



Kubernetes Operator-Framework Workshop

Presented by:

Matt Dorn - *Principal Software Engineer*



HOW MANY PEOPLE HERE USE KUBERNETES/OPENSHIFT REGULARLY?

DEVELOP IN GOLANG REGULARLY?

USE ANSIBLE REGULARLY?

ANYONE ATTEMPTED TO BUILD A CUSTOM KUBERNETES CONTROLLER
FROM SCRATCH?

ANYONE TRIED THE OPERATOR-SDK, ANSIBLE-OPERATOR,
OR HELM-APP OPERATOR YET?

OUR GOAL IS TO HELP YOU
SUCCEED WITH OUR TOOLS.

IN THIS ENTRY-LEVEL SESSION WE WILL
BE EXPLORING...



**OPERATOR
FRAMEWORK**

Support Channels

Google Groups

<https://groups.google.com/forum/#!forum/operator-framework>

Kubernetes Slack (slack.kubernetes.io)

#kubernetes-operators

OpenShift Commons - Operator Framework

<https://commons.openshift.org/sig/OpenshiftOperators.html>

Every third Friday of the month at 12pm EST

Logistics

9:00am to 3:00pm

Lunch: 12:00pm to 1:00pm

So...what is an Operator?



Operators

An Operator represents human operational knowledge in software, to reliably manage an application.

\$ oc get operators

the server doesn't have a
resource type "operators"

LET'S GO BACK A FEW YEARS.

[← Back to All Blogs](#)

Introducing Operators: Putting Operational Knowledge into Software

November 03, 2016 • By Brandon Philips

Tags: [announcements](#) [Operators](#)

A Site Reliability Engineer (SRE) is a person that operates an application by writing software. They are an engineer, a developer, who knows how to develop software specifically for a particular application domain. The resulting piece of software has an application's operational domain knowledge programmed into it.

Our team has been busy in the Kubernetes community designing and implementing this concept to reliably create, configure, and manage complex application instances atop Kubernetes.

We call this new class of software Operators. An Operator is an application-specific controller that extends the Kubernetes API to create, configure, and manage instances of complex stateful applications on behalf of a Kubernetes user. It builds upon the basic Kubernetes resource and controller concepts but includes domain or application-specific knowledge to automate common tasks.

3

It builds upon the basic Kubernetes resource and controller concepts but includes domain or application-specific knowledge to automate common tasks.

1

2

1

Resource

2

Controller

3

Knowledge

1

Resources

Pod

ConfigMap

Route

2

Controllers

ReplicaSet

Deployment

DaemonSet

Domain or Application Specific Knowledge?

Installing.

Self-Heal.

Scale (properly).

Clean Up.

Update.

Backup.

Restore.

etc.

An “Operator” takes
advantage of what
Kubernetes does best...


```
$ oc proxy
```

```
$ curl localhost:8001
```

```
$ curl http://localhost:8001/api/v1/ | jq .resources[].name
"bindings"
"componentstatuses"
"configmaps"
"endpoints"
"events"
"limitranges"
"namespaces"
"namespaces/finalize"
"namespaces/status"
"nodes"
"nodes/proxy"
"nodes/status"
"persistentvolumeclaims"
"persistentvolumeclaims/status"
"persistentvolumes"
"persistentvolumes/status"
"Pods"
```

...

```
$ kubectl get pod kube-dns-1187388186-rr1jb -n kube-system -o yaml
```

```
(curl -XGET ../api/v1/namespaces/kube-system/pods/kube-dns-1187388186-rr1jb)
```

```
apiVersion: v1
```

```
kind: Pod
```

```
metadata:
```

```
  name: kube-dns-1187388186-rr1jb
```

```
  namespace: kube-system
```

```
  ownerReferences:...
```

```
Spec:
```

```
  Containers:
```

```
    name: kubedns
```

```
    image: gcr.io/google_containers/k8s-dns-kube-dns-amd64:1.14.4
```

Operators take advantage of
Custom Resource Definitions.

