Proposal

Master of Science

Implementation of an Enterprise Service Bus (ESB) with OpenShift und Camel

by

Ing. Thomas Herzog B.Sc

Supervisor (FH):	DI (FH) Peter Kulczycki
Supervisor (Gepardec):	Dr. Erhard Siegl
External company	Gepardec
Handed in:	December 30, 2017

1. Statement of the problem

Large enterprises work with several independent applications and systems which have the need to interexchange data. The data is represented in several formats and versions, which increases the complexity of the communication between the applications and systems. Another challenges are the configuration management and the maintenance of the integrations of the applications and systems into each other, which affects the whole application life cycle.

The concepts of DevOps, PaaS, IaaS and microservice architecture differ to the classic development life cycle, release management and a server infrastructure where DevOps brings development and IT operations together and the classic server infrastructure will be replaced by Platform as a Service (PaaS) or Infrastructure as a Service (IaaS) or more likely both.

An Enterprise Service Bus (ESB) is an enterprise pattern which in a large enterprises is often used to connect applications and systems together. There are several frameworks out there such as JBoss Fuse [1], which is based on a JBoss application server, which helps to design, implement and run an ESB application. Openshift is a open source cloud system where Gepardec assumes it could be used to realize an ESB application or at least several aspect of it. The idea is to split a monolithic ESB application into microservices and host them in an Openshift cloud, where Openshift provides features for monitoring, deploying, scaling and securing microservices.

The following questions shall be answered by this thesis:

1. How can components, internal and external microservices be integrated in an Openshift cloud with as little configuration as possible ?
2. How can different versions and stages of microservices be managed in an Openshift cloud ?
3. How to secure microservices hosted in an Openshift cloud ?

Along with using an cloud environment such as Openshift changes will have to be made in the the application life cycle and interaction between the departments and service providers, because the classic approaches will not apply anymore when microservices are hosted in an cloud environment such as Openshift.

[1] `https://developers.redhat.com/products/fuse/overview/`

2. State of the art

It seems these days a software gets classified into two categories of architectures. On the one hand there is the monolithic architecture, where the whole software is managed and organized in a single source repository, with one single complex build and deployment. On the other hand there is the very popular microservice architecture, where the software components are organized, build, test and deployed independently from each other and therefore the application can be managed more flexibly.

The upcoming of cloud systems such as Openshift have become very popular these days as they simplify the management life cycle of an infrastructure the services run on. Cloud systems such as Openshift abstract us from the actual infrastructure, what is a main factor when using Platform as a Service (PaaS) systems. Features such as load balancing, triggered deployments, isolated application processes, integrated security and templates allow developers to define an infrastructure by one or more templates which when instantiated represent our infrastructure hosted in Openshift.

The good integration of Continuous Integration/Deployment via Jenkins build server in Openshift allows to use Openshift as the platform to build, test and host services, where either a Jenkins pipeline can create, trigger and test an Openshift build and/or deployment, or Openshift gets triggered by an hook and starts an build and/or deployment itself.

This redefines the infrastructure developers are used to build, test and host their services on. There is not only an application server anymore, there is also build and deploy mechanisms, a load balancer, integrated security and monitoring. In a common development and application life cycle, there is the build and deploy separated from the service load balancing, which is separated from the monitoring. There is a very good integration of several aspects of an development and application life cycle in Openshift, where the whole build, deploy and how to host the services in the infrastructure can be defined via templates and is therefore reproducible.

Openshift should allow the integration of an ESB application, where Openshift will take part of several aspects of the development, release and application life cycle and the ESB application will have to consider the microservice architecture approach in its software architecture to be able to be well integrated in Openshift.

3. Utilization of the state of the art

It will be analyzed how an ESB application can be designed with a microservice architecture and how it can be integrated in Openshift. If possible an ESB specific aspect or functionality will be implemented with Openshift provided mechanisms. The development and release life cycle should be almost fully integrable in Openshift where a single build server environment will become obsolete, but nevertheless Jenkins is fully integrated in Openshift an can be used to configure an Openshift cluster.

This strong integration of the development and release life cycle in Openshift will require changes to be made in the management of the source code, the build definitions, tests and the application life cycle of an application hosted in an Openshift cloud. The separation of an ESB application into isolated services with their own development and release life cycle will bring the possibility of partial releases, but will also increase the complexity when it comes to the interaction between the services.

While the management of an ESB application, which is separated into isolated services, increases in complexity, the complexity of the build, release and necessary infrastructure of a single service will become less complex. Instead of spending time with a hard to maintain and hard to evolve infrastructure, more time can be spent for management and organization of the application life cycle which becomes more important and complex, when there are several independent services involved.

Another major aspect is security and how secrets are accessed by developers and applications. Openshift contains a secret management where secrets of different types such as SSH-Authentication, Basic-Authentication or simple String data can be managed. These secrets can be injected as environment variables or files into the service Docker containers to provide the secret for a service process. With Openshift the management of the security can be completely separated from the application development and referenced secrets can be represented by placeholders (e.g. environment variables) which get injected by Openshift during container start. During development time, no developer needs access to any secret anymore.

4. Prototype

An ESB application will be implemented and integrated in Openshift. The application will not only be hosted in Openshift, the application will also be build and released via Openshift mechanisms.

The actual functionality will not be the focus of the prototype, the focus will be on the integration of the organization of the source, configuration, build and release of the ESB application in the cloud system Openshift. The prototype shall be capable of answer the in section 1 raised questions and shall prove that an ESB can be well integrated in the cloud environment Openshift and that not only the application itself can be hosted in an cloud system such as Openshift, but also it's development and release life cycle can be integrated into Openshift.

