KubeCon | CloudNativeCon

China 2018

Rage Against the APIMachinery: Production-Ready Operators

你好

- Kanister: Open-source Operator
  - Framework for application-level data management
  - Example Apps: MySQL, MongoDB, PostgreSQL, ElasticSearch
  - Generic Infra Support: Volumes, ObjectStore
- K10’s API
  - 2 CRD-controllers
  - 1 Aggregated API server

Motivation

- Operators are easy to bootstrap
- They’ll solve your app-management problems
- ... but are they ready for production?

WDDD

Does your Operator feel like a native Kubernetes API?

Operators

- Domain Specific: Manage your App’s lifecycle
- Use familiar Kubernetes tools
- CustomResourceDefinitions + Controller

Production Operators

- Follow API conventions
- Support native clients: kubectl + SDKs
- Correctly configure RBAC
- Create Kubernetes Events
- Support testing
- Handle transitions changes safely

API conventions

https://github.com/kubernetes/community/blob/master/contributors/devel/api-conventions.md

Follow best practices for:

- ObjectMeta
- Naming
- Spec vs. Status
- Declarative vs. Imperative
- Conditionals
- Optional vs. Required

Bootstrapping an Operator

Operator kits: Ancient history (last year)

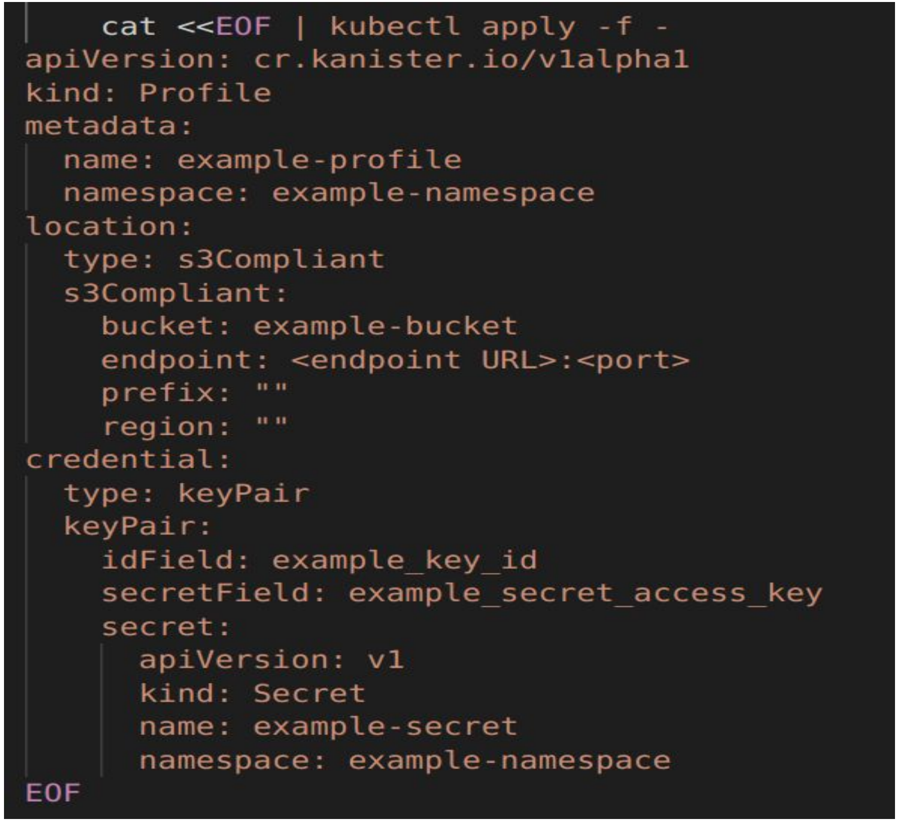
- Rook
- Giant Swarm

The Modern Era

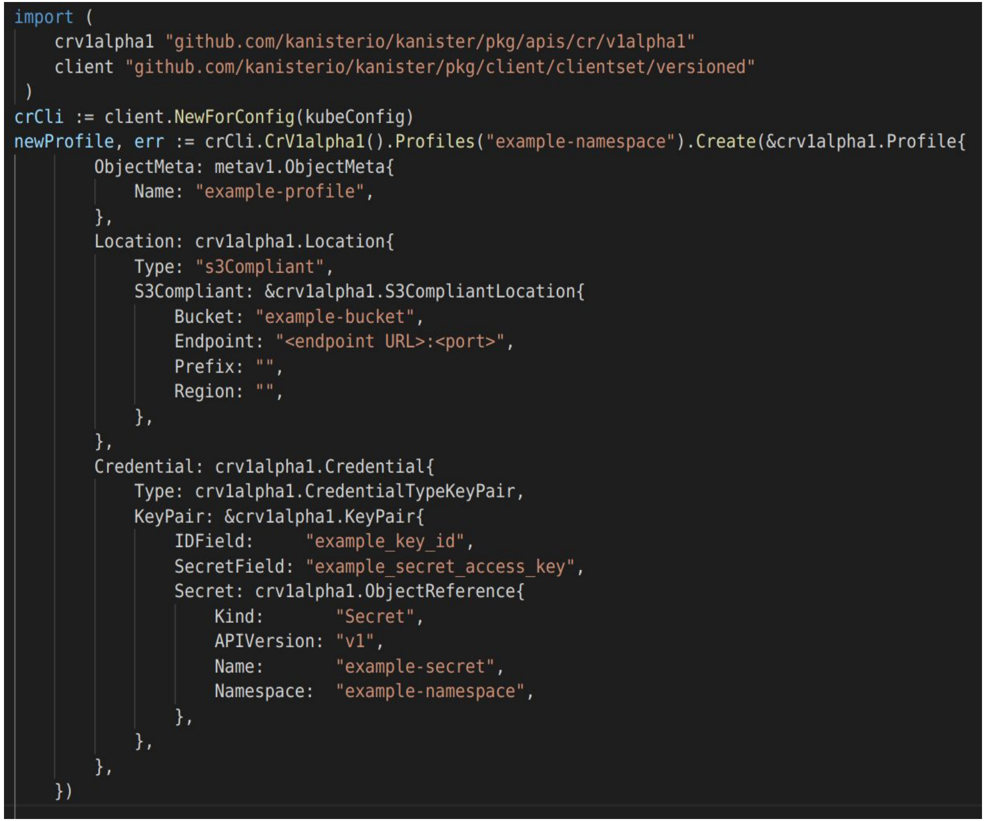
- Operator SDK
- Kubebuilder
- Metacontroller

