

Kubernetes Lab 3

- 1- How many static pods exist in this cluster in all namespaces?
- 2-On which nodes are the static pods created currently?

Terminal

+

controlplane \$ kubectl get pods --all-namespaces

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
kube-system	coredns-fb8b8dccf-2bmg4	1/1	Running	1	60m
kube-system	coredns-fb8b8dccf-zj469	1/1	Running	1	60m
kube-system	etcd-controlplane	1/1	Running	0	59m
kube-system	katacoda-cloud-provider-664686787f-6kv7w	0/1	CrashLoopBackOff	17	60m
kube-system	kube-apiserver-controlplane	1/1	Running	0	59m
kube-system	kube-controller-manager-controlplane	1/1	Running	0	59m
kube-system	kube-keepalived-vip-z8xkn	1/1	Running	1	59m
kube-system	kube-proxy-2nqv8	1/1	Running	0	59m
kube-system	kube-proxy-8zh9s	1/1	Running	0	60m
kube-system	kube-scheduler-controlplane	1/1	Running	0	59m
kube-system	weave-net-cgrwf	2/2	Running	1	60m
kube-system	weave-net-rxmvw	2/2	Running	1	59m

controlplane \$

- 3- What is the path of the directory holding the static pod definition files?
/etc/Kubernetes/manifests

- 4- Create a static pod named static-busybox that uses the busybox image and the command sleep 1000

controlplane \$ kubectl run static-busybox --image busybox -- [sleep][1000] --restart=Never --dry-run -o yaml > /etc/kub					
ernetes/manifests/static-busybox.yaml					
kubectl run --generator=deployment/apps.v1 is DEPRECATED and will be removed in a future version. Use kubectl run --gen					
erator=run-pod/v1 or kubectl create instead.					
controlplane \$ vi /etc/kubernetes/manifests/static-busybox.yaml					
controlplane \$ kubectl get po					
NAME	READY	STATUS	RESTARTS	AGE	
static-busybox-749694c4b9-fhb47	0/1	CrashLoopBackOff	4	118s	
controlplane \$ kubectl get po					
NAME	READY	STATUS	RESTARTS	AGE	
static-busybox-749694c4b9-fhb47	0/1	CrashLoopBackOff	4	2m19s	
controlplane \$ kubectl get po					
NAME	READY	STATUS	RESTARTS	AGE	
static-busybox-749694c4b9-fhb47	0/1	CrashLoopBackOff	4	3m4s	
controlplane \$ kubectl run static-busybox --image busybox --restart=Never --dry-run -o yaml > /etc/kubernetes/manifests					
/static-busybox.yaml					
controlplane \$ kubectl get po					
	READY	STATUS	RESTARTS	AGE	NAME
static-busybox-749694c4b9-fhb47	0/1	RunContainerError	5	3m33s	
static-busybox-controlplane	0/1	Completed	0	9s	
controlplane \$					

5- Edit the image on the static pod to use busybox:1.28.4

```
Terminal +
labels:
  run: static-busybox
  name: static-busybox
spec:
  containers:
  - image: busybox
    name: static-busybox
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
controlplane $ vi /etc/kubernetes/manifests/static-busybox.yaml
controlplane $ cat /etc/kubernetes/manifests/static-busybox.yaml
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: static-busybox
    name: static-busybox
spec:
  containers:
  - image: busybox:1.28.4
    name: static-busybox
    resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
```

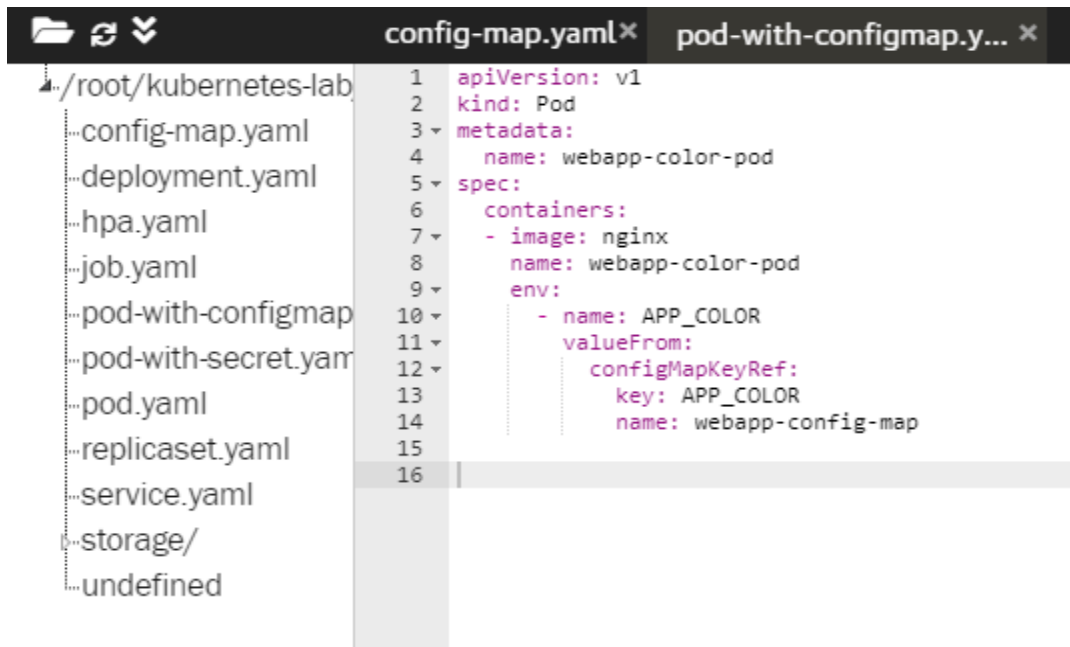
6- How many ConfigMaps exist in the environment?

