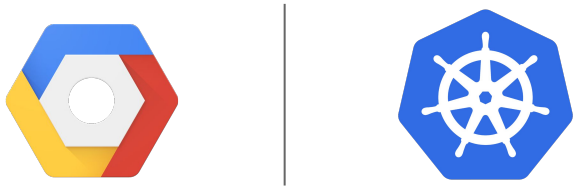


Lets Build:
Kubernetes Operator

Say Hi
+
Ask for help

slack

googlecloud-community.slack.com

#kubecamp

Jim Angel

Cloud
Consultant

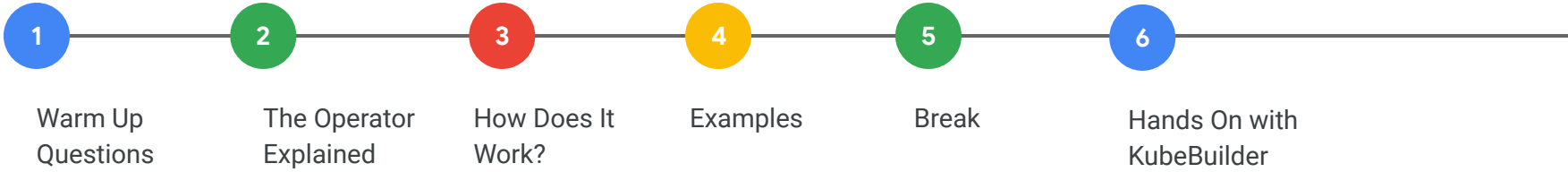
jimangel@google.com

Peter Blum

Cloud
Engineer

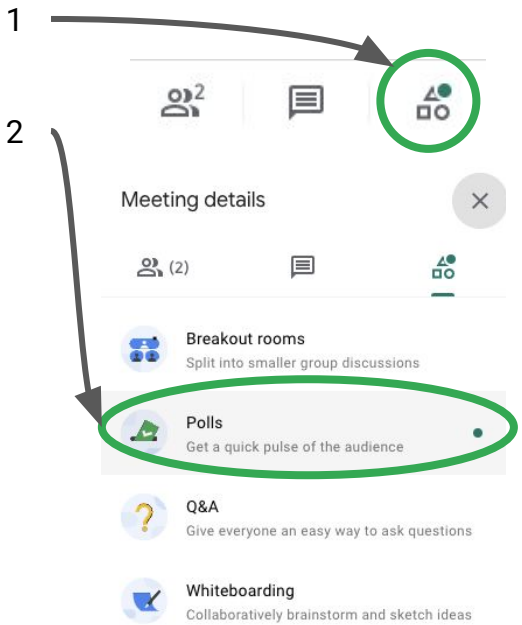
blump@google.com

Agenda


1
2
3
4
5
6
Warm Up Questions
The Operator Explained
How Does It Work?
Examples
Break
Hands On with KubeBuilder

Warm Up Questions

Polls



1. Do you use Kubernetes Regularly?
2. Are you familiar with Golang?
3. Do you know what a Kubernetes Controller is?
4. Do you know what a Kubernetes Custom Resource is?

The Operator

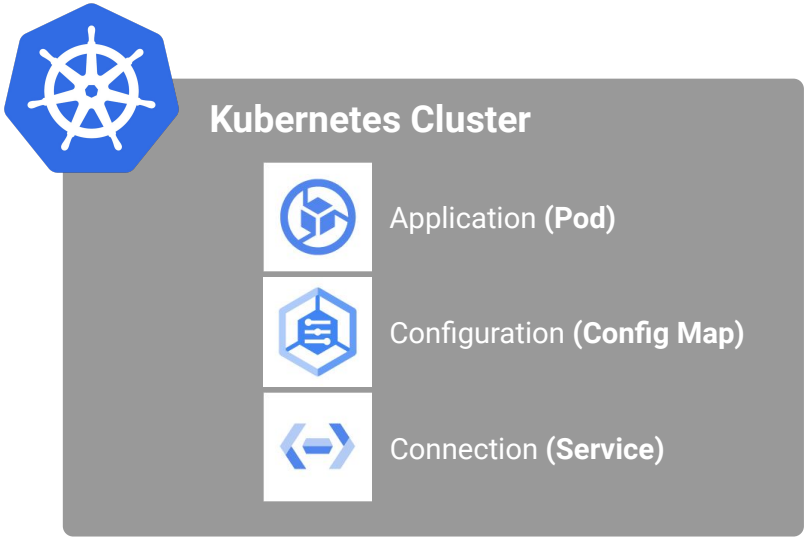
Performs complex actions on stateful applications using 3 things:

1. Kubernetes Resources
2. Kubernetes Controller
3. Application Knowledge

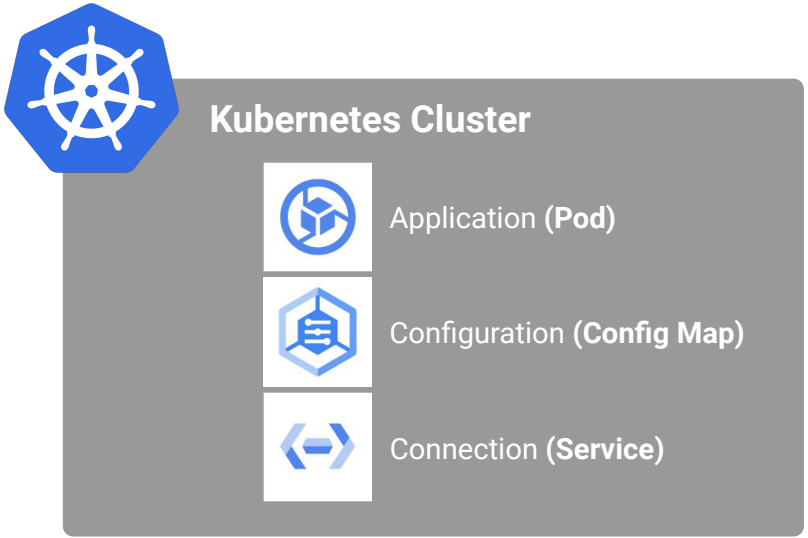
Okay....

The Operator...Explained!

Cloud Native Application

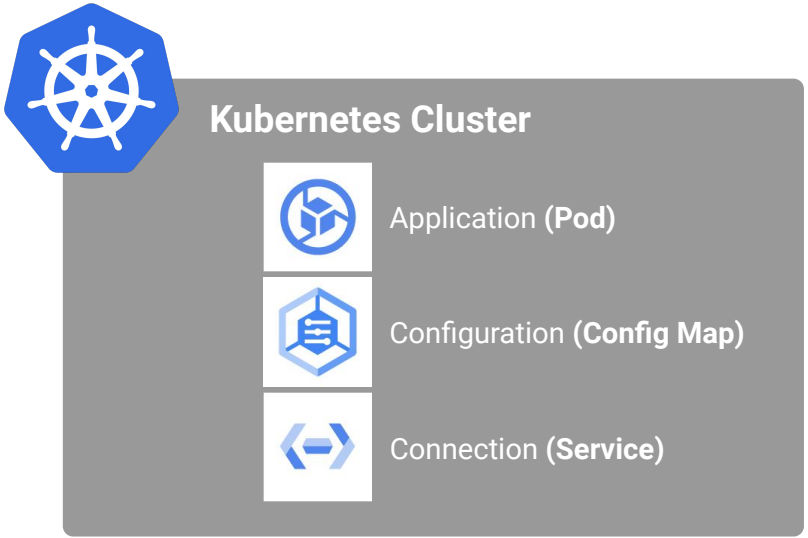


Cloud Native Application



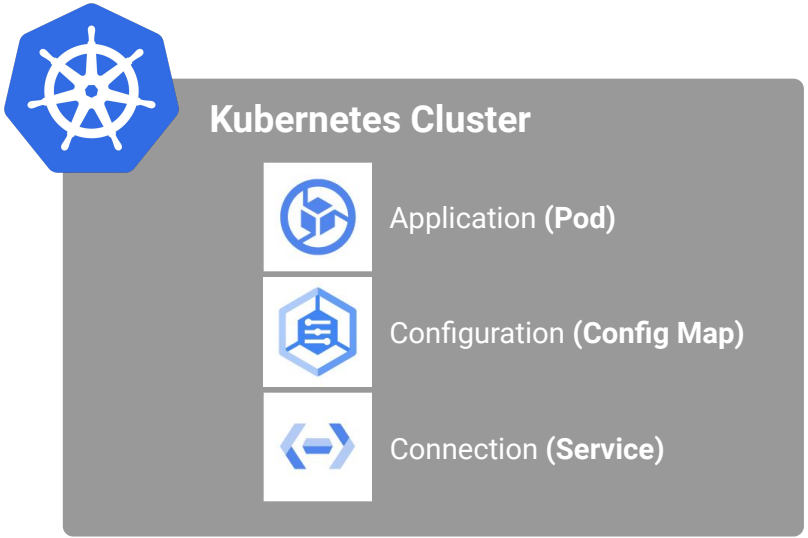
High Availability ?
Scaling ?
New Rollouts ?
Backups ?

Cloud Native Application



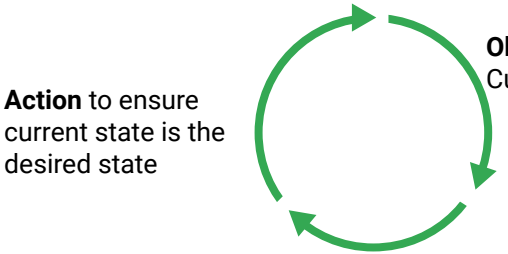
✅ High Availability → *ReplicaSet*
✅ Scaling → *HPA*
✅ New Rollouts → *Deployment*
~~Backups → *Not Needed*~~

Cloud Native Application

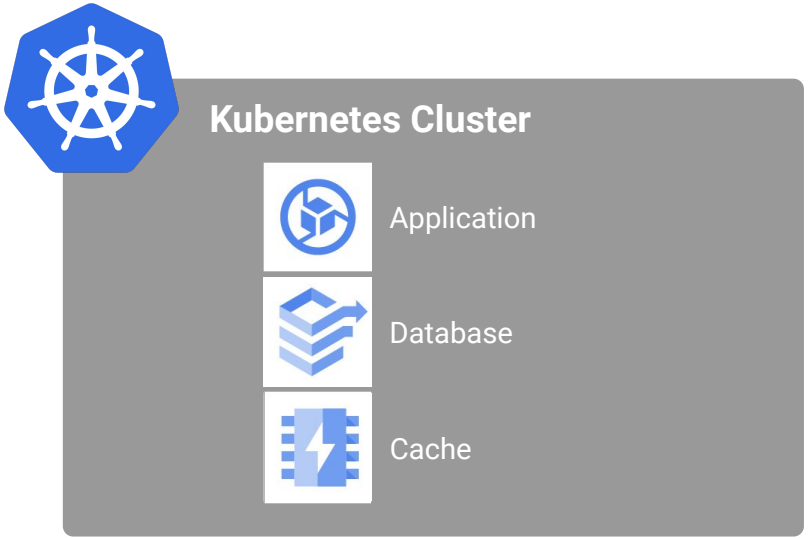


✅ High Availability → *ReplicaSet*
✅ Scaling → *HPA*
✅ New Rollouts → *Deployment*
~~Backups → *Not Needed*~~

Control Loop



More Complex Application



❓ **High Availability**
- What should the failover look like?

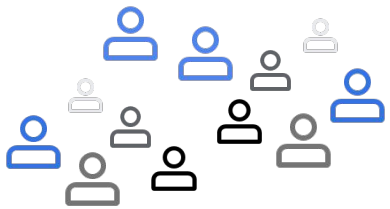
❓ **Scaling**
- When should each component be scaled?

❓ **New Rollouts**
- Which component should restart first?

❓ **Backups**
- What items need to be backed up?

