cloudflight

DevConf 2020, Brno

Operators for Beginners

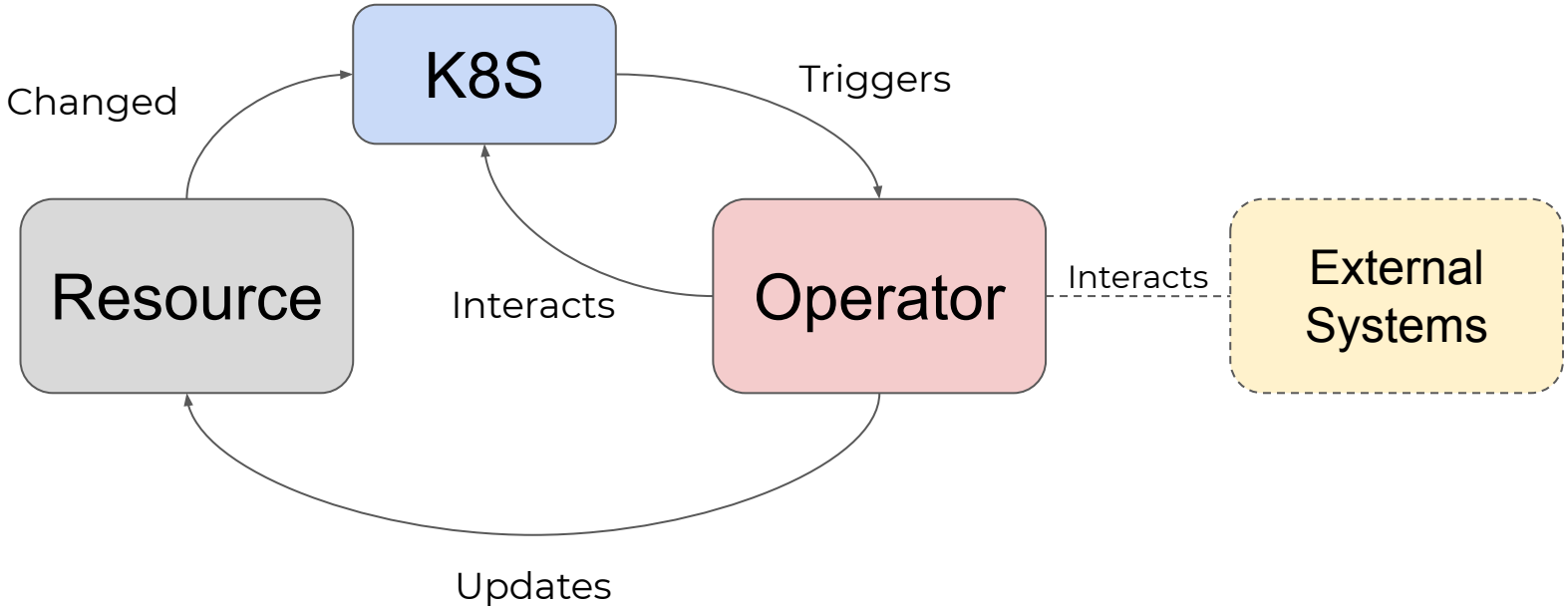
Dominik Süß

Operators?

Operators!

- An Operator is a method of packaging, deploying and managing a Kubernetes application.
- Integrated into the Kubernetes API
- Can be controlled via standard Kubernetes tooling
- Packages **Operational** Knowledge

Operator Architecture



Custom Resources

Abusing Kubernetes as KV Store

```yaml
apiVersion: operator.cloudflight.io/v1alpha1
kind: Key
metadata:
  name: foo
value: "BAR"
```

Abusing Kubernetes as KV Store

```
$ kubectl get keys
NAME    VALUE
foo     BAR
```

Example Operators

Example Operators

Community

Prometheus Operator

provided by Red Hat

Manage the full lifecycle of configuring and managing Prometheus and Alertmanager servers.

Community

Community Jaeger Operator

provided by CNCF

Provides tracing, monitoring and troubleshooting for microservices-based

Crunchy Postgres Cluster

provided by CrunchyData.com

A Postgres Operator from Crunchydata.com

Community

AWS S3 Operator

provided by Red Hat

Manage the full lifecycle of installing, configuring and managing AWS S3 Provisioner.

