Extending Kubernetes using Custom Resource

I'm **Reyhan Sofian Haqqi**

Senior Software Engineer @ Kata.ai

Follow me:
@reyhan_sofian
https://facebook.com/reyhansofian

Talk Overview

- Kubernetes Resources Overview

- Custom Resources Overview

- Custom Resources Features

- How to Custom Resources

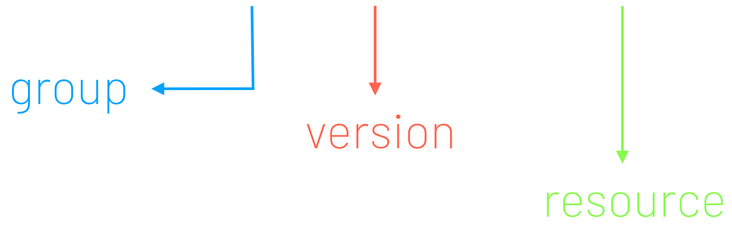
Kubernetes Resources Overview

Kubernetes Objects

Pod, Service, Volume, Namespace, etc

Accessible via REST API

e.g. `/api/apps/v1/deployments`



Persisted in consistent and highly-available key value store

Controller do the reconciliation process

watch both the desired state and the actual state

change the actual state to the desired state

what if we need new k8s feature?

Create a KEP!


@LateNightSeth
BOOM. EASY AS THAT.

but, what if it’s not generic enough
to be a k8s feature?

Custom Resources

an extension of the Kubernetes API

that's not necessarily available in a default Kubernetes installation

CRD makes k8s modular, extensible and maintainable

it keeps k8s core at minimum

History of CRD

March 2016

Third-party Resource (beta)

k8s 1.2

July 2017

CRD (beta)

k8s 1.7

September 2017

Third-party Resource (deprecated)

k8s 1.8

September 2019

CRD (GA 🎉)

k8s 1.16

Who use CRD?


KubeDB
JENKINSX

Custom Resources Features

CLI out of the box

it's your friend, `kubectl` ❤️

HOW TO PRONOUNCE `kubectl`?

Follow declarative k8s API conventions

like `.spec`, `.metadata`, `.status`, etc

Programming language is **not required** for CRD

but you still need to understand YAML tho

Programming language is **required** for CRD controller

you can choose any languages for it

Watches for update on new object (via informers)

then CRUD on other objects or vice versa

Validation

using OpenAPI v3.0 validation

or using Validating Admission Webhook

Defaulting

using Mutating Admission Webhook

auth{n, z}

use k8s auth{n, z} features out of the box

RBAC, ABAC, Webhook, etc


Auth{n, z}
Admission Controller (Validating & Mutating)
API Server

How to Custom Resources

Create CRD

```yaml
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: autoscalers.labs.example.com
spec:
  group: labs.example.com
  versions:
    - name: v1alpha1
      served: true
      storage: true
  scope: Namespaced
  names:
    plural: autoscalers
    singular: autoscaler
    kind: Autoscaler
    shortNames:
    - as
```

1. Name for this CRD. It must be in `<plural>.<group>` format
2. Group name of the API. It's a collection of objects that logically related.
   Pro tip: use FQDN
3. Version of the CRD. It supports multiple versions
4. CRD available on a single namespace (`Namespaced`) or a cluster-wide (`Cluster`)
5. Plural name to be use in the URL
6. Singular name to be use in the CLI and for display
7. To be use in the resource manifests. Usually use CamelCase singular type
8. Shorter name of the resource. Can be used on CLI

it will create REST API endpoint

`/apis/labs.example.com/v1alpha1/namespaces/*/autoscalers/`

group

version

scope

resource

Create Custom Objects

```yaml
apiVersion: labs.example.com/v1alpha1
kind: Autoscaler
metadata:
  name: custom-autoscaler
spec:
  target:
    apiVersion: apps/v1
    kind: Deployment
    name: my-deployment
  minReplicas: 1
  maxReplicas: 30
  metrics:
    - type: PrometheusMetrics
      query: |-
        round(
          sum(irate(nginx_ingress_controller_requests{ingress=~"ingress"}[2m])
        ) by (ingress), 0.001) > 30
```

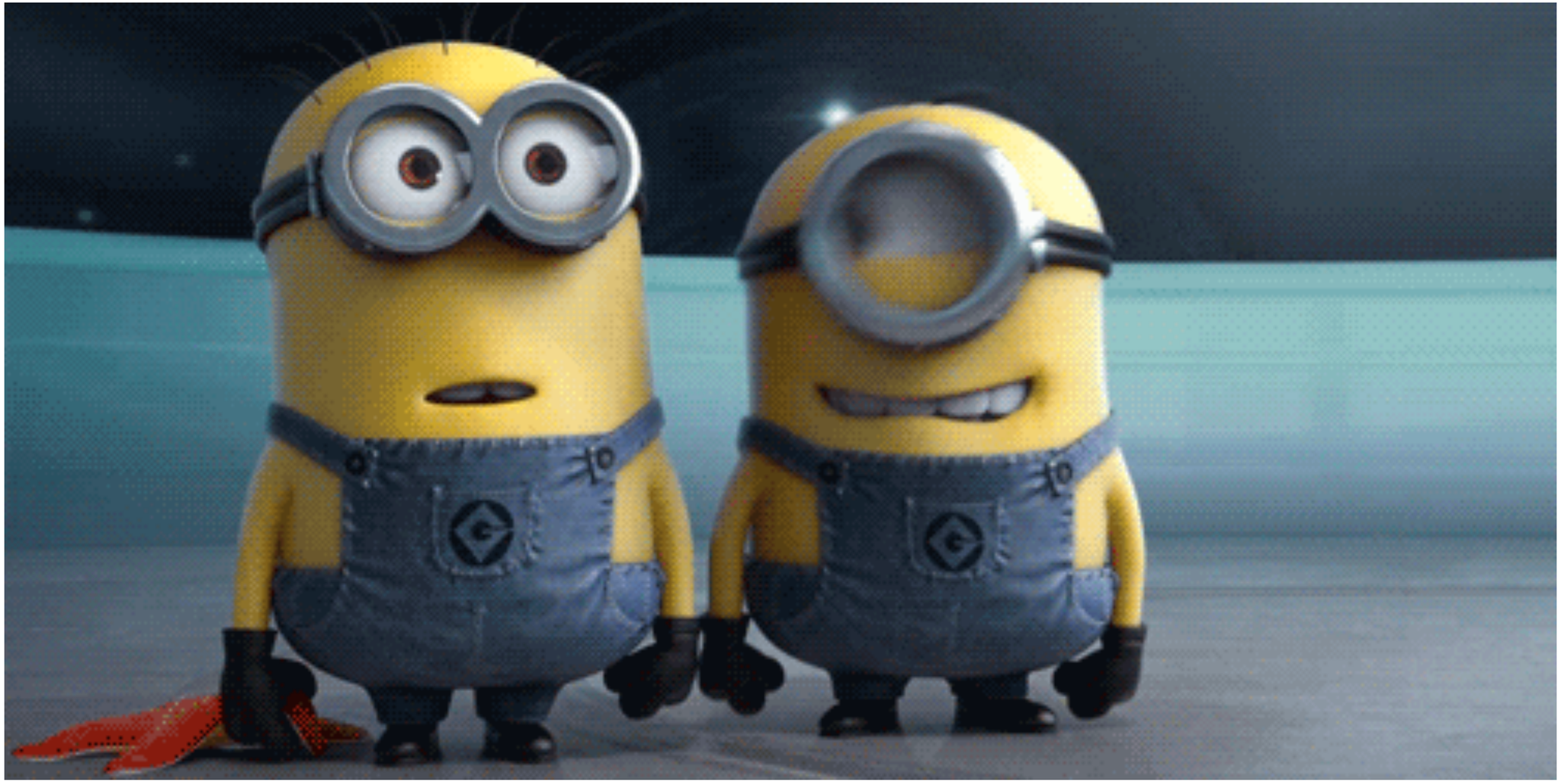
you can get the CRD and CR using CLI

`kubectl get crd,autoscaler`

we can do CRUD to manipulate the CR data

but you need a **custom controller** to make it do the magic

and it’s beyond this talk scope!



References

- Custom Resources Docs: link link

- Kubernetes: Up and Running: link

- Introduction to Writing k8s Controller: link

- Introduction to Operator SDK: link

- Metacontroller: link

- Kubebuilder: link

Thanks!