...Formerly known as Third Party Resources (TPRs)

CRDs allow us to **extend** the Kubernetes API.

Let's extend the Kubernetes API by creating our very own object/resource via CRDs.

Create the CRD

```
$ cat my-new-crd.yaml

apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: mysql.db.example.com
Spec:
  group: db.example.com
  version: v1
  scope: Namespaced
  names:
    plural: mysqls
    singular: mysql
    kind: MySql
    shortNames:
      - ms

$ kubectl create -f my-new-crd.yaml
```

Let's first verify the creation of the CRD object/resource.

Verify CRD Creation via CLI

```
$ kubectl get crd
```

NAME	KIND
mysql.db.example.com	CustomResourceDefinition.v1beta1.apiextensions.k8s.io

Verify CRD Creation via API

```
curl -XGET localhost:8001/apis/apiextensions.k8s.io/v1beta1/customresourcedefinitions
```

```
{
  "kind": "CustomResourceDefinitionList",
  "apiVersion": "apiextensions.k8s.io/v1beta1",
  "metadata": {
    "selfLink": "/apis/apiextensions.k8s.io/v1beta1/customresourcedefinitions",
    "resourceVersion": "229273"
  },
  "items": [
    {
      "metadata": {
        "name": "mysql.db.example.com",
        "selfLink":
"/apis/apiextensions.k8s.io/v1beta1/customresourcedefinitions/mysql.db.example.c
        "uid": "8e4d17df-b085-11e7-9176-080027b424ef",
        "resourceVersion": "228836",
        "creationTimestamp": "2017-10-14T02:15:32Z"
      },
      . . . .
    }
  ],
  . . . .
}
```

Let's now actually verify our new **mysql** resource/object!

Verify New Database Resource via CLI

```
$ kubectl get mysql
```

```
No resources found.
```

Verify New Database Resource via API

```
curl -XGET localhost:8001/apis/db.example.com/v1/namespaces/default/mysqls
```

```
{
  "apiVersion": "db.example.com/v1",
  "items": [],
  "kind": "MySQLList",
  "metadata": {
    "resourceVersion": "240591",
    "selfLink": "/apis/stable.example.com/v1/namespaces/default/mysqls"
  }
}
```

Let's create a new **database** object.

Create a new **mysql** object

```
$ cat new-mysql-object.yaml
```

```
apiVersion: "db.example.com/v1"  
kind: MySql  
metadata:  
  name: wordpress  
spec:  
  user: wp  
  password: secret  
  foo: bar
```

```
$ kubectl create -f new-mysql-object.yaml
```

Let's verify the creation of the **mysql** object.

