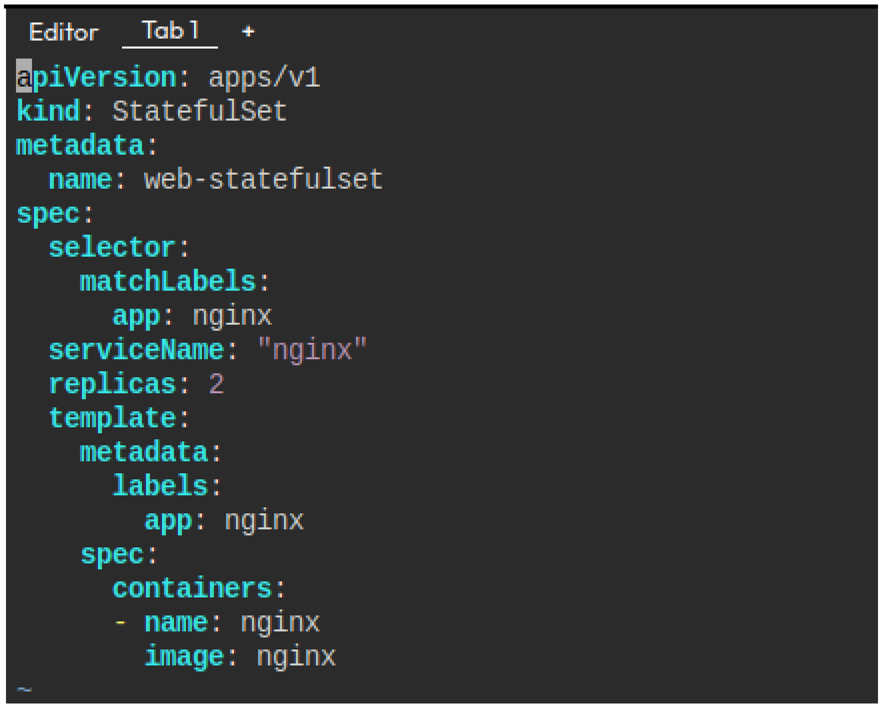
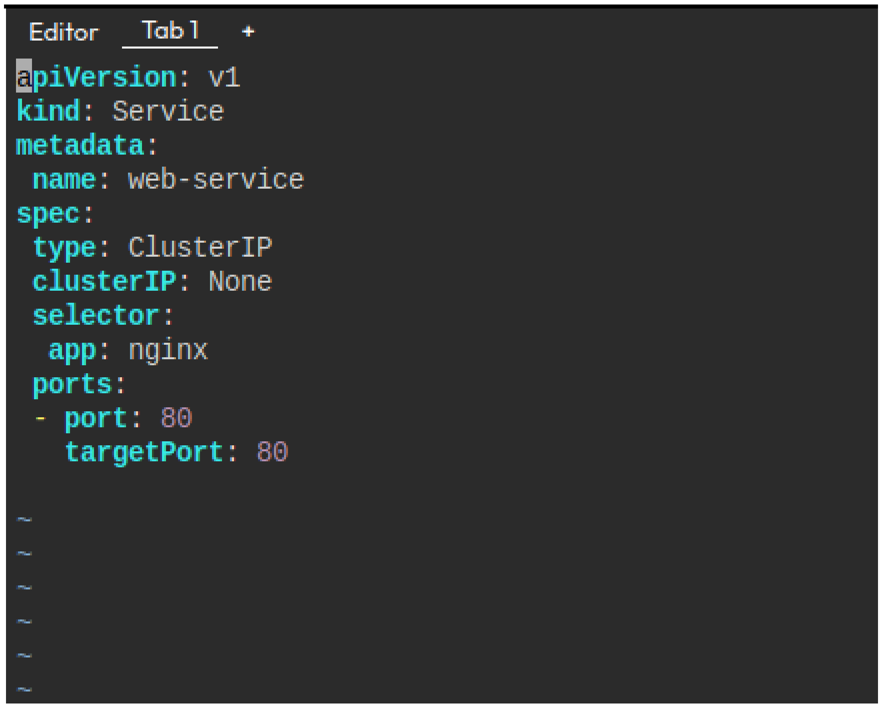
Q1: create a stateful set called with the named: web-statefulset and create a headless service called web-service

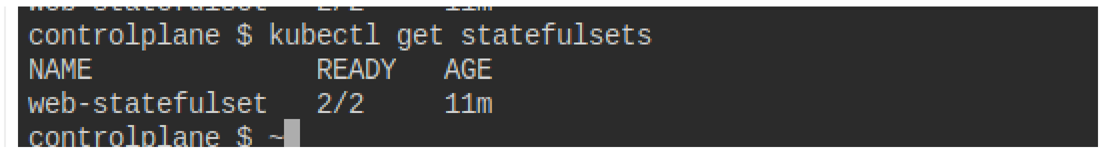


```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web-statefulset
spec:
  selector:
    matchLabels:
      app: nginx
  serviceName: "nginx"
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
~
```

