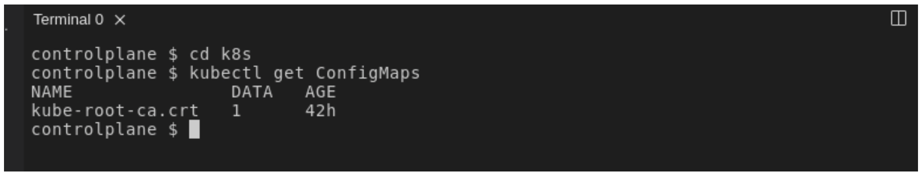
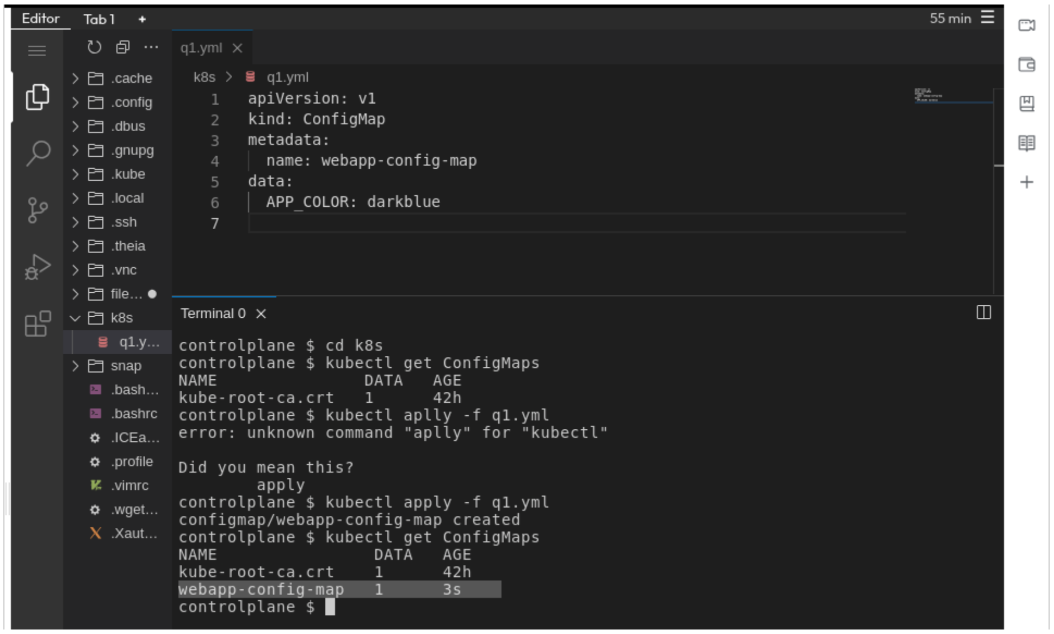
1. How many ConfigMaps exist in the cluster?



```
controlplane $ cd k8s
controlplane $ kubectl get ConfigMaps
NAME               DATA   AGE
kube-root-ca.crt   1      42h
controlplane $
```

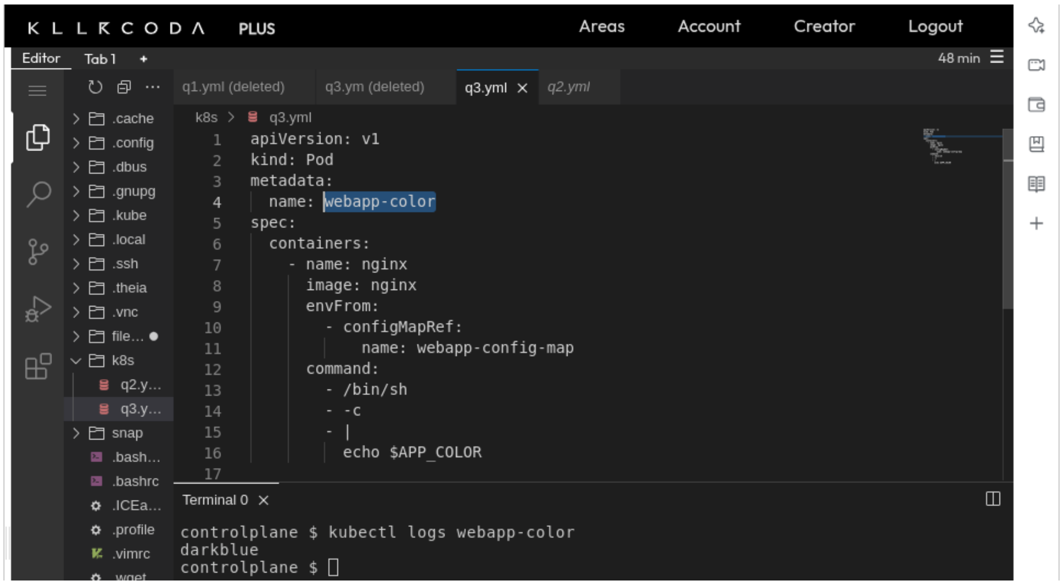
2. Create a new ConfigMap Use the spec given below. ➔ ConfigName Name: webapp-config-map ➔ Data: APP_COLOR=darkblue



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

```
controlplane $ cd k8s
controlplane $ kubectl get ConfigMaps
NAME               DATA   AGE
kube-root-ca.crt   1      42h
controlplane $ kubectl aplly -f q1.yml
error: unknown command "aplly" for "kubectl"

Did you mean this?
        apply
controlplane $ kubectl apply -f q1.yml
configmap/webapp-config-map created
controlplane $ kubectl get ConfigMaps
NAME                DATA   AGE
kube-root-ca.crt    1      42h
webapp-config-map   1      3s
controlplane $
```

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



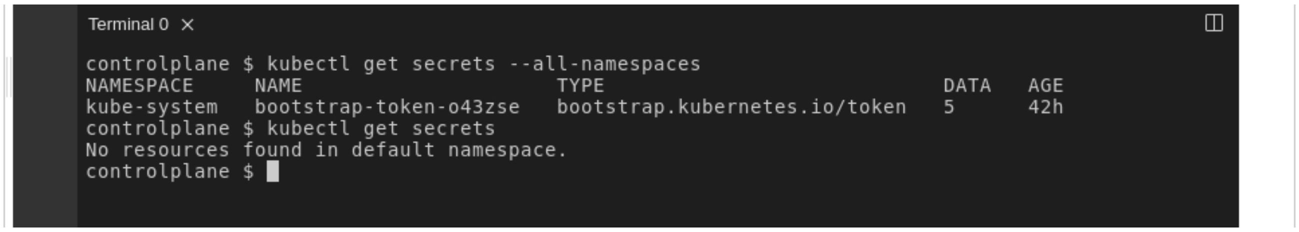
```
apiVersion: v1
kind: Pod
metadata:
  name: webapp-color
spec:
  containers:
    - name: nginx
      image: nginx
      envFrom:
        - configMapRef:
            name: webapp-config-map
      command:
        - /bin/sh
        - -c
        - |
          echo $APP_COLOR
```

```
controlplane $ kubectl logs webapp-color
darkblue
controlplane $
```

