoopBackOff   1 (10s ago)   20s
hawila@hawila-pc:~/test/Lab4$
```

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

```
Lab4 > ! pod.yaml > {} spec > [ ] containers > {} 0 > name
all.json
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: db-pod
5  spec:
6    containers:
7      - image: mysql:5.7
8        name: db-pod
9
10
```

```
hawi1a@hawi1a-pc: ~/test/Lab4
hawi1a@hawi1a-pc:~/test/Lab4$ kubectl apply -f pod.yaml
pod/db-pod created
hawi1a@hawi1a-pc:~/test/Lab4$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
db-pod    0/1     ContainerCreating   0           6s
```

12- why the db-pod status not ready

```
Reason:      CrashLoopBackOff
Last State:   Terminated
Reason:       Error
Exit Code:    1
```

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

Secret Name: db-secret

Secret 1: MYSQL_DATABASE=sql01

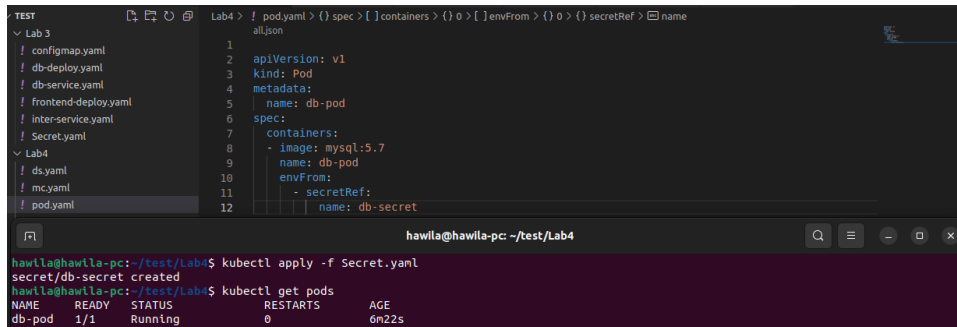
Secret 2: MYSQL_USER=user1

Secret3: MYSQL_PASSWORD=password

Secret 4: MYSQL_ROOT_PASSWORD=password123

```
Lab4 > ! Secret.yaml > {} data > MySQL_ROOT_PASSWORD
all.json
1  apiVersion: v1
2  kind: Secret
3  metadata:
4    name: db-secret
5  data:
6    MYSQL_DATABASE: sql01
7    MYSQL_USER: user1
8    MYSQL_PASSWORD: password
9    MYSQL_ROOT_PASSWORD: password123
```

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



The screenshot shows a terminal window with a file explorer on the left and a code editor in the center. The file explorer shows a directory structure with files like configmap.yaml, db-deploy.yaml, db-service.yaml, frontend-deploy.yaml, inter-service.yaml, Secret.yaml, ds.yaml, mc.yaml, and pod.yaml. The code editor shows the content of pod.yaml, which is a Kubernetes Pod specification. The pod is named 'db-pod' and uses the 'mysql:5.7' image. It has an environment variable 'db-secret' loaded from a secret named 'db-secret'. The terminal window shows the command 'kubectl apply -f Secret.yaml' being executed, which creates the secret 'db-secret'. Then, the command 'kubectl get pods' is executed, showing the 'db-pod' in a 'Running' state.

```
Lab4 > ! pod.yaml > {} spec > [ ] containers > { } 0 > [ ] envFrom > { } 0 > { } secretRef > {} name
all,json
1
2 apiVersion: v1
3 kind: Pod
4 metadata:
5   name: db-pod
6 spec:
7   containers:
8     - image: mysql:5.7
9       name: db-pod
10      envFrom:
11        - secretRef:
12          name: db-secret
```

```
hawi1a@hawi1a-pc: ~/test/Lab4
hawi1a@hawi1a-pc:~/test/Lab4$ kubectl apply -f Secret.yaml
secret/db-secret created
hawi1a@hawi1a-pc:~/test/Lab4$ kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
db-pod  1/1     Running   0           6m22s
```