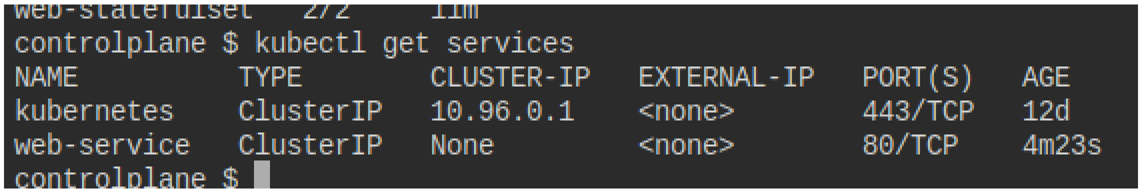
Creating the service



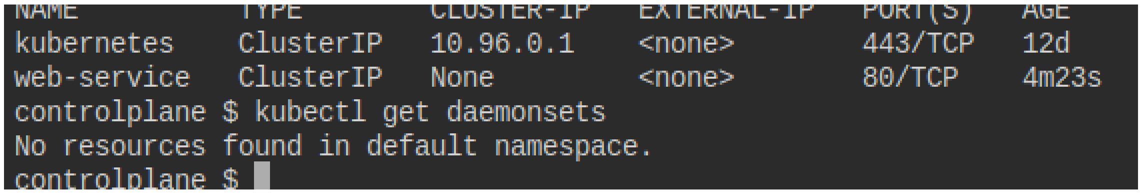
```
apiVersion: v1
kind: Service
metadata:
 name: web-service
spec:
 type: ClusterIP
 clusterIP: None
 selector:
  app: nginx
 ports:
 - port: 80
   targetPort: 80

~
~
~
~
~
~
```



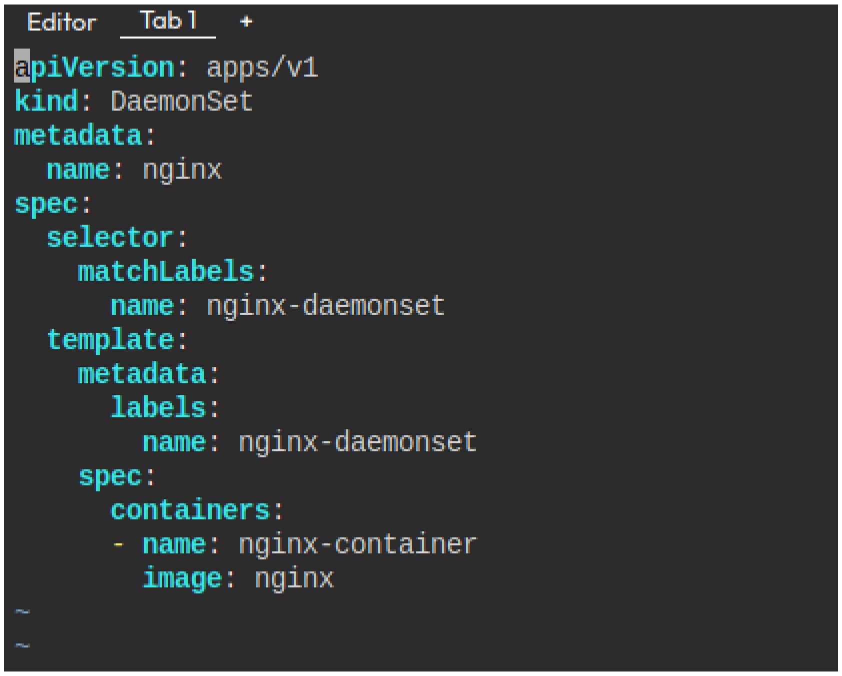
```
controlplane $ kubectl get statefulsets
NAME                READY   AGE
web-statefulset     2/2     11m
controlplane $ ~
```


```
web-statefulset   2/2     11m
controlplane $ kubectl get services
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes    ClusterIP   10.96.0.1    <none>        443/TCP   12d
web-service   ClusterIP   None         <none>        80/TCP    4m23s
controlplane $ 
```

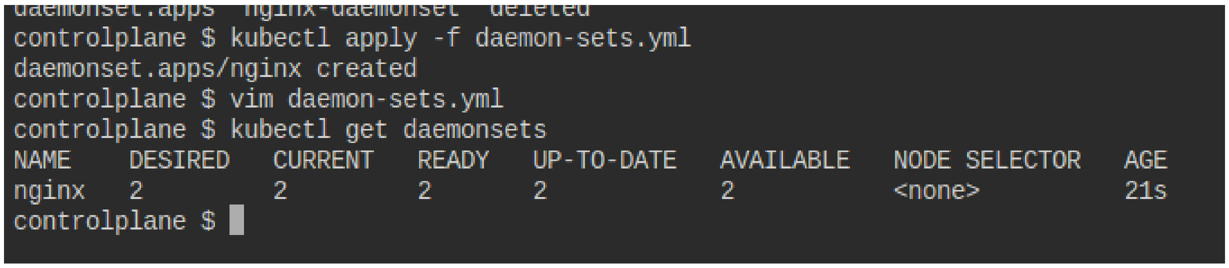
Q2: get daemonsets


```
NAME          TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
kubernetes    ClusterIP   10.96.0.1    <none>        443/TCP   12d
web-service   ClusterIP   None         <none>        80/TCP    4m23s
controlplane $ kubectl get daemonsets
No resources found in default namespace.
controlplane $ 
```

Q3: create a daemon set called nginx

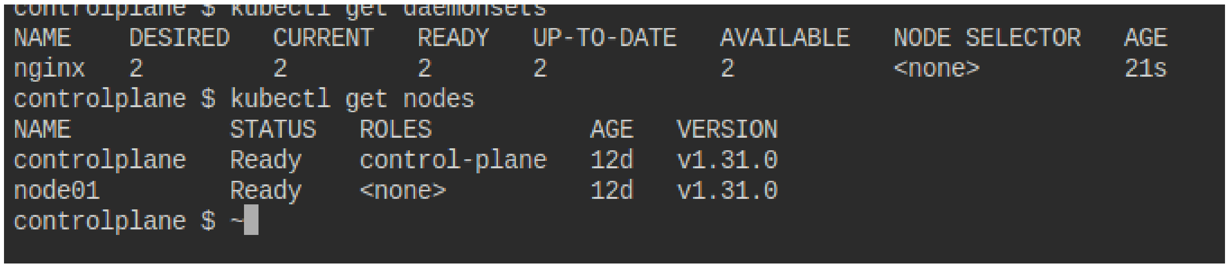


```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: nginx
spec:
  selector:
    matchLabels:
      name: nginx-daemonset
  template:
    metadata:
      labels:
        name: nginx-daemonset
    spec:
      containers:
      - name: nginx-container
        image: nginx
~
~
```


```
daemonset.apps "nginx-daemonset" deleted
controlplane $ kubectl apply -f daemon-sets.yml
daemonset.apps/nginx created
controlplane $ vim daemon-sets.yml
controlplane $ kubectl get daemonsets
NAME    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
nginx   2         2         2       2            2           <none>          21s
controlplane $ 
```

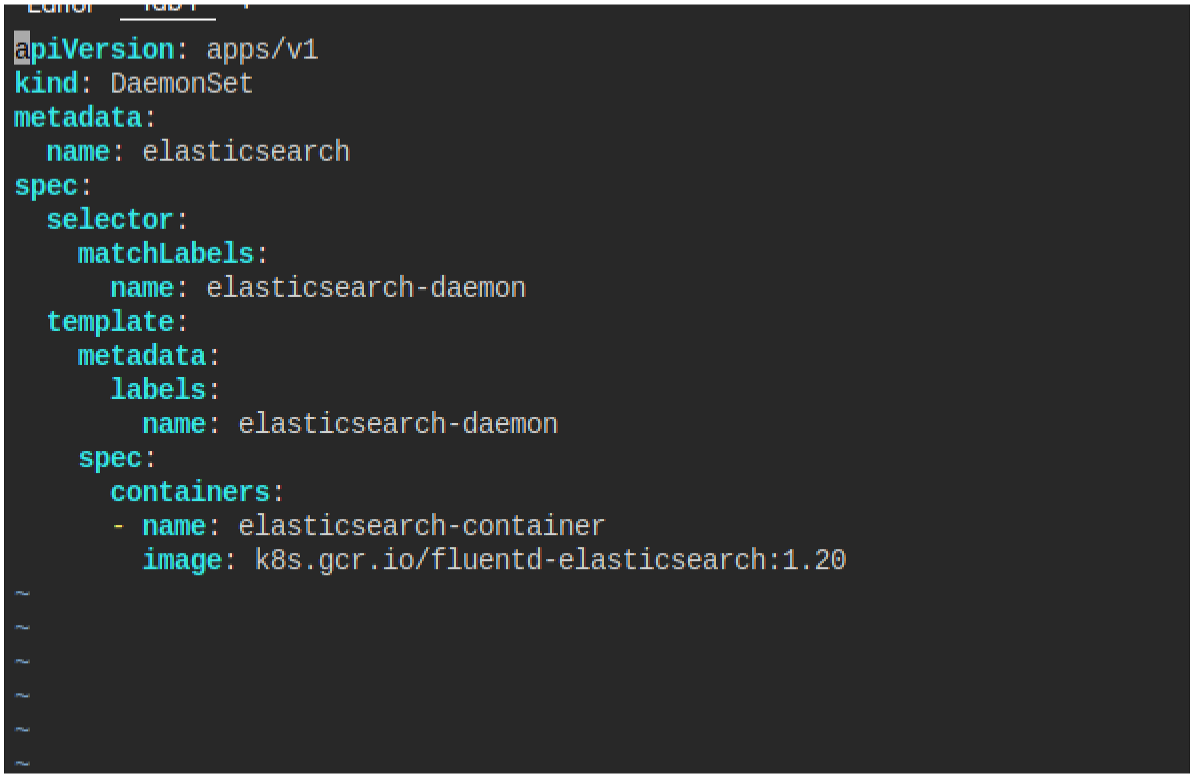
Q4: how many pods have been created with the daemonset

