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

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, March 21, 2017

_____

Markus Paeschke

Acknowledgments

thank your supervisors

thank your colleagues

thank your family and friends

Berlin, March 21, 2017

Abstract

Research Area - write about the research area Lorem ipsum dolor sit amet, consectetur adipisic- ing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Application Area - write about the application area Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Research Issue - write about the research issue Lorem ipsum dolor sit amet, consectetur adipisic- ing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Own Approach - write about the own approach Lorem ipsum dolor sit amet, consectetur adipisic- ing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Scientific Contributions - write about the scientific contributions Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Validation & Outlook - write about the validation and outlook Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Zusammenfassung

Forschungsbereich Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

wrEingrenzungite Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Problemstellung Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Eigener Ansatz Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Wissenschaftlicher Beitrag Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Validierung & Ausblick Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

1.1 Motivation

The Internet of Things (IoT) is one of the biggest topic in the recent years. Companies with a focus in that area have an enormous market growth with plenty of new opportunities, use cases, technologies, services and devices. Bain & Company predicts an annual revenue of $450 billion for companies who selling hardware, software and comprehensive solutions in the IoT context by 2020.[BCJ+16] In order to limit the vast area of IoT, more and more standards are defined and subtopics established. The European Research Cluster on the Internet of Things (IERC) divided them into eight categories: Smart Cities, Smart Health-care, Smart Transport and Smart Industry also known as Industry 4.0 to mention only a few. All of them are well connected, for example a Smart Factory, which is a part of the Smart Industry, can get a delivery from a self driving truck (Smart Transport) which navigates through a Smart City to get to the factory. Such information networks are one of the main goals of IoT. In the Industry 4.0 for example multiple smart factories should be interconnect into a distributed and autonomous value chain. Also the automation in a single factory will be increased which helps to have a more flexible and efficient production process. Currently a factory has a high degree of automation, but due to a lack of intelligence and communication between the machines and the underlying system, they can not react to changing requirements or unexpected situations. One solution to achieve that are Cyber-Physical Systems (CPSs). These are virtual systems which are connected with embedded systems to monitor and control physical processes.[cf. Lee08, p. 363] A normal Cyber Systems (CSs) is passive, means it could not interact with the physical world, with the appearance of CPSs things can communicate so the system has significantly more intelligence in sensors and actuators.[cf. Poo10, p. 1363 f.]

Another solution is to change the fundamental architecture of such a system from a monolithic to more distributed multicloud architecture. With Fog Computing the cloud moves away from centralized data centers to the edge of the underlying network.[cf. PL15, p. 380] Such a network can have thousands of nodes with multiple sensors, machines or smart components connected to them. An "intermediate layer between the IoT environment and the Cloud"[BHSW16, p.236] enables a lot of new possibilities like pre-computation and storage of gathered data, which reduces traffic and resource overhead in the cloud, it keeps sensitive data on-premise[cf. BHSW16, p.236] and enables real-time applications to take decisions based on analytics running near the device and a lower latency. On the other hand there

are also a lot of challenges in these highly heterogeneous and hybrid environment. As an example in some scenarios multiple low power devices have to interact with each other, lossy signals and short range radio technologies are widely used and nodes can appear and disappear frequently.[cf. YMSG+14, p. 325] Especially the last case is elaborated because the underlying system has to handle that. Furthermore the required applications running on these nodes can be change commonly and have to be deployed and removed in a dynamical way. Virtualization with Virtual Machines (VMs) is a common approach in Cloud systems to provide elasticity of large-scale shared resources.[cf. PHM+16, p. 117] A more lightweight, less resource and time consuming solution is container virtualization. "Furthermore, they are flexible tools for packaging, delivering and orchestration software infrastructure services as well as application"[PHM+16, p. 117]. Orchestration tools like Kubernetes[1] and Docker Swarm[2] which deploy, scale and manage containers to clusters of hosts have become established in the last years. Moving this technology over to the IoT area many challenges can be solved. Dynamically deployed applications at the edge of a network can store and preprocesses gather data even if a node have no connection to the cloud because of lossy signals. Traffic can be reduced by only transmitting aggregated data back to the cloud. More often small low-power devices with limited computational power are be used as IoT nodes which also profit rather from lightweight container solutions than from resource consuming VMs. This thesis shows the capabilities of container orchestration for the IoT and smart factories using the Open Baton framework. Therefor a plugin will be created which can orchestrate containers based on functional and non-functional constraints to fog nodes.

1.2 Objective

This thesis describes an approach to design and implement a fog service orchestration engine for smart factories. The aim of this work is to create a plugin for Open Baton, an European Telecommunications Standards Institute (ETSI) Network Function Virtualistion (NFV) compliant Management And Orchestration (MANO) framework, which can deploy containers to network nodes, which are typically lower power devices at the edge of a network, especially for the area of Industry 4.0 and smart factories. Furthermore the plugin should consider specific functional and non-functional constraints while deploying the Network Functions (NFs). A condition can be a hardware requirement, a required software or a dependencies to another node. The technical prerequisite and detailed requirements for the plugin as well as the usability of Open Baton as the tool of choice have to be worked out. The cooperation with the Fraunhofer FOKUS plays a prominent role for this thesis, because they have a lot of knowledge and experience especially in this area.

1.3 Scope

As mentioned before, the prototype will be an plugin for Open Baton, which means that the creation of an MANO engine is out of scope for this thesis. Open Baton has a modular architecture so that the proof-of-concept implementation should be possible without modifying

[1] `https://kubernetes.io`
[2] `https://docs.docker.com/engine/swarm`

the framework himself. The container virtualization will be realized with Docker[3], an open source container platform. Docker has on Application Programming Interface (API) where third party apps can communicate with and can control the engine himself. The functional and non-functional constraints have to be implemented in the Network Function Virtualistion Orchestrator (NFVO) and could base on the YAML Ainť Markup Language (YAML) schemas which are part of the Topology and Orchestration Specification for Cloud Applications (TOSCA) standard. These schemas should be extendable and should fit the needs of the constraint functionality.

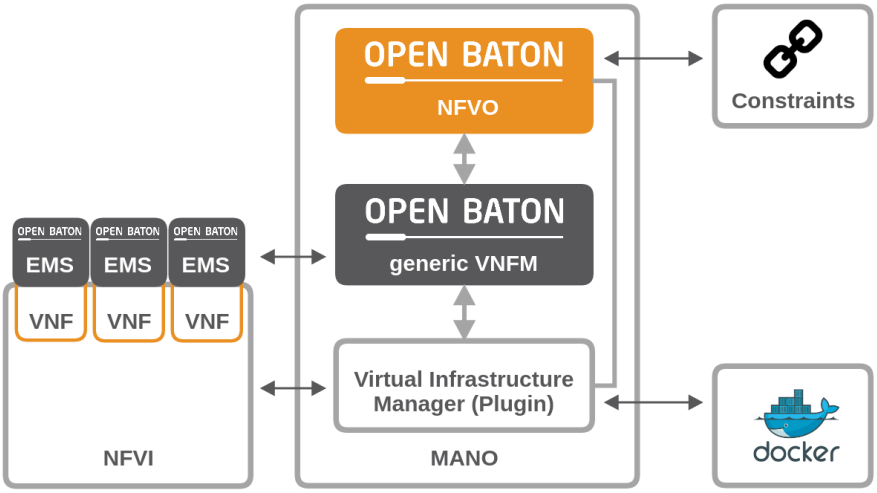

Figure 1: Conceptual architecture.

Figure 1 shows an abstract architecture concept where Docker is part of the Virtual Infrastructure Manager (VIM) and the constraints are part of the NFVO. Furthermore the engine will be tested on a raspberry pi[4] cluster. It is out of scope to create a system which guaranteed to be executed on arbitrary devices, but theoretically this could be achieved with a system like Open Baton and Docker.

