Red Hat OpenShift

Operator-Framework Workshop

automated, effective, and scalable operators

Matthew Dorn
Principal Engineer

Melvin Hillsman
Service Reliability Engineer

Red Hat

Red Hat OpenShift

Operator-Framework Workshop

automated, effective, and scalable operators

Red Hat Operator Enablement Team

Red Hat

Red Hat Home Exercises Support

Support

Workshop Content
Download Slides

Kubernetes Operators E-Book
Download a free copy of the O'Reilly Kubernetes Operators E-Book

Operator Hub
www.operatorhub.io

Submit Your Operator
github.com/operator-framework/community-operators

Chat
#kubernetes-operators

Mailing List
operator-framework

GitHub Issues
operator-sdk

OpenShift Commons - Operator Framework
Every third Tuesday of the month at 9am Pacific

Red Hat OpenShift

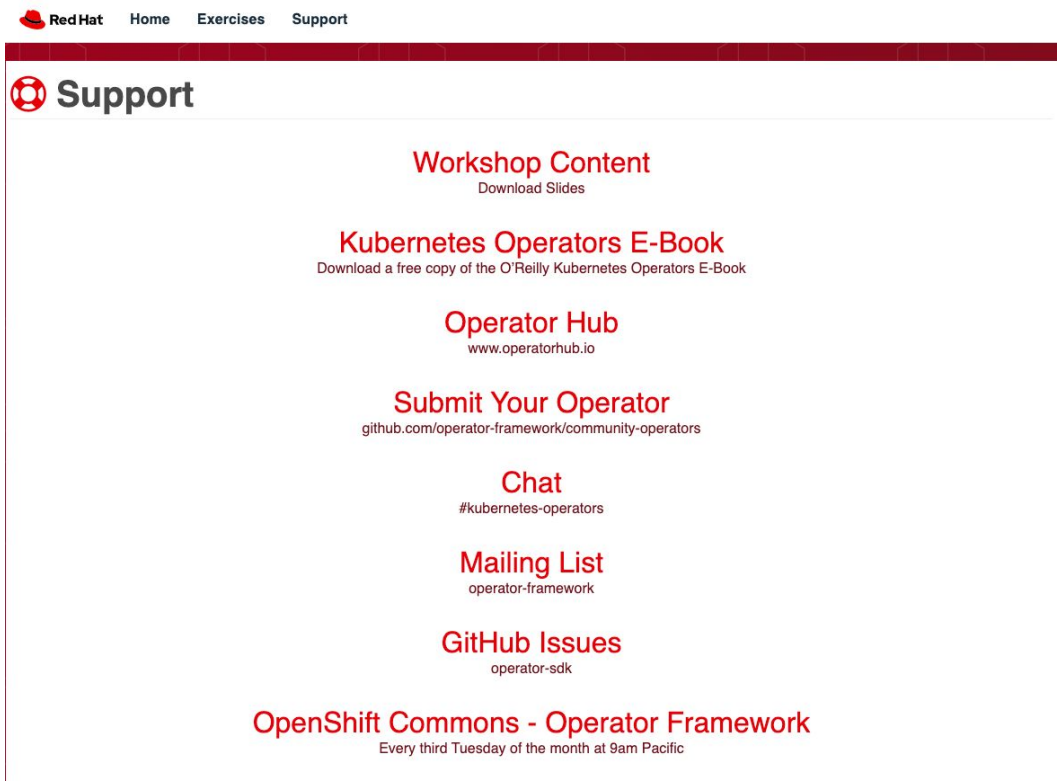
PRODUCTS LEARN COMMUNITY SUPPORT FREE TRIAL REPORT AN ISSUE

Interactive Learning Portal

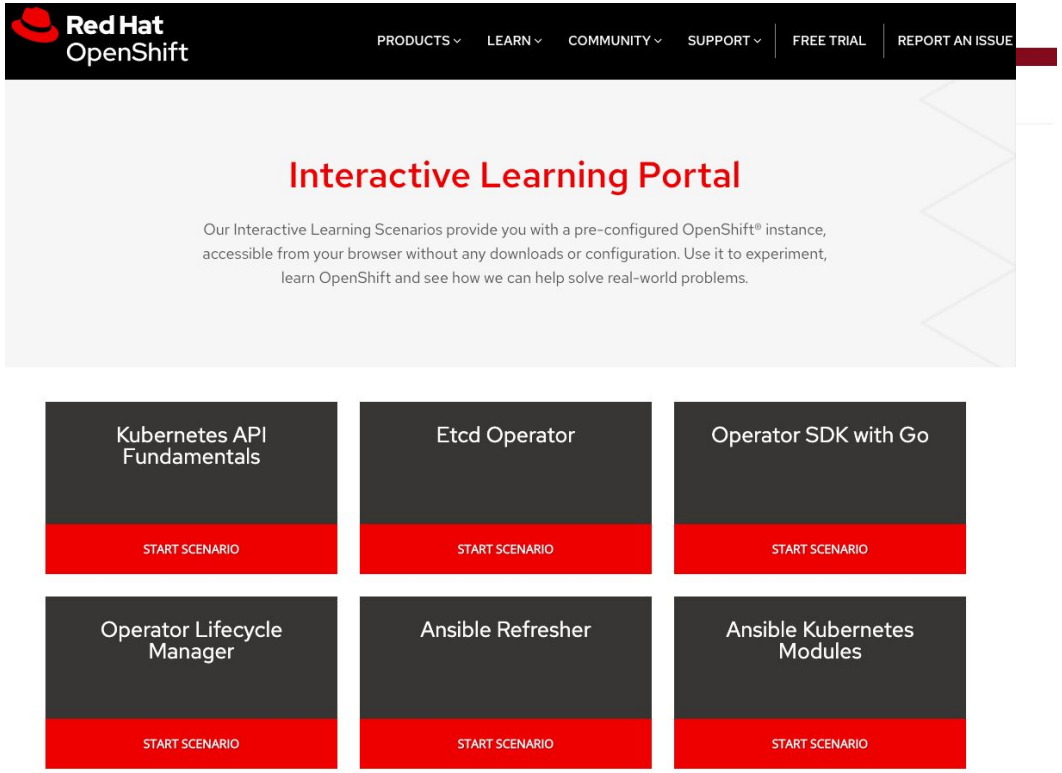
Our Interactive Learning Scenarios provide you with a pre-configured OpenShift® instance, accessible from your browser without any downloads or configuration. Use it to experiment, learn OpenShift and see how we can help solve real-world problems.

Kubernetes API Fundamentals START SCENARIO	Etcd Operator START SCENARIO	Operator SDK with Go START SCENARIO
Operator Lifecycle Manager START SCENARIO	Ansible Refresher START SCENARIO	Ansible Kubernetes Modules START SCENARIO

workshop.coreostrain.me/support



learn.openshift.com/training



Red Hat

- How many people here use Kubernetes and/or OpenShift regularly?
- Who develops in Golang regularly?
- Does anyone use Ansible regularly?
- How many of you have attempted to build a custom Kubernetes controller from scratch?
- Has anyone tried the Operator-SDK, Ansible-Operator, or Helm-App Operator?

USE KUBERNETES/OPENSHIFT REGULARLY?

DEVELOP IN GOLANG REGULARLY?

USE ANSIBLE REGULARLY?

ATTEMPTED OR SUCCEEDED TO BUILD A CUSTOM KUBERNETES CONTROLLER FROM SCRATCH?

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

HERE TO HELP YOU (AND OUR INTERNAL/EXTERNAL PARTNERS) **SUCCEED** WITH OUR TOOLS.

Red Hat

OPERATOR FRAMEWORK

Sources:
https://github.com/operator-framework

OPERATOR
SDK

OPERATOR
METERING

OPERATOR
LIFECYCLE MANAGER

Sources:
https://github.com/operator-framework/operator-sdk
https://github.com/operator-framework/operator-lifecycle-manager
https://github.com/operator-framework/operator-metering

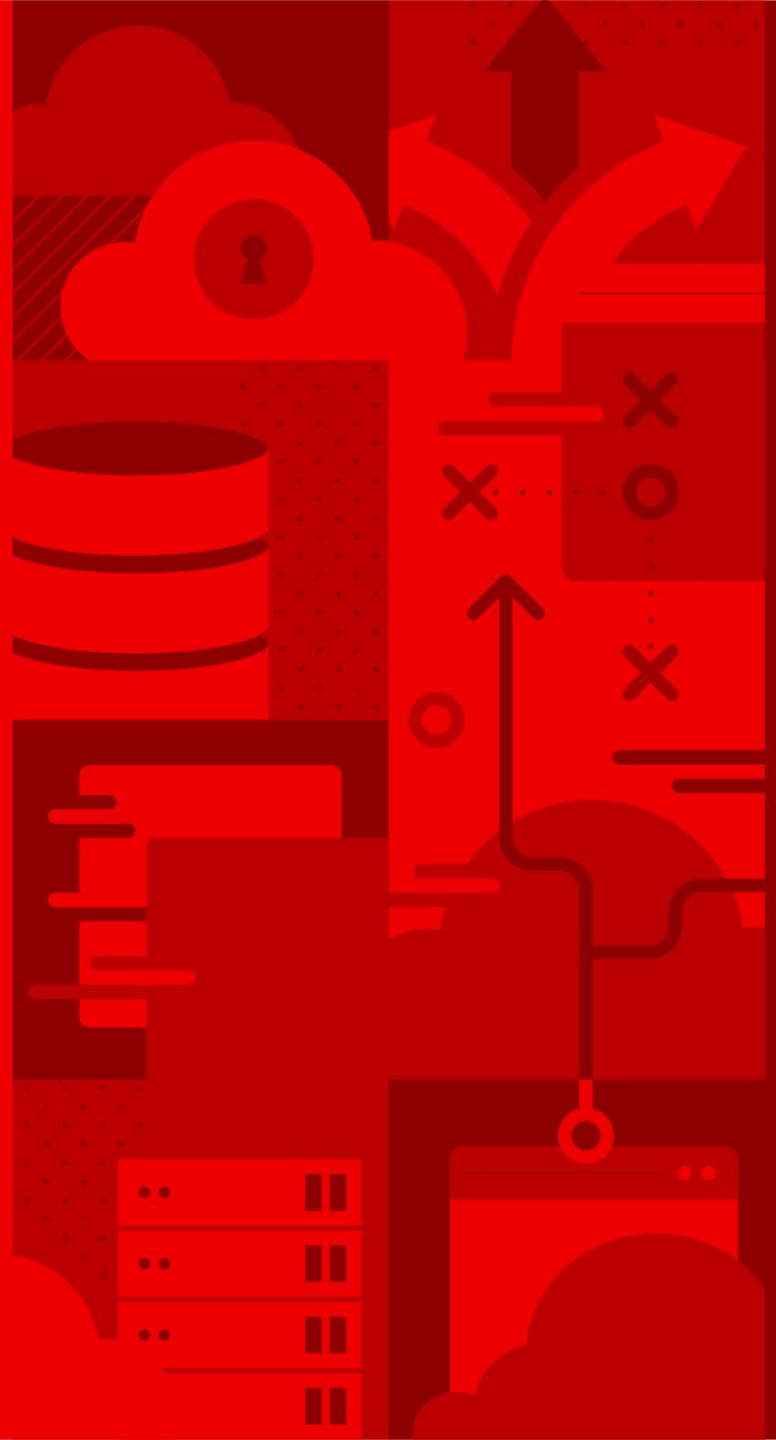
What we'll discuss today

What is an Operator

- History
- Resources
- Controllers
- Application and/or Domain Specific Knowledge

WHAT IS AN **OPERATOR**?

Red Hat



Operators

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

Red Hat

What is an Operator

← 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 instan 1 f comple 2 teful applications on behalf of a Kubernet 3 er. It builds upon the basic Kubernetes resource and controller concepts but includes domain or application-specific knowledge to automate common tasks.

Source:
https://coreos.com/blog/introducing-operators.html

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

1

Resource

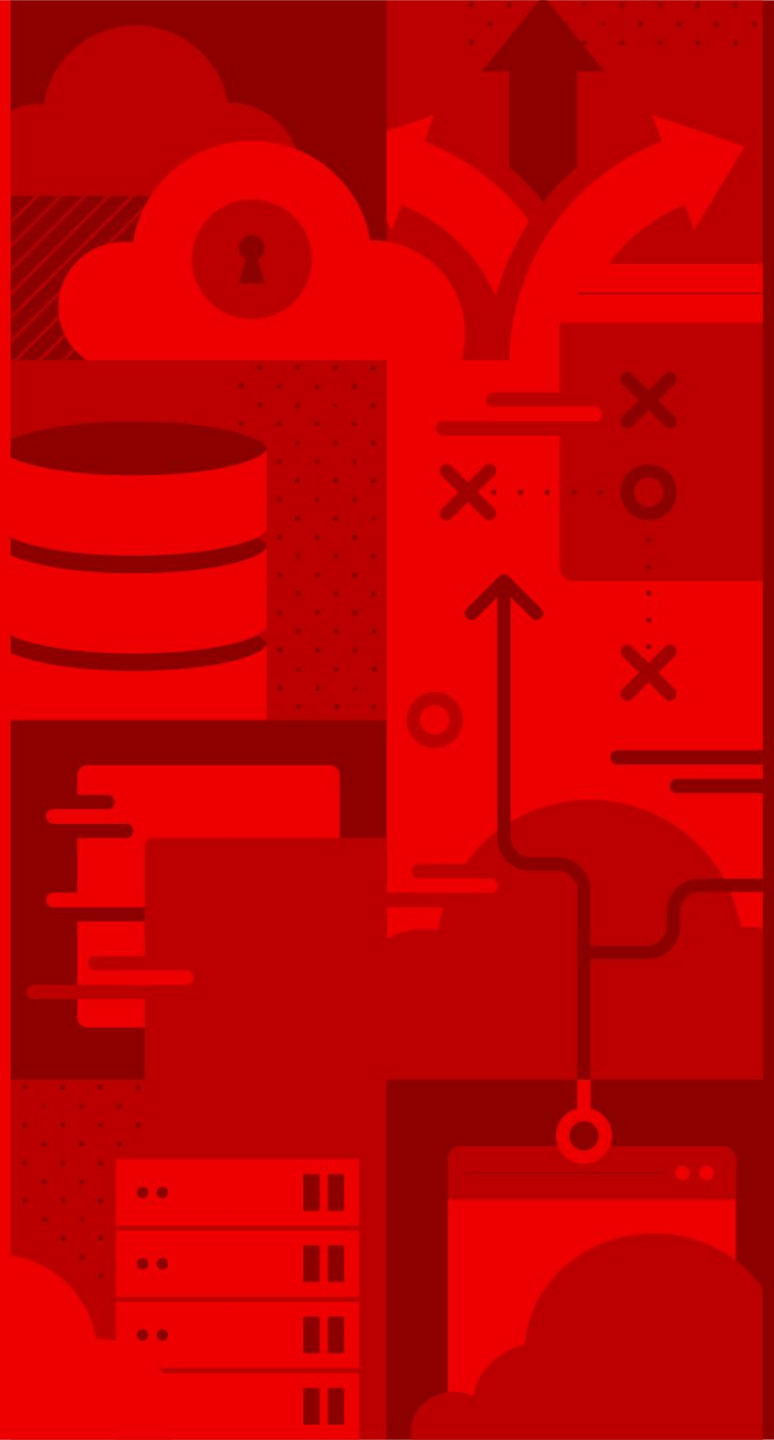
2

Controller

3

Knowledge

Source:
https://coreos.com/blog/introducing-operators.html



Resource

an endpoint in the Kubernetes API that stores a collection of API objects of a certain kind

Red Hat

Pod

the basic execution unit of a Kubernetes application–the smallest and simplest unit in the Kubernetes object model that you create or deploy. A Pod represents processes running on your Cluster.

Source:
https://kubernetes.io/docs/concepts/workloads/pods/pod/

ConfigMap

provides a way to inject configuration data into Pods. The data stored in a ConfigMap object can be referenced in a volume of type configMap and then consumed by containerized applications running in a Pod.

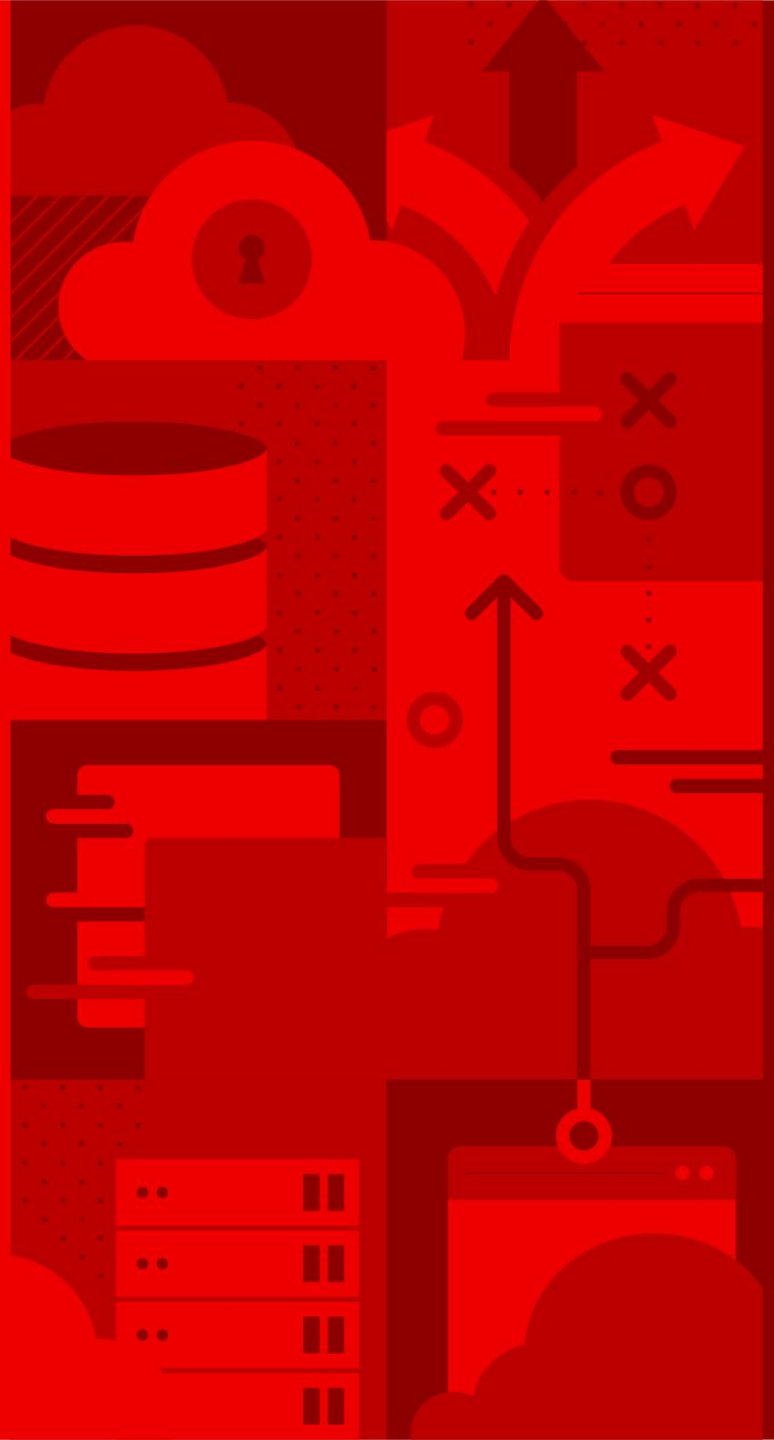
Source:
https://kubernetes.io/docs/concepts/storage/volumes/#configmap

Route (Ingress)

a way to expose a service by giving it an externally-reachable hostname like www.example.com.

Source:
https://docs.openshift.com/enterprise/3.0/architecture/core_concepts/routes.html

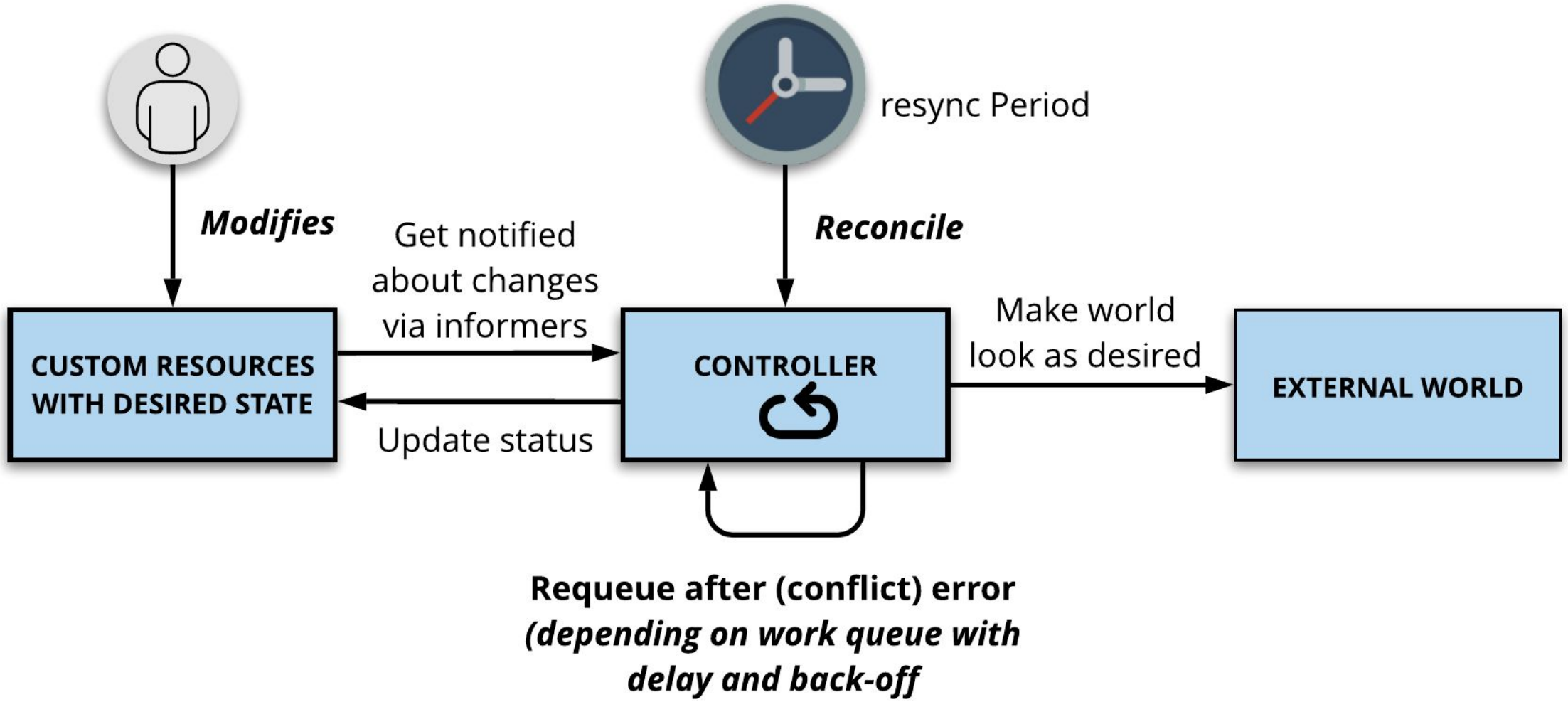
Red Hat



Controller

control loop that watches the state of your cluster and moves the current cluster state closer to the desired state

Red Hat


resync Period
Modifies
Reconcile
Get notified
about changes
via informers
CUSTOM RESOURCES
WITH DESIRED STATE
CONTROLLER
Make world
look as desired
EXTERNAL WORLD
Update status
Requeue after (conflict) error
(depending on work queue with
delay and back-off

ReplicaSet Controller

defined with fields, including a selector that specifies how to identify Pods it can acquire, a number of replicas indicating how many Pods it should be maintaining, and a pod template specifying the data of new Pods it should create to meet the number of replicas criteria.

Source:
https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

Deployment Controller

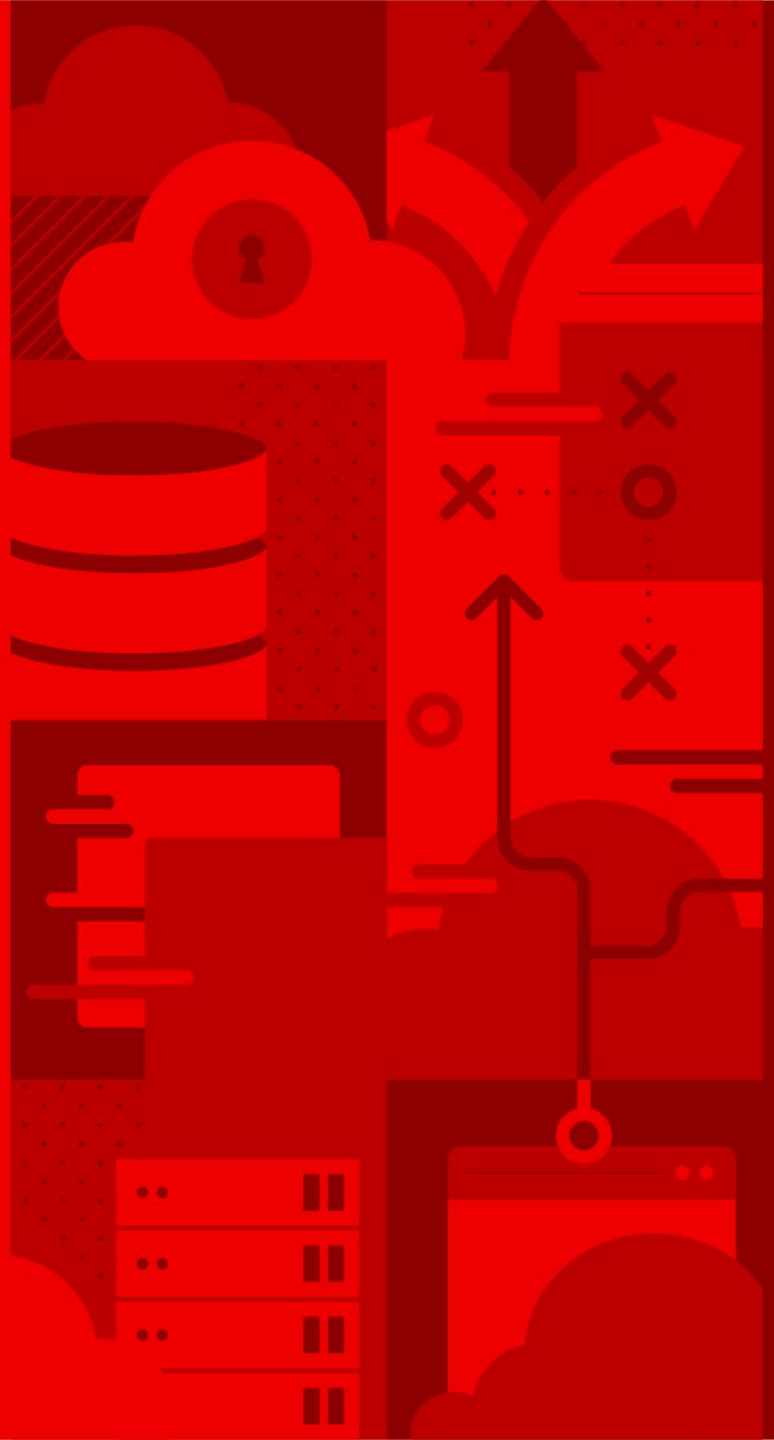
provides declarative updates for Pods and ReplicaSets. You describe a desired state in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate.

Source:
https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

DaemonSet Controller

ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected.

Source:
https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/



Knowledge

domain or application specific; usually must be learned from users and/or administrators rather than developers

Red Hat

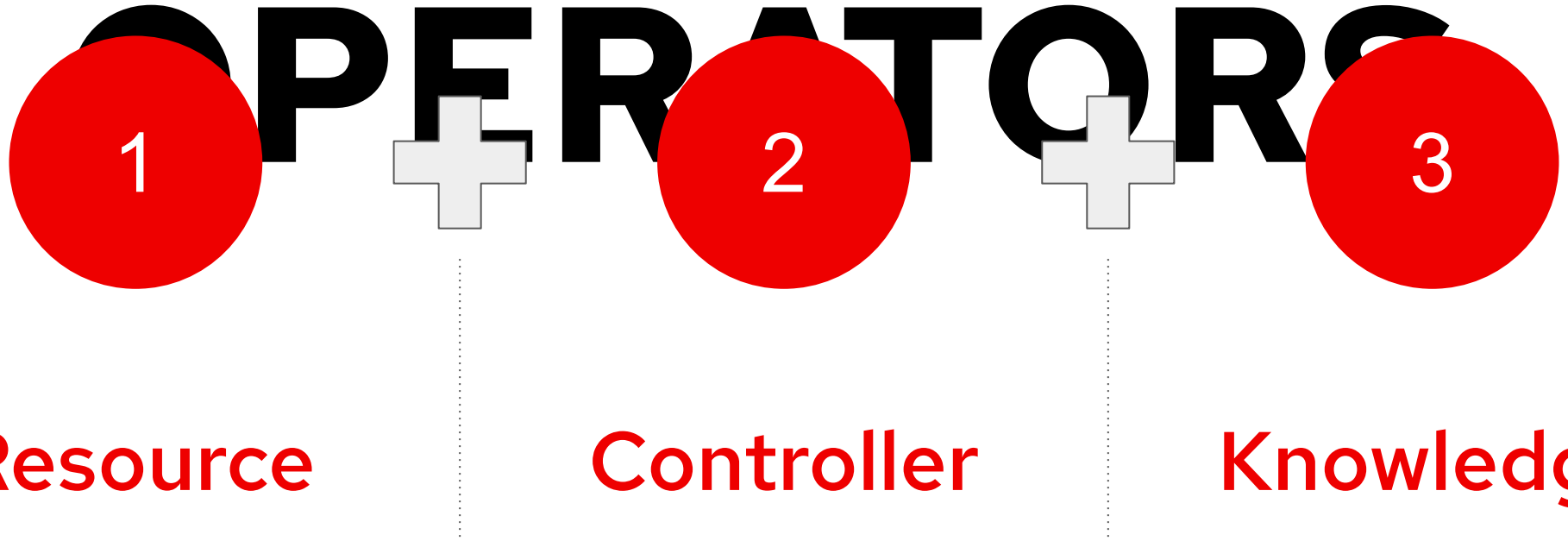
Domain or Application Specific Knowledge

real-world experience with managing your application(s)

- Install
- Self Heal
- Scale
- Update
- Backup
- Clean Up
- Observability
- Resiliency

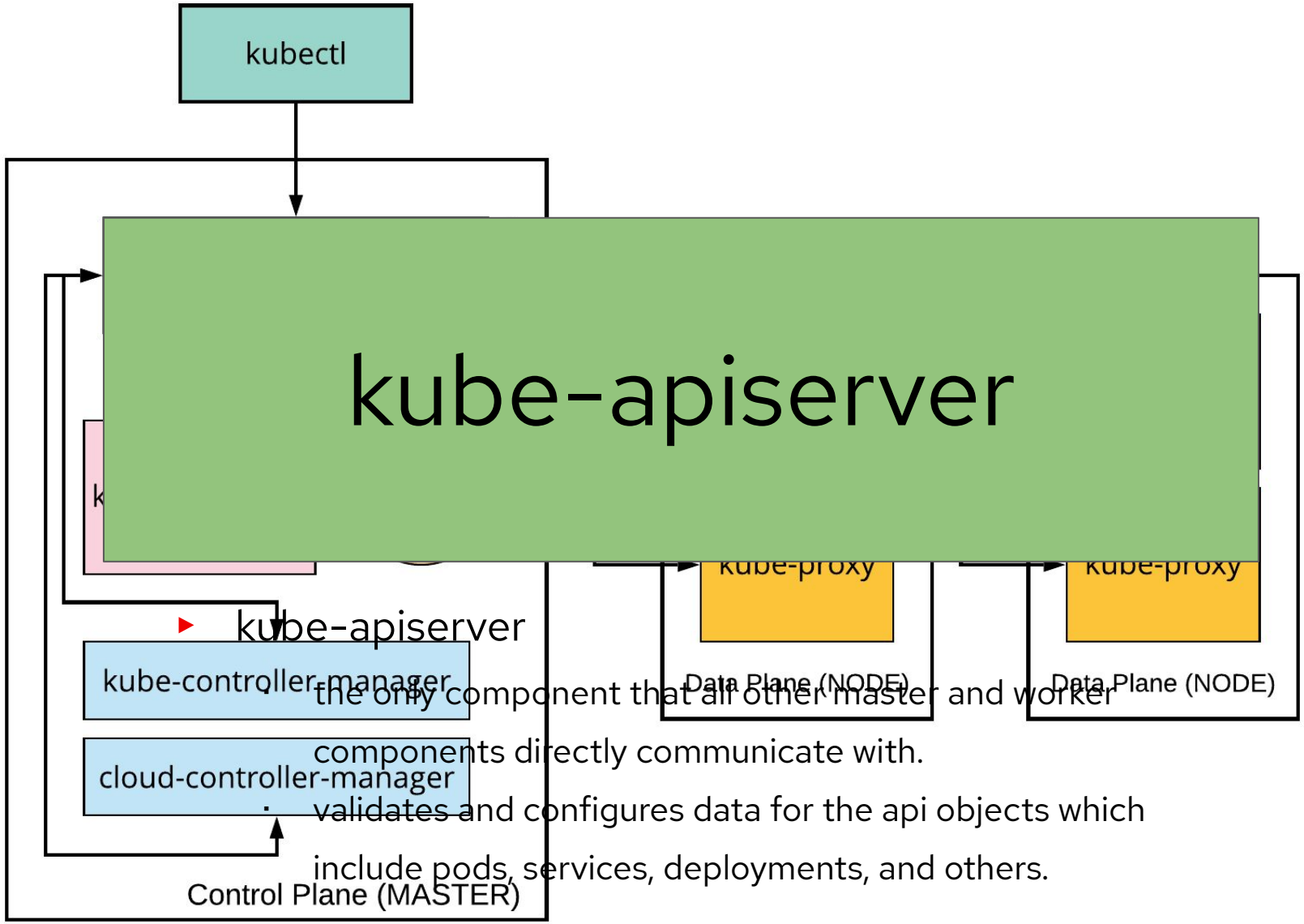
Red Hat

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



Source:
https://coreos.com/blog/introducing-operators.html

An Operator takes advantage of what Kubernetes does best



kube-apiserver

- kube-apiserver
  - the only component that all other master and worker components directly communicate with.
  - validates and configures data for the api objects which include pods, services, deployments, and others.

Source:
https://medium.com/@karthikeyan_krishnaswamy/overview-of-kubernetes-34d8e0e59b26

```
oc api-resources -v6 --api-group=

oc proxy

curl localhost:8001/api/v1
```

```
curl -s localhost:8001/api/v1 | jq -r .resources[].name

bindings
componentstatuses
configmaps
endpoints
events
limitranges
namespaces
namespaces/finalize
namespaces/status
nodes
...
```

```
redhat:mhillsma deploy $ oc get -n openshift-dns pods
NAME                READY   STATUS    RESTARTS   AGE
...
dns-default-vxvth   3/3     Running   0          5d8h
```

```
(curl -s -XGET localhost:8001/api/v1/namespaces/openshift-dns/pods | jq -r
.items[].metadata.name)

"dns-default-478pn"
"dns-default-4fv5s"
"dns-default-vxvth"
"dns-default-7k289"
"dns-default-fw7gv"
"dns-default-j7mzv"
```

```
redhat:mhillsma deploy $ oc get -n openshift-dns pod/dns-default-vxvth -o yaml
apiVersion: v1
kind: Pod
metadata:
...
  name: dns-default-vxvth

(curl -XGET localhost:8001/api/v1/namespaces/openshift-dns/pods/dns-default-vxvth)

apiVersion: v1
kind: Pod
metadata:
  name: dns-default-vxvth
  namespace: openshift-dns
  ownerReferences:
...
```

Operators take advantage of Custom Resource Definitions

(formerly called Third Party Resources – TPRs)

CRDs allow us to **EXTEND** the Kubernetes API

- modify the API without recompiling
- create our very own API resource/object
- resource/object exists but nothing acts on its presence and this is where controllers come in

Walkthrough: Creating a Custom Resource Definition

Red Hat

Check for existence of the MySQL resource/object

oc get mysql

error: the server doesn't have a resource type "mysql"

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

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

oc create -f my-new-crd.yaml
```

Verify creation of the resource/object (CRD) via CLI

```
oc get crd

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

Verify creation of the resource/object (CRD) via API

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

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

Verify existence of the resource/object (mysql) via CLI

oc get mysql

No resources found.

Verify existence of the resource/object (mysql) via API

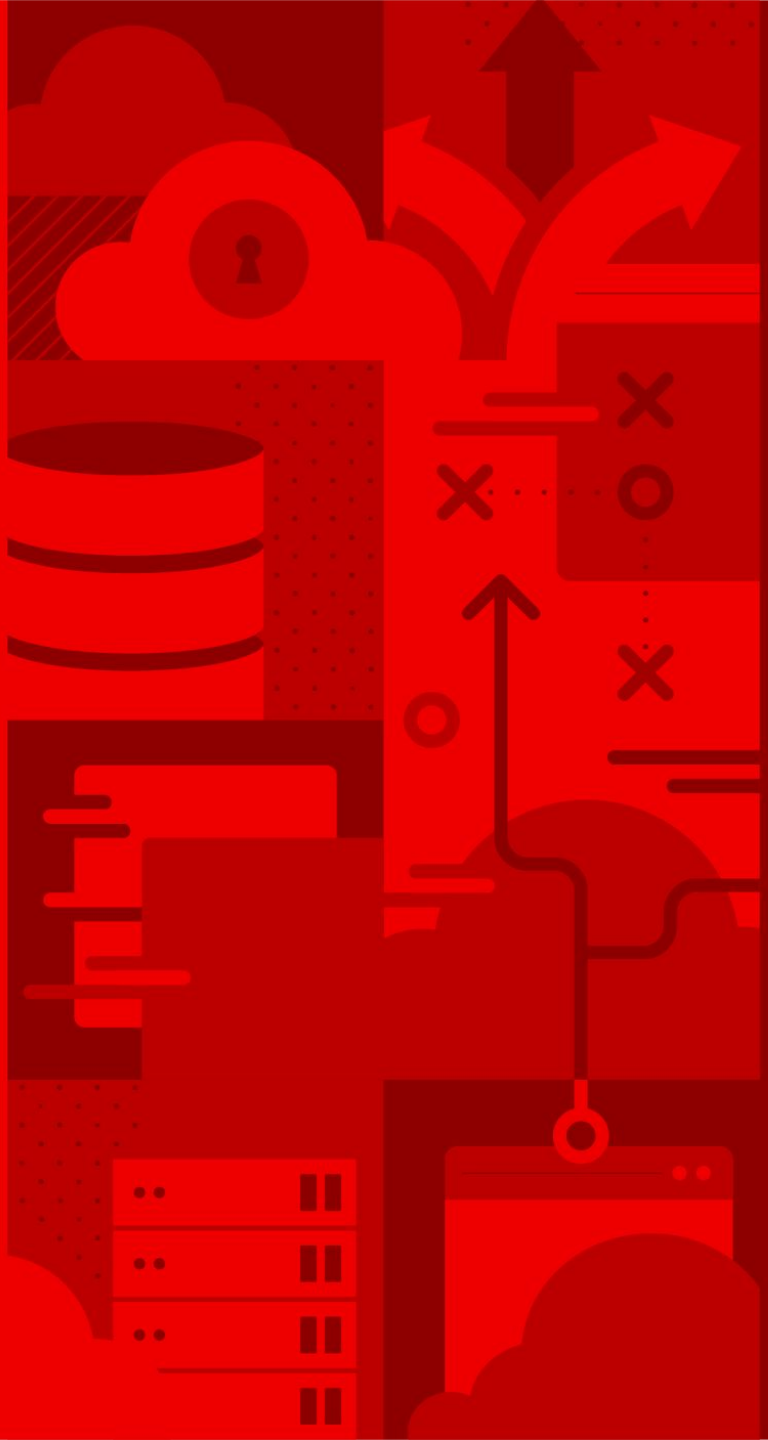
```
curl -XGET localhost:8001/apis/db.example.com/v1/namespaces/default/mysqls
```


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

```
$ 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
```

```
$ oc get mysql

NAME        AGE
wordpress   5s
```

```
$ oc 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
```

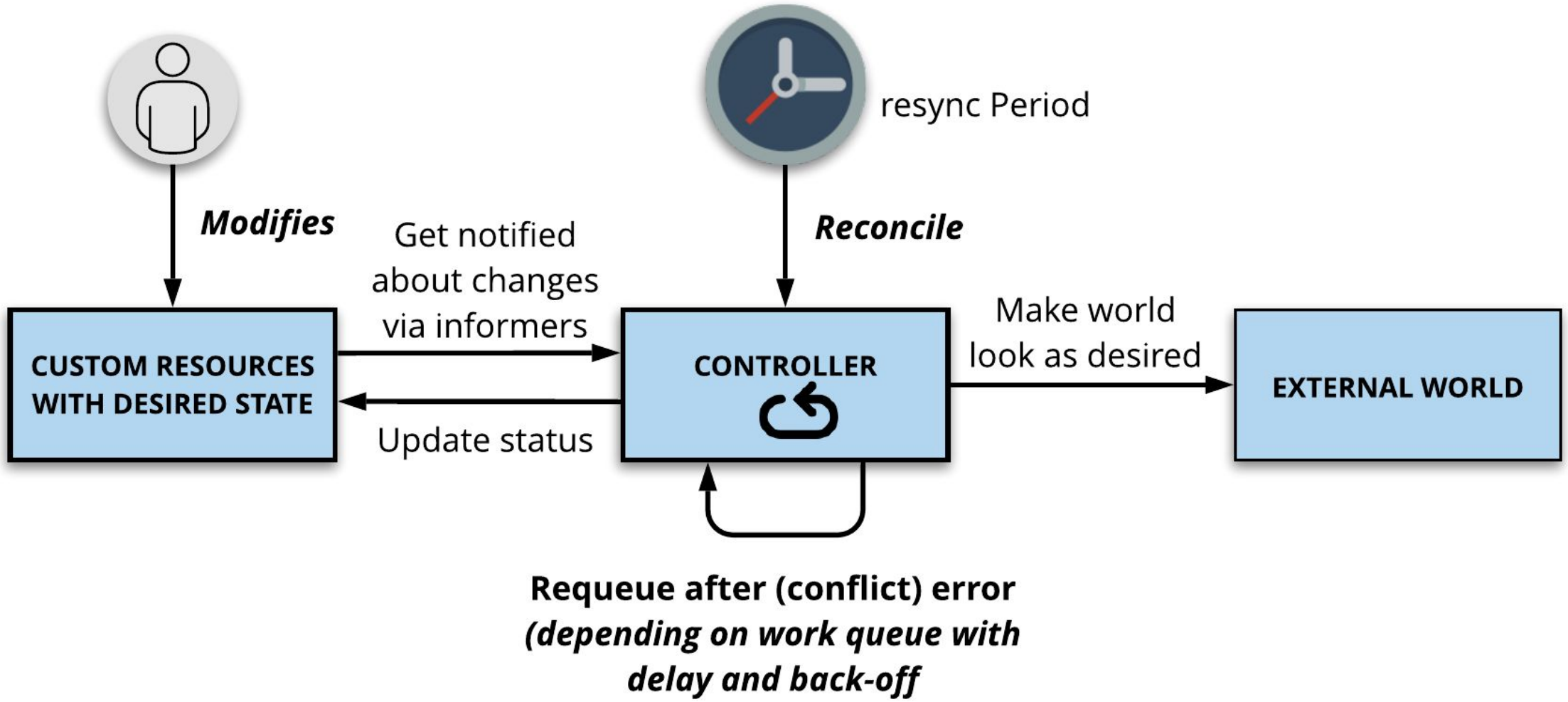


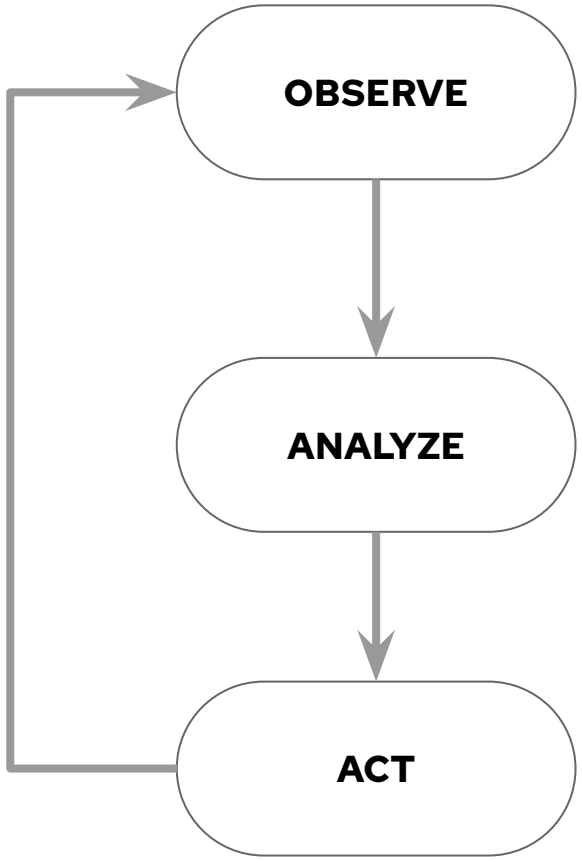
A Custom Resource
needs a controller
to **ACT**
upon its presence.

Red Hat

What do we mean by **ACT**?

- Create
- Read
- Update
- Delete

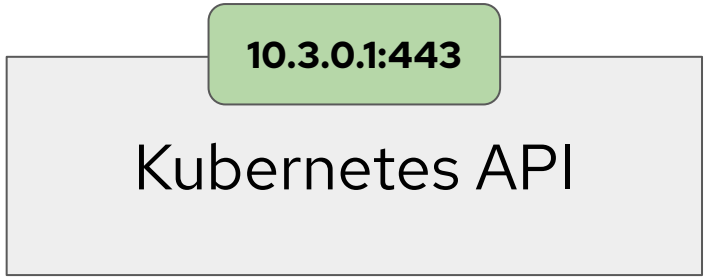

resync Period
Modifies
Reconcile
Get notified
about changes
via informers
Make world
look as desired
CUSTOM RESOURCES
WITH DESIRED STATE
CONTROLLER
EXTERNAL WORLD
Update status
Requeue after (conflict) error
(depending on work queue with
delay and back-off



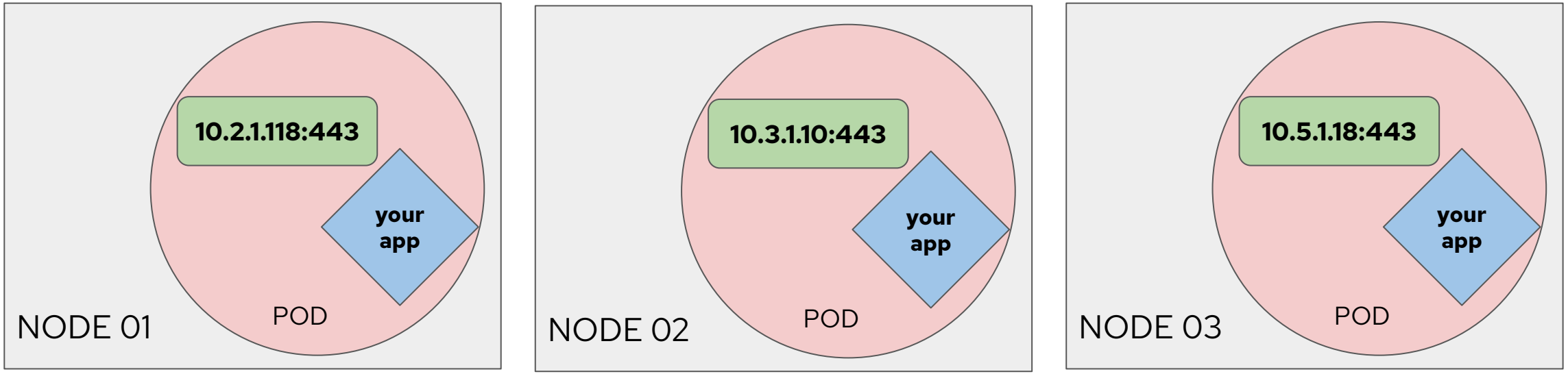
Current state of the cluster.

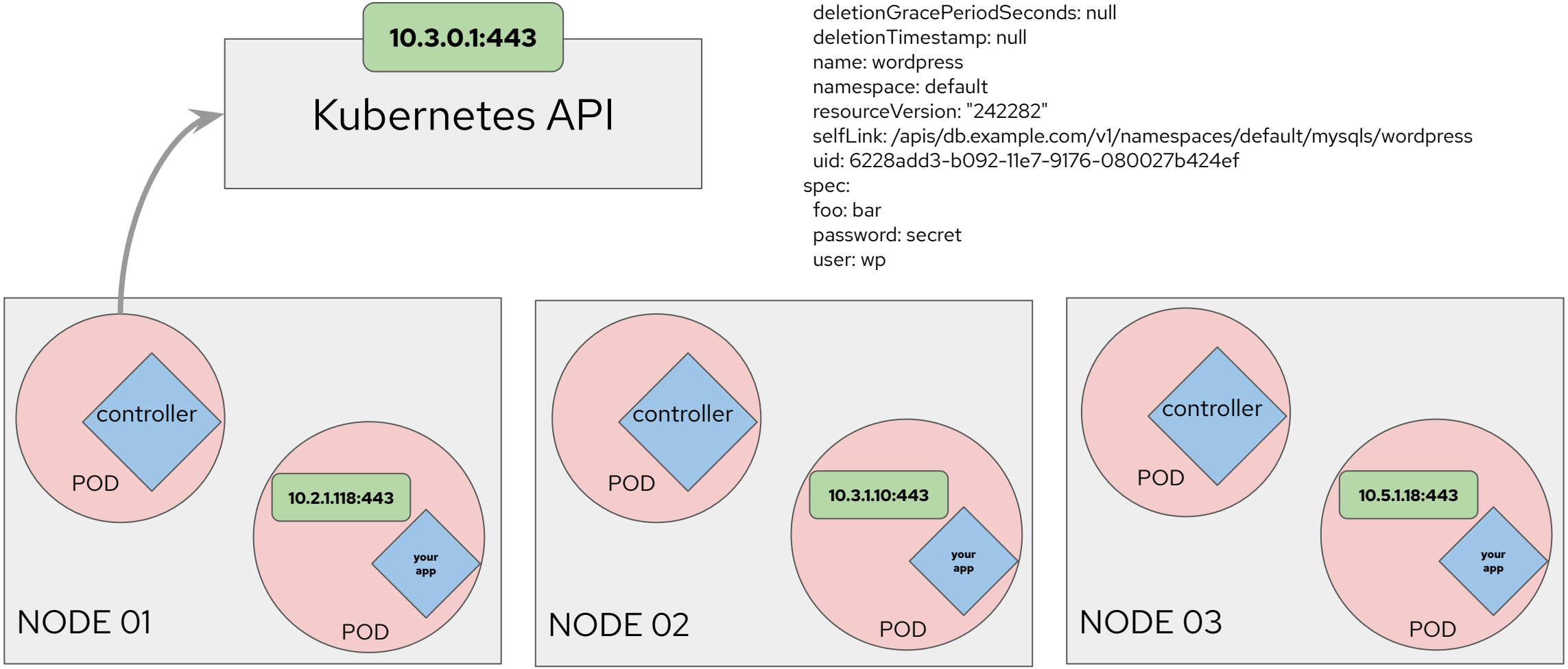
Compare current state to desired state.

Perform all the actions necessary to make current state meet desired state.




```
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
```






```
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
```

What do we mean by **ACT**?

- Create
- Read
- Update
- Delete

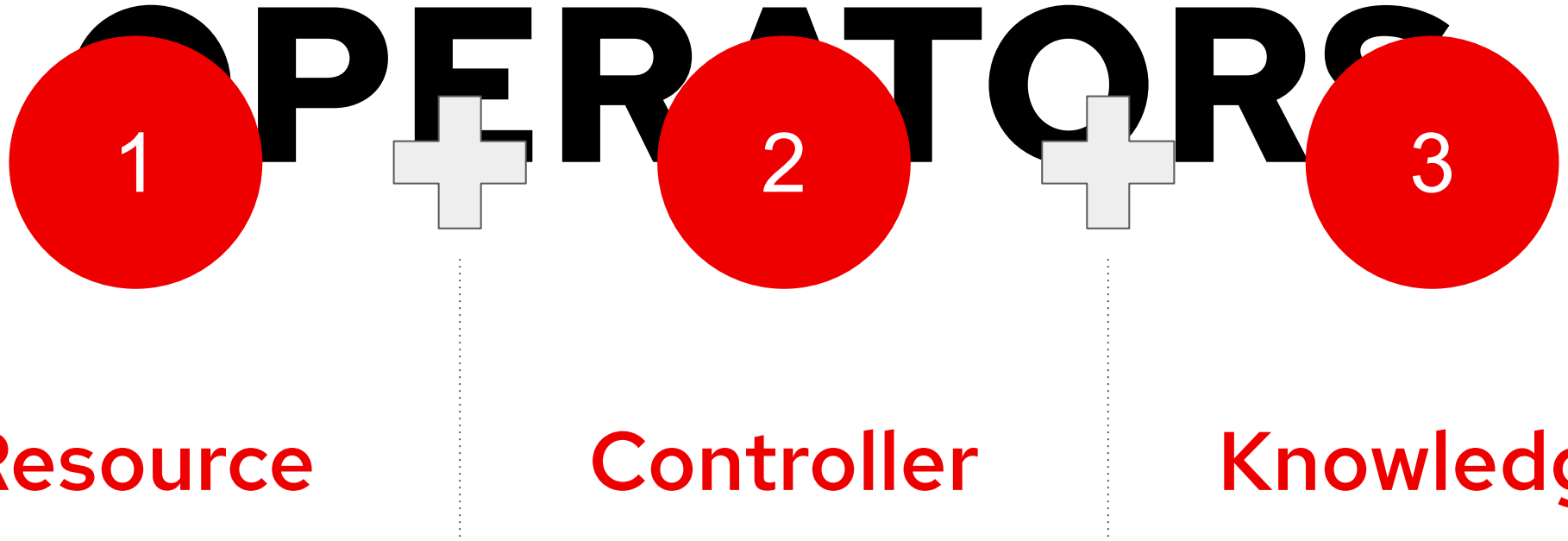
Create, Read, Update, Delete...Probably Not Enough

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 acccount 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

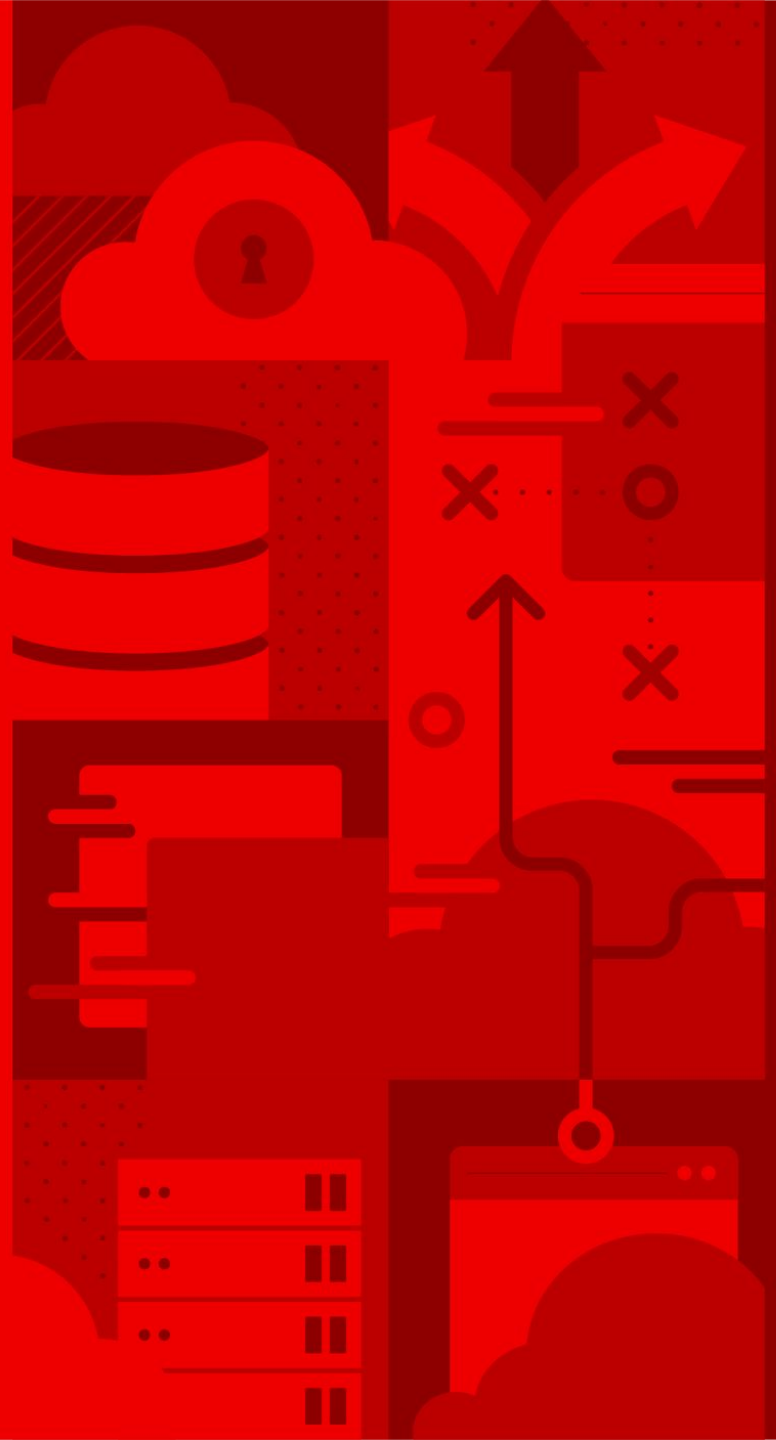
Securing a new MySQL installation
Running mysqld as an unprivileged user
Methods of starting the server
Invoking mysqld directly
Invoking safe_mysqld
Invoking mysql.server
Specifying startup options
Checking tables at startup
Shutting down the server
Regaining control of the server if you can't connect
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

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



Source:
https://coreos.com/blog/introducing-operators.html

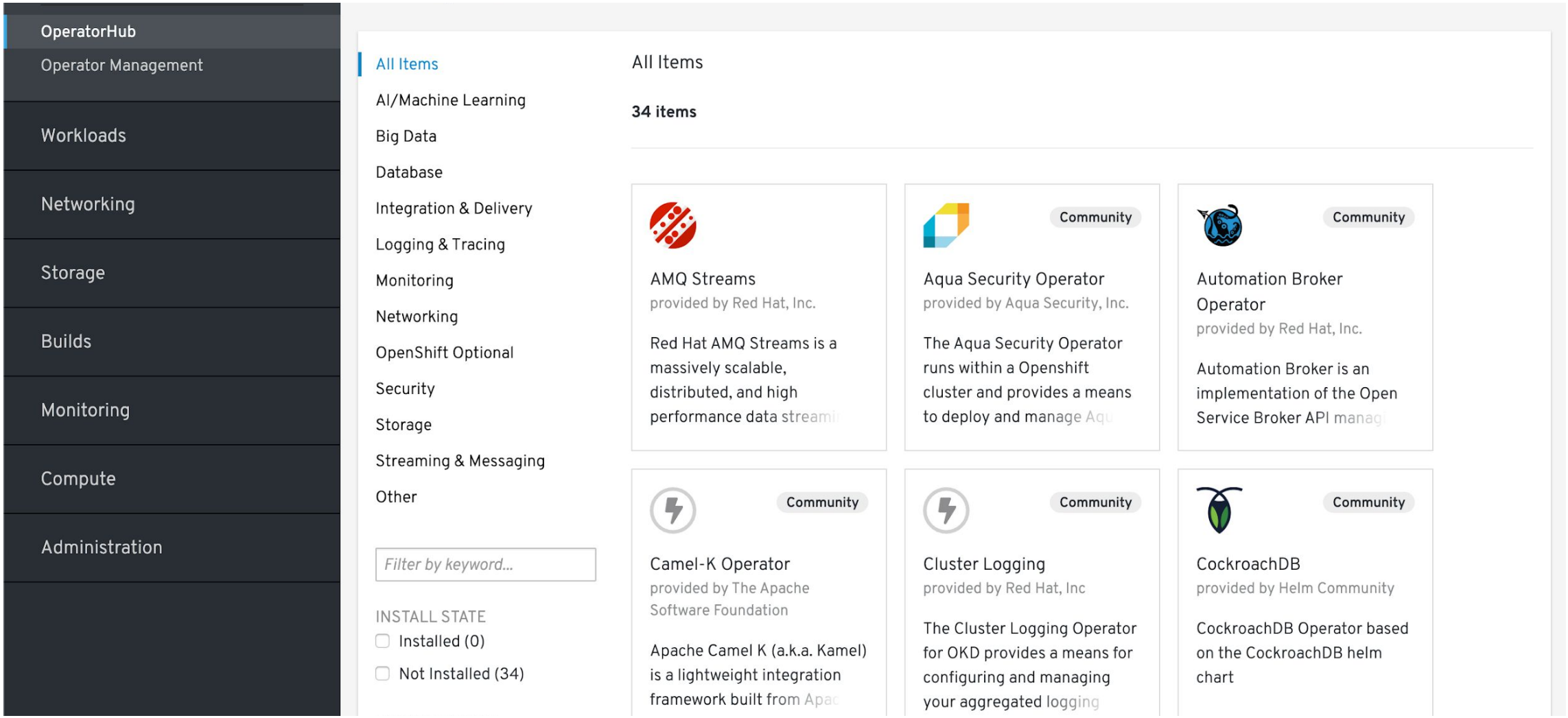


Why do Operators matter to us at Red Hat?

Red Hat

Why Operators Matter to Red Hat

- Build an ecosystem of software on OpenShift that can be as easy, safe, and reliable to use and operate as a Cloud Service.
- Low-touch, remotely managed, one-click-updates.
- Super easy to deploy in an Operator in a Kubernetes environment.


OperatorHub
Operator Management
Workloads
Networking
Storage
Builds
Monitoring
Compute
Administration
All Items
AI/Machine Learning
Big Data
Database
Integration & Delivery
Logging & Tracing
Monitoring
Networking
OpenShift Optional
Security
Storage
Streaming & Messaging
Other
Filter by keyword...
INSTALL STATE
Installed (0)
Not Installed (34)
All Items
34 items
AMQ Streams
provided by Red Hat, Inc.
Red Hat AMQ Streams is a massively scalable, distributed, and high performance data streami
Community
Aqua Security Operator
provided by Aqua Security, Inc.
The Aqua Security Operator runs within a Openshift cluster and provides a means to deploy and manage Aqu
Community
Automation Broker Operator
provided by Red Hat, Inc.
Automation Broker is an implementation of the Open Service Broker API manag
Community
Camel-K Operator
provided by The Apache Software Foundation
Apache Camel K (a.k.a. Kamel) is a lightweight integration framework built from Apac
Community
Cluster Logging
provided by Red Hat, Inc
The Cluster Logging Operator for OKD provides a means for configuring and managing your aggregated logging
Community
CockroachDB
provided by Helm Community
CockroachDB Operator based on the CockroachDB helm chart

Walkthrough: Deploy an Operator (the “hard” way)

(1) Deploy EtcdCluster CRD

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

apiVersion: apiextensions.k8s.io/v1
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
```

(2) Deploy EtcdCluster Operator

$ cat etcd-operator.yaml

```
apiVersion: apps/v1
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: acccounting
        - name: MY_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
```

(2) Deploy etcd Operator

```
$ oc create -f etcd-operator.yaml
$ oc get pods
```

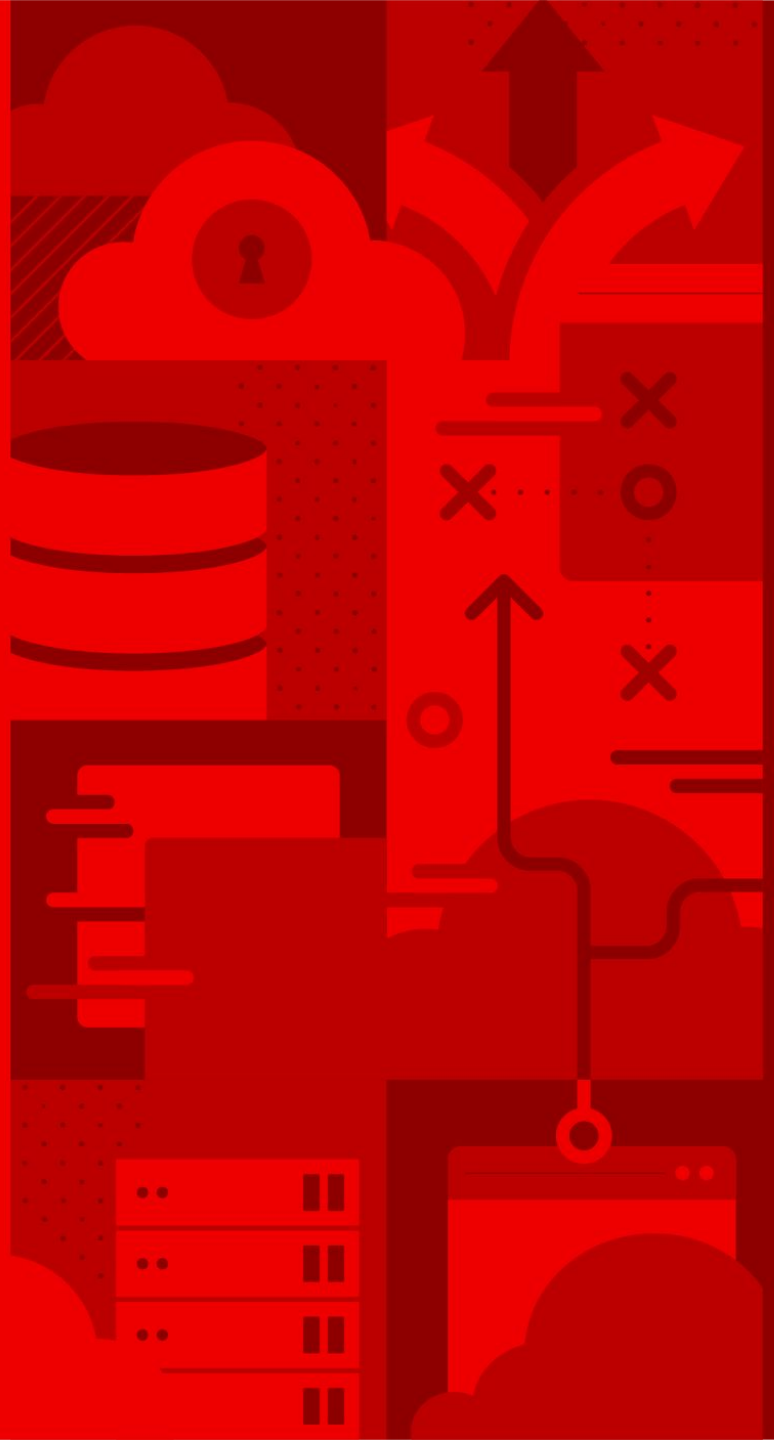
NAMESPACE	NAME	READY	STATUS	RESTARTS	AGE
accounting	etcd-operator-67666dc65f-xwfvq	1/1	Running	0	1s

(3) View 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"
```

(4) Deploy etcdCluster

```
$ oc create -f etcd-instance.yaml
$ oc get etcdcluster

NAMESPACE      NAME             AGE
default        myetcdcluster    1s
```



How do I create my very own Operator?

Red Hat

Life Before the Operator SDK

If only it were as simple as...

Resources

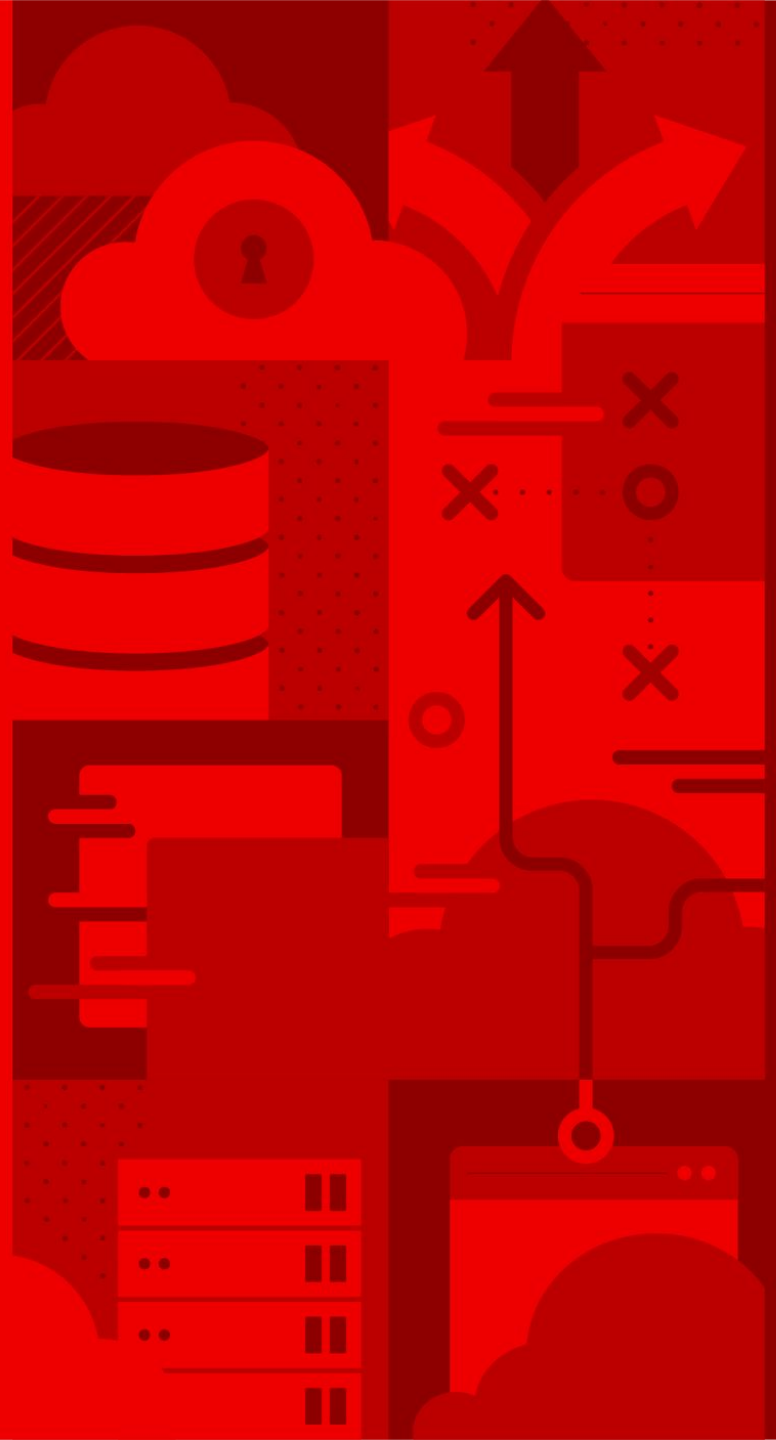
```
type MyCustomResourceDefinition struct {
   // API obj kind & schema version
   metav1.TypeMeta
   // Standard object metadata (optional)
   Metadata api.ObjectMeta
   // Describe how the resource appears
   Spec v1beta1.CustomResourceDefinitionSpec
   // State of the CRD
   Status CustomResourceDefinitionStatus
}
```

Controllers

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

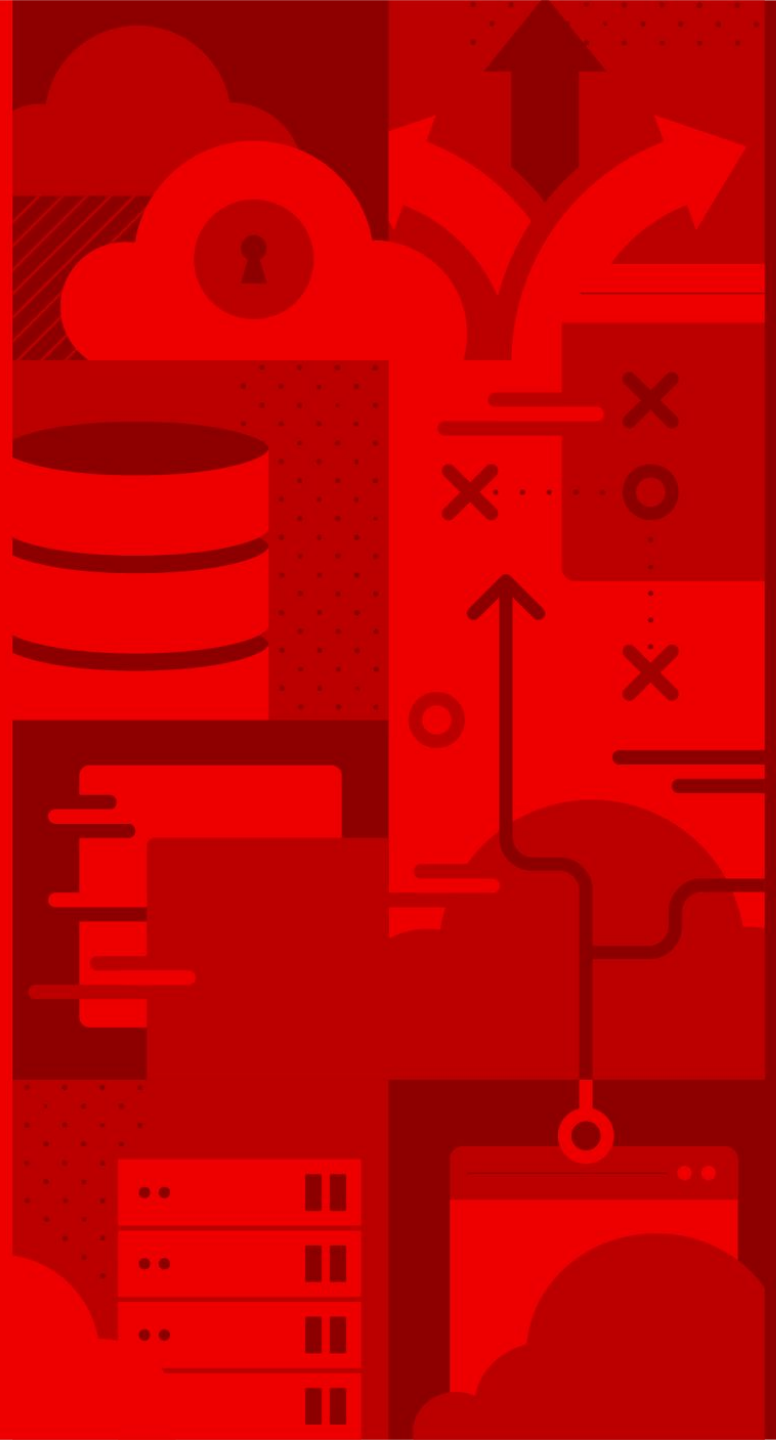
Writing Operator from scratch is Challenging

- Research client-library.
- Repo organization.
- Write boiler-plate code.
- Use code generators.
- Knowledge of informers/shared informers and work queues for object cache and event handling.



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

Red Hat



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

Red Hat

OPERATOR FRAMEWORK

Sources:
https://github.com/operator-framework

OPERATOR SDK

OPERATOR LIFECYCLE MANAGER

Sources:
https://github.com/operator-framework/operator-sdk
https://github.com/operator-framework/operator-lifecycle-manager
https://github.com/operator-framework/operator-metering

Operator SDK

WHAT IS OPERATOR SDK?

This project is a component of the Operator Framework, an open source toolkit to manage Kubernetes native applications, called Operators, in an effective, automated, and scalable way.

WHAT CAN I DO WITH OPERATOR SDK?

The Operator SDK provides the tools to build, test, and package Operators. Initially, the SDK facilitates the marriage of an application's business logic (for example, how to scale, upgrade, or backup) with the Kubernetes API to execute those operations. Over time, the SDK can allow engineers to make applications smarter and have the user experience of cloud services. Leading practices and code patterns that are shared across Operators are included in the SDK to help prevent reinventing the wheel.

The Operator SDK is a framework that uses the controller-runtime library to make writing operators easier by providing:

- High level APIs and abstractions to write the operational logic more intuitively
- Tools for scaffolding and code generation to bootstrap a new project fast
- Extensions to cover common Operator use cases

Operator SDK

DEVELOP IN GO, ANSIBLE, OR HELM

GO

1. Create a new operator project using the SDK Command Line Interface (CLI)
2. Define new resource APIs by adding Custom Resource Definitions (CRD)
3. Define Controllers to watch and reconcile resources
4. Write the reconciling logic for your Controller using the SDK and controller-runtime APIs
5. Use the SDK CLI to build and generate the operator deployment manifests

ANSIBLE

1. Create a new operator project using the SDK Command Line Interface (CLI)
2. Write the reconciling logic for your object using ansible playbooks and roles
3. Use the SDK CLI to build and generate the operator deployment manifests
4. Optionally add additional CRD's using the SDK CLI and repeat steps 2 and 3

HELM

1. Create a new operator project using the SDK Command Line Interface (CLI)
2. Create a new (or add your existing) Helm chart for use by the operator's reconciling logic
3. Use the SDK CLI to build and generate the operator deployment manifests
4. Optionally add additional CRD's using the SDK CLI and repeat steps 2 and 3

Operator SDK

Level I	Level II	Level III	Level IV	Level V
Basic Install	**Seamless Upgrades**	**Full Lifecycle**	**Deep Insights**	**Auto Pilot**
Automated application provisioning and configuration management	Patch and minor version upgrades supported	App lifecycle, storage lifecycle (backup, failure recovery)	Metrics, alerts, log processing and workload analysis	Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning

HELM

ANSIBLE

GO

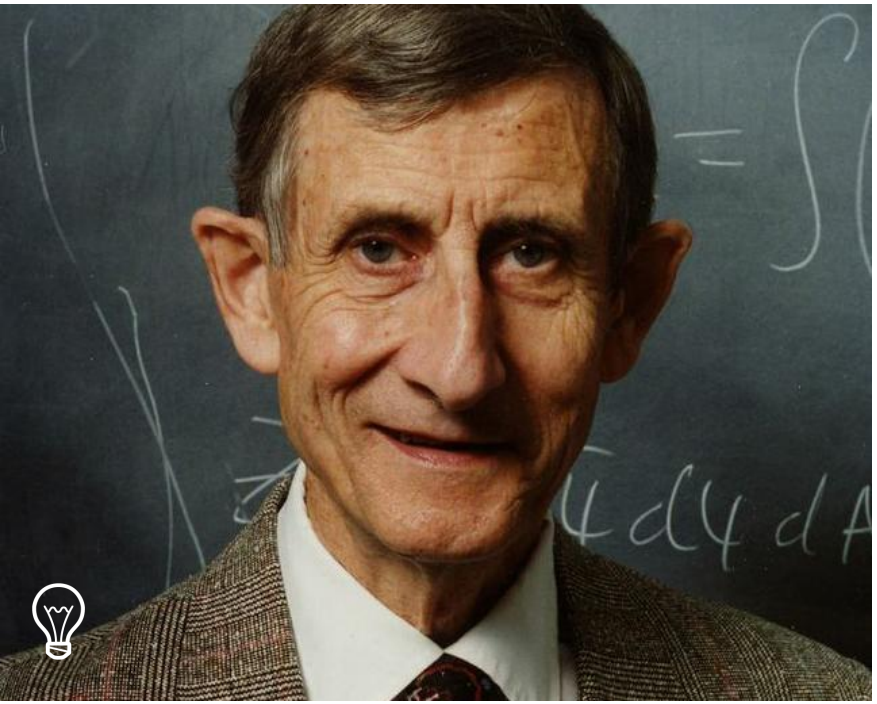
Operator Lifecycle Manager

WHAT IS OPERATOR LIFECYCLE MANAGER?

This project is a component of the Operator Framework, an open source toolkit to manage Kubernetes native applications, called Operators, in a streamlined and scalable way.

OLM FEATURES

OVER-THE-AIR UPDATES AND CATALOGS	DEPENDENCY MODEL	DISCOVERABILITY	CLUSTER STABILITY	DECLARATIVE UI CONTROLS
OLM provides rich update mechanisms to keep Kubernetes native applications up to date automatically.	With OLMs packaging format Operators can express dependencies on the platform and on other Operators.	OLM makes Operators and their services available for cluster users to select and install.	OLM will prevent conflicting Operators owning the same APIs being installed, ensuring cluster stability.	OLM enables Operators to behave like managed service providers through the APIs they expose.



"There is a great satisfaction in building good tools for other people to use."

Freeman John Dyson
Theoretical Physicist

Source:
https://en.wikipedia.org/wiki/Freeman_Dyson
http://ffden-2.phys.uaf.edu/webproj/212_spring_2015/Kai_Davids_Schell/phys212_website/origin.html
https://www.brainyquote.com/authors/freeman-dyson-quotes