More Complex Application

Ops Team



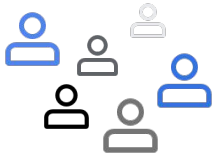
Manual tasks and checklists

- Follow a checklist to backup
- Run a command to determine current demand
- During peak times follow another checklist to scale up

Toil - “work that is manual, repetitive, automatable, tactical, devoid of enduring value, and that scales linearly as an application grows.”

Is there a better way?

Eliminating Toil

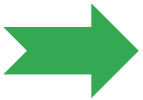


Manual Tasks

- How to deploy the application?
- How to scale up/down replicas?
- How to recover?

Teams who “operate” these applications

Human Operator



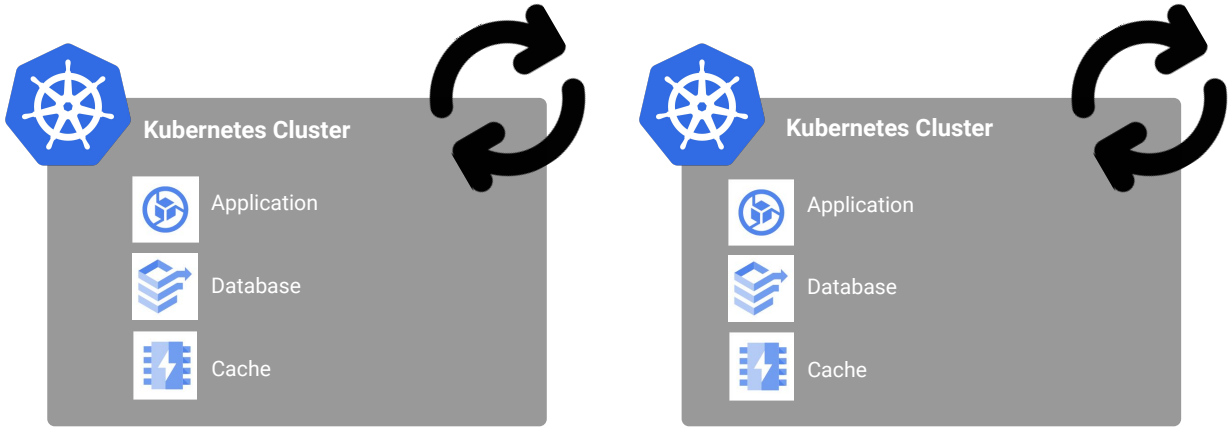
Automated Tasks

- Codified application knowledge
- Reusable patterns

Software who “operate” these applications

Software Operator

Reusable Automation



Operator brings a single standardized automated process to managing complex applications

So....how does this work?

Kubernetes Operator

1. Custom Resource
2. Controller
3. Application Knowledge

Step 1 - Custom Resource

1 Custom Resource

- Definition of the application our operator will manage for us
- Allows us to extend the Kubernetes API

Step 1 - Custom Resource

1 Custom Resource

```yaml
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
  name: mongodbs.db.example.com
spec:
  group: db.example.com
  version: v1alpha1
  scope: Namespaced
  names:
    plural: mongodbs
    singular: mongodb
    kind: MongoDb
    shortNames:
    - mdb
```

```
kubectl apply -f mongodb-crd.yml
```

```
kubectl get mongodbs
No resources found in default namespace.
```

Step 1 - Custom Resource

1 Custom Resource

```yaml
apiVersion: db.example.com/v1beta1
kind: MongoDb
metadata:
  name: webscale
spec:
  user: admin
  password: secret
  size: 3
```

```
kubectl apply -f mongodb-instance.yml
mongodb.db.example.com/webscale created
```

```
kubectl get mdb
NAME        AGE
webscale    44s
```

Step 1 - Custom Resource

1 Custom Resource

```
kubectl get mongodbs -o yaml
apiVersion: v1
items:
- apiVersion: db.example.com/v1alpha1
  kind: MongoDb
  metadata:
    annotations:
      kubectl.kubernetes.io/last-applied-configuration: |
        {"apiVersion":"db.example.com/v1alpha1","kind":"MongoDb","metadata":{"annotat
ions":{},"name":"webscale","namespace":"default"},"spec":{"password":"secret","size":
3,"user":"admin"}}
    creationTimestamp: "2021-05-25T11:37:11Z"
```

Step 2 - Controller

2 Controller

We have a custom resource definition (CRD)....cool.