Kiali Operator

provided by Red Hat

Kiali project provides answers to the questions: What microservices are part of my Istio service mesh and how

Example Custom Resources

```
apiVersion: monitoring.coreos.com/v1
kind: PodMonitor
metadata:
  name: example
spec:
  selector:
    matchLabels:
      k8s-app: prometheus
```

Example Custom Resources

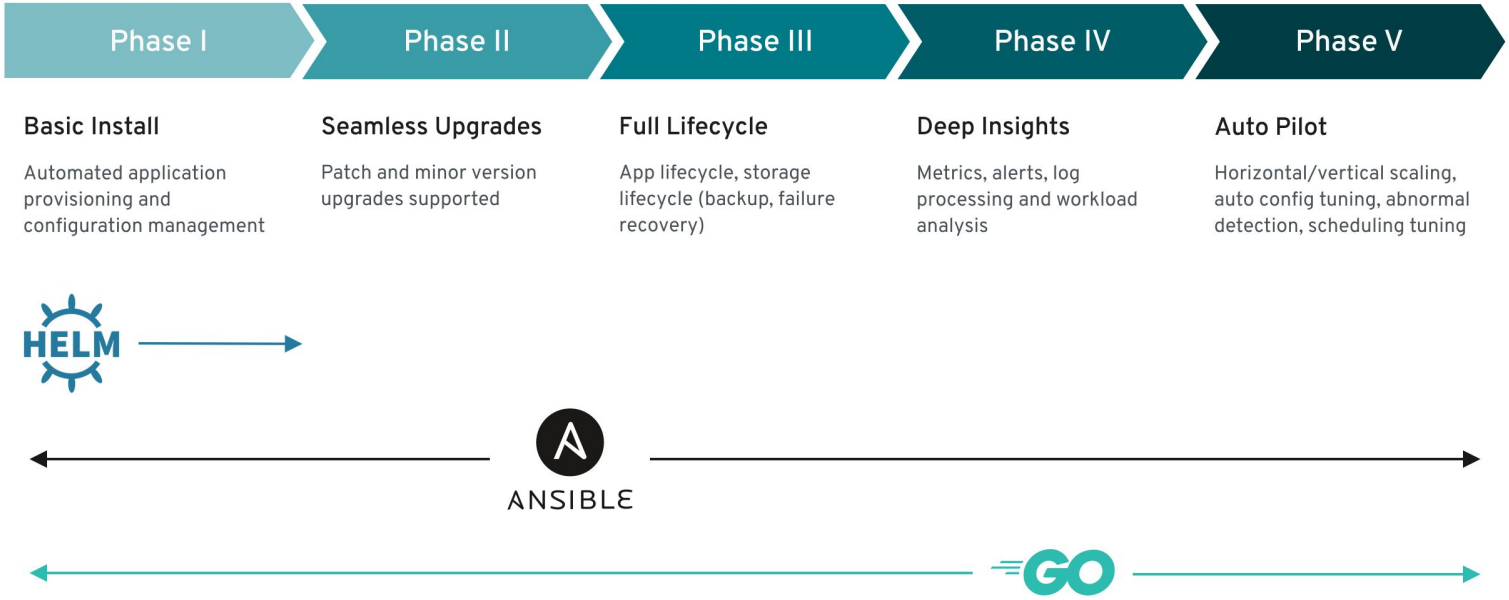
```
apiVersion: elasticsearch.k8s.elastic.co/v1
kind: Elasticsearch
metadata:
  name: elasticsearch-sample
spec:
  version: 7.5.1
  nodeSets:
   - name: default
   config:
       node.master: true
       node.data: true
```

Developing the Operator

Frameworks

- operator-sdk
- controller-rs
- kubebuilder
- ...

