Institut für Parallele und Verteilte Systeme
Abteilung Anwendersoftware
Universität Stuttgart

# Master Thesis

## Orchestrating data governance workloads as stateful services in cloud environments using Kubernetes Operator Framework

***Bearbeiter:*** ***< Student >***
***Prüfer:*** *Bernhard Mitschang*
***Betreuer:*** *Cataldo Mega*

**Background:** Data is a strategic asset to every enterprise and is therefore subject to data governance. The aim for data governance is to strategically managed business relevant data through its entire lifecycle from creation to disposition. The technical aspect of data governance are realized as set of policies and practices implemented to support business processes, corporate policies and regulatory compliance.

Once captured all data of all types must be classified, categorized and securely stored such to meet high legal and regulatory requirements. These requirements imply the existence of an enterprise wide IT infrastructure of secure data repositories that facilitates data sharing between different user, groups or applications. Sharing data is always according to their sensitivity, relevance and jurisdictional restrictions. Given the huge amount of data and the need to interconnect and share it with potentially any other company on the globe, the thought is to align current IT with today's cloud technology to cut cost down. The vision is a sort of a hybrid multi-cloud approach as an alternative to a closed legacy infrastructure platform. In this context we call unstructured data = 'content' in order to distinguish it from metadata, which is usually structured. Therefore unstructured data stores are called from now on 'content repositories'.

In the past companies have stored their 'content' using home grown monolithic data governance application stacks on top of distributed content repositories. In the past they were called Enterprise Content Management Systems (ECM). ECM system are typically operated on bare metal or at most in a virtualized IT infrastructure. And though they have and still offer reliable and performant content services they do lack the flexibility to support CI/CD i.e. automated continuous integration / continuous deployment in a fast, efficient and cost effective way. Based on our analysis we think the major problem derives from the fact that the existing ECM systems are still using an outdated client server architecture' model and legacy IT infrastructure and as such they are not able to exploit new cloud technologies nor be able to support CI/CD models. As a consequence, if companies want to exploit the 'Cloud' they must entertain the effort to 'transform' their current approach to be able to benefit from moving into the cloud. Aside from the organizational and cultural changes required by a cloud transformation process, the technical challenge is to refactor and enhance the application solutions design and then develop a cloud deployment model before rolling out a cloud specific implementation build. In essence what this means, is to enforce a paradigm shift from 'client server' to a 'cloud computing' architecture model. Our expectation is to benefit from the exploitation of the economies-of-scale, the advantage of the inherent multi-tenancy capability and the ubiquity of the service offering in an increasingly secure and trusted cloud environment.

**Key words:** Kubernetes, stateful services, cloud native databases, elastic topologies, scale-up and scale down

## Master Thesis Content

**Previous work:**

In previous master thesis, see [1] and [2], a first refactoring step was taken. In there, the monolithic design of a typical content repository application stack was decomposed into smaller self-contained components and encapsulated into Docker container images. For the set of related stateless and stateful services a cloud deployment model was developed but the implementation covered only the set of stateless services. No deployment model was developed for the set of stateful services due to time constraints and the complexity exposed by the Kubernetes stateful services framework. We learned that for database stateful services database specific expert knowledge is required for being able to develop a custom DB2 Kubernetes operator and pre-requisite supporting artifacts.

This MA-thesis will focus on the handling of stateful services in Kubernetes. More specifically, developing the deployment models of the catalog database, the content repository and the persistent storage required by ECM services. And we want all of the above orchestrated as stateful services by the Kubernetes runtime system.

Figure 1 one below shows a typical deployment scenario of a legacy content management application stack. All core components are deployed on traditional on-premise IT environments. Fundamental for a production system is the ability to guarantee consistency, high availability (HA) and disaster recovery (DR). System requirements that are specified in a set of service level agreements.

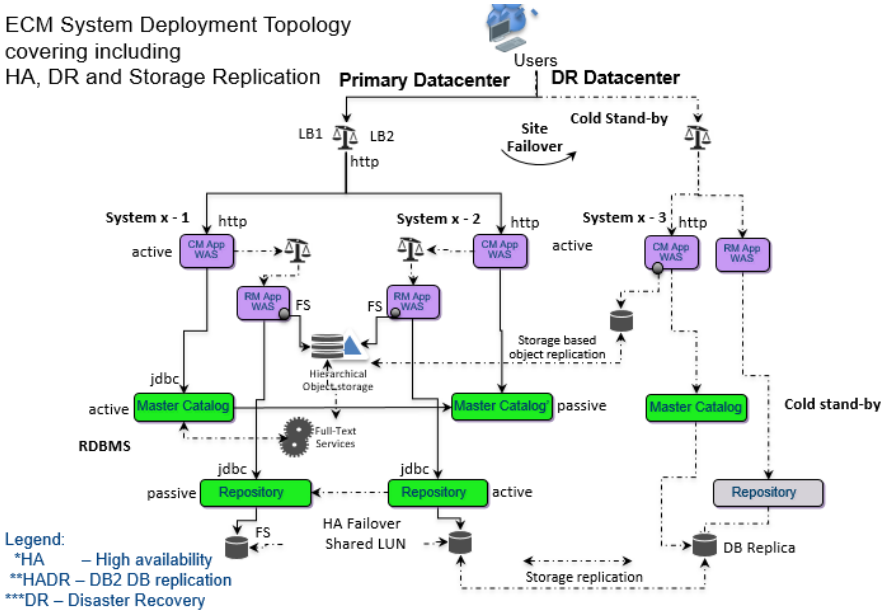


Figure: 1 Legacy ECM system deployment model including HA, DR and storage replication

