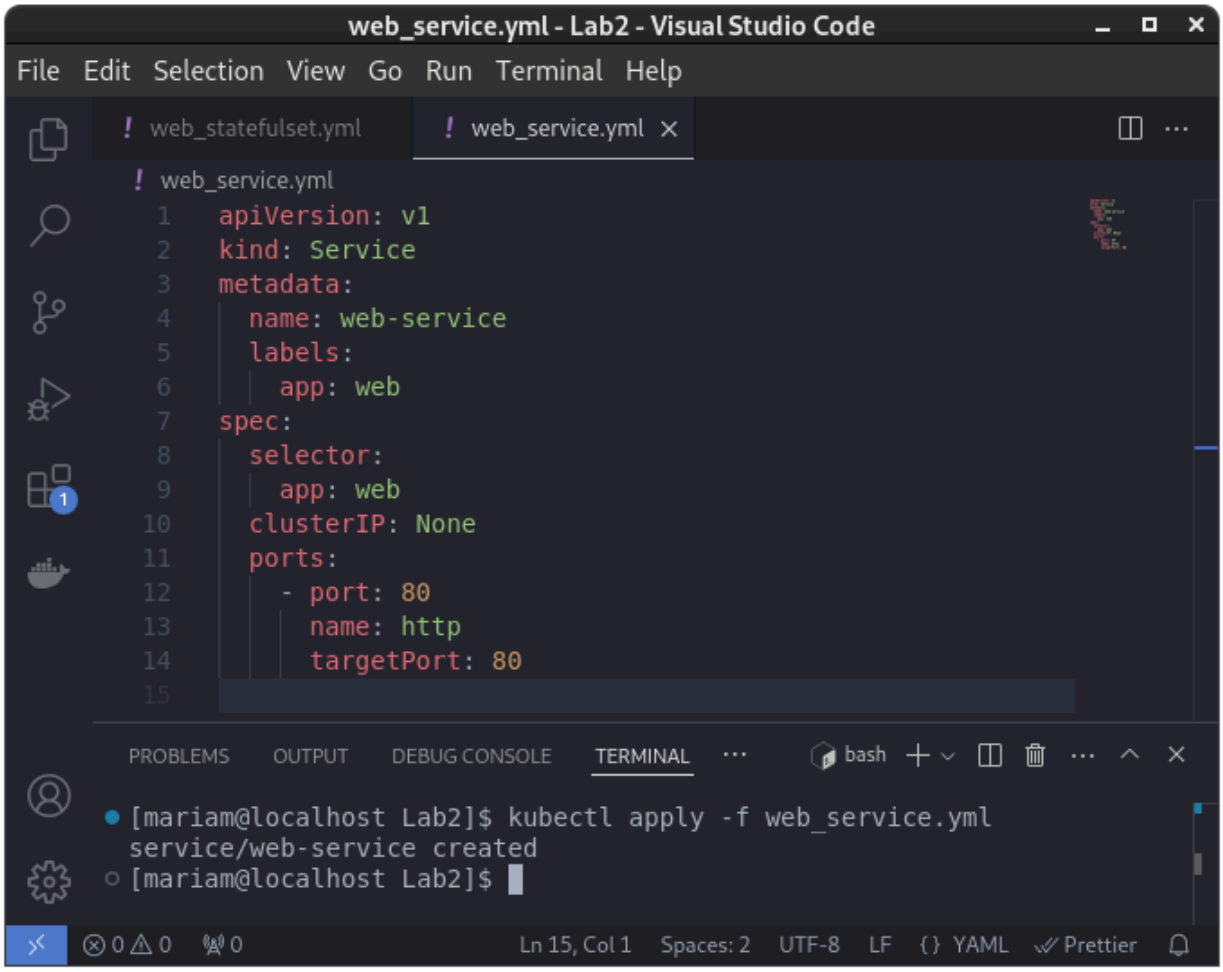
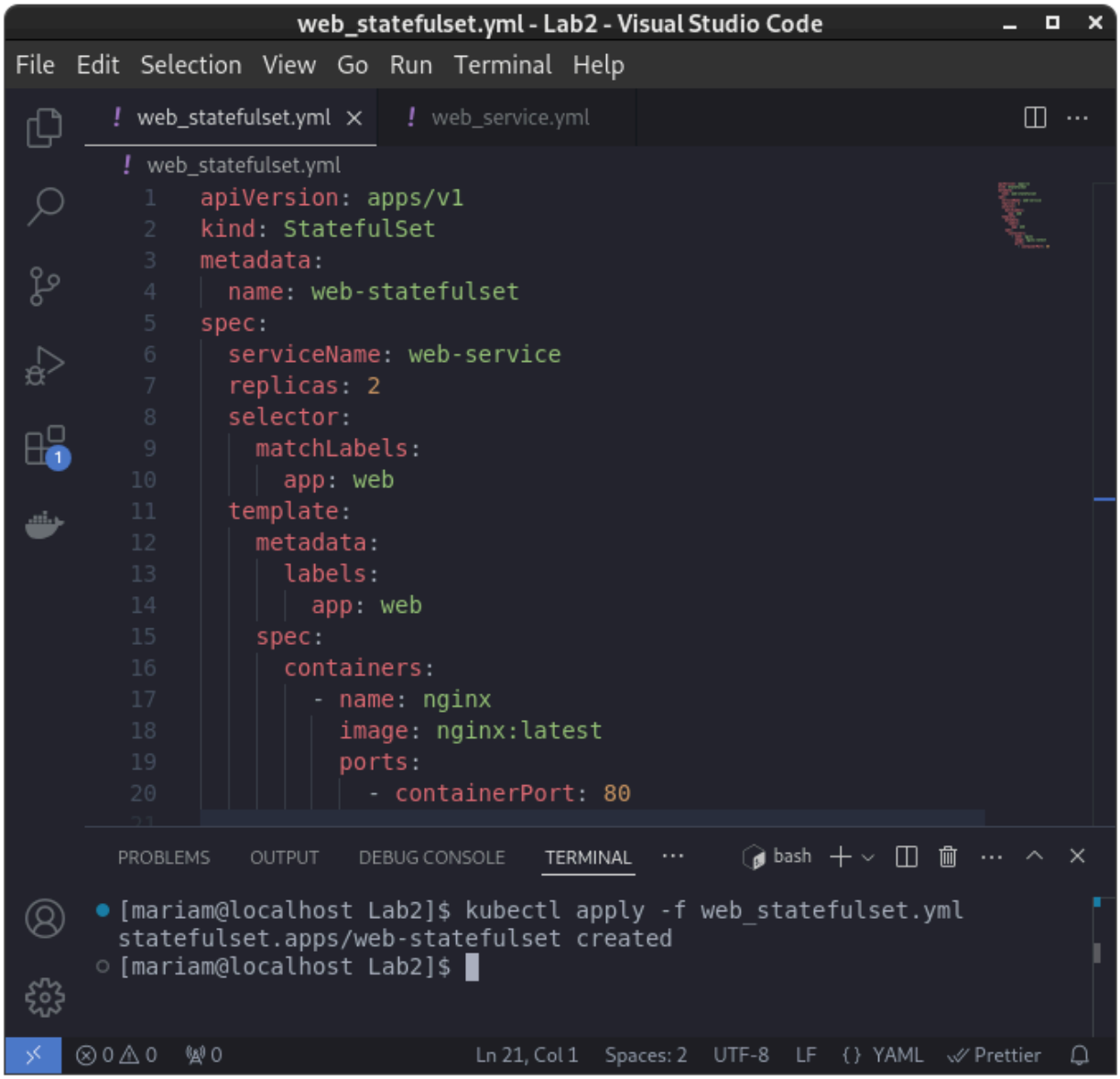
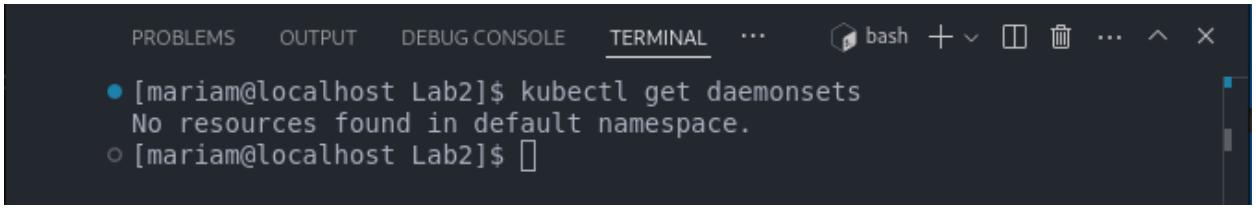
1. Create a StatefulSet named web-statefulset with 2 replicas using the nginx image.
The StatefulSet should have a Headless Service named web-service

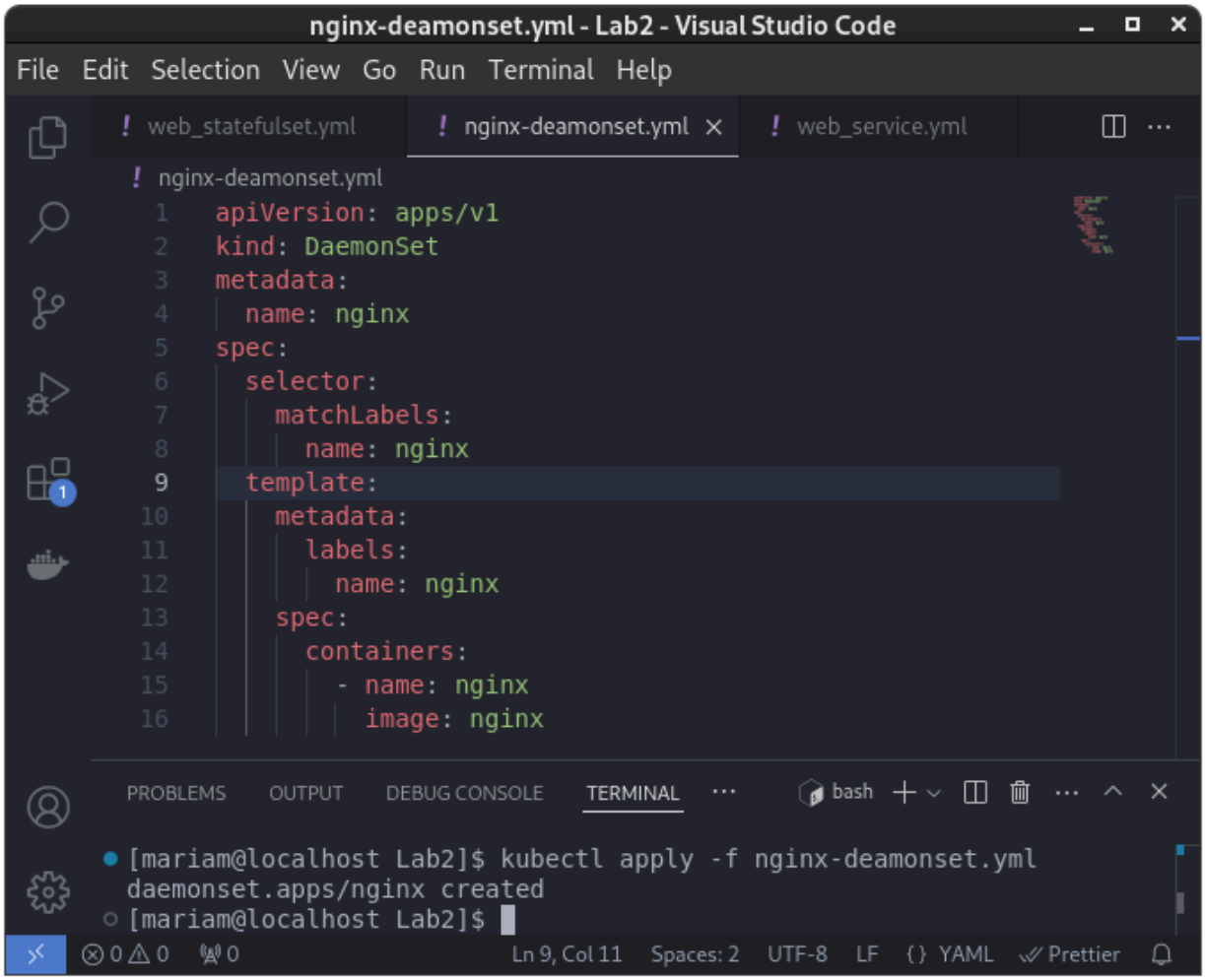