4. How many Secrets exist in the cluster?

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

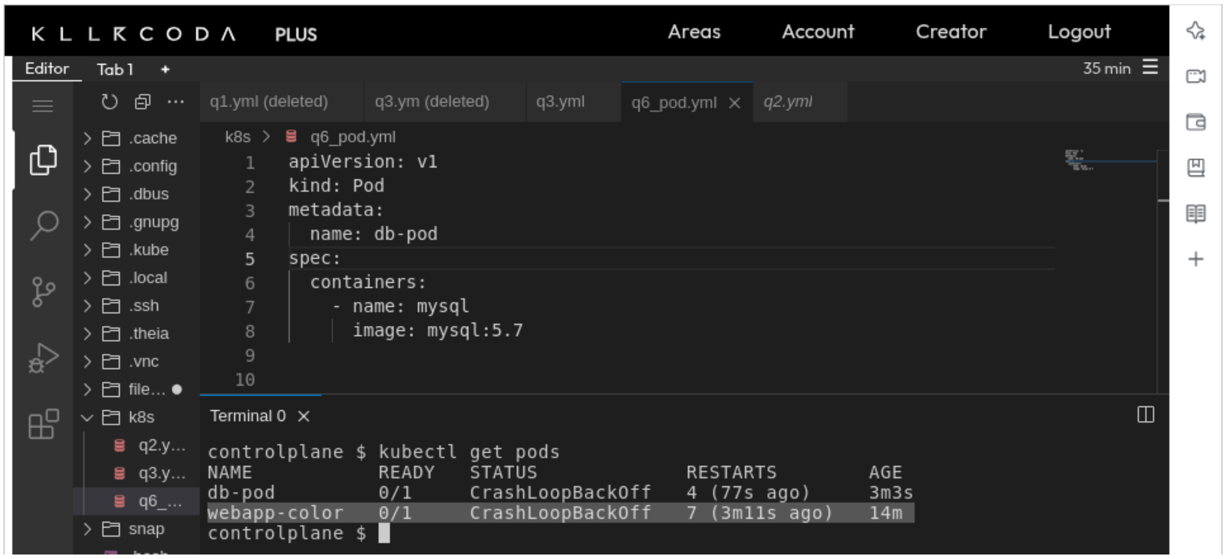
Answer off 4 &5



```
controlplane $ kubectl get secrets --all-namespaces
NAMESPACE     NAME                      TYPE                            DATA   AGE
kube-system   bootstrap-token-o43zse    bootstrap.kubernetes.io/token   5      42h
controlplane $ kubectl get secrets
No resources found in default namespace.
controlplane $
```

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

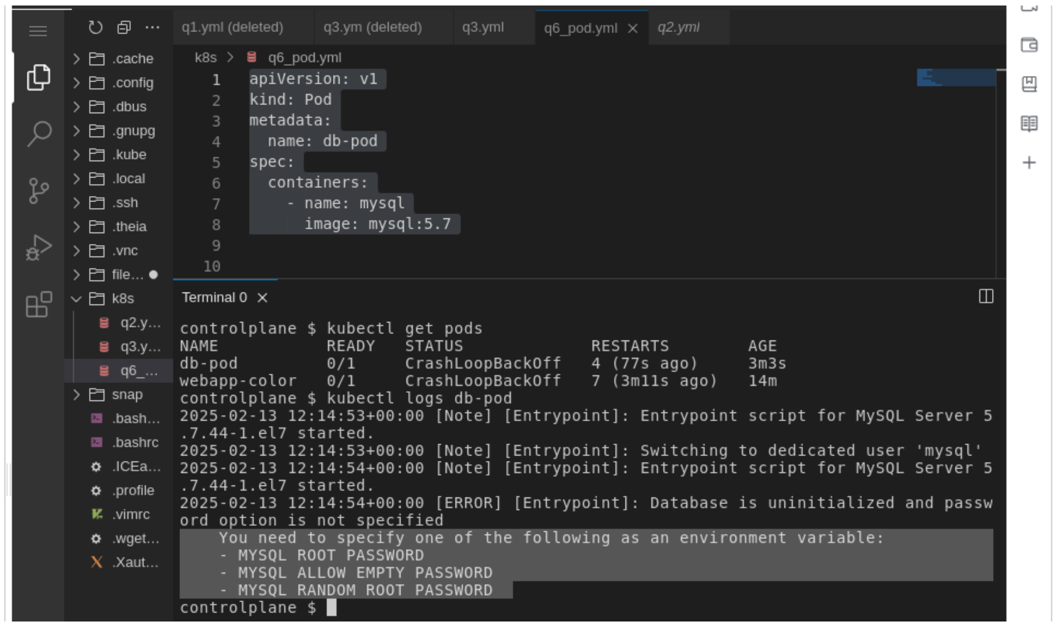
->Dp-pod crashed



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

```
controlplane $ kubectl get pods
NAME           READY   STATUS             RESTARTS        AGE
db-pod         0/1     CrashLoopBackOff   4 (77s ago)     3m3s
webapp-color   0/1     CrashLoopBackOff   7 (3m11s ago)   14m
controlplane $
```

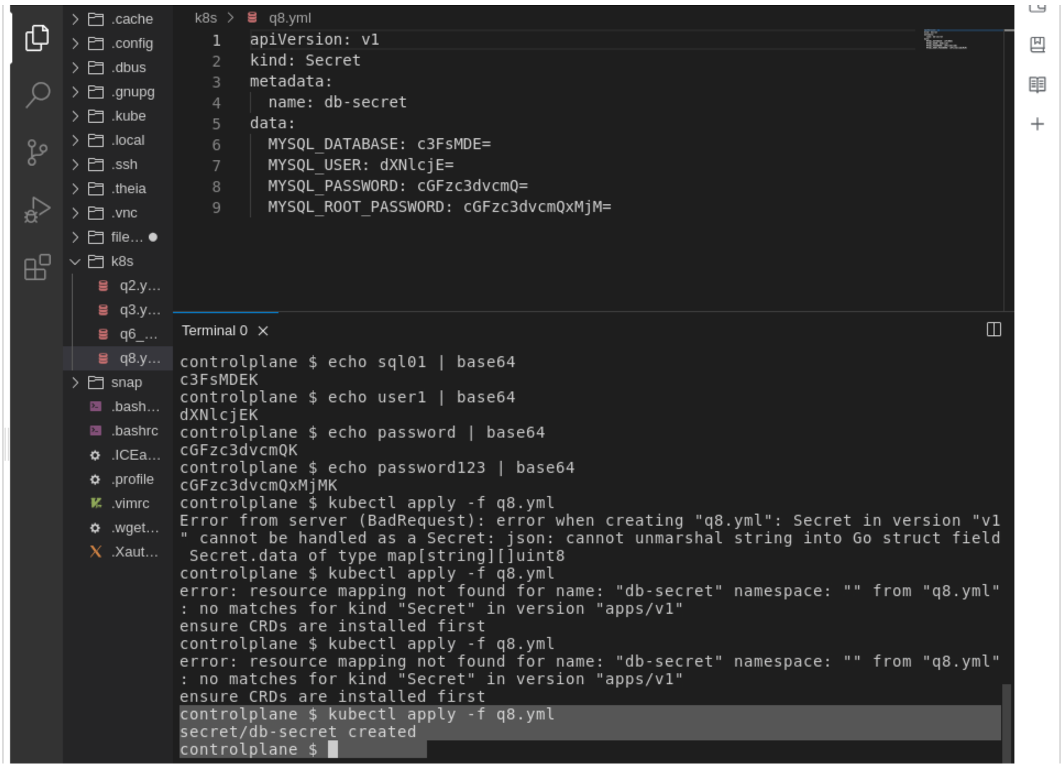
7. why is the db-pod status not ready? -> missing environment variables like MYSQL_ROOT_PASSWORD