Operator SDK



https://docs.openshift.com/container-platform/4.2/operators/olm-what-operators-are.html

Helm Operator

github.com/CloudflightIO/operator-basics

Helm
Operator

- Uses simple Helm Charts
- Basic Install Capability
- Best for small applications and existing Charts

Generated Structure

```
redis-operator
├── build/{Dockerfile}
├── deploy/{crds/,operator.yaml,role.yaml}
├── helm-charts/redis/{Chart.yaml,...}
└── watches.yaml
```

The watches.yaml file

```
---
- version: v1alpha1
  group: example.cloudflight.io
  kind: Redis
  chart: /opt/helm/helm-charts/redis
```

Using the CR Values

```
# values.yaml
version: '5.0.7'
storage: '1Gi'
resources: {}
```

Using the CR Values

```yaml
apiVersion: example.cloudflight.io/v1alpha1
kind: Redis
metadata:
  name: example-redis
spec:
  storage: '200Gi'
```

Using the CR Values

```
apiVersion: v1
kind: PersistentVolumeClaim
spec:
  ...
  resources:
    requests:
      storage: {{ .Values.storage }}
```

Helm Operator Recap

- Existing Helm-Charts
- Very small Applications
- No complex integrations

Ansible Operator

github.com/CloudflightIO/operator-basics

Ansible
Operator

- Runs ansible playbooks using ansible-runner
- Full Lifecycle Management
- Integration through the k8s module
- Update the status using k8s_status

Generated Structure

```
elasticsearch-operator
├── build/{Dockerfile}
├── deploy/{crds/,operator.yaml,role.yaml}
├── molecule/{test-cluster,test-local}
├── roles/elasticsearch/{tasks/,files/,...}
└── watches.yaml
```

Using the CR

```
# defaults/main.yml
cluster_name: elasticsearch
replicas: 3
storage: 10Gi
resources:
  limits:
    cpu: "500m"
...
```

