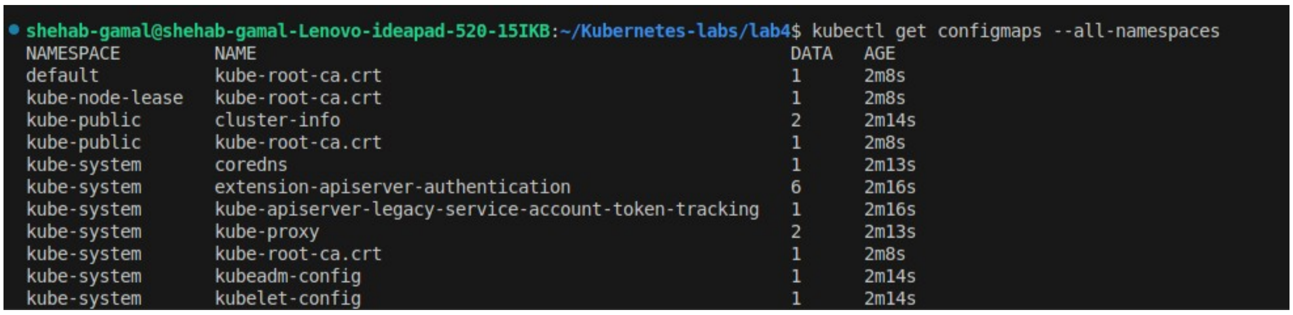
Lab 4 K8s

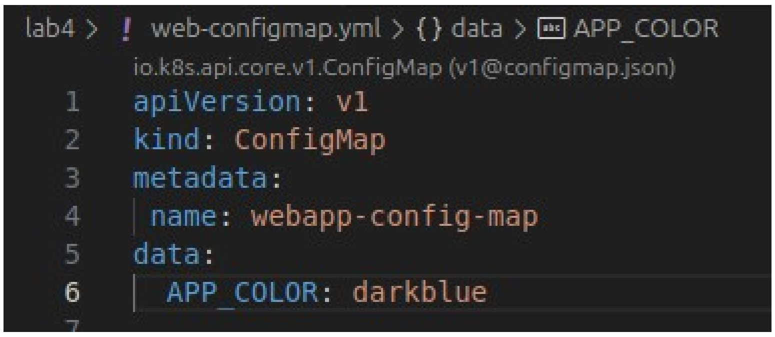
1. How many ConfigMaps exist in the cluster?



```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get configmaps --all-namespaces
NAMESPACE         NAME                                                   DATA   AGE
default           kube-root-ca.crt                                       1      2m8s
kube-node-lease   kube-root-ca.crt                                       1      2m8s
kube-public       cluster-info                                           2      2m14s
kube-public       kube-root-ca.crt                                       1      2m8s
kube-system       coredns                                                1      2m13s
kube-system       extension-apiserver-authentication                     6      2m16s
kube-system       kube-apiserver-legacy-service-account-token-tracking   1      2m16s
kube-system       kube-proxy                                             2      2m13s
kube-system       kube-root-ca.crt                                       1      2m8s
kube-system       kubeadm-config                                         1      2m14s
kube-system       kubelet-config                                         1      2m14s
```

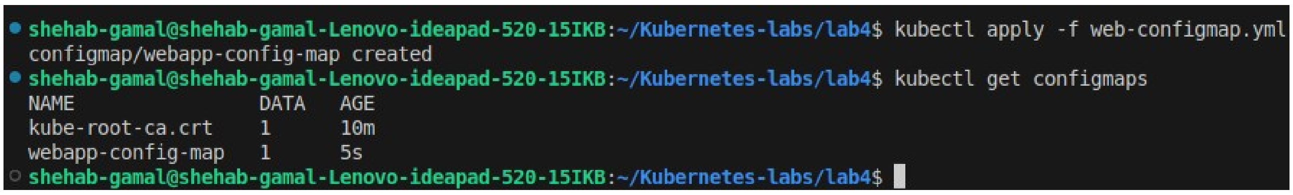
2. Create a new ConfigMap Use the spec given below.

