Kubernetes Operators

State Seeking Infrastructure As Code

Carson Anderson

Domo

@carsonoid

@carson_ops

Obligatory "What is an Operator" Slide

- An "Operator" is any bit of code that watches a Kubernetes system and seeks a desired state
- They are often used to manage databases or other software in Kubernetes
- They represent the "management" part of a managed service which runs inside of Kubernetes
- They take all the operational knowledge for the managed Software and put it into code

But "What is an Operator" Really?

A management pattern

Why Operators?

A "Practical" Example

Wasteful
Individual
GIF
Mirror

WIGM: Goals

- Basic web server
- One GIF hosted per web server
- Easy to host many instances
- (Bonus) No dedicated images

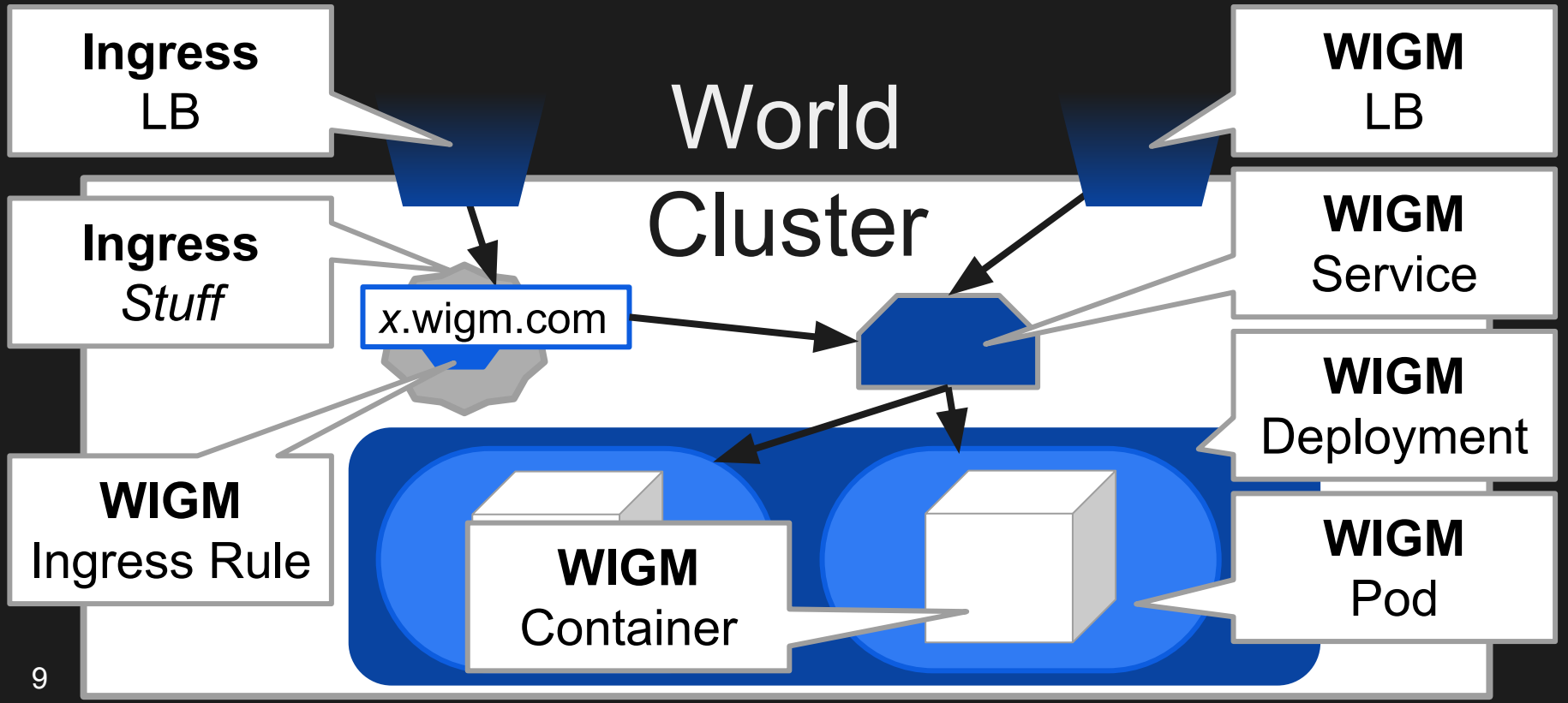
WIGM: Inputs

- ENV variables:
  - GIF_NAME
  - GIF_SOURCE_LINK
- Operational decisions
  - Create cloud load balancer?
  - Create ingress rule?

