

# Kubernetes Operators

In Action

Skokie Kubernetes Meetup User Group

Ken Lee

12/14/2020

# Who am I?

Ken Lee



@keunlee

Red Hat

# What we'll be discussing today

The Problem ...Which Birthed the Operator Pattern

What's an Operator?

How does an Operator Work?

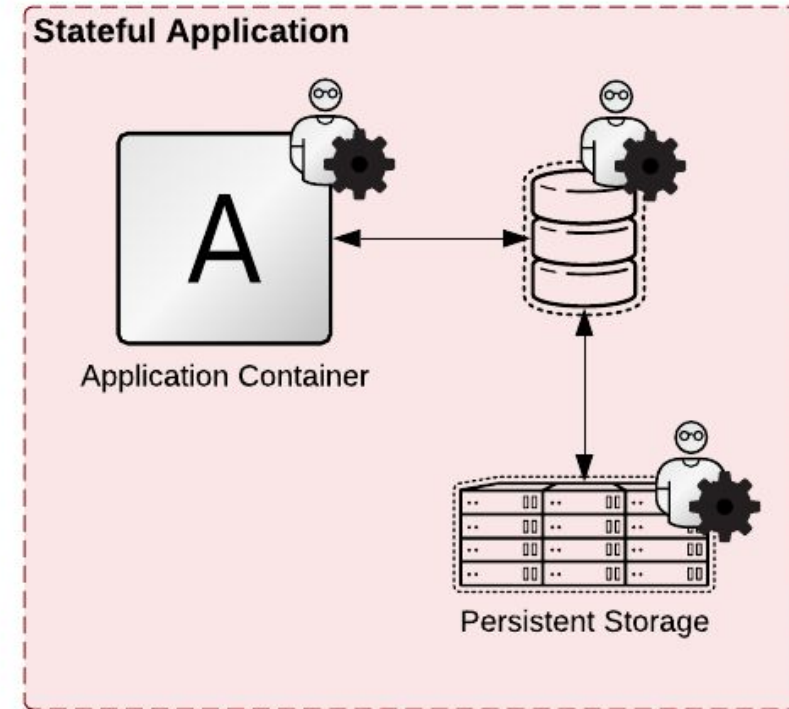
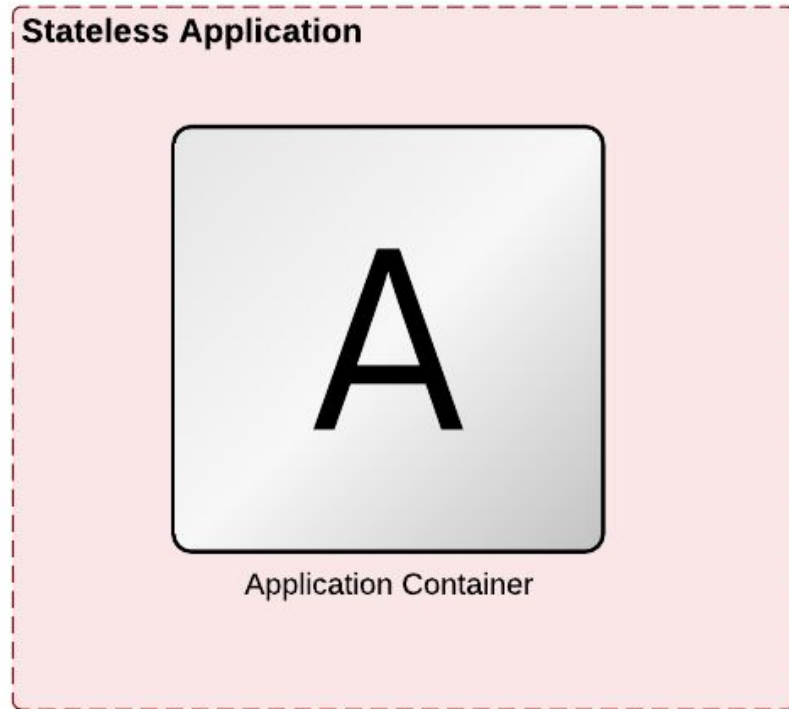
How do I make One?

Demo

---

# The Problem ... Which Birthed the Operator Pattern

# Stateless versus Stateful



How do you effectively automate  
Stateful applications on  
Kubernetes?