2. How many DaemonSets are created in the cluster in all namespaces?
=> 0



3. Create a DaemonSet named “nginx” with image “nginx”.



4. How many pods have been created within the nginx DaemonSet and why?

```
[mariam@localhost Lab2]$ kubectl get daemonset nginx
NAME    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
  AGE
nginx   1         1         1       1            1           <none>
  83s
[mariam@localhost Lab2]$
```

```
[mariam@localhost Lab2]$ kubectl get daemonset nginx
NAME    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
  AGE
nginx   1         1         1       1            1           <none>
  83s
[mariam@localhost Lab2]$ kubectl get nodes
NAME       STATUS   ROLES           AGE   VERSION
minikube   Ready    control-plane   24h   v1.32.0
[mariam@localhost Lab2]$
```

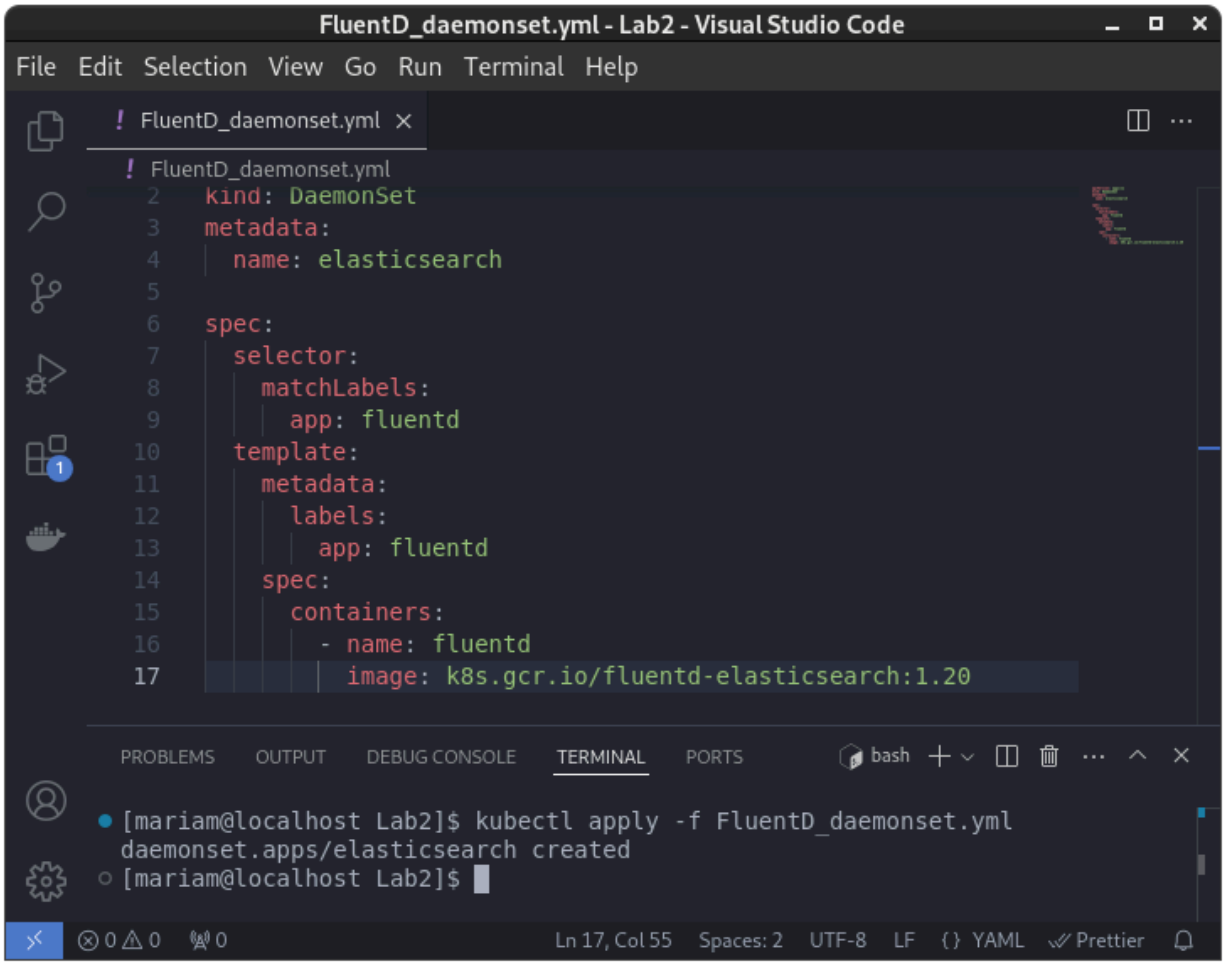
5. Create a pod named “ingot”

ingot_pod.yml - Lab2 - Visual Studio Code

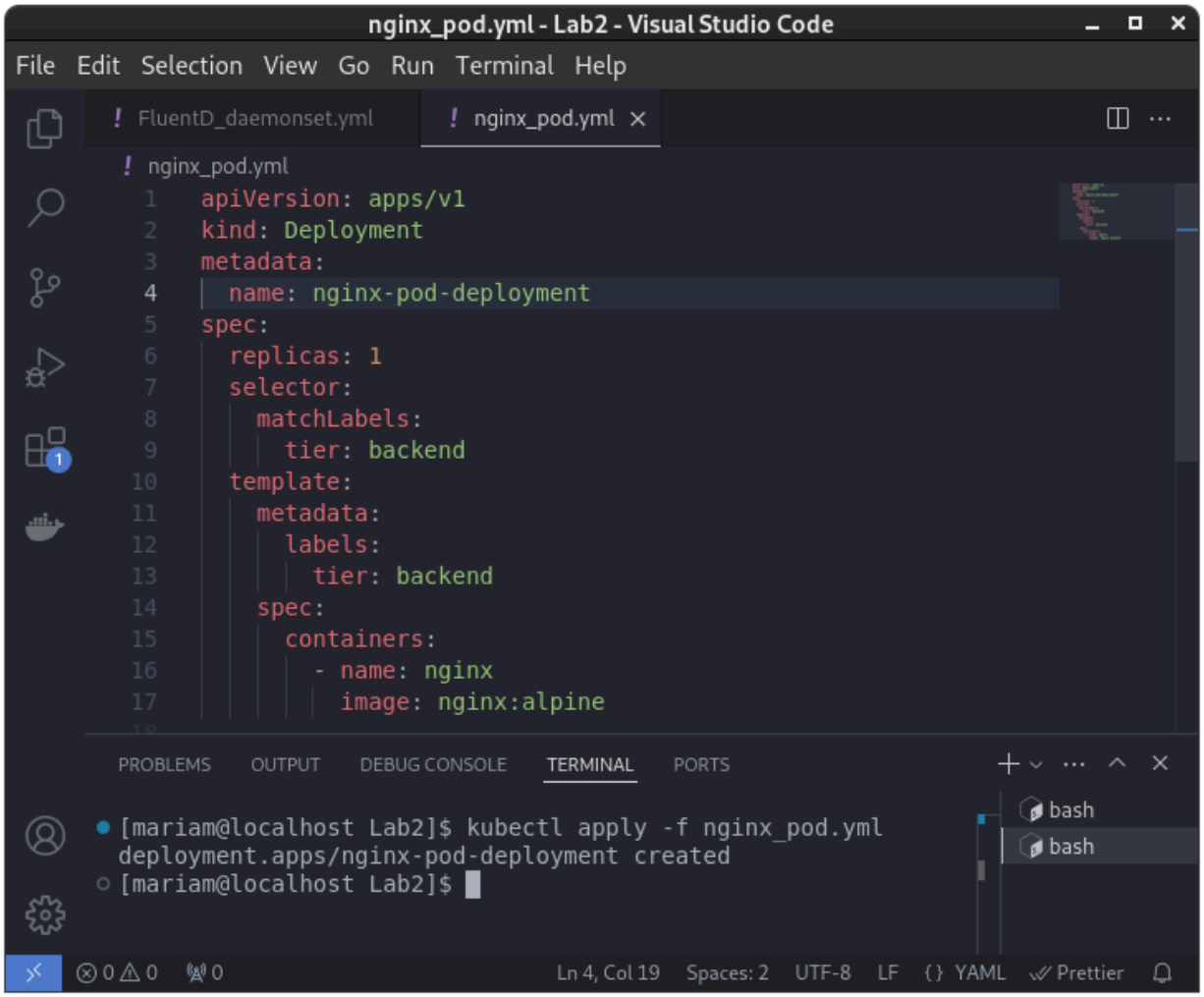
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: ingot
spec:
  containers:
    - name: ingot-pod