Using the CR

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: "{{ meta.name }}"
  namespace: "{{ meta.namespace }}"
data:
  elasticsearch.yml: |-
    cluster.name: {{ cluster_name }}
```

Ansible Operator Recap

- Medium Complexity
- Allows for integration with other components
- Existing Ansible knowledge
- Hard to go beyond this

Go Operator

Go
Operator

- Written using Go
- Can do everything a Go program can do
- Automatically generate CRDs
- Allows for simpler external integrations

Implementing the Operator

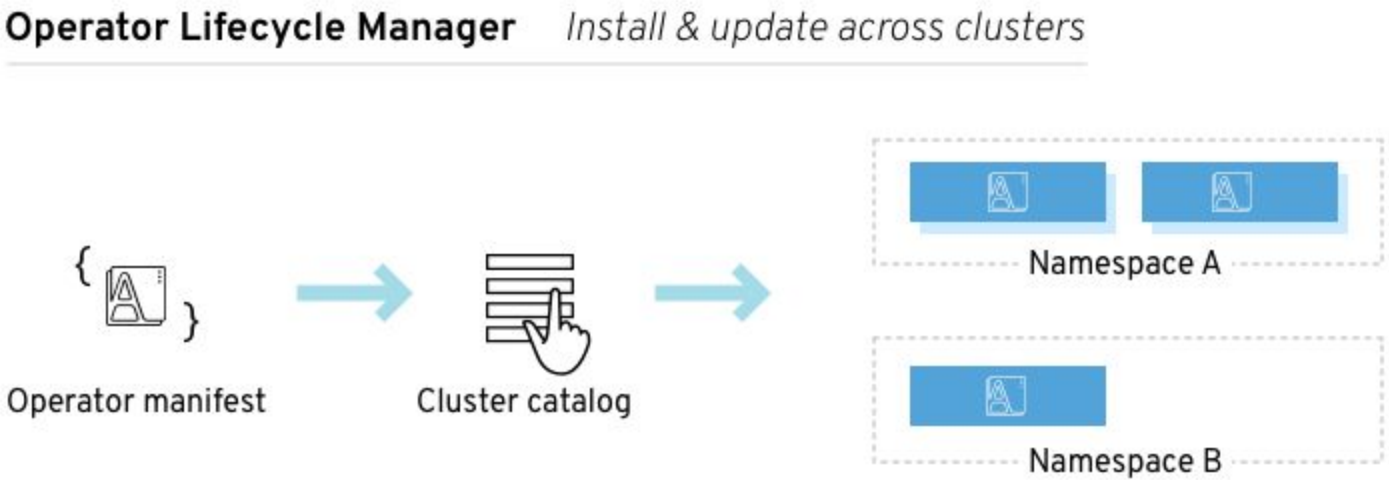
```
func (r *ReconcileCR) Reconcile(request reconcile.Request)
(reconcile.Result, error) {
…
}
```

Go Operator Recap

- Medium to High Complexity
- Allows for easy integrations
- “Flagship Operator”

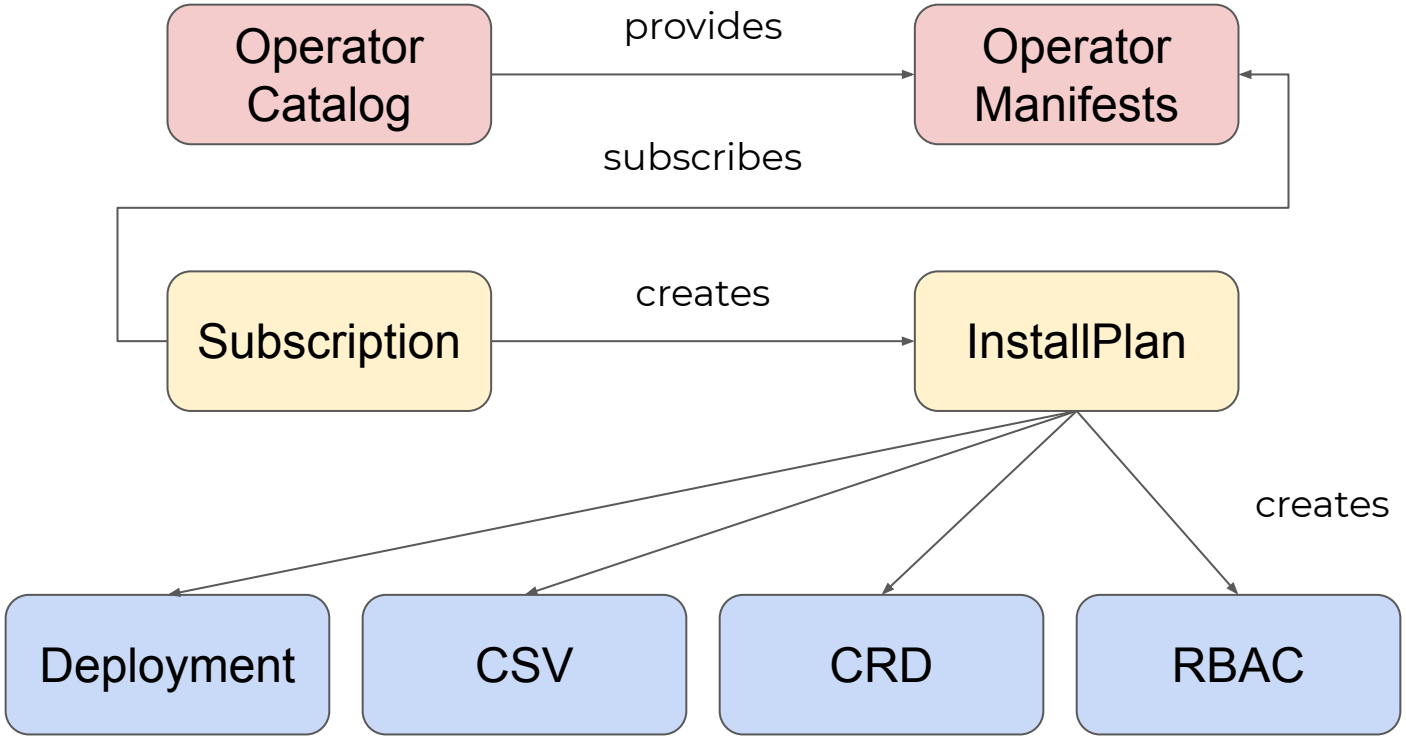
Deploying the Operator

The OLM



https://docs.openshift.com/container-platform/4.1/applications/operators/olm-understanding-olm.html

OLM Architecture



Recap

- Operators perform actions on resources
- You can define your own resources
- Operators are not magic

Q&A