K8s_LAB04

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

```
Editor   Tab 1   +
apiVersion: v1
kind: Pod
metadata:
  name: red
  labels:
    app: red
spec:
  containers:
  - name: redis
    image: redis
  initContainers:
  - name: busybox
    image: busybox
    command: ['sh','-c',"sleep 20"]
~
~
~
```

```
Editor   Tab 1   +
Initialising Kubernetes... done

controlplane $ vim pod.yaml
controlplane $ kubectl  apply -f pod.yaml
pod/red created
controlplane $ kubectl get pods
NAME   READY   STATUS     RESTARTS   AGE
red    0/1     Init:0/1   0          12s
controlplane $ kubectl get pods
NAME   READY   STATUS    RESTARTS   AGE
red    1/1     Running   0          68s
controlplane $
```

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

Editor Tab 1 +

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

Editor Tab 1 +

```
Initialising Kubernetes... done

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

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
  labels:
    app: pv-log
spec:
  accessModes: ReadWriteMany
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/pv/log"
~
~
~
~
~
~
```

```
controlplane $ vim persistentVolume.yaml
controlplane $ kubectl apply -f persistentVolume.yaml
persistentvolume/pv-log created
controlplane $
```

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: PersistentVolume
metadata:
  name: pv-log
  labels:
    app: pv-log
spec:
  accessModes: ReadWriteMany
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/pv/log"
~
```

```
Editor   Tab 1   +
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: claim-log-1
  namespace: default
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi

~
~
~
```

```
Editor   Tab 1   +
Initialising Kubernetes... done

controlplane $ vim persistentVolume.yaml
controlplane $ vim persistentVolumeClaim.yaml
controlplane $ kubectl apply -f persistentVolume.yaml
persistentvolume/pv-log created
controlplane $ kubectl apply -f persistentVolumeClaim.yaml
persistentvolumeclaim/claim-log-1 created
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
  labels:
    app: nginx
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
        claimName: claim-log-1
  containers:
  - name: nginx
    image: nginx
    volumeMounts:
      - mountPath: "/var/log/nginx"
        name: task-pv-storage
~
~
~
```

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

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

Answer: 2 DaemonSets.

```
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   7d7h
kube-system   kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   7d7h
controlplane $
```

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

```
controlplane $ kubectl 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   7d7h
kube-proxy   2         2         2       2            2           kubernetes.io/os=linux   7d7h
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
controlplane $
```

9- Deploy a DaemonSet for FluentD Logging. Use the given specifications.

Name: elasticsearch

Namespace: kube-system

Image: k8s.gcr.io/fluentd-elasticsearch:1.20

Editor Tab 1 +