Clients: kubectl + SDKs

KubeCon CloudNativeCon China 2018



```
    cat <<EOF | kubectl apply -f -
apiVersion: cr.kanister.io/v1alpha1
kind: Profile
metadata:
  name: example-profile
  namespace: example-namespace
location:
  type: s3Compliant
  s3Compliant:
    bucket: example-bucket
    endpoint: <endpoint URL>:<port>
    prefix: ""
    region: ""
credential:
  type: keyPair
  keyPair:
    idField: example_key_id
    secretField: example_secret_access_key
    secret:
      apiVersion: v1
      kind: Secret
      name: example-secret
      namespace: example-namespace
EOF
```



```
import (
    crv1alpha1 "github.com/kanisterio/kanister/pkg/apis/cr/v1alpha1"
    client "github.com/kanisterio/kanister/pkg/client/clientset/versioned"
 )
crCli := client.NewForConfig(kubeConfig)
newProfile, err := crCli.CrV1alpha1().Profiles("example-namespace").Create(&crv1alpha1.Profile{
        ObjectMeta: metav1.ObjectMeta{
            Name: "example-profile",
        },
        Location: crv1alpha1.Location{
            Type: "s3Compliant",
            S3Compliant: &crv1alpha1.S3CompliantLocation{
                Bucket: "example-bucket",
                Endpoint: "<endpoint URL>:<port>",
                Prefix: "",
                Region: "",
            },
        },
        Credential: crv1alpha1.Credential{
            Type: crv1alpha1.CredentialTypeKeyPair,
            KeyPair: &crv1alpha1.KeyPair{
                IDField:     "example_key_id",
                SecretField: "example_secret_access_key",
                Secret: crv1alpha1.ObjectReference{
                    Kind:       "Secret",
                    APIVersion: "v1",
                    Name:       "example-secret",
                    Namespace:  "example-namespace",
                },
            },
        },
    })
```

Code Generation

https://github.com/kubernetes/code-generator

- deepcopy-gen
- client-gen
- Informer-gen
- lister-gen

```
// +genclient

// +genclient:noStatus

// +k8s:deepcopy-gen:interfaces=k8s.io/apimachinery/pkg/runtime.Object
```

RBAC

- RBAC is a double-edged sword
- If you see the object, it doesn’t guarantee the Operator can
- Follow PoLP (principle of least authority) for the controller ServiceAccount

Eventing

```
// Initialize Event Recorder
broadcaster := record.NewBroadcaster()
broadcaster.StartEventWatcher(
    func(event *core.Event) {
        _, err := client.Core().Events(event.Namespace).Create(event)
    },
)
source := core.EventSource{Component: "Widget Controller"}
recorder := broadcaster.NewRecorder(scheme.Scheme, source)

// Record Event
recorder.Event(obj, corev1.EventTypeNormal, "Started", "Started work on Widget!")
```

Graceful Changes

- Know your app
- Better be safe than sorry
- Scale down is not the same as scale up

Testing: REST Configs

In-Cluster

```
cfg, err := rest.InClusterConfig()
```

Out-of-Cluster

```
cfg, err := clientcmd.NewNonInteractiveDeferredLoadingClientConfig(
    clientcmd.NewDefaultClientConfigLoadingRules(),
    &clientcmd.ConfigOverrides{},
).ClientConfig()
```

Testing: Fake Clients

```
// Always return a new Widget with the request name.
reaction := func(action testing.Action) (bool, runtime.Object, error) {
    get, _ := action.(testing.GetAction)
    ret := &v1.Widget{
        ObjectMeta: metav1.ObjectMeta{
            Name: get.GetName(),
        },
    }
    return true, w, nil
}

// Create fake Clientset
cli := fake.NewSimpleClientset()
cli.PrependReactor("get", "widgets", reaction)
```

RAAAAAAAGEEEEEEE (Paper Cuts)

- CRD Lifecycle
- Object Versioning
- Code Generation
- Validation
    - Open api schema
    - Admission Controllers

Kubernetes has the features to support robust Operators. You just need to use them.

KubeCon | CloudNativeCon

China 2018

We are Hiring!