➔ ConfigName Name: webapp-config-map
➔ Data: APP_COLOR=darkblue



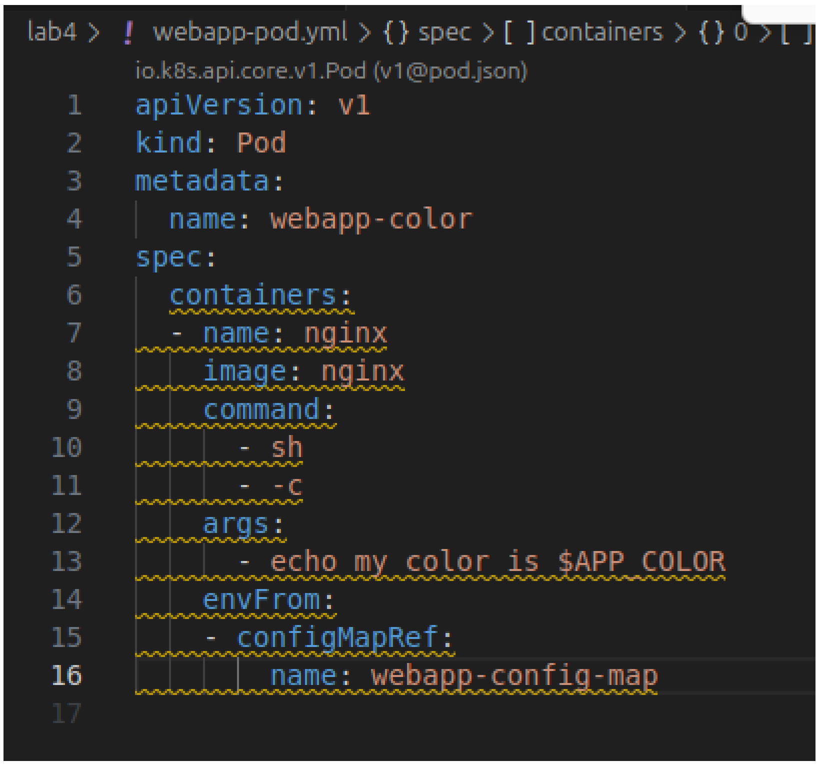
lab4 > ! web-configmap.yml > {} data > APP_COLOR
io.k8s.api.core.v1.ConfigMap (v1@configmap.json)

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: webapp-config-map
data:
  APP_COLOR: darkblue
```



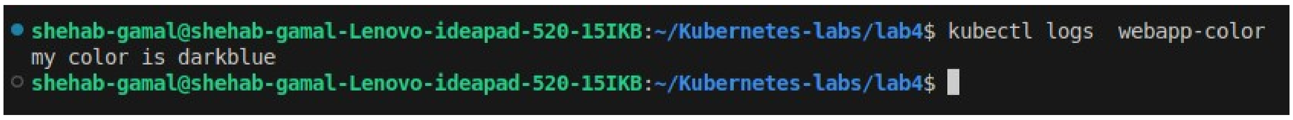
```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl apply -f web-configmap.yml
configmap/webapp-config-map created
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get configmaps
NAME                DATA   AGE
kube-root-ca.crt    1      10m
webapp-config-map   1      5s
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

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

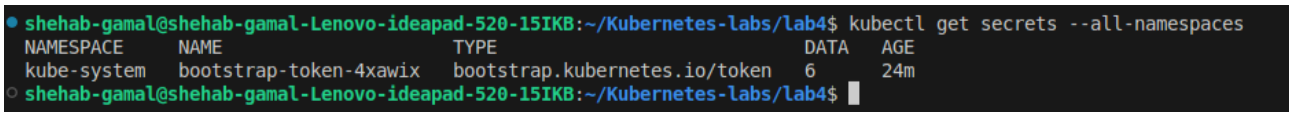


lab4 > ! webapp-pod.yml > {} spec > [ ] containers > {} 0 > [ ]
io.k8s.api.core.v1.Pod (v1@pod.json)

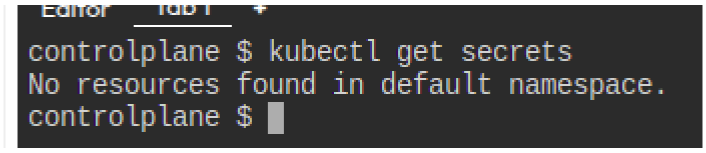
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp-color
spec:
  containers:
  - name: nginx
    image: nginx
    command:
      - sh
      - -c
    args:
      - echo my color is $APP_COLOR
    envFrom:
    - configMapRef:
        name: webapp-config-map
```


```
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl logs  webapp-color
  my color is darkblue
○ shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

4. How many Secrets exist in the cluster?

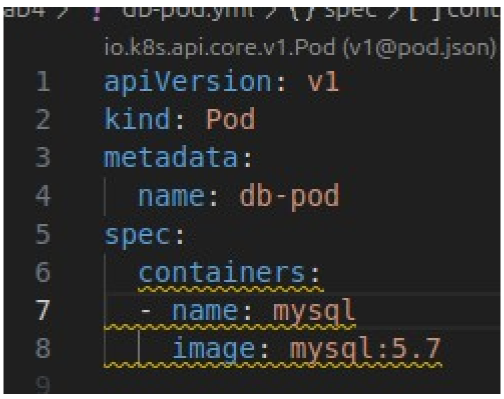

```
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get secrets --all-namespaces
  NAMESPACE     NAME                    TYPE                            DATA   AGE
  kube-system   bootstrap-token-4xawix  bootstrap.kubernetes.io/token   6      24m
○ shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

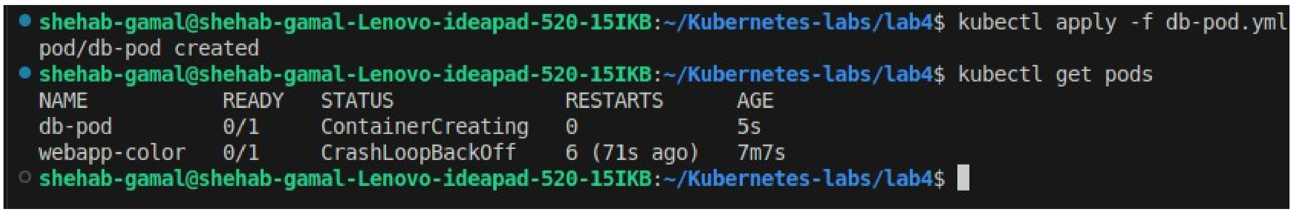
5. How many secrets are defined in the default-token secret?


```
controlplane $ kubectl get secrets
No resources found in default namespace.
controlplane $
```

Neither minikube nor killrcoda have default-token secret

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

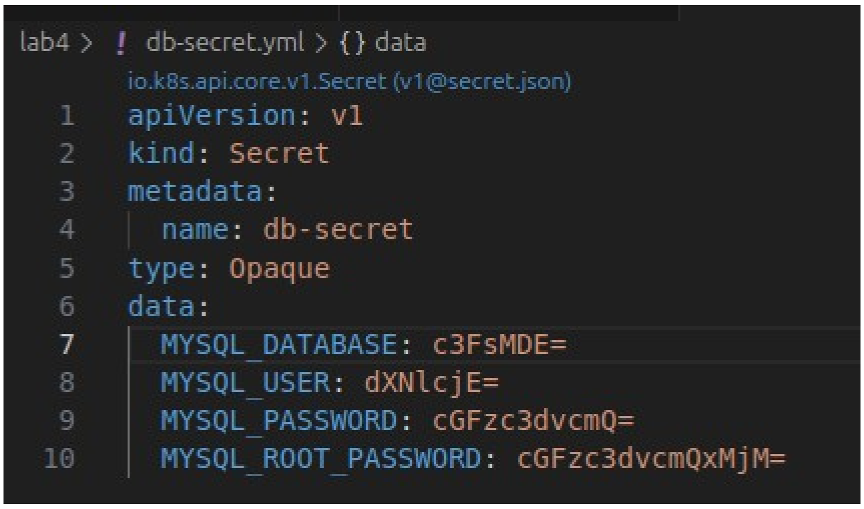

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.7
```


```
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl apply -f db-pod.yml
  pod/db-pod created
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get pods
  NAME           READY   STATUS              RESTARTS      AGE
  db-pod         0/1     ContainerCreating   0             5s
  webapp-color   0/1     CrashLoopBackOff    6 (71s ago)   7m7s
○ shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

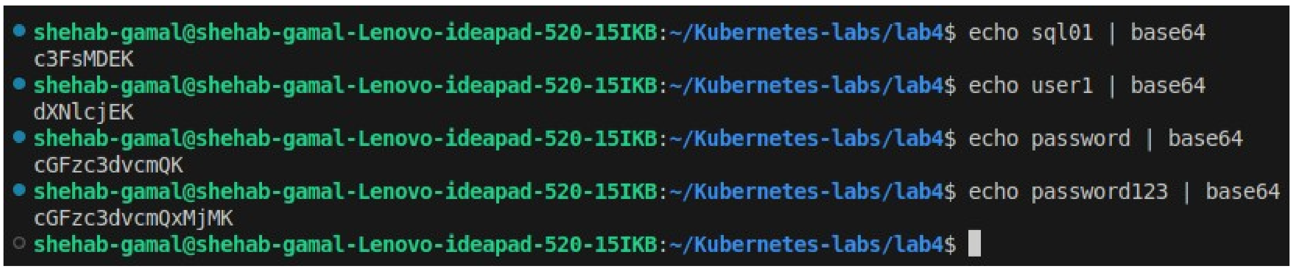
7. why is the db-pod status not ready?

Becaue MySQL image need its enviroment variable (MySql_root_password) to be set before starting.

8. 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
➔ Secret 3: MYSQL_PASSWORD=password
➔ Secret 4: MYSQL_ROOT_PASSWORD=password123

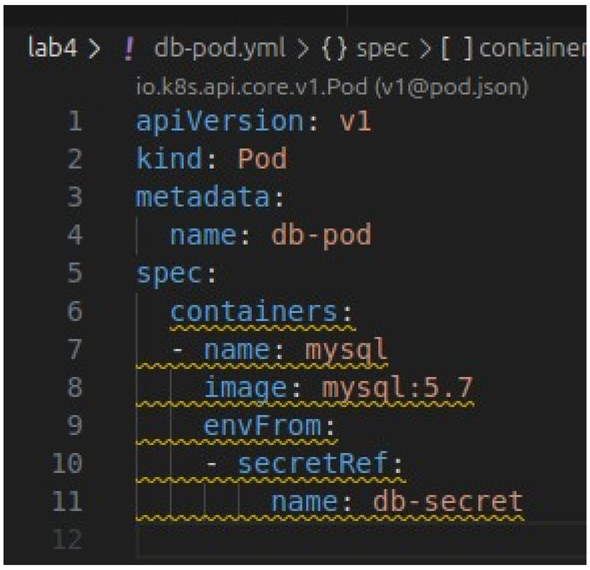

lab4 > ! db-secret.yml > {} data
io.k8s.api.core.v1.Secret (v1@secret.json)

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



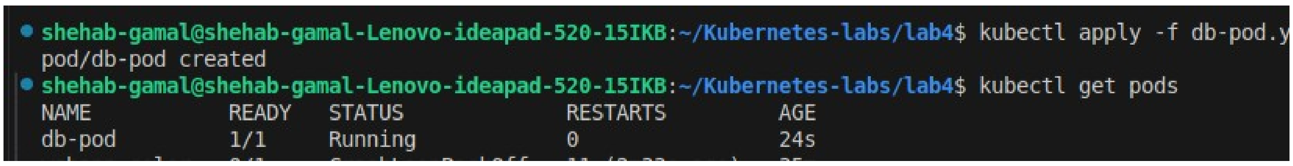
```
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ echo sql01 | base64
  c3FsMDEK
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ echo user1 | base64
  dXNlcjEK
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ echo password | base64
  cGFzc3dvcmQK
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ echo password123 | base64
  cGFzc3dvcmQxMjMK
○ shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ 
```

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


lab4 > ! db-pod.yml > {} spec > [ ] container
io.k8s.api.core.v1.Pod (v1@pod.json)

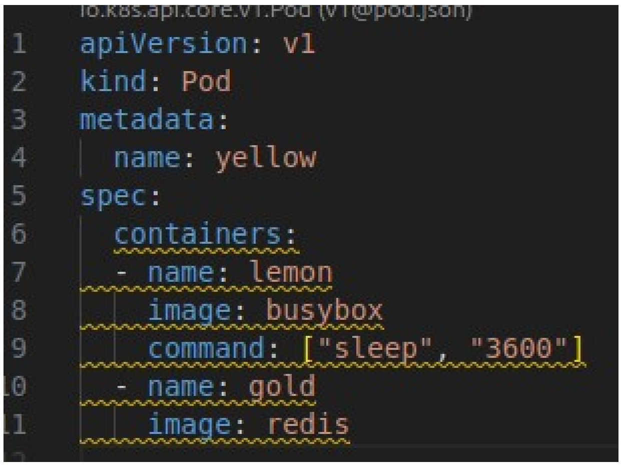
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: db-pod
spec:
  containers:
  - name: mysql
    image: mysql:5.7
    envFrom:
    - secretRef:
        name: db-secret