```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: elasticsearch
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
~
~
~
```

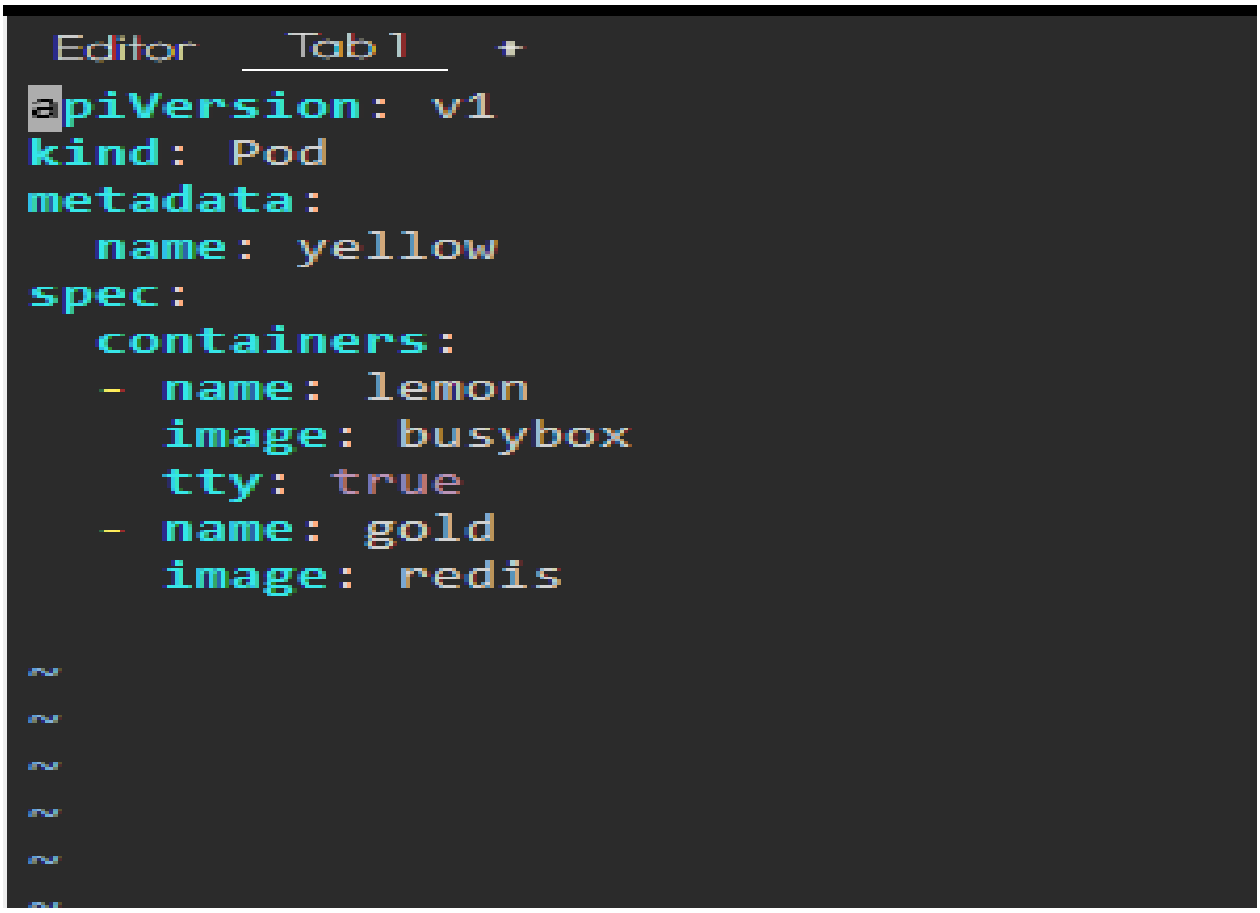
Editor Tab 1 +

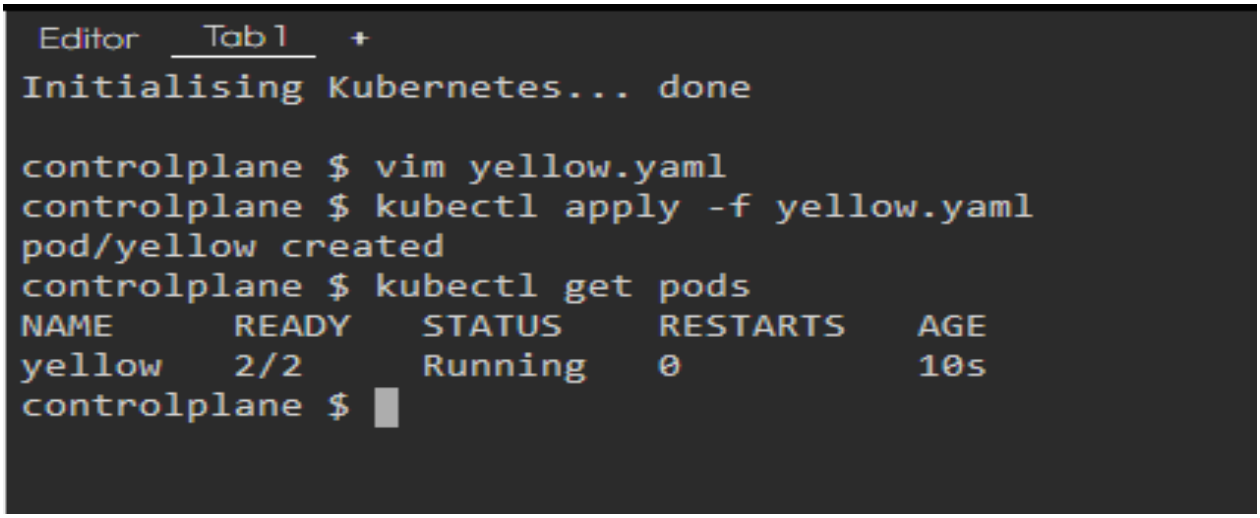
```
Initialising Kubernetes... done

controlplane $ vim daemonset.yaml
controlplane $ kubectl apply -f daemonset.yaml
daemonset.apps/elasticsearch created
controlplane $ kubectl 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   8d
elasticsearch   2         2         2       2            2           <none>                   52s
kube-proxy      2         2         2       2            2           kubernetes.io/os=linux   8d
controlplane $
```

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





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

POD status

Editor Tab 1 +

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

Editor Tab 1 +

```
controlplane $ vim db-pod.yaml
controlplane $ kubectl apply -f db-pod.yaml
pod/db-pod created
controlplane $ kubectl get pods
NAME     READY   STATUS    RESTARTS      AGE
db-pod   0/1     Error     2 (15s ago)   24s
yellow   2/2     Running   0             12m
controlplane $
```

12- why the db-pod status not ready?

Answer: Because we need to specify one of the following as an environment variable

MYSQL_ROOT_PASSWORD, MYSQL_ALLOW_EMPTY_PASSWORD, MYSQL_RANDOM_ROOT_PASSWORD

```
controlplane $ kubectl logs db-pod
2023-02-03 22:10:49+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.41-1.el7 started.
2023-02-03 22:10:49+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2023-02-03 22:10:49+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5.7.41-1.el7 started.
2023-02-03 22:10:49+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
controlplane $
```

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

```
Editor   Tab 1   +
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.

Delete and recreate the pod if required.

```
Editor    Tab 1    +
apiVersion: v1
kind: Pod
metadata:
   name: db-pod
spec:
   containers:
   - name: db-pod
     image: mysql:5.7
     envFrom:
        - secretRef:
            name: db-secret
~
~
~
```

```
Editor    Tab 1    +
Initialising Kubernetes... done

controlplane $ vim secret.yaml
controlplane $ kubectl apply -f secret.yaml
secret/db-secret created
controlplane $ vim pod.yaml
controlplane $ kubectl apply -f pod.yaml
pod/db-pod created
controlplane $ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
db-pod    1/1     Running   0          19s
controlplane $
```