WIGM: Outputs

- Kubernetes resources
  - Deployment
  - Service
  - Cloud load balancer (Optional)
  - Ingress rule (Optional)

WIGM: Outputs Explained



WIGM Server

Pure Upstream NGINX

Bash Bootstrap

Base Image: nginx:1.5-alpine

```
## Expects: GIF_SOURCE_LINK, GIF_NAME
apk update && apk add curl

# go to data dir
cd /usr/share/nginx/html

# fetch original
curl -Lo wigm.gif "$GIF_SOURCE_LINK"

# write page using heredoc
cat > index.html <<EOF
<head>
  <title>WIGM: $GIF_NAME</title>
</head>
<body>
  <h1>WIGM: $GIF_NAME</h1>
  <img src="wigm.gif" />
</body>
EOF

# start nginx
exec nginx -g "daemon off;"
```

Let's Deploy WIGMs

1. Via YAML
2. Via Helm
3. Via operators
    - Metacontroller (Webhooks)
    - Operator SDK (Golang)

WIGM Via YAML

YAML File Structure

```
./wigm/yaml/
├── resources.yaml
└── releases
    ├── yamlrelease.yaml
    └── holdem.yaml
```

YAML

resources.yaml Deployment Section

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wigm-host-NAME
  labels:
    app: wigm
    gif: NAME
spec:
  replicas: 2
  selector:
    matchLabels:
      app: wigm
      gif: NAME
  template:
    metadata:
      labels:
        app: wigm
        gif: NAME
    spec:
      [...]
```

YAML

resources.yaml
Deployment
Section
(Continued)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  [...]
spec:
  [...]
  template:
    [...]
    spec:
      containers:
      - name: gifhost
        image: nginx:1.15-alpine
        ports:
        - containerPort: 80
        env:
        - name: GIF_NAME
          value: NAME
        - name: GIF_SOURCE_LINK
          value: LINK
        command:
          [...]
```

YAML

resources.yaml Deployment Section (Continued)

```
[...]
    spec:
      containers:
      - name: gifhost
      [...]
        command:
        - sh
        - -exc
        - |

          # go to data dir
          [...]

          # fetch original
          [...]

          # write page using heredoc
          [...]

          # start nginx
          exec nginx -g "daemon off;"
```

YAML

resources.yaml
Service Section

```
---
kind: Service
apiVersion: v1
metadata:
  name: wigm-host-NAME
  labels:
    app: wigm
    gif: NAME
spec:
  selector:
    app: wigm
    gif: NAME
  # Uncomment line below to request
  # a cloud LB for the service
  # type: LoadBalancer
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

YAML

resources.yaml
Ingress Section

```
# Delete or comment to disable ingress
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: wigm-host-NAME
  labels:
    app: wigm
    gif: NAME
spec:
  rules:
  - host: NAME.wigm.carson-anderson.com
    http:
      paths:
      - path: /
        backend:
          serviceName: wigm-host-NAME
          servicePort: 80
```

YAML Release 1

Name	yamlrelease	Load Balancer?	DEF	Ingress?	DEF
Link	https://media.giphy.com/media/JloRBVCgTP6RW/giphy.gif				

YAML Release 1 Process

- Copy
- Paste
- Find/Replace
  - NAME -> yamlrelease
  - LINK -> https://…
- "kubectl create -f …"

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wigm-host-yamlrelease
  labels:
    app: wigm
    gif: yamlrelease
spec:
  [...]
    spec:
     [...]
       env:
        - name: GIF_NAME
          value: yamlrelease
        - name: GIF_SOURCE_LINK
          value: https://media.giphy.com/media/JIoRBVCgTP6RW/giphy.gif
```

YAML Release 2

Name	holdem	Load Balancer?	YES	Ingress?	NO
Link	https://media.giphy.com/media/l2JehUhBRqoJXC6t2/giphy.gif				

YAML Release 2 Process

- Copy
- Paste
- Find/Replace
  - NAME -> holdem
  - LINK -> https://…
- Uncomment LB Line
- Delete Ingress YAML
- "kubectl create -f …"

```
---
kind: Service
apiVersion: v1
metadata:
  name: wigm-host-holdem
  labels:
    app: wigm
    gif: holdem
spec:
  selector:
    app: wigm
    gif: NAME
  # Uncomment line below to request
  # a cloud LB for the service
  type: LoadBalancer
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

YAML Update Process

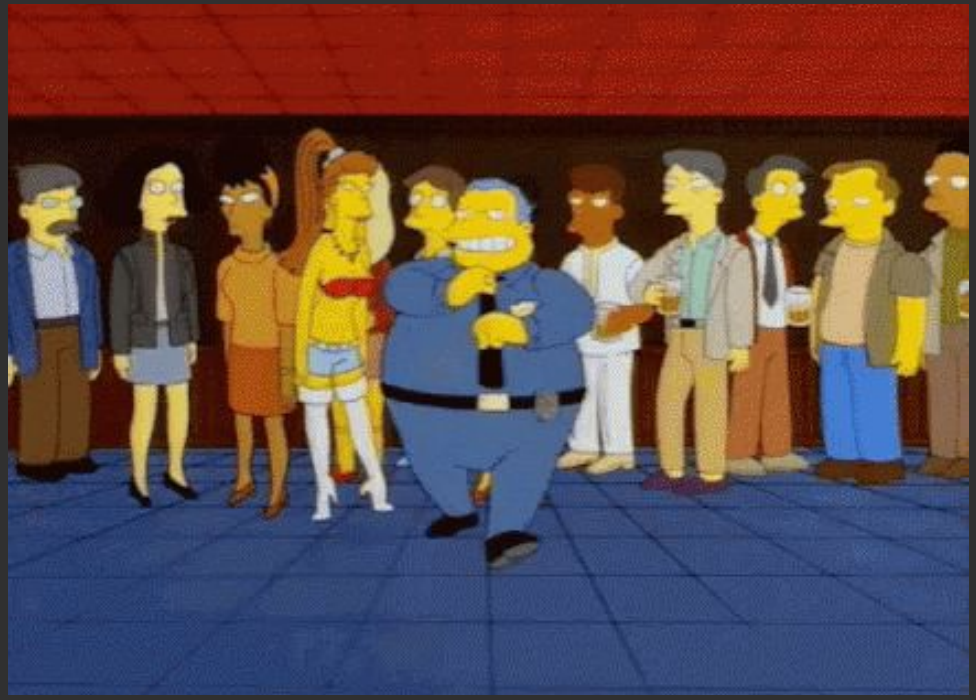
- Edit
- Apply
- Hope

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wigm-host-yamlrelease
  labels:
    app: wigm
    gif: yamlrelease
spec:
  [...]
    spec:
     [...]
       env:
        - name: GIF_NAME
          value: yamlrelease
        - name: GIF_SOURCE_LINK
          value: https://media.giphy.com/media/JloRBVCgTP6RW/giphy.gif
```

