

Introduction to Kubernetes Extensions

Kubernetes

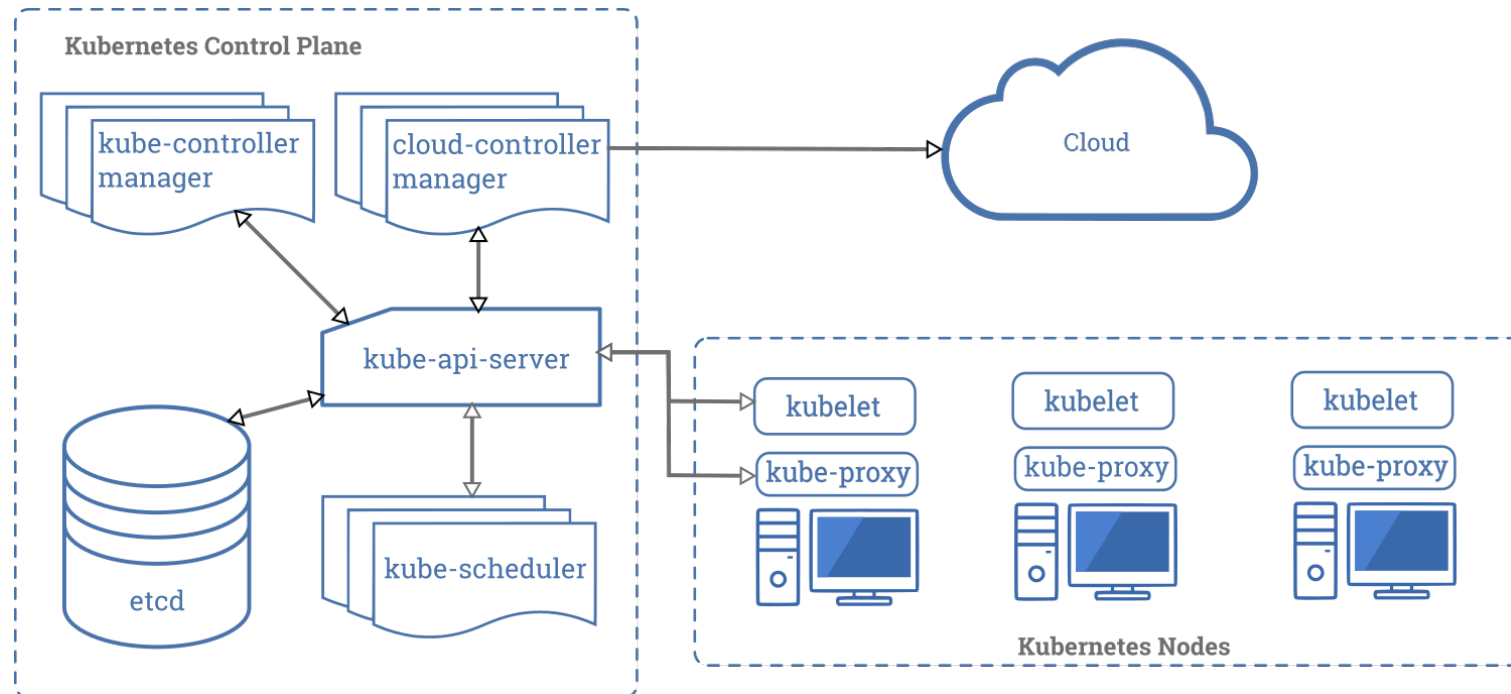
A *Custom Resource (CR)* is an extension of the Kubernetes API. To understand Custom Resource Definitions (CRD) and Operators, you have to understand the Kubernetes API, the resource controller watch loop that Kubernetes uses to control the system, and how it is replicated by an operator.

Kubernetes provides two ways to add custom resources to your cluster:

- CRDs are simple and can be created without programming an Operator,
- API Aggregation (AA) requires programming an Operator, but allows more control over API behaviors like how data is stored and conversion between API versions.

Kubernetes Architecture

- At its core, Kubernetes is a database (etcd) with "watchers" and "controllers" that react to changes in etcd,
- The kube-api-server is an API server that exposes the Kubernetes API,
- The kube-scheduler watches for new Pods not assigned to a node and selects a node to run the Pod on,
- The kube-controller-manager runs the controller loops that make Kubernetes,
- The cloud-controller-manager interacts with the underlying cloud provider,
- The etcd key-value store represents the user defined desired state of the objects,
- The kubelet makes sure that containers are running in a Pod,
- The kube-proxy is a network proxy that maintains network rules on nodes,



API Groups

All Kubernetes objects are considered an API resource and have a corresponding endpoint in the Kubernetes API. The [Kubernetes API](#) is divided into [API Groups](#) to make it easier to extend the API, disabling APIs, supporting different versions, support API Plugin.

API groups:

- core,
- apps,
- extensions,
- batch,
- autoscaling,
- storage.k8s.io,
- admissionregistration.k8s.io,
- **apiextensions.k8s.io,**
- policy,
- scheduling.k8s.io,
- settings.k8s.io,
- apiregistration.k8s.io,
- certificates.k8s.io,
- rbac.authorization.k8s.io,
- authorization.k8s.io,
- networking.k8s.io,
- auditregistration.k8s.io

apiextensions.k8s.io

- **MetaData APIs**
 - **CustomResourceDefinition**

Extending Kubernetes

Customization can be divided into:

- configuration,
- Extensions.

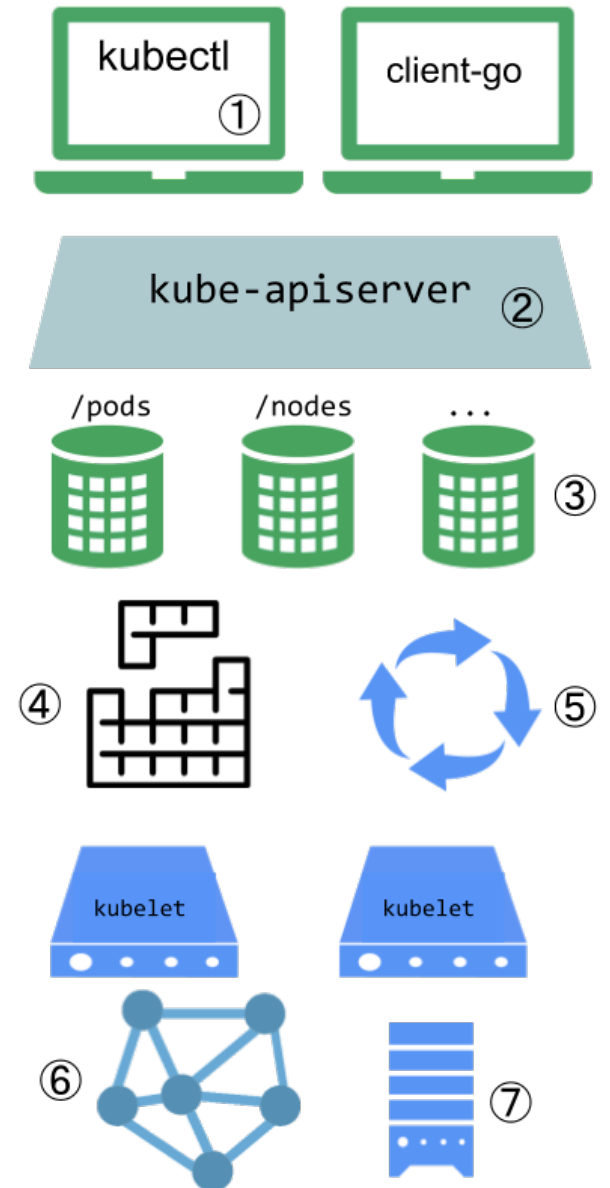
You can extend the Kubernetes by using the Controller pattern or the webhook model.

Controllers read a “spec”, do work, and then update the “status”.

When Kubernetes is the client that calls out to a remote service, it is called a Webhook and the remote service is the Webhook Backend.

Extension points:

1. Kubectl, kubectl plugins,
2. Extensions in the apiserver allow authenticating or blocking requests, editing content and handling deletion,
3. Custom Resources (CR) using CustomResourceDefinition API and Operators,
4. Scheduler Extensions,
5. Controllers are often used with CR,
6. Node-level Network Plugins, CNI Plugins or Kubenet plugins,
7. Storage Plugins,



Custom Resource Definition (CRD)

In June 2017, Kubernetes v1.7 introduced Custom Resource Definitions (CRD).

A *Custom Resource (CR)* is an extension of the Kubernetes API that allows you to store an API object of a kind. Even many core Kubernetes functions are now built using custom resources.

A *Custom Resource Definition (CRD)* file defines your own object kinds and lets the API Server handle its lifecycle.

Example: CRD


guestbook.yaml


```
apiVersion: apiextensions.k8s.io/v1
kind: CustomResourceDefinition
metadata:
  name: guestbooks.apps.ibm.com
spec:
  group: apps.ibm.com
  versions:
    - name: v1
      served: true
      storage: true
      schema:
        openAPIV3Schema:
          type: object
          properties:
            spec:
              type: object
              properties:
                guestbookTitle:
                  type: string
                guestbookSubtitle:
                  type: string
  scope: Namespaced
  names:
    plural: guestbooks
    singular: guestbook
    kind: Guestbook
    shortNames:
      - gb
```

my-guestbook.yaml

```
apiVersion: "apps.ibm.com/v1"
kind: Guestbook
metadata:
  name: my-guestbook
spec:
  guestbookTitle: "The Chemical Wedding of Remko"
  guestbookSubtitle: "First Day"
```

```
% kubectl create -f guestbook-crd.yaml
customresourcedefinition.apiextensions.k8s.io/guestbooks.apps.ibm.com created
% kubectl create -f my-guestbook.yaml
guestbook.apps.ibm.com/my-guestbook created
% kubectl get guestbooks
NAME          AGE
my-guestbook  3m51s
% kubectl describe guestbook my-guestbook
Name:         my-guestbook
Namespace:    default
Labels:       <none>
Annotations:  <none>
API Version:  apps.ibm.com/v1
Kind:         Guestbook
Metadata:
  Creation Timestamp: 2020-04-20T05:40:46Z
  Generation:       1
  Resource Version:  295507
  Self Link:         /apis/apps.ibm.com/v1/namespaces/default/guestbooks/my-guestbook
  UID:               e683caef-33f5-4f83-b54e-85ee163e0abf
Spec:
  Guestbook Subtitle: First Day
  Guestbook Title:   The Chemical Wedding of Remko
Events:              <none>
```

 **kubernetes**

 Search

my-guestbook0/0^vX

Custom Resource Definitions > guestbooks.apps.ibm.com

Storage Classes

Namespace

default

Overview

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Discovery and Load Balancing

Ingresses


Services

Config and Storage

Config Maps

Persistent Volume Claims

Secrets

 Custom Resource Definitions

Metadata

Name	Creation time	Age	UID
guestbooks.apps.ibm.com	Apr 20, 2020	17 minutes	2dc92a69-1e5e-489f-a9fa-c5e7af780aab

Resource Information

Version	Scope	Group
v1	Namespaced	apps.ibm.com

Accepted Names

Plural	Singular	Kind	List Kind	Short Names
guestbooks	guestbook	Guestbook	GuestbookList	gb

Objects

Name	Namespace	Age
my-guestbook	default	10 minutes

1 – 1 of 1

Versions

Name	Served	Storage
v1	True	True

Operator Pattern

Custom Resources alone let you store and retrieve structured data. The [Operator pattern](#) combines custom resources and custom controllers. Combined with a *custom controller*, custom resources provide a true *Declarative API*. You can use custom controllers to encode domain knowledge for specific applications into an extension of the Kubernetes API.

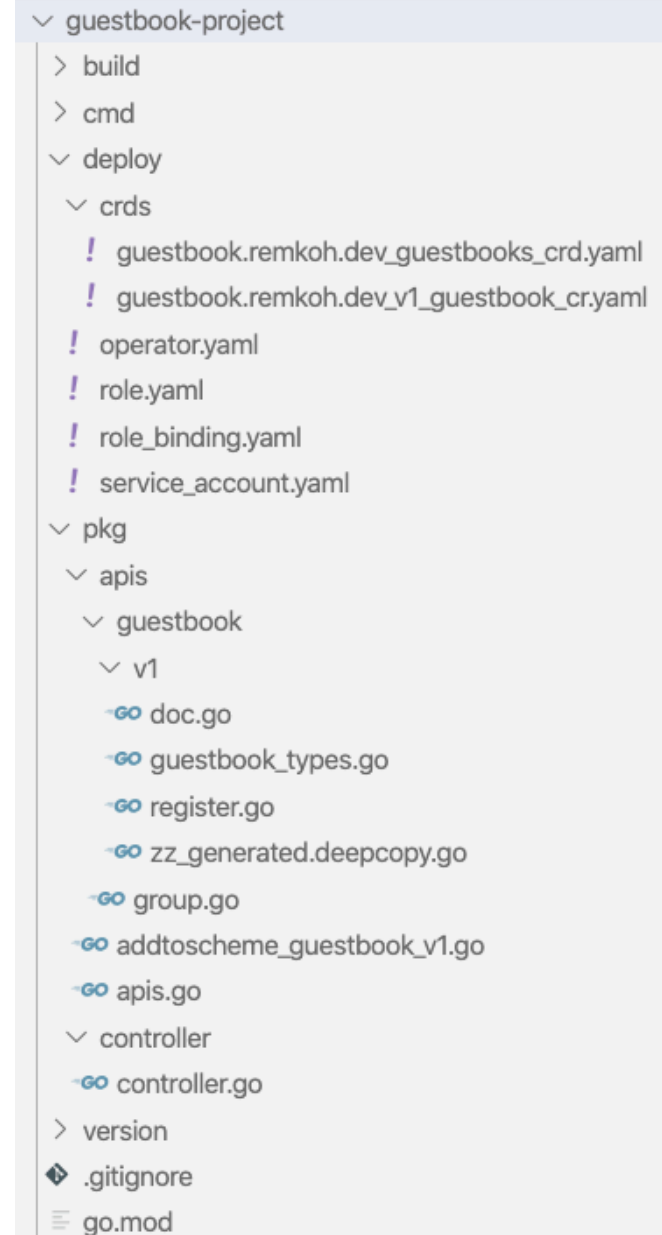
Consider API aggregation if:	Prefer a stand-alone API if:
Your API is <u>Declarative</u> .	Your API does not fit the <u>Declarative</u> model.
Read and write your new types using kubectl.	kubectl support is not required
View your new types in a Kubernetes UI, such as dashboard, alongside built-in types.	Kubernetes UI support is not required.
You are developing a new API.	You already have a program that serves your API and works well.
Accept the format restriction that Kubernetes puts on REST resource paths, such as API Groups and Namespaces. (See the API Overview .)	You need to have specific REST paths to be compatible with an already defined REST API.
Your resources are naturally scoped to a cluster or namespaces of a cluster.	Cluster or namespace scoped resources are a poor fit; you need control over the specifics of resource paths.
You want to reuse <u>Kubernetes API support features</u> .	You don't need those features.

