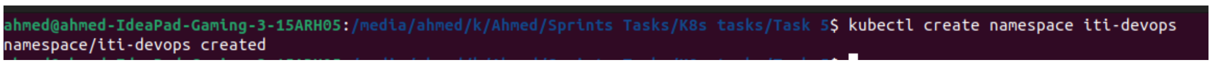
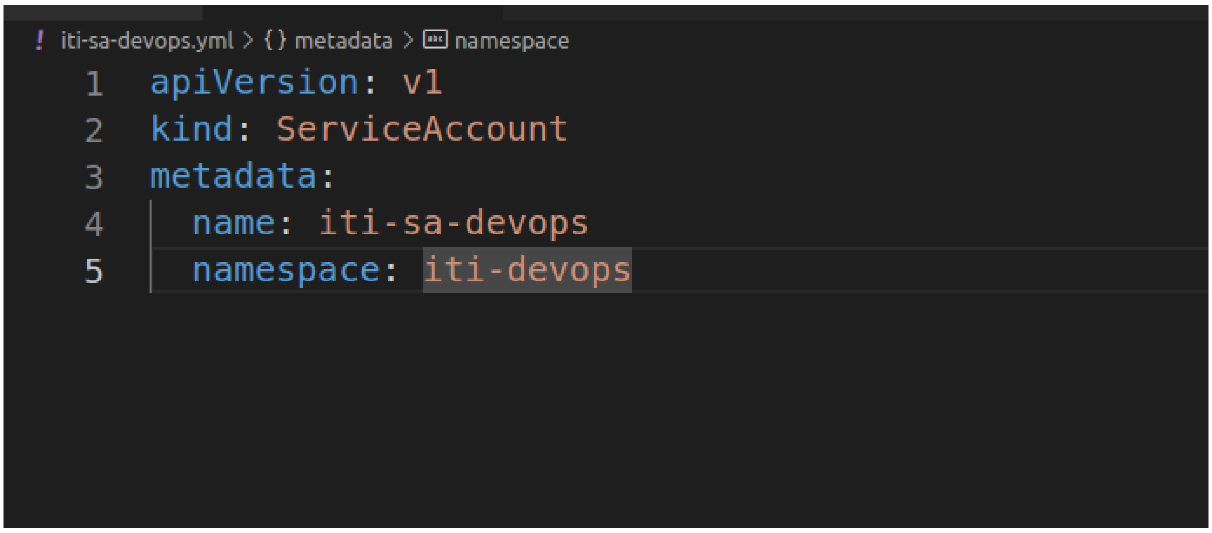
1- create a namespace iti-devops

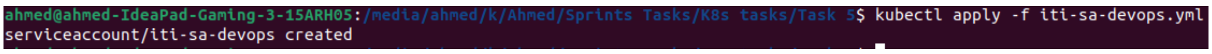


```
ahmed@ahmed-IdeaPad-Gaming-3-15ARH05:/media/ahmed/k/Ahmed/Sprints Tasks/K8s tasks/Task 5$ kubectl create namespace iti-devops
namespace/iti-devops created
```

2- create a service account iti-sa-devops under the same namespace

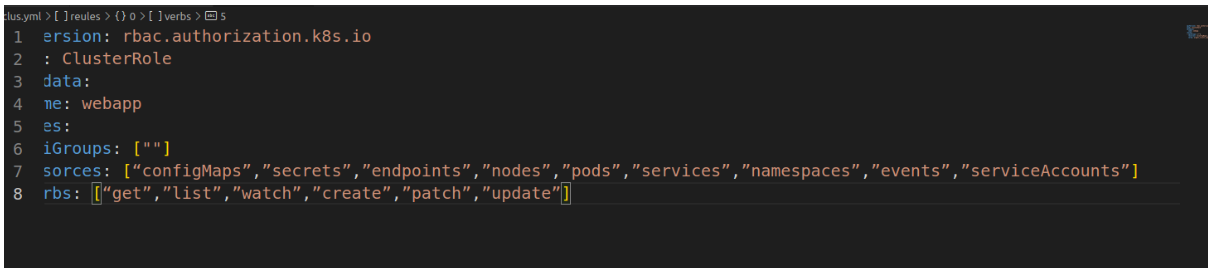


```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: iti-sa-devops
  namespace: iti-devops
```