DEMO

YAML In Action

WIGM Via Helm Templates



Helm File Structure

```
./wigm/helm/
├── Chart.yaml
├── values.yaml
├── templates
│   ├── deployment.yaml
│   ├── ingress.yaml
│   └── service.yaml
└── releases
    ├── escape-values.yaml
    └── helmrelease-values.yaml
```

Helm

values.yaml

```
# Default values for wigm.
# This is a YAML-formatted file.
# Declare variables to be passed into
your templates.

gif:
  name: "" # Defaults to helm stack name
  link: "" # Required!

service:
  create_cloud_lb: false

ingress:
  enabled: true
```

Helm

templates/
deployment.yaml

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: wigm-host-{{ .Release.Name }}
  labels:
    app: wigm
    gif: {{ .Release.Name }}
spec:
    [...]
    spec:
      containers:
      - name: gifhost
        [...]
        env:
       - name: GIF_NAME
          {{- if .Values.gif.name }}
          value: {{ .Values.gif.name | quote  }}
          {{- else }}
          value: {{ .Release.Name }}
          {{- end }}
        - name: GIF_SOURCE_LINK
          value: {{ .Values.gif.link }}
```

Helm

templates/ service.yaml

```
---
kind: Service
apiVersion: v1
metadata:
  name: wigm-host-{{ .Release.Name }}
  labels:
    app: wigm
    gif: {{ .Release.Name }}
spec:
  selector:
    app: wigm
    gif: {{ .Release.Name }}
  {{ if .Values.service.create_cloud_lb }}
  type: LoadBalancer
  {{ end }}
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

Helm

templates/ ingress.yaml

```
{{ if .Values.ingress.enabled }}
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: wigm-host-{{ .Release.Name }}
  labels:
    app: wigm
    gif: {{ .Release.Name }}
spec:
  rules:
  - host: {{ .Release.Name }}.wigm.carson-anderson.com
    http:
      paths:
      - path: /
        backend:
          serviceName: wigm-host-{{ .Release.Name }}
          servicePort: 80
{{ end }}
```

Helm Release 1

Name	helmrelease	Load Balancer?	DEF	Ingress?	DEF
Link	https://media.giphy.com/media/wa6hNG157vUA25ncAu/giphy.gif				

Helm New Release Process

- Create Values File

```
cat > releases/helmrelease.yaml <<EOF
gif:
  link: https://media.giphy.com/media/wa6hNG157vUA25ncAu/giphy.gif
EOF
```

- Run Helm Install

```
helm install \
  -n helmrelease \
  -f releases/helmrelease.yaml .
```

Helm Release 2

Name	Escape!	Load Balancer?	YES	Ingress?	NO
Link	https://media.giphy.com/media/3o6Mb8Py77B1Bqrrb2/giphy.gif				

Helm Release 2 Process

- Create Values File

```
cat > releases/escape.yaml <<EOF
gif:
  name: "Escape!"
  link: "https://media.giphy.com/media/3o6Mb8Py77B1Bqrrb2/giphy.gif"

service:
  create_cloud_lb: true

ingress:
  enabled: false
EOF
```

- Run Helm Install

```
helm install \
  -f releases/escape.yaml .
```

Helm Update Process

- Edit Values File

```
cat > releases/helmrelease.yaml <<EOF
gif:
  name: "State Updates!"
  link: "https://media.giphy.com/media/wa6hNG157vUA25ncAu/giphy.gif"
EOF
```

- Run Helm Upgrade

```
helm upgrade helmrelease \
  -f releases/helmrelease.yaml .
```

Helm Release 3

Name		Load Balancer?	DEF	Ingress?	DEF
Link	https://media.giphy.com/media/RqzMtDuGECzL2/giphy.gif				

