**Bonus Challenge: Label-Based Access Control**

**Analysis**

**Problem**: Standard Kubernetes RBAC cannot filter by labels. RBAC controls access to **resource types**, not individual resource instances based on their attributes.

**What RBAC CAN do:**

- Allow/deny access to entire resource types (all pods, all deployments)
- Limit by namespace
- Limit by verb (get, create, delete)

**What RBAC CANNOT do:**

- Filter by labels
- Filter by specific field values
- Conditional access based on resource content

**Solution Approach: Admission Controllers**

**OPA (Open Policy Agent) Gatekeeper** or **Kyverno** can be used to enforce label requirements.

**How it works:**

1. RBAC grants permission to create/delete pods
2. Admission controller validates that pods have required label
3. If label is missing, admission controller rejects the request

Hence, I have decided to go with **Kyverno**.

## Implementation with Kyverno:

## Step 1: Install Kyverno

```
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC> kubectl create -f https://github.com/kyverno/kyverno/releases/download/v1.10.0/install.yaml
namespace/kyverno created
serviceaccount/kyverno-admission-controller created
serviceaccount/kyverno-background-controller created
serviceaccount/kyverno-cleanup-controller created
serviceaccount/kyverno-cleanup-jobs created
serviceaccount/kyverno-reports-controller created
configmap/kyverno created
configmap/kyverno-metrics created
Warning: unrecognized format "int64"
customresourcedefinition.apiextensions.k8s.io/backgroundscanreports.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/cleanuppolicies.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/clusteradmissionreports.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/clusterbackgroundscanreports.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/clustercleanuppolicies.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/clusterpolicies.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/policies.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/policyexceptions.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/updaterequests.kyverno.io created
customresourcedefinition.apiextensions.k8s.io/clusterpolicyreports.wgpolicyk8s.io created
customresourcedefinition.apiextensions.k8s.io/policyreports.wgpolicyk8s.io created
clusterrole.rbac.authorization.k8s.io/kyverno:admission-controller created
```

## Step 2: Create ServiceAccount with pod permissions

part5 > Label_based_Access_Control > ! automation-sa.yaml

```yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: automation-sa
  namespace: default
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: automation-role
  namespace: default
rules:
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["create", "delete", "get", "list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: automation-binding
  namespace: default
subjects:
  - kind: ServiceAccount
    name: automation-sa
    namespace: default
roleRef:
  kind: Role
  name: automation-role
  apiGroup: rbac.authorization.k8s.io
```

## Step 3: Create Kyverno Policy

Label_based_Access_Control > ! enforce-automation-label.yaml

```yaml
apiVersion: kyverno.io/v1
kind: ClusterPolicy
metadata:
  name: require-automation-label
spec:
  # Enforce = block requests that violate the policy
  validationFailureAction: enforce

  # Disable background mode so the policy only applies to NEW requests
  background: false

  rules:
    - name: check-automation-label

      # MATCH section defines WHO + WHAT this policy applies to
      match:
        # Match only requests made by the ServiceAccount automation-sa
        subjects:
          - kind: ServiceAccount
            name: automation-sa
            namespace: default

        # Match only Pod resources (create/update/delete requests on Pods)
        resources:
          kinds:
            - Pod

      # VALIDATION section defines WHAT must be true
      validate:
        # Message returned when validation fails
        message: "Pods created by automation-sa must have label 'managed-by: automation'"

        # REQUIRED label pattern
        pattern:
          metadata:
            labels:
              managed-by: "automation"
```

## Step 4: Test the policy

## Test 1: Create pod WITHOUT required label (failed)

```
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC\Label_based_Access_Control> kubectl apply -f test-pod-no-label.yaml --as=system:serviceaccount:default:automation-sa
Error from server: error when creating "test-pod-no-label.yaml": admission webhook "validate.kyverno.svc-fail" denied the request:

resource Pod/default/test-no-label was blocked due to the following policies

require-automation-label:
  check-automation-label: 'validation error: Pods created by automation-sa must have
    label ''managed-by: automation''. rule check-automation-label failed at path /metadata/labels/'
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC\Label_based_Access_Control>
```

## Test 2: Create pod WITH required label (succeed)

```
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC\Label_based_Access_Control> kubectl apply -f test-pod-with-label.yaml --as=sys
tem:serviceaccount:default:automation-sa
pod/test-with-label created
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC\Label_based_Access_Control> kubectl get pod test-with-label
NAME              READY   STATUS    RESTARTS   AGE
test-with-label   1/1     Running   0          7s
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC\Label_based_Access_Control>
```

## Test 3: Delete the labeled pod (succeed)

```
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC\Label_based_Access_Control> kubectl apply -f test-pod-with-label.yaml --as=sys
tem:serviceaccount:default:automation-sa
pod/test-with-label created
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC\Label_based_Access_Control> kubectl get pod test-with-label
NAME              READY   STATUS    RESTARTS   AGE
test-with-label   1/1     Running   0          18s
PS C:\Users\LEGION\Desktop\Fall 2025 Sem\AWS dev Ops\COMP-488_Assignment_RBAC\Label_based_Access_Control> kubectl delete pod test-with-label --as=system:ser
viceaccount:default:automation-sa
pod "test-with-label" deleted
```