1.4 Outline

In chapter 2 an introduction in relevant topics like the IoT with the Industry 4.0, CPSs and Fog Computing as subtopics, virtualization especially container virtualization and orchestration and NFV is given. Followed by a comparison of several established tools available on the market. The definition of the requirements as well as features and capabilities of the prototype will be shown in chapter 3. Based on that, the architecture of the system is illustrated in chapter 4. The proof-of-concept implementation will be worked out in chapter 5 and the functionality of the plugin will be demonstrated. Chapter 6 summarizes the results of the work and evaluates the viability in terms of software quality, usability and feature-completeness. Finally the gained learnings as well as an outlook for further improvements of the plugin will be argued in chapter 7.

[3] `https://www.docker.com`
[4] `https://www.raspberrypi.org`

Chapter 2

State of the art

This chapter will give an overview into the background and concepts of this thesis. In the first section the Internet of Things (IoT) an related subtopics like smart factories and Smart Cities are considered. Cyber-Physical Systems (CPSs), 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 (VMs), followed by Container Virtualization. Both are related to each other and sharing some basic ideas. Container Orchestration as an own subsection show some possibilities of Container Virtualization. The last subsection Network Function Virtualistion (NFV) concludes with an introduction into the virtualization of network node functions to create communication services.

2.1 Internet of Things

The IoT has been a subject of great media- and economically growth in the recent years. In the year 2008 the number of devices which are connected to the Internet was higher than the human population.[cf. Eva11, p. 3] Cisco Internet Business Solutions Group predicted that the number will grow up to 50 billion in 2020, this equates to around 6 devices per person.[cf. Eva11, p. 4] Most of todayś interactions are Human-to-Human (H2H) or Human-to-Machine (H2M) communication. The IoT on the other hand aim for the Machine-to-Machine (M2M) communication. This allows every physical devices to be interconnected and to communicate with each other. These devices are also called "Smart Devices". Creating a network where all physical objects and people are connected via software is one primary goal of the IoT.[cf. RD15, p.206][cf. KKL13, p.2] When objects are able to capture and monitor their environment, a network can perceive external stimuli and respond to them.[cf. Itu05, p. 40] Therefore a new dimension of information and communication technology will be created, where users have access to everything at any time, everywhere. In addition to smart devices, subcategories are also emerging from the IoT which, in addition to the physical devices, also describe technologies such as protocols and infrastructures. The "Smart Home" has been a prominent topic in media and business for many years. Smart City or Industrie 4.0 are also becoming established and are increasingly popular. But the Internet started with the appearance of bar codes and Radio Frequency Identification (RFID) chips.[cf. KKL13, p. 13] The second step, which is more or less the current situation, sensors, physical devices, technical devices, data and software are connected to each other.[cf. KKL13, p. 13] This was achieved,

in particular, by cloud computing, which provides the highly efficient memory and computing power that is indispensable for such networks.[cf. RD15, p. 206] The next step could be a "Cognitive Internet of Things", which enables easier object and data reuse across application areas, for example through interoperable solutions, high-speed Internet connections and a semantic information distribution.[cf. KKL13, p. V] Just as the omnipresent information processing in everyday life, also known as "Ubiquitous Computing", which was first mentioned in the "The Computer for the 21st Century"[Wei91] by Marks Weiser, it will take some time until it is ubiquitous.

2.1.1 Industry 4.0 and smart factories

The industry as an changing environment is currently in the state of the so called "fourth industrial revolution". The first industrial revolution was driven by steam powered machines. Mass production and division of labor was the primary improvement of the second industrial revolution, whereas the third revolution was characterized by using electronics and the integration of Information Technology (IT) into manufacturing processes.[cf. LPS16, p. 1] In the recent years the size, cost and power consumption of chipsets are reduced which made it possible to embed sensors into devices and machines much easier and cheaper.[cf. BHSW16, p. 1] The Industry 4.0 is the fourth step in this evolution and was first mentioned with the German term "Industrie 4.0" at the Hannover Fair in 2011.[cf. LPS16, p. 1] "Industrie 4.0 is a collective term for technologies and concepts of value chain organization."[cf. HPO15, p. 11]