# OPERATORS



---

# What's an Operator?



# What's an Operator?

*"An operator is a Kubernetes controller that understands 2 domains: Kubernetes and something else. By combining knowledge of both domains, it can automate tasks that usually require a human operator that understands both domains."*

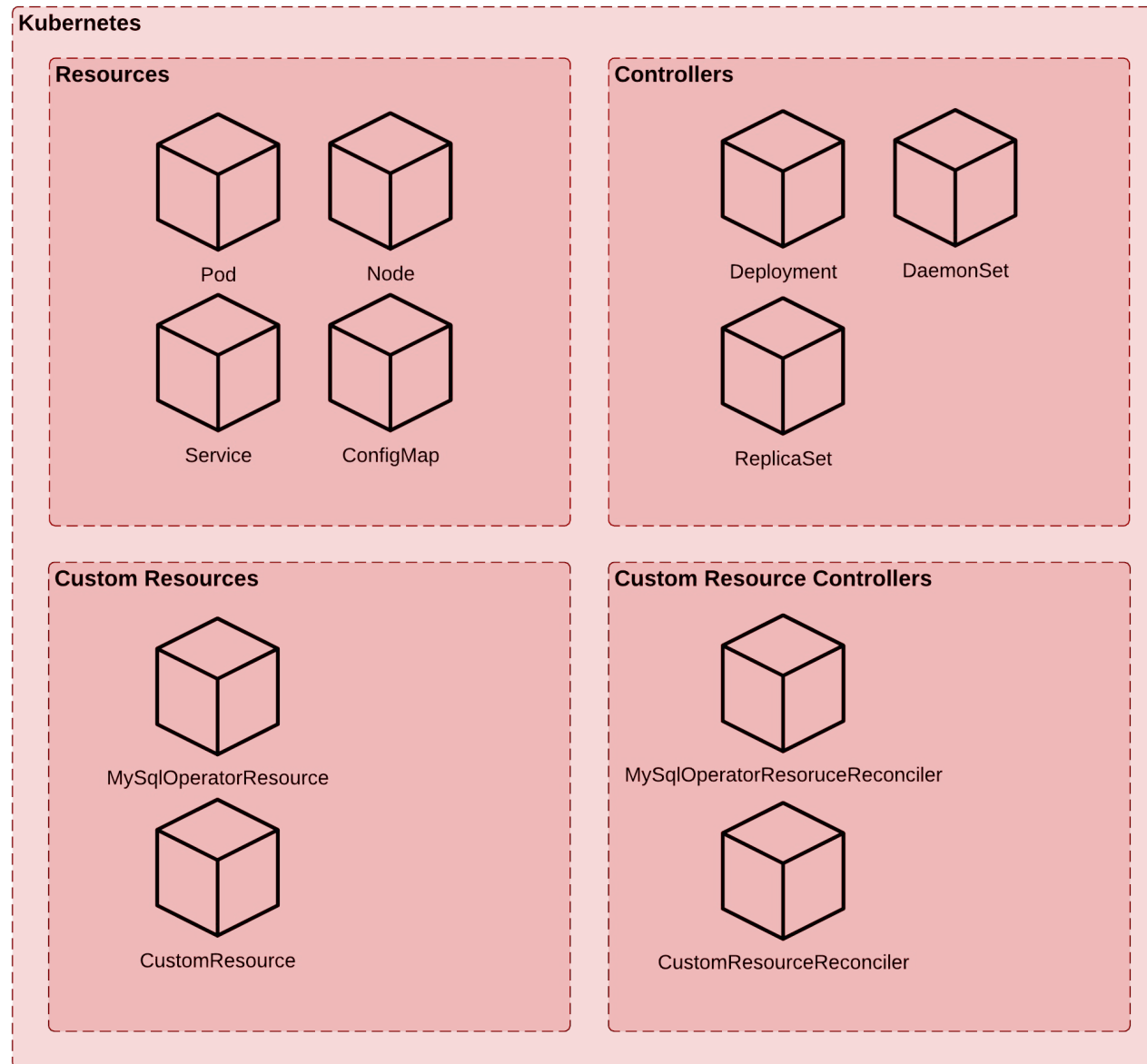
-Jimmy Zelinskie - Product and Engineering - CoreOS

