KubeCon | CloudNativeCon

Europe 2018

Service Catalog Deep Dive

Michael Kibbe mkibbe@google.com

Morgan Bauer mbauer@us.ibm.com @ibmhb

Agenda

- Where are we now?
- Where were we, how did we get here?
- Where are we going?

Current Status – Where are we?

- OSB API
- Types
- Build
- API Server
- Controller
- svcat CLI

OSB API

- Many in this group participate
- Some thing swe've driven
  - json schema
  - async binding
  - Get
  - cluster id
  - Generic actions

Resource Model

- Follows along the main types of OSBAPI
- Turns the five RPC resources into standard kubernetes objects
- Spec & Status

Build Flow

- Travis + jenkins
- Single Hyperkube-like binary output
- Charts into https://svc-catalog-charts.storage.googleapis.com/
- Images into quay.io for multiple arches

API Server

- Upstream API Machinery reuse
- Code Generator reuse
  - Client, listers, informers, some extensions
  - Types: conversion, defaulting

Controller Design

- Multiple Controllers in a manager
- One for each major API type
- Uses generated code for clients and watchers

svcat cli

- Donated from microsoft
- Standalone binary cli
  - svcat get brokers
- Can be used as a kubectl plugin
  - kubectl svcat get brokers
  - Pretty output

How did we get here?

- Timeline
- Issues
- Challenges

Timeline

- August 2016 idea as kube implementation of OSBAPI
- November 2016 - (1.4) first F2F in Boulder, first etcd backed apiserver
- December 2016 - (1.5) Initial Code Drop
- January 2017 – second F2F
- February 2017 – Add TPR based storage, first client-go release
- March 2017 – (1.6) apimachinery release, alpha API Aggregation
- April 2017 – Svc-cat use of API Aggregation
- June 2017 (1.7) - CRD enter beta, API Aggregation Beta
- October 2017 (1.8) – remove k/k dependency, drop TPR
- December 2017 (1.9) - (?)
- April 2018 (1.10) - API Aggregation Stable

Issues

- Upstream apimacachinery vendor & rebase
  - Over multiple major versions of kube
  - Rebase hell
- Mismatch between OSB and Kube resource behavior
  - Guids vs names
  - Imperative vs declarative
  - Broker source of truth vs kube source of truth
- Controller issues
  - Again reusing a lot of upstream code which we did not know how to properly use, or it had changed from underneath us by the time we thought we understood it.

Challenges

- Bleeding edge feature usage
  - API Aggregation
    - Alpha through to stable. Push for docs.
    - Can't override core resources that don't exist, naming conflicts
  - API Machinery
    - Inaccessibility of etcd
  - Code Generators
  - RBAC rules
  - All of the above interact with Helm Charts for installation
  - TPR backend
  - Pod preset moving out of core to us

Where are we going?

- CRDs or Blob-store resource
- Mutating webhooks for pod-presets
- GA planning being done
- Cluster and namespace scoped versions of all resources

tl;dr

- Use CRDs
  - If you must make an apiserver, use apiserver-builder
- Use the example-controller and avoid modifying anything but the sync loop
- Spec and state is decent, but could be better if they were separate objects with separate lifecycles
- Keep your state machine documented
  - Keep your code modular

Questions?

- Resources
  - Service Catalog Meeting Agenda
  - https://github.com/kubernetes-incubator/service-catalog
  - Sevice Catalog Meeting Youtube Playlist