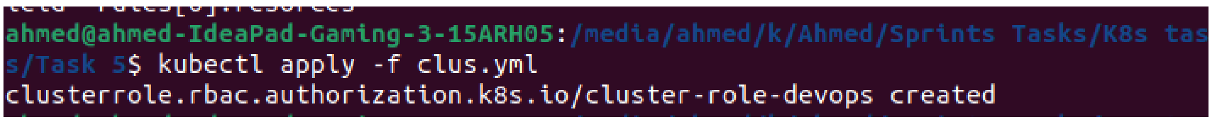


```
ahmed@ahmed-IdeaPad-Gaming-3-15ARH05:/media/ahmed/k/Ahmed/Sprints Tasks/K8s tasks/Task 5$ kubectl apply -f iti-sa-devops.yml
serviceaccount/iti-sa-devops created
```

3. create a clusteRole which should be named as cluster-role-devops to grant permissions

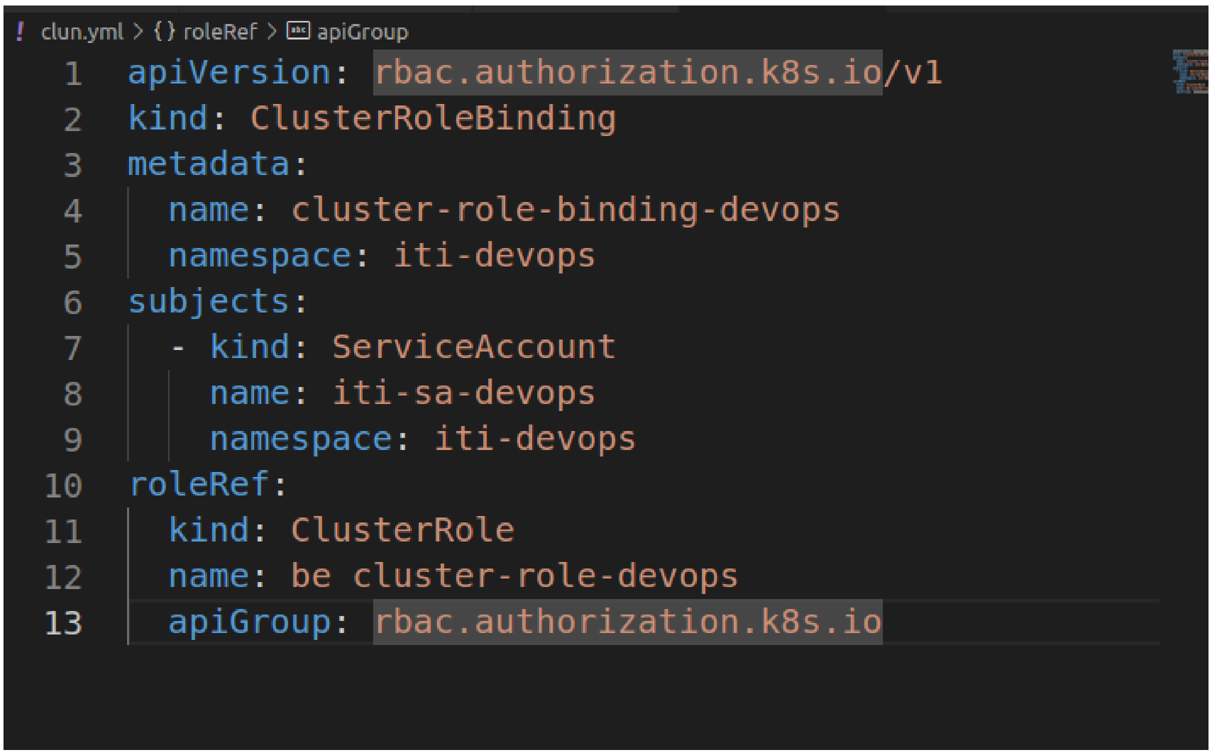


```
clus.yml > [ ] reules > { } 0 > [ ] verbs > 5
ersion: rbac.authorization.k8s.io
: ClusterRole
data:
me: webapp
es:
iGroups: [""]
sorces: [“configMaps”,”secrets”,”endpoints”,”nodes”,”pods”,”services”,”namespaces”,”events”,”serviceAccounts”]
rbs: [“get”,”list”,”watch”,”create”,”patch”,”update”]
```