7- Create a new ConfigMap Use the spec given below. ConfigName Name: webapp-config-map
Data: APP_COLOR=darkblue

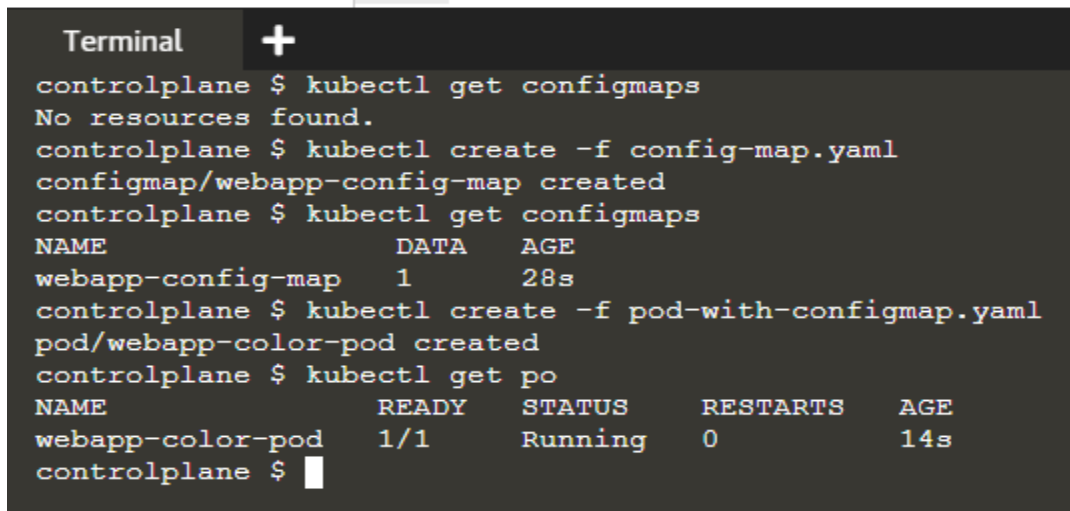
```
config-map.yaml x
1 apiVersion: v1
2 kind: ConfigMap
3 metadata:
4   name: webapp-config-map
5 data:
6   APP_COLOR: darkblue
7
8
9
10
11

Terminal +
controlplane $ kubectl get configmaps
No resources found.
controlplane $ kubectl create -f config-map.yaml
configmap/webapp-config-map created
controlplane $ kubectl get configmaps
NAME          DATA  AGE
webapp-config-map  1      28s
controlplane $
```

8- Create a webapp-color POD with nginx image and use the created ConfigMap



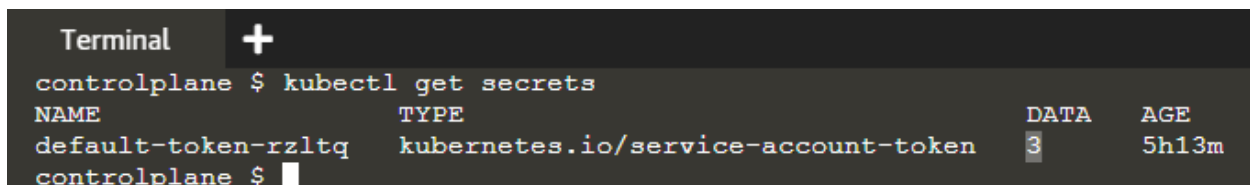
```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: webapp-color-pod
5 spec:
6   containers:
7   - image: nginx
8     name: webapp-color-pod
9     env:
10    - name: APP_COLOR
11      valueFrom:
12        configMapKeyRef:
13          key: APP_COLOR
14          name: webapp-config-map
```



```
controlplane $ kubectl get configmaps
No resources found.
controlplane $ kubectl create -f config-map.yaml
configmap/webapp-config-map created
controlplane $ kubectl get configmaps
NAME          DATA   AGE
webapp-config-map 1       28s
controlplane $ kubectl create -f pod-with-configmap.yaml
pod/webapp-color-pod created
controlplane $ kubectl get po
NAME          READY   STATUS    RESTARTS   AGE
webapp-color-pod 1/1     Running   0          14s
controlplane $
```

9- How many Secrets exist on the system?

