**Lab Exercise 23- Custom Resources in Kubernetes**

A **Custom Resource (CR)** in Kubernetes allows users to define and manage custom application configurations or resources. By extending the Kubernetes API, custom resources let you define your domain-specific resources using familiar Kubernetes patterns.

To manage custom resources, Kubernetes uses **Custom Resource Definitions (CRDs)**, which you must create first. These CRDs act as the schema for your custom resources.

**Why Use Custom Resources?**

1. Extend Kubernetes to manage application-specific objects.
2. Create higher-level abstractions for your applications or infrastructure.
3. Automate and standardize the management of complex workloads.

**Steps to Create and Use a Custom Resource**

**1. Define a Custom Resource Definition (CRD)**

A CRD is used to define the schema for your custom resource.

**Example CRD (YAML):**

apiVersion: apiextensions.k8s.io/v1

kind: CustomResourceDefinition

metadata:

name: myresources.example.com

spec:

group: example.com # Define API group

names:

kind: MyResource # Custom resource kind

listKind: MyResourceList # Custom resource list kind

plural: myresources # Plural name

singular: myresource # Singular name

scope: Namespaced # Can be "Cluster" or "Namespaced"

versions:

- name: v1 # API version

served: true

storage: true

schema:

openAPIV3Schema:

type: object

properties:

spec:

type: object

properties:

replicas:

type: integer

image:

type: string

* Save this file as crd.yaml and apply it using:

kubectl apply -f crd.yaml

**2. Create a Custom Resource**

After defining the CRD, you can create instances of your custom resource.

**Example Custom Resource (YAML):**

apiVersion: example.com/v1

kind: MyResource

metadata:

name: myresource-sample

spec:

replicas: 3

image: nginx:latest

* Save this file as myresource.yaml and apply it using:

kubectl apply -f myresource.yaml

**3. Interact with the Custom Resource**

* **List Custom Resources:**

kubectl get myresources

* **Describe a Custom Resource:**

kubectl describe myresource myresource-sample

* **Delete a Custom Resource:**

kubectl delete myresource myresource-sample

**Use Case Example: Managing NGINX Deployments**

**CRD:**

apiVersion: apiextensions.k8s.io/v1

kind: CustomResourceDefinition

metadata:

name: nginxresources.example.com

spec:

group: example.com

names:

kind: NginxResource

listKind: NginxResourceList

plural: nginxresources

singular: nginxresource

scope: Namespaced

versions:

- name: v1

served: true

storage: true

schema:

openAPIV3Schema:

type: object

properties:

spec:

type: object

properties:

replicas:

type: integer

image:

type: string

**Custom Resource:**

apiVersion: example.com/v1

kind: NginxResource

metadata:

name: nginx-example

spec:

replicas: 2

image: nginx:latest

**Benefits of Using Custom Resources**

1. **Abstraction**: Simplifies complex configurations by abstracting them into domain-specific objects.
2. **Extensibility**: Adds new capabilities to Kubernetes without modifying its core.
3. **Automation**: Combine with controllers to automate application lifecycle management.