THE LINUX FOUNDATION | Training & Certification

Exercise 14.1: Create a Custom Resource Definition

> **Overview**
>
> The use of `CustomResourceDefinitions (CRD)`, has become a common manner to deploy new objects and operators. Creation of a new operator is beyond the scope of this course, basically it is a watch-loop comparing a spec to the current status, and making changes until the states match. A good discussion of creating a operators can be found here: https://operatorframework.io/.

First we will examine an existing CRD, then make a simple CRD, but without any particular action. It will be enough to find the object ingested into the API and responding to commands.

1. View the existing CRDs.

    ```
    student@cp:~$ kubectl get crd --all-namespaces
    ```

    ```
    NAME                                        CREATED AT
    bgpconfigurations.crd.projectcalico.org     2020-04-19T17:29:02Z
    bgppeers.crd.projectcalico.org              2020-04-19T17:29:02Z
    blockaffinities.crd.projectcalico.org       2020-04-19T17:29:02Z
    <output_omitted>
    ```

2. We can see from the names that these CRDs are all working on `Calico`, out network plugin. View the `calico.yaml` file we used when we initialized the cluster to see how these objects were created, and some CRD templates to review.

    ```
    student@cp:~$ less calico.yaml
    ```

    ```
    <output_omitted>
    ---
    # Source: calico/templates/kdd-crds.yaml

    apiVersion: apiextensions.k8s.io/v1beta1
    kind: CustomResourceDefinition
    metadata:
      name: bgpconfigurations.crd.projectcalico.org
    <output_omitted>
    ```

3. Now that we have seen some examples, we will create a new YAML file.

    ```
    student@cp:~$ vim crd.yaml
    ```

    YAML — **crd.yaml**

    ```yaml
    1  apiVersion: apiextensions.k8s.io/v1
    2  kind: CustomResourceDefinition
    3  metadata:
    4    # name must match the spec fields below, and be in the form: <plural>.<group>
    5    name: crontabs.stable.example.com
    6  spec:
    7    # group name to use for REST API: /apis/<group>/<version>
    8    group: stable.example.com
    9    # list of versions supported by this CustomResourceDefinition
    10   versions:
    ```

   LINUX FOUNDATION | Training & Certification

```yaml
11      - name: v1
12        # Each version can be enabled/disabled by Served flag.
13        served: true
14        # One and only one version must be marked as the storage version.
15        storage: true
16        schema:
17          openAPIV3Schema:
18            type: object
19            properties:
20              spec:
21                type: object
22                properties:
23                  cronSpec:
24                    type: string
25                  image:
26                    type: string
27                  replicas:
28                    type: integer
29    # either Namespaced or Cluster
30    scope: Namespaced
31    names:
32      # plural name to be used in the URL: /apis/<group>/<version>/<plural>
33      plural: crontabs
34      # singular name to be used as an alias on the CLI and for display
35      singular: crontab
36      # kind is normally the CamelCased singular type. Your resource manifests use this.
37      kind: CronTab
38      # shortNames allow shorter string to match your resource on the CLI
39      shortNames:
40      - ct
```

4. Add the new resource to the cluster.

```
student@cp:~$ kubectl create -f crd.yaml
```

```
customresourcedefinition.apiextensions.k8s.io/crontabs.stable.example.com created
```

5. View and describe the resource. The new line may be in the middle of the output. You'll note the **describe** output is unlike other objects we have seen so far.

```
student@cp:~$ kubectl get crd
```

```
NAME                                CREATED AT
<output_omitted>
crontabs.stable.example.com         2021-06-13T03:18:07Z
<output_omitted>
```

```
student@cp:~$ kubectl describe crd crontab<Tab>
```

```
Name:         crontabs.stable.example.com
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  apiextensions.k8s.io/v1
Kind:         CustomResourceDefinition
<output_omitted>
```

6. Now that we have a new API resource we can create a new object of that type. In this case it will be a crontab-like

image, which does not actually exist, but is being used for demonstration.

```
student@cp:~$ vim new-crontab.yaml
```

```yaml
1  apiVersion: "stable.example.com/v1"
2      # This is from the group and version of new CRD
3  kind: CronTab
4      # The kind from the new CRD
5  metadata:
6    name: new-cron-object
7  spec:
8    cronSpec: "*/5 * * * *"
9    image: some-cron-image
10     #Does not exist
```

7. Create the new object and view the resource using short and long name.

```
student@cp:~$ kubectl create -f new-crontab.yaml
```

```
crontab.example.com/new-cron-object created
```

```
student@cp:~$ kubectl get CronTab
```

```
NAME               AGE
new-cron-object    22s
```

```
student@cp:~$ kubectl get ct
```

```
NAME               AGE
new-cron-object    29s
```

```
student@cp:~$ kubectl describe ct
```

```
Name:         new-cron-object
Namespace:    default
Labels:       <none>
Annotations:  <none>
API Version:  stable.example.com/v1
Kind:         CronTab

<output_omitted>

Spec:
  Cron Spec:  */5 * * * *
  Image:      some-cron-image
Events:       <none>
```

8. To clean up the resources we will delete the CRD. This should delete all of the endpoints and objects using it as well.

```
student@cp:~$ kubectl delete -f crd.yaml
```

```
customresourcedefinition.apiextensions.k8s.io "crontabs.stable.example.com" deleted
```

```
student@cp:~$ kubectl get ct
```

```
Error from server (NotFound): Unable to list "stable.example.com/v1,
Resource=crontabs": the server could not find the requested resource
```

```
(get crontabs.stable.example.com)
```