The boxes in purple in figure 1 represent the set of stateless services offered by the content repository system. And we have already covered them in the previous work. The boxes in green and the storage systems is what have to be transformed into Kubernetes stateful services using data base specific skills.

The technical details of the traditional ECM deployment model can be gathered from [4] the IBM Redbook “Content Manager Backup/Recovery and High Availability: Strategies, Options, and Procedures”. This IBM Redbook contains the expert knowledge that can be used to develop a cloud ready deployment model using the required Kubernetes stateful services artefacts.

**Goals:**

The aim for this work is to design and deploy an equivalent or better content services application stack running on a Kubernetes cluster. With a good understanding of the technical details required, a set of stateful database services should be developed. In simple terms, we want to build stateful service wrappers around an IBM DB2 database system using the Kubernetes stateful services SDK. The design goal is that

the data base service must be able to i) grow up to scale horizontally over a number of nodes in an elastic fashion, ii) survive disasters, and iii) be available everywhere.

For this to happen, first a comparison between traditional databases and cloud native counter parts should be performed [5]. Find out if and how both satisfy the above given requirements i- iii. In a second step understand the mechanism Kubernetes uses, to elastically scale up and down the overall deployment topology based on system workload load and resource component state. Finally use the previous MA work results and expand the available deployment model complementing it with the set of stateful database services and put them under control of the Kubernetes orchestrator.

**Approach:**

**Part I: Problem analysis and concept design relative to stateful services deployed and managed in clouds (2m)**

To start with:

- Familiarize yourself with the basic architecture design of the content repository application stack used in the previous work. Focus on the backend and related stateful services.
  Read and understand MA [1],[2], and [3] listed in the reference section, but dep dive only MA [1] and [2] as a starting point.
- Familiarize yourself with the basic architecture design of Kubernetes/Docker. Understand what virtualized and containerized cloud environments are and the aspects of cloud orchestration technology, the dynamic deployment capabilities as well the service life cycle management that the Kubernetes eco-system offers. I.e. operators, control loop, custom resource definition (CRD)

  ***Note: Use the self-study tutorial videos listed below.***

- Deep dive into the Kubernetes stateful services architecture design. Learn about stateful sets, operator pattern, persistent volumes and how to use YAML to declaratively define services templates used for an automated deployment on to a Kubernetes cluster.
- At this point look at the enterprise content repository components: like the catalog database, the content repositories and the persistent store and sketch out a Kubernetes specific architecture design that use the Kubernetes stateful services SDK to design required custom operators and custom resource definitions for the above mentioned components. Then develop the deployment scripts required to setup and rollout the set of stateful content repository services.

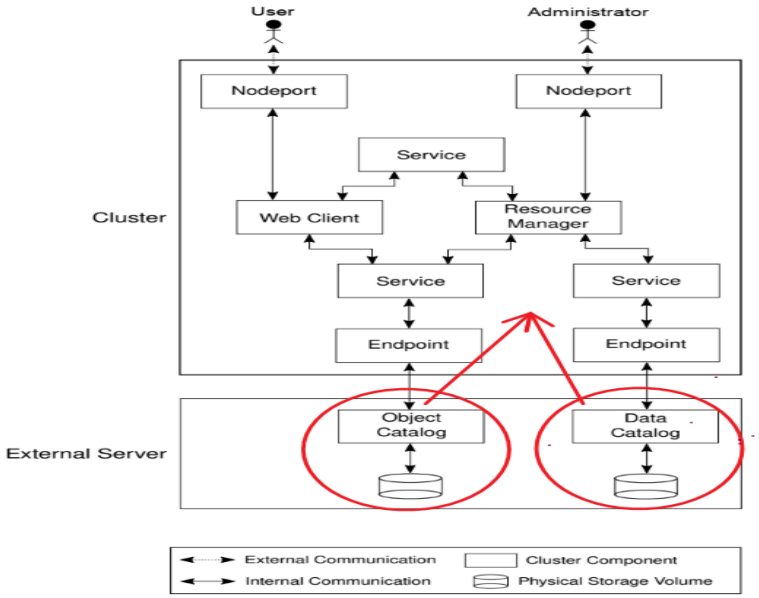


Figure 2: Phase 1 content repository prototype using external services.