2 Pods becuase there are two nodes without restrictions the controlplane as well as one worker node



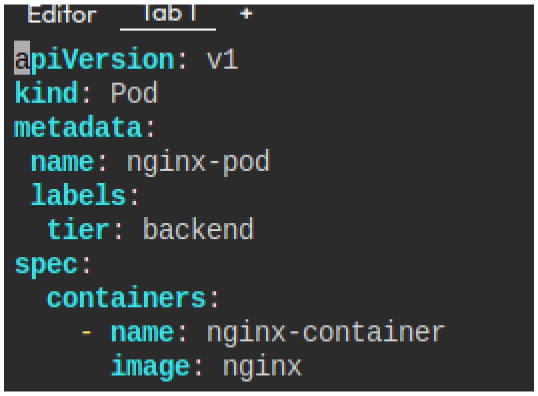
```
controlplane $ kubectl get daemonsets
NAME    DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
nginx   2         2         2       2            2           <none>          21s
controlplane $ kubectl get nodes
NAME           STATUS   ROLES           AGE   VERSION
controlplane   Ready    control-plane   12d   v1.31.0
node01         Ready    <none>          12d   v1.31.0
controlplane $ ~
```

Q6: create a logging daemonset called elasticsearch with image fluentd



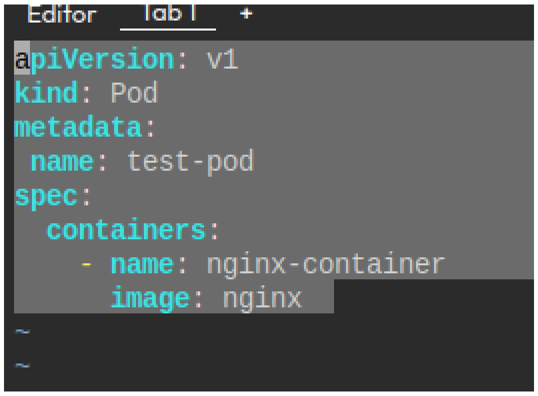
```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: elasticsearch
spec:
  selector:
    matchLabels:
      name: elasticsearch-daemon
  template:
    metadata:
      labels:
        name: elasticsearch-daemon
    spec:
      containers:
      - name: elasticsearch-container
        image: k8s.gcr.io/fluentd-elasticsearch:1.20
~
~
~
~
~
~
```

Q7: create an nginx-pod with tier=backend



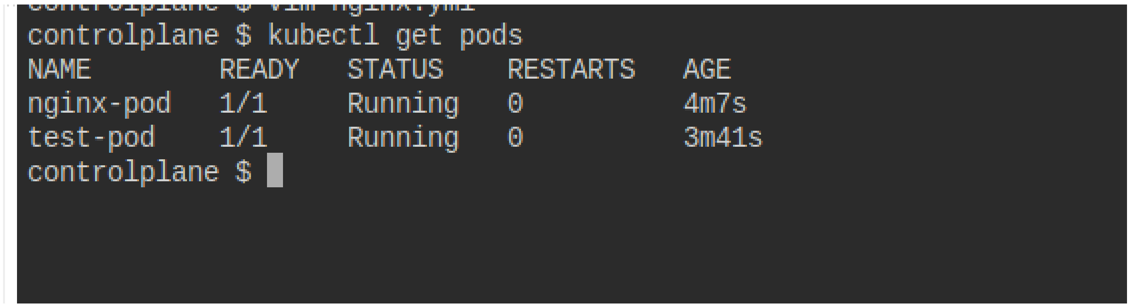
```
apiVersion: v1
kind: Pod
metadata:
 name: nginx-pod
 labels:
  tier: backend
spec:
  containers:
    - name: nginx-container
      image: nginx
```