Helm Release 3 Process

- Run Helm Install

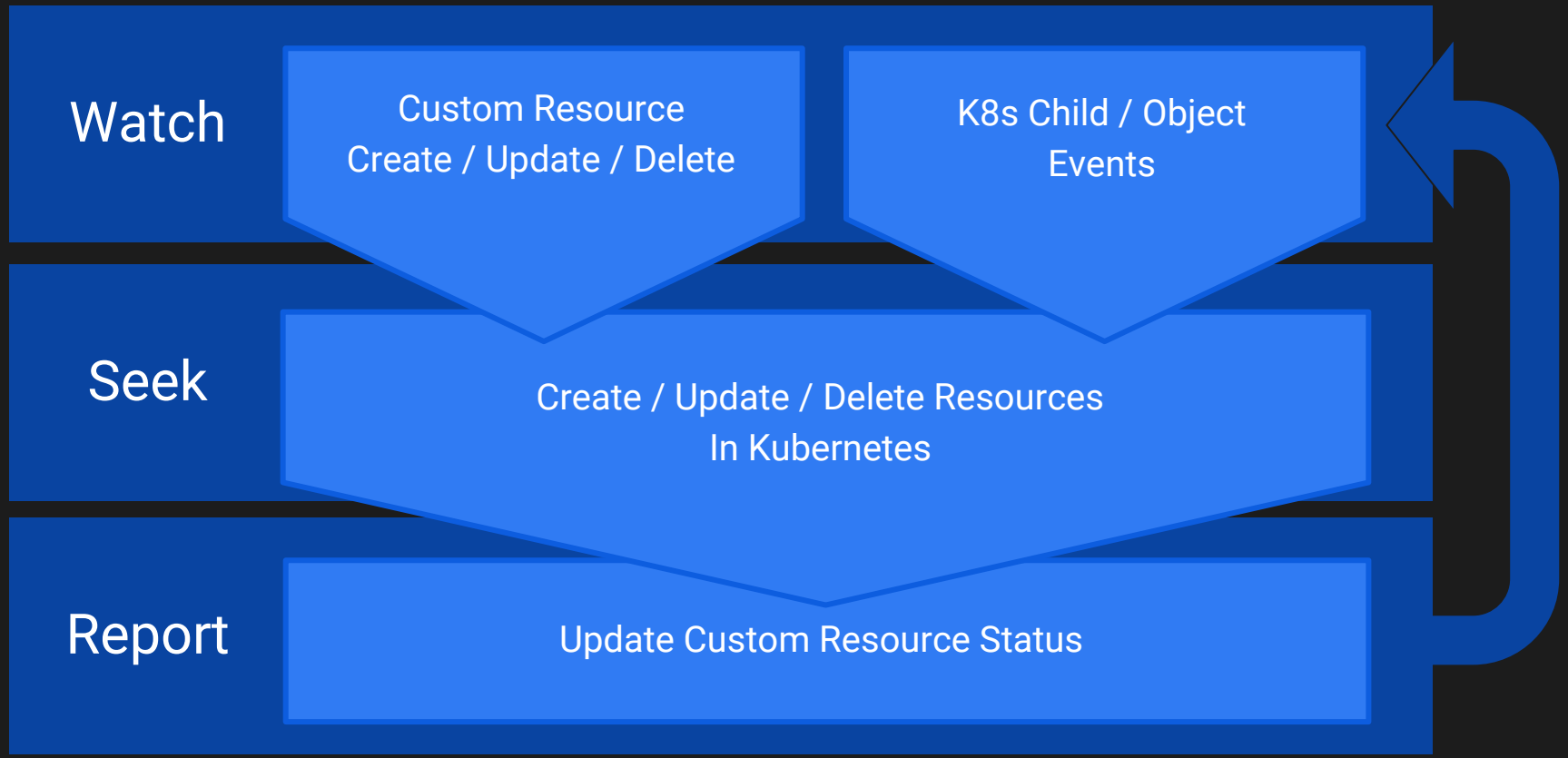
```
helm install \
  --set \
   gif.url=https://media.giphy.com/media/RqzMtDuGECzL2/giphy.gif  \
  .
```

DEMO

Helm In Action

WIGM Via Operators

Operator Principles



Operators

Custom Resource Definition

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: wigmgifs.wigm.carson-anderson.com
spec:
  group: wigm.carson-anderson.com
  names:
    kind: WigmGif
    listKind: WigmGifList
    plural: wigmgifs
    singular: wigmgif
  scope: Namespaced
  version: v1
  subresources:
    status: {}
```

Operators

Custom Resources

```
apiVersion: wigm.carson-anderson.com/v1
kind: WigmGif
metadata:
  name: operators
spec:
  gif:
    name: "Deploying With Operators Like:"
    link: https://m.../giphy.gif

  service:
    create_cloud_lb: true


  ingress:
    enabled: true
```

Operator file structure

```
./wigm/operators/
└── releases/
    ├── operatorrelease.yaml
    └── satisfied.yaml
```

Operator Release 1

Name	operators	Load Balancer?	DEF	Ingress?	DEF
Link	https://media.giphy.com/media/3PhKYCVdMi87u/giphy.gif				

Operator Release 1 Process

- Create Custom Resource

```
cat > releases/operatorrelease.yaml <<EOF
apiVersion: wigm.carson-anderson.com/v1
kind: WigmGif
metadata:
  name: operators
spec:
  gif:
    link: https://media.giphy.com/[...]/giphy.gif
EOF

kubectl apply -f \
  releases/operatorrelease.yaml
```

- kubectl apply
- API POST

Operator Release 2

Name	satisfied	Load Balancer?	YES	Ingress?	NO
Link	https://media.giphy.com/media/l2JegGMtnxw0Nq3pC/giphy.gif				

Operator Release 2 Process

- Create Custom Resource

- kubectl apply
- API POST

```
cat > releases/satisfied.yaml <<EOF
apiVersion: wigm.carson-anderson.com/v1
kind: WigmGif
metadata:
  name: satisfied
spec:
  gif:
    link: https://media.giphy.com/media/l2JegGMtnxw0Nq3pC/giphy.gif
  service:
    create_cloud_lb: true
  ingress:
    enabled: false
EOF

kubectl apply -f releases/satisfied.yaml
```

Operator Update Process

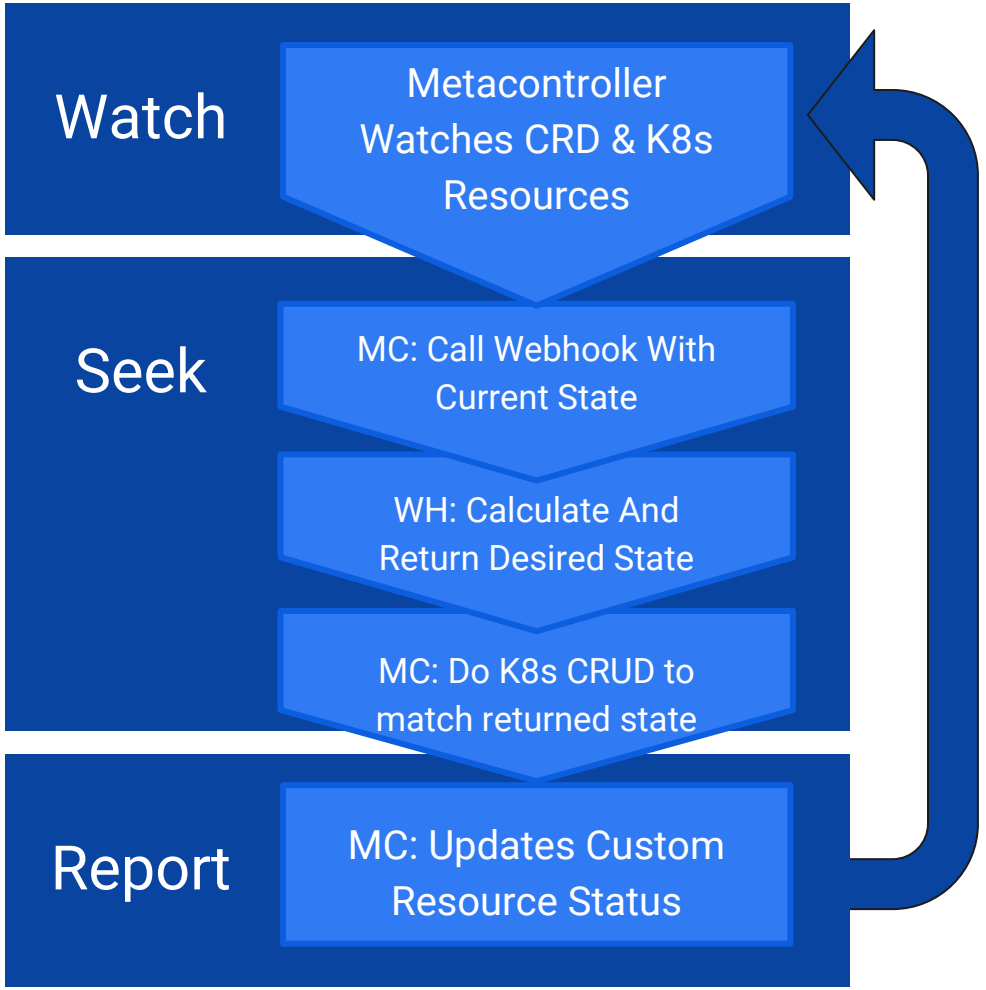
- Kubectl edit …
- kubectl patch …
- kubectl apply …
- API PATCH

```
cat > releases/operatorrelease.yaml <<EOF
apiVersion: wigm.carson-anderson.com/v1
kind: WigmGif
metadata:
  name: operators
spec:
  gif:
    link: https://media.giphy.com/[...]/giphy.gif
EOF

kubectl apply -f \
  releases/operatorrelease.yaml
```

WIGM Via Metacontroller

Operators

Metacontroller



Metacontroller Setup

- Install Metacontroller into cluster
- Run webhook server
  - In cluster
  - Externally
- Register webhook with metacontroller
  - CompositeController custom resource

Metacontroller File Structure

```
./wigm/operators/with-metacontroller/
├── controller-metaconfig.yaml
├── controller-deploy.yaml
└── controller.py
```

Operators

Metacontroller

CompositeController

```
apiVersion: metacontroller.k8s.io/v1alpha1
kind: CompositeController
metadata:
  name: wigm-controller
spec:
  generateSelector: true
  parentResource:
    apiVersion: wigm.carson-anderson.com/v1
    resource: wigmgifs
  childResources:
  - apiVersion: apps/v1
    resource: deployments
    updateStrategy:
      method: InPlace
  - apiVersion: v1
    resource: services
    updateStrategy:
      method: InPlace
  - apiVersion: extensions/v1beta1
    resource: ingresses
    updateStrategy:
      method: InPlace
  hooks:
    sync:
      webhook:
        url: http://wigm-controller.wigm/sync
```

Operators

Metacontroller
controller.py
sync

