Deploy HaProxy

1- Create a namespace haproxy-controller-devops.

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

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

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.

5- Create a backend deployment which should be named as backend-deployment-devopsunder 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.

6- Create a service for backend which should be named as service-backend-devops underthe 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.

7- Create a deployment for frontend which should be named haproxy-ingress-devopsunder 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.

8- Create a service for frontend which should be named as ingress-service-devops undersame 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.

```
controlplane:~$ vi namespace.yaml
controlplane:~$ kubectl apply -f namespace.yaml
namespace/haproxy-controller-devops created
controlplane:~$ kubectl get namespaces
NAME                        STATUS   AGE
default                     Active   34d
haproxy-controller-devops   Active   13s
kube-node-lease             Active   34d
kube-public                 Active   34d
kube-system                 Active   34d
local-path-storage          Active   34d
controlplane:~$ vi service-account.yaml
controlplane:~$ kubectl ap
api-resources  (Print the supported API resources on the server)
api-versions   (Print the supported API versions on the server, in the form of "group/version")
apply          (Apply a configuration to a resource by file name or stdin)
controlplane:~$ kubectl apply -f service-account.yaml
serviceaccount/haproxy-service-account-devops created
```

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

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

```
controlplane:~$ vi cluster-role.yaml
controlplane:~$ kubectl apply -f cluster-role.yaml
clusterrole.rbac.authorization.k8s.io/haproxy-cluster-role-devops created
controlplane:~$ vi cluster-role-pinding.yml
controlplane:~$ kubectl apply -f cluster-role-pinding.yml
clusterrolebinding.rbac.authorization.k8s.io/haproxy-cluster-role-binding-devops created
controlplane:~$
```

```yaml
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"]
~
~
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: haproxy-cluster-role-binding-devops
  namespace: haproxy-controller-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
~
~
```

```
controlplane:~$ vi backend-deployment.yaml
controlplane:~$ kubectl apply -f backend-deployment.yaml
deployment.apps/backend-deployment-devops created
controlplane:~$ vi backend-service.yaml
controlplane:~$ vi backend-service.yaml
controlplane:~$ kubectl apply -f backend-service.yaml
service/service-backend-devops created
controlplane:~$ vi frontend-deployment.yaml
controlplane:~$ kubectl apply -f frontend-deployment.yaml
deployment.apps/haproxy-ingress-devops created
controlplane:~$ vi HA-proxy-service.yaml
controlplane:~$ kubectl apply -f HA-proxy-service.yaml
service/ingress-service-devops created
controlplane:~$
```

```
controlplane:~$ vi backend-deployment.yaml
controlplane:~$ kubectl apply -f backend-deployment.yaml
deployment.apps/backend-deployment-devops created
controlplane:~$ vi backend-service.yaml
controlplane:~$ vi backend-service.yaml
controlplane:~$ kubectl apply -f backend-service.yaml
service/service-backend-devops created
controlplane:~$ vi frontend-deployment.yaml
controlplane:~$ kubectl apply -f frontend-deployment.yaml
deployment.apps/haproxy-ingress-devops created
controlplane:~$ vi HA-proxy-service.yaml
controlplane:~$ kubectl apply -f HA-proxy-service.yaml
service/ingress-service-devops created
controlplane:~$
```