Q8: create a testpod named test-pod



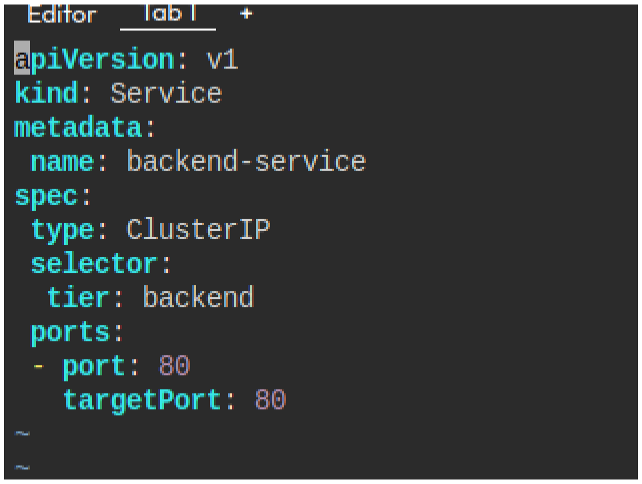
```
apiVersion: v1
kind: Pod
metadata:
 name: test-pod
spec:
  containers:
    - name: nginx-container
      image: nginx
~
~
```

Checking that both are running

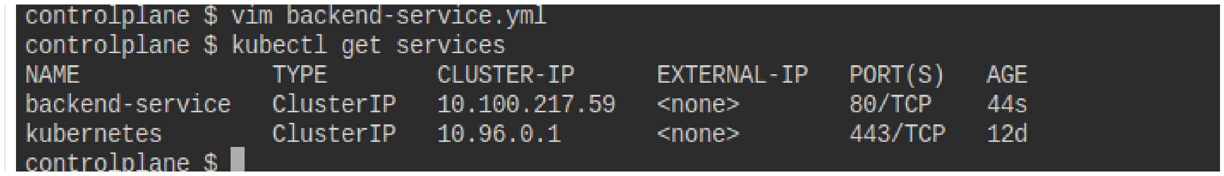


```
controlplane $ kubectl get pods
NAME        READY   STATUS    RESTARTS   AGE
nginx-pod   1/1     Running   0          4m7s
test-pod    1/1     Running   0          3m41s
controlplane $
```

Q9: creating backend service

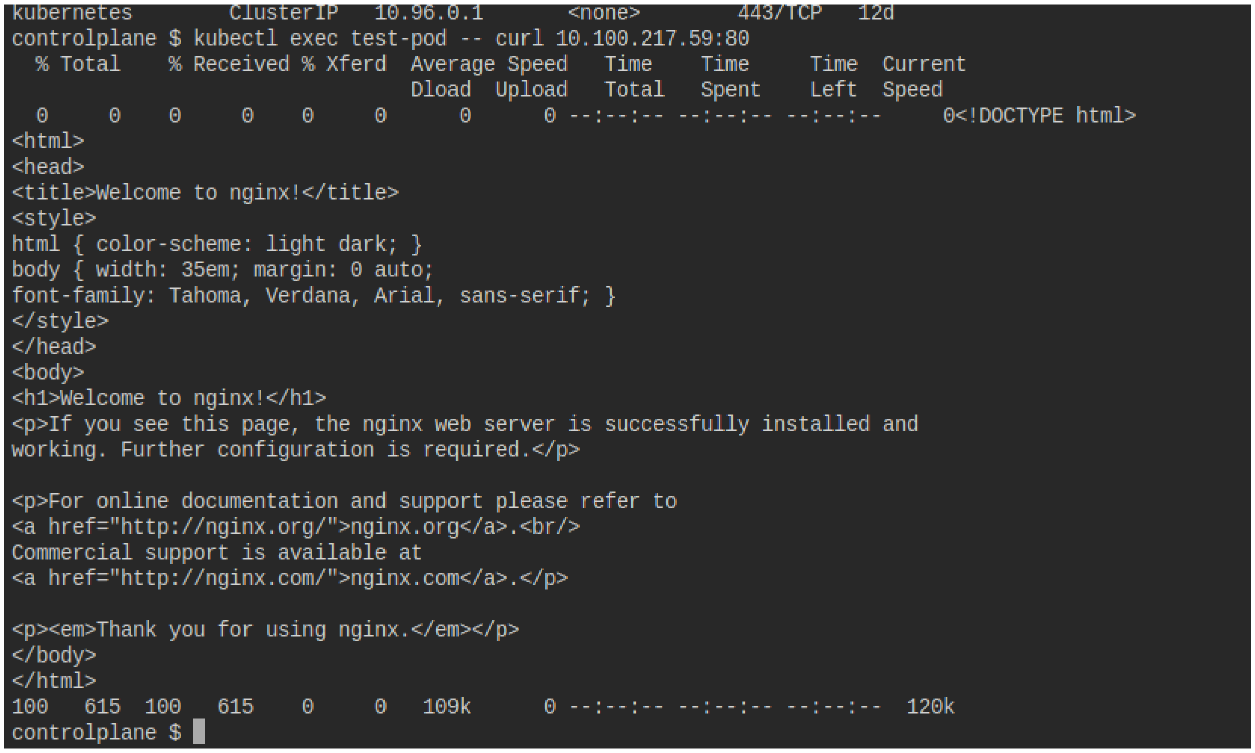


```
apiVersion: v1
kind: Service
metadata:
 name: backend-service
spec:
 type: ClusterIP
 selector:
  tier: backend
 ports:
 - port: 80
   targetPort: 80
~
~
```



