CRDs Aren’t Just For Addons

KubeCon NA, Seattle
12/2018

Tim Hockin <thockin@google.com>
Principal Software Engineer
@thockin

NOTE: This talk is forward looking

History of CRD (née TPR)

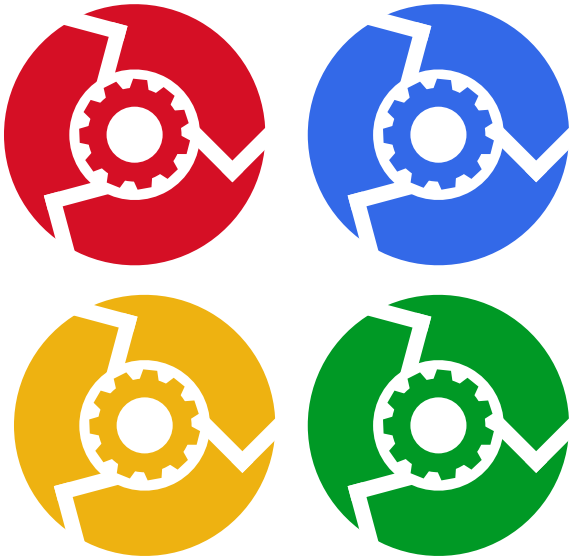
Started as a way to prototype new features for Kubernetes itself, and for users to leverage some of our API machinery

Why?

- Kube-style APIs are simple and powerful
- We already have an API server (and storage)

Limited: No schema, validation, defaulting

Result: a 2nd class experience



Birth of the Operator pattern

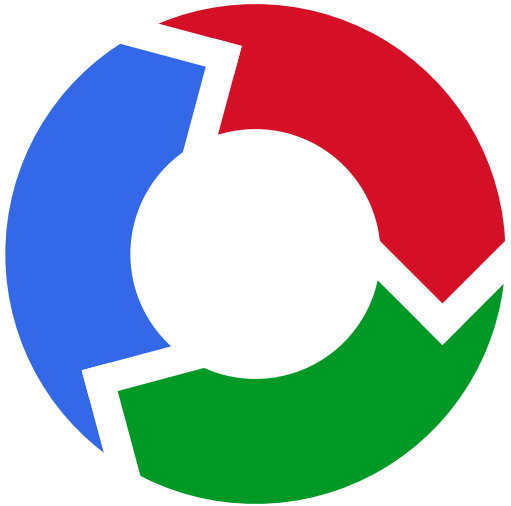
As limited as it was, it was useful

People started creating really interesting software-robots to automate things: “operators”

Custom abstractions to control almost anything

Often used for managing stateful apps

Driven by declarative APIs, actuated asynchronously by controllers



Becoming more native

Webhook admission controllers

- Mutating (set values, e.g. defaults)
- Validating (synchronous input validation)

Schema

- OpenAPI v3 schema definition
- Declarative validation (simple)

Almost all the pieces are in place to make truly native-feeling APIs

Becoming more native

Webhook admission controllers

- Mutating (set values, e.g. defaults)
- Validating (synchronous input validation)

Schema

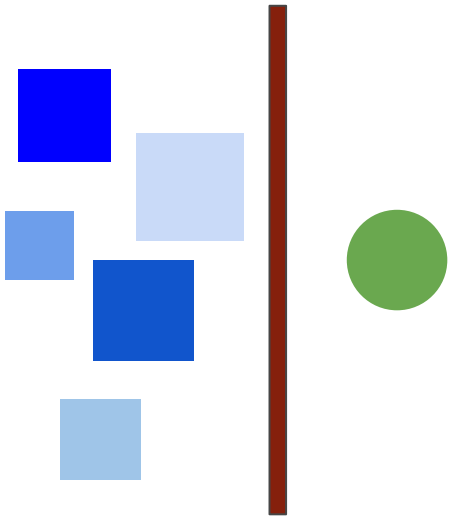
- OpenAPI v3 schema definition
- Declarative validation (simple)

Almost all the pieces are in place to make truly native-feeling APIs
WITHOUT changing Kubernetes code!

The role of CRDs in Kubernetes

Being used for out-of-tree, third-party things

- Add-ons
- Stateful orchestration (e.g. MySQL)
- Domain-specific APIs (e.g. Istio)
- Higher levels of abstraction (e.g. kNative)



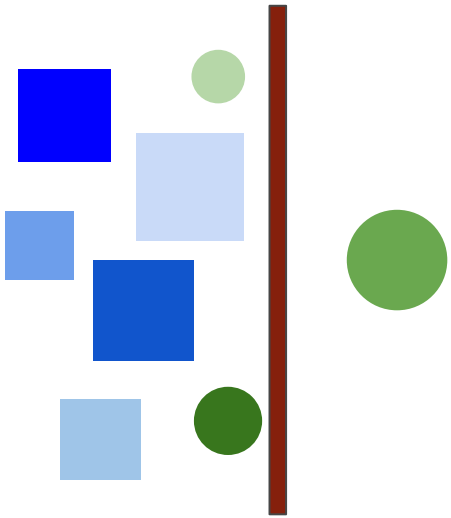
The role of CRDs in Kubernetes

Being used for out-of-tree, third-party things

- Add-ons
- Stateful orchestration (e.g. MySQL)
- Domain-specific APIs (e.g. Istio)
- Higher levels of abstraction (e.g. kNative)

More recently: for in-tree, first-party things!

- Storage Snapshots (integrates w/ PersistentVolumes)
- RuntimeClass (integrates w/ Kubelet and CRI)
- CSI
- Expect more...



