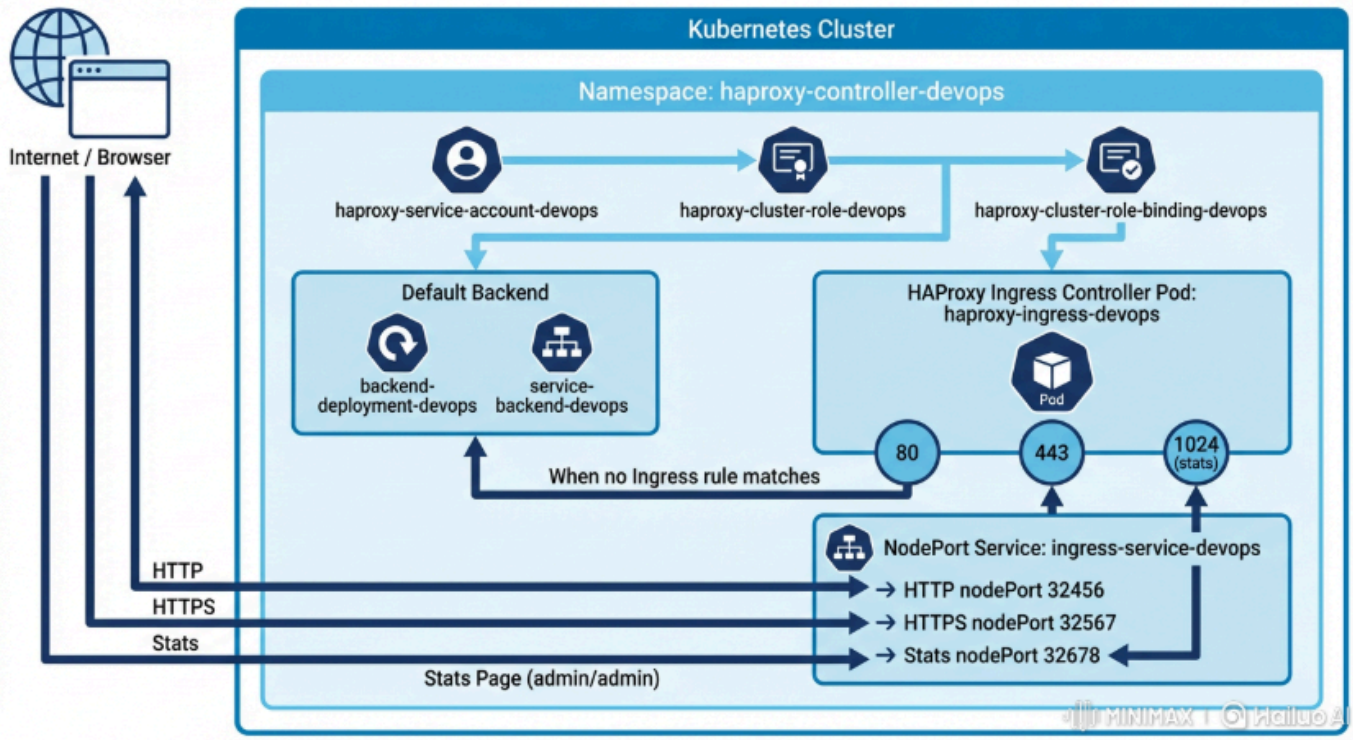
Kubernetes Project

project content

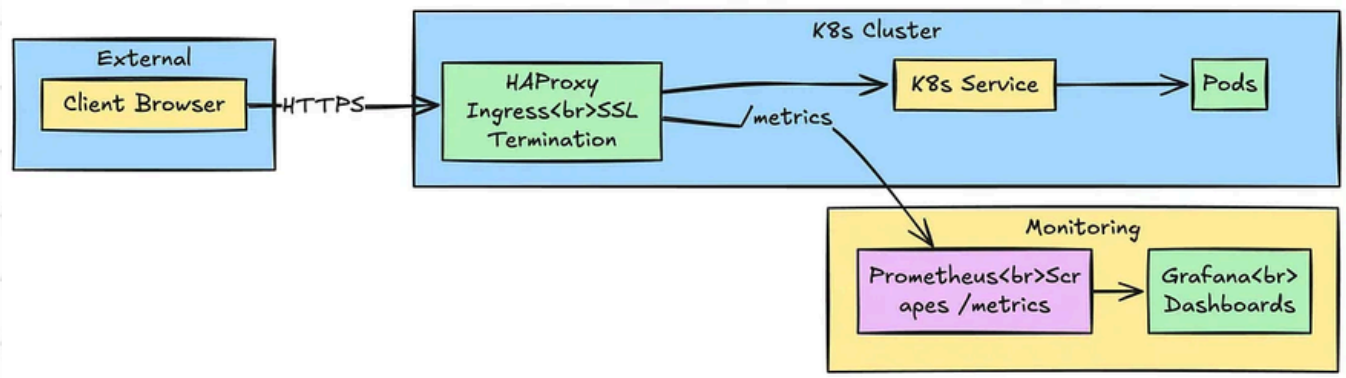
1. Namespace Creation
2. ServiceAccount Creation
3. ClusterRole Creation
4. ClusterRoleBinding Creation
5. Backend Deployment
6. Backend Service
7. Frontend (HAProxy) Deployment
8. Frontend Service (NodePort)

The sources I used first before starting the project

1. HAProxy Ingress Controller Kubernetes
2. Kubernetes-ingress (Github)
3. Core-Components-and-Resources
4. Scheduling
5. Application-Lifecycle-Management
6. Security
7. Networking



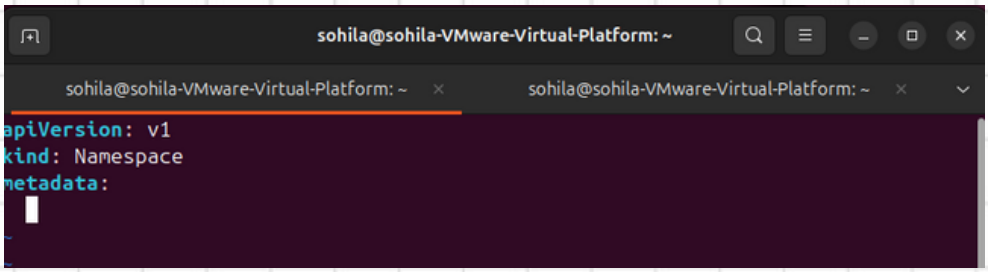
HAProxy Ingress Controller Kubernetes



I- Create a namespace haproxy-controller-devops.

```
sohila@sohila-VMware-Virtual-Platform:~$ cd ~
sohila@sohila-VMware-Virtual-Platform:~$ rm -rf haproxy-project
sohila@sohila-VMware-Virtual-Platform:~$ rm -rf haproxy-project
sohila@sohila-VMware-Virtual-Platform:~$ mkdir manifests
sohila@sohila-VMware-Virtual-Platform:~$ vim namespace.yaml
sohila@sohila-VMware-Virtual-Platform:~$ k apply namespace.yaml
error: Unexpected args: [namespace.yaml]
See 'kubectl apply -h' for help and examples
sohila@sohila-VMware-Virtual-Platform:~$ k apply -f namespace.yaml
namespace/haproxy-controller-devops created
sohila@sohila-VMware-Virtual-Platform:~$
```

file content



We isolate HAProxy resources in a separate space within the cluster to prevent confusion and maintain Kubernetes organization.

What we do ??

We create a new namespace instead of putting everything in the default namespace.

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

```
sohila@sohila-VMware-Virtual-Platform:~$ vim serviceaccount.yaml
sohila@sohila-VMware-Virtual-Platform:~$ k apply -f serviceaccount.yaml
serviceaccount/haproxy-service-account-devops created
sohila@sohila-VMware-Virtual-Platform:~$
```

file content

```
sohila@sohila-VMware-Virtual-Platform:~$ cat serviceaccount.yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: haproxy-service-account-devops
  namespace: haproxy-controller-devops
sohila@sohila-VMware-Virtual-Platform:~$
```

The ServiceAccount is the identity that HAProxy Controller uses to access and interact with the Kubernetes API.