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

```
controlplane $ kubectl get pods
NAME           READY   STATUS             RESTARTS        AGE
db-pod         0/1     CrashLoopBackOff   4 (77s ago)     3m3s
webapp-color   0/1     CrashLoopBackOff   7 (3m11s ago)   14m
controlplane $ kubectl logs db-pod
2025-02-13 12:14:53+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5
.7.44-1.el7 started.
2025-02-13 12:14:53+00:00 [Note] [Entrypoint]: Switching to dedicated user 'mysql'
2025-02-13 12:14:54+00:00 [Note] [Entrypoint]: Entrypoint script for MySQL Server 5
.7.44-1.el7 started.
2025-02-13 12:14:54+00:00 [ERROR] [Entrypoint]: Database is uninitialized and passw
ord 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 $
```

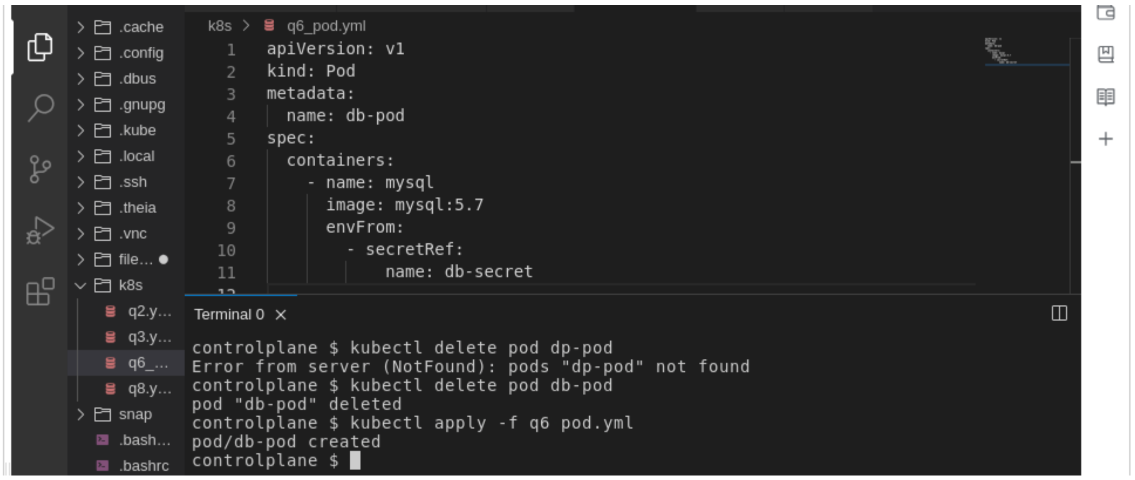
8. Create a new secret named db-secret with the data given below: ➔ Secret Name: db-secret ➔ Secret 1: MYSQL_DATABASE=sql01 ➔ Secret 2: MYSQL_USER=user1 ➔ Secret 3: MYSQL_PASSWORD=password ➔ Secret 4: MYSQL_ROOT_PASSWORD=password123



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

```
controlplane $ echo sql01 | base64
c3FsMDEK
controlplane $ echo user1 | base64
dXNlcjEK
controlplane $ echo password | base64
cGFzc3dvcmQK
controlplane $ echo password123 | base64
cGFzc3dvcmQxMjMK
controlplane $ kubectl apply -f q8.yml
Error from server (BadRequest): error when creating "q8.yml": Secret in version "v1
" cannot be handled as a Secret: json: cannot unmarshal string into Go struct field
 Secret.data of type map[string][]uint8
controlplane $ kubectl apply -f q8.yml
error: resource mapping not found for name: "db-secret" namespace: "" from "q8.yml"
: no matches for kind "Secret" in version "apps/v1"
ensure CRDs are installed first
controlplane $ kubectl apply -f q8.yml
error: resource mapping not found for name: "db-secret" namespace: "" from "q8.yml"
: no matches for kind "Secret" in version "apps/v1"
ensure CRDs are installed first
controlplane $ kubectl apply -f q8.yml
secret/db-secret created
controlplane $
```

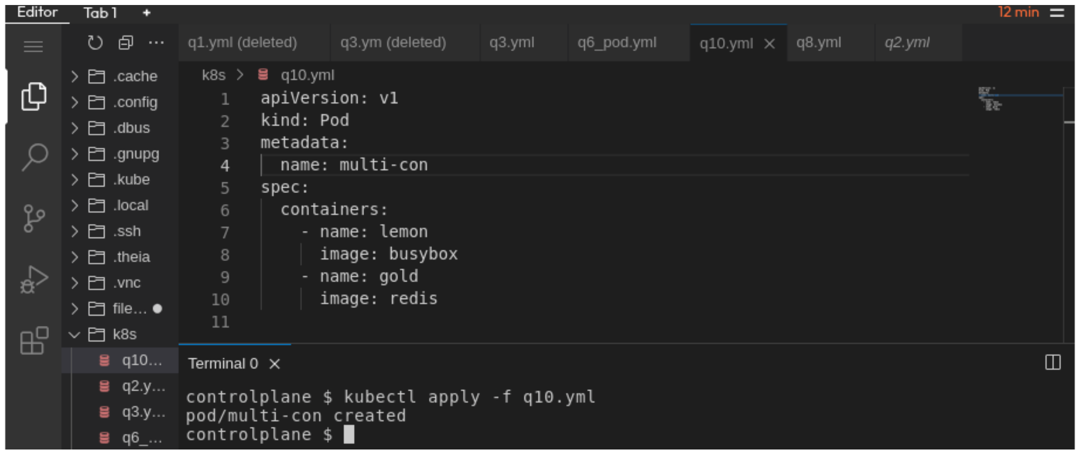
9. Configure db-pod to load environment variables from the newly created secret. Delete and recreate the pod if required.



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

```
controlplane $ kubectl delete pod dp-pod
Error from server (NotFound): pods "dp-pod" not found
controlplane $ kubectl delete pod db-pod
pod "db-pod" deleted
controlplane $ kubectl apply -f q6 pod.yml
pod/db-pod created
controlplane $
```

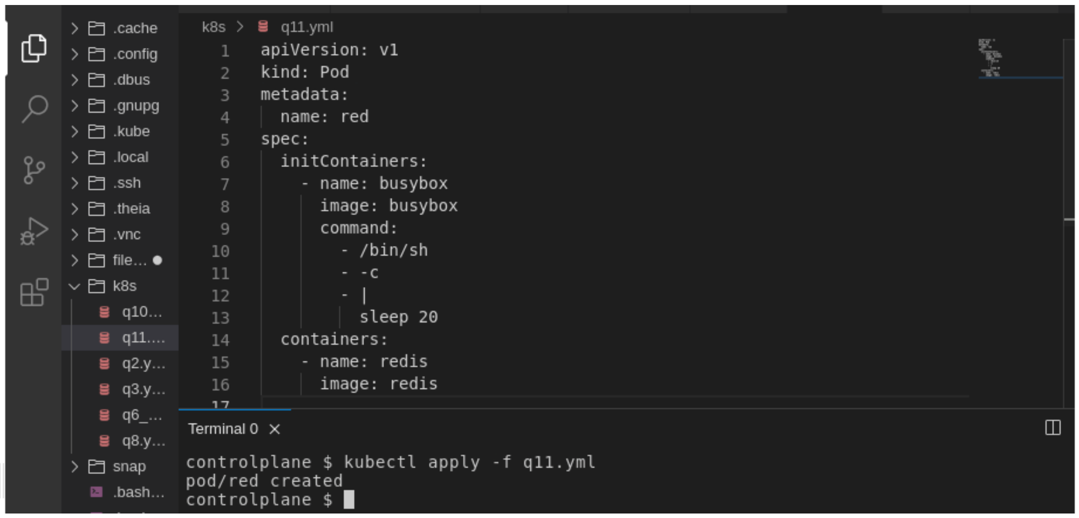
10. Create a multi-container pod with 2 containers. ➔ Name: yellow ➔ Container 1 Name: lemon ➔ Container 1 Image: busybox ➔ Container 2 Name: gold ➔ Container 2 Image: redis



```
apiVersion: v1
kind: Pod
metadata:
  name: multi-con
spec:
  containers:
    - name: lemon
      image: busybox
    - name: gold
      image: redis
```

```
controlplane $ kubectl apply -f q10.yml
pod/multi-con created
controlplane $
```

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

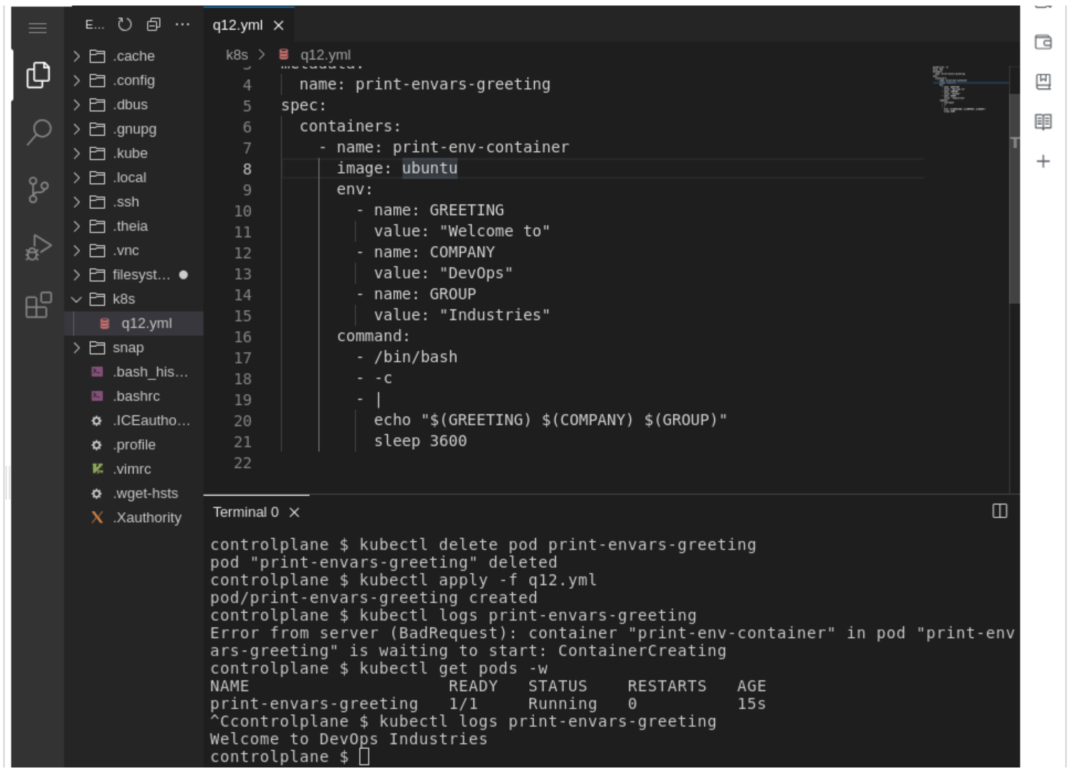


```
apiVersion: v1
kind: Pod
metadata:
  name: red
spec:
  initContainers:
    - name: busybox
      image: busybox
      command:
        - /bin/sh
        - -c
        - |
          sleep 20
  containers:
    - name: redis
      image: redis
```

```
controlplane $ kubectl apply -f q11.yml
pod/red created
controlplane $
```

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

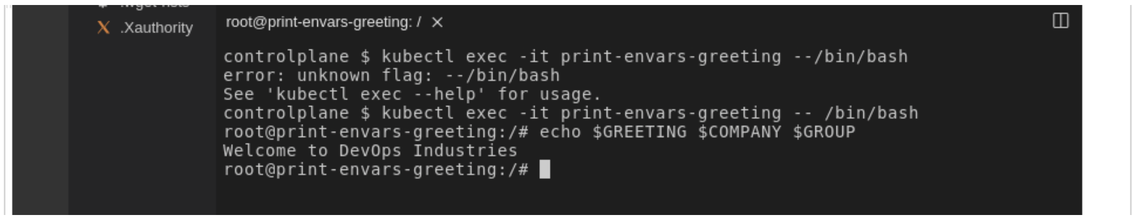
command to echo ["$(GREETING) $(COMPANY) $(GROUP)"] message and sleep the container 3600.



```
  name: print-envars-greeting
spec:
  containers:
    - name: print-env-container
      image: ubuntu
      env:
        - name: GREETING
          value: "Welcome to"
        - name: COMPANY
          value: "DevOps"
        - name: GROUP
          value: "Industries"
      command:
        - /bin/bash
        - -c
        - |
          echo "$(GREETING) $(COMPANY) $(GROUP)"
          sleep 3600
```

