Technische Universität Berlin (TU Berlin)
Fachbereich IV - Elektrotechnik und Informatik

Master Thesis:

Design and Development of a Fog Service Orchestration Engine for Smart Factories

Master Thesis
from

Markus Paeschke

Supervisor: Prof. Dr.-Ing. Thomas Magedanz
Dr.-Ing. Alexander Willner
Mathias Brito

Acknowledgments

thank your supervisors

thank your colleagues

thank your family and friends

Berlin, February 11, 2017

Markus Paeschke
Trachtenbrodtstr. 32
10409 Berlin

I hereby declare that the following thesis “Design and Development of a Fog Service Orchestration Engine for Smart Factories” has been written only by the undersigned and without any assistance from third parties.

Furthermore, I confirm that no sources have been used in the preparation of this thesis other than those indicated in the thesis itself.

Berlin, February 11, 2017

Markus Paeschke

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Background and Motivation

1.2 Problem Statement

1.3 Assumptions and Scope

1.4 Objectives and Contributions

1.5 Methodology and Outline

Chapter 2

Related Work

This chapter will give an overview into the background and concepts of this thesis. In the first section the Internet of Things an related subtopics like Smart Factories and Smart Cities are considered. Cyber Physical Systems, which are important for the development of Smart Factories are also covered in this section. Virtualization in general is the main topic of the second section. First we dive into the area of Virtual Machines, followed by Container Virtualization. Both are related to each other and sharing some basic ideas. Container Orchestration as an own subsection show some possibilites of Container Virtualization. The last subsection Network Function Virtualization concludes with an introduction into the virtualization of network node functions to create communication services.

2.1 Internet of Things

2.1.1 Smart Factories

2.1.2 Cyber Physical Systems

2.1.3 Smart Cities

2.2 Virtualization

2.2.1 Virtual Machines

2.2.2 Container Virtualization

Docker

"First, we must know what exactly Docker is and does. Docker is a container managment system that helps easily manage Linux Container (LXC) in an easier and unversal fashion. This lets you create images in virtual environment ons your laptop and run commands or operations against them. The actions you do to the containers that you run in these environments locally on your own machine will be the same commands or operations you run against them

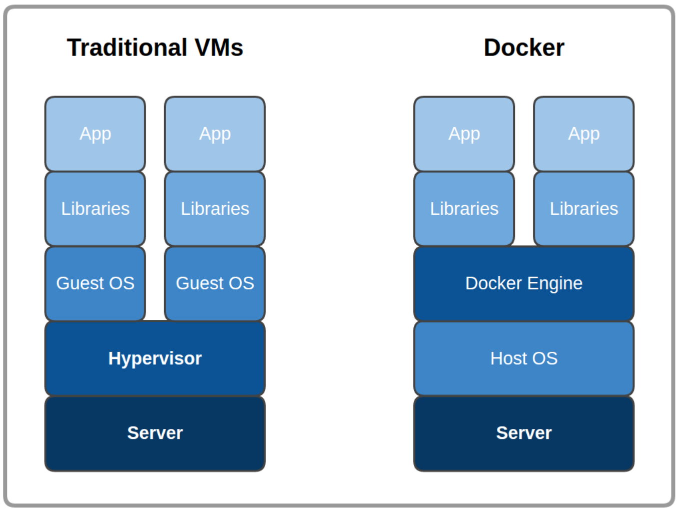

Figure 1: Structure traditional VMs vs. Docker. Adapted from: [Gal15], p. 2

when they are running in your production environment. This helps in not having to do things differently when you go from a development environment like that on your local machine to a production environemnt like that on your local machine to a production environemnt on your server. Now, let's take a look at the differences between Docker containers and the typical virtual machine environments.

In the followin illustration, we can see the typical Docker setup on the right-hand side versus the typical VM setup on the left-hand side:

This illustration gives us a lot of insight into the biggest key benefit of Docker, that is, there is no nedd for a complete operating system every titme we need to bring up a new container, which cuts down on the overall size of containers. Docker relies on using the host OS's Linux kernel (since almost all the versions of Linux use the standard kernel models) fot he OS it was build upon, such as Red Hat, CentOS, Ubuntu, and so on. For this reason, you can have almost any Linux OS as your host operationg system (Ubuntu in the previous illustartion) and be able to layer other OSes on top of the host. For example, in the earlier illustration, we could have Red Hat running for one app (the one on the left) and Debian running for the other app (the on on the right), but there would never be need to actually install Red Hat or Debian on the host. Thus, another benefit of Docker ist the size of aimages when they are born. They are not build with the largest piece: the kernel or the operating sytem. This makes them incredibley small, compact, and easy to ship."[Gal15]

2.2.3 Container Orchestration

Kubernetes

Docker Swarm

2.2.4 Network Function Virtualization

2.3 Conclusion

Chapter 3

Requirements Analysis

3.1 Introduction

3.2 System requirements

3.3 Technologies

3.4 Use-Case-Analysis

3.5 Delineation from existing solutions

3.6 Conclusion

Chapter 4

Design

4.1 Introduction

4.2 Development environment

4.3 Evaluation of existing frameworks

4.3.1 Docker

4.3.2 Docker Swarm

4.3.3 Kubernetes

4.3.4 Open Baton

4.3.5 ETSI MANO

4.3.6 TOSCA

4.4 Architecture of the system

4.4.1 Orchestration layer

4.4.2 Constraint layer

4.4.3 User interface

4.5 Conclusion

Chapter 5

Implementation

5.1 Introduction

5.2 Project structure

5.3 Used external libraries

5.4 Custom code

5.5 Implementation of the orchestration layer

5.6 Implementation of the constraint layer

5.7 Implementation of the user interface

5.8 Conclusion

Chapter 6

Evaluation

6.1 Introduction

6.2 Experimental Validation

6.3 Performance Evaluation

6.4 Observational Validation

6.5 Deployments

6.6 Code Verification

6.7 Comparative Analysis

6.8 Conclusion

Chapter 7

Summary and Further Work

7.1 Overview

7.2 Conclusion and Impact

7.3 Outlook

Acronyms

Glossary

Algorithmus a

Chiffrierung a

Dechiffrierung a

Bibliography

[And] *Android Instant Apps.* https://developer.android.com/topic/instant-apps/index.html, . – Accessed: 2017-01-25