```
[...]
def sync(self, parent, children):
  name = parent["metadata"]["name"]

  # default status
  status = {
  [...]
  }

  # Generate the desired child objects
  children = [
    self.get_deployment(parent, name),
    self.get_service(parent, name),
  ]

  #conditionally create ingress, default TRUE
  if parent["spec"].get("ingress", {}).get("enabled", True):
    children.append(
      self.get_ingress(parent, name)
    )

  return {"status": status, "children": children}
```

Operators

Metacontroller
controller.py
get_deployment

```
def get_deployment(self, parent, name):
    gifname = parent["spec"]["gif"].get("name", name)
    giflink = parent["spec"]["gif"]["link"]

    deployment = {
      "apiVersion": "apps/v1",
      "kind": "Deployment",
      "metadata": {
        "name": "wigm-host-%s"%(name),
        "labels": { "app": "wigm",
                    "gif": name }
      },
      "spec": {
        [...]
          "spec": {
            "containers": [
              {
               [...]
                "env": [
                  { "name": "GIF_NAME",
                    "value": gifname },
                  { "name": "GIF_SOURCE_LINK",
                    "value": giflink
                  }
                [...]
    return deployment
```

Operators

Metacontroller
controller.py
get_service

```
def get_service(self, parent, name):
    service = {
      "kind": "Service",
      "apiVersion": "v1",
      "metadata": {
        "name": "wigm-host-%s"%(name),
        "labels": {
          "app": "wigm",
          "gif": name
        }
      },

    # conditionally set service kind
    if parent["spec"].get(
         "service", {}).get("create_cloud_lb"):
      service["spec"]["type"] = "LoadBalancer"

    return service
```

Operators

Metacontroller
controller.py
get_ingress

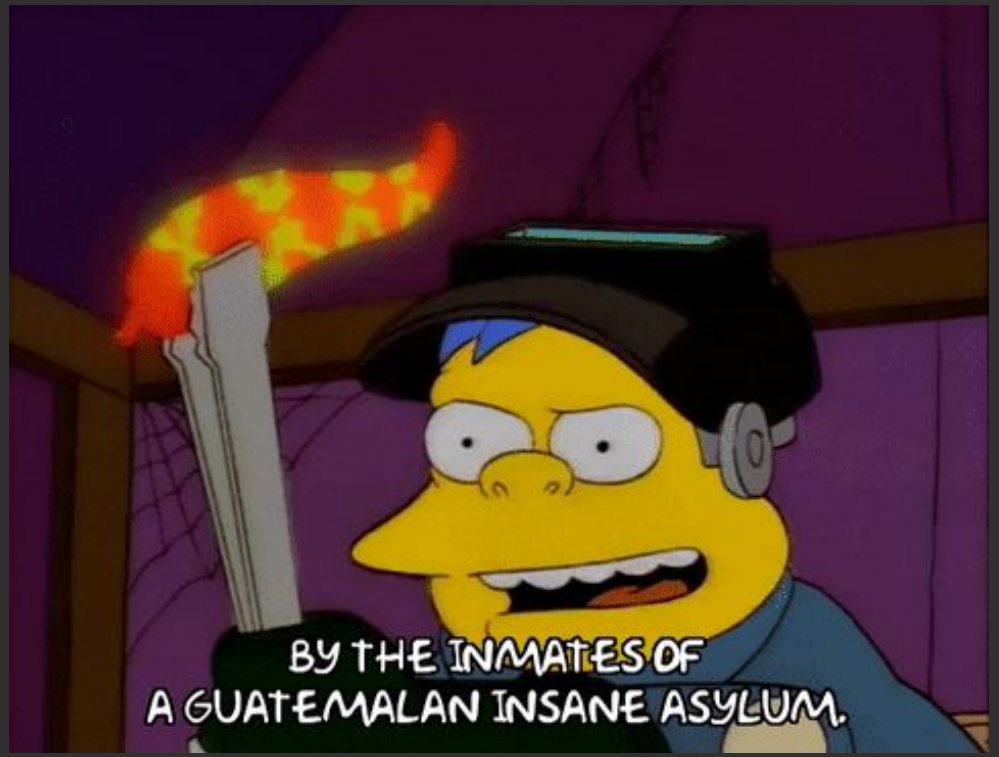
```
# if parent["spec"].get("ingress", {}).get("enabled",):
#   children.append(
#    self.get_ingress(parent, name)
#   )