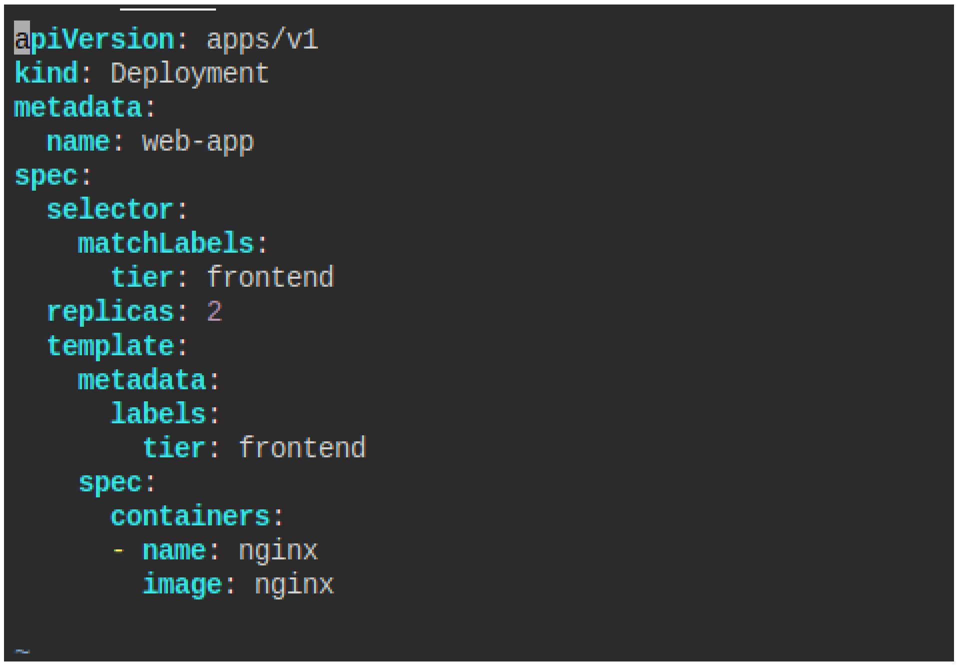
```
controlplane $ vim backend-service.yml
controlplane $ kubectl get services
NAME              TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)   AGE
backend-service   ClusterIP   10.100.217.59   <none>        80/TCP    44s
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP   12d
controlplane $
```

Q10: curl the service from inside the testpod

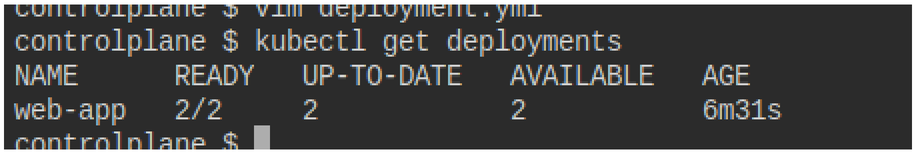


```
kubernetes        ClusterIP   10.96.0.1       <none>        443/TCP   12d
controlplane $ kubectl exec test-pod -- curl 10.100.217.59:80
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0<!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>
100   615  100   615    0     0   109k      0 --:--:-- --:--:-- --:--:--  120k
controlplane $
```

Q11: creating a deployment called web-app

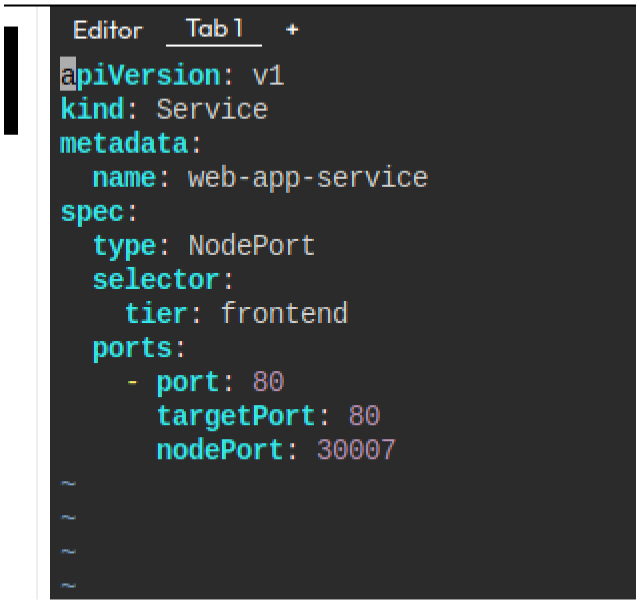


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  selector:
    matchLabels:
      tier: frontend
  replicas: 2
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: nginx