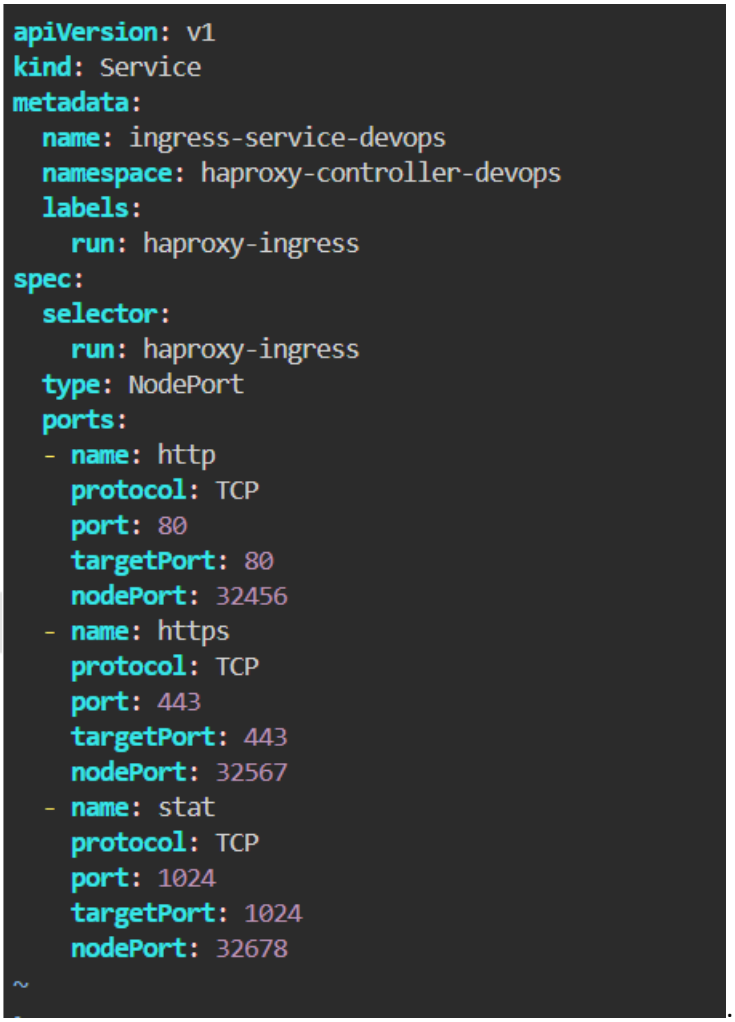
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment-devops
  namespace: haproxy-controller-devops
  labels:
    run: ingress-default-backend
spec:
  replicas: 1
  selector:
    matchLabels:
      run: ingress-default-backend
  template:
    metadata:
      labels:
     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

```yaml
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
~
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: haproxy-ingress-devops
  namespace: haproxy-controller-devops
  labels:
    run: haproxy-ingress
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
        ports:
        - name: http
          containerPort: 80
        - name: https
          containerPort: 443
        - name: stat
          containerPort: 1024
        livenessProbe:
          httpGet:
            path: /healthz
            port: 1024
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: haproxy-ingress-devops
  namespace: haproxy-controller-devops
  labels:
    run: haproxy-ingress
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
        ports:
        - name: http
          containerPort: 80
        - name: https
          containerPort: 443
        - name: stat
          containerPort: 1024
        livenessProbe:
          httpGet:
            path: /healthz
            port: 1024
```

```yaml
      ports:
      - name: http
        containerPort: 80
      - name: https
        containerPort: 443
      - name: stat
        containerPort: 1024
      livenessProbe:
        httpGet:
          path: /healthz
          port: 1024
      resources:
        requests:
          cpu: 500m
          memory: 50Mi
      env:
      - name: TZ
        value: Etc/UTC
      - name: POD_NAME
        valueFrom:
          fieldRef:
            fieldPath: metadata.name
      - name: POD_NAMESPACE
        valueFrom:
          fieldRef:
            fieldPath: metadata.namespace
```



```yaml
apiVersion: v1
kind: Service
metadata:
  name: ingress-service-devops
  namespace: haproxy-controller-devops
  labels:
    run: haproxy-ingress
spec:
  selector:
    run: haproxy-ingress
  type: NodePort
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 32456
  - name: https
    protocol: TCP
    port: 443
    targetPort: 443
    nodePort: 32567
  - name: stat
    protocol: TCP
    port: 1024
    targetPort: 1024
    nodePort: 32678
~
```

```
controlplane:~$ kubectl get namespaces
NAME                        STATUS   AGE
default                     Active   34d
haproxy-controller-devops   Active   18m
kube-node-lease             Active   34d
kube-public                 Active   34d
kube-system                 Active   34d
local-path-storage          Active   34d
controlplane:~$ kubectl get serviceaccount -n haproxy-controller-devops
NAME                            SECRETS   AGE
default                         0         18m
haproxy-service-account-devops  0         16m
controlplane:~$ kubectl get clusterrole
NAME                                                                   CREATED AT
admin                                                                  2025-03-22T19:51:47Z
calico-kube-controllers                                                2025-03-22T19:51:51Z
calico-node                                                            2025-03-22T19:51:51Z
cluster-admin                                                          2025-03-22T19:51:47Z
edit                                                                   2025-03-22T19:51:47Z
flannel                                                                2025-03-22T19:51:51Z
haproxy-cluster-role-devops                                            2025-04-26T19:17:53Z
kubeadm:get-nodes                                                      2025-03-22T19:51:49Z
local-path-provisioner-role                                            2025-03-22T19:53:52Z
system:aggregate-to-admin                                              2025-03-22T19:51:47Z
system:aggregate-to-edit                                               2025-03-22T19:51:47Z
system:aggregate-to-view                                               2025-03-22T19:51:47Z
system:auth-delegator                                                  2025-03-22T19:51:47Z
system:basic-user                                                      2025-03-22T19:51:47Z
system:certificates.k8s.io:certificatesigningrequests:nodeclient       2025-03-22T19:51:47Z
system:certificates.k8s.io:certificatesigningrequests:selfnodeclient   2025-03-22T19:51:47Z
system:certificates.k8s.io:kube-apiserver-client-approver              2025-03-22T19:51:47Z
system:certificates.k8s.io:kube-apiserver-client-kubelet-approver      2025-03-22T19:51:47Z
system:certificates.k8s.io:kubelet-serving-approver                    2025-03-22T19:51:47Z
system:certificates.k8s.io:legacy-unknown-approver                     2025-03-22T19:51:47Z
system:controller:attachdetach-controller                              2025-03-22T19:51:47Z
```

```
controlplane:~$ kubectl get clusterrolebinding
NAME                                                   ROLE
  AGE
calico-kube-controllers                                ClusterRole/calico-kube-controllers
  34d
canal-calico                                           ClusterRole/calico-node
  34d
canal-flannel                                          ClusterRole/flannel
  34d
cluster-admin                                          ClusterRole/cluster-admin
  34d
haproxy-cluster-role-binding-devops                    ClusterRole/haproxy-cluster-role-devops
  11m
kubeadm:cluster-admins                                 ClusterRole/cluster-admin
  34d
kubeadm:get-nodes                                      ClusterRole/kubeadm:get-nodes
  34d
kubeadm:kubelet-bootstrap                              ClusterRole/system:node-bootstrapper
```