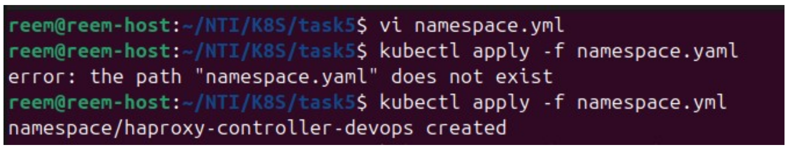
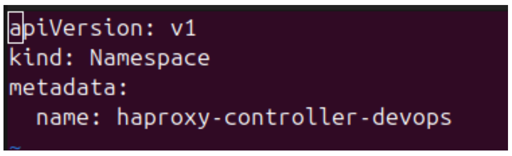
Deploy HaProxy

1- Create a namespace haproxy-controller-devops.

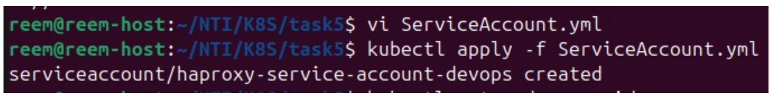


```
reem@reem-host:~/NTI/K8S/task5$ vi namespace.yml
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f namespace.yaml
error: the path "namespace.yaml" does not exist
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f namespace.yml
namespace/haproxy-controller-devops created
```

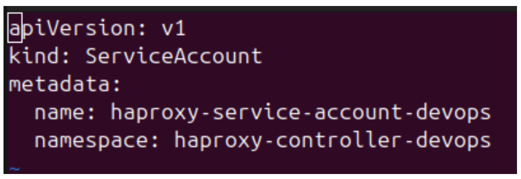


```
apiVersion: v1
kind: Namespace
metadata:
  name: haproxy-controller-devops
~
```

2- Create a ServiceAccount haproxy-service-account-devops under the same namespace.

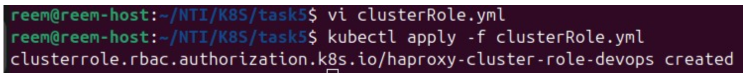


```
reem@reem-host:~/NTI/K8S/task5$ vi ServiceAccount.yml
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f ServiceAccount.yml
serviceaccount/haproxy-service-account-devops created
```

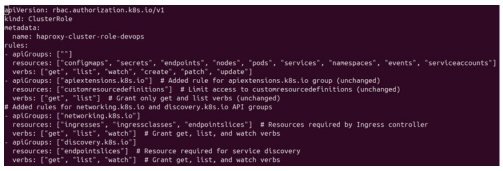


```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: haproxy-service-account-devops
  namespace: haproxy-controller-devops
~
```

3- Create a ClusterRole which should be named as haproxy-cluster-role-devops, to grant permissions "get", "list", "watch", "create", "patch", "update" to "Configmaps",”secrets”,"endpoints","nodes","pods","services", "namespaces","events","serviceaccounts".

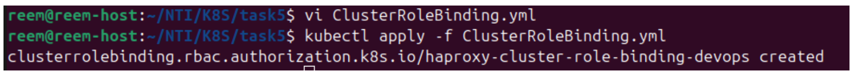


```
reem@reem-host:~/NTI/K8S/task5$ vi clusterRole.yml
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f clusterRole.yml
clusterrole.rbac.authorization.k8s.io/haproxy-cluster-role-devops created
```

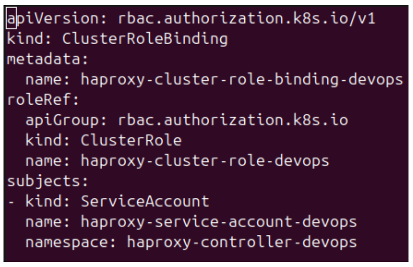


```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: haproxy-cluster-role-devops
rules:
- apiGroups: [""]
  resources: ["configmaps", "secrets", "endpoints", "nodes", "pods", "services", "namespaces", "events", "serviceaccounts"]
  verbs: ["get", "list", "watch", "create", "patch", "update"]
- apiGroups: ["apiextensions.k8s.io"]  # Added rule for apiextensions.k8s.io group (unchanged)
  resources: ["customresourcedefinitions"]  # Limit access to customresourcedefinitions (unchanged)
  verbs: ["get", "list"]  # Grant only get and list verbs (unchanged)
# Added rules for networking.k8s.io and discovery.k8s.io API groups
- apiGroups: ["networking.k8s.io"]
  resources: ["ingresses", "ingressclasses", "endpointslices"]  # Resources required by Ingress controller
  verbs: ["get", "list", "watch"]  # Grant get, list, and watch verbs
- apiGroups: ["discovery.k8s.io"]
  resources: ["endpointslices"]  # Resource required for service discovery
  verbs: ["get", "list", "watch"]  # Grant get, list, and watch verbs
```