~
```

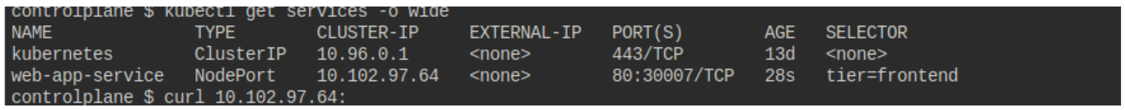


```
controlplane $ vim deployment.yml
controlplane $ kubectl get deployments
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
web-app   2/2     2            2           6m31s
controlplane $
```

Q12: creating the service and applying

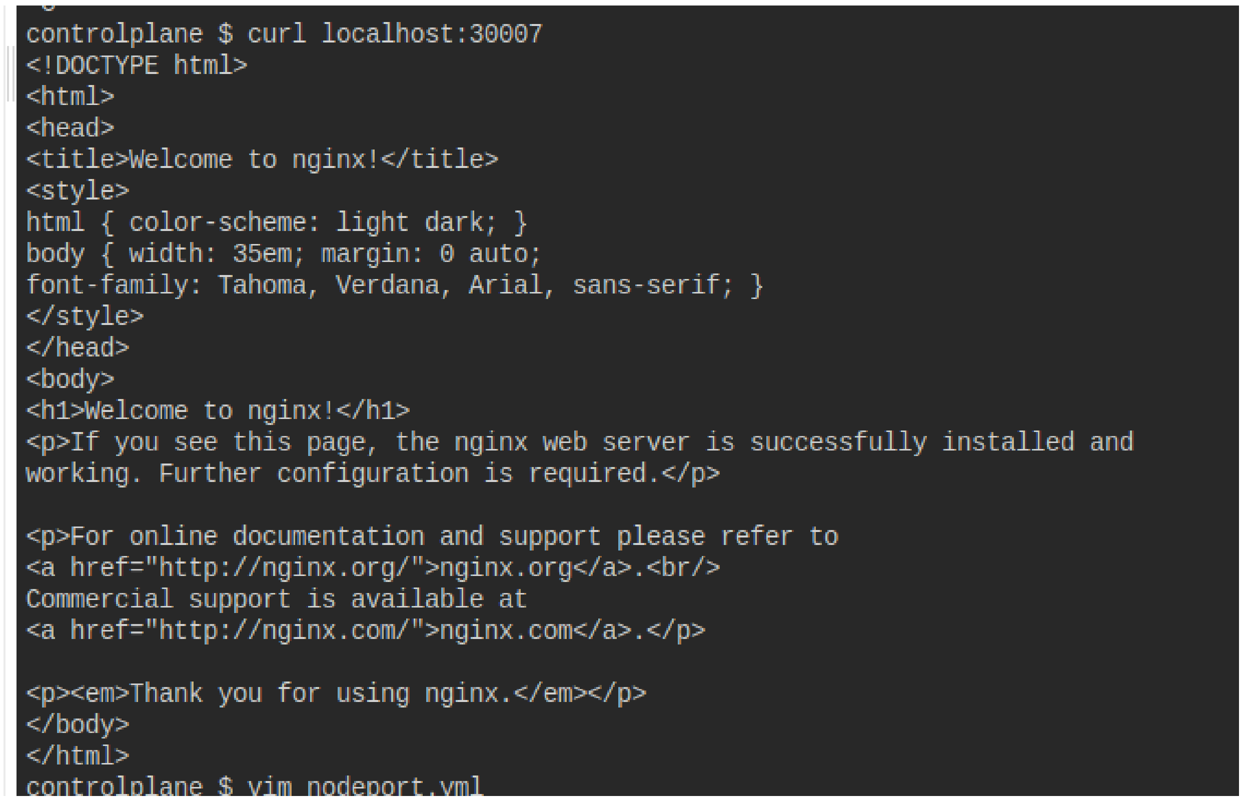


```
apiVersion: v1
kind: Service
metadata:
  name: web-app-service
spec:
  type: NodePort
  selector:
    tier: frontend
  ports:
    - port: 80
      targetPort: 80
      nodePort: 30007
~
~
~
~
```



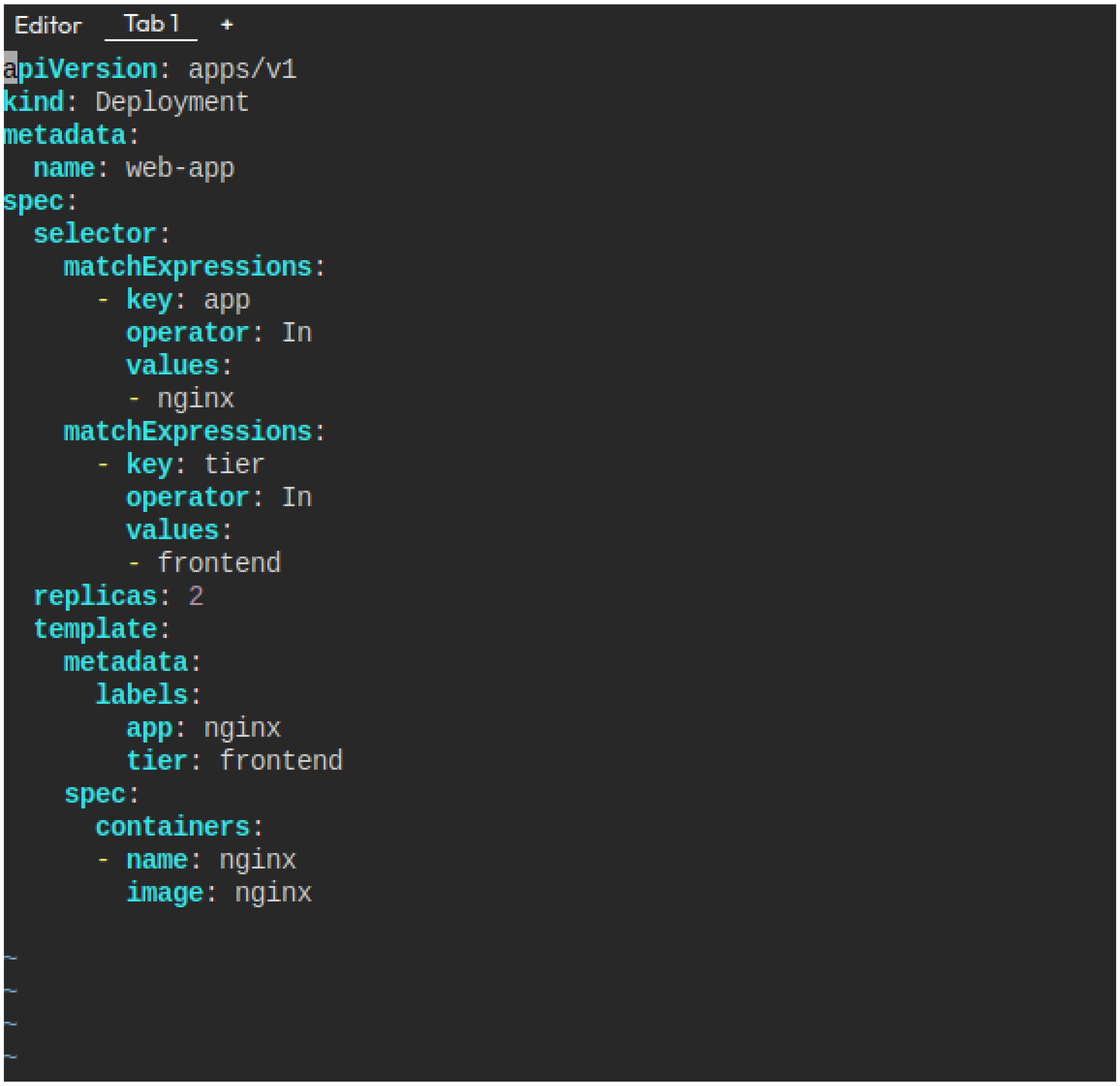
```
controlplane $ kubectl get services -o wide
NAME              TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE   SELECTOR
kubernetes        ClusterIP   10.96.0.1      <none>        443/TCP        13d   <none>
web-app-service   NodePort    10.102.97.64   <none>        80:30007/TCP   28s   tier=frontend
controlplane $ curl 10.102.97.64:
```

Q13: accessing from the control node



```
controlplane $ curl localhost:30007
<!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>
controlplane $ vim nodeport.yml
```

Q14: create a set based selector on a deployment



```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-app
spec:
  selector:
    matchExpressions:
      - key: app
        operator: In
        values:
        - nginx
    matchExpressions:
      - key: tier
        operator: In
        values:
        - frontend
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: nginx

~
~
~
~
```

Q15: When should we use a loadbalancer
When you want ot expose the current cluster to outside traffic which can be done using an external service but it can be initiated using the kubernetes built in services