Significantly higher productivity, efficiency, and self-managing production processes where everything from machines up to goods can communicate and cooperate with each other directly are the visions of the Industry 4.0.[Lyd16, cf.] It also aims for an intelligent connection between different companies and units. Autonomous production and logistics processes creating a real-time lean manufacturing ecosystem that is more efficient and flexible.[Lyd16, cf.] "This will facilitate smart value-creation chains that include all of the life-cycle phases of the product from the initial product idea, development, production, use, and maintenance to recycling."[Lyd16] At the end, the system can use customer wishes in every step in the process to be flexible and responsive.[Lyd16, cf.]

	Cyber-Physical Systems	Internet of Things	Internet of Services	Smart Factory
Interoperability	X	X	X	X
Virtualization	X	-	-	X
Decentralization	X	-	-	X
Real-Time Capability	-	-	-	X
Service Orientation	-	-	X	-
Modularity	-	-	X	-

Table 1: Design principles of each Industry 4.0 component.[cf. HPO15, p. 11]

Table 1 shows the six design principles which can be from the Industrie 4.0 components. They can help companies to identify and implement Industry 4.0 scenarios.[cf. HPO15, p. 11]

1. *Interoperability* CPS of various manufacturers are connected with each other. Standards will be the key success factor in this area.[cf. HPO15, p. 11]

2. *Virtualization* CPS are able to monitor physical processes via sensors. The resulting data is linked to virtual plant and simulation models. These models are virtual copies of physical world entities.[cf. HPO15, p. 11]
3. *Decentralization* CPS are able to make decisions on their own, for example when RFID chips send the necessary working steps to the machine. Only in cases of failure the systems delegate task to a higher level.[cf. HPO15, p. 11]
4. *Real-Time Capability* Data has to be collected and analyzed in real time and the status of the plant is permanently tracked and analyzed. This enables the CPS to react to a failure of a machine and can reroute the products to another machine.[cf. HPO15, p. 11]
5. *Service Orientation* CPS are available over the Internet of Services (IoS) and can be offered both internally and across company borders to different participants. The manufacturing process can be composed based on specific customer requirements.[cf. HPO15, p. 11]
6. *Modularity* The system is able to be adjusted in case of seasonal fluctuations or changed product characteristics, by replacing or expanding individual modules.[cf. HPO15, p. 11]

Another important aspect of Industry 4.0 is the implementation of process automation with the focused on three distinct aspects. Starting with the vertical integration, which contains the connection and communication of subsystems within the factory enables flexible and adaptable manufacturing systems.[cf. Vbw14, p. 7 ff.] The horizontal integration, as the second aspect, enables technical processes to be integrated in cross-company business processes and to be synchronized in real time through multiple participants to optimize value chain outputs.[cf. Vbw14, p. 7 ff.] Finally end-to-end engineering, planning, and process control for each step in the production process.[Lyd16, cf.]