4- Create a ClusterRoleBinding which should be named as haproxy-cluster-role-binding-devops under the same namespace. Define roleRef apiGroup should be rbac.authorization.k8s.io, kind should be ClusterRole, name should be haproxy-cluster-role-devops and subjects kind should be ServiceAccount, name should be haproxy-service-account-devops and namespace should be haproxy-controller-devops.



```
reem@reem-host:~/NTI/K8S/task5$ vi ClusterRoleBinding.yml
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f ClusterRoleBinding.yml
clusterrolebinding.rbac.authorization.k8s.io/haproxy-cluster-role-binding-devops created
```

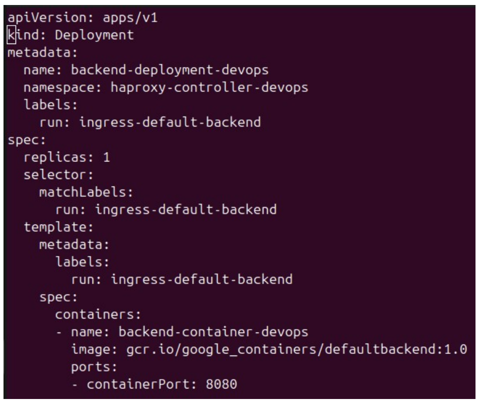


```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: haproxy-cluster-role-binding-devops
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: haproxy-cluster-role-devops
subjects:
- kind: ServiceAccount
  name: haproxy-service-account-devops
  namespace: haproxy-controller-devops
```

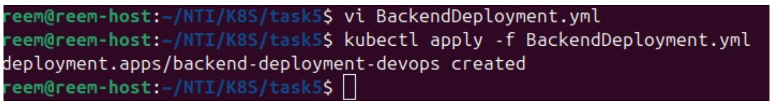
5- Create a backend deployment which should be named as backend-deployment-devops under the same namespace, labels run should be ingress-default-backend under metadata.

Configure spec as replica should be 1, selector's matchLabels run should be ingress-default-backend. Template's labels run under metadata should be

ingress-default-backend. The container should named as backend-container-devops, use image gcr.io/google_containers/defaultbackend:1.0 (use exact name of image as mentioned) and its containerPort should be 8080.

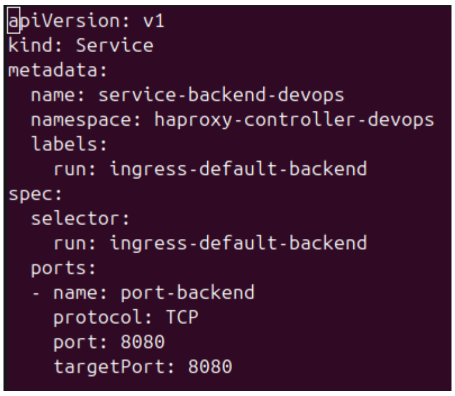


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment-devops
  namespace: haproxy-controller-devops
  labels:
    run: ingress-default-backend
spec:
  replicas: 1
  selector:
    matchLabels:
      run: ingress-default-backend
  template:
    metadata:
      labels:
        run: ingress-default-backend
    spec:
      containers:
      - name: backend-container-devops
        image: gcr.io/google_containers/defaultbackend:1.0
        ports:
        - containerPort: 8080
```

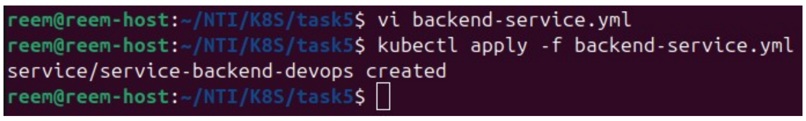


```
reem@reem-host:~/NTI/K8S/task5$ vi BackendDeployment.yml
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f BackendDeployment.yml
deployment.apps/backend-deployment-devops created
reem@reem-host:~/NTI/K8S/task5$
```

6- Create a service for backend which should be named as service-backend-devops under the same namespace, labels run should be ingress-default-backend. Configure spec as selector's run should be ingress-default-backend, port should be named as port-backend, protocol should be TCP, port should be 8080 and targetPort should be 8080.



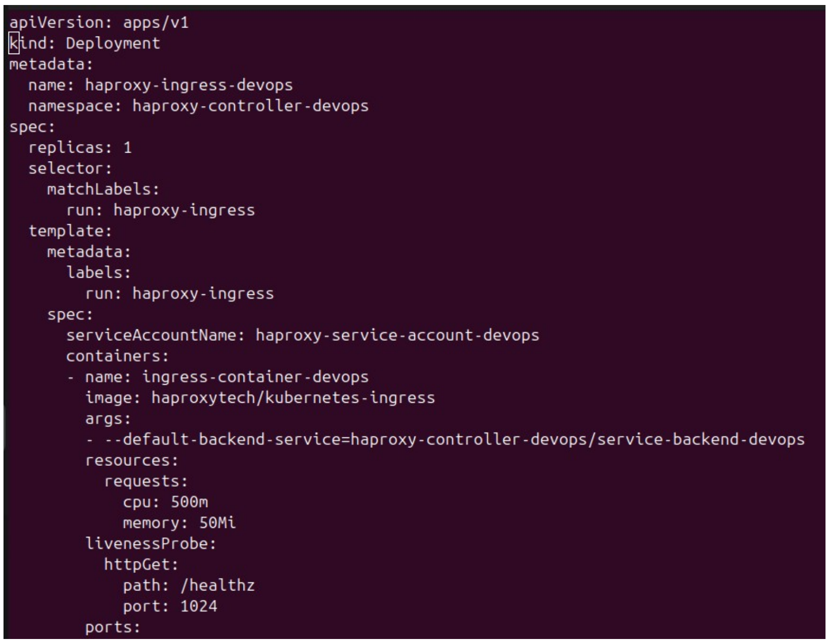
```
apiVersion: v1
kind: Service
metadata:
  name: service-backend-devops
  namespace: haproxy-controller-devops
  labels:
    run: ingress-default-backend
spec:
  selector:
    run: ingress-default-backend
  ports:
  - name: port-backend
    protocol: TCP
    port: 8080
    targetPort: 8080
```



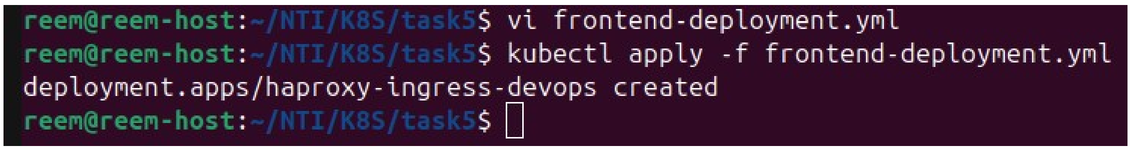
```
reem@reem-host:~/NTI/K8S/task5$ vi backend-service.yml
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f backend-service.yml
service/service-backend-devops created
reem@reem-host:~/NTI/K8S/task5$
```

7- Create a deployment for frontend which should be named haproxy-ingress-devops under the same namespace. Configure spec as replica should be 1, selector's matchLabels should be haproxy-ingress, template's labels run should be haproxy-ingress under metadata. The container name should be ingress-container-devops under the same service account haproxy-service-account-devops, use image haproxytech/kubernetes-ingress, give args as --default-backend-service=haproxy-controller-devops/service-backend-devops, resources requests for cpu should be 500m and for memory should be 50Mi, livenessProbe httpGet path should be /healthz its port should be 1024. The first port name should be http and its containerPort should be 80, second port name should be https and its containerPort should be 443 and third port name should be stat its containerPort should be 1024. Define environment as first env name should be TZ its value should be Etc/UTC, second env name should be POD_NAME its

