To CRD, or Not to CRD, 
That Is the Question
Intros

Sam Gunaratne
Pivotal, London
@sam_gun

Ed King
Pivotal, London
@edking2
This Talk

- Motivations
- Where we’ve come from:
  - The k8s API, CRDs & Operators
- Where we **may** be heading:
  - Building on top of the k8s API
  - Pros, cons and considerations
- Closing remarks and Questions
Motivations
K8s is API-centric
## The k8s API

<table>
<thead>
<tr>
<th>Declarative</th>
<th>State separation</th>
<th>Level-based</th>
<th>Transparent</th>
</tr>
</thead>
</table>
| - What vs How
  - World can be built from a set of config
  - Easy to record changes | - Desired state
  - Observed state | - Does not rely on individual changes
  - Less prone to error | - One control plane
  - Facilities composability |
Example
k8s api
"I want 1 pod to be running"
"I want 1 pod to be running"
apiVersion: v1
kind: Pod
metadata:
  name: my-pod-1
  namespace: default
spec:
  containers:
    name: my-ctr-1
    image: reg.io/my-img:v1.0.0
k8s api

kubectl apply -f pod

desired
kubectl apply -f pod
kubectl apply -f pod
kubectl apply -f pod

current pods: 1
kubectl apply -f pod

current pods: 1
apiVersion: v1
kind: Pod
metadata:
  name: my-pod-1
  namespace: default
spec:
  containers:
    name: my-ctr-1
    image: reg.io/my-img:v1.0.0
status:
  state: running
  ready: true
Custom Resource Definitions
apiVersion: v1

type: Cake

metadata:
  name: my-delicious-cake-1
  namespace: default

spec:
  name: victoria-sponge
  ingredients:
    - sugar
    - eggs
    - flour
  bakeTime: 30m
kubectl apply -f cake
kubectl apply -f

desired

cake
kubectl apply -f pod cake

k8s api

- pod
- cake
- foo

desired
observed
kubectl apply -f

desired

observed
March 2016
Third party resources
March 2016
Third party resources

Nov 2016
Operator framework
Operator framework

Third party resources  CRD beta

March 2016  July 2017

Nov 2016
March 2016
Third party resources

Nov 2016
Operator framework

July 2017
CRD beta

July 2018
Knative
CRD GA?
Every resource as a CRD?
Exciting other things?

March 2016
Third party resources

Nov 2016
Operator framework

July 2017
CRD beta

July 2018
Knative

Oct 2018
AWS operator

Next
CRD GA?
Every resource as a CRD?
Exciting other things?
Let me paint you a picture
{  
    "action": "switch_on",
    "lights": [  
        "lamp-1",
        "lamp-2"
    ],
    "room": "kitchen"
}
The diagram illustrates a system involving service discovery with three main services: lights, locks, and rooms. The gateway processes JSON data, which is then used to discover and interact with these services through their respective APIs and databases.
Distractions ...

- Storage
- High Availability
- Reliability
- The API contracts of each service (e.g. company standards)
- Team collaboration over APIs

These all take time and effort away from the main focus of each team, which is to provide the best service they can.
gateway

json

gateway

service discovery

lights controller

locks controller

rooms controller
k8s api

- lights controller
- locks controller
- rooms controller
“I want the kitchen lights to be on”
kubectl apply -f

k8s api

desired

observed
apiVersion: v1
kind: Room
metadata:
  name: kitchen
  namespace: default
spec:
lights:
  - name: lamp-1
    brightness: 0.5
  - name: lamp-2
    brightness: 1.0
kubectl apply -f room

k8s api

desired
k8s api

room
light

light

desired

kubectl apply -f
apiVersion: v1
kind: Light
metadata:
  name: light-1
  namespace: default
spec:
  brightness: 0.5
kubectl apply -f

room
light

k8s api

light

‘lamp-1’ brightness: 0.0
‘lamp-2’ brightness: 0.0

desired
observed
kubectl apply -f

room
light
light

‘lamp-1’ brightness: 0.5
‘lamp-2’ brightness: 1.0
`kubectl apply -f room`
apiVersion: v1
kind: Light
metadata:
  name: light-1
  namespace: default
spec:
  brightness: 0.5
status:
  currentBrightness: 0.5
'lamp-1' brightness: 0.5
'lamp-2' brightness: 1.0
apiVersion: v1
kind: Room
metadata:
  name: kitchen
  namespace: default
spec:
  lights:
  - name: lamp-1
    brightness: 0.5
  - name: lamp-2
    brightness: 1.0
status:
  numLightsOn: 2
'lamp-1' brightness: 0.5
'lamp-2' brightness: 1.0
Pros, Cons & Considerations
Storage

Access to an etcd datastore for “free”!

- Developers do not need to worry about operational overheads
- ... But etcd is not a relational database
High Availability

API is highly available for “free”!

- When deployed in a multi-master configuration, the k8s API is Highly Available
- Developers do not need to worry about operational overheads!
Performance

Performance is largely dependent on etcd

- “Noisy” services could have a negative impact on performance
- 1 huge k8s cluster vs lots of little ones
  - Perhaps not so much of an issue in smaller clusters
- API machinery are thinking about scaling targets
  - Fill in the survey!
Programming Model

Declarative vs Imperative

- The k8s API is entirely declarative and eventually consistent
- Great for stability and reliability of the system!
- Writing reconciliation
- Not everything fits!
Team Collaboration

CRDs as the standard interface between teams

- Teams ship controllers and CRDs
- Team A’s controller could watch for changes to Team B’s Custom Resources
- Part of what it means to be “kubernetes-native”
Other API Features

What other API features should you consider?

- AuthN/Z
- Pagination
- Querying
- Binary Data

- Resource relationship
- Versioning
- Quotas
- Tooling (kubectl/UIs etc)
To CRD, or Not to CRD?
To the entire Kubernetes community:

Abby Fuller @abbyfuller · Jun 18
I don't know who needs to hear this but not everything needs to be a Kubernetes CRD
Thank You!