Extras

» Third Party Resources

Overview

Third Party Resources allow Kubernetes users to extend the Kubernetes API….via the Kubernetes API!

Begin by running a proxy to the Kubernetes API server.

```
kubectl proxy
```

Observe the **thirdpartyresources** *api resource* within the **extensions** *api group*.

```
curl http://localhost:8001/apis/extensions/v1beta1 | jq .resources[].name
```

Let's create a new ThirdPartyResource resource object manifest for databases.

```
cat >> databases-tpr.yaml <<EOF
apiVersion: extensions/v1beta1
kind: ThirdPartyResource
metadata:
  name: databases.example.com
description: "A specification of a SQL database."
versions:
  - name: v1
EOF
```

Create the ThirdPartyResource resource object.

```
kubectl create -f databases-tpr.yaml
```

You should now see the Kubernetes API reflect a brand new *api group* called **example.com**.

```
curl http://localhost:8001/apis | jq .groups[].name
```

Within the example.com group there will an *api version* **v1** (per our tpr resource object). The database resource resides here.

```
curl http://localhost:8001/apis/extensions.com/v1 | jq
```

Notice how `kubectl` now recognizes databases as a present resource (although there will be no current resource objects at this time).

```
kubectl get databases
```

Run the command again using the –v=8 option to see the API transactions. You will see an API call being made to

```
curl http://localhost:8001/apis/example.com/v1/namespaces/default/databases
```

```
kubectl get databases --v=8
```

Let's create a new **resource object** manifest for the database **resource**.

```
cat >> wordpress-database.yaml <<EOF
apiVersion: example.com/v1
kind: Database
metadata:
  name: wordpress
spec:
  user: wp
  password: secret
  foo: bar
EOF
```