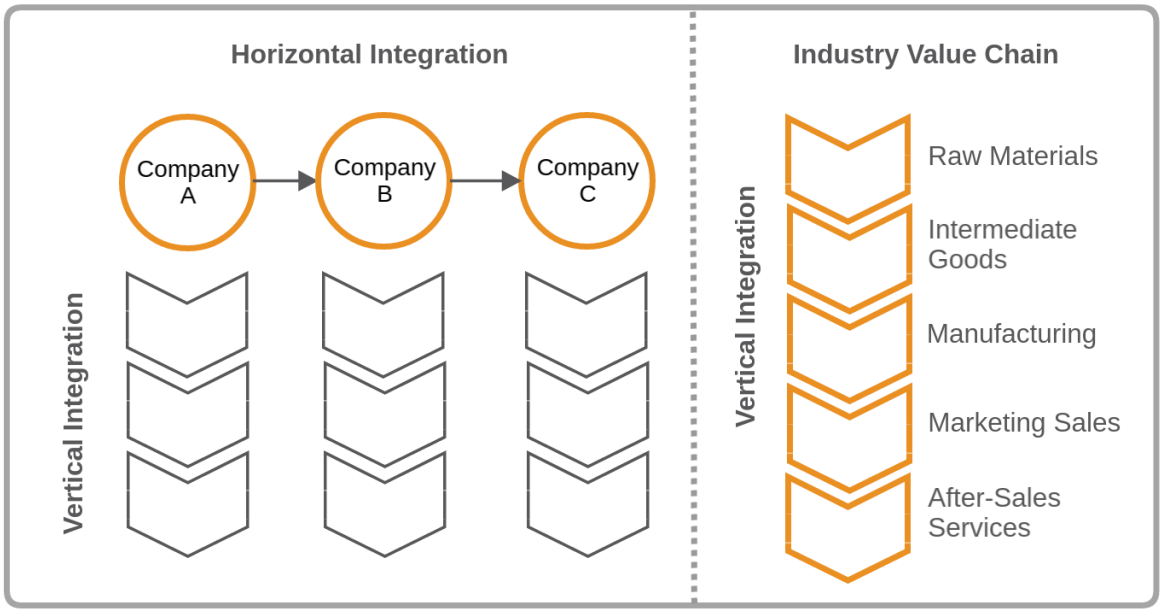

Figure 2: Horizontal vs. Vertical Integration. Adapted from: [Jur13]

Figure 2 illustrates this concept. The left side shows the whole production process over company boundaries on the horizontal scale, as well as the industry value chain on the vertical scale which is specific for each company. On the right side there is an exemplary industry value chain which starts with the raw materials and ends with the sale of the product to illustrates an more specific example of the vertical integration. From a technical site this means each machine in a factory has exactly to know what they have to do. The underlying system has to be modular and move away from a monolithic centralized system, to a decentralized system which is located locally near the machines himself. The communication path between them have to grow shorter. The machines have to be self organized and should communicate between each other even if the core system is not reachable because of lossy signals or other connection issues. If this can be achieved there will be an highly flexible, individualized and resource friendly mass production, which can be cheaper, faster and can have a much higher fault tolerance.

2.1.2 Cyber Physical Systems

As we already now, in smart factories every physical device is connected to each other. Everything can be captured and monitored in each step of a production process. With CPSs every physical entity has a digital representation in the virtual system.[cf. Poo10, p. 1363] Before a Cyber System (CS) was passive, which means there was no communication between the physical and the virtual world.[cf. Poo10, p. 1364] While new technologies in the physical world, like new materials, hardware and energy, are developed, the technologies in the virtual worlds are also being improved, for example through the use of new protocols, networking, storage and computing technologies.[cf. Poo10, p. 1364] This adds more intelligence in such systems, as well as a much more flexible and modular structure. A CPS can organize production automatically and autonomously, which eliminate the need of having a central process control.[LPS16, cf.] Thereby the system can handle lossy signals and short range radio technologies, which are widely used in such a context.[YMSG$^+$14, cf.] In summary CPSs can help to enable the vision of smart factories in both the horizontal as well as the vertical integration.

2.1.3 Fog Computing

In the beginning of Cloud Computing most of the systems based on a monolithic architecture. Over time the system was broken down to a more distributes multicloud architecture, similar to microservices. With the appearance of Fog Computing the Cloud also moves from centralized data centers to the edge of the underlying network. Main goal is to improve the efficiency, reduce the traffic and the amount of data which is transferred to the cloud and also process, analyze and store data locally, as well as keeping sensitive data inside the network for security reasons.[cf. BHSW16, p. 236][cf. YMSG$^+$14, p. 325][cf. LPS16, p. 4] In contrast to the goals the definition and understanding of Fog Computing differs. One perspective is to that the processing of the data take place on smart devices, e.g. sensors, embedded systems, etc., at the end of the network or in smart router or other gateway devices.[cf. LPS16, p. 4] Another interpretation is that fog computing appears as an intermediate layer between smart devices and the cloud.[cf. BHSW16, p. 236] Processing the data near devices enables lower latency and real-time applications can take decisions based on analytics running there. That is important because a continuous connection to the cloud can not always be ensured.

