How to get started with Operators for Kubernetes

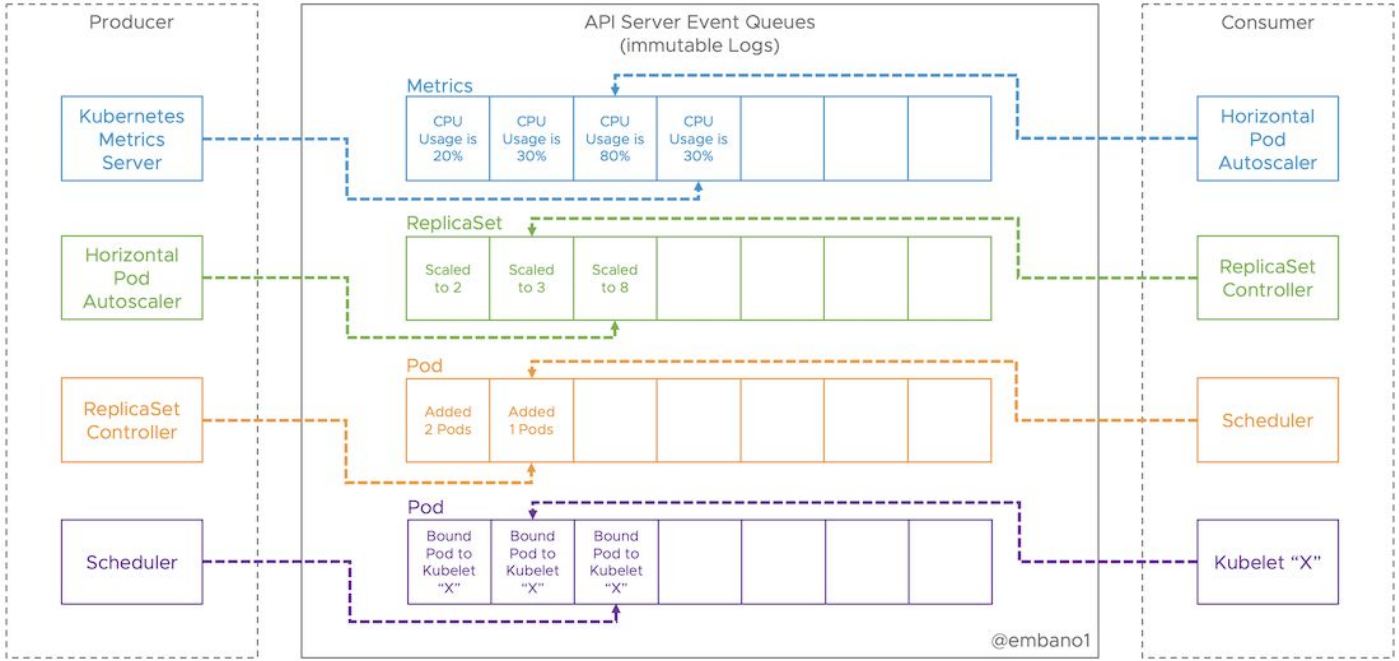
Max Körbächer

Max Körbächer

- Happy Kuberneting anno 2016
- Kubernetes Release Team since K8s v1.17
- Manager Cloud Native Engineering @ Storm Reply

The way of how Kubernetes works

Kubernetes is event driven



Source: Michael Gasch

Running simple, stateless deployments

Simple & Stateless?

- no long-run persistence needed
- can be shutdown “any time”
- can run multiple time without issues
- can run in different versions without issues

“There are two ways of constructing a software design: One way is to make it so simple that there are obviously no deficiencies, and the other way is to make it so complicated that there are no obvious deficiencies. The first method is far more difficult.” - C.A.R. Hoare

Running complex, stateful deployments

aka DON’T TOUCH!

- Need stable persistency
- Can’t deal well with unpredictable issues
- Long(er) startup time
- Not flexible in scheduling

What we are talking about?

- Databases
- Legacy migrations
- Architecture designs which can’t be event driven, stateless for some reasons
- Applications with many or heavy dependencies

Such applications in a dynamic environment like K8s causes additional operational effort for Day2 and later.

Operators!

A Kubernetes Operator abstracts the deployment of diverse and state sensitive applications by including domain-specific knowledge.