Operator Framework SDK

```
operator-sdk new $OPERATOR_PROJECT --type go --repo github.com/$DOCKER_USERNAME/$OPERATOR_NAME
operator-sdk add api --api-version=$OPERATOR_GROUP/$OPERATOR_VERSION --kind=$CRD_KIND
operator-sdk add controller --api-version=$OPERATOR_GROUP/$OPERATOR_VERSION --kind=$CRD_KIND
operator-sdk build docker.io/$DOCKER_USERNAME/$OPERATOR_NAME
docker login docker.io -u $DOCKER_USERNAME
docker push docker.io/$DOCKER_USERNAME/$OPERATOR_NAME
sed -i "s|REPLACE_IMAGE|docker.io/$DOCKER_USERNAME/$OPERATOR_NAME|g" deploy/operator.yaml
oc create sa $OPERATOR_PROJECT
oc create -f deploy/role.yaml
oc create -f deploy/role_binding.yaml
oc create -f deploy/crds/${OPERATOR_GROUP}_${CRD_KIND,,}_s_crd.yaml
oc create -f deploy/operator.yaml
oc create -f deploy/crds/${OPERATOR_GROUP}_${OPERATOR_VERSION}_${CRD_KIND,,}_cr.yaml
oc get deployment $OPERATOR_PROJECT
oc get pod -l app=example-${CRD_KIND,,}
oc describe ${CRD_KIND,,}s.${OPERATOR_GROUP} example-${CRD_KIND,,}
```

Operator Framework SDK

```
// Reconcile reads state of the cluster for a Guestbook object and makes changes based on the state
// read and what is in the Guestbook.Spec
// TODO(user): User must modify this Reconcile function to implement their own Controller logic.
// This example creates a Pod as an example
func (r *ReconcileGuestbook) Reconcile(request reconcile.Request) (reconcile.Result, error) {
    ...
    // Fetch the Guestbook instance
    instance := &guestbookv1.Guestbook{}
    ...
    // Define a new Pod object
    pod := newPodForCR(instance)
    ...
}
```



The image shows a file explorer view of a directory named 'guestbook-project'. The directory structure is as follows:

- guestbook-project
 - build
 - cmd
 - deploy
 - crds
 - ! guestbook.remkoh.dev_guestbooks_crd.yaml
 - ! guestbook.remkoh.dev_v1_guestbook_cr.yaml
 - ! operator.yaml
 - ! role.yaml
 - ! role_binding.yaml
 - ! service_account.yaml
 - pkg
 - apis
 - guestbook
 - v1
 - doc.go
 - guestbook_types.go
 - register.go
 - zz_generated.deepcopy.go
 - group.go
 - addtoscheme_guestbook_v1.go
 - apis.go
 - controller
 - controller.go
 - version
 - .gitignore
 - go.mod

Kubernetes Applications

<https://github.com/kubernetes-sigs/application>

```
apiVersion: app.k8s.io/v1beta1
kind: Application
metadata:
  name: "guestbook"
  labels:
    app.kubernetes.io/name: "guestbook"
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: "guestbook"
  componentKinds:
    - group: v1
      kind: Deployment
    - group: v1
      kind: Service
  descriptor:
    type: "guestbook"
  keywords:
    - "gb"
    - "guestbook"
  links:
    - description: Github
      url: "https://github.com/IBM/guestbook"
    version: "0.1.0"
  description: "The Guestbook application is an example app to demonstrate key Kubernetes functionality."
  maintainers:
    - name: IBM Developer
      email: developer@ibm.com
  owners:
    - name: IBM Developer
      email: developer@ibm.com
```

Lab Time

<https://github.com/remkohdev/kubernetes-extensions>