[BHSW16] Brito, M. S. D. ; Hoque, S. ; Steinke, R. ; Willner, A.: Towards Programmable Fog Nodes in Smart Factories. In: *2016 IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS*W)*, 2016, S. 236–241

[Cor] *CoreOS.* https://coreos.com/, . – Accessed: 2017-01-17

[Doca] *Docker Remote API.* https://docs.docker.com/engine/reference/api/docker_remote_api/, . – Accessed: 2017-01-19

[Docb] *Docker Swarm.* https://www.docker.com/products/docker-swarm, . – Accessed: 2017-01-17

[ETS14] Network Functions Virtualisation (NFV); Management and Orchestration. (2014), Dec, Nr. TJA1043. http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf. – Accessed: 2017-01-14

[FRS+13] Fantana, N. L. ; Riedel, T. ; Schlick, J. ; Ferber, S. ; Hupp, J. ; Miles, S. ; Michahelles, F. ; Svensson, S.: IoT Applications — Value Creation for Industry. In: Vermesan, Dr. O. (Hrsg.) ; Friess, Dr. P. (Hrsg.): *Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems.* River Publishers, 2013. – ISBN 978–87–92982–73–5, S. 153–206

[Gal15] Gallagher, Scott: *Mastering Docker.* Packt Publishing Ltd., 2015. – ISBN 978–1–78528–703–9

[git] *git.* https://git-scm.com/, . – Accessed: 2017-01-23

[Kub] *Kubernetes.* https://kubernetes.io/, . – Accessed: 2017-01-17

[kur] *kura.* http://www.eclipse.org/kura/, . – Accessed: 2017-01-23

[Lee08] Lee, E. A.: Cyber Physical Systems: Design Challenges. In: *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2008. – ISSN 1555–0885, S. 363–369

[Lin] *Linux Containers.* https://linuxcontainers.org/, . – Accessed: 2017-01-19

[LPS16] Lom, M. ; Pribyl, O. ; Svitek, M.: Industry 4.0 as a part of smart cities. In: *2016 Smart Cities Symposium Prague (SCSP)*, 2016, S. 1–6

[LYK16] Lee, H. ; Yoo, S. ; Kim, Y. W.: An energy management framework for smart factory based on context-awareness. In: *2016 18th International Conference on Advanced Communication Technology (ICACT)*, 2016, S. 685–688

[Mih] *Mihini*. `https://wiki.eclipse.org/Mihini`, . – Accessed: 2017-01-23

[OAS15] TOSCA Simple Profile for Network Functions Virtualization (NFV). (2015), May, Nr. TJA1043. `http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd01/tosca-nfv-v1.0-csd01.pdf`. – Accessed: 2017-01-14

[Opea] *OpenBaton Documentation*. `http://openbaton.github.io/documentation/`, . – Accessed: 2017-01-14

[Opeb] *OpenShift*. `https://www.openshift.com/`, . – Accessed: 2017-01-19

[PHM+16] Pahl, C. ; Helmer, S. ; Miori, L. ; Sanin, J. ; Lee, B.: A Container-Based Edge Cloud PaaS Architecture Based on Raspberry Pi Clusters. In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, 2016, S. 117–124

[PL15] Pahl, C. ; Lee, B.: Containers and Clusters for Edge Cloud Architectures – A Technology Review. In: *2015 3rd International Conference on Future Internet of Things and Cloud*, 2015, S. 379–386

[Poo10] Poovendran, R.: Cyber Physical Systems: Close Encounters Between Two Parallel Worlds. In: *Proceedings of the IEEE* 98 (2010), Aug, Nr. 8, S. 1363–1366. `http://dx.doi.org/10.1109/JPROC.2010.2050377`. – DOI 10.1109/JPROC.2010.2050377. – ISSN 0018–9219

[RD15] Rui, J. ; Danpeng, S.: Architecture Design of the Internet of Things Based on Cloud Computing. In: *2015 Seventh International Conference on Measuring Technology and Mechatronics Automation*, 2015. – ISSN 2157–1473, S. 206–209

[res] *resin.io*. `https://resin.io/`, . – Accessed: 2017-01-23

[RN16] Ramalho, F. ; Neto, A.: Virtualization at the network edge: A performance comparison. In: *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2016, S. 1–6

[SMD15] Stubbs, J. ; Moreira, W. ; Dooley, R.: Distributed Systems of Microservices Using Docker and Serfnode. In: *2015 7th International Workshop on Science Gateways*, 2015, S. 34–39

[TRA15] Tosatto, A. ; Ruiu, P. ; Attanasio, A.: Container-Based Orchestration in Cloud: State of the Art and Challenges. In: *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, 2015, S. 70–75

[YMSG+14] Yannuzzi, M. ; Milito, R. ; Serral-Gracià, R. ; Montero, D. ; Nemirovsky, M.: Key ingredients in an IoT recipe: Fog Computing, Cloud computing, and more Fog Computing. In: *2014 IEEE 19th International Workshop* *on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2014. – ISSN 2378–4865, S. 325–329