```
valueFrom:
 fieldRef:
   fieldPath: should be metadata.name and third env name should be POD_NAMESPACE
its
 valueFrom:
   fieldRef:
     fieldPath: should be metadata.namespace.
```

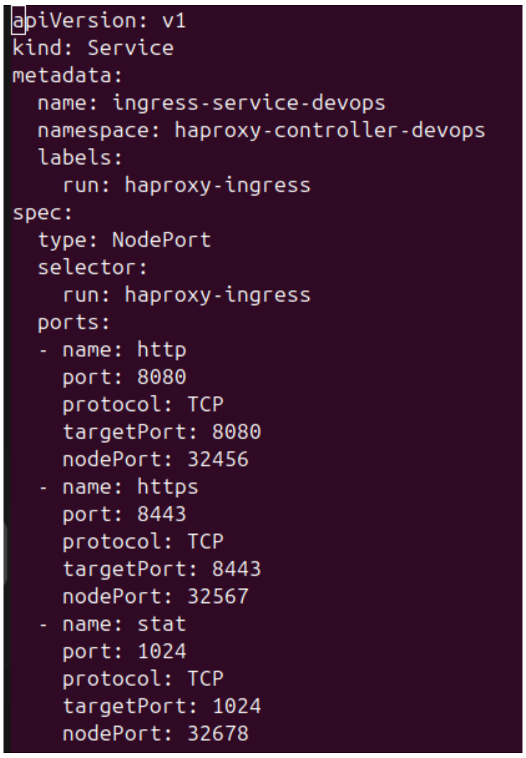


```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: haproxy-ingress-devops
  namespace: haproxy-controller-devops
spec:
  replicas: 1
  selector:
    matchLabels:
      run: haproxy-ingress
  template:
    metadata:
      labels:
        run: haproxy-ingress
    spec:
      serviceAccountName: haproxy-service-account-devops
      containers:
      - name: ingress-container-devops
        image: haproxytech/kubernetes-ingress
        args:
        - --default-backend-service=haproxy-controller-devops/service-backend-devops
        resources:
          requests:
            cpu: 500m
            memory: 50Mi
        livenessProbe:
          httpGet:
            path: /healthz
            port: 1024
        ports:
```

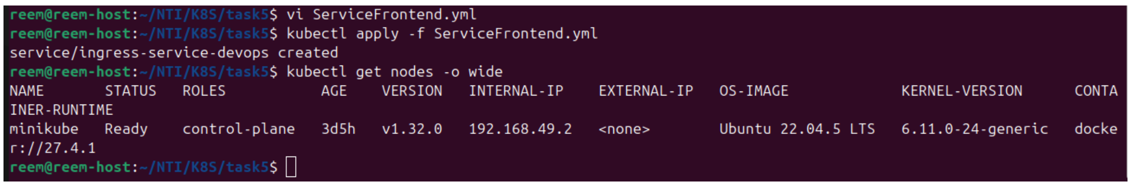


```
reem@reem-host:~/NTI/K8S/task5$ vi frontend-deployment.yml
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f frontend-deployment.yml
deployment.apps/haproxy-ingress-devops created
reem@reem-host:~/NTI/K8S/task5$ 
```

8- Create a service for frontend which should be named as ingress-service-devops under same namespace, labels run should be haproxy-ingress. Configure spec as selectors' run should be haproxy-ingress, type should be NodePort. The first port name should be http, its port should be 80, protocol should be TCP, targetPort should be 80 and nodePort should be 32456. The second port name should be https, its port should be 443, protocol should be TCP, targetPort should be 443 and nodePort should be 32567. The third port name should be stat, its port should be 1024, protocol should be TCP, targetPort should be 1024 and nodePort should be 32678.



```
apiVersion: v1
kind: Service
metadata:
  name: ingress-service-devops
  namespace: haproxy-controller-devops
  labels:
    run: haproxy-ingress
spec:
  type: NodePort
  selector:
    run: haproxy-ingress
  ports:
  - name: http
    port: 8080
    protocol: TCP
    targetPort: 8080
    nodePort: 32456
  - name: https
    port: 8443
    protocol: TCP
    targetPort: 8443
    nodePort: 32567
  - name: stat
    port: 1024
    protocol: TCP
    targetPort: 1024
    nodePort: 32678
```



```
reem@reem-host:~/NTI/K8S/task5$ vi ServiceFrontend.yml
reem@reem-host:~/NTI/K8S/task5$ kubectl apply -f ServiceFrontend.yml
service/ingress-service-devops created
reem@reem-host:~/NTI/K8S/task5$ kubectl get nodes -o wide
NAME       STATUS   ROLES           AGE    VERSION   INTERNAL-IP    EXTERNAL-IP   OS-IMAGE             KERNEL-VERSION      CONTA
INER-RUNTIME
minikube   Ready    control-plane   3d5h   v1.32.0   192.168.49.2   <none>        Ubuntu 22.04.5 LTS   6.11.0-24-generic   docke
r://27.4.1
reem@reem-host:~/NTI/K8S/task5$ 
```