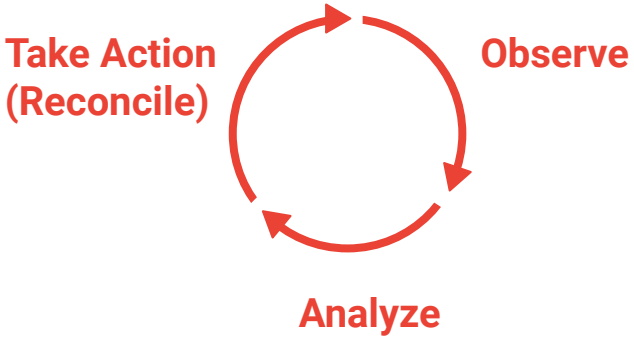
Next we need something to perform an **action** upon the presence of this new CRD.

Welcome.....*Controller* to the stage!

Step 2 - Controller

2 Controller

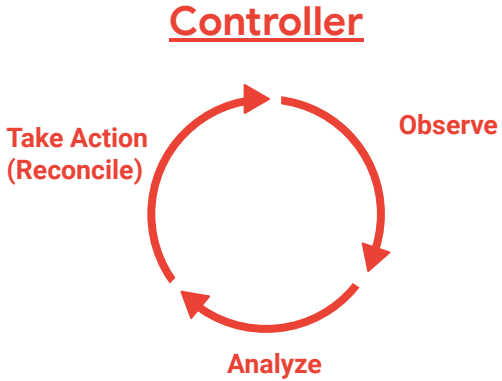
One Big Loop!



Step 2 - Controller

2 Controller

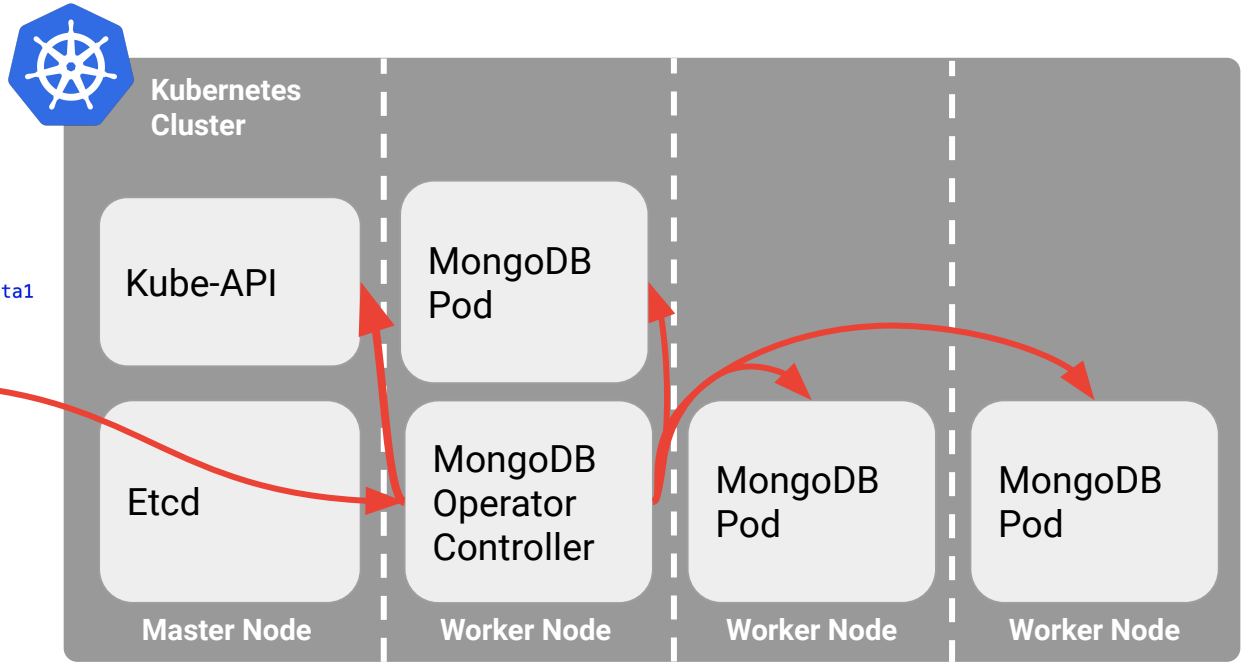
Observe - Current state of your CRD
Analyze - Desired state of your CRD to the current state.
Take Action - Perform tasks to bring the current state of your CRD to the desired state