      image: apline
      command: ["sleep","3600"]
```

```
[mariam@localhost Lab2]$ kubectl apply -f ingot_pod.yml
pod/ingot created
[mariam@localhost Lab2]$
```

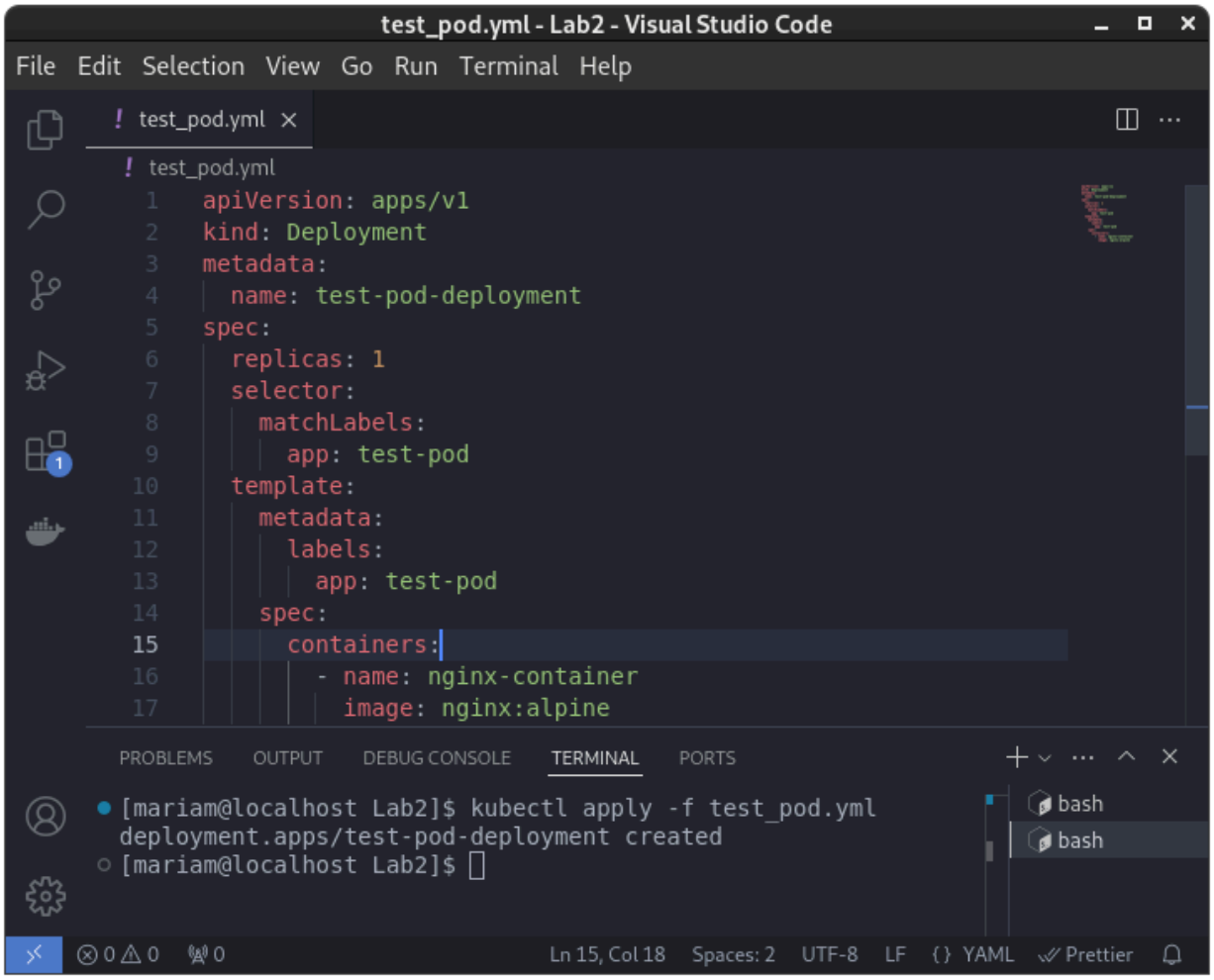
6. Deploy a DaemonSet for FluentD Logging. Use the given specifications.
➔ Name: elasticsearch ➔ Image: k8s.gcr.io/fluentd-elasticsearch:1.20



