Aufgabe 1. kubectl create namespace qa-team
dev team

Aufgabe 2. kubectl create serviceaccount test-team-sa -n test-team
qa-team-sa ~~n qa~~team

Aufgabe 3. Role

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: test-team
  name: test-team-role
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
- apiGroups: [""]
  resources: ["configmaps"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

kubectl apply -f test-team-role.yaml

Aufgabe 4. RoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: test-team-rolebinding
  namespace: test-team
subjects:
- kind: ServiceAccount
  name: test-team-sa
  namespace: test-team
roleRef:
  kind: Role
  name: test-team-role
  apiGroup: rbac.authorization.k8s.io
```

kubectl apply -f test-team-rolebinding-sa.yaml

Check*

```
kubectl auth can-i create deployments --namespace=test-team --as=system:serviceaccount:test-team:test-team-sa
kubectl auth can-i update deployments --namespace=test-team --as=system:serviceaccount:test-team:test-team-sa
kubectl auth can-i create configmaps --namespace=test-team --as=system:serviceaccount:test-team:test-team-sa
kubectl auth can-i update configmaps --namespace=test-team –as=system:serviceaccount:test-team:test-team-sa
```

Aufgabe 5. ClusterRole QA

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: qa-team-clusterrole
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get", "list"]
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

```
kubectl apply -f qa-team-clusterrole.yaml
```

Aufgabe 6. ClusterRoleBinding QA

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: qa-team-clusterrolebinding
subjects:
- kind: ServiceAccount
  name: qa-team-sa
  namespace: qa-team
roleRef:
  kind: ClusterRole
  name: qa-team-clusterrole
  apiGroup: rbac.authorization.k8s.io
```

```
kubectl apply -f qa-team-clusterrolebinding.yaml
```

Check

```
kubectl auth can-i list pods --all-namespaces --as=system:serviceaccount:qa-team:qa-team-sa
kubectl auth can-i get pods --all-namespaces –as=system:serviceaccount:qa-team:qa-team-sa
```

Aufgabe 7. Context

```
# Create the context for the QA team
kubectl config set-context qa-team-context \
  --user=system:serviceaccount:qa-team:qa-team-sa \
  --cluster=minikube \
  --namespace=qa-team

# Create the context for the Test team
kubectl config set-context test-team-context \
  --user=system:serviceaccount:test-team:test-team-sa \
  --cluster=minikube \
  --namespace=test-team
```

Aufgabe 8.

a) Switch to test context

*****Make sure secret for test-team-sa is configured*****

* Check secret:

```
kubectl get sa test-team-sa -n test-team
```

*If no secret: create test-team-sa-token.yaml:

```
apiVersion: v1
kind: Secret
metadata:
  name: test-team-sa-token
  namespace: test-team
  annotations:
    kubernetes.io/service-account.name: test-team-sa
type: kubernetes.io/service-account-token
```

```
kubectl apply -f test-team-sa-token.yaml
```

**Check:

```
kubectl get secrets -n test-team
TOKEN=$(kubectl get secret test-team-sa-token -n test-team -o jsonpath='{.data.token}' | base64 --
decode)
echo $TOKEN
```

**Set with token:

```
kubectl config set-credentials test-team-sa –token=$TOKEN
```

**Modify context:

```
kubectl config set-context test-team-context --cluster=minikube –namespace=test-team
```

**Switch

```
kubectl config use-context test-team-context
```

******

test-deployment.yaml:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-deployment
spec:
  replicas: 1
  selector:
    matchLabels:
      app: test-app
  template:
    metadata:
      labels:
        app: test-app
    spec:
      containers:
      - name: test-container
```

```
        image: nginx:latest
        ports:
- containerPort: 80
```

*Modify test-team-role:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: test-team
  name: test-team-role
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
- apiGroups: [""]
  resources: ["configmaps"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

b) Delete deployment:

```
kubectl delete deployment test-deployment -n test-team
```

c) test config map

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-configmap
data:
  key1: value1
  key2: value2
```

```
kubectl apply -f test-configmap.yaml -n test-team
```

d) Verify the update:

```
kubectl describe configmap test-configmap -n test-team
kubectl get configmap test-configmap -n test-team -o yaml
```

e) test-team need clusterrole to create namespace, so switch to mikube first

```
kubectl create namespace test-team-2
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: test-team-2
  name: test-team-2-role
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
- apiGroups: [""]
  resources: ["configmaps"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]

apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: test-team-2-rolebinding
  namespace: test-team-2
subjects:
- kind: ServiceAccount
  name: test-team-sa
  namespace: test-team
roleRef:
  kind: Role
  name: test-team-2-role
  apiGroup: rbac.authorization.k8s.io

kubectl apply -f test-team-2-role.yaml
kubectl apply -f test-team-2-rolebinding.yaml

kubectl config use-context test-team-context

*deployment for team2
apiVersion: apps/v1
kind: Deployment
metadata:
  name: test-deployment
  namespace: test-team-2
spec:
  replicas: 2
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp-container
        image: nginx:latest
        ports:
```

```
- containerPort: 80
```

```
kubectl apply -f test-team-2-deployment.yaml –context=test-team-context
kubectl get deployments -n test-team-2 –context=test-team-context
```

f) Make sure to have secrets and roles

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: qa-deployment
  namespace: test-team-2
spec:
  replicas: 2
  selector:
    matchLabels:
      app: qaapp
  template:
    metadata:
      labels:
        app: qaapp
    spec:
      containers:
      - name: qaapp-container
        image: nginx:latest
        ports:
- containerPort: 80
```

g) kubectl get pods --all-namespaces –context=qa-team-context
h) kubectl get deployments --all-namespaces --context=qa-team-context