The Operator Pattern

Managing Stateful Services in Kubernetes

Jakob Karalus, @krallistic

codecentric

$whoami

- Data Science + DevOps
- Codecentric
- CKA
- Twiiter: @krallistic
- Github: github.com/krallistic

codecentric AG

10x in Germany

HAMBURG
BERLIN
MÜNSTER
DORTMUND
SOLINGEN
AMSTERDAM
BREDA
FRANKFURT AM MAIN
NÜRNBERG
KARLSRUHE
STUTTGART
MÜNCHEN

Normal Kubernetes Deployment

- Write some Deployment, Services, Configmaps etc
- Deploy them to K8s
- Maybe create a Helm Chart
- YAML YAML YAML

Success?!

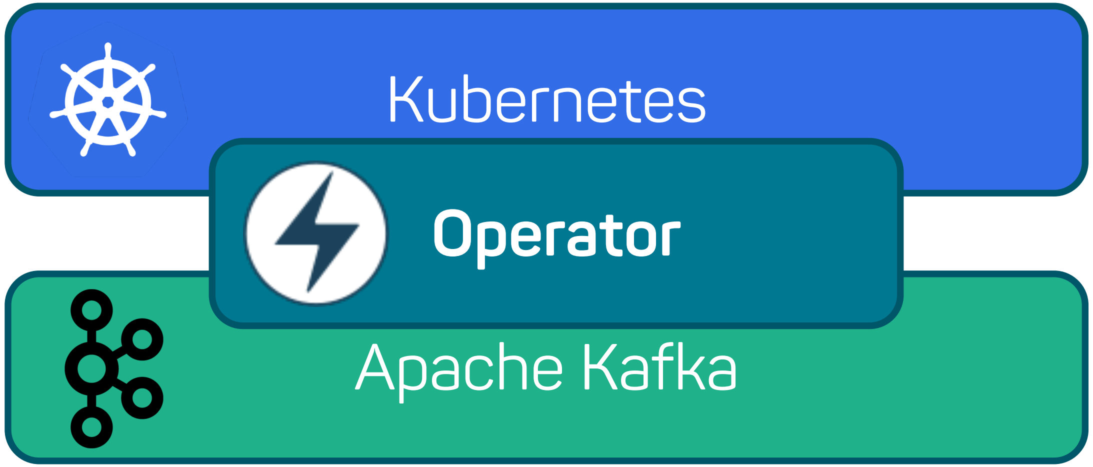
But Day 2 Operations?

- Backups?
- Upscaling? Reshuffle Data?
- Downscaling? Without Dataloss?
- Healing? Restore Backups?
- Configuration? Tedious Templating?

If only we could automate this!

In a Kubernetes native way!

Operators



Operators

- Human Operational Software
  - Custom Software
- Kubernetes Native:
  - CustomResourceDefinition
- So lets write one:
  - High Level

CustomResourceDefinition

- Defines a new API
- Seamless integration with existing API
- Kubectl support

```
apiVersion: apiextensions.k8s.io/v1beta1
kind: CustomResourceDefinition
metadata:
name: crontabs.stable.example.com
spec:
group: stable.example.com
version: v1
scope: Namespaced
names:
plural: crontabs
singular: crontab
kind: CronTab
shortNames:
- ct
validation:
# openAPIV3Schema is the schema for validating
custom objects.
```

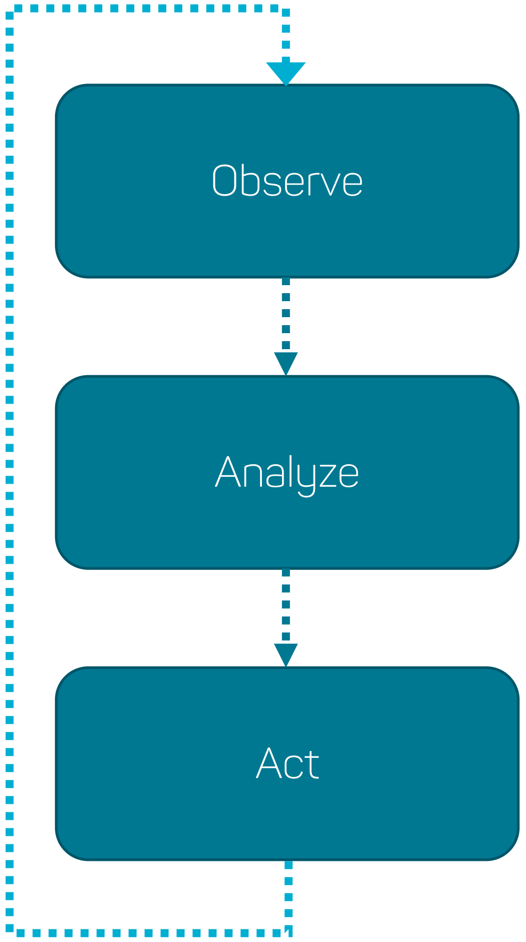
CustomResourceDefinition

- Actual Object in new API
- No functionality.

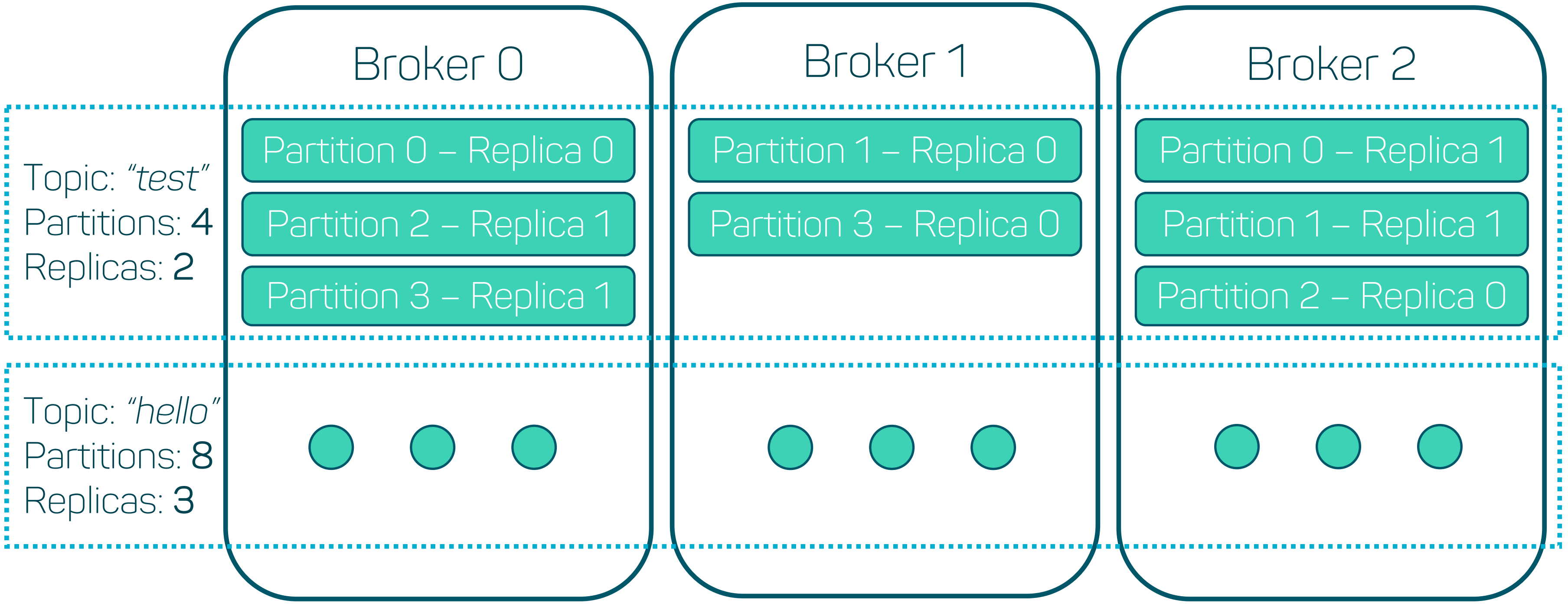
```
apiVersion: "stable.example.com/v1"
kind: CronTab
metadata:
name: my-new-cron-object
spec:
cronSpec: "* * * * */5"
image: my-awesome-cron-image
replicas: 5
```

Control Loop

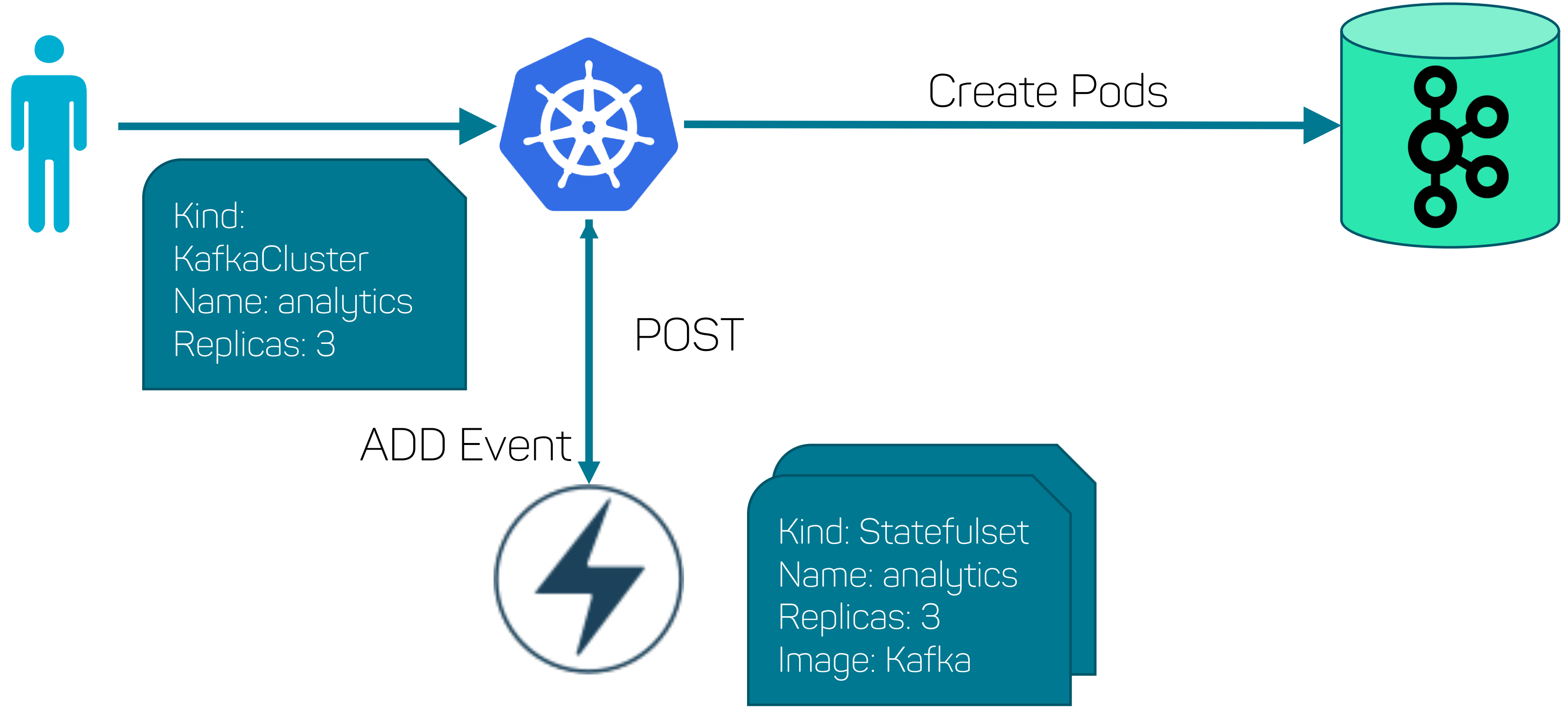
- Operator create WATCH on CR Objects
- Analyze difference Actual vs Desired State
- Act on changes



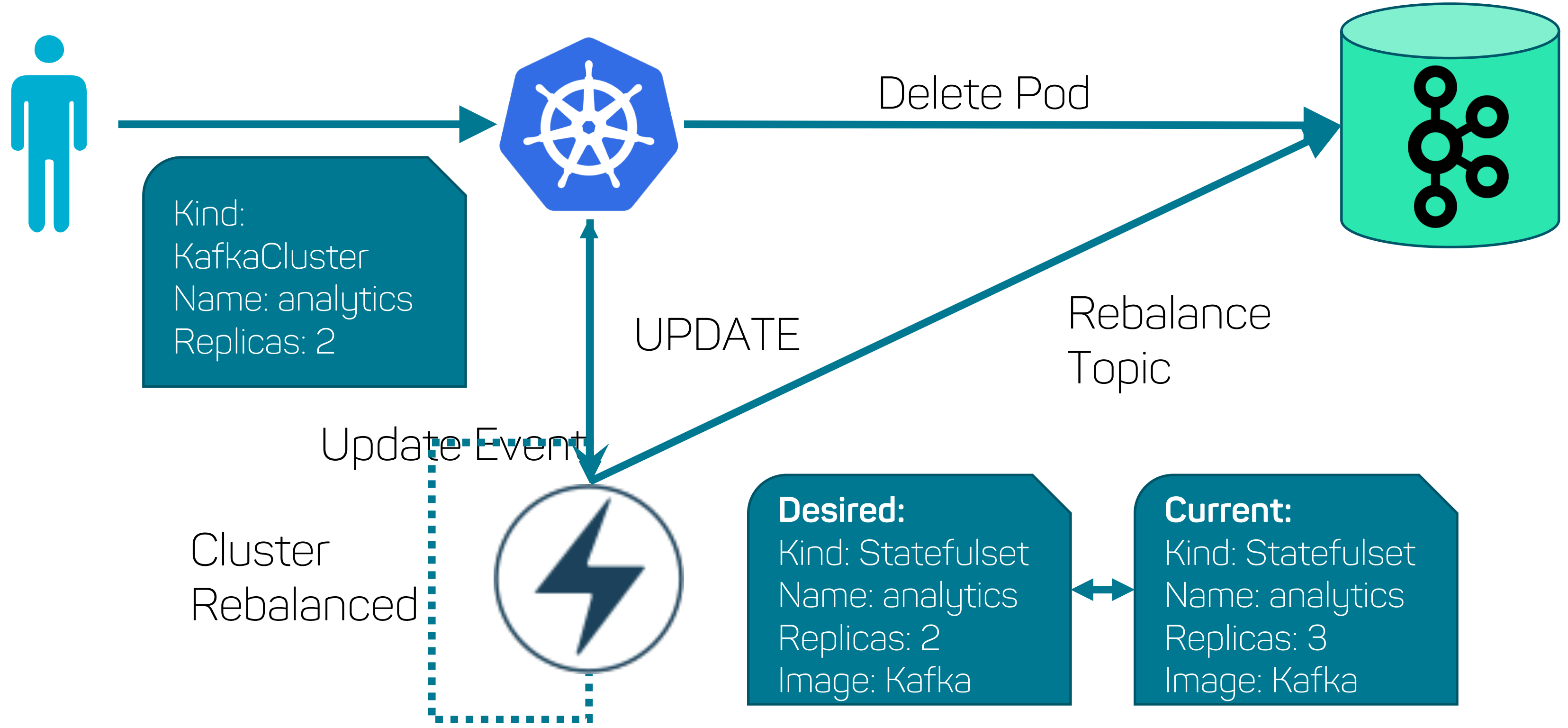
Kafka Basics



Create Cluster



Downsize Cluster



Kind: KafkaCluster
Name: analytics
Replicas: 2

Desired:
Kind: Statefulset
Name: analytics
Replicas: 2
Image: Kafka

Current:
Kind: Statefulset
Name: analytics
Replicas: 3
Image: Kafka

Rebalance Topics with Hot Partitions

Other Operators

Elasticsearch

Postgres

Tensorflow

Take a step back

- Are we reinventing the Wheel?
- Helm?
- Mesos Frameworks?
- Nomad Custom Scheduler?
- Docker Swarm Plugins?

Operators vs Helm vs Controller

- Helm itself a Operator (somewhat, working on it https://github.com/kubernetes/helm/issues/3089))
- Controllers
  - Operator = Controller + CRD
  - Operator = External Software
  - Controller = Internal
- Only do operators if you **cant** solve it with Helm.

Code!

- Create API Spec
- Generate some Objects needed by Informer etc (Since 1.8)
  - See: https://blog.openshift.com/kubernetes-deep-dive-code-generation-customresources/ (Excellent, by sttts)
- Generator Controller
  - Informer
- Main

Best Practice Operators

- Microservices, single Deployment
- Stateless, use CRD for States
- Operations should be Idempotent
- Leverage K8S Objects as most as possible
- CRD should be versioned, backwards compatible

Questions? Discuss!

Kubernetes Extensibility

- Custom Resource Definitions
- API Aggregation
- Initializers
- Scheduler Extenders
- Custom Schedulers
- Flex Volumes
- Cloud Provider
- CRI & CNI
- Admission Webhook

Comparison

Task	Mesos	Kuberentes
Custom Resource Placement	Write a framework	Write a custom scheduler
Special resource init	Write a framework	Initializer
API access	Every Framework has its own API	Unified API
Special lifecyle	Write a framework	Kubernetes Operator
Custom execution	Write a framework + executioner	CRI Interface + Scheduler

Stay connected

Address
codecentric AG
Hochstraße 11
42697 Solingen

Contact Info
E-Mail: info@codecentric.de
www.codecentric.de

Telephone
Telefon: +49 (0) 212. 23 36 28 0
Telefax: +49 (0) 212.23 36 28 79

codecentric

Hello, World!
CONTAINERISIERUNG KNOWLEDGE SHARING
NEARSHORING AGILITÄT CODING CULTURE
BIG DATA
PERFORMANCE-OPTIMIERUNG MICROSERVICES
JAVA ARCHITEKTUR PERFORMANCE CLOUD