Step 2 - Controller

2 Controller

```
apiVersion: db.example.com/v1beta1
kind: MongoDb
metadata:
  name: webscale
spec:
  user: admin
  password: secret
  size: 3
```



....but that's not reality ಠ_ಠ

3 Application Knowledge

Being a MongoDB administrator comes with many many...many tasks.

Need to be **codified** and placed into the operator.

Application knowledge...the most important part. Application knowledge that **you** have!

....but that's not reality ಠ_ಠ

3 Application Knowledge

Checking Collections at startup
Specifying command line args
Running mongod as an unprivileged user
Administrating privileges
Backing up data volumes
Crash recovery
Linux security
Mongodb updates
Linux Updates
Log file maintenance
Safe shutdown
.....

Show me the examples!

Example: ReplicaSet

You can think of the *replicaset* as a custom resource defined in the core kubernetes API which the *replicaset controller* operates on.

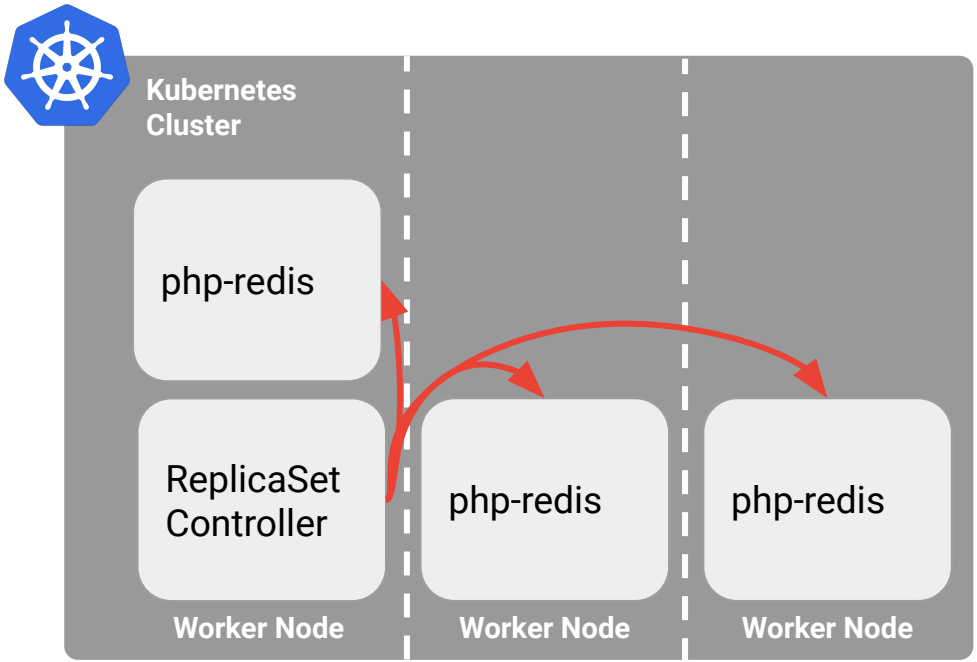
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
spec:
  replicas: 3
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/frontend:v3
```

Example: ReplicaSet

```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
spec:
  replicas: 3
    spec:
      containers:
      - name: php-redis
        image: gcr.io/google_samples/frontend:v3
```

Desired (Pods) --- Current (Pods)

3	Scale ⬆️	0
3	Scale ⬆️	1
3	Scale ⬆️	2
3	👍	3



Other Examples:

Kubernetes (Core API):
Deployment
DaemonSet
...

Istio:
VirtualRoute
Gateway
...

Your Operator:
Custom Resource
...

Break Time

Hands on section up next!

5

Google

Lets build it!

Guestbook Example