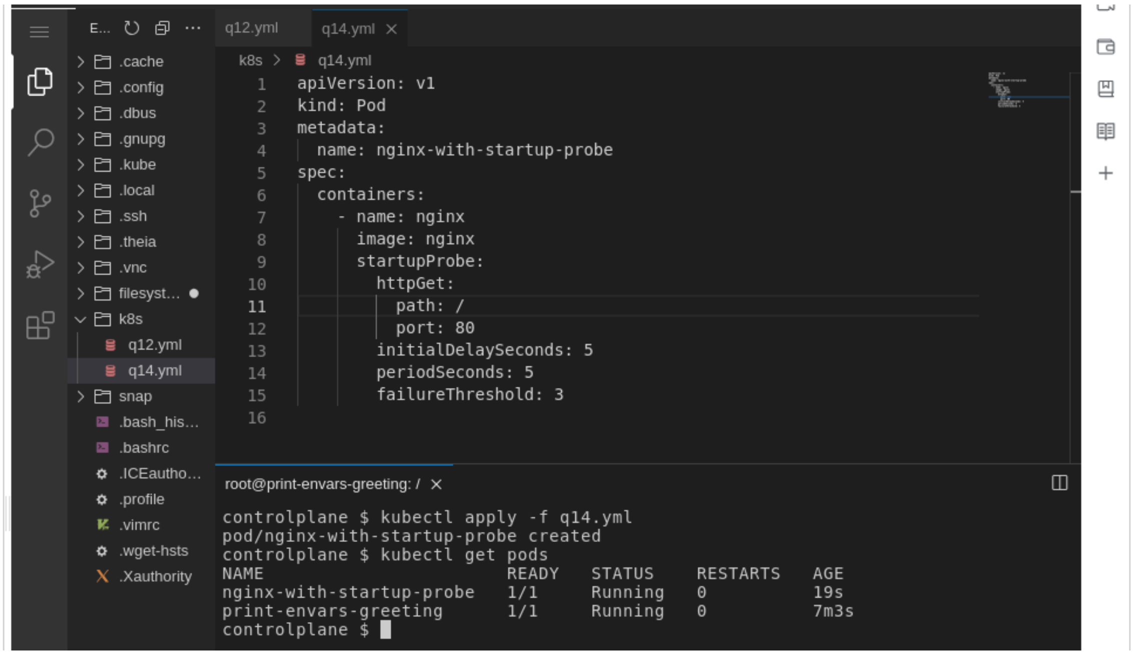
```
controlplane $ kubectl delete pod print-envars-greeting
pod "print-envars-greeting" deleted
controlplane $ kubectl apply -f q12.yml
pod/print-envars-greeting created
controlplane $ kubectl logs print-envars-greeting
Error from server (BadRequest): container "print-env-container" in pod "print-env
ars-greeting" is waiting to start: ContainerCreating
controlplane $ kubectl get pods -w
NAME                    READY   STATUS    RESTARTS   AGE
print-envars-greeting   1/1     Running   0          15s
^Ccontrolplane $ kubectl logs print-envars-greeting
Welcome to DevOps Industries
controlplane $
```

13. You can check the output using command.



```
controlplane $ kubectl exec -it print-envars-greeting --/bin/bash
error: unknown flag: --/bin/bash
See 'kubectl exec --help' for usage.
controlplane $ kubectl exec -it print-envars-greeting -- /bin/bash
root@print-envars-greeting:/# echo $GREETING $COMPANY $GROUP
Welcome to DevOps Industries
root@print-envars-greeting:/#
```

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?

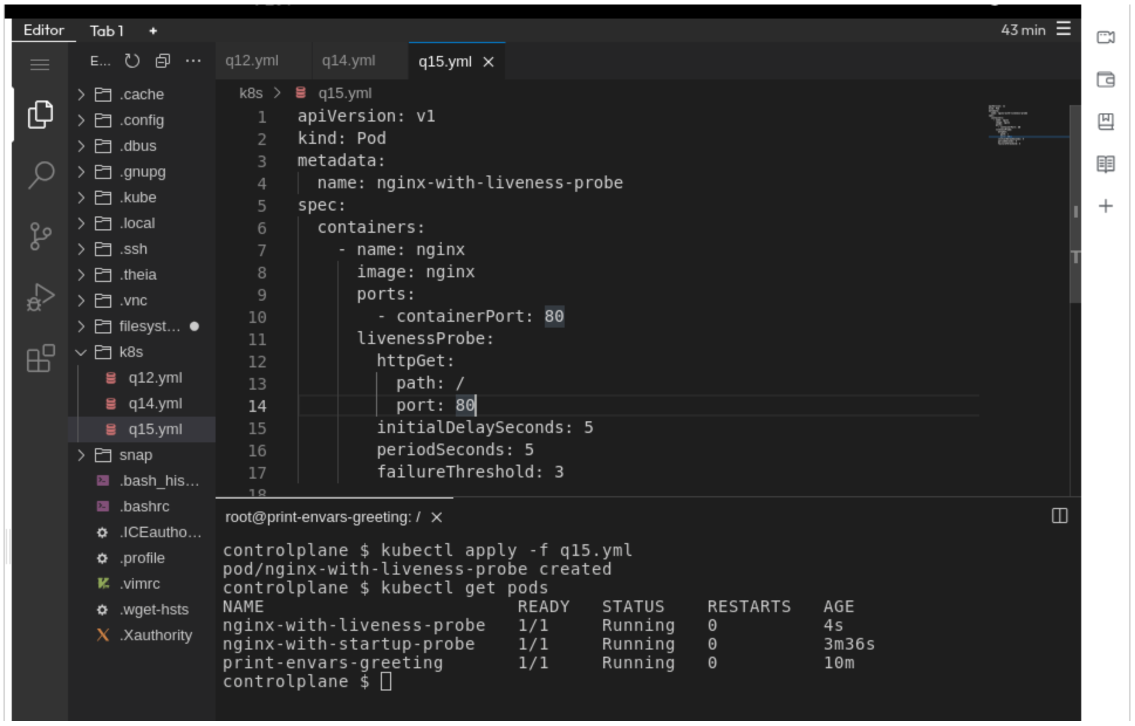


```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-with-startup-probe
spec:
  containers:
    - name: nginx
      image: nginx
      startupProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 5
        periodSeconds: 5
        failureThreshold: 3
```

```
controlplane $ kubectl apply -f q14.yml
pod/nginx-with-startup-probe created
controlplane $ kubectl get pods
NAME                       READY   STATUS    RESTARTS   AGE
nginx-with-startup-probe   1/1     Running   0          19s
print-envars-greeting      1/1     Running   0          7m3s
controlplane $ 
```