---

def get_ingress(self, parent, name):
    ingress = {
      "apiVersion": "extensions/v1beta1",
      "kind": "Ingress",
      "metadata": {
        "name": "wigm-host-%s"%(name),
       [...]
      },
      "spec": {
        "rules": [
          {
           "host": "%s.wigm.carson-anderson.com"%(name),
           [...]
    }

    return ingress
```

DEMO

Metacontroller In Action

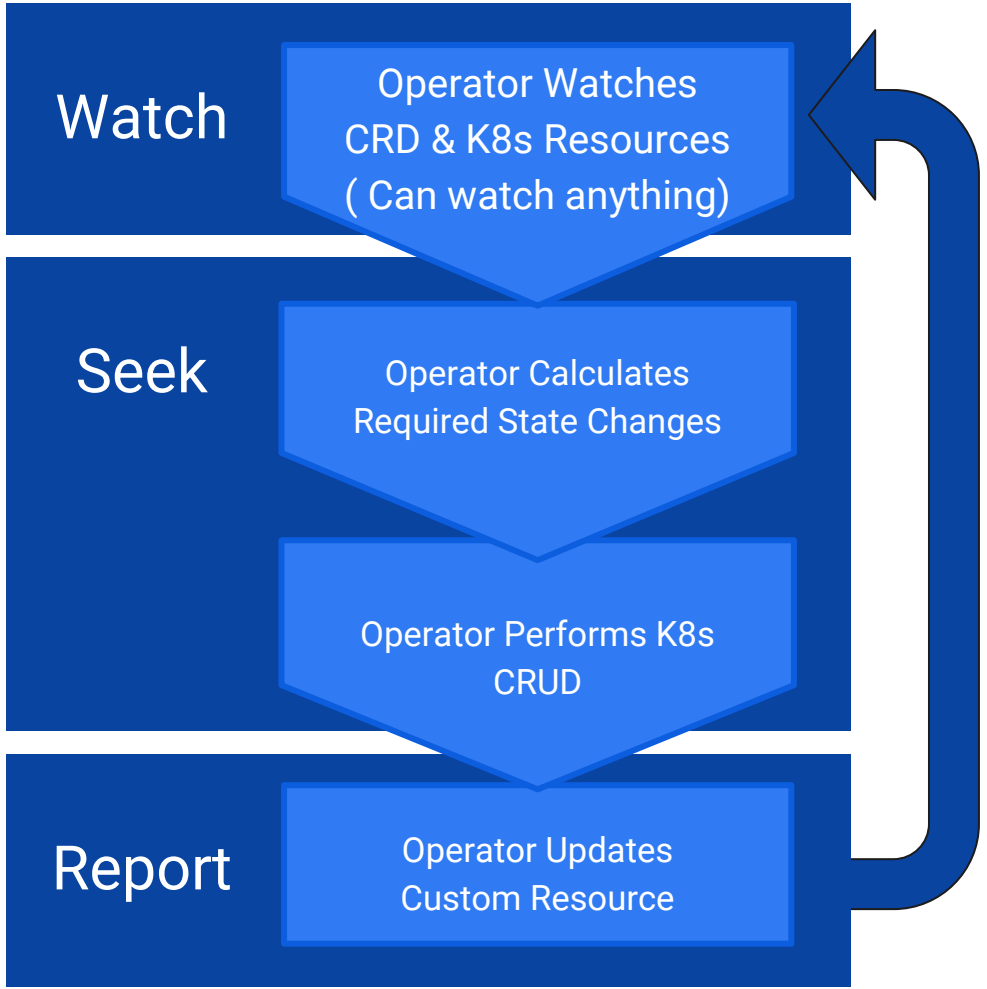
WIGM Via

Operator-SDK (Golang)



Operators

Operator SDK



Operator-SDK Setup

- Build
- Run locally

OR

- Deploy operator into cluster

Operators

Operator SDK New

```
# Generate boilerplate controller
operator-sdk new wigm

# Go to controller root
cd wigm
```

Operator-SDK Deploy Files

```
./wigm/operators/with-operator-sdk/
└── deploy
    ├── operator.yaml
    ├── role_binding.yaml
    ├── role.yaml
    └── service_account.yaml
```

Operators

Operator SDK Generate Types

```
# Generate boilerplate CRD and API Spec
operator-sdk add api \
--api-version=wigm.carson-anderson.com/v1 \
--kind=WigmGif
```

Operator-SDK Deploy Files

```
./wigm/operators/with-operator-sdk/
└── deploy
    ├── crds
    │   └── wigm_v1_wigmgif_crd.yaml
    ├── operator.yaml
    ├── role_binding.yaml
    ├── role.yaml
    └── service_account.yaml
```

Operator-SDK Types Files

```
./wigm/operators/with-operator-sdk/
└── pkg
    └── apis
        └── wigm
            └── v1
                └── wigmgif_types.go
```

Operators

Operator SDK Edit Types

```
# File ./pkg/apis/wigm/v1/wigmgif_types.go
[...]
// WigmGifSpec defines the desired state of
WigmGif
type WigmGifSpec struct {
 Gif      GifProperties [...]
 Service *ServiceProperties [...]
 Ingess  *IngressProperties [...]
}

type GifProperties struct {
 Name string `json:"name,omitempty"`
 Link string `json:"link"`
}

type ServiceProperties struct {
 CreateCloudLB bool `json:"create_cloud_lb"`
}

type IngressProperties struct {
 Enabled bool `json:"enabled"`
}
```

Operators

Power of Embedded Types

```
# File ./pkg/apis/wigm/v1/wigmgif_types.go
import (
  corev1 "k8s.io/api/core/v1"
)
[...]
// WigmGifSpec defines the desired state of
WigmGif
type WigmGifSpec struct {
 Gif GifProperties [...]
 Service *ServiceProperties [...]
}

type GifProperties struct {
 Name string `json:"name,omitempty"`
 Link string `json:"link"`
}

type ServiceProperties struct {
 CreateCloudLB bool `json:"create_cloud_lb"`
 ServiceSpec corev1.ServiceSpec [...]
}
```

Operators

Generate Control Loop