Reason

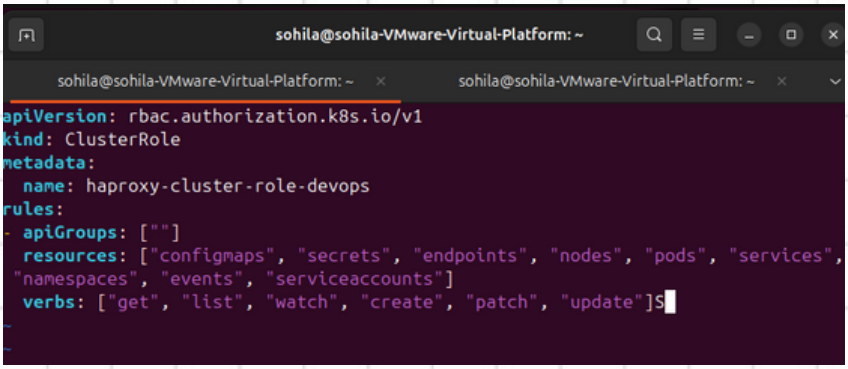
The Controller itself cannot use a regular user account; it must have a dedicated ServiceAccount in order to:

- Control the permissions it receives.
- Prevent any unnecessary access.

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"

```
sohila@sohila-VMware-Virtual-Platform:~$ vim clusterrole.yaml
sohila@sohila-VMware-Virtual-Platform:~$ k apply -f clusterrole.yaml
clusterrole.rbac.authorization.k8s.io/haproxy-cluster-role-devops created
sohila@sohila-VMware-Virtual-Platform:~$
```

file content



Goal

We determine the type of permissions HAProxy Controller needs to function.

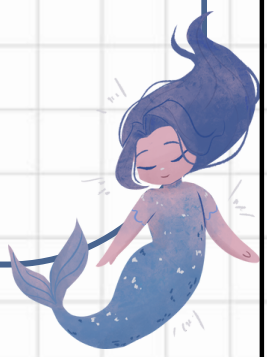
What does the Controller need to do??

- Read ConfigMaps → Get Configurations.
- Read Secrets → If there are Certificates.
- Monitor Endpoints → to know the Backends.
- Monitor Pods → to know who is working and who is not.
- Monitor Nodes → sometimes needs information about the Cluster.

Required Permissions??

Actions:

- get, list, watch
- And for ConfigMaps, it also needs:
- create, patch, update



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.

```
sohila@sohila-VMware-Virtual-Platform:~$ vim clusterrolebinding.yaml
sohila@sohila-VMware-Virtual-Platform:~$ k apply -f clusterrolebinding.yaml
clusterrolebinding.rbac.authorization.k8s.io/haproxy-cluster-role-binding-devops
 created
```

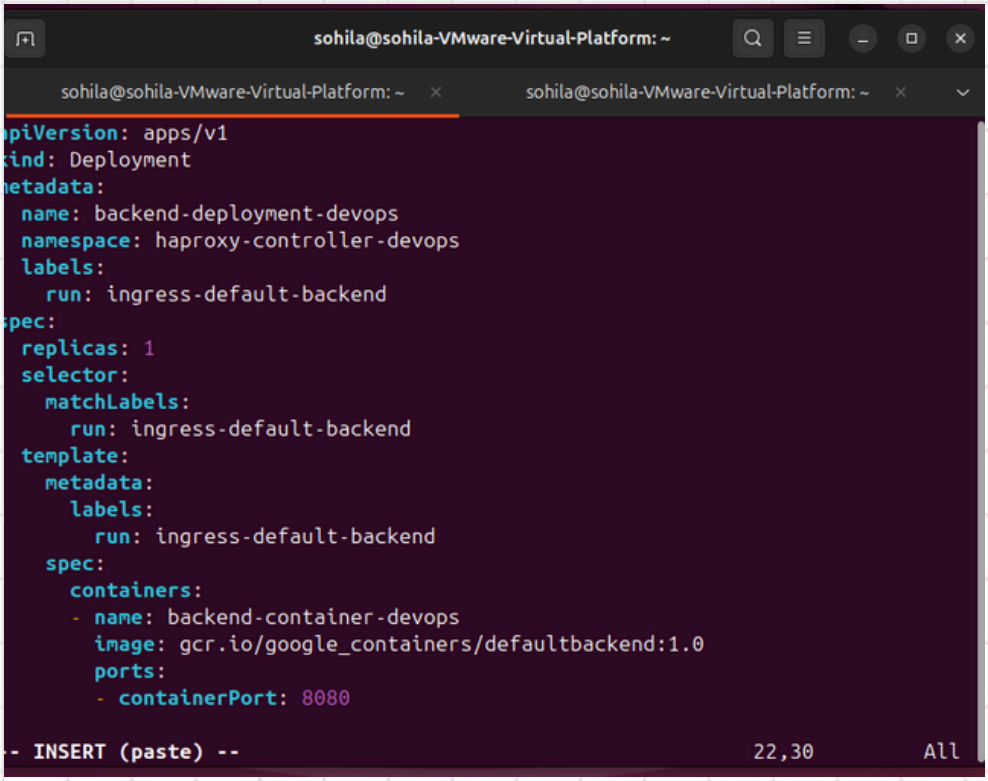
file content

```
sohila@sohila-VMware-Virtual-Platform: ~
sohila@sohila-VMware-Virtual-Platform: ~   sohila@sohila-VMware-Vir
piVersion: rbac.authorization.k8s.io/v1
ind: ClusterRoleBinding
etadata:
  name: haproxy-cluster-role-binding-devops
oleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: haproxy-cluster-role-devops
ubjects:
- kind: ServiceAccount
  name: haproxy-service-account-devops
  namespace: haproxy-controller-devopsS
```

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 I, 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:I.0 (use exact name of image as mentioned) and its containerPort should be 8080

```
sohila@sohila-VMware-Virtual-Platform:~$ vim backend-deployment.yaml
sohila@sohila-VMware-Virtual-Platform:~$ k apply -f backend-deployment.yaml
deployment.apps/backend-deployment-devops created
sohila@sohila-VMware-Virtual-Platform:~$
```

file content



Goal

We run HAProxy Controller inside the cluster to:

- Monitor Ingress objects.
- Generate HAProxy configuration.
- Perform load balancing for traffic.

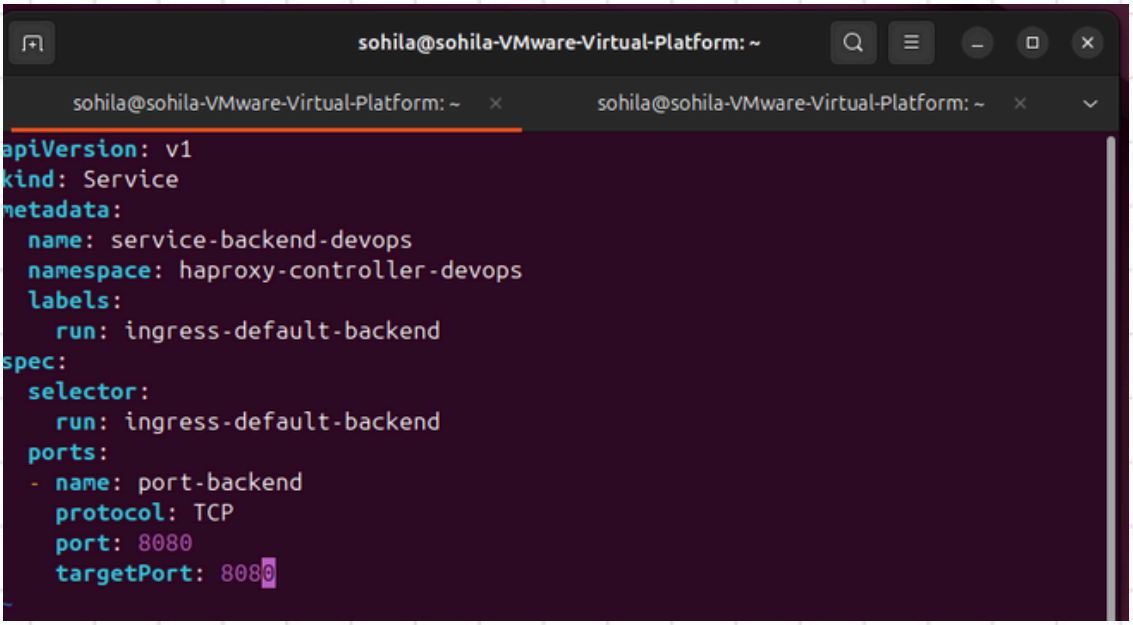
Steps

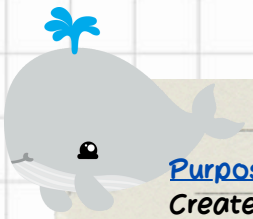
1. Deploy HAProxy Controller.
2. Connect it to the ServiceAccount.
3. Create a service if you need to expose it.

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

```
sohila@sohila-VMware-Virtual-Platform:~$ vim backend-service.yaml
sohila@sohila-VMware-Virtual-Platform:~$ k apply -f backend-service.yaml
service/service-backend-devops created
```

file content





Purpose of this step

Create a Service to connect the backend deployment to the Cluster
to allow access to the Pod running the backend.

Why are we creating a service?

The pod will have a changing IP address.
The service provides a stable virtual IP address.
The HAProxy Controller needs to consult a stable backend.

Without the service:

HAProxy will not be able to connect to the backend deployment.

7- Create a deployment for frontend which should be named haproxy-ingress-devops under the same namespace. Configure spec as replica should be I, 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 I024. 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 I024. Define environment as first env name should be TZ its value should be Etc/UTC, second env name should be POD_NAME its valueFrom: f ieldRef: f ieldPath: should be metadata.name and third env name should be POD_NAMESPACE its valueFrom: f ieldRef: f ieldPath: should be metadata.namespace.

```
sohila@sohila-VMware-Virtual-Platform:~$ vim haproxy-deployment.yaml
sohila@sohila-VMware-Virtual-Platform:~$ k apply -f haproxy-deployment.yaml
deployment.apps/haproxy-ingress-devops created
```

file content

sohila@sohila-VMware-Virtual-Platform: ~

sohila@sohila-VMware-Virtual-Platform: ~

```
piVersion: apps/v1
ind: Deployment
etadata:
  name: haproxy-ingress-devops
  namespace: haproxy-controller-devops
pec:
  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:
        - name: http
          containerPort: 80
        - name: https
          containerPort: 443
        - name: stat
          containerPort: 1024


        env:
        - name: TZ
          value: Etc/UTC
        - name: POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
        - name: POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
```

Purpose of this step

Launch HAProxy Ingress Controller within the cluster to receive requests from users and distribute them to services.

Why are we doing deployment?

- To manage the operation of HAProxy pods.
- It can restart in case of failure.
- It determines the number of replicas (here = 1).

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 I024, protocol should be TCP, targetPort should be I024 and nodePort should be 32678

```
sohila@sohila-VMware-Virtual-Platform:~$ vim ingress-service.yaml
sohila@sohila-VMware-Virtual-Platform:~$ k apply -f ingress-service.yaml
service/ingress-service-devops created
```

file content

sohila@sohila-VMware-Virtual-Platform: ~

sohila@sohila-VMware-Virtual-Platform: ~ × sohila@sohila-VMware-Virtu

```yaml
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
    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
```

Purpose of this step

To provide an entry point from outside the cluster to direct traffic to HAProxy Deployment.

Why are we using a NodePort service?

HAProxy must be available to users outside the cluster.

That is why we are using NodePort instead of ClusterIP.

NodePort provides:

- Fixed ports on each node
- The user can access the Node IP + NodePort
- Then the service directs traffic to the HAProxy Pod