

**Q1)** Create a pod named volume-share-datacenter.

For first container, use image centos:latest, container should be named as volume-container-datacenter-1, and run a command '/bin/bash', '-c' and 'sleep 10000'. Volume volume-share should be mounted at path /tmp/news.

For second container, use image centos:latest, container should be named as volume-container-datacenter-2, and run a command '/bin/bash', '-c' and 'sleep 10000'. Volume volume-share should be mounted at path /tmp/cluster.

Volumes to be named as volume-share and use emptyDir: { }.

```
1|apiVersion: v1
2|kind: Pod
3|metadata:
4|  name: volume-share-datacenter
5|spec:
6|  volumes:
7|    - name: volume-share
8|      emptyDir: {}
9|  containers:
10|    - name: volume-share-datacenter-1
11|      image: centos:latest
12|      command: ["/bin/bash"]
13|      args: ["-c", echo welcome-container1;sleep 1000;]
14|
15|      volumeMounts:
16|        - name: volume-share
17|          mountPath: /tmp/news
18|
19|    - name: volume-share-datacenter-2
20|      image: centos:latest
21|      command: ["/bin/bash"]
22|      args: ["-c", echo welcome-container2;sleep 1000;]
23|
24|      volumeMounts:
25|        - name: volume-share
26|          mountPath: /tmp/cluster
```

```
Activities Terminal Apr 15 10:56 PM
mostafa@mostafa-Predator: ~/Desktop/iti/Kubernetes
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$ kubectl exec -it volume-share-datacenter --container volume-share-datacenter-1 -- /bin/bash
[root@volume-share-datacenter /]# cd /tmp/news/
[root@volume-share-datacenter news]# cat > news.txt
Welcome from datacenter-1!!
^C
[root@volume-share-datacenter news]# cat news.txt
Welcome from datacenter-1!!
[root@volume-share-datacenter news]# exit
exit
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$ kubectl exec -it volume-share-datacenter --container volume-share-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]# exit
exit
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$
```

**Q2)** Create a pod named web-server.

Create an emptyDir volume name: shared-logs.

Create two containers from nginx and ubuntu images with latest tag only and remember to mention tag i.e nginx:latest, nginx container name should be nginx-container and ubuntu container name should be sidecar-container on web-server pod.

Add command on sidecar-container "sh","-c","while true; do cat /var/log/nginx/access.log /var/log/nginx/error.log; sleep 30; done"

Mount volume /var/log/nginx on both containers, all containers should be up and running

```
Y webServer.yaml 1 Kubernetes • webServer.yaml/...
io.k8s.api.core.v1.Pod (v1@pod.json)
1  apiVersion: v1
2  kind: Pod
3  metadata:
4  |   name: web-server
5  spec:
6  |   containers:
7  |   - name: nginx-container
8  |     image: nginx:latest
9  |     volumeMounts:
10 |     - mountPath: /var/log/nginx
11 |       name: web-server-1
12 |   - name: sidecar-container
13 |     image: ubuntu:latest
14 |     command: ["/bin/sh"]
15 |     args: ["-c", "while true; do cat /var/log/nginx/access.log /var/log/nginx/error.log; sleep 30; done"]
16 |     volumeMounts:
17 |     - mountPath: /var/log/nginx
18 |       name: web-server-2
19 |   volumes:
20 |   - name: shared-logs
21 |     emptyDir: {}
22
23
```

**Q3)** Create a new service account with the name pvviewer. Grant this Service account access to list all PersistentVolumes in the cluster by creating an appropriate cluster role called pvviewer-role and ClusterRoleBinding called pvviewer-role-binding.

The screenshot shows the Visual Studio Code editor with two files open. The left file, `pvviewer_role_binding.yaml`, defines a `ClusterRoleBinding` that binds the `pvviewer-role` role to the `pvviewer` service account in the `default` namespace. The right file, `pvviewer_role.yaml`, defines a `ClusterRole` named `pvviewer-role` with permissions to list `persistentvolumes` in the `default` namespace.

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: ClusterRoleBinding
3 metadata:
4   name: pvviewer-role-binding
5   namespace: default
6 subjects:
7 - kind: ServiceAccount
8   name: pvviewer
9   namespace: default
10  apiGroup: ""
11 roleRef:
12   kind: ClusterRole
13   name: pvviewer-role
14   apiGroup: rbac.authorization.k8s.io

1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: ClusterRole
3 metadata:
4   namespace: default
5   name: pvviewer-role
6 rules:
7 - apiGroups: [""]
8   resources: ["persistentvolumes"]
9   verbs: ["list"]
```

## HAProxy

The screenshot shows a terminal window with the following commands and output:

```
mostafa@mostafa-Predator: ~/Desktop/iti/Kubernetes$ kubectl create ns haproxy-controller-devops
namespace/haproxy-controller-devops created
mostafa@mostafa-Predator: ~/Desktop/iti/Kubernetes$ kubectl create sa haproxy-service-account-devops -n haproxy-controller-devops
serviceaccount/haproxy-service-account-devops created
mostafa@mostafa-Predator: ~/Desktop/iti/Kubernetes$ kubectl get serviceaccounts
NAME      SECRETS  AGE
default   0         8d
mostafa@mostafa-Predator: ~/Desktop/iti/Kubernetes$ kubectl get serviceaccounts -n haproxy-controller-devops
NAME      SECRETS  AGE
haproxy-service-account-devops  0         58s
mostafa@mostafa-Predator: ~/Desktop/iti/Kubernetes$
```

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".

```
haproxy-cluster-role-devops.yaml Kubernetes • haproxy-cluster-role-devops.yaml/({} metadata
io.k8s.api.rbac.v1.ClusterRole (v1@clusterrole.json)
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: ClusterRole
3 metadata:
4   name: haproxy-cluster-role-devops
5 rules:
6 - apiGroups: ["" ]
7   resources: ["Configmaps","secrets","endpoints","nodes","pods","services","namespaces","events","serviceaccounts"]
8   verbs: ["get", "list", "watch", "create", "patch", "update"]
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

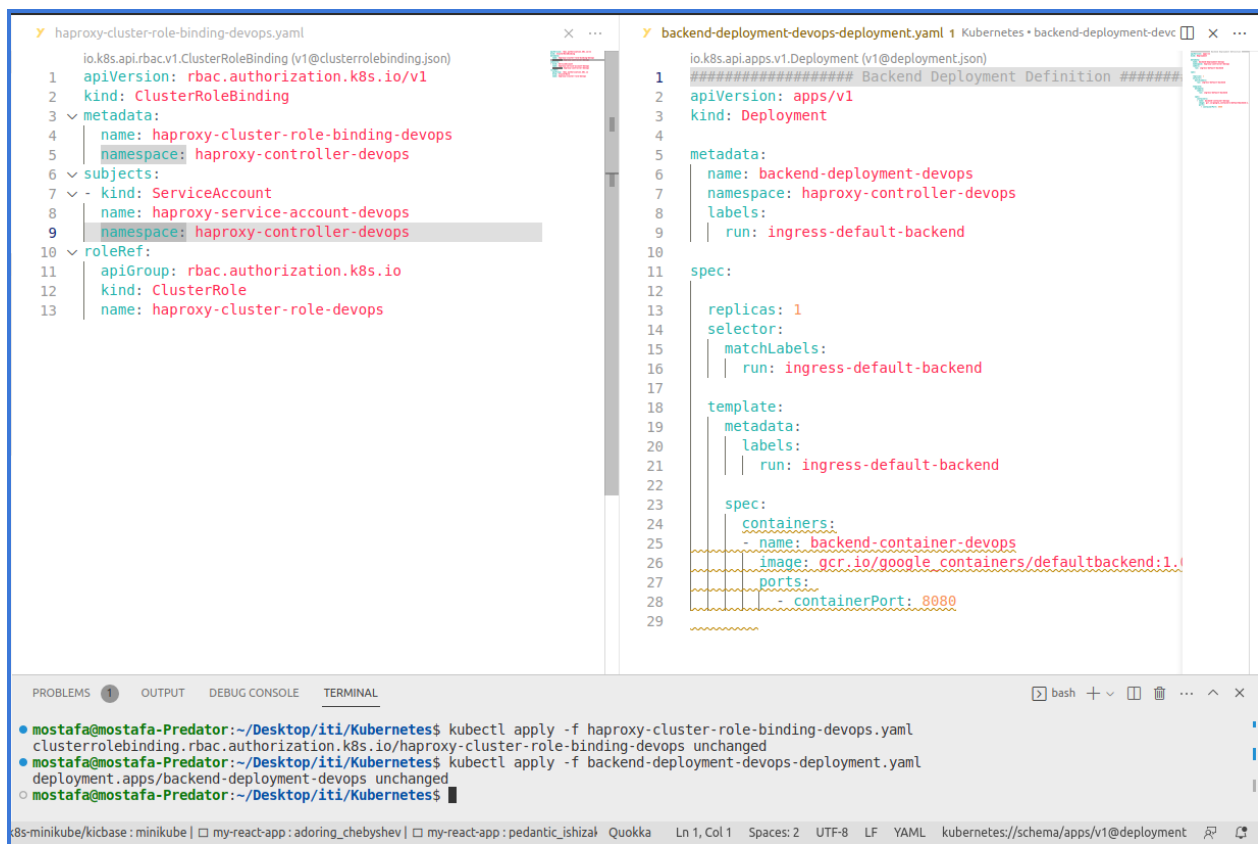
```
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$ kubectl apply -f haproxy-cluster-role-devops.yaml
clusterrole.rbac.authorization.k8s.io/haproxy-cluster-role-devops unchanged
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$
```

k8s-minikube/kicbase: minikube | my-react-app: adoring\_chebyshev | my-react-app: pedantic\_ishizak Quokka UTF-8 LF YAML kubernetes://schema/rbac.authorization.k8s.io/v1@clusterrole

**Q4)** 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.

=====

**Q5)** Create a backend deployment which should be named as backend-deployment-devops under the same namespace, labels "run=ingress-default-backend", replicas=1 container name=backend-container-devops, image gcr.io/google\_containers/defaultbackend:1.0 containerPort=8080.



```
haproxy-cluster-role-binding-devops.yaml
1  io.k8s.api.rbac.v1.ClusterRoleBinding (v1@clusterrolebinding.json)
2  apiVersion: rbac.authorization.k8s.io/v1
3  kind: ClusterRoleBinding
4  metadata:
5    name: haproxy-cluster-role-binding-devops
6    namespace: haproxy-controller-devops
7  subjects:
8    - kind: ServiceAccount
9      name: haproxy-service-account-devops
10     namespace: haproxy-controller-devops
11  roleRef:
12    apiGroup: rbac.authorization.k8s.io
13    kind: ClusterRole
14    name: haproxy-cluster-role-devops

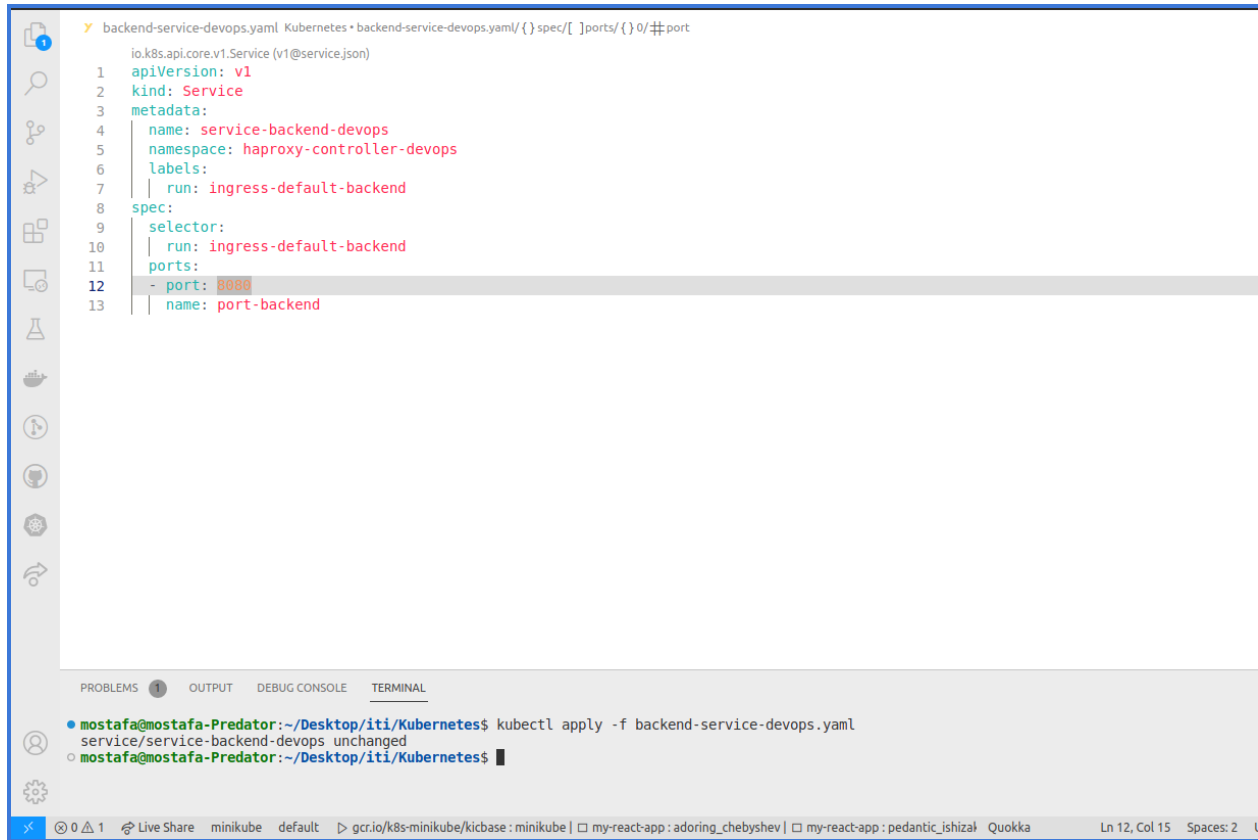
backend-deployment-devops-deployment.yaml 1 Kubernetes • backend-deployment-devops
1  io.k8s.api.apps.v1.Deployment (v1@deployment.json)
2  apiVersion: apps/v1
3  kind: Deployment
4
5  metadata:
6    name: backend-deployment-devops
7    namespace: haproxy-controller-devops
8    labels:
9      run: ingress-default-backend
10
11  spec:
12    replicas: 1
13    selector:
14      matchLabels:
15        run: ingress-default-backend
16
17    template:
18      metadata:
19        labels:
20          run: ingress-default-backend
21
22      spec:
23        containers:
24          - name: backend-container-devops
25            image: gcr.io/google_containers/defaultbackend:1.0
26            ports:
27              - containerPort: 8080
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

```
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$ kubectl apply -f haproxy-cluster-role-binding-devops.yaml
clusterrolebinding.rbac.authorization.k8s.io/haproxy-cluster-role-binding-devops unchanged
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$ kubectl apply -f backend-deployment-devops-deployment.yaml
deployment.apps/backend-deployment-devops unchanged
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$
```

8s-minikube/kicbase: minikube | my-react-app: adoring\_chebyshev | my-react-app: pedantic\_ishizak | Quokka | Ln 1, Col 1 | Spaces: 2 | UTF-8 | LF | YAML | kubernetes://schema/apps/v1@deployment

**Q6)** Create a service for backend which should be named as “service-backend-devops” under the same namespace, port should be named as port-backend.



The image shows a VS Code editor window with a file named `backend-service-devops.yaml` in the `Kubernetes • backend-service-devops.yaml/({) spec/[] ports/({) 0/#port` view. The file contains a Kubernetes Service definition:

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: service-backend-devops
5    namespace: haproxy-controller-devops
6    labels:
7      run: ingress-default-backend
8  spec:
9    selector:
10     run: ingress-default-backend
11    ports:
12     - port: 8080
13       name: port-backend
```

Below the editor, the `TERMINAL` tab is active, showing the following commands and output:

```
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$ kubectl apply -f backend-service-devops.yaml
service/service-backend-devops unchanged
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes$
```

The status bar at the bottom indicates the current file is `gcr.io/k8s-minikube/kicbase: minikube` and the cursor is at `Ln 12, Col 15` with `Spaces: 2`.

**Q7)** Create a deployment for frontend which should be named haproxy-ingress-devops under the same namespace.

replicas=1,

labels "run=haproxy-ingress".

container name=ingress-container-devops

service account=haproxy-service-account-devops

image=haproxytech/kubernetes-ingress, give args as

--default-backend-service=haproxy-controller-devops/service-backend-devops,

resources

requests

cpu=500m

memory=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

third port name should be stat its containerPort should be 1024.

Define environment as TZ=Etc/UTC,

POD\_NAME its

valueFrom:

fieldRef:

fieldPath: metadata.name

POD\_NAMESPACE its

valueFrom:

fieldRef:

fieldPath: metadata.namespace.



```
io.k8s.api.apps.v1.Deployment (v1@deployment.json)
1 ##### frontend Deployment Definition #####
2 apiVersion: apps/v1
3 kind: Deployment
4 metadata:
5   name: haproxy-ingress-devops
6   namespace: haproxy-controller-devops
7   labels:
8     run: haproxy-ingress
9 spec:
10  selector:
11    matchLabels:
12      run: haproxy-ingress
13  template:
14    metadata:
15      labels:
16        run: haproxy-ingress
17    spec:
18      containers:
19        - name: ngress-container-devops
20          image: haproxytech/kubernetes-ingress
21          args:
22            - "--default-backend-service=haproxy-controller-devops/service"
23          env:
24            - # variable 1 for TZ (path)
25              name: TZ
26              value: Etc/UTC
27            - # variable 2 for the pod name
28              name: POD_NAME
29
30  # variable 2 for the pod name
31  - name: POD_NAME
32    valueFrom:
33      fieldRef:
34        fieldPath: metadata.name
35
36  # variable 3 for the pod namespace
37  - name: POD_NAMESPACE
38    valueFrom:
39      fieldRef:
40        fieldPath: metadata.namespace
41
42  ports:
43    - name: http
44      containerPort: 80
45    - name: https
46      containerPort: 443
47    - name: stat
48      containerPort: 1024
49
50  livenessProbe:
51    httpGet:
52      path: /healthz
53      port: 1024
54
55  resources:
56    limits:
57      memory: "50Mi"
58      cpu: "500m"
59
```

mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes\$ kubectl apply -f frontend-deployment-devops.yaml  
deployment.apps/haproxy-ingress-devops unchanged  
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes\$

Q8) Create a service for frontend which should be named as ingress-service-devops under same namespace, 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.

```
io.k8s.api.core.v1.Service (v1@service.json)
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: Ingress-service-devops
5   namespace: haproxy-controller-devops
6   labels:
7     run: haproxy-ingress
8 spec:
9  selector:
10    run: haproxy-ingress
11  ports:
12    - name: http
13      port: 80
14      protocol: TCP
15      targetPort: 80
16      nodePort: 32456
17
18    - name: https
19      port: 443
20      protocol: TCP
21      targetPort: 443
22      nodePort: 32567
23
24    - name: stat
25      port: 1024
26      protocol: TCP
27      targetPort: 1024
28      nodePort: 32678
29  type: NodePort
30
31
```

mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes\$ code frontend-service-devops.yaml  
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes\$ kubectl apply -f frontend-service-devops.yaml  
Service/Ingress-service-devops unchanged  
mostafa@mostafa-Predator:~/Desktop/iti/Kubernetes\$