<https://bit.ly/3iS6AFx>

# What's an Operator?

**Operator = Resource(s) + Controller(s) + Domain Specific Knowledge**

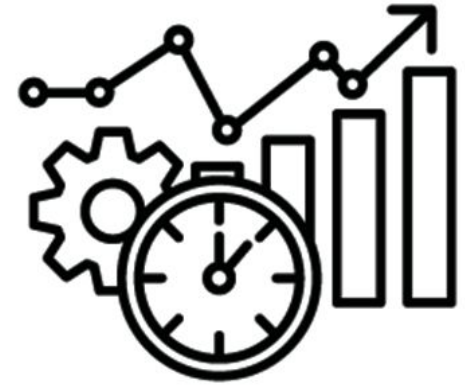
# Kubernetes: Resources + Controllers



# Domain Specific Knowledge/Operations

Examples of Domain Specific Knowledge/Operations (but not limited too)

- Fulfilling Configuration requirements
- Fulfilling Installation requirements
- Fulfilling Logging/Security requirements
- Fulfilling HA/Scaling requirements
- Application start-up and shutdown routines
- Process and workflow triggers
- Etc.



# Example: Conventional vs Operator based Deployments

## Conventional Deployment

**Deployment Artifacts:** separately managed artifacts that make up a deployment

```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx
  labels:
    run: my-nginx
spec:
  type: NodePort
  ports:
    - port: 8080
      targetPort: 80
      protocol: TCP
      name: http
    - port: 443
      protocol: TCP
      name: https
  selector:
    run: my-nginx
```

**Service**

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
spec:
  selector:
    matchLabels:
      run: my-nginx
  replicas: 3
  template:
    metadata:
      labels:
        run: my-nginx
    spec:
      volumes:
        - name: configmap-volume
          configMap:
            name: my-config
      containers:
        - name: webapp
          image: nginx
          ports:
            - containerPort: 443
            - containerPort: 80
          volumeMounts:
            - mountPath: /etc/nginx/conf.d
              name: configmap-volume
```

**Deployment**

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
  namespace: default
data:
  app.properties: |
    env=dev
    timeout=30
    setting.value=1
```

**Config Map**

versus

## Kubernetes Application

**Configmap**  
configurations

**Deployment:** deployment-definition

**Pod**  
[deployment template defined]

**Pod**  
[deployment template defined]

**Pod**  
[deployment template defined]

**Service**  
service ports

## Operator-based Deployment

**Deployment Artifacts:** A single operator instance - given a CRD and Controller Implementation.

### Operator

```
apiVersion: operator.local/v1alpha1
kind: EncapsulatedApplicationOperator
metadata:
  name: my-app
spec:
  size: 3
```

**Custom Resource Definition**

### Custom Resource Controller

#### Domain Specific Knowledge/Operations

Codified Business Logic which executes operations normally handled externally (i.e. human intervention, 3rd parties, etc.)

Post-Operation(s)

Preparation(s)

Configuration

Execute Business Rules

Other Domain Specific Operations

Deploy K8S Resources

#### Reconciliation Logic

Codified Logic for reconciling an operator's desired state to current state.