CRDs are no longer 2nd class

Vision

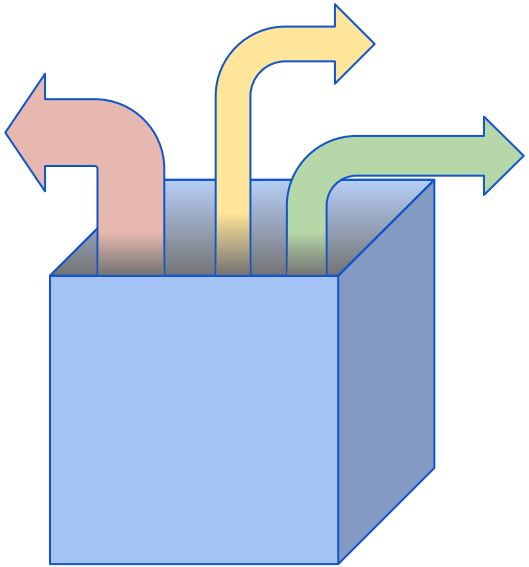
Now: most/all new APIs are CRDs

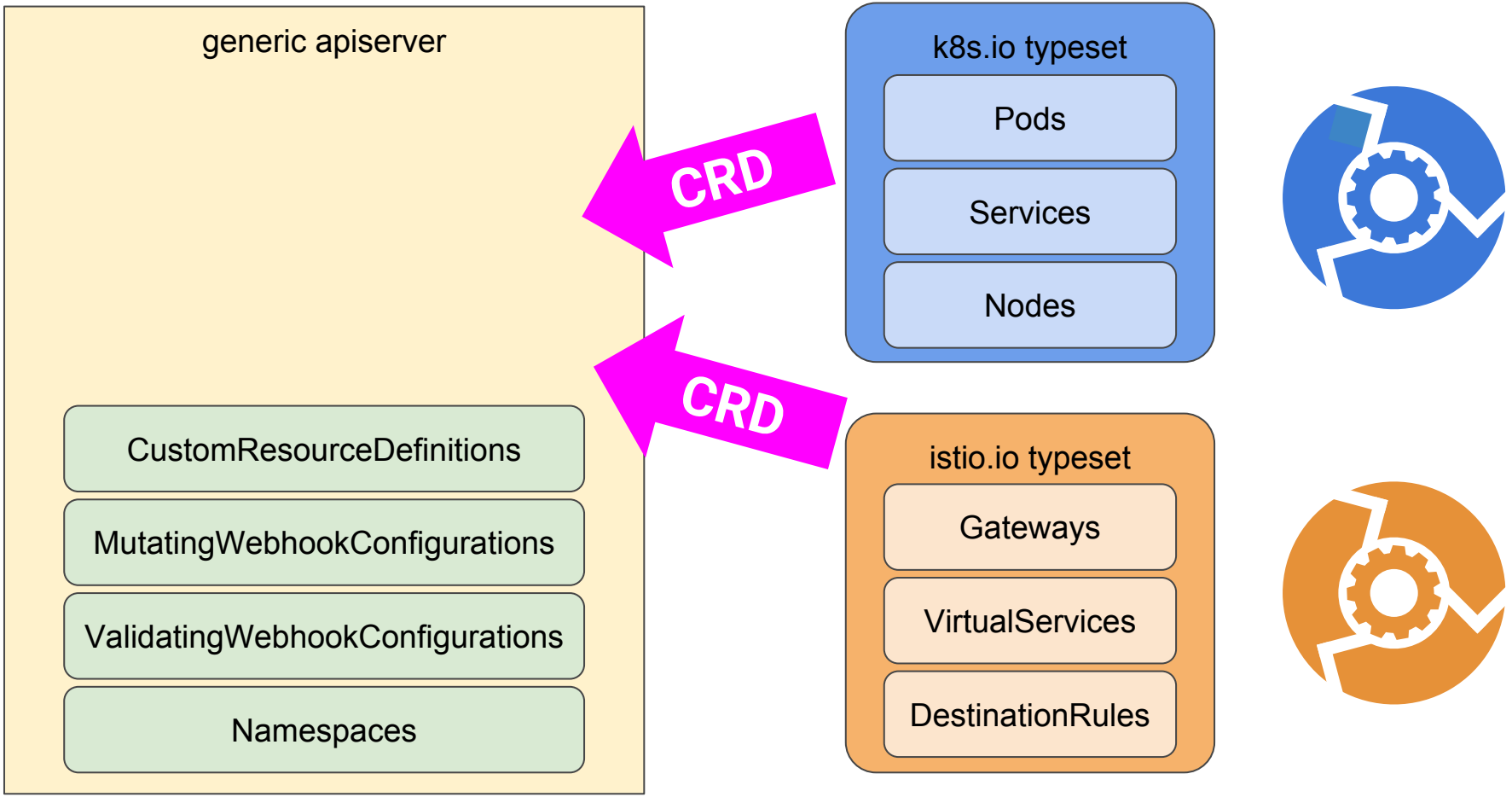
Eventually: Everything becomes a CRD (except things to run CRDs)

- Built-in: Namespaces, CRDs, Admission, etc.
- CRDs: Pods, Services, Nodes, Deployments, ...
- Kubernetes is a set of operators

CRD will need to get more powerful (a good thing).
We need to simplify some APIs (also a good thing).

There should be nothing that we can do that you can't




generic apiserver
CustomResourceDefinitions
MutatingWebhookConfigurations
ValidatingWebhookConfigurations
Namespaces
CRD
CRD
k8s.io typeset
Pods
Services
Nodes
istio.io typeset
Gateways
VirtualServices
DestinationRules

Work in progress

Kubernetes is getting more powerful here

- CRD versioning
- Sub-resources
- Better validation

The generic apiserver should be the only apiserver

Kubernetes API machinery is a project of its own

Serving “cloud-native” APIs should be trivial