Verify database object via CLI

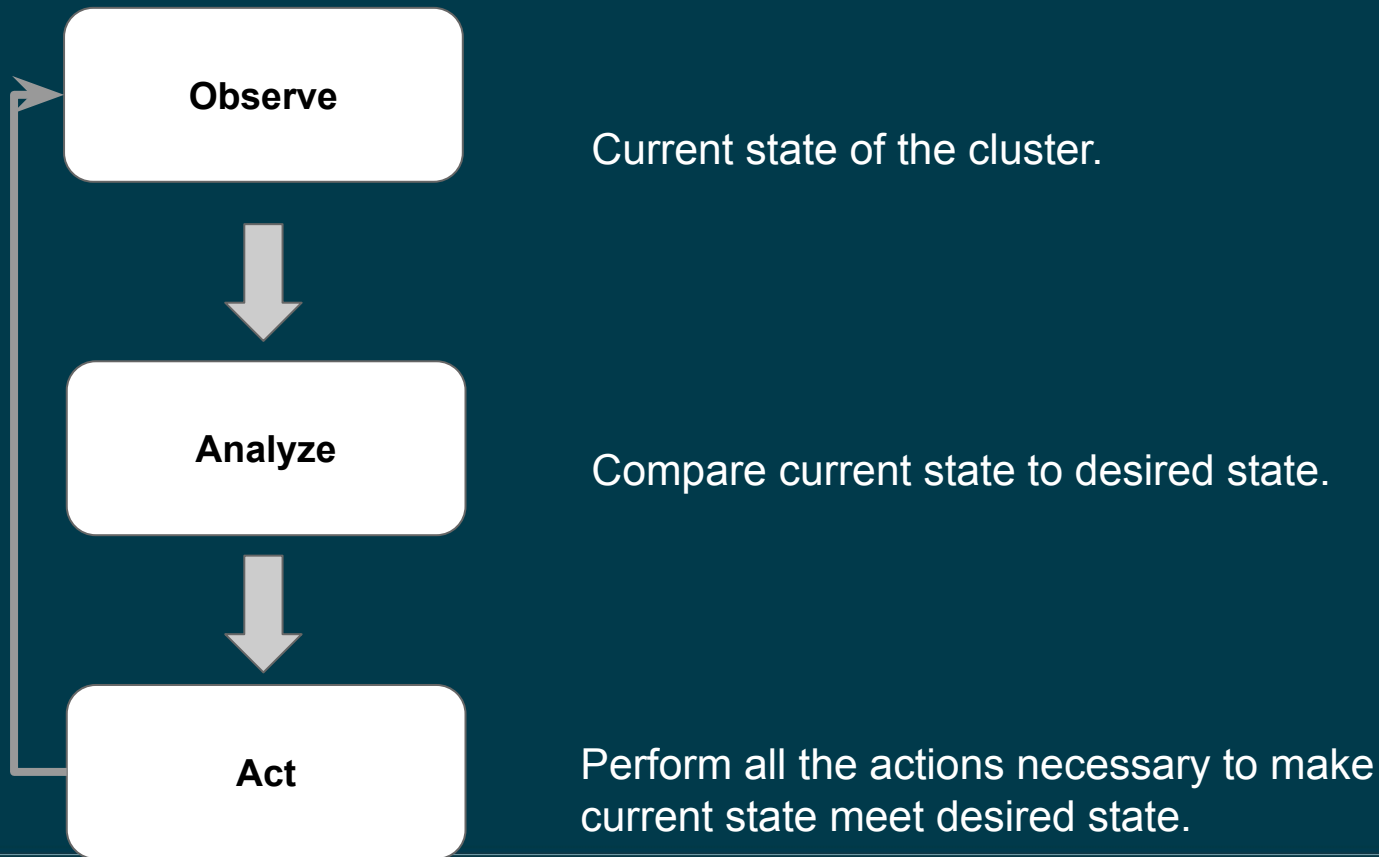
```
$ kubectl get mysql
NAME      AGE
wordpress 5s
```

```
$ kubectl get mysql wordpress -o yaml
```

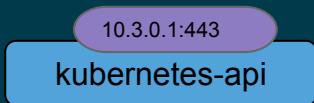
```
apiVersion: db.example.com/v1
kind: MySQL
metadata:
  clusterName: ""
  creationTimestamp: 2017-10-14T03:23:26Z
  deletionGracePeriodSeconds: null
  deletionTimestamp: null
  name: wordpress
  namespace: default
  resourceVersion: "238701"
  selfLink: /apis/db.example.com/v1/namespaces/default/mysqls/wordpress
  uid: 0afd1584-b08f-11e7-9176-080027b424ef
spec:
  foo: bar
  password: secret
  user: wp
```

A Custom Resource needs a controller
to **ACT**
upon its presence.