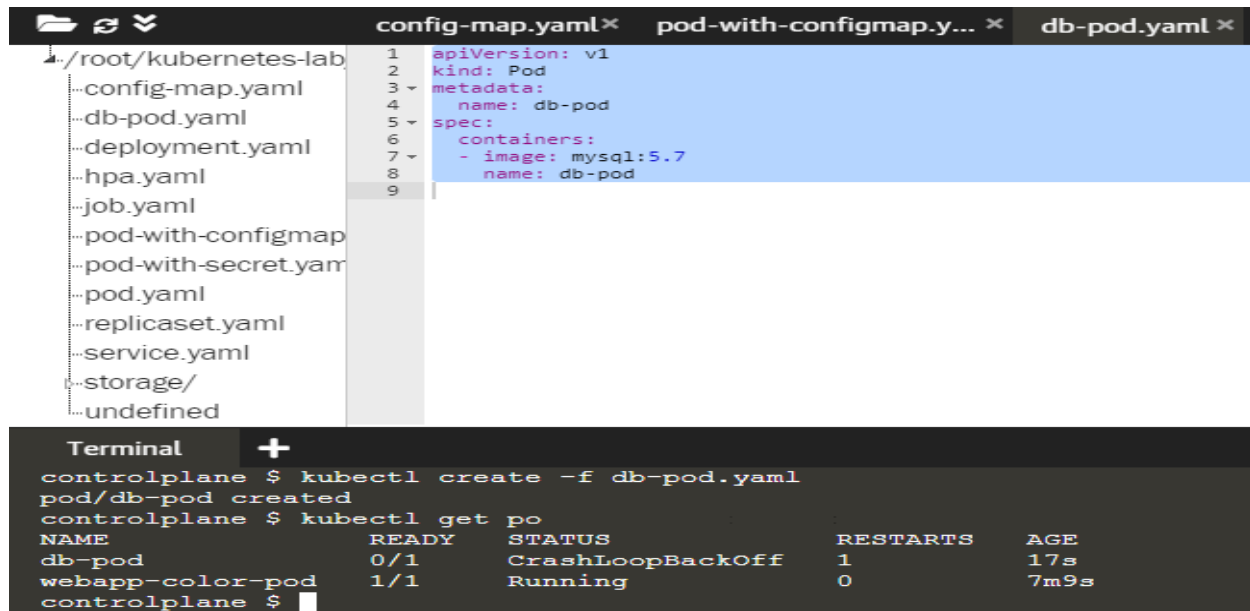
10- How many secrets are defined in the default-token secret?



```
controlplane $ kubectl get secrets
NAME                  TYPE                      DATA   AGE
default-token-rzltq  kubernetes.io/service-account-token  3       5h13m
controlplane $
```

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

12- why the db-pod status not ready



The screenshot shows a code editor with a file explorer on the left listing files like config-map.yaml, db-pod.yaml, deployment.yaml, hpa.yaml, job.yaml, pod-with-configmap.yaml, pod-with-secret.yaml, pod.yaml, replicaset.yaml, service.yaml, storage/, and undefined. The main editor shows the content of db-pod.yaml:

```
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
```