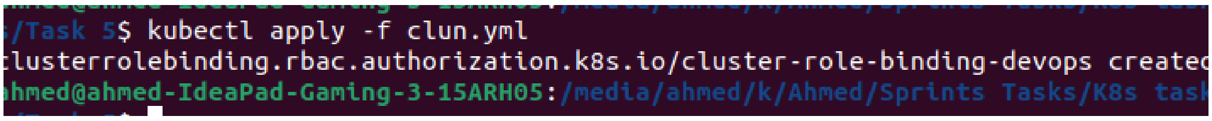


```
ahmed@ahmed-IdeaPad-Gaming-3-15ARH05:/media/ahmed/k/Ahmed/Sprints Tasks/K8s tas
s/Task 5$ kubectl apply -f clus.yml
clusterrole.rbac.authorization.k8s.io/cluster-role-devops created
```

4. create a ClusterRoleBinding



```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-role-binding-devops
  namespace: iti-devops
subjects:
  - kind: ServiceAccount
    name: iti-sa-devops
    namespace: iti-devops
roleRef:
  kind: ClusterRole
  name: be cluster-role-devops
  apiGroup: rbac.authorization.k8s.io
```

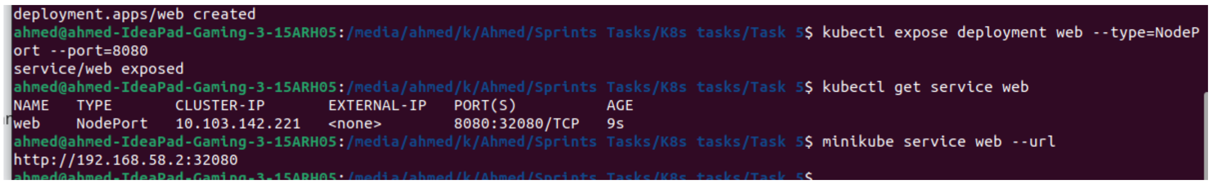


```
/Task 5$ kubectl apply -f clun.yml
clusterrolebinding.rbac.authorization.k8s.io/cluster-role-binding-devops created
ahmed@ahmed-IdeaPad-Gaming-3-15ARH05:/media/ahmed/k/Ahmed/Sprints Tasks/K8s task
```