Kubernetes Controllers



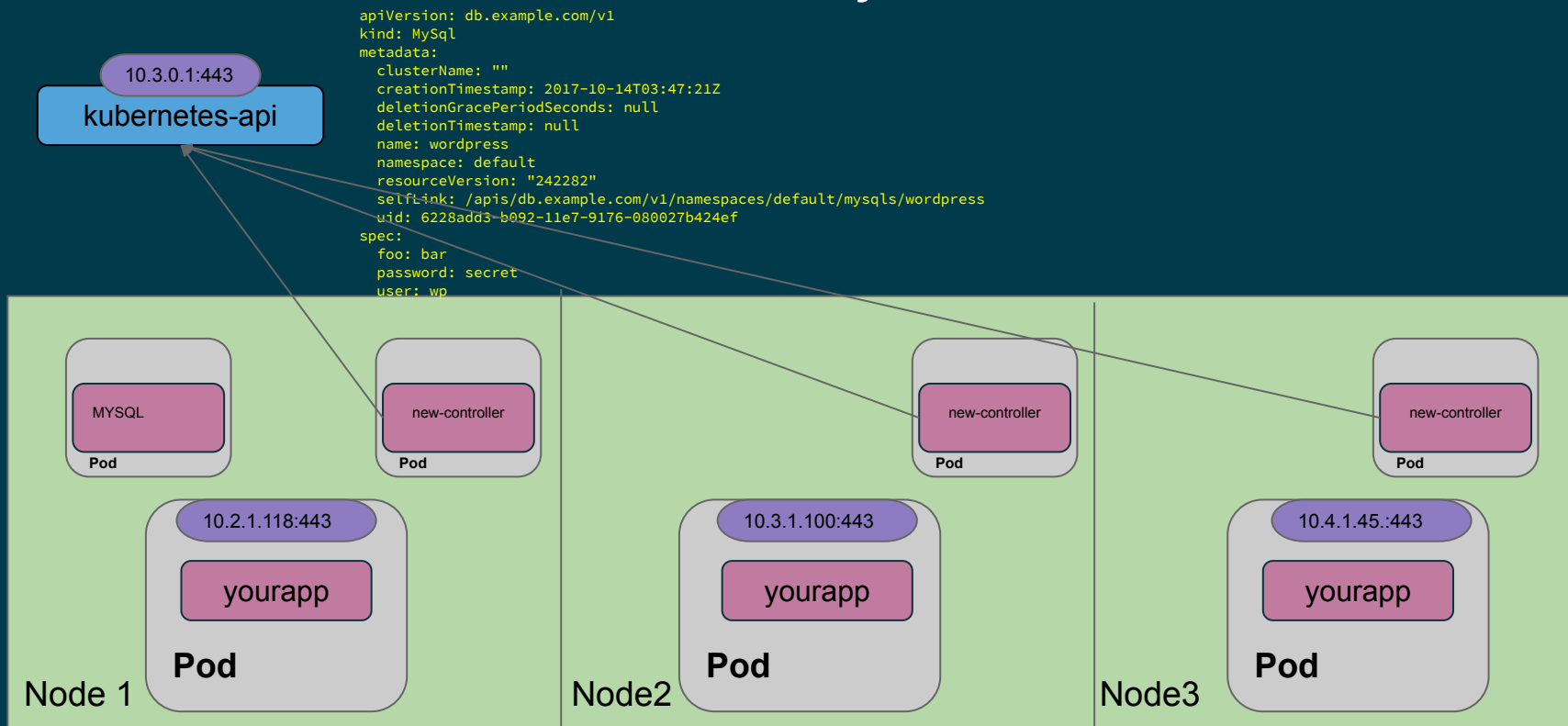
.....and all is quiet..



```
apiVersion: db.example.com/v1
kind: MySQL
metadata:
  clusterName: ""
  creationTimestamp: 2017-10-14T03:47:21Z
  deletionGracePeriodSeconds: null
  deletionTimestamp: null
  name: wordpress
  namespace: default
  resourceVersion: "242282"
  selfLink: /apis/db.example.com/v1/namespaces/default/mysqls/wordpress
  uid: 6228add3-b092-11e7-9176-080027b424ef
spec:
  foo: bar
  password: secret
  user: wp
```



We need a custom controller to notice the new **database** object and **ACT!**



ACT?

CREATE.

READ.

UPDATE.

DELETE.

But that's probably not enough..

- Server startup/shutdown
- Mastering the mysqladmin administrative client
 - Using the mysql interactive client
- User account maintenance
 - Log file maintenance
- Database backup/copying
 - Hardware tuning
- Multiple server setups
- Software updates and upgrades
 - File system security
- Server security
- Repair and maintenance
 - Crash recovery
- Preventive maintenance
- Understanding the mysqld server daemon
 - Performance analysis
- Choosing what else to install (e.g. Apache, Perl +modules, PHP)
- Which version of MySQL (stable, developer, source, binary)
- Creating a user account for the mysql user and group
 - Download and unpack a distribution
- Compile source code and install (or rpm)
- Initialize the data directory and grant tables with mysql_install_db
 - Starting the server
- Installing Perl DBI support
 - Installing PHP
- Installing Apache
- Obtaining and installing the samp_db sample database

- Securing a new MySQL installation
- Running mysqld as an unprivileged user
 - Methods of starting the server
 - Invoking mysqld directly
 - Invoking safe_mysqld
 - Invoking mysql.server
 - Specifying startup options
 - Checking tables at startup
 - Shutting down the server
- Regaining control of the server if you can't connect
- Creating new users and granting privileges
- Determining who can connect from where
 - Who should have what privileges?
 - Administrator privileges
 - Revoking privileges
 - Removing users
- deciding/finding the Data Directory's location
 - Structure of the Data Directory
 - How mysqld provides access to data
- Running multiple servers on a single Data Directory
 - Database representation
- Table representation (form, data and index files)
- OS constraints on DB and table names
- Data Directory structure and performance, resources, security
- MySQL status files (.pid, .err, .log, etc)
- Relocating Data Directory contents

- Creating new users and granting privileges
- Determining who can connect from where
 - Who should have what privileges?
 - Administrator privileges
 - Revoking privileges
 - Removing users
- Methods: mysqldump vs. direct copying
 - Backup policies
 - Scheduled cycles
 - Update logging
- Consistent and comprehensible file-naming
 - Backing up the backup files
 - Off-site / off-system backups
- Backing up an entire database with mysqldump
 - Compressed backup files
 - Backing up individual tables
- Using mysqldump to transfer databases to another server
- mysqldump options (flush-logs, lock-tables, quick, opt)
 - Direct copying methods
- Database replication (live and off-line copying)
 - Recovering an entire database
 - Recovering grant tables
- Recovering from mysqldump vs. tar/cpio files
- Using update logs to replay post-backup queries
- Editing update logs to avoid replaying erroneous queries
 - Recovering individual tables
 - Default parameters

To recap...

Custom Resource Definitions (CRD)

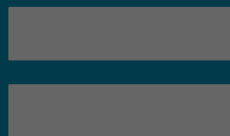
~~Third Party Resources (TPR)~~



Custom Controller



Your Knowledge!



Operators!




Why do Operators matter?

We want an “as-a-service” platform experience!

Build an ecosystem of software on Kubernetes that
can be as easy, safe, and reliable to use and
operate as a Cloud Service.

Low-touch, remotely managed, one-click-updates.

Operator Examples

NAME ↑		STATUS	
	etcd 0.6.1 by CoreOS, Inc	Enabled (1 namespace) Show namespace	<button>Enable</button>
	Prometheus 0.14.0 by CoreOS, Inc	Enabled (1 namespace) Show namespace	<button>Enable</button>
	Vault 0.1.3 by CoreOS, Inc	Enabled (1 namespace) Show namespace	<button>Enable</button>

Super easy to deploy an Operator in a
Kubernetes environment.