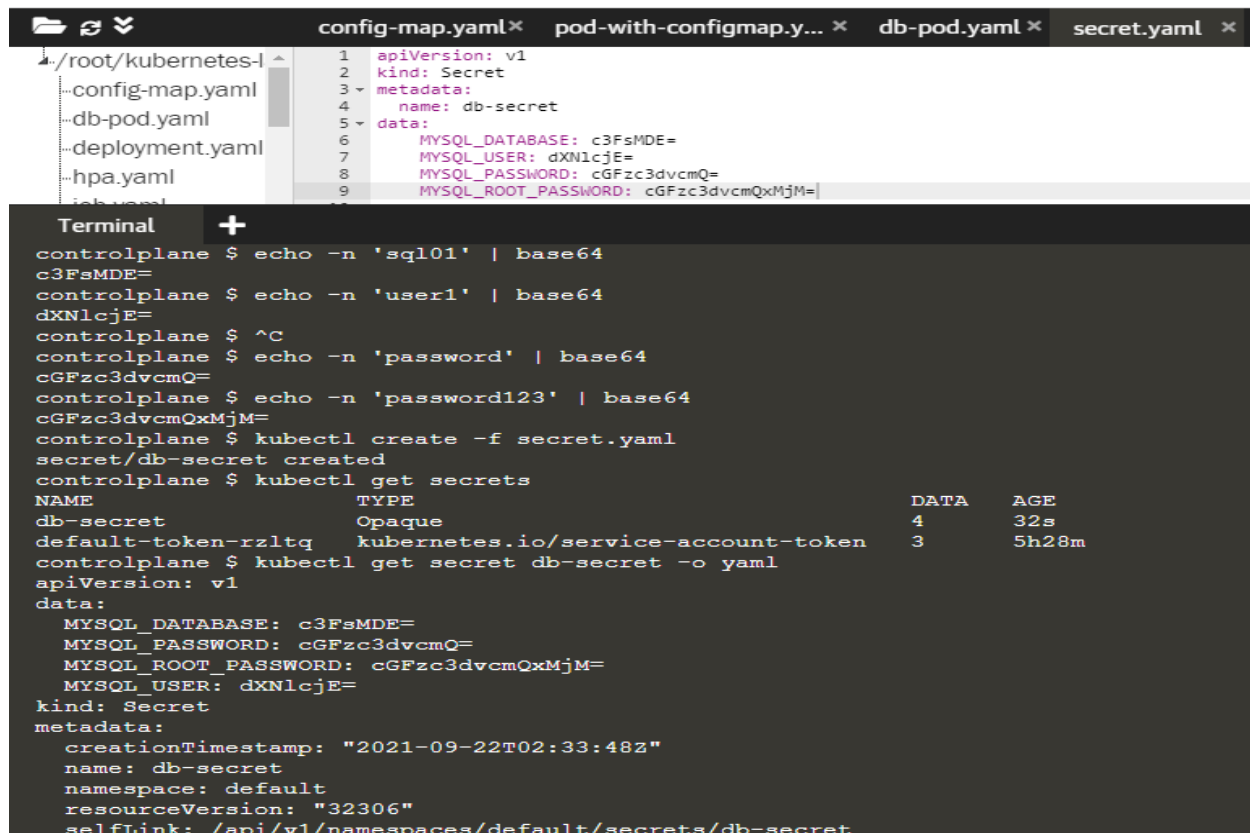
Below the editor is a terminal window with the following commands and output:

```
controlplane $ kubectl create -f db-pod.yaml
pod/db-pod created
controlplane $ kubectl get po
NAME          READY   STATUS             RESTARTS   AGE
db-pod        0/1     CrashLoopBackOff   1          17s
webapp-color-pod 1/1     Running            0          7m9s
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



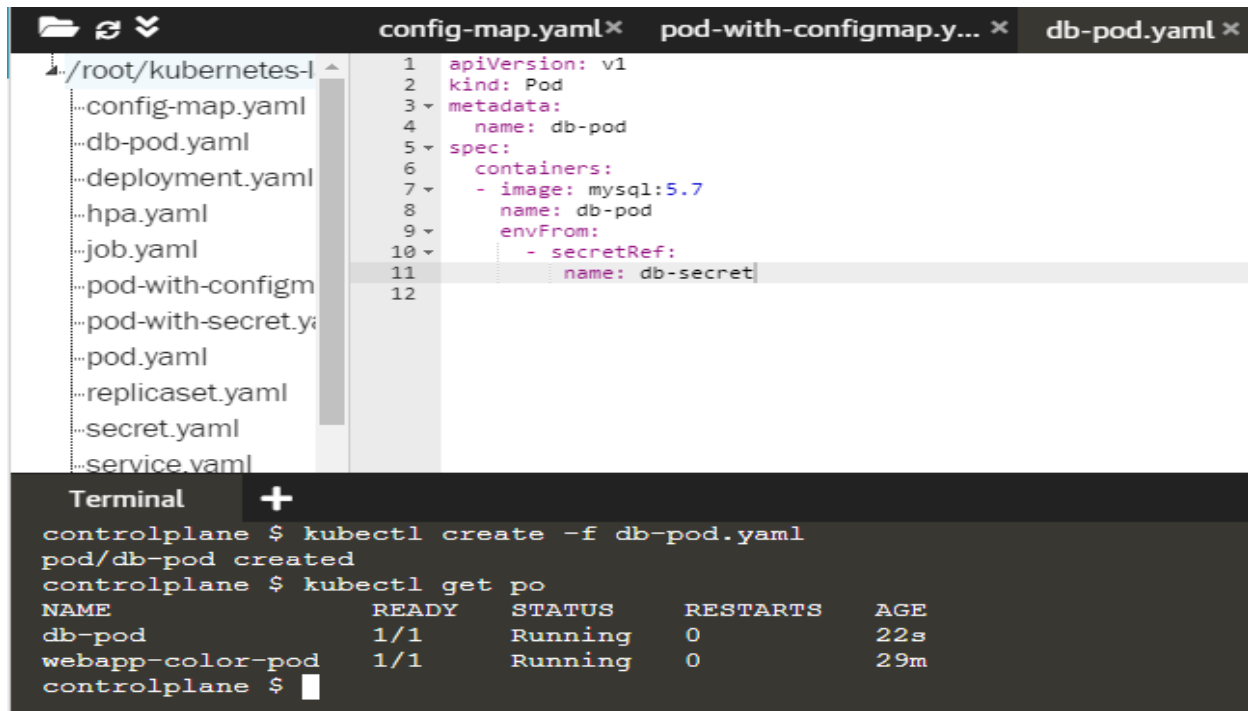
The screenshot shows a code editor with a file explorer on the left listing files like config-map.yaml, db-pod.yaml, deployment.yaml, hpa.yaml, job.yaml, pod-with-configmap.yaml, pod-with-secret.yaml, pod.yaml, replicaset.yaml, service.yaml, storage/, and undefined. The main editor shows the content of secret.yaml:

```
1 apiVersion: v1
2 kind: Secret
3 metadata:
4   name: db-secret
5 data:
6   MYSQL_DATABASE: c3FsMDE=
7   MYSQL_USER: dXNlcjE=
8   MYSQL_PASSWORD: cGFzc3dvcmQ=
9   MYSQL_ROOT_PASSWORD: cGFzc3dvcmQxMjM=
```

Below the editor is a terminal window with the following commands and output:

```
controlplane $ echo -n 'sql01' | base64
c3FsMDE=
controlplane $ echo -n 'user1' | base64
dXNlcjE=
controlplane $ ^C
controlplane $ echo -n 'password' | base64
cGFzc3dvcmQ=
controlplane $ echo -n 'password123' | base64
cGFzc3dvcmQxMjM=
controlplane $ kubectl create -f secret.yaml
secret/db-secret created
controlplane $ kubectl get secrets
NAME          TYPE          DATA   AGE
db-secret      Opaque        4       32s
default-token-rzltq  kubernetes.io/service-account-token  3       5h28m
controlplane $ kubectl get secret db-secret -o yaml
apiVersion: v1
data:
  MYSQL_DATABASE: c3FsMDE=
  MYSQL_PASSWORD: cGFzc3dvcmQ=
  MYSQL_ROOT_PASSWORD: cGFzc3dvcmQxMjM=
  MYSQL_USER: dXNlcjE=
kind: Secret
metadata:
  creationTimestamp: "2021-09-22T02:33:48Z"
  name: db-secret
  namespace: default
  resourceVersion: "32306"
  selfLink: /api/v1/namespaces/default/secrets/db-secret
```

14- Configure db-pod to load environment variables from the newly created secret. Delete and recreate the pod if required.



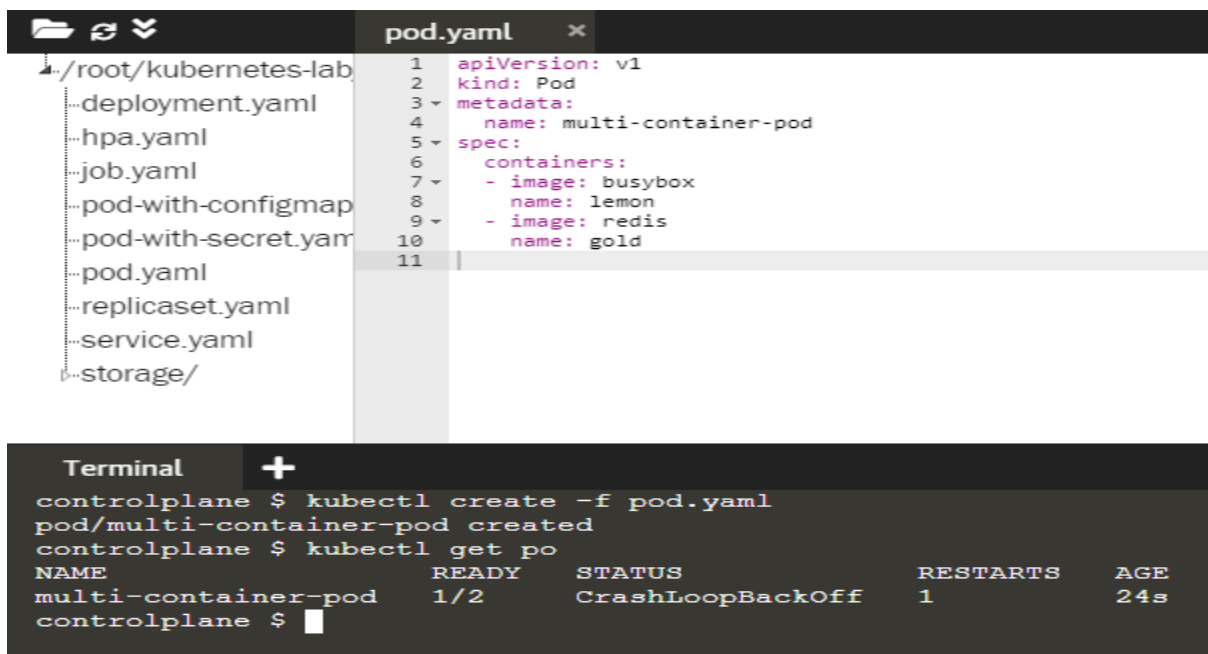
The screenshot shows a code editor with three tabs: `config-map.yaml`, `pod-with-configmap.y...`, and `db-pod.yaml`. The `db-pod.yaml` tab is active, displaying the following YAML configuration:

```
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      envFrom:
10     - secretRef:
11       name: db-secret
```

Below the code editor is a terminal window with the following output:

```
controlplane $ kubectl create -f db-pod.yaml
pod/db-pod created
controlplane $ kubectl get po
NAME                READY   STATUS    RESTARTS   AGE
db-pod              1/1    Running   0           22s
webapp-color-pod    1/1    Running   0           29m
controlplane $
```

15- 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



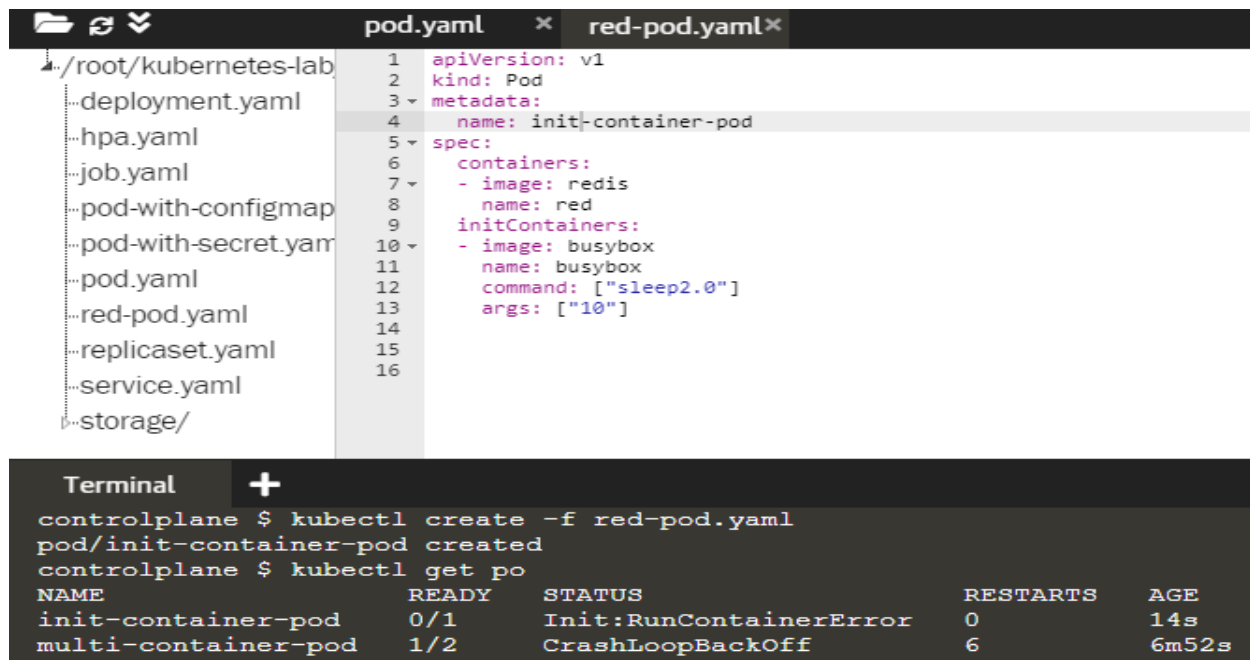
The screenshot shows a code editor with a tab for `pod.yaml`. The `pod.yaml` tab is active, displaying the following YAML configuration:

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: multi-container-pod
5  spec:
6    containers:
7    - image: busybox
8      name: lemon
9    - image: redis
10     name: gold
```

Below the code editor is a terminal window with the following output:

```
controlplane $ kubectl create -f pod.yaml
pod/multi-container-pod created
controlplane $ kubectl get po
NAME                READY   STATUS             RESTARTS   AGE
multi-container-pod  1/2    CrashLoopBackOff   1           24s
controlplane $
```

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

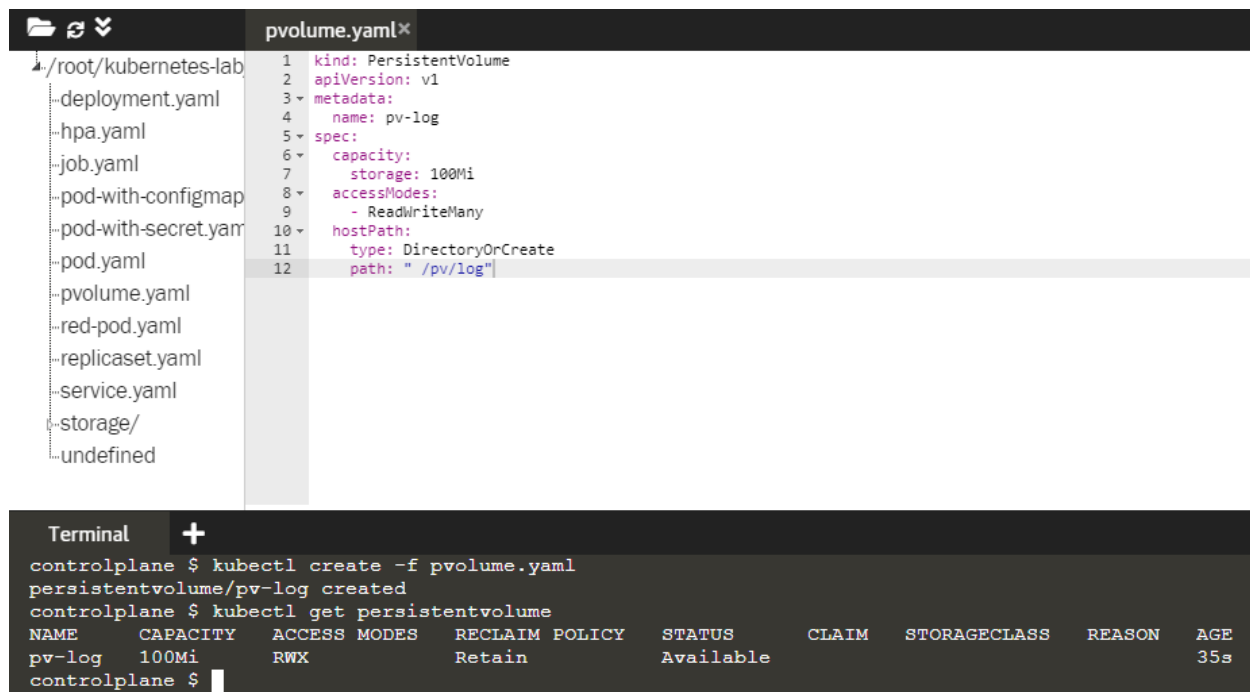


The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like deployment.yaml, hpa.yaml, job.yaml, pod-with-configmap.yaml, pod-with-secret.yaml, pod.yaml, red-pod.yaml, replicaset.yaml, service.yaml, and storage/. The code editor shows the content of red-pod.yaml, which is a Kubernetes Pod manifest. The manifest includes an initContainer named 'init-container-pod' that uses the busybox image and runs the command 'sleep 20'. The main container is named 'red' and uses the redis image. Below the code editor is a terminal window showing the command 'kubectl create -f red-pod.yaml' and the output 'pod/init-container-pod created'. The terminal also shows the command 'kubectl get po' and the output of the 'get po' command, which shows the status of the pod 'init-container-pod' as 'Init:RunContainerError' and the status of the pod 'multi-container-pod' as 'CrashLoopBackOff'.

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: init-container-pod
5 spec:
6   containers:
7   - image: redis
8     name: red
9   initContainers:
10  - image: busybox
11    name: busybox
12    command: ["sleep2.0"]
13    args: ["10"]
```

```
controlplane $ kubectl create -f red-pod.yaml
pod/init-container-pod created
controlplane $ kubectl get po
NAME                                READY   STATUS                    RESTARTS   AGE
init-container-pod                  0/1     Init:RunContainerError    0           14s
multi-container-pod                 1/2     CrashLoopBackOff          6           6m52s
```

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



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like deployment.yaml, hpa.yaml, job.yaml, pod-with-configmap.yaml, pod-with-secret.yaml, pod.yaml, pvolume.yaml, red-pod.yaml, replicaset.yaml, service.yaml, storage/, and undefined. The code editor shows the content of pvolume.yaml, which is a Kubernetes PersistentVolume manifest. The manifest includes a capacity of 100Mi, access modes of ReadWriteMany, and a host path of /pv/log. Below the code editor is a terminal window showing the command 'kubectl create -f pvolume.yaml' and the output 'persistentvolume/pv-log created'. The terminal also shows the command 'kubectl get persistentvolume' and the output of the 'get persistentvolume' command, which shows the status of the volume 'pv-log' as 'Available'.

```
1 kind: PersistentVolume
2 apiVersion: v1
3 metadata:
4   name: pv-log
5 spec:
6   capacity:
7     storage: 100Mi
8   accessModes:
9     - ReadWriteMany
10  hostPath:
11    type: DirectoryOrCreate
12    path: "/pv/log"
```

```
controlplane $ kubectl create -f pvolume.yaml
persistentvolume/pv-log created
controlplane $ kubectl get persistentvolume
NAME      CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS   CLAIM  STORAGECLASS  REASON  AGE
pv-log    100Mi     RWX           Retain          Available             StorageClass  Reason      35s
controlplane $
```

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

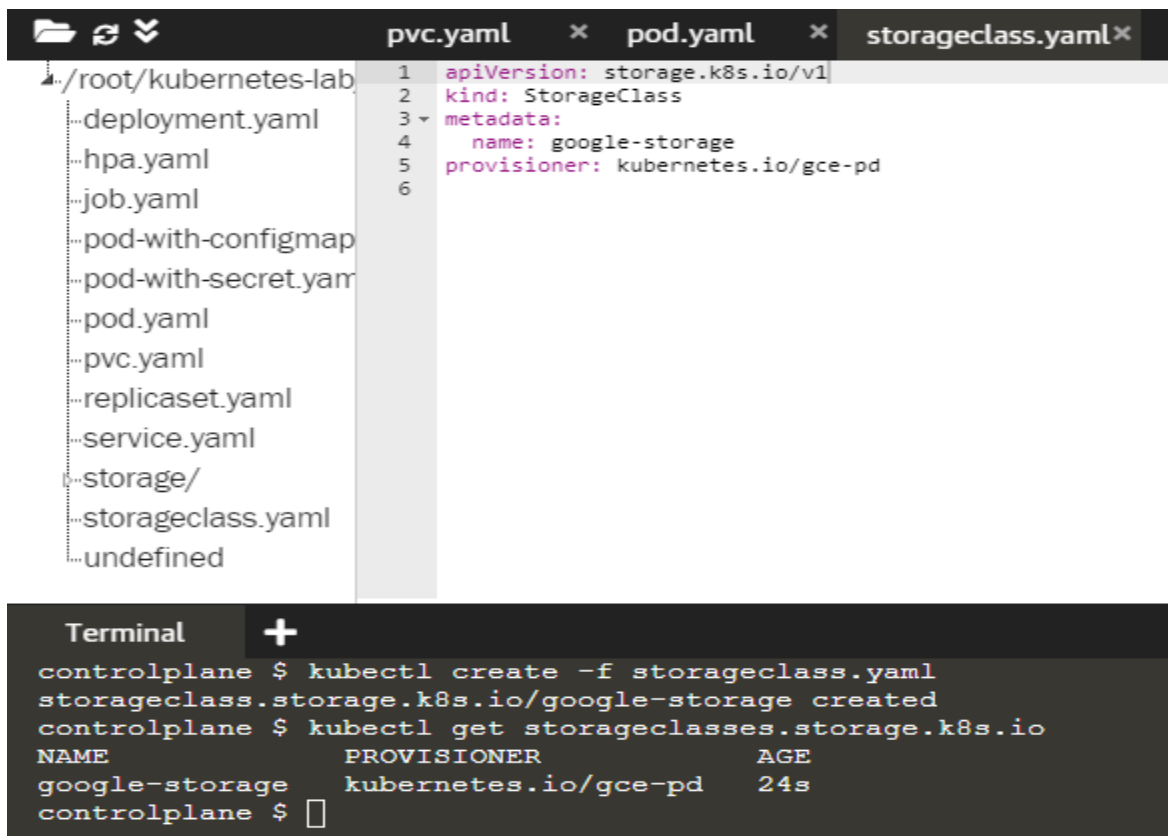


The screenshot shows a code editor with a file explorer on the left listing various Kubernetes manifests. The main editor displays the content of `pvclaim.yaml`, which defines a PersistentVolumeClaim named `claim-log-1` with a storage request of 50Mi and ReadWriteMany access mode. Below the editor is a terminal window showing the successful creation and verification of the claim.

```
1 kind: PersistentVolumeClaim
2 apiVersion: v1
3 metadata:
4   name: claim-log-1
5 spec:
6   accessModes:
7     - ReadWriteMany
8   resources:
9     requests:
10      storage: 50Mi
```

```
controlplane $ kubectl create -f pvclaim.yaml
persistentvolumeclaim/claim-log-1 created
controlplane $ kubectl get persistentvolumeclaim
NAME          STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS   AGE
claim-log-1   Bound    pv-log   100Mi      RWX              google-storage 13s
controlplane $
```

19- 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



The screenshot shows a code editor with a file explorer on the left. The main editor displays the content of `storageclass.yaml`, which defines a StorageClass named `google-storage` using the `kubernetes.io/gce-pd` provisioner. Below the editor is a terminal window showing the successful creation and verification of the storage class.

```
1 apiVersion: storage.k8s.io/v1
2 kind: StorageClass
3 metadata:
4   name: google-storage
5 provisioner: kubernetes.io/gce-pd
6
```

```
controlplane $ kubectl create -f storageclass.yaml
storageclass.storage.k8s.io/google-storage created
controlplane $ kubectl get storageclasses.storage.k8s.io
NAME          PROVISIONER          AGE
google-storage  kubernetes.io/gce-pd  24s
controlplane $
```

