j1. Create a project with different user groups and implement group policies.
Answer:

Create a Namespace (Project)

```
kubectl create namespace project-team
```

Define User Groups (Conceptually)

- `dev-team`
- `qa-team`

Create Roles for Groups

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

Role for `qa-team` (Read-only access)

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: project-team
  name: qa-role
rules:
  - apiGroups: [""]
    resources: ["pods", "services"]
    verbs: ["get", "list", "watch"]
```

Bind Roles to Groups

RoleBinding for `dev-team`

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: dev-team-binding
  namespace: project-team
subjects:
  - kind: Group
    name: dev-team         # Name from your auth provider (e.g., Azure
AD group)
    apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: dev-role
  apiGroup: rbac.authorization.k8s.io
```

RoleBinding for `qa-team`

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: qa-team-binding
  namespace: project-team
subjects:
  - kind: Group
    name: qa-team
    apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: Role
  name: qa-role
  apiGroup: rbac.authorization.k8s.io
```

Apply All YAMLs

```
kubectl apply -f dev-role.yaml
kubectl apply -f qa-role.yaml
kubectl apply -f dev-rolebinding.yaml
kubectl apply -f qa-rolebinding.yaml
```

2. Apply branch policies such that only the project administrator can access the master branch; contributors cannot.
Answer: