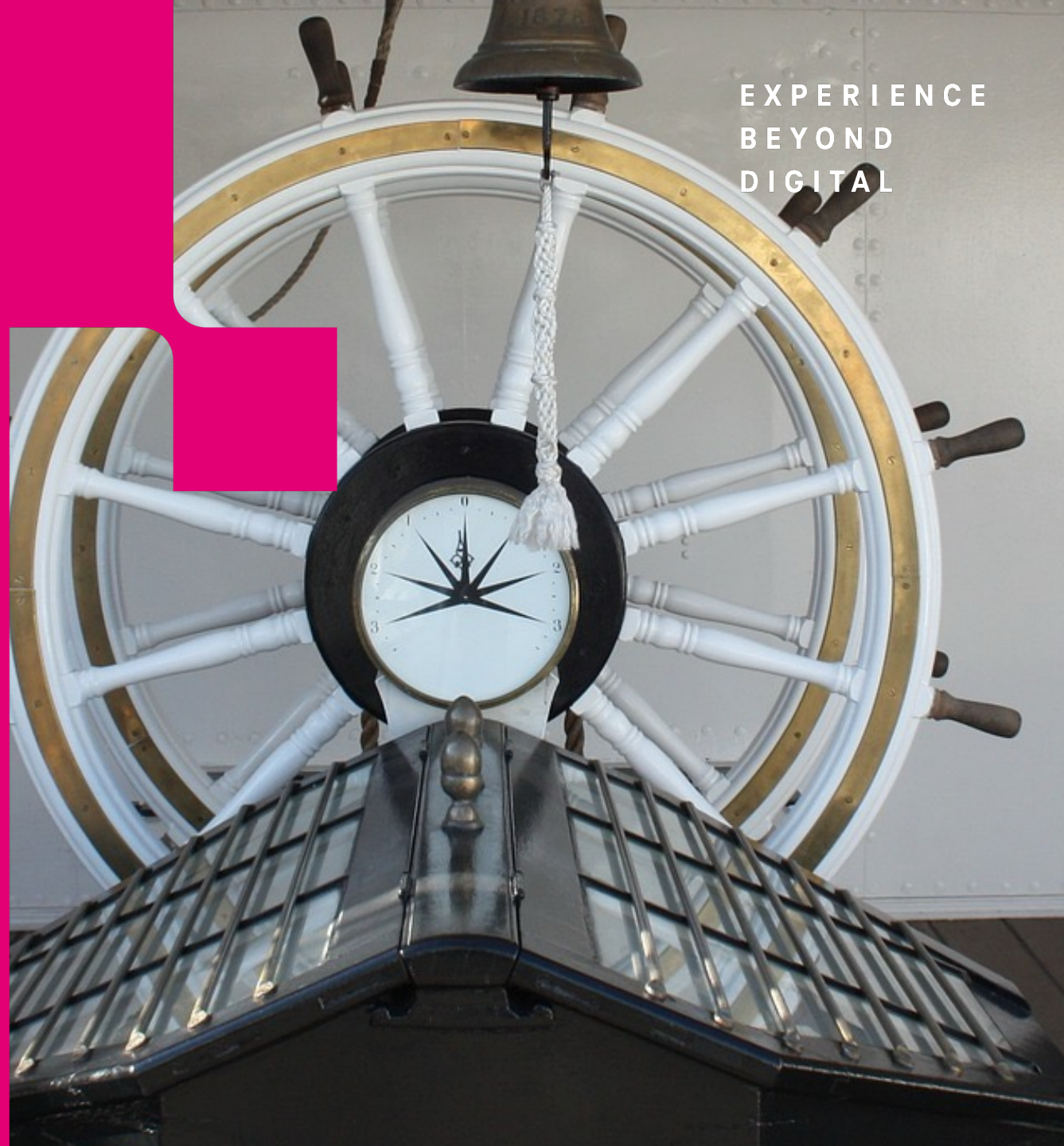
Telekom MMS

Open Source Project

CoreMedia Content Cloud k8s Operator



EXPERIENCE
BEYOND
DIGITAL

Agenda

- Who am I?
- What is a Kubernetes Operator?
- Challenges Deploying CoreMedia in Kubernetes
- Demo
- What does cmcc-operator provide?
- Benefits of using cmcc-operator
- Why not a Helm Chart?
- Summary & Questions & Contact

Who am I?

