**Kubernetes Custom Resources (CRDs) - Notes**

**1. Introduction to Custom Resources**

* Kubernetes provides **built-in resources** like Deployments, Services, Pods, ConfigMaps, and Secrets.
* **Custom Resources (CRs)** extend Kubernetes by introducing **new API objects** (e.g., Istio's VirtualService, ArgoCD's Application).
* **Why CRDs?**
  + Kubernetes lacks built-in support for every use case (e.g., GitOps, security, service mesh).
  + **Custom Resource Definitions (CRDs)** enable extending Kubernetes API to create new resource types.

**2. Key Components of Custom Resources**

**A. Custom Resource Definition (CRD)**

* **What is a CRD?**
  + A **YAML file** defining a **new Kubernetes resource type**.
  + Example: Istio defines VirtualService CRD for traffic routing.
* **Purpose:**
  + **Extends Kubernetes API** to recognize new resources.
  + **Validates** Custom Resources (CRs) before acceptance.

**B. Custom Resource (CR)**

* **What is a CR?**
  + An **instance** of a CRD (similar to how a Deployment is an instance of apps/v1).
  + Example: A user defines a VirtualService CR to configure traffic rules.
* **Who creates it?**
  + **Developers/DevOps engineers** use CRs to apply new functionalities (e.g., Istio, ArgoCD).

**C. Custom Controller**

* **What is a Custom Controller?**
  + A **program** that **monitors** CRs and executes predefined logic.
* **How it works:**
  + Detects CR creation, modification, or deletion.
  + Performs actions based on CR specifications (e.g., Istio configures proxies).

**3. How CRDs Work**

**Step-by-Step Flow:**

1. **Deploy CRD** (e.g., VirtualService CRD for Istio).
   * Installed using kubectl apply, Helm, or Operators.
2. **User creates a CR** (e.g., my-virtual-service.yaml).
   * Kubernetes **validates** the CR against the CRD schema.
3. **Custom Controller detects the CR** and executes required actions.
   * Example: Istio controller applies VirtualService traffic rules.

**Comparison: Native vs. Custom Resources**

| **Native Resource (e.g., Deployment)** | **Custom Resource (e.g., VirtualService)** |
| --- | --- |
| Built into Kubernetes (apps/v1). | Defined via CRD (networking.istio.io/v1alpha3). |
| Managed by built-in controllers. | Managed by custom controllers (e.g., Istio). |

**4. Writing Custom Controllers**

* **Language:** Primarily **Golang** (client-go, controller-runtime).
* **Key Steps:**
  1. **Define CRD** (YAML).
  2. **Write Controller Logic** (watch for CRs, process events).
  3. **Deploy Controller** (run as a Kubernetes Pod).
* **Popular Frameworks:**
  1. operator-sdk (for Kubernetes Operators).
  2. kubebuilder (for building controllers).

**5. Example: Istio Installation (CRD + Controller)**

1. **Install CRDs:**
2. helm repo add istio https://istio-release.storage.googleapis.com/charts
3. helm install istio-base istio/base -n istio-system
4. **Install Istio Controller (istiod)**:
5. helm install istiod istio/istiod -n istio-system
6. **Create a VirtualService CR:**
7. apiVersion: networking.istio.io/v1alpha3
8. kind: VirtualService
9. metadata:
10. name: my-virtual-service
11. spec:
12. hosts: ["my-app.example.com"]
13. http:
14. - route:
15. - destination:
16. host: my-app

**6. Key Takeaways**

✅ **CRDs extend Kubernetes API** (introduce new resource types).  
✅ **Custom Resources (CRs)** are user-defined instances of CRDs.  
✅ **Custom Controllers** monitor CRs and implement logic.  
✅ **DevOps engineers deploy CRDs & Controllers**; **users create CRs**.  
✅ **To write controllers, use Golang (client-go, controller-runtime).**

📌 **Action Item:**

* Install **Istio/ArgoCD** and explore their CRDs (kubectl get crd).

🔗 **Next Steps:** Learn about **Kubernetes Operators** (advanced controllers).

**Stay tuned for more DevOps & Kubernetes content!** 🚀

# \*\*Kubernetes Custom Resources (CRDs) - Notes\*\*

## \*\*1. Introduction to Custom Resources in Kubernetes\*\*

- Kubernetes comes with \*\*built-in resources\*\* like Deployments, Services, Pods, ConfigMaps, and Secrets.

- \*\*Custom Resources (CRs)\*\* extend Kubernetes’ functionality by adding \*\*new API objects\*\* (e.g., Istio’s `VirtualService`, ArgoCD’s `Application`).

- \*\*Why Custom Resources?\*\*

- Kubernetes cannot natively support every possible use case (security, GitOps, service mesh, etc.).