```

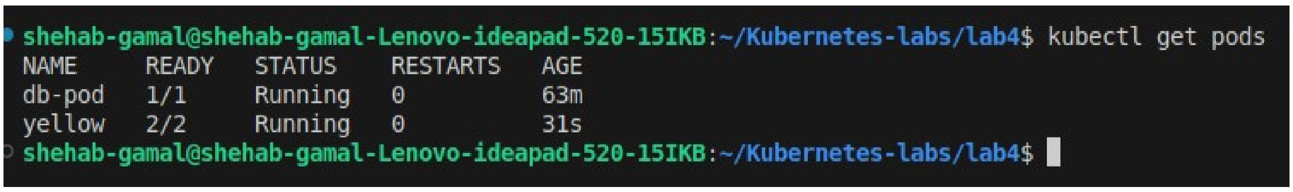


```
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl apply -f db-pod.y
  pod/db-pod created
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get pods
  NAME            READY   STATUS             RESTARTS          AGE
  db-pod          1/1     Running            0                 24s
```

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



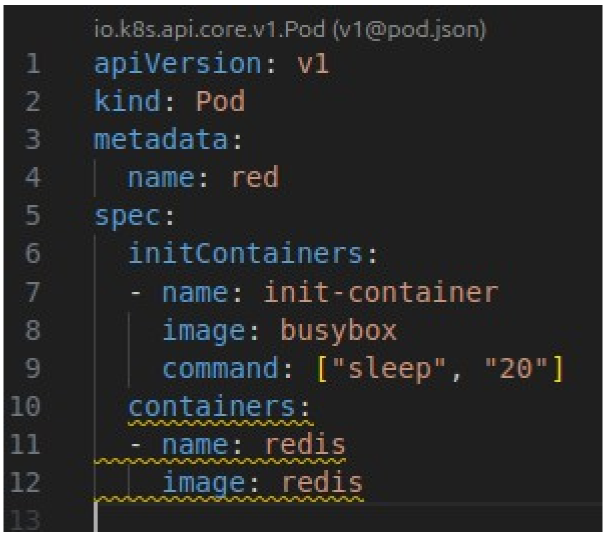
```
io.k8s.api.core.v1.Pod (v1@pod.json)
apiVersion: v1
kind: Pod
metadata:
  name: yellow
spec:
  containers:
  - name: lemon
    image: busybox
    command: ["sleep", "3600"]
  - name: gold
    image: redis
```



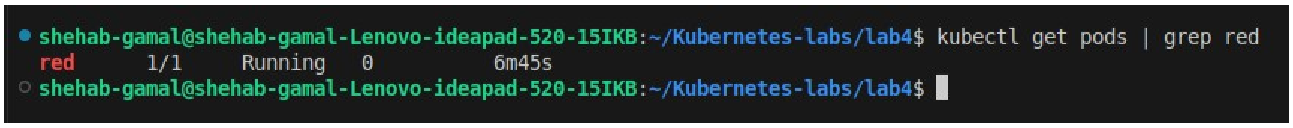
```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get pods
NAME      READY   STATUS    RESTARTS   AGE
db-pod    1/1     Running   0          63m
yellow    2/2     Running   0          31s
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

without command: [“sleep”, “3600”], kubernetes will keep starting lemon container as it sees it as completed so gold container wouldn’t be able to run.

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

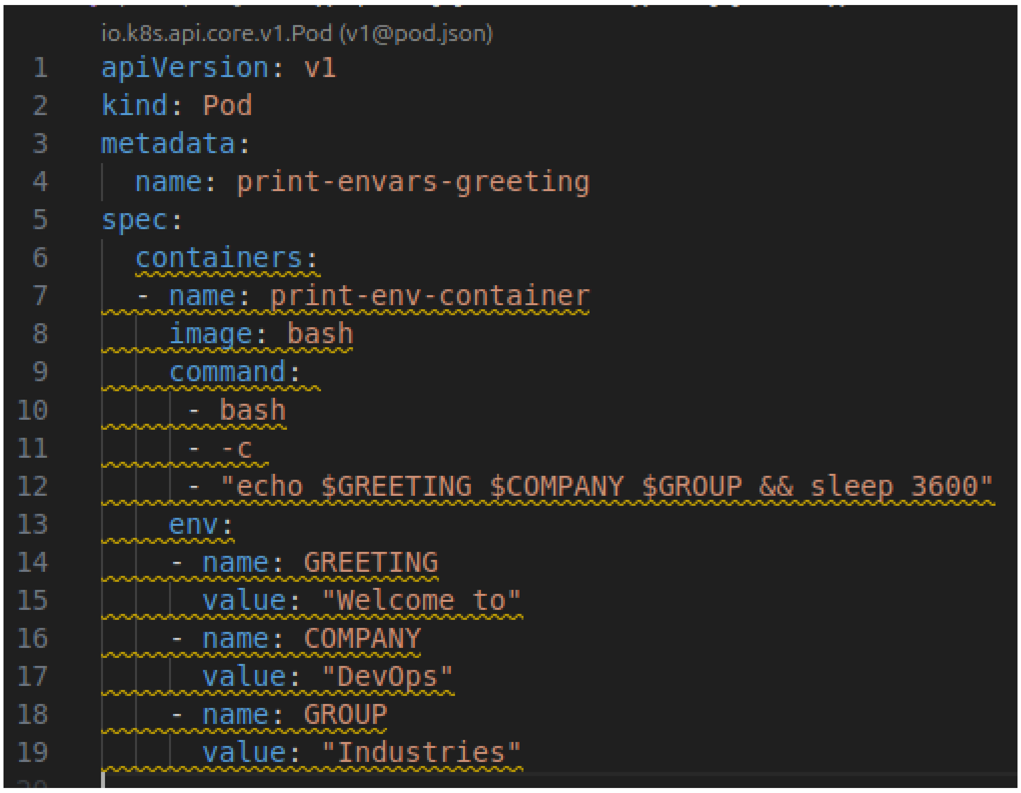


```
io.k8s.api.core.v1.Pod (v1@pod.json)
apiVersion: v1
kind: Pod
metadata:
  name: red
spec:
  initContainers:
  - name: init-container
    image: busybox
    command: ["sleep", "20"]
  containers:
  - name: redis
    image: redis
```

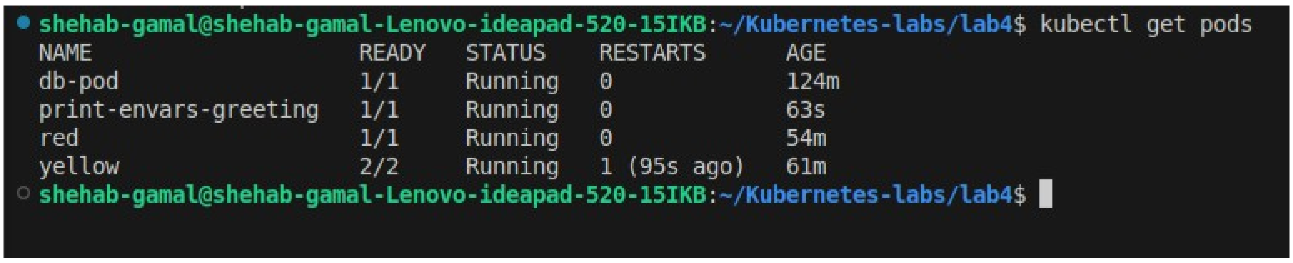


```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get pods | grep red
red       1/1     Running   0          6m45s
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

12.Create a pod named print-envars-greeting, Configure spec as, the container name should be print-env-container and use bash image, Create three environment variables:
➔ GREETING and its value should be “Welcome to”
➔ COMPANY and its value should be “DevOps”
➔ GROUP and its value should be “Industries”
Use command to echo ["$(GREETING) $(COMPANY) $(GROUP)"] message and sleep the container 3600.

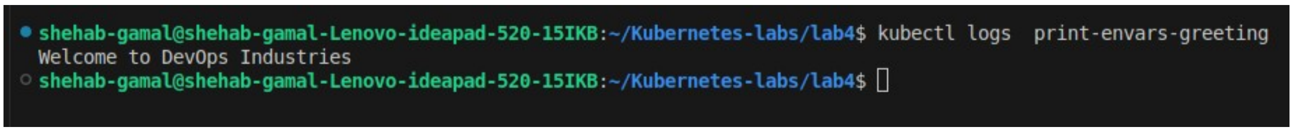


```
io.k8s.api.core.v1.Pod (v1@pod.json)
apiVersion: v1
kind: Pod
metadata:
  name: print-envars-greeting
spec:
  containers:
  - name: print-env-container
    image: bash
    command:
      - bash
      - -c
      - "echo $GREETING $COMPANY $GROUP && sleep 3600"
    env:
    - name: GREETING
      value: "Welcome to"
    - name: COMPANY
      value: "DevOps"
    - name: GROUP
      value: "Industries"
```



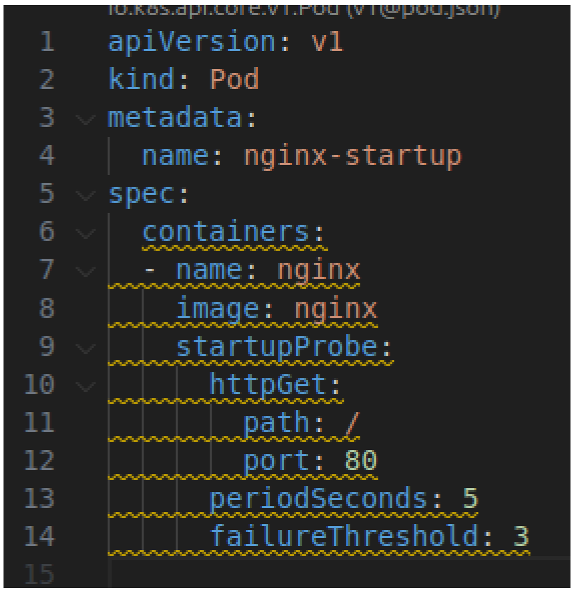
```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get pods
NAME                    READY   STATUS    RESTARTS      AGE
db-pod                  1/1     Running   0             124m
print-envars-greeting   1/1     Running   0             63s
red                     1/1     Running   0             54m
yellow                  2/2     Running   1 (95s ago)   61m
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

13.You can check the output using <kubctl logs -f [pod-name]> command.

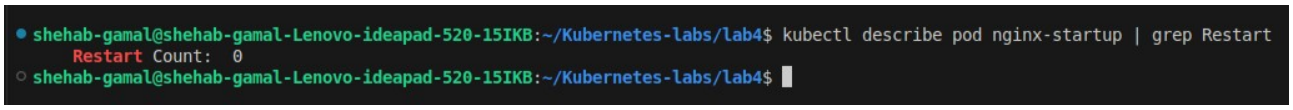


```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl logs  print-envars-greeting
Welcome to DevOps Industries
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

14.Create a pod with a container running the nginx image.
➔ Configure a startupProbe that checks if Nginx is ready using curl.
➔ Set the probe to check every 5 seconds with a failure threshold of 3.
➔ What happens if the container takes longer to start than expected?



