Q1: where is the kubectl config file located
Located in the home directory of the user under .kube/config
Q2: how many clusters are defined in the config
1 cluster is defined only

```
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSURCVENDQWUyZ0F3SUJBZ0lJTXJnME1KZmlhOWN3RFFZSktvWklodmNOQV
    server: https://172.30.1.2:6443
  name: kubernetes
```

Q3:what is the current configured user in the default context?
The default user configured is kubernetes-admin

```
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
```

Q4: create a PV called pv-log

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-log
spec:
  storageClassName: ""
  capacity:
    storage: 100Mi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: /pv/log

~
~
```

Q5: create a PVC

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: claim-log-1
spec:
  storageClassName: ""
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 50Mi

~
~
~
```

Q6: create an nginx pod and attach to the pVC

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webapp
spec:
  volumes:
    - name: bla-log
      persistentVolumeClaim:
        claimName: claim-log-1
  containers:
    - name: webap
      image: nginx
      volumeMounts:
        - mountPath: /var/log/nginx
          name: bla-log
```

Q7: creating shared containers using emptyDir

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volume-share-datacenter
spec:
  volumes:
    - name: volume-share
      emptyDir:
        sizeLimit: 200Mi
  containers:
    - name: volume-container-datacenter-1
      image: centos
      command:
        - /bin/bash
        - -c
      args:
        - sleep 10000
      volumeMounts:
        - mountPath: /tmp/news
          name: volume-share

    - name: volume-container-datacenter-2
      image: centos
      command:
        - /bin/bash
        - -c
      args:
        - sleep 10000
      volumeMounts:
        - mountPath: /tmp/cluster
          name: volume-share
```

```
controlplane $ kubectl exec --stdin --tty volume-share-datacenter -c volume-container-datacenter-1  -- /bin/bash
[root@volume-share-datacenter /]# cd /tmp/news
[root@volume-share-datacenter news]# ls
[root@volume-share-datacenter news]# echo "Welcome from datacenter-1!" > news.txt
[root@volume-share-datacenter news]#
```

```
exit
controlplane $ kubectl exec --stdin --tty volume-share-datacenter -c volume-container-datacenter-2  -- /bin/bash
[root@volume-share-datacenter /]# cd /tmp/cluster/
[root@volume-share-datacenter cluster]# ls
news.txt
[root@volume-share-datacenter cluster]# cat news.txt
Welcome from datacenter-1!
[root@volume-share-datacenter cluster]#
```

Q8: creating ubuntu shared with nginx logs

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: webserver
spec:
  volumes:
    - name: shared-logs
      emptyDir:
        sizeLimit: 200Mi
  containers:
    - name: "nginx-container"
      image: nginx:latest
      volumeMounts:
        - mountPath: /var/log/nginx
          name: shared-logs

    - name: "sidecar-container"
      image: ubuntu:latest
      command:
        - /bin/bash
        - -c
      args:
        - while true; do cat  /var/log/nginx/access.log /var/log/nginx/error.log; sleep 30; done
      volumeMounts:
        - mountPath: /var/log/nginx
          name: shared-logs
```

```
controlplane $ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
volume-share-datacenter   2/2     Running   0          12m
webapp                    1/1     Running   0          20m
webserver                 2/2     Running   0          53s
```

Webapp is currently running

Q9: create a service called pvviewer

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: pvviewer
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  namespace: default
  name: pvviewer-role
rules:
- apiGroups: [""]
  resources: ["persistentvolumes"]
  verbs: ["list"]
```

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: pvviewer-role-binding
subjects:
- kind: ServiceAccount
  name: pvviewer
  namespace: default
  apiGroup: ""
roleRef:
  kind: ClusterRole
  name: pvviewer-role
  apiGroup: ""
```

```
controlplane $ kubectl apply -f .
serviceaccount/pvviewer unchanged
clusterrole.rbac.authorization.k8s.io/pvviewer-role unchanged
clusterrolebinding.rbac.authorization.k8s.io/pvviewer-role-binding created
```

```
NAME                    ROLE                        AGE     USERS   GROUPS   SERVICEACCOUNTS
pvviewer-role-binding   ClusterRole/pvviewer-role   3m16s                    default/pvviewer
```

Successfully binded

```
controlplane $ kubectl describe clusterrole pvviewer-role
Name:         pvviewer-role
Labels:       <none>
Annotations:  <none>
PolicyRule:
  Resources          Non-Resource URLs  Resource Names  Verbs
  ---------          -----------------  --------------  -----
  persistentvolumes  []                 []              [list]
controlplane $
```

These are the permissions given to the role

Q10: create a configmap with nginx configurations

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-config
data:
  nginx.conf: |
    events {}
    http {
          server {
            listen 80;
            location / {
              return 200 'hello from custom nginx-config'
        }
      }
    }
```

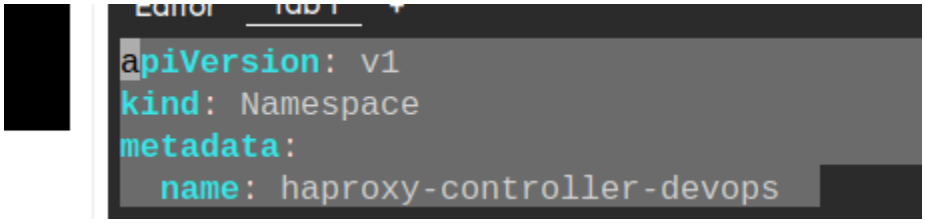
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  volumes:
    - name: nginx-confmap
      configMap:
        name: nginx-config
  containers:
    - name: nginx-pod
      image: nginx
      volumeMounts:
        - mountPath: /etc/nginx/nginx.conf
          name: nginx-confmap
```

```
error: you must specify at least one command for the container
controlplane $ kubectl exec nginx-pod -- /bin/bash
controlplane $ kubectl exec --tty --stdin nginx-pod -- /bin/bash
root@nginx-pod:/# curl http
curl: (6) Could not resolve host: http
root@nginx-pod:/# curl localhost
hello from custom nginx-configroot@nginx-pod:/#
```

Checking that the configurations are in place

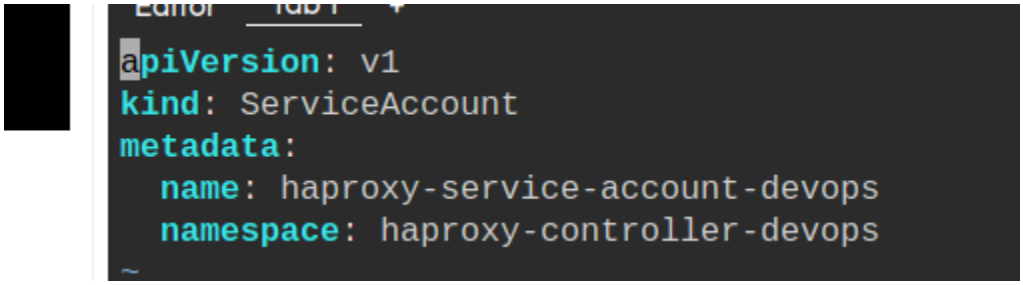
HAPROXY:

1- create haproxy namespace



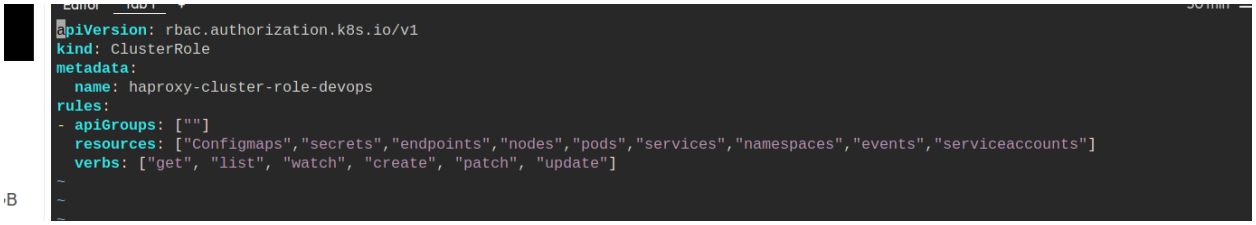
```
apiVersion: v1
kind: Namespace
metadata:
  name: haproxy-controller-devops
```

2- create service account



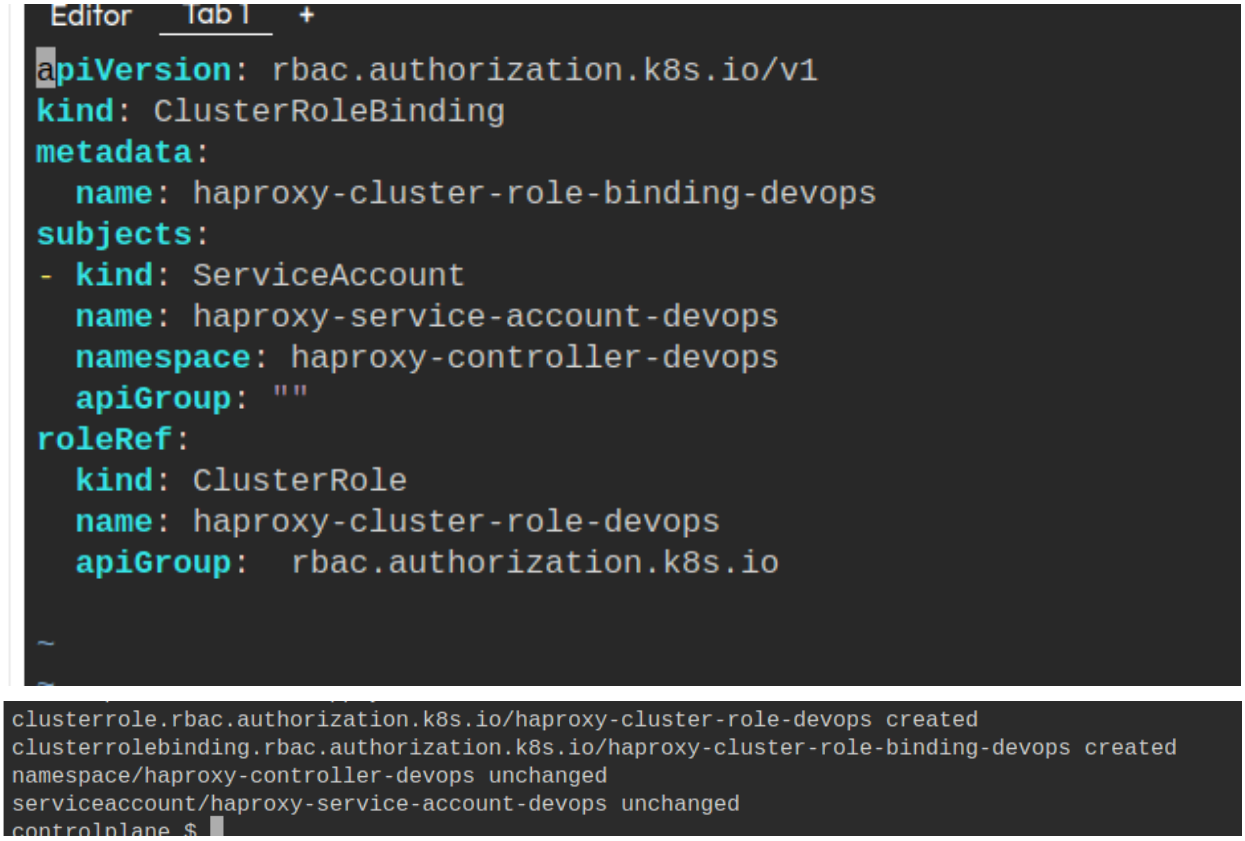
```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: haproxy-service-account-devops
  namespace: haproxy-controller-devops
```

3- create a ClusterRole



```
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"]
```

4- create a clusterRoleBinding



5- create a backend deployment in the haproxy-controller-devops namespace

```
Editor   Tab 1   +
apiVersion: apps/v1
kind: Deployment
metadata:
  name: backend-deployment-devops
  labels:
    run: ingress-default-backend
  namespace: haproxy-controller-devops
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
~
```

6- creating a service for the backend

```
Editor   Tab 1   +
apiVersion: v1
kind: Service
metadata:
  name: port-backend
  namespace: haproxy-controller-devops
spec:
  type: ClusterIP
  selector:
    run: ingress-default-backend
  ports:
    - port: 80
      targetPort: 8080
~
~
~
~
~
~
~
~
~
~
~
```

7- creating the frontend service with the given requirements

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name:  haproxy-ingress-devops
  labels:
    run: haproxy-ingress
    namespace: haproxy-controller-devops
spec:
  replicas: 1
  selector:
     matchLabels:
      run: haproxy-ingress
  template:
    metadata:
      namespace: haproxy-controller-devops
      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
          resources:
            requests:
              cpu: 500m
              memory: 50Mi
          livenessProbe:
            httpGet:
              port: 1024
              path: /health
          env:
           - name: TZ
             value: Etc/UT
           - name: POD_NAME
             valueFrom:
                fieldRef:
                  fieldPath: metadata.name
           - name: POD_NAMESPACE
             valueFrom:
                fieldRef:
                  fieldPath: metadata.namespace
```

8- creating a service for the frontend

```
Editor   Tab 1   +
apiVersion: v1
kind: Service
metadata:
  name:  ingress-service-devops
  namespace:  haproxy-controller-devops
spec:
  type: NodePort
  selector:
   run: haproxy-ingress
  ports:
    - port: 80
      targetPort: 80
      nodePort: 32456
      name: http
    - port: 443
      targetPort: 443
      nodePort: 32567
      name: https
    - port: 1024
      targetPort: 1024
      nodePort: 32678
      name: stat
```

9- both are deployed in the haproxy controller devops

```
deployment.apps/backend-deployment-devops unchanged
controlplane $ kubectl get deploy -n haproxy-controller-devops
NAME                       READY   UP-TO-DATE   AVAILABLE   AGE
backend-deployment-devops   1/1     1            1           36m
haproxy-ingress-devops      1/1     1            1           13s
controlplane $
```

10- curl the node port of the current deployment

kubectl get nodes -o wide

Curl IP/health:32456 of the node contianing the deployment