Building Fault Tolerant Custom Resource Controllers

Morgan Bauer mbauer@us.ibm.com
Wei Huang wei.huang1@ibm.com
Srinivas Brahmaroutu srbrahma@us.ibm.com
Agenda

- Intro to Kubernetes
- What are CRD/Controllers
- Leader Election Process
- Why Fault Tolerance
- Controller vs Controllee Patterns
- Custom Resource: name-spaced vs. clustered
- Testability and Best Practices
Introduction to Kubernetes

• At its core, Kubernetes is a database (etcd). With "watchers" & "controllers" that react to changes in the DB. The controllers are what make it Kubernetes. This pluggability and extensibility is part of its "secret sauce".

• DB represents the user's desired state
  • Watchers attempt to make reality match the desired state

"API Server" is the HTTP/REST front-end to the DB

More on controllers later...
Custom Resources and Controllers

• CRDs
  • Extension to Kubernetes, user defined object
  • A custom controller implements the declarative model
    • Ex: Operator pattern

• Real experiences from us, custom extension mechanism
  • Modified kube src in 1.4, 1.5, rebasing our stuff
  • Move in tree to out tree
    • 1.5: tpr, tpr is way broken
    • 1.7: transfer to CRDs
    • 1.11: validation of CRD schema
Leader Election – Why?

• What are the current failure modes?

• Can I just run a controller Deployment specifying replicas==1, and rely on Kubernetes native failover strategy?

• Then how about specifying replicas>1?
Leader Election – How?

• Leader election
  • history
  • provided by etcd

• Reuse k8s pkg “client-go/tools/leaderelection”
  • PR based on official sample-controller

• Best Practice
  • LeaseDuration vs. RenewDeadline vs. RetryPeriod

• Deploy it as Deployment specifying replicas>1

• Ensure PodAntiAffinity
Demo

- Show installed controller/custom resources
- Show leader election logs
- Show time to restart and elect a controller
Controller vs Controllee

• Controller pattern: CR object as a controller
• Use `kubectl create [cr]` or API
  • For example, in sample-controller, customer creates a CR object, then the CRD controller creates a nginx Deployment
  • Other examples, etcd and Prometheus operator developed by CoreOS
Controller vs Controllee (cont.)

• Controllee pattern: CR object as a controllee - customer still creates standard kube objects, and CR object is transparent to them
  • For example, user creates a Service with type: LoadBalancer, or creates any kube object contains some annotations. The CRD controller watches the changes, and create specific CR object.
CRD Name-spaced vs Clustered

• If the CR object is a controller, then go with “namespaced”

• If the CR object is a controllee, then go with “clustered”
Demo

• CRD with Custom Controller

• Controllee pattern

• RBAC Security
Testing and Best Practices

• Make your client code testable - interface over struct

• Validation on CR object metadata

• For “controller” pattern, migration considerations

• Wait to be elected leader before starting watches
Questions

• Resources
  • Custom Resource Definition
  • Etcd operator
  • Etcd API Documentation
  • Leader Election

• Wei Huang – wei.huang1@ibm.com @hweicdl
• Morgan Bauer – mbauer@us.ibm.com @ibmhb
• Srinivas Brahmaroutu – srbrahma@us.ibm.com @brahmaroutu