Kubernetes Operators
for Application Developers

Josh Wood
joshix@redhat.com
Developer Advocate
2019-01-25 — DevConf.cz — Brno
About Josh

joshix@redhat.com
https://GitHub.com/joshix
@joshixisjosh9

Developer Advocate, Red Hat OpenShift and Kubernetes
Formerly: DocOps, CoreOS
Any application on any platform must be installed, configured, managed, and upgraded over time.
   Patching is critical to security

Anything that isn’t automated is slowing you down.

Operators are automated software managers for Kubernetes clusters: Install and Lifecycle.
You can use Operators today

You find a killer new database, perfect match for your project’s requirements

You don’t want to become an expert killernewDB DBA

Operators make it easier to run foundation services
You can build Operators today

With the Operator Framework SDK

https://github.com/operator-framework/operator-sdk
Scaling stateless apps: Easy
ReplicaSet

$ kubectl scale depl/staticweb --replicas=3
ReplicaSet

![Diagram showing ReplicaSet and Pod relationships with kubectl](image-url)
ReplicaSet

**ReplicaSet**

```
ReplicaSet
app=web, env=prod
```

```
Pod
app=web
env=prod
count=3
```

```
desired=3
```

kubectl
What about apps that... store data?
Creating a database is easy

$ kubectl run db --image=quay.io/my/killernewdb
Running a database is harder

• **Resize/Upgrade** - coordination for availability
• **Reconfigure** - tedious generation / templating
• **Backup** - requires coordination among instances
• **Healing** - restore backups, rejoin db cluster
If only Kubernetes knew...

How? Extend Kubernetes
The goal

$ cat database-cluster.yaml
spec:
  clusterSize: 3
  readReplicas: 2
  version: v4.0.1
[...]
What are Operators?

**Application-specific controllers** that extend the Kubernetes API to create, configure, and manage instances of complex stateful applications on behalf of a Kubernetes user

1. Extend the Kubernetes API through the Custom Resources (CR(D)) mechanism

2. Custom controller watches CRs, acts in response
Example: etcd Operator

$ cat deployment.yaml
spec:
  clusterSize: 3
  version: v3.2.23
[...]
etcd Operator

Cluster “A” has 2 running pods:
- name: A-000, version 3.0.9
- name: A-001, version 3.1.0

Differences from desired config:
- should be version 3.1.0
- should have 3 members

How to get to desired config:
- Recover 1 member
- Back up cluster
- Upgrade to 3.1.0
$ kubectl create -f example/deployment.yaml
$ sh create_role.sh
   # RBAC role and rights
$ kubectl create -f deployment.yaml
   # Deploy the etcd Operator
$ kubectl create -f etcd-cluster.yaml
   # Create an etcd cluster
$ kubectl {delete,apply,get}
   # etcd failure recovery, version
   upgrade, Operator failure recovery

$> etcdctl put foo bar # While using etcd API
What are Operators good for?

Databases

File, block, and object storage

Applications with their own notion of “cluster”

Apps for distribution on Kubernetes
Play with Operators right now

https://learn.openshift.com/operatorframework/
Use Operators today

https://github.com/operator-framework/awesome-operators

• etcd
• Prometheus
• Elasticsearch
• MySQL
• Postgres (crunchy)
• “and many more!”
Build your Operator

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

An Operator is a custom Kubernetes controller for your app.

The SDK makes it easier to build Operators:
• High level APIs and abstractions to write operational logic
• Scaffolding and code generation to bootstrap new projects
• Extensions to cover common Operator use cases
What are Operators good for?

Get persistence faster, cheaper
Reliability automated down the stack

Advanced:
Make your app a Kubernetes Native
Operator Maturity

<table>
<thead>
<tr>
<th>Phase I</th>
<th>Phase II</th>
<th>Phase III</th>
<th>Phase IV</th>
<th>Phase V</th>
</tr>
</thead>
<tbody>
<tr>
<td>Installation</td>
<td>Upgrades</td>
<td>Lifecycle</td>
<td>Insights</td>
<td>Auto-pilot</td>
</tr>
<tr>
<td>Configuration</td>
<td>Patch and minor</td>
<td>App lifecycle,</td>
<td>Metrics, alerts,</td>
<td>Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning...</td>
</tr>
<tr>
<td>and one-shot</td>
<td>version upgrades</td>
<td>storage lifecycle</td>
<td>log processing</td>
<td></td>
</tr>
<tr>
<td>deployment</td>
<td>supported</td>
<td>(backup, failure</td>
<td>and workload</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>recovery)</td>
<td>analysis</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Resources

- https://coreos.com/operators

- Learn.OpenShift.com: https://learn.openshift.com/operatorframework/

- Etcd Operator examples from this talk: Recovery, backup, upgrade: https://github.com/coreos/etcd-operator/

- Awesome Operators! https://github.com/operator-framework/awesome-operators

- Operator Framework and SDK on Github https://github.com/operator-framework/

THANK YOU

plus.google.com/+RedHat
linkedin.com/company/red-hat
youtube.com/user/RedHatVideos
facebook.com/redhatinc
twitter.com/RedHatNews