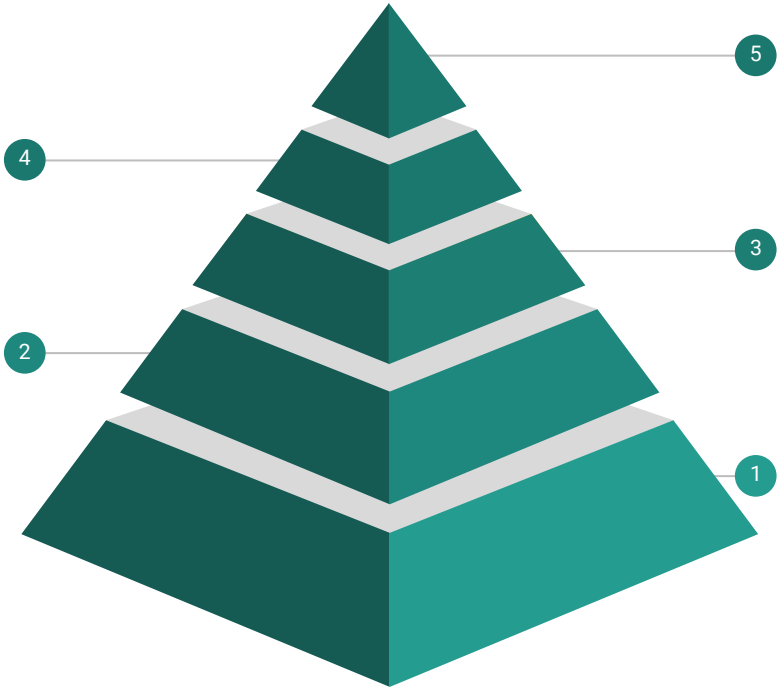
- Lifecycle Management
- Configuration
- Updates
- Handle Failures

Operator are the result of translating operative knowledge, and developer-specific knowledge of an application into own software.

Operator utilize Custom Resource Definitions (CRD) to extend natively the K8s resources and APIs.

Long story short: software runs software

Capability Levels of Operator



Auto Pilot
Horizontal scaling, auto config tuning, anomaly detection, scheduling optimization

Deep Insights
Metrics, alerts & log processing and workload analysis

Full Lifecycle
App lifecycle (automated canary deployments, rollbacks by failure, and so on), storage lifecycle (backup, failure recovery)

Seamless Upgrades
Patch and minor release upgrades

Basic Install
Automated application provisioning and configuration management

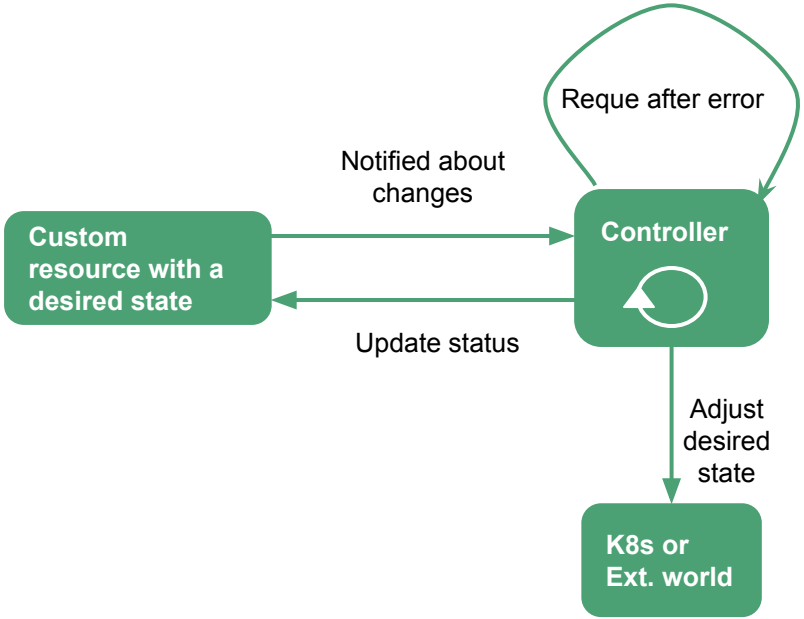
How does an Operator work? - Control(ler)/-loop

Controllers act on core resources like deployments, statefulsets or service, but they also work with custom resources.

