**Kubernetes RBAC Role-Based Access Control**

**technol Inc. recently migrated its e-commerce platform to Kubernetes for better scalability and deployment efficiency. However, as the development team expanded, there was a growing concern about security and resource access control.**

**Previously, all developers had full access to Kubernetes resources, leading to:**

* **Accidental pod deletions during debugging.**
* **Unauthorized deployments, causing production outages.**

**The CTO has mandated a Role-Based Access Control (RBAC) policy to improve security by limiting what users can do within the cluster.**

**Task**

You are tasked with building a secure **e-commerce** application on Kubernetes where two types of users need access to pods:

1. **Admin**: Full access to create, delete, and modify pods.
2. **Developer**: Read-only access to list and view pods.

The goal is to implement **Role-Based Access Control (RBAC)** to enforce these permissions. The **ecommerce** namespace should isolate resources from the default namespace

**Objectives**

Create a **Kubernetes Cluster** where **Role-Based Access Control (RBAC)** restricts non-admin users from creating, deleting, or modifying pods. Only users with the **developer** role should have limited read-only access, while **admin** users retain full control.

**Verify Installations:**

* docker --version
* kubectl version --client
* minikube version

**Start a Kubernetes Cluster**

* minikube start --driver=docker --cpus=4 --memory=8192

**Verify Cluster is Running:**

* minikube status
* kubectl get nodes

**Project structure**

kubernetes-rbac-project/

|── namespace.yaml # Defines the ecommerce namespace

|── service-account.yaml # Defines two service accounts

|── roles/

│ |── admin-role.yaml # Full access role

│ └── developer-role.yaml # Read-only role

|── role-bindings/

│ |── admin-binding.yaml # Binds admin role to admin-user

│ |── developer-binding.yaml # Binds developer role to developer-user

|── test/

│ |── verify-access.sh # Shell script to test permissions

│ |── deploy-nginx.sh # Deploy Nginx pod

│ └── check-pods.sh # Check running pods

└── README.md

**You are supposed to finish these task on the Kubernetes Cluster**

**1. Create Namespace:**

Q1: Create a namespace called ecommerce to isolate Kubernetes resources.

Hint: Use YAML or kubectl commands.

2. **Create Service Accounts:**

Q2: Create two service accounts in the ecommerce namespace:

admin-user for full access.

developer-user for read-only access.

Hint: Service accounts are essential for authentication in RBAC.

3. **Define RBAC Roles:**

Q3: Define two RBAC roles in the ecommerce namespace:

admin-role: Full access to pods, services, and deployments.

developer-role: Read-only access to pods.

Hint: Roles define a set of permissions on Kubernetes resources.

4. **Bind Roles to Service Accounts:**

Q4: Create RoleBindings to associate the roles with respective service accounts.

admin-role to admin-user

developer-role to developer-user

Hint: RoleBindings link Roles to Service Accounts.

5. **Verify RBAC Permissions:**

Q5: Test the following scenarios:

* Can the admin-user create a pod?
* Can the developer-user delete a pod?
* Can the developer-user list all running pods?
* Hint: Use kubectl auth can-i commands.

**Verify RBAC Permissions**

**Admin User (Full Access)**

kubectl auth can-i create pods --as=system:serviceaccount:ecommerce:admin-user -n ecommerce

**Expected Output:**

yes

**Developer User (Read-Only)**

kubectl auth can-i delete pods --as=system:serviceaccount:ecommerce:developer-user -n ecommerce

**Expected Output:**

no

**Step 6: Test Pods Creation and Deletion**

**Admin Can Create Pod:**

kubectl run nginx --image=nginx --restart=Never -n ecommerce --as=system:serviceaccount:ecommerce:admin-user

**Developer Cannot Create Pod:**

kubectl run nginx --image=nginx --restart=Never -n ecommerce --as=system:serviceaccount:ecommerce:developer-user

**Expected Output:**

Error from server (Forbidden): pods is forbidden