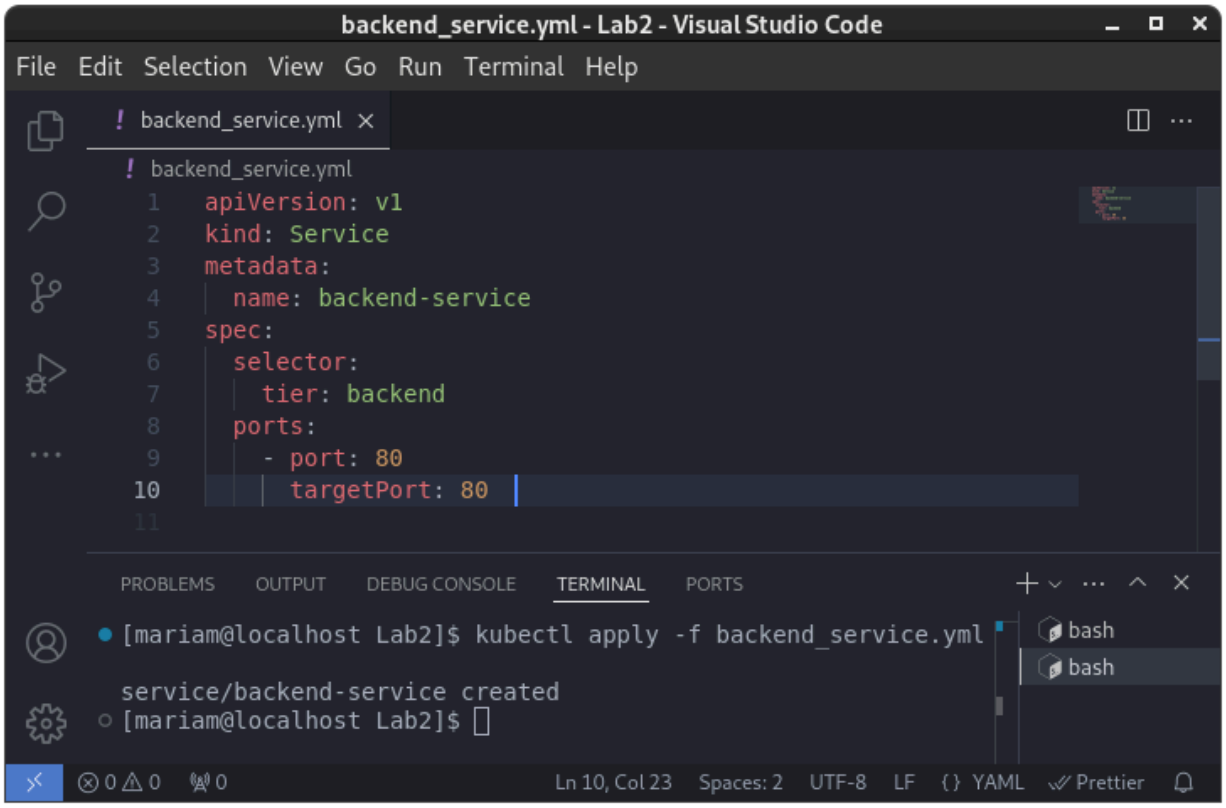
7. Deploy a pod named nginx-pod using the nginx:alpine image with the labels set to: tier=backend.



8. Deploy a test pod using the nginx:alpine image.



9. Create a service backend-service to expose the backend application within the cluster on port 80.

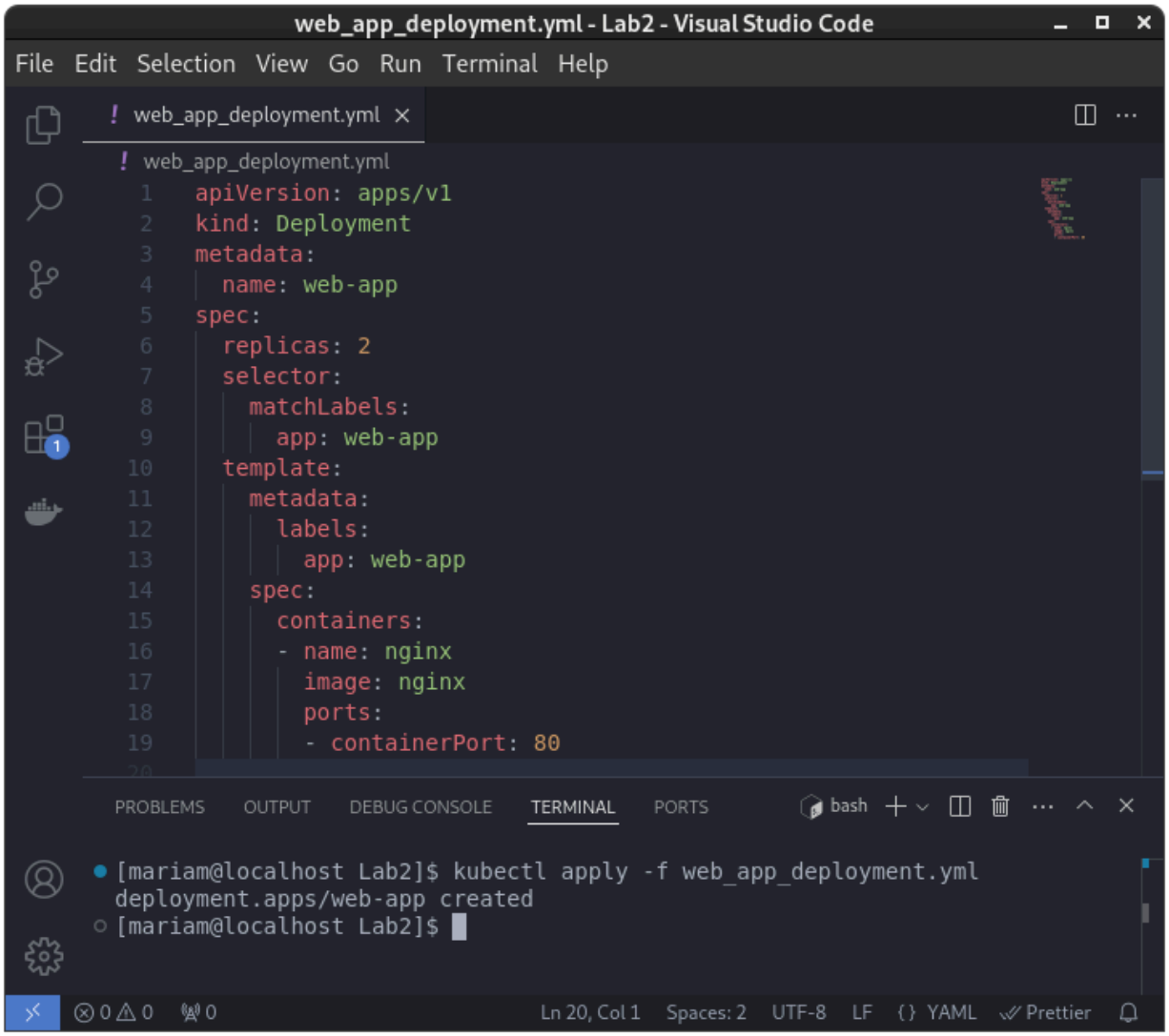


10. Try to curl the backend-service from the test pod. What is the response?

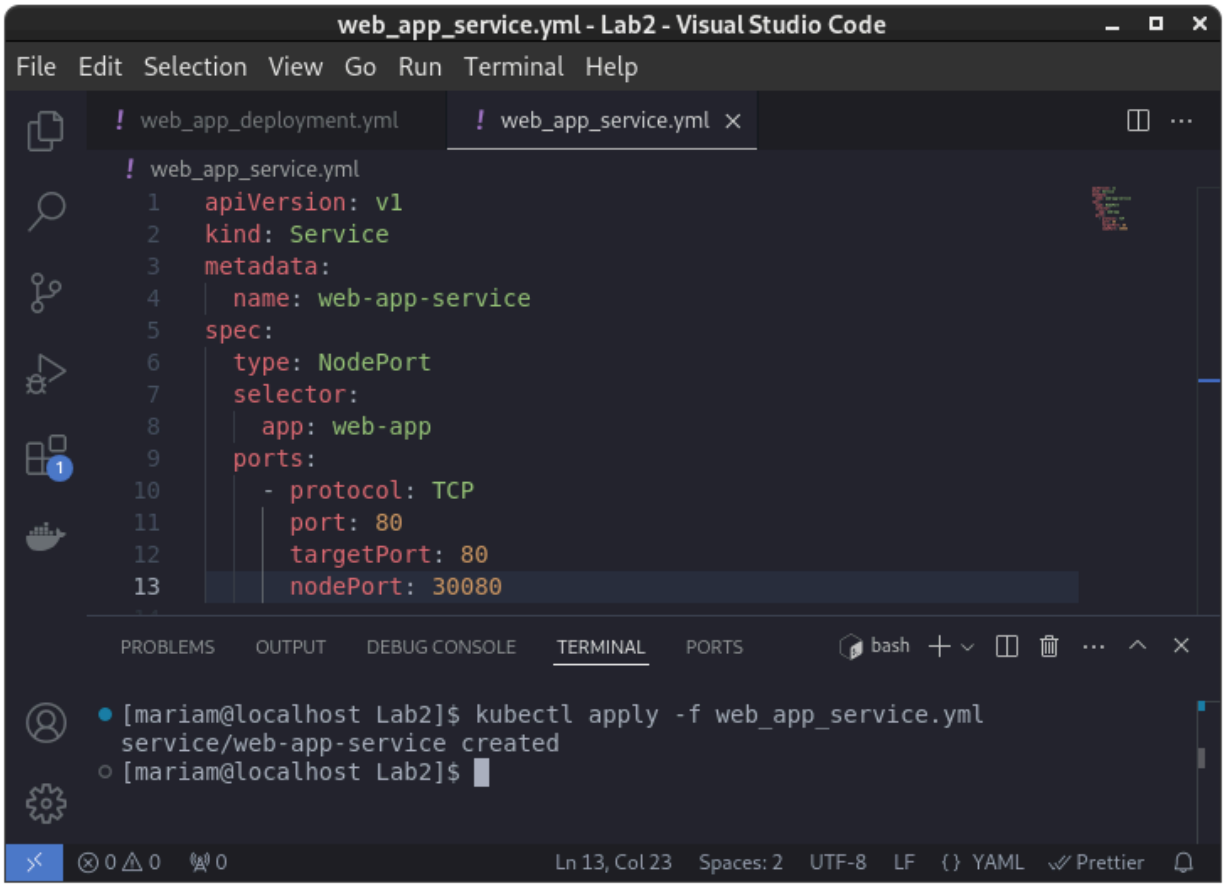
```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    kubectl

[mariam@localhost Lab2]$ kubectl exec -it test-pod-deployment-69fffdc958-5l
vss -- sh
/ # curl backend-service
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
/ #
```

The response is an html page from nginx, which means the backend-service is correctly routing traffic to the nginx server that’s running in the cluster.

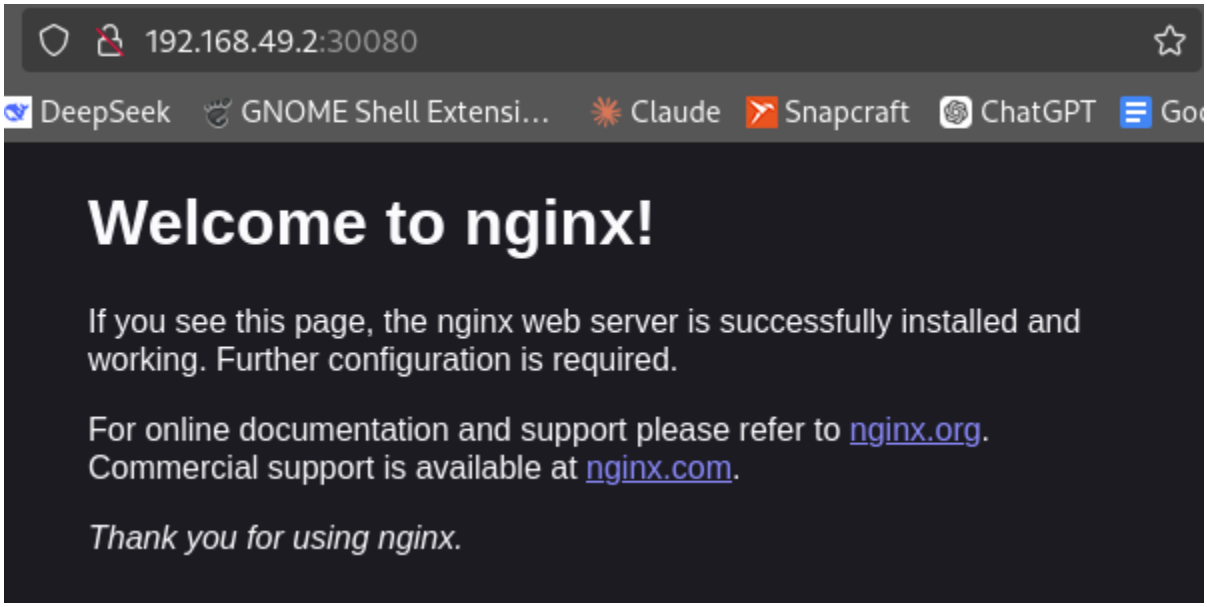
11. Create a deployment named web-app using the image nginx with 2 replicas



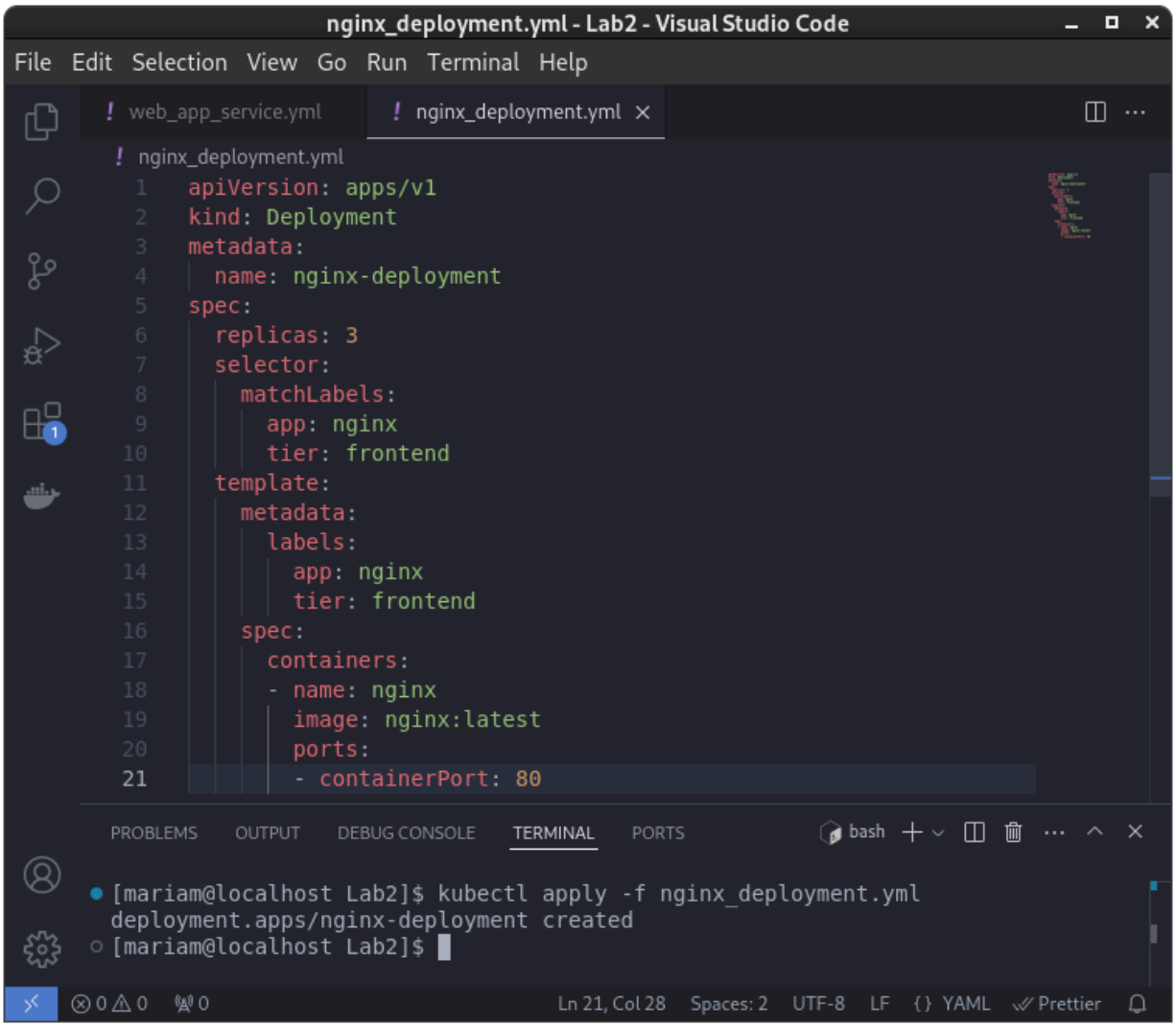
12. Expose the web-app as service web-app-service application on port 80 and nodeport 30082 on the nodes on the cluster



13.Access the web app from the node



14.Create a deployment nginx with pod labels
➔ app:nginx
➔ tier:frontend
and set-based selectors on replicasets that allow filtering objects based on specific
conditions. Given the valid operators In, Exists.



15.When can we use the Loadbalancer service?

- This service in Kubernetes is used to expose applications externally, typically in cloud environments. It automatically provisions an external load balancer to distribute traffic to the Pods in cluster.