Controllers are implemented by a control loop:

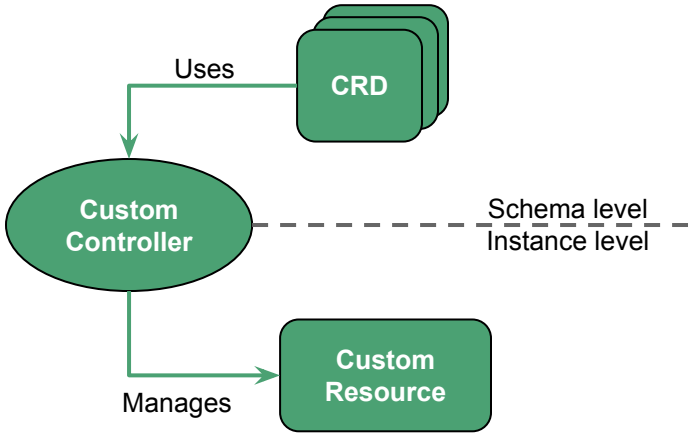
1. Check state of resources, ideally by events
2. Change state of the object or ext. resource
3. Update status of the resource

Independent of the complexity of your controller, in the end you will do always these steps.



Important to note, but no time to go into depths:
Informers, Work Queues & Events

How does an Operator work? - Operator



An operator sum up the operational knowledge of an SRE and their domain specific knowledge, to automate the common tasks.

A Custom Resource Definition (CRD) captures the domain knowledge, while a custom resource represents this on an instance.

Therefore a custom controller manages the custom resource by e.g. trigger a backup or start a new pod with a new software version.

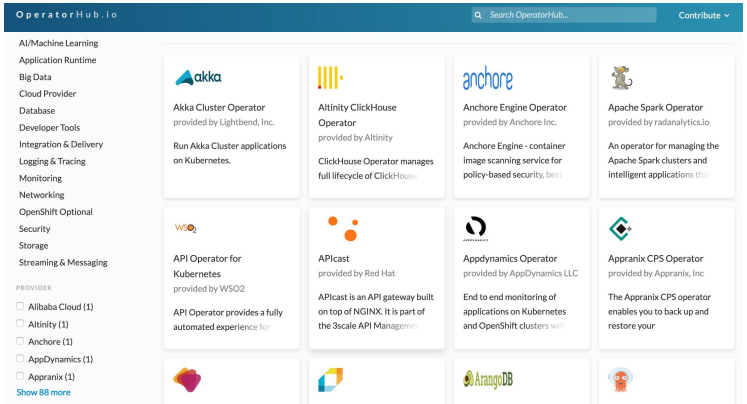
Some options to get started

Declarative/
Opinionated

KUDO → Meta-controller → Operator Framework/ SDK → Kubebuilder

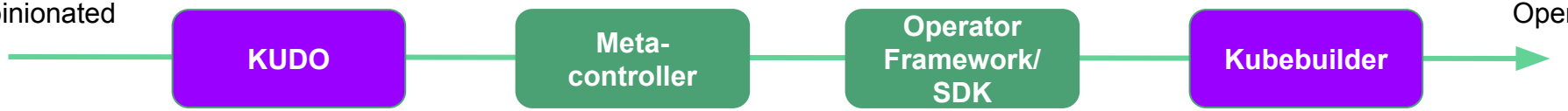
Flexible/
Open

Or for the lazy people:



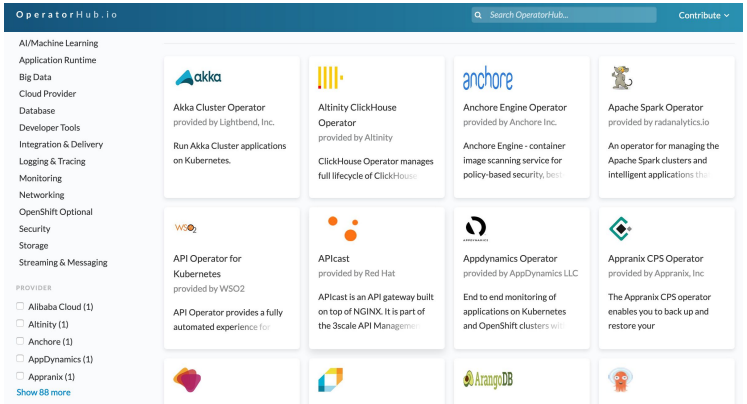
Some options to get started

Declarative/
Opinionated

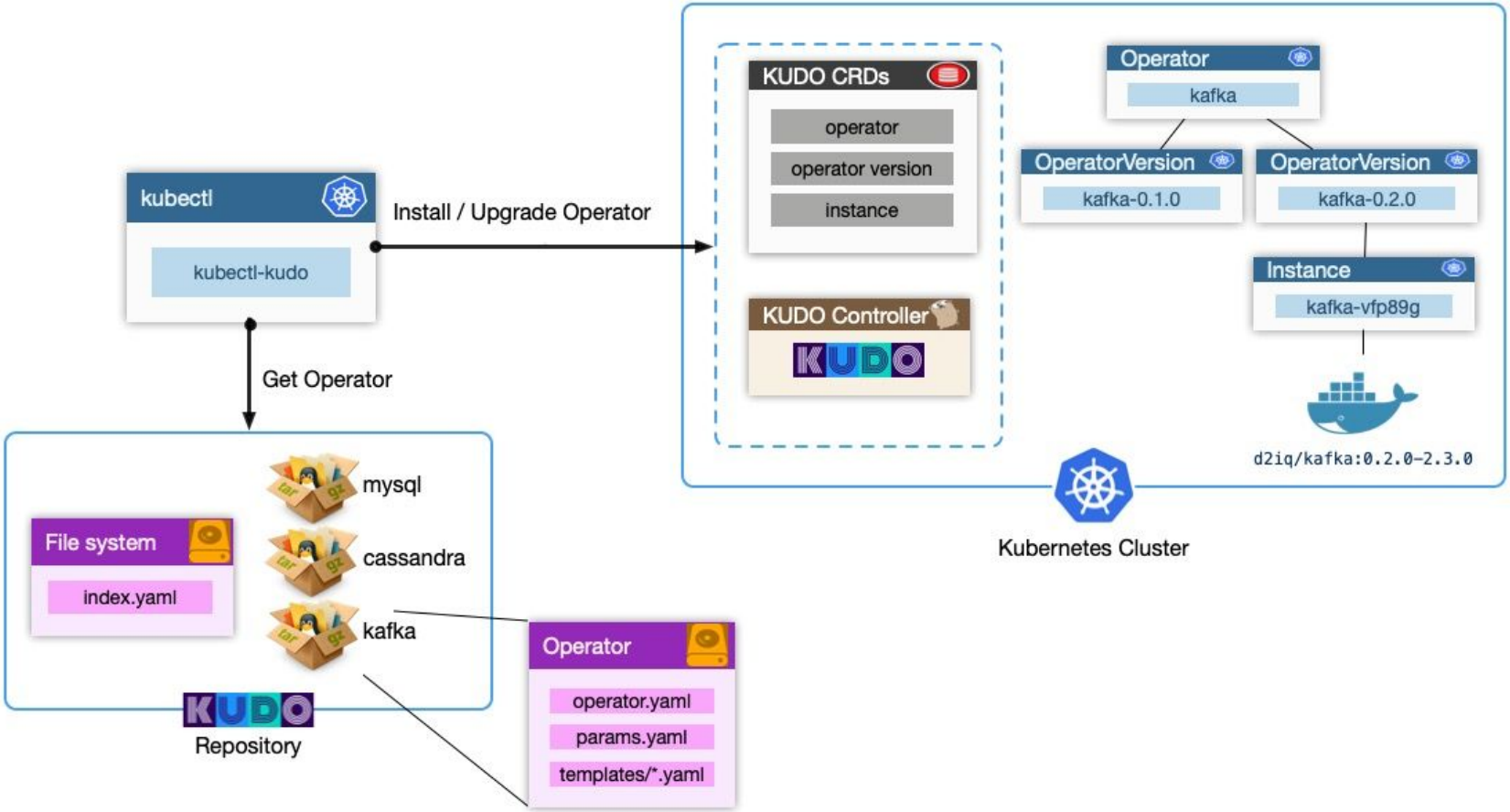


Flexible/
Open

Or for the lazy people:




KUDO


kubectl
kubectl-kudo
Install / Upgrade Operator
Get Operator
KUDO CRDs
operator
operator version
instance
KUDO Controller
Operator
kafka
OperatorVersion
kafka-0.1.0
OperatorVersion
kafka-0.2.0
Instance
kafka-vfp89g
d2iq/kafka:0.2.0-2.3.0
Kubernetes Cluster
File system
index.yaml
mysql
cassandra
kafka
Repository
Operator
operator.yaml
params.yaml
templates/*.yaml

kube
builder

Q&A