- To round up the architectural work compare traditional databases vs cloud native data bases. Use IBM DB2 and PostgreSQL as legacy RDBMS systems and compare them against CockroachDB and Google F1/Spanner so called cloud native databases. Explain the key differences and document the pros and cons. Then consolidate the documentation and present your intermediate results.

**Part II: Prototype implementation of the DB2 stateful database services orchestrated and managed by Kubernetes (2m)**

Use the current prototype system. It provides a basic content management application stack with its catalog database and content repository (see [1], and [2]). In this context, IBM DB2 is used as the legacy database system. Use DB2 and the above mentioned expert knowledge source to develop a custom DB2 operator. In addition choose an available cluster file system to persist database data and content. (NFS, Ceph or others). Then

- Use the ECM prototype shown in figure 2 above and enhance and extend its deployment model as designed in your work of part I.
- Define a Kubernetes service consisting of multiple pods running multiple containers each of which running an ECM application component. Something similar to what is shown in figure2.
- Perform a number of tests that shows your approach is working as expected.
  - Your data base service scales horizontally over a number of nodes in an elastic fashion.
  - Your service survives a node failure
  - Your services is available from everywhere in the cluster.
- Consolidate the documentation so far and present your intermediate results

**Part III: (2m)**

- Evaluate the results and consolidate documentation you have developed so far
- Explain the new cloud' delivery and the enhanced service offering model.
  Show the benefits of a possible successful content repository cloud transformation strategy.
- Final presentation and explanation if the final results

**Outcome and results:** Show how an enhanced legacy ECM system deployment model can be used to roll-out in an automated fashion a set of stateful content management services in a virtualized and containerized cloud environment using open source technologies based on Kubernetes/Docker. For this thesis you can use the IPVS-AS Lab on a pre-provisioned OpenStack cluster of CentOS VMs and an ECM prototype system developed using Kubernetes / Docker cloud technologies running on a Kubernetes cluster. As an alternative you can use a IBM / RedHat sponsored cloud sandbox developing the prototype on a RedHat Openshift / Kubernetes environment.

**References:**
**Previous work:**
Familiarize yourself with the current ECM system architecture design and its context.


1. C. Trybek. “Investigating the Orchestration of Containerized Enterprise Content Management Workloads in Cloud Environments Using Open Source Cloud Technology Based on Kubernetes and Docker”. Master Thesis. Germany: University of Stuttgart, September 2021.
2. P. Hagemann. “Evaluating dynamic load balancing of ECM workload pattern employed in cloud environments managed by a Kubernetes/Docker eco-system”. Master Thesis. Germany: University of Stuttgart, September 2021,
3. G. Shao. “About the Design Changes Required for Enabling ECM Systems to Exploit Cloud Technology”. Master Thesis. Germany: University of Stuttgart, December 2020.

4. Expert knowledge: Content Manager Backup/Recovery and High Availability: Strategies, Options, and Procedures Link->
5. Cloud native alternatives: What's Really New with NewSQL? Link

**Tutorial Videos:**

Please perform a self-study on Docker, Kubernetes and the Kubernetes stateful services framework using the following tutorial videos:

- Docker tutorial: https://www.youtube.com/watch?v=3c-iBn73dDE
- Kubernetes Intro: - https://www.youtube.com/watch?v=s_o8dwzRlu4
- Kubernetes architecture: https://www.youtube.com/watch?v=aSrqRSk43lY
- Kubernetes StatefulSets: https://www.youtube.com/watch?v=pPQKAR1pA9U
- Kubernetes persistent volumes: https://www.youtube.com/watch?v=0swOh5C3OVM
- Kubernetes operators - https://www.youtube.com/watch?v=i9V4oCa5f9I

**Related work:**


- Dynamic cloud service topology adaption for minimizing resources while meeting performance goals Mega, C; Waizenegger, T. ; Lebutsch, D. ; Schleipen, S. ; Barney, J.M. In IBM Journal of Research and Development Issue 2 • Date March-April 2014
- Lange, Christoph: Dynamic Scaling of Enterprise Cloud-Applications Based on Measured Performance Baselines, Masterarbeit Nr. 1, 2014 University of Stuttgart
- Ritter T., Mitschang B., Mega C. (2012) Dynamic Provisioning of System Topologies in the Cloud. In: Poler R., Doumeingts G., Katzy B., Chalmeta R. (eds) Enterprise Interoperability V. Proceedings of the I-ESA Conferences, vol 5. Springer, London. https://doi.org/10.1007/978-1-4471-2819-9_34
- Fritz, Florian: Maximization of resource utilization through dynamic provisioning and deprovisioning in the cloud, Diplomarbeit Nr. 3078, 2011.
- Börner, Andreas: Orchestration and Provisioning of Dynamic System Topologies, Diplomarbeit Nr. 41, 2011.
- Frank Wagner, PhD Thesis - University of Stuttgart - A Virtualization Approach to Scalable Enterprise Content Management