- \*\*CRDs (Custom Resource Definitions)\*\* allow \*\*extending Kubernetes API\*\* to introduce new resource types.

---

## \*\*2. Key Components of Custom Resources\*\*

### \*\*A. Custom Resource Definition (CRD)\*\*

- \*\*What is a CRD?\*\*

- A \*\*YAML file\*\* that defines a \*\*new type of resource\*\* in Kubernetes.

- Example: Istio defines a `VirtualService` CRD to manage traffic routing.

- \*\*Purpose:\*\*

- \*\*Extends Kubernetes API\*\* to support new resource types.

- \*\*Validates\*\* user-submitted Custom Resources (CRs).

### \*\*B. Custom Resource (CR)\*\*

- \*\*What is a CR?\*\*

- An \*\*instance\*\* of a CRD (like a `Deployment` is an instance of the `apps/v1` API).

- Example: A user creates a `VirtualService` CR to configure Istio routing rules.

- \*\*Who creates it?\*\*

- \*\*Users (Developers/DevOps)\*\* define CRs to use extended features (e.g., Istio, ArgoCD).

### \*\*C. Custom Controller\*\*

- \*\*What is a Custom Controller?\*\*

- A \*\*program\*\* that \*\*watches\*\* for CRs and \*\*executes logic\*\* (e.g., Istio controller applies traffic rules).

- \*\*How it works:\*\*

- Watches for CRs (create/update/delete events).

- Takes action based on CR specifications (e.g., configures Istio proxies).

---

## \*\*3. How CRDs Work (Step-by-Step Flow)\*\*

1. \*\*DevOps Engineer deploys CRD\*\* (e.g., `VirtualService` CRD for Istio).

- Done via `kubectl apply`, Helm, or Operators.

2. \*\*User creates a Custom Resource (CR)\*\* (e.g., `my-virtual-service.yaml`).

- Kubernetes \*\*validates\*\* the CR against the CRD.

3. \*\*Custom Controller detects the CR\*\* and performs actions.

- Example: Istio controller configures Envoy proxies based on `VirtualService`.

### \*\*Comparison with Native Kubernetes Resources\*\*

| \*\*Native Resource (e.g., Deployment)\*\* | \*\*Custom Resource (e.g., VirtualService)\*\* |

|---------------------------------------|-------------------------------------------|

| Built into Kubernetes (`apps/v1`). | Defined via CRD (`networking.istio.io/v1alpha3`). |

| Managed by built-in controllers (Deployment Controller). | Managed by custom controllers (Istio Controller). |

---

## \*\*4. Writing Custom Controllers\*\*

- \*\*Language:\*\* Primarily \*\*Golang\*\* (using `client-go` and `controller-runtime`).

- \*\*Key Steps:\*\*

1. \*\*Define CRD\*\* (YAML).

2. \*\*Write Controller Logic\*\* (watch CRs, process events).

3. \*\*Deploy Controller\*\* (as a Pod in the cluster).

- \*\*Popular Frameworks:\*\*

- `operator-sdk` (for Kubernetes Operators).

- `kubebuilder` (for building controllers).

---

## \*\*5. Example: Istio Installation (CRD + Controller)\*\*

1. \*\*Install CRDs:\*\*

```sh

helm repo add istio https://istio-release.storage.googleapis.com/charts

helm install istio-base istio/base -n istio-system

```

2. \*\*Install Custom Controller (Istiod):\*\*

```sh

helm install istiod istio/istiod -n istio-system

```

3. \*\*User creates a `VirtualService` CR:\*\*

```yaml

apiVersion: networking.istio.io/v1alpha3

kind: VirtualService

metadata:

name: my-virtual-service

spec:

hosts: ["my-app.example.com"]

http:

- route:

- destination:

host: my-app

```

---

## \*\*6. Key Takeaways\*\*

✅ \*\*CRDs extend Kubernetes API\*\* (add new resource types).

✅ \*\*Custom Resources (CRs)\*\* are instances of CRDs (users define them).

✅ \*\*Custom Controllers\*\* watch CRs and implement logic (e.g., Istio, ArgoCD).

✅ \*\*DevOps Engineers deploy CRDs & Controllers\*\*; \*\*users create CRs\*\*.

✅ \*\*Writing controllers? Use Golang (`client-go`, `controller-runtime`)\*\*.

📌 \*\*Action Item:\*\*

- Try installing \*\*Istio/ArgoCD\*\* and explore their CRDs (`kubectl get crd`).

🔗 \*\*Next Steps:\*\* Learn about \*\*Kubernetes Operators\*\* (advanced controllers).

---

\*\*Subscribe for more DevOps & Kubernetes content!\*\* 🚀