yields

yields

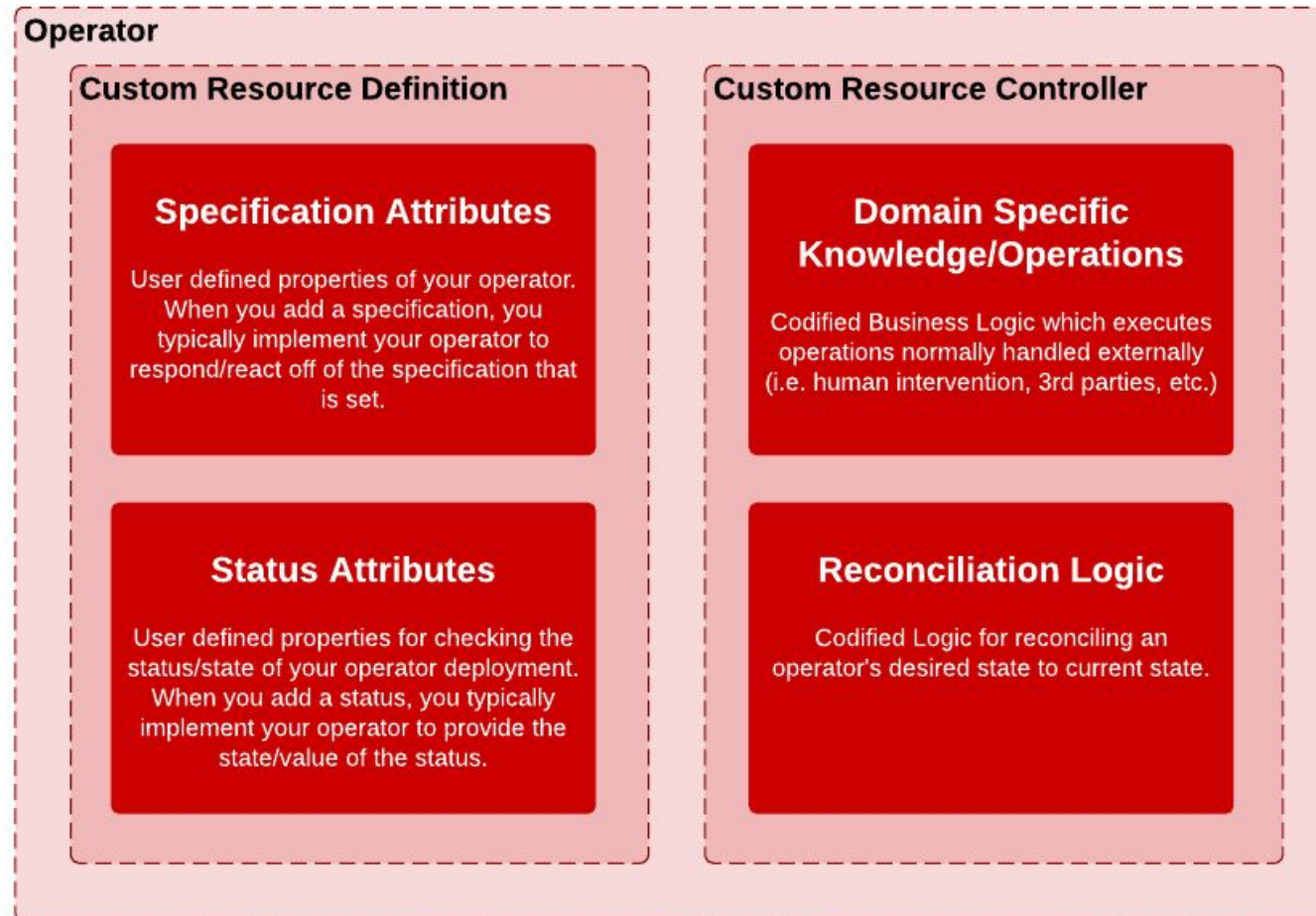
\*\*\* Operational specific tasks are carried out manually or potentially automated through other means.