However fog computing should not be seen as a competitor of cloud computing, it is a perfect ally for use cases where cloud computing alone is not feasible.[cf. YMSG+14, p. 325]

2.2 Virtualization

According to the National Institute of Standards and Technology (NIST) the definition of virtualization is: "Virtualization is the simulation of the software and/or hardware upon which other software runs. This simulated environment is called a virtual machine (VM)."[SSH11, p. ES-1]. This means a VM, also referred as guest system, can be executed in a real system, which is referred as host system. A VM has its own Operating System (OS) which is completely isolated from the other VMs and the host system.[cf. CMF+16, p. 2] Basically there are two types of virtualization: Process virtualization where the virtualizing software also known as Virtual Machine Monitor (VMM) is executed by the host OS and only an application will be executed inside the guest OS and on the other side there is the the system virtualization where the whole OS as well as the application are running inside the virtualizing software. Figure 3 illustrate both concepts. Examples for process virtualization could be the Java Virtual Machine (JVM)[1], the .Net framework[2] or Docker[3], where VMWare[4], Oracle Virtual Box[5], XEN[6] or Microsoft Hyper-V[7] are only some examples for system virtualization. The benefits of all virtualization techniques are the rapid provisioning of resources which could be Random Access Memory (RAM), disk storage, computation power or network bandwidth. Beside that, no human interaction is necessary during the provisioning process. Elasticity which scales a system in a cost-efficient manner in both directions, up and down. Customer as well as the provider profit from such a system. Security based on the isolation of the VMs is another huge benefit. Different processes can not interfere with each other and the data of a single user can not be accessed by other users of the same hardware. A challenge despite all the mentioned benefits is the performance. Running VMs increases the overhead and reduces the overall performance of a system. Therefore the specific use case have to consider these behavior.

[1] https://www.java.com
[2] https://www.microsoft.com/net
[3] https://www.docker.com
[4] http://www.vmware.com
[5] https://www.virtualbox.org
[6] https://www.xenproject.org
[7] https://www.microsoft.com/de-de/cloud-platform/server-virtualization

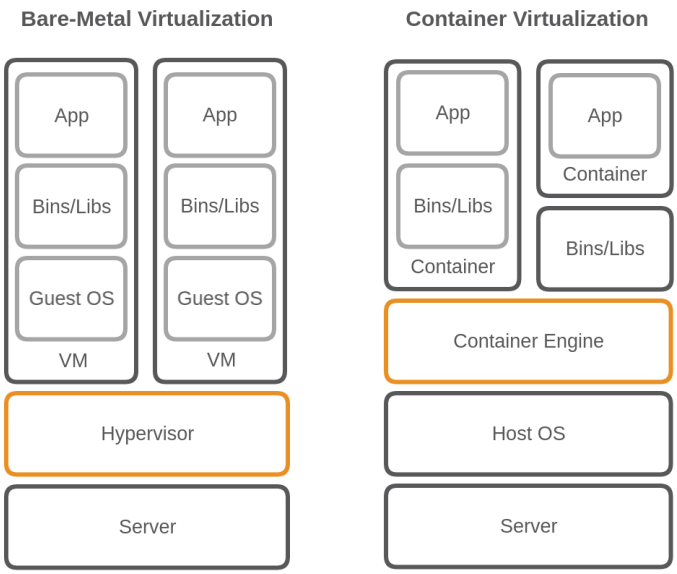

Figure 3: Structure bare-metal virtualization vs. container virtualization. Adapted from: [Gal15, p. 2]

2.2.1 Virtual Machines

VMs are the core virtualization mechanism in cloud computing. There are also two different designs for hardware virtualization. The first and more popular type for cloud computing is the *bare-metal virtualization*. It needs only a basic OS to schedule VMs. