Role Based Access Control - Kubernetes

In **Kubernetes**, **RBAC (Role-Based Access Control)** is a system for regulating user and service account permissions. Let's break it down for the four roles you've mentioned — **Builder**, **Deployer**, **Developer**, and **Admin** — based on typical usage in OpenShift or Kubernetes environments, and how their **RBAC privileges** are generally assigned.

Role Overview & RBAC Privileges

Role	Typical Use Case	Kubernetes/Openshift RBAC Privileges
Builder	Automates image builds or CI pipelines	`create`, `update`, `get`, `watch` on `builds`, `buildconfigs`, `imagestreams`, `pods`
Deployer	Deploys apps, often as a service account	`create`, `update`, `patch`, `get`, `watch` on `deployments`, `replicasets`, `pods`, `secrets` (read-only)
Developer	Develops, runs, and debugs apps	`get`, `create`, `list`, `update`, `delete` on `pods`, `services`, `deployments`, `routes`, `configmaps`, `secrets` (non-sensitive), `ingresses`
Admin	Full control over namespace or cluster	`*` (all verbs) on all resources within a namespace or the cluster

---

Detailed Explanation

1. Builder

- **Used by**: CI tools (e.g., Jenkins, Tekton), OpenShift `BuildConfig`
- **Typical Actions**:
    - Launching builds (`builds`, `buildconfigs`)
    - Accessing source image streams

RBAC Example:

```
 kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: builder
rules:
  - apiGroups: ["build.openshift.io"]
    resources: ["builds", "buildconfigs"]
    verbs: ["get", "list", "create", "update"]
  - apiGroups: ["image.openshift.io"]
    resources: ["imagestreams"]
    verbs: ["get", "list"]
```

-

---

2. Deployer

- **Used by**: GitOps tools (e.g., Argo CD), deployment scripts, service accounts
- **Typical Actions**:
    - Managing deployments and rollouts
    - Reading secrets (not modifying them)

RBAC Example:

```
 kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: deployer
rules:
  - apiGroups: ["apps"]
    resources: ["deployments", "replicasets"]
    verbs: ["create", "update", "get", "list", "watch"]
  - apiGroups: [""]
    resources: ["pods", "secrets"]
    verbs: ["get", "list"]
```

-

---

3. Developer

- **Used by**: Application engineers and frontend/backend developers
- **Typical Actions**:
    - Creating and modifying services, pods, config maps
    - Debugging applications

RBAC Example:

```
 kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: developer
rules:
  - apiGroups: [""]
    resources: ["pods", "services", "configmaps", "secrets"]
    verbs: ["get", "list", "create", "update", "delete"]
  - apiGroups: ["apps"]
    resources: ["deployments"]
    verbs: ["create", "update", "delete", "get", "list"]
```

-

---

4. Admin

- **Used by**: Cluster operators, DevOps teams
- **Typical Actions**:
    - Full namespace management
    - Assigning roles to others
    - Creating RBAC bindings

RBAC Example:

```
 kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: admin
rules:
  - apiGroups: ["*"]
    resources: ["*"]
    verbs: ["*"]
```

- In OpenShift, `admin`, `edit`, `view` are pre-defined roles. The same pattern can be used in Kubernetes with `Role` or `ClusterRole`.

Summary

Role	Scope	Can Edit RBAC?	Has Cluster Access?
Builder	Namespace	❌	❌
Deployer	Namespace	❌	❌
Developer	Namespace	❌	❌
Admin	Namespace / Cluster	✅	✅