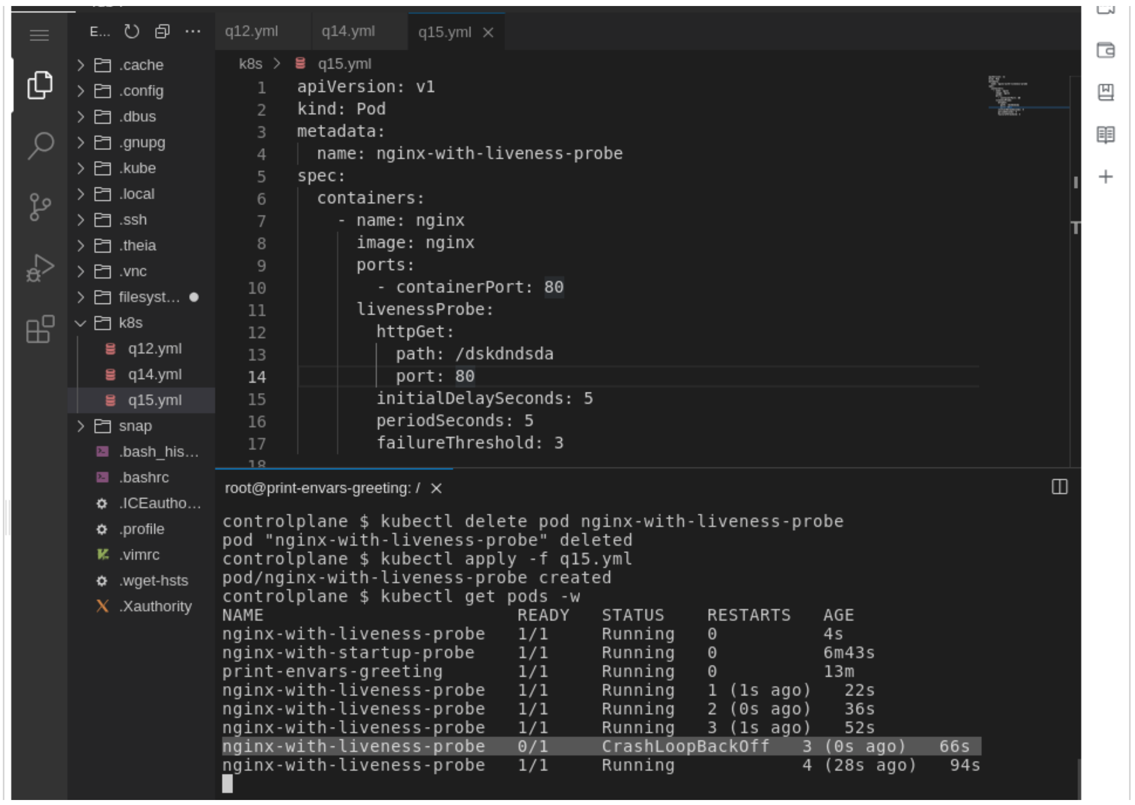
The startupProbe will keep checking every 5 seconds after the initial delay of 5 seconds.
If the Nginx service is not ready within the failureThreshold -> 3 failures, the container will be considered unhealthy, and Kubernetes will mark it as failed.
Kubernetes will not send traffic to the container until the startupProbe succeeds.
If the probe fails, Kubernetes will try to restart the pod.

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



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-with-liveness-probe
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
      livenessProbe:
        httpGet:
          path: /
          port: 80
        initialDelaySeconds: 5
        periodSeconds: 5
        failureThreshold: 3
```

```
controlplane $ kubectl apply -f q15.yml
pod/nginx-with-liveness-probe created
controlplane $ kubectl get pods
NAME                        READY   STATUS    RESTARTS   AGE
nginx-with-liveness-probe   1/1     Running   0          4s
nginx-with-startup-probe    1/1     Running   0          3m36s
print-envars-greeting       1/1     Running   0          10m
controlplane $
```



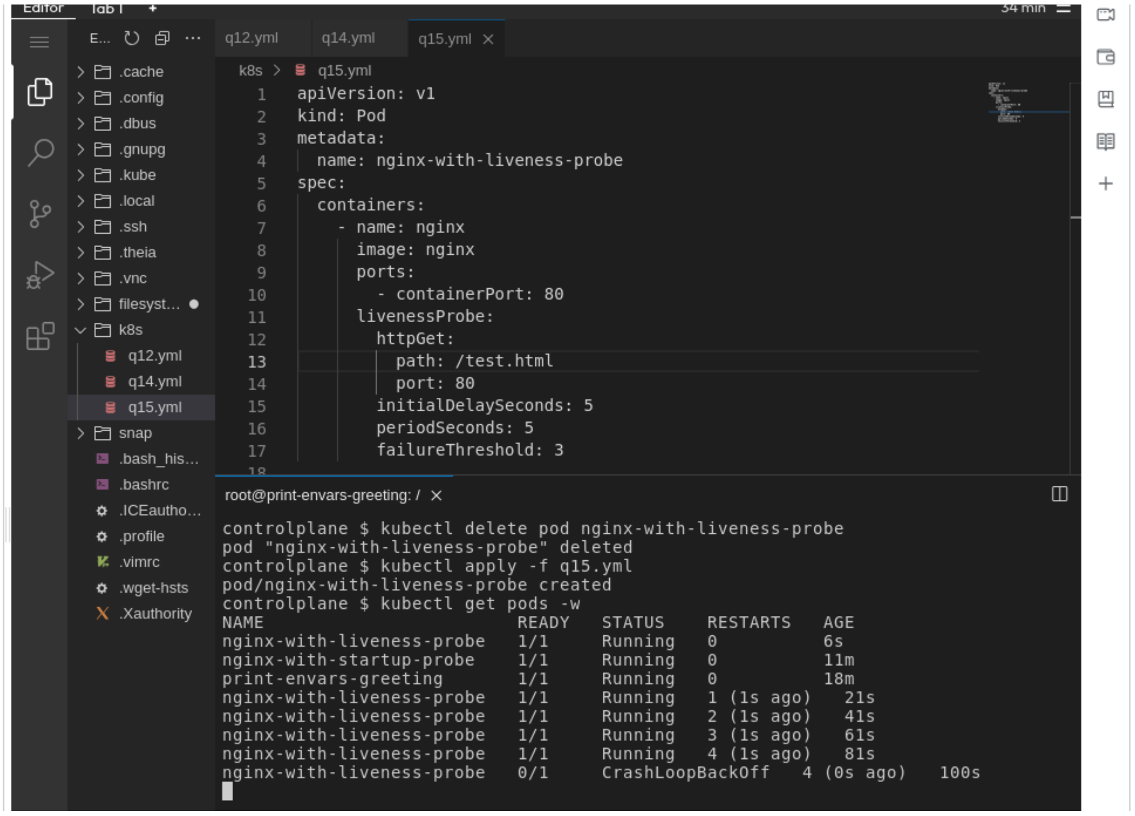
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-with-liveness-probe
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
      livenessProbe:
        httpGet:
          path: /dskdndsda
          port: 80
        initialDelaySeconds: 5
        periodSeconds: 5
        failureThreshold: 3
```

```
controlplane $ kubectl delete pod nginx-with-liveness-probe
pod "nginx-with-liveness-probe" deleted
controlplane $ kubectl apply -f q15.yml
pod/nginx-with-liveness-probe created
controlplane $ kubectl get pods -w
NAME                        READY   STATUS    RESTARTS   AGE
nginx-with-liveness-probe   1/1     Running   0          4s
nginx-with-startup-probe    1/1     Running   0          6m43s
print-envars-greeting       1/1     Running   0          13m
nginx-with-liveness-probe   1/1     Running   1 (1s ago)    22s
nginx-with-liveness-probe   1/1     Running   2 (0s ago)    36s
nginx-with-liveness-probe   1/1     Running   3 (1s ago)    52s
nginx-with-liveness-probe   0/1     CrashLoopBackOff   3 (0s ago)    66s
nginx-with-liveness-probe   1/1     Running            4 (28s ago)   94s
```

16. What happens to the pod?

The liveness probe will continue to pass every 5 seconds, and the pod will remain in the Running state.
Kubernetes will consider the pod healthy and continue serving traffic to it.

