KubeBuilder: SDK for extending Kubernetes

Fan Zhang
GitHub: @fanzhangio
fanz@vmware.com

Mengqi Yu
GitHub: @mengqiy
mengqiy@google.com
Agenda

● Kubernetes Resources, CRD
● How to extend Kubernetes API by CRD
● Kubebuilder - Framework and SDK
● Demo
Kubernetes Resources

Core API
- Workloads
- Discovery & LoadBalancer
- Config & Storage
- Cluster

Properties
- Declarative
- Level-based
- Asynchronous

Kubernetes Object
- TypeMeta
  - Kind
  - APIVersion
- ObjectMeta
  - Standard metadata fields
- Spec
  - Desired state
- Status
  - Observed state
CustomResourceDefinitions (CRD)

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: boundeddeployments.example.cncf.io
spec:
  group: example.cncf.io
  names:
    kind: BoundedDeployment
    plural: boundeddeployments
  scope: Namespaced
  validation:
    openAPIV3Schema:
      properties:
        apiVersion:
          type: string
        kind:
          type: string
        metadata:
          type: object
        spec:
          properties:
            replicas:
              format: int32
              type: integer
            status:
              type: object
        version: v1alpha1
```

# Install CRD into cluster
```sh
kubectl apply -f crd.yaml
```

# Create an CR instance
```sh
kubectl apply -f instance.yaml
```

```yaml
apiVersion: example.cncf.io/v1alpha1
kind: BoundedDeployment
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: boundeddeployment-sample
spec:
  replicas: 1
```
CustomResource Lifecycle

1) Create or update CR object

2) Admission Review Request

3) Admission Review Response with patches

4) Admission Review Request

5) Admission Review Response with decision (admit or reject)

6) Persist if admitted

7) Watch Reconcile

Conversion webhook

Coming soon
# How to extend Kubernetes by CRD

## From-scratch Approach

<table>
<thead>
<tr>
<th>CRD</th>
<th>Controller</th>
<th>Admission Webhook</th>
<th>Extra work</th>
</tr>
</thead>
</table>
| ● Implement API  
  ○ API type model  
  ○ Scheme register  
  ● Code generation  
  ○ deepcopy-gen | ● Code generation  
  ○ Versioned clientset  
  ○ Informers  
  ○ ListWatch  
  ● Implement Controller  
  ○ EventHandler  
  ○ Workqueue  
  ○ Worker  
  ○ Reconciler | ● Manage TLS certificates  
  ● Implement webhook server  
  ○ Handler | ● Scripts for code generation  
  ● Write manifests  
  ● Run controller  
  ● Run webhook server  
  ● Build and publish images  
  ● Deploy |
From-Scratch Approach

k8s.io/code-generator
k8s.io/client-go

diagram is from
k8s.io/sample-controller
<table>
<thead>
<tr>
<th>From-scratch</th>
<th>Kubebuilder</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main entry point, Signal handler</td>
<td>Scaffolding</td>
</tr>
<tr>
<td>API type</td>
<td></td>
</tr>
<tr>
<td>Scheme register</td>
<td>Controller-tools</td>
</tr>
<tr>
<td>Code Generation: deepcopy-gen</td>
<td></td>
</tr>
<tr>
<td>Versioned clientset: client-gen</td>
<td></td>
</tr>
<tr>
<td>Informer:informer-gen</td>
<td></td>
</tr>
<tr>
<td>Lister:lister-gen</td>
<td></td>
</tr>
<tr>
<td>Kubernetes rest client</td>
<td></td>
</tr>
<tr>
<td>EventHandler/HandlerFuncs</td>
<td></td>
</tr>
<tr>
<td>Controller run and worker funcs</td>
<td></td>
</tr>
<tr>
<td>Workqueue processing</td>
<td></td>
</tr>
<tr>
<td>Reconciliation</td>
<td></td>
</tr>
</tbody>
</table>
Kubebuilder Sequence

User → Cluster

Config
(1) Kubebuilder init (--domain)
dep ensure (Y/N)

Run kubectl from CLI

Scaffolding Project
domain,vendor(optional)

(2) Create API type by Group, Version, Kind
(Y/N)  (TODO) User implements API

(2) Create Controller
(Y/N)  (TODO) User implements reconcile

(2) Create Webhook
(Y/N)  (TODO) User implements Handler

Dependencies:
1. Controller-tool
2. Controller-runtime

Build and Test CRD and controller by make tool
Deploy CRD
Build and publish image
Deploy Controller and Webhook by Manager

User → Cluster
Controller-tools

- Parse API resource user implemented
- Generate codes - Deepcopy()
- Generate manifests and documents
  - CRD
  - RBAC
  - Webhook manifests
- Provide a binary tool: Controller-gen
Controller-runtime

- SDK for building controllers and webhooks
- Abstracted building blocks
  - Provide shared dependencies
  - Encapsulate common controller pattern
  - Provide common foundation for libraries and frameworks building higher-level abstractions

*Why Are We Copying and Pasting So Much? – Solly Ross, Philip Wittrock*
https://github.com/mengqiy/kubebuilder-demo
Kubebuilder Recap

- Easy to use
- Minimum operations
- Streamline workflow
Resources

- Kubectl repo: [https://github.com/kubernetes-sigs/kubebuilder/](https://github.com/kubernetes-sigs/kubebuilder/)
- Slack channel: #kubebuilder
- Workshop from @pwittrock
  - [https://github.com/pwittrock/kubebuilder-workshop](https://github.com/pwittrock/kubebuilder-workshop)
- Welcome to contribute!!!

Thank you!
Q & A
Supplemental Slides
Kubebuilder

Framework and SDK for developing CRD and Controller Effectively

Initialize Project
- Project structure
- Dependencies
- Main entry point
- Makefile
- Dockerfile
- Kustomize integration

Scaffold APIs & Controller & Webhook
- Boilerplates
- Controller-tool
- Controller-runtime
- Annotations

Generating Code & Manifests
- Code generation
- Manifests generation
  - CRD
  - RBAC
  - Namespace
  - StatefulSet
  - AdmissionWebhook
- Make Tool
  - Build
  - Test
  - Install
From-scratch vs KubeBuilder

### From-scratch

- **Main entry point, Signal handler**
- **API type**
- **Scheme register**
- **Code Generation: deepcopy-gen**
- **Versioned clientset: client-gen**
- **Informer:informer-gen**
- **Lister:lister-gen**
- **Kubernetes rest client**
- **EventHandler/HandlerFuncs**
- **Controller run and worker funcs**
- **Workqueue processing**
- **Reconciliation**