```
# Generate boilerplate controller
operator-sdk add controller \
--api-version=wigm.carson-anderson.com/v1 \
--kind=WigmGif
```

Operator-SDK Controller Files

```
./wigm/operators/with-operator-sdk/
└── pkg
    └── controller
        └── wigmgif
            └── wigmgif_controller.go
```

Operators

WigmGIF Controller Psudocode

```
controller = NewController()

controller.Watch(WigmGifs)
controller.Watch(Child Deployments)
controller.Watch(Child Services)
controller.Watch(Child Ingresses)

for wigmggif in controller.Changes() {
  controller.SyncDeployment(wigmggif)
  if error {
    handleError()
  }
  controller.SyncService(wigmggif)
  if error {
    handleError()
  }
  controller.SyncIngress(wigmggif)
  if error {
    handleError()
  }
}
```

Operators

WigmGIF Controller

```
# /pkg/controller/wigmgif/wigmgif_controller.go

func add(mgr manager.Manager, r reconcile.Reconciler)
error {
      // Create a new controller
      c, err := controller.New("wigmgif-controller", mgr,
controller.Options{Reconciler: r})
      if err != nil {
            return err
      }

      // Watch for changes to primary resource WigmGif
      err = c.Watch(&source.Kind{Type: &wigmv1.WigmGif{}},
&handler.EnqueueRequestForObject{})
      if err != nil {
            return err
      }

      // Watch for changes to child deployments and
requeue the owner WigmGif
      err = c.Watch(&source.Kind{Type:
&appsv1.Deployment{}}, &handler.EnqueueRequestForOwner{
            IsController: true,
            OwnerType:    &wigmv1.WigmGif{},
      })
      [...]
```

Operators

WigmGIF Controller Syncs

```
func (r *ReconcileWigmGif) Reconcile(request reconcile.Request)
(reconcile.Result, error) {

[...]

// Sync the deployment, updating status in the passed instance
      if r, err := r.syncDeployment(instance); err != nil ||
r.Requeue == true {
            return r, err
      }

      // Sync the service, updating status in the passed instance
      if r, err := r.syncService(instance); err != nil ||
r.Requeue == true {
            return r, err
      }

      // Sync the ingress, updating status in the passed instance
      if r, err := r.syncIngress(instance); err != nil ||
r.Requeue == true {
            return r, err
      }

      // update the status
      if err := r.client.Status().Update(context.TODO(),
instance); err != nil
[...]
```

Operators

WigmGIF Controller Get Deployment

```
func newDeploymentForWG(wg *wigmv1.WigmGif)
*appsv1.Deployment {
      name := wg.GetName()

      giflink := wg.Spec.Gif.Link
      gifname := name
      if wg.Spec.Gif.Name != "" {
            gifname = wg.Spec.Gif.Name
      }

      labels := map[string]string{
            "app": "wigm",
            "gif": name,
      }

      deployment := &appsv1.Deployment{
            ObjectMeta: metav1.ObjectMeta{
                  Name:      "wigm-host-" + name,
                  Namespace: wg.GetNamespace(),
                  Labels:    labels,
            },
            Spec: appsv1.DeploymentSpec{
                  [...]
                  Containers: []corev1.Container{{
                        Name:  "gifhost",
                        Image: "nginx:1.15-alpine",
                        Env: []corev1.EnvVar{{
                              Name:  "GIF_NAME",
                              Value: gifname,},
```

Operators

WigmGIF Controller Get Service

```
func newServiceForWG(wg *wigmv1.WigmGif) *corev1.Service
{
      [...]

    if (
        wg.Spec.Service != nil &&
        wg.Spec.Service.CreateCloudLB
        ) {
        service.Spec.Type = corev1.ServiceTypeLoadBalancer
    }

      [...]
}
```

Operators

Golang Fake Client Testing

```
func TestWigmGifBasicCreate(t *testing.T) {
[...]
     // Create a fake client to mock API calls.
     cl := fake.NewFakeClient(objs...)
[...]
     res, err := r.Reconcile(req)

[...]
     err = cl.Get(context.TODO(),
          types.NamespacedName{
          Name: expectedDeployment.GetName(),
          Namespace: expectedDeployment.GetNamespace()},
          foundDeployment)
     if err != nil {
          t.Errorf("Error getting expected
deployment: %s", err)
     }
```

DEMO

Golang Operator In Action

The Evolution WIGM

YAML

HELM

Operators

Management Pattern Comparison

	Develop Speed	Portability	Release Creation	Release Updates	Testability	Automatic State Seek	Performance
YAML			Manual	Manual	Manual		
Helm		Req. local copy of the templates and values		Manually Applied			
Operator: Metacontroller	Webhook Redeploy	Supports any K8s API tool or Code				Batched	
Operator: operator-sdk	Golang Learning Curve	Supports any K8s API tool or Code					

More of Me

github.com/carsonoid/talk-kubernetes-operators

K3s (demos): github.com/rancher/k3s

@carsonoid

@carson_ops

kube-decon.carson-anderson.com

dynamic-kubernetes.carson-anderson.com

salt-decon.carson.anderson.com