- **Stefan Bethke**
- **Principal Software Architect at T-Systems Multimedia Solutions, a long-time CoreMedia partner**
- **stefan.bethke@telekom.de**
- **About 20 years of experience designing and developing CoreMedia-based projects, including extensive complex customizations and integrations for customers both within Deutsche Telekom and outside**

What is a Kubernetes Operator?

A Kubernetes operator is a method of packaging, deploying, and managing a Kubernetes application. A Kubernetes application is both deployed on Kubernetes and managed using the Kubernetes API (application programming interface) and kubectl tooling.

A Kubernetes operator is an application-specific controller that extends the functionality of the Kubernetes API to create, configure, and manage instances of complex applications on behalf of a Kubernetes user.
It builds upon the basic Kubernetes resource and controller concepts, but includes domain or application-specific knowledge to automate the entire life cycle of the software it manages.

In Kubernetes, controllers of the control plane implement control loops that repeatedly compare the desired state of the cluster to its actual state. If the cluster's actual state doesn’t match the desired state, then the controller takes action to fix the problem.

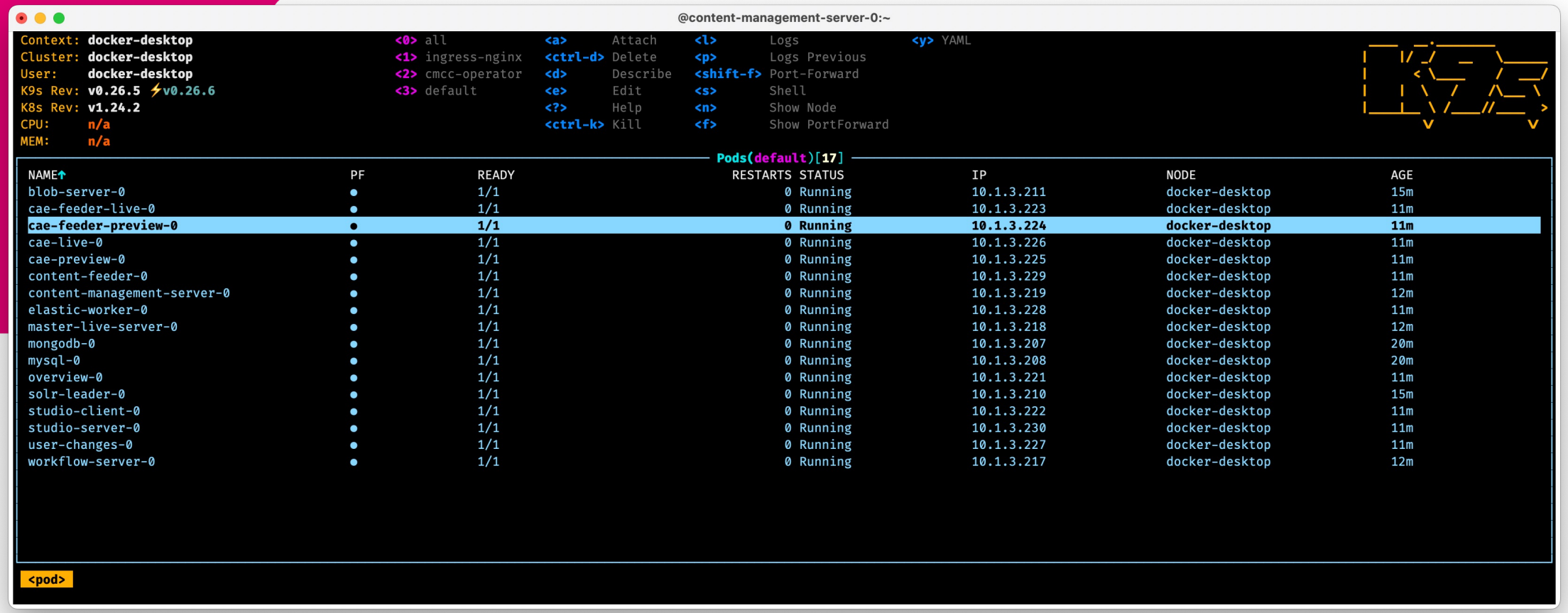
An operator is a custom Kubernetes controller that uses custom resources (CR) to manage applications and their components. High-level configuration and settings are provided by the user within a CR. The Kubernetes operator translates the high-level directives into the low level actions, based on best practices embedded within the operator’s logic.

https://www.redhat.com/en/topics/containers/what-is-a-kubernetes-operator

Challenges Deploying CoreMedia in Kubernetes

- **Large number of servers (20+)**
- **Complex configuration of pods**
- Many dependencies
- Many settings required
  - Env vars for Spring Boot properties
  - Many of almost the same settings required in different pods, but not identical
- Many system architecture options available
- Initializing a development system can be complicated and time consuming
- Running tasks (like import, export, re-index) can be complicated
- **Many additional resources**
- Secrets, volumes, ingresses, services

Demo


@content-management-server-0:~
Context: docker-desktop
Cluster: docker-desktop
User: docker-desktop
K9s Rev: v0.26.5 ⚡v0.26.6
K8s Rev: v1.24.2
CPU: n/a
MEM: n/a
<0> all
<1> ingress-nginx
<2> cmcc-operator
<3> default
<a> Attach
<ctrl-d> Delete
<d> Describe
<e> Edit
<?> Help
<ctrl-k> Kill
<l> Logs
<p> Logs Previous
<shift-f> Port-Forward
<s> Shell
<n> Show Node
<f> Show PortForward
<y> YAML
Pods(default)[17]
NAME↑ PF READY RESTARTS STATUS IP NODE AGE
blob-server-0 ● 1/1 0 Running 10.1.3.211 docker-desktop 15m
cae-feeder-live-0 ● 1/1 0 Running 10.1.3.223 docker-desktop 11m
cae-feeder-preview-0 ● 1/1 0 Running 10.1.3.224 docker-desktop 11m
cae-live-0 ● 1/1 0 Running 10.1.3.226 docker-desktop 11m
cae-preview-0 ● 1/1 0 Running 10.1.3.225 docker-desktop 11m
content-feeder-0 ● 1/1 0 Running 10.1.3.229 docker-desktop 11m
content-management-server-0 ● 1/1 0 Running 10.1.3.219 docker-desktop 12m
elastic-worker-0 ● 1/1 0 Running 10.1.3.228 docker-desktop 11m
master-live-server-0 ● 1/1 0 Running 10.1.3.218 docker-desktop 12m
mongodb-0 ● 1/1 0 Running 10.1.3.207 docker-desktop 20m
mysql-0 ● 1/1 0 Running 10.1.3.208 docker-desktop 20m
overview-0 ● 1/1 0 Running 10.1.3.221 docker-desktop 11m
solr-leader-0 ● 1/1 0 Running 10.1.3.210 docker-desktop 15m
studio-client-0 ● 1/1 0 Running 10.1.3.222 docker-desktop 11m
studio-server-0 ● 1/1 0 Running 10.1.3.230 docker-desktop 11m
user-changes-0 ● 1/1 0 Running 10.1.3.227 docker-desktop 11m
workflow-server-0 ● 1/1 0 Running 10.1.3.217 docker-desktop 12m
<pod>

What does cmcc-operator provide?

- Create, update, and destroy a complete CoreMedia environment in a Kubernetes cluster
- Optionally deploy MariaDB and MongoDB alongside CoreMedia for a quick development setup
- Configure against hosted databases
- Use a Custom Resource or a Config Map for configuration parameters
- Initial setup in a multi-step sequence
- Import content during setup, and as part of updates
- Automatically configure scaled CAEs, RLS, Solrs
- Build Ingresses automatically from site mapping table
- Uses secrets for all logins; optionally create random secrets (DBs, UAPI)
- Any cluster distribution (Docker Desktop, k3d, AKS, ...)

Benefits of using cmcc-operator

- **Quickly create development or testing environments**
- **Reliably create and update production environment**
- **Using rolling updates for components, allowing zero-downtime upgrades**
- **Straightforward configuration with sensible defaults for most parameters**
- **Customizable to a large degree**
- Additional components can be added easily
- **Extensible**
- Implemented in Java using Java Operator Toolkit

Why not a Helm Chart?

- Helm works well if you can declaratively configure a limited number of parameters
- CoreMedia has too many optional data structures and dependencies
- Programmatically creating YAML files using the Go template engine is a bad match (remember HTML parsing with Regex, or producing XML with JSP?)
- Helm cannot dynamically control things
- You cannot easily talk to components from your Helm chart

Summary & Questions & Contact

- Kubernetes Operators are the right choice for complex applications
- cmcc-operator is designed to take advantage of these possibilities
- Production-ready, customizable and extensible

- https://github.com/t-systems-MMS/cmcc-operator
- Stefan Bethke <stefan.bethke@telekom.de> (and Xing and LinkedIn and...)