---

# How does an Operator work?

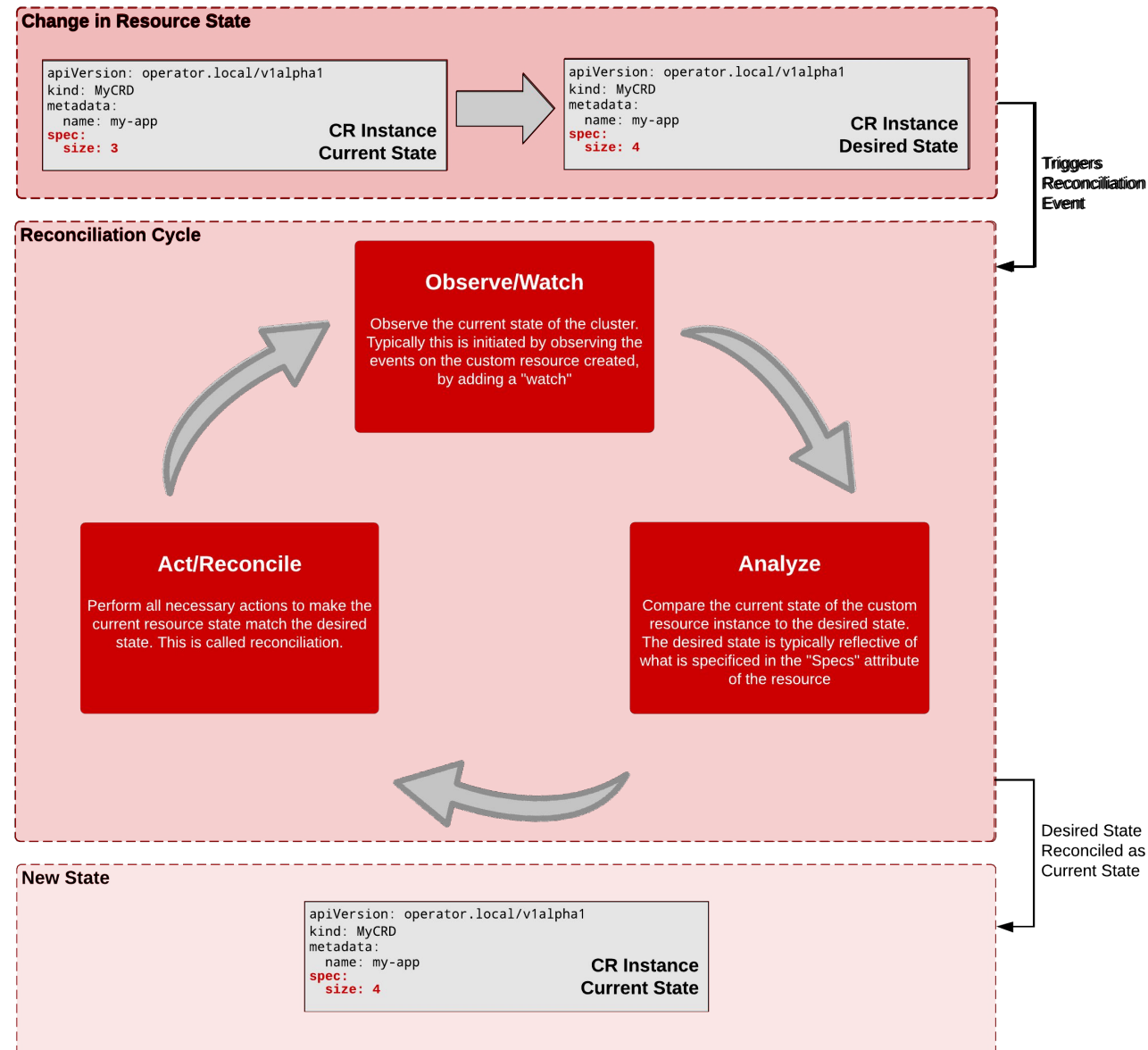
# Operator Components

**Operator = Resource(s) + Controller(s) + Domain Specific Knowledge**





# Custom Resource Controller - Reconciliation Cycle - Example





How do I make one?

---

# How do I make one?

# Resources - Operator Frameworks and Libraries

- <https://sdk.operatorframework.io/build/>

- Golang
- Ansible
- Helm



- <https://book.kubebuilder.io/>

- Golang



- <https://kudo.dev/>

- Yaml



- <https://github.com/metacontroller/metacontroller>

- Jsonnet

- <https://github.com/zalando-incubator/kopf>

- Python

- <https://github.com/ContainerSolutions/java-operator-sdk>

- Java

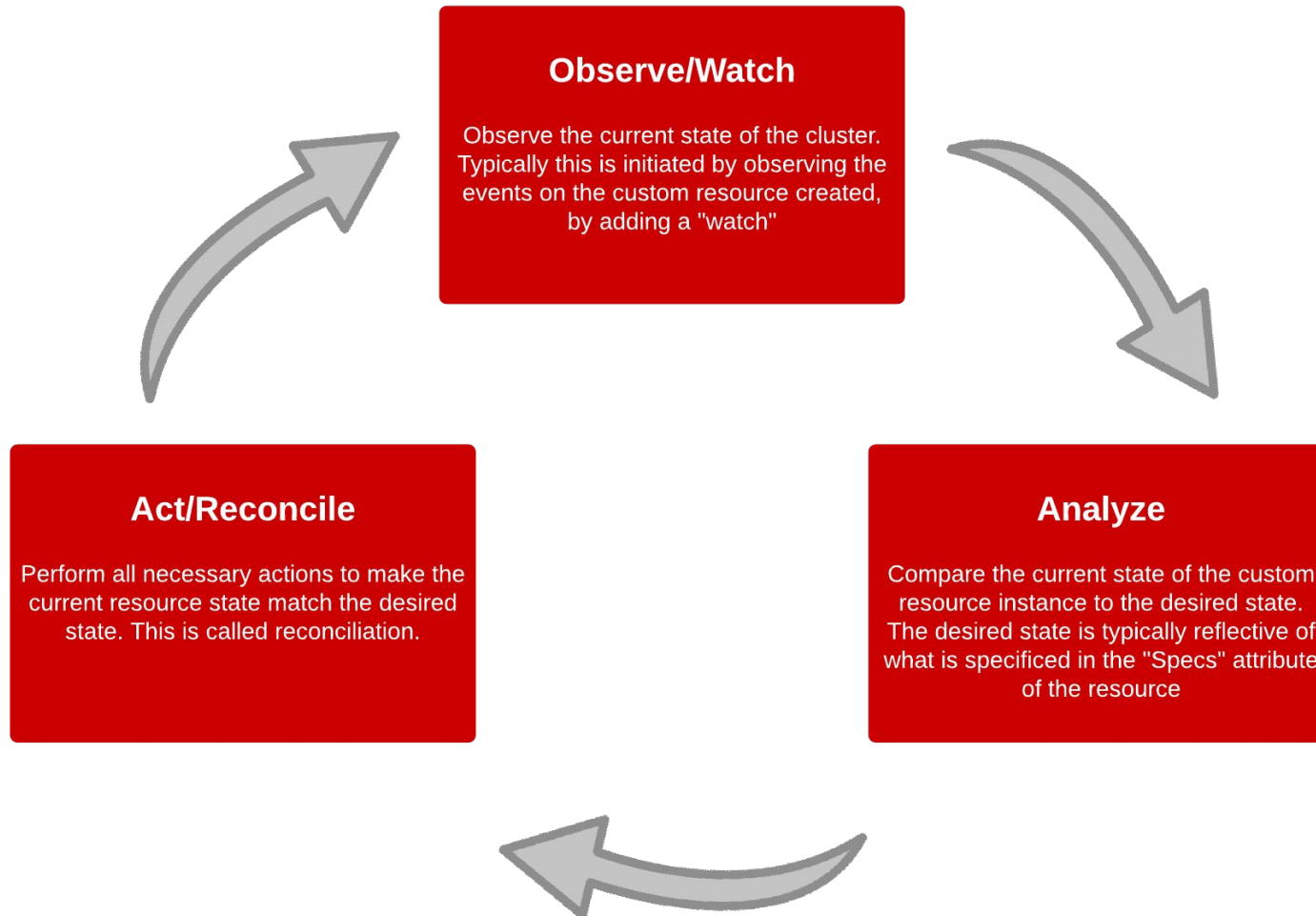
- <https://github.com/dot-i/k8s-operator-node>

- Typescript/NodeJS

- <https://github.com/TremoloSecurity/kubernetes-javascript-operator>

- Javascript

# Know the Reconciliation Cycle



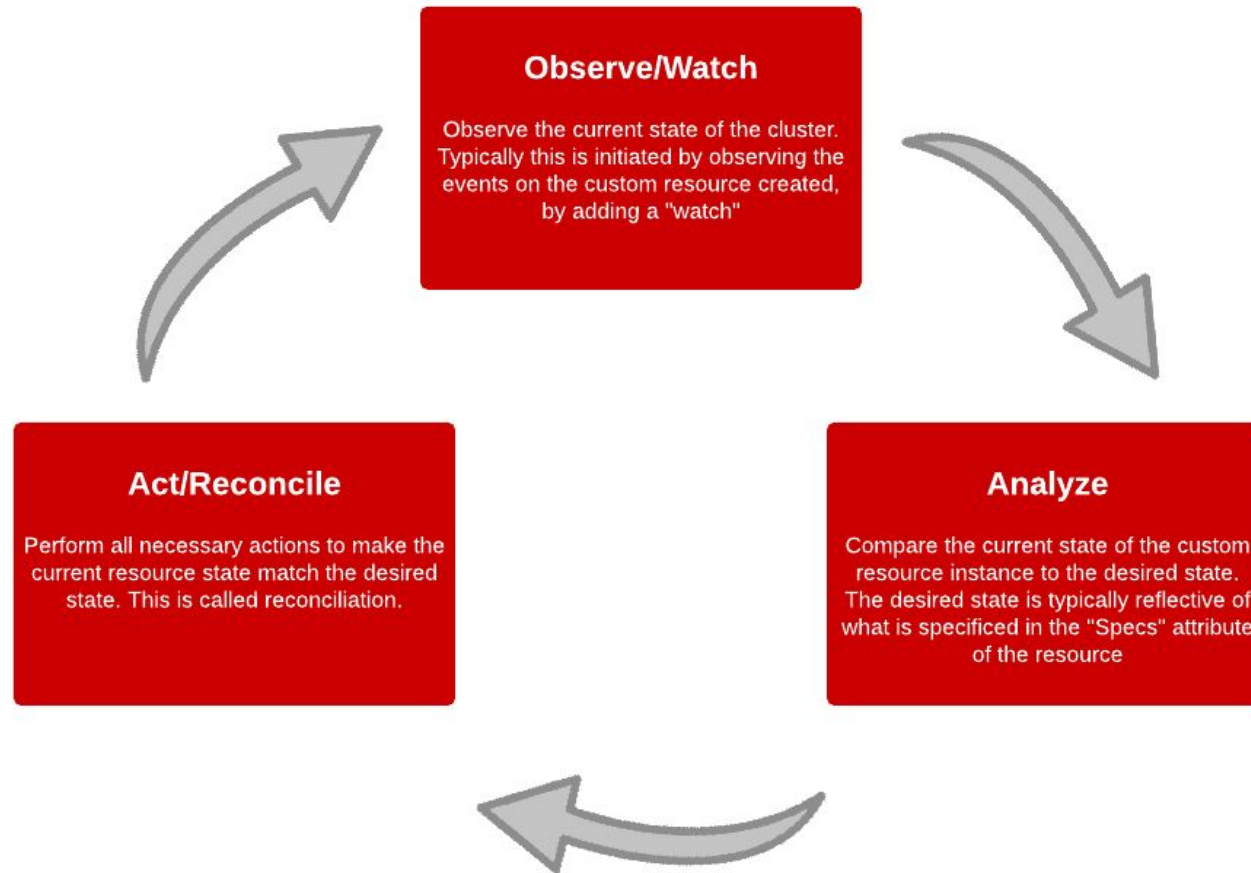
## Resources - Learn more about Operators

- <https://kubernetes.io/docs/concepts/extend-kubernetes/operator/>
- <https://enterpriseproject.com/article/2019/2/kubernetes-operators-plain-english>
- <https://coreos.com/blog/introducing-operators.html>
- <https://www.openshift.com/blog/operator-framework-moves-to-cncf-for-incubation>
- <https://www.openshift.com/blog/kubernetes-operators-best-practices>
- [https://www.youtube.com/watch?v=8\\_DaCcRMp5I&t=3453s](https://www.youtube.com/watch?v=8_DaCcRMp5I&t=3453s)
- <https://github.com/k8s-operators-over-ez/k8s-operators-over-ez.labs>

---

# Demo

# Example Operator



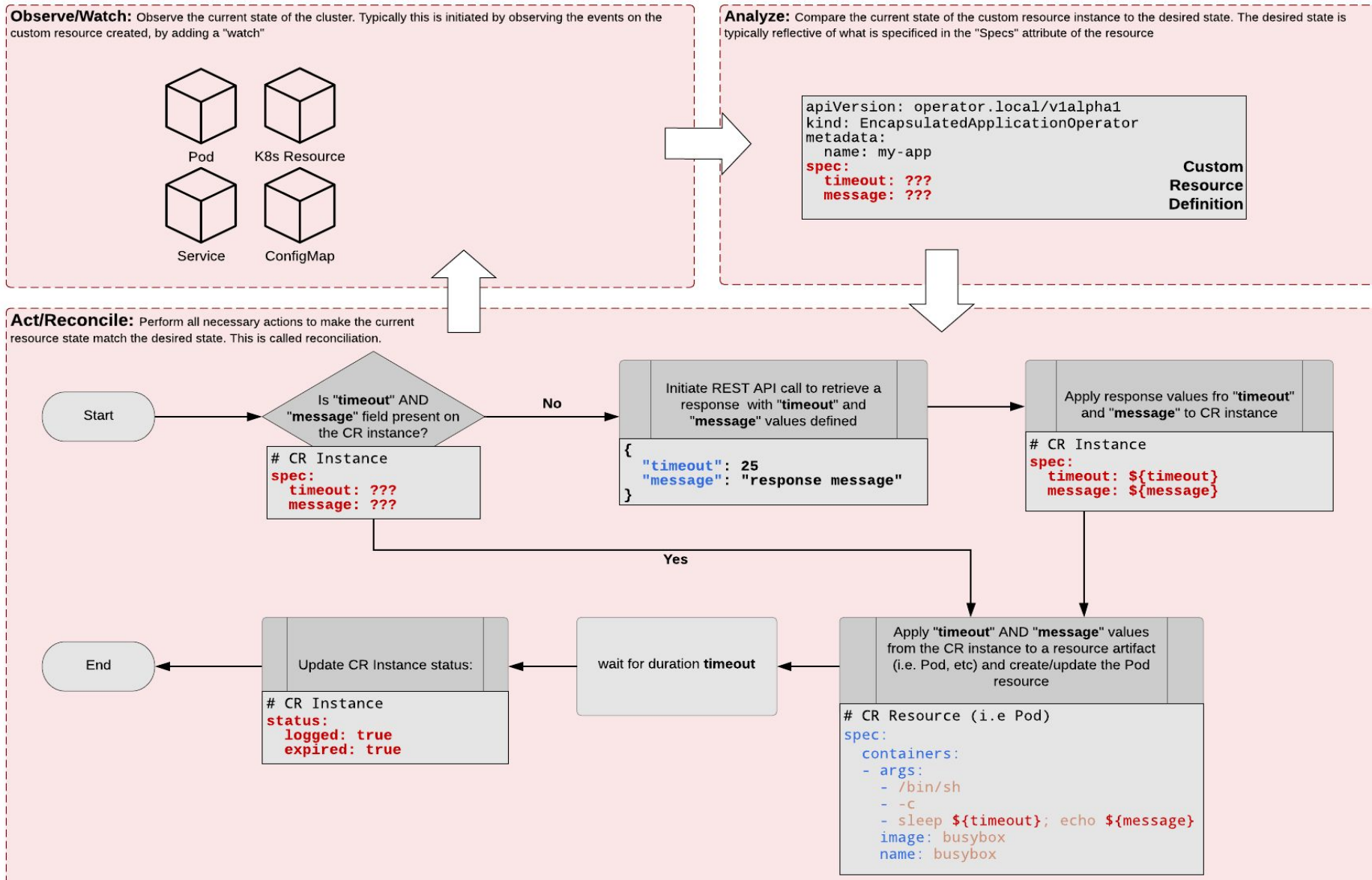
```
apiVersion:
operators-over-ez.mydomain.com/v1alpha1
kind: OpsOverEasy
metadata:
  name: opsovereasy-sample
spec:
  # Add fields here
  timeout: 15
  message: "my message"
```

**Custom Resource (CR) Instance**

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    run: busybox
    name: busybox
spec:
  containers:
    - args:
      - /bin/sh
      - -c
      - sleep ${timeout}; echo ${message}
      image: busybox
      name: busybox
      dnsPolicy: ClusterFirst
      restartPolicy: Never
status: {}
```

**CR Artifact (i.e. Pod)**

# Example Operator



---

# Thank you



---

# Appendix

# Additional Points of Interest

- What about Statefulsets? Can't I use those for managing and persisting State?
  - Short answer, YES
  - Think about what a Statefulset is. It's a resource controller too. The controller will manage the state of your pods with the use of persistent storage and a headless service.
  - Operators, offer a way for **you** to manage the state of your application, through **your** code.
- What about Helm Charts? When would you use a Chart vs an Operator?
  - We can try to use this as a general rule of thumb. If you need to codify operational knowledge of your K8S application as well as maintain state, then leveraging the Operator Pattern to facilitate the development of your K8S application, will serve you well.
  - However, if that's not the case, or the Operator pattern is just not your thing, you're not out of luck. You can still leverage constructs like Statefulsets to help you maintain state in your Kubernetes application, yet alone package a Statefulset configuration as part of your Helm Chart. The thing you have to keep in mind is how you manage and automate Domain Specific tasks and operations.