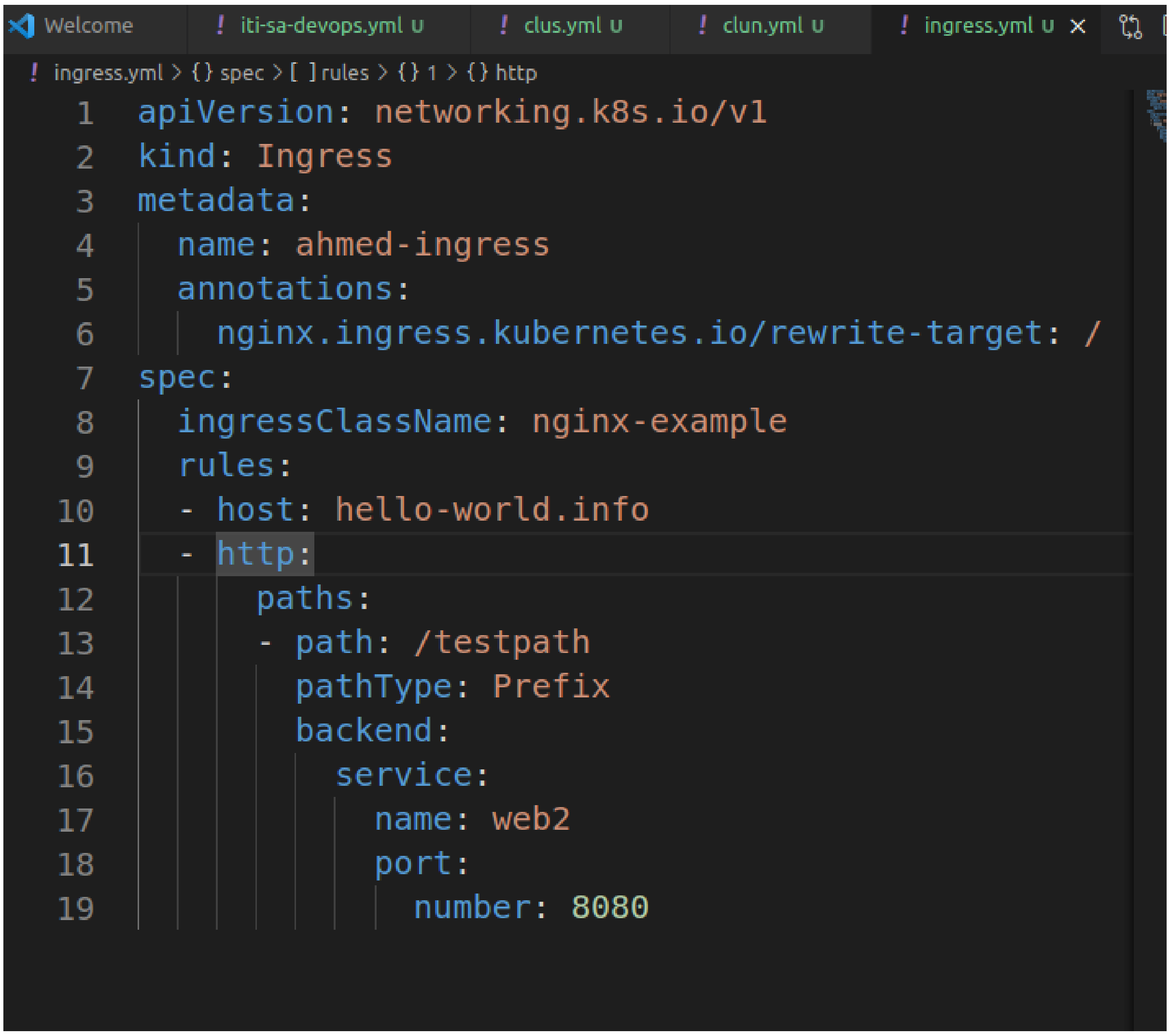
5- What is the difference between statefulSets and deployments?

A StatefulSet keeps a sticky identity for each of its Pods, unlike a Deployment. Despite being made from the identical specifications, these pods cannot be swapped out since they each have a persistent identifier that they keep up through any schedule changes.

6- Set up Ingress on Minikube with the NGINX Ingress Controller



```
deployment.apps/web created
ahmed@ahmed-IdeaPad-Gaming-3-15ARH05:/media/ahmed/k/Ahmed/Sprints Tasks/K8s tasks/Task 5$ kubectl expose deployment web --type=NodeP
ort --port=8080
service/web exposed
ahmed@ahmed-IdeaPad-Gaming-3-15ARH05:/media/ahmed/k/Ahmed/Sprints Tasks/K8s tasks/Task 5$ kubectl get service web
NAME   TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
web    NodePort   10.103.142.221   <none>        8080:32080/TCP   9s
ahmed@ahmed-IdeaPad-Gaming-3-15ARH05:/media/ahmed/k/Ahmed/Sprints Tasks/K8s tasks/Task 5$ minikube service web --url
http://192.168.58.2:32080
ahmed@ahmed-IdeaPad-Gaming-3-15ARH05:/media/ahmed/k/Ahmed/Sprints Tasks/K8s tasks/Task 5$
```



```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ahmed-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx-example
  rules:
  - host: hello-world.info
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: web2
            port:
              number: 8080
```