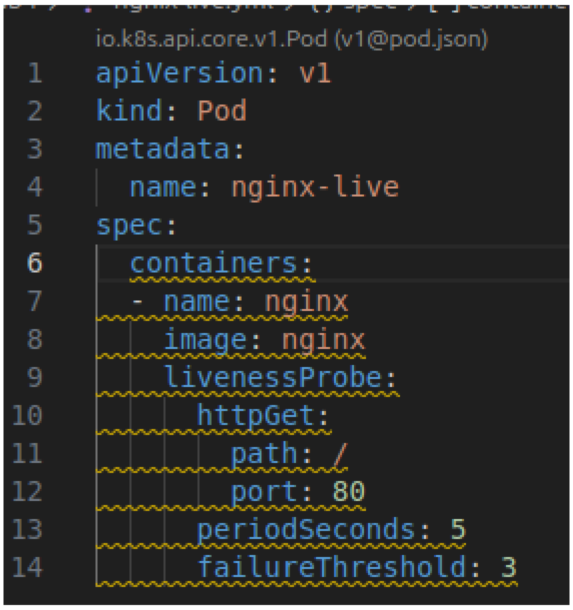
```yaml
io.k8s.api.core.v1.Pod (v1@pod.json)
apiVersion: v1
kind: Pod
metadata:
  name: nginx-startup
spec:
  containers:
  - name: nginx
    image: nginx
    startupProbe:
      httpGet:
        path: /
        port: 80
      periodSeconds: 5
      failureThreshold: 3
```



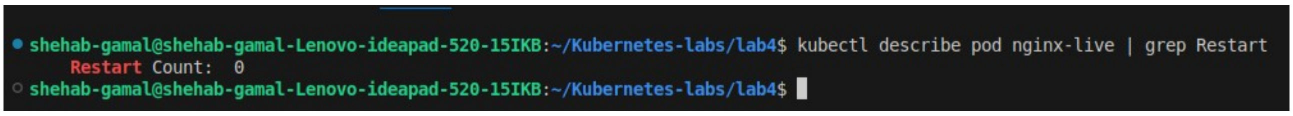
```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl describe pod nginx-startup | grep Restart
    Restart Count:  0
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ 
```

if the container takes longer to start than expected, then the startupProbe will fail then it will try for 2 more times and if it still fails then the pod will be stopped and restarted.

15.Deploy an Nginx pod with a livenessProbe that checks /



```yaml
io.k8s.api.core.v1.Pod (v1@pod.json)
apiVersion: v1
kind: Pod
metadata:
  name: nginx-live
spec:
  containers:
  - name: nginx
    image: nginx
    livenessProbe:
      httpGet:
        path: /
        port: 80
      periodSeconds: 5
      failureThreshold: 3
```

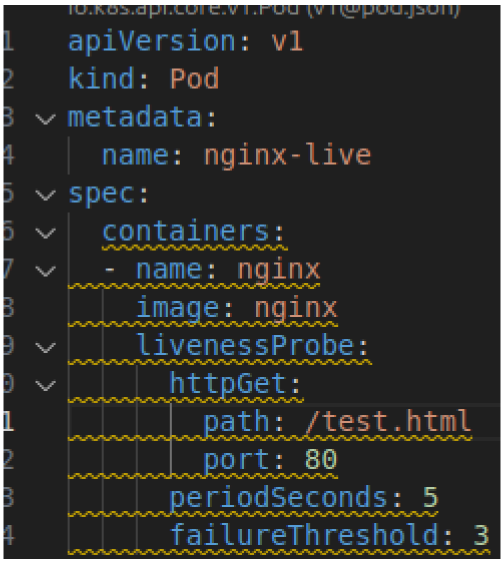


```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl describe pod nginx-live | grep Restart
    Restart Count:  0
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ 
```

16.What happens to the pod?

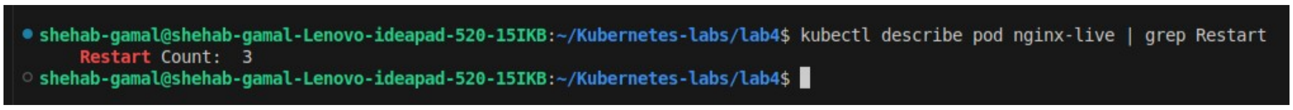
LivenessProbe will always keep checking the container every 5 seconds and if at any point it fails then it will try for 2 more times and if it still fails then the pod will be stopped and restarted.

17.Edit the livenessProbe inside the pod to /test.html.



```yaml
io.k8s.api.core.v1.Pod (v1@pod.json)
apiVersion: v1
kind: Pod
metadata:
  name: nginx-live
spec:
  containers:
  - name: nginx
    image: nginx
    livenessProbe:
      httpGet:
        path: /test.html
        port: 80
      periodSeconds: 5
      failureThreshold: 3
```

18.What happens to the pod after the edit?

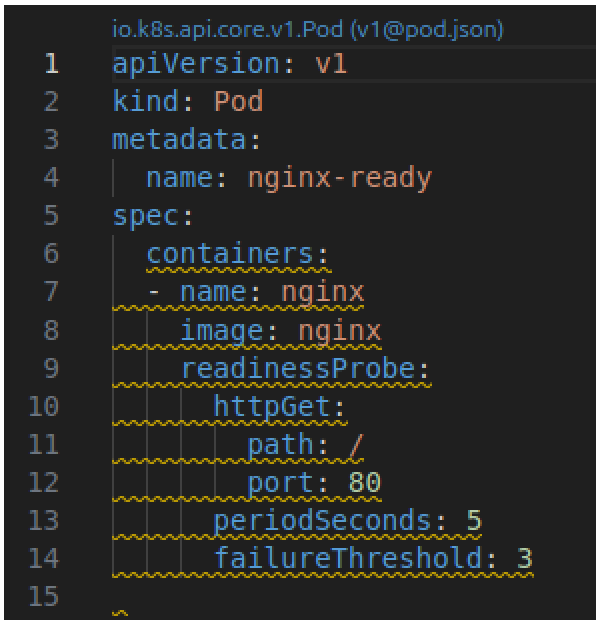


```
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl describe pod nginx-live | grep Restart
    Restart Count:  3
shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$
```

Since test.html don’t exist, then livenessProbe keeps failing and the pod gets restarted continuously.

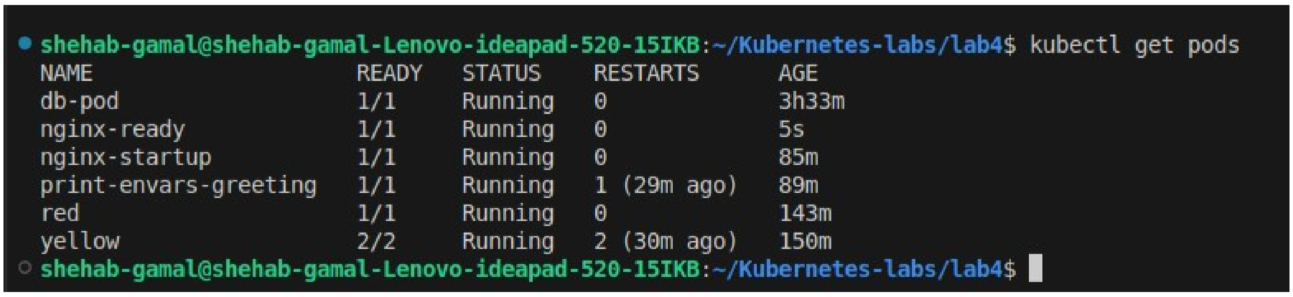
19.Create a pod running a simple Node.js web server.

20.Use a readinessProbe to check the HTTP endpoint (/health).



```yaml
io.k8s.api.core.v1.Pod (v1@pod.json)
apiVersion: v1
kind: Pod
metadata:
  name: nginx-ready
spec:
  containers:
  - name: nginx
    image: nginx
    readinessProbe:
      httpGet:
        path: /
        port: 80
      periodSeconds: 5
      failureThreshold: 3

```



```
● shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ kubectl get pods
  NAME                     READY   STATUS    RESTARTS      AGE
  db-pod                   1/1     Running   0             3h33m
  nginx-ready              1/1     Running   0             5s
  nginx-startup            1/1     Running   0             85m
  print-envars-greeting    1/1     Running   1 (29m ago)   89m
  red                      1/1     Running   0             143m
  yellow                   2/2     Running   2 (30m ago)   150m
○ shehab-gamal@shehab-gamal-Lenovo-ideapad-520-15IKB:~/Kubernetes-labs/lab4$ 
```

Note: I used nginx image instead of node because it kept crashing when configuring it as a web server.

21.Test what happens when the application is not ready.

If ReadinessProbe was checking an incorrect path for example The pod will start but will never become ready so ReadinessProbe will keep failing and if you try to access the pod from a service, it won't receive any traffic because it's not ready.