5. Measurements of the prototype

There will be structural and organizational differences between the ESB prototype and an ESB implementation which is not integrated in an cloud environment such as Openshift and does not make use of the microservice architecture and DevOps. These structural differences are expected to arise in the organization of the sources, secrets, build and deploy of an ESB application in Openshift.

The build, test and deploy times of an ESB application should radically decrease because of the separation of the ESB application into microservices which get build, test and deployed independently. This can be compared to an ESB application which does not make use of the microservice architecture as well.

Therefore that the ESB application will be integrated in the cloud system Openshift, some aspects of the ESB application will be implemented with Openshift mechanisms, where it can be shown how the impact of this integration into Openshift is on the ESB application.

6. Further usage and discussion

The evaluation results will be discussed and how they may influence the decisions made by the target audience, who are responsible for the application life cycle of an ESB application. The evaluation results could also have an impact on decision makers on a management level, which can see how an enterprise application such as an ESB application behaves in a cloud system such as Openshift.

Especially how the ESB application gets integrated in the cloud environment Openshift and how may responsibilities for several aspects of an ESB have moved to Openshift gives a lot room for discussion. The integration in Openshift should provide new possibilities for managing the application and release life cycle of an ESB application.

A. Target audience

The target audience of the thesis will be especially software developers who are working on enterprise software, microservices or software which could be hosted in an cloud environment such as Openshift. Also cloud architects, software product managers and CTOs are part of the target audience because the thesis could help or inspire them to integrate their software applications in a cloud system such as Openshift.

Therefore that a key factor of DevOps is the joint work of operations and the software development, the thesis could also be interesting for persons in charge of hosting infrastructure such as service providers like IT&Tel, because the thesis could help them to understand the developer point of view when an ESB application is hosted in Openshift.

Parts of the thesis could help people to understand the interconnection between containerization (Docker), CaaS (Kubernetes) and PaaS (Openshift) and why there is a need for such technologies these days.

B. Structuring

This sections deals with the structuring and the extend of the master thesis.

1. Abstract ***(1 page)***
2. Introduction ***(2 pages)***
    2.1. Motivation
    2.2. Objectives
3. Infrastructure as Code (IaC) ***(~5 pages)***
    3.1. The need for IaC
    3.2. Core concepts
4. Containerization with Docker ***(~5 pages)***
    4.1. The need for containerization
    4.2. Core concepts
    4.3. Virtualization vs Containerization
5. Container as a Service (CaaS) with Kubernetes ***(~5 pages)***

   5.1. Need for container orchestration

   5.2. Core concepts

   5.3. VM orchestration vs container orchestration

6. PaaS with Openshift ***(~5 pages)***

   6.1. Need for application orchestration

   6.2. Core concepts

   6.3. Application orchestration vs container orchestration

7. Enterprise Service Bus (ESB) ***(~5 pages)***

   7.1. Need for an ESB

   7.2. Core concepts

   7.3. Challenges of hosting an ESB

8. Prototype ESB in Openshift ***(~10 pages)***

   8.1. Source code management

   8.2. Configuration management

   8.3. Secret management

   8.4. Build and release management

   8.5. ESB Infrastructure in Openshift

9. Prototype analysis and measurements ***(~10 pages)***

   9.1. Comparison of the Source code management

   9.2. Comparison of the Configuration management

   9.3. Comparison of the Secret management

   9.4. Comparison of the Build and release management

   9.5. Openshift Integration

10. Discussion and further work ***(~3 pages)***

11. Conclusion ***(1 page)***

12. Prospects ***(1 page)***

C. Milestones

This section deals with the planning of the work on the master thesis which is organized by milestones.

To-Date	Task	Milestone
01/20/18	Learn the ropes of docker	CI/CD in openshift
	Learn the ropes of openshift	
	Learn the ropes of CI/CD in openshift	
01/31/18	Specify application build artifacts	Abstract build/release concept
	Specify application integration in Openshift	
	Specify application release in Openshift	
02/15/18	Study/write IaC (probe chapter)	Chapter 'IaC'
02/28/18	Study/write Docker	Chapter 'Docker'
03/15/18	Study/write Kubernetes	Chapter 'CaaS - Kubernetes'
03/31/18	Study/write Openshift	Chapter 'PaaS - Openshift'
04/15/18	Study/write ESB	Chapter 'ESB'
05/15/18	Implement ESB servcies	Prototype Implementation
	Implement build for services	
	Implement release/rollback for services	
06/01/18	Write Prototype/Measurements/Discussion	Chapter 'Prototype'

D. Literature index

This section deals with the literature providing the theoretical background of the thesis. [Pic17] [Kie16] [Sam15] [Hoh03] [Inc17a] [Fow17] [Inc17b]

References

[Fow17] Martin Fowler, *martinfowler.com*, 2017.

[Hoh03] Hohpe, Gregor. and Woolf, Bobby., *Enterprise integration patterns*, Pearson Education, Inc., 75 Arlington Street, Suite 300, 2003.

[Inc17a] Docker Inc., *Docker documentation*, 2017.

[Inc17b] Red Hat Inc., *Openshift container platform 3.5 documentation*, 2017.

[Kie16] Kief Morris, *Infrastructure as code: Managing servers in the cloud*, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2016.

[Pic17] Picozzi, Stefano. and Hepburn, and Mike. OĆonnor, Noel, *Devops with openshift*, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2017.

[Sam15] Sam Newman, *Building microservices: Designing fine-grained systems*, O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472, 2015.