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



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-with-liveness-probe
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
      livenessProbe:
        httpGet:
          path: /test.html
          port: 80
        initialDelaySeconds: 5
        periodSeconds: 5
        failureThreshold: 3
```

```
controlplane $ kubectl delete pod nginx-with-liveness-probe
pod "nginx-with-liveness-probe" deleted
controlplane $ kubectl apply -f q15.yml
pod/nginx-with-liveness-probe created
controlplane $ kubectl get pods -w
NAME                        READY   STATUS    RESTARTS   AGE
nginx-with-liveness-probe   1/1     Running   0          6s
nginx-with-startup-probe    1/1     Running   0          11m
print-envars-greeting       1/1     Running   0          18m
nginx-with-liveness-probe   1/1     Running   1 (1s ago)   21s
nginx-with-liveness-probe   1/1     Running   2 (1s ago)   41s
nginx-with-liveness-probe   1/1     Running   3 (1s ago)   61s
nginx-with-liveness-probe   1/1     Running   4 (1s ago)   81s
nginx-with-liveness-probe   0/1     CrashLoopBackOff   4 (0s ago)   100s
```

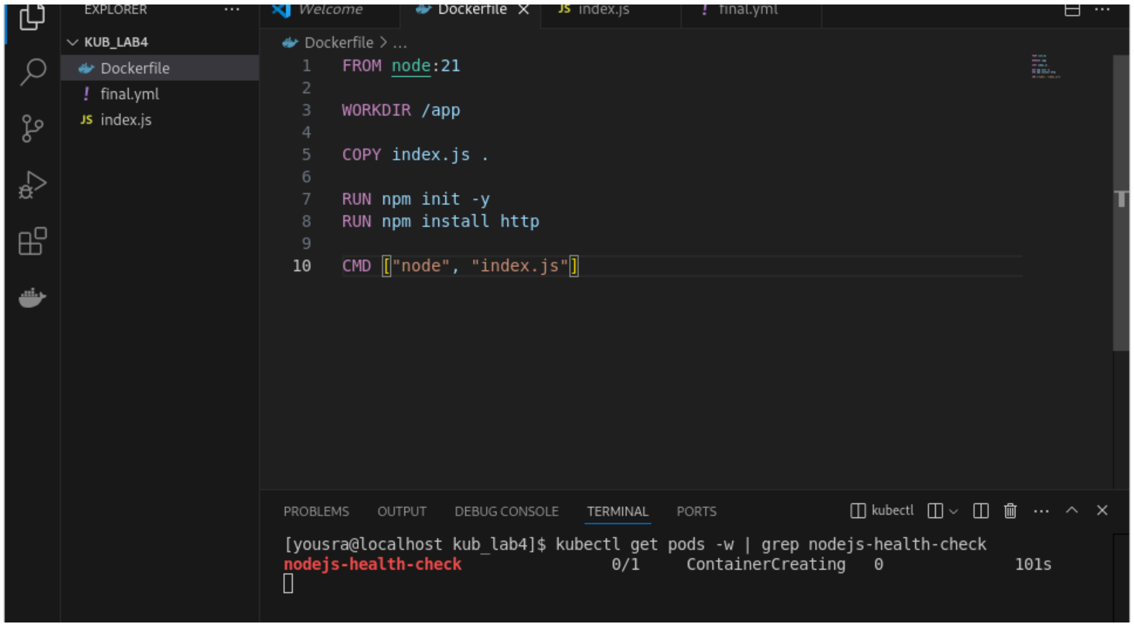
18. What happens to the pod after the edit?

/test.html doesn’t exist or the probe path is incorrect, the livenessProbe will fail, and the pod will keep restarting until it either passes the probe or is manually corrected.
If the path /test.html is valid (or if you create that file), the probe will succeed, and the pod will remain healthy.

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

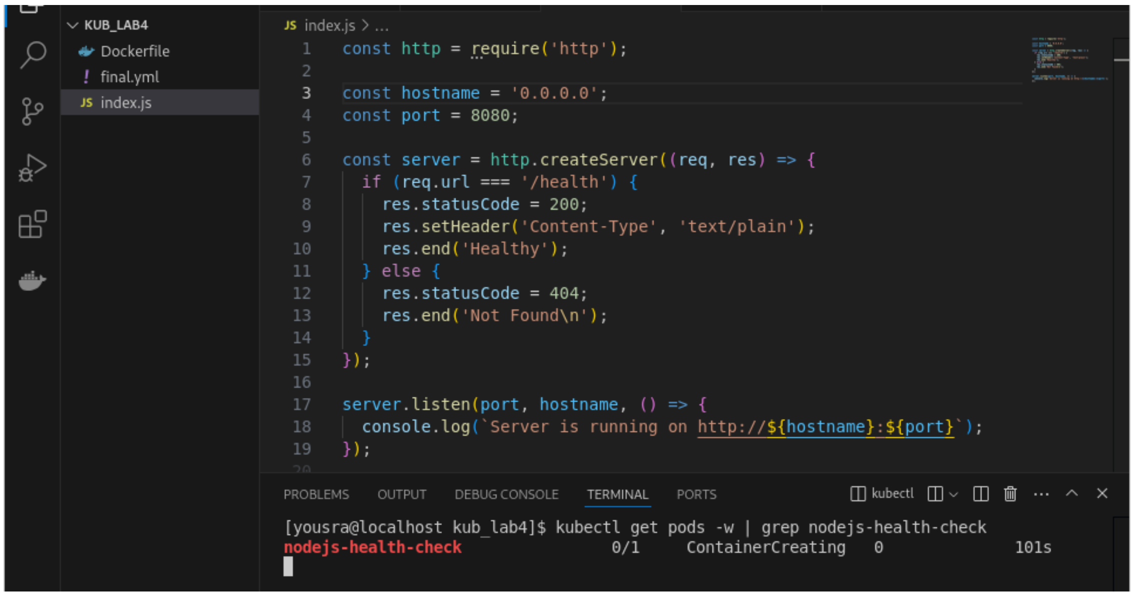
20. Use a readinessProbe to check the HTTP endpoint (/health).

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



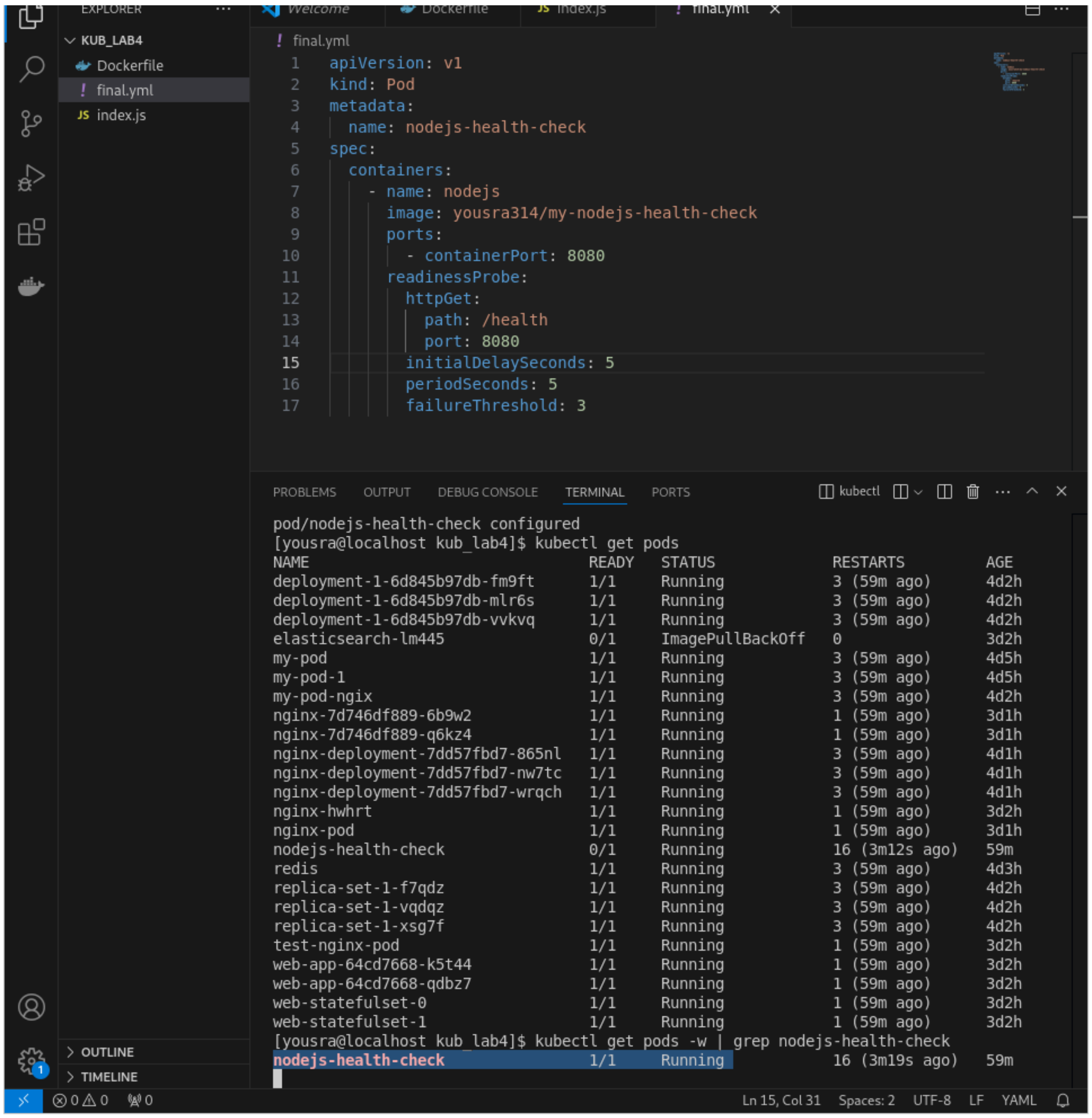
```
FROM node:21

WORKDIR /app

COPY index.js .

RUN npm init -y
RUN npm install http

CMD ["node", "index.js"]
```

```
[yousra@localhost kub_lab4]$ kubectl get pods -w | grep nodejs-health-check
nodejs-health-check                0/1     ContainerCreating   0          101s
```



```
const http = require('http');

const hostname = '0.0.0.0';
const port = 8080;

const server = http.createServer((req, res) => {
  if (req.url === '/health') {
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/plain');
    res.end('Healthy');
  } else {
    res.statusCode = 404;
    res.end('Not Found\n');
  }
});

server.listen(port, hostname, () => {
  console.log(`Server is running on http://${hostname}:${port}`);
});
```

```
[yousra@localhost kub_lab4]$ kubectl get pods -w | grep nodejs-health-check
nodejs-health-check                0/1     ContainerCreating   0          101s
```



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nodejs-health-check
spec:
  containers:
    - name: nodejs
      image: yousra314/my-nodejs-health-check
      ports:
        - containerPort: 8080
      readinessProbe:
        httpGet:
          path: /health
          port: 8080
        initialDelaySeconds: 5
        periodSeconds: 5
        failureThreshold: 3
```

```
pod/nodejs-health-check configured
[yousra@localhost kub_lab4]$ kubectl get pods
NAME                              READY   STATUS             RESTARTS        AGE
deployment-1-6d845b97db-fm9ft     1/1     Running            3 (59m ago)     4d2h
deployment-1-6d845b97db-mlr6s     1/1     Running            3 (59m ago)     4d2h
deployment-1-6d845b97db-vvkvq     1/1     Running            3 (59m ago)     4d2h
elasticsearch-lm445               0/1     ImagePullBackOff   0               3d2h
my-pod                            1/1     Running            3 (59m ago)     4d5h
my-pod-1                          1/1     Running            3 (59m ago)     4d5h
my-pod-ngix                       1/1     Running            3 (59m ago)     4d2h
nginx-7d746df889-6b9w2            1/1     Running            1 (59m ago)     3d1h
nginx-7d746df889-q6kz4            1/1     Running            1 (59m ago)     3d1h
nginx-deployment-7dd57fbd7-865nl  1/1     Running            3 (59m ago)     4d1h
nginx-deployment-7dd57fbd7-nw7tc  1/1     Running            3 (59m ago)     4d1h
nginx-deployment-7dd57fbd7-wrqch  1/1     Running            3 (59m ago)     4d1h
nginx-hwhrt                       1/1     Running            1 (59m ago)     3d2h
nginx-pod                         1/1     Running            1 (59m ago)     3d1h
nodejs-health-check               0/1     Running            16 (3m12s ago)  59m
redis                             1/1     Running            3 (59m ago)     4d3h
replica-set-1-f7qdz               1/1     Running            3 (59m ago)     4d2h
replica-set-1-vqdqz               1/1     Running            3 (59m ago)     4d2h
replica-set-1-xsg7f               1/1     Running            3 (59m ago)     4d2h
test-nginx-pod                    1/1     Running            1 (59m ago)     3d2h
web-app-64cd7668-k5t44            1/1     Running            1 (59m ago)     3d2h
web-app-64cd7668-qdbz7            1/1     Running            1 (59m ago)     3d2h
web-statefulset-0                 1/1     Running            1 (59m ago)     3d2h
web-statefulset-1                 1/1     Running            1 (59m ago)     3d2h
[yousra@localhost kub_lab4]$ kubectl get pods -w | grep nodejs-health-check
nodejs-health-check               1/1     Running            16 (3m19s ago)  59m
```