Create the CRD

```
$ cat etcd-cluster-crd.yaml
```

```
apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name:
etcdclusters.etcd.database.coreos.com
spec:
  group: etcd.database.coreos.com
  names:
    kind: EtcdCluster
    listKind: EtcdClusterList
    plural: etcdclusters
    shortNames:
    - etcdclus
    - etcd
    singular: etcdcluster
  scope: Namespaced
  version: v1beta2
  versions:
  - name: v1beta2
    served: true
    storage: true
```

Deploy etcd Operator

```
$ cat etcd-operator.yaml
```

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: etcd-operator
spec:
  replicas: 1
  template:
    metadata:
      labels:
        name: etcd-operator
    spec:
      containers:
      - name: etcd-operator
        image: quay.io/coreos/etcd-operator:v0.9.2
        command:
        - etcd-operator
        # Uncomment to act for resources in all namespaces. More information in doc/clusterwide.md
        #- -cluster-wide
        env:
        - name: MY_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
        - name: MY_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
```

Deploy etcd Operator

```
$ kubectl create -f etcd-operator.yaml
```

```
$ kubectl get pods
```

NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
default	etcd-operator-67666dc65f-xwfvq	1/1	Running	0	1s

View the etcdCluster Custom Resource

```
$ cat etcd-instance.yaml
```

```
apiVersion: "etcd.database.coreos.com/v1beta2"
kind: "EtcdCluster"
metadata:
  name: "example-etcd-cluster"
spec:
  size: 3
  version: "3.2.13"
```

Deploy etcdCluster

```
$ kubectl create -f etcd-instance.yaml
```

```
$ kubectl get etcdcluster
```

NAMESPACE	NAME	AGE
default	myetcdcluster	1s

How do you create your own Operator?

Life before the Operator SDK...

If only it were as simple as....

Resources

```
type MyCustomResourceDefinition struct
{
    // API obj kind & schema version
    metav1.TypeMeta

    // Standard object metadata (optional)
    Metadata api.ObjectMeta

    // Describe how the resource appears
    Spec v1beta1.CustomResourceDefinitionSpec

    // State of the CRD
    Status CustomResourceDefinitionStatus
}
```

Controllers

```
for {
    current := getCurrentState()
    desired := getDesiredState()
    makeChanges(current, desired)
}
```

Custom Operators require
many building blocks and
boilerplate code.

...research/download tools to
interact with the API.

Kubernetes Client Libraries

Officially-supported Kubernetes client libraries

The following client libraries are officially maintained by [Kubernetes SIG API Machinery](#).

Language	Client Library	Sample Programs
Go	github.com/kubernetes/client-go/	browse
Python	github.com/kubernetes-client/python/	browse
Java	github.com/kubernetes-client/java	browse
dotnet	github.com/kubernetes-client/csharp	browse
JavaScript	github.com/kubernetes-client/javascript	browse

Community-maintained client libraries

The following Kubernetes API client libraries are provided and maintained by their authors, not the Kubernetes team.

Language	Client Library
Clojure	github.com/yanetan16/clj-kubernetes-api
Go	github.com/ericchiang/k8s
Java (OSGi)	bitbucket.org/amdatulabs/amdatu-kubernetes
Java (Fabric8, OSGi)	github.com/fabric8io/kubernetes-client
Lisp	github.com/brendandburns/cl-k8s
Node.js (TypeScript)	github.com/Ggyoo/node-k8s-client

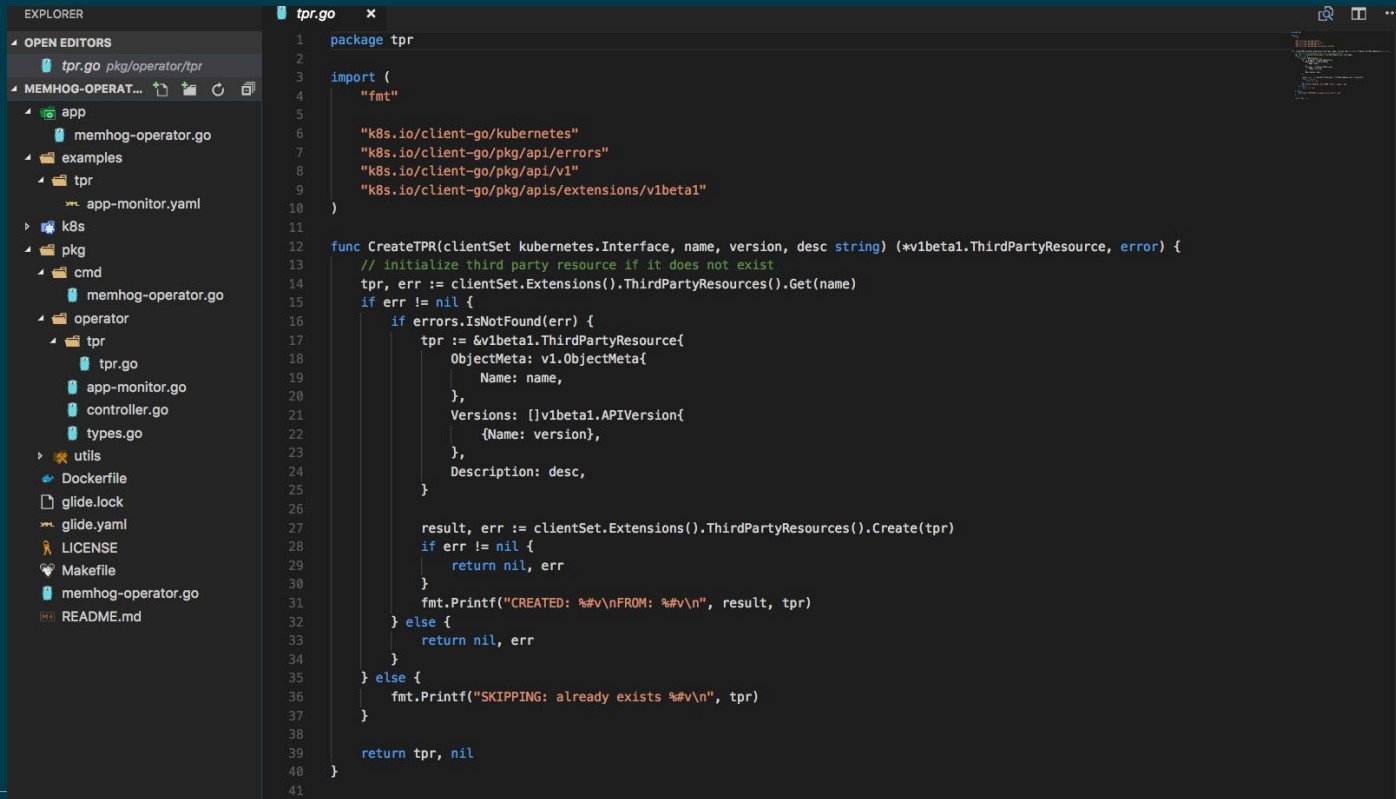
Knowledge of informers/shared
informers for object cache
and event handling.

Communicating desired
state/actual state via annotations.

Tracking kube-related resources.

Test scaffolding & repo organization.

Custom Operator Source



```
1 package tpr
2
3 import (
4     "fmt"
5
6     "k8s.io/client-go/kubernetes"
7     "k8s.io/client-go/pkg/api/errors"
8     "k8s.io/client-go/pkg/api/v1"
9     "k8s.io/client-go/pkg/apis/extensions/v1beta1"
10 )
11
12 func CreateTPR(clientSet kubernetes.Interface, name, version, desc string) (*v1beta1.ThirdPartyResource, error) {
13     // Initialize third party resource if it does not exist
14     tpr, err := clientSet.Extensions().ThirdPartyResources().Get(name)
15     if err != nil {
16         if errors.IsNotFound(err) {
17             tpr := &v1beta1.ThirdPartyResource{
18                 ObjectMeta: v1.ObjectMeta{
19                     Name: name,
20                 },
21                 Versions: []v1beta1.APIVersion{
22                     {Name: version},
23                 },
24                 Description: desc,
25             }
26
27             result, err := clientSet.Extensions().ThirdPartyResources().Create(tpr)
28             if err != nil {
29                 return nil, err
30             }
31             fmt.Printf("CREATED: %#v\nFROM: %#v\n", result, tpr)
32         } else {
33             return nil, err
34         }
35     } else {
36         fmt.Printf("SKIPPING: already exists %#v\n", tpr)
37     }
38
39     return tpr, nil
40 }
41 }
```

We need an easier way
to **create** Operators.

We need an easier way
to **manage** Operators.



Operator Framework

The Operator Framework is an open source toolkit to manage Kubernetes native applications, called Operators, in an effective, automated, and scalable way.

<https://operatorhub.io/>

Verified

Repositories 24

People 59

Teams 2

Pinned repositories

operator-sdk

SDK for building Kubernetes applications.
Provides high level APIs, useful abstractions, and project scaffolding.

Go ★ 1.8k 🍴 404

operator-lifecycle-manager

A management framework for extending
Kubernetes with Operators

Go ★ 332 🍴 141

operator-metering

Operator metering is responsible for collecting
metrics and other information about what's
happening in a Kubernetes cluster, and providing
a way to create reports on the collected data.

Go ★ 165 🍴 37

Find a repository...

Type: All ▾

Language: All ▾

New

community-operators

The canonical source for Kubernetes Operators that appear on
OperatorHub.io, OpenShift Container Platform and OKD.

Shell ★ 120 🍴 149 Updated 6 hours ago



Top languages

Go Java Shell JavaScript
Python

Most used topics

Manage

kubernetes operator

operator-metering

Operator-SDK

3



ANSIBLE



Operator Lifecycle Manager (OLM)

Enable cluster admins to manage Operators on any Kubernetes cluster (dependency management).

Operator Metering

Record historical cluster usage and generate usage reports showing usage breakdowns by pod or namespace over arbitrary time periods.

OperatorHub.io

OperatorHub.io

Search OperatorHub...

Contribute

Welcome to OperatorHub.io

OperatorHub.io is a new home for the Kubernetes community to share Operators. Find an existing Operator or list your own today.

PROVIDER

- ☐ Amazon Web Services (1)
- ☐ CNCF (1)
- ☐ Couchbase (1)
- ☐ Crunchy Data Solutions (1)
- ☐ Dynatrace (1)

[Show 6 more](#)

CAPABILITY LEVEL

- ☐ Basic Install (5)
- ☐ Seamless Upgrades (2)
- ☐ Full Lifecycle (6)

13 ITEMS

VIEW SORT A-Z

AWS Service Operator
provided by Amazon Web Services, Inc.

The AWS Service Operator allows you to manage AWS

Couchbase Operator
provided by Couchbase

The Couchbase Autonomous Operator allows users to easily deploy, manage, and maintain

Dynatrace OneAgent
provided by Dynatrace LLC

Install full-stack monitoring of Kubernetes clusters with the Dynatrace OneAgent.

etcd
provided by CNCF

Creates and maintain highly-available etcd clusters on Kubernetes

Jaeger Tracing
provided by Jaeger

Provides tracing, monitoring and troubleshooting microservices-based

Kubernetes Federation
provided by Red Hat

Gain Hybrid Cloud capabilities between your clusters with Kubernetes Federation.

MongoDB
provided by MongoDB, Inc

The MongoDB Enterprise Kubernetes Operator enables easy deploys of MongoDB

Percona Xtradb Cluster Operator
provided by Percona

Percona MySQL Operator manages the lifecycle of

Follow Along with Us!

workshop.coreostrain.me

username: operator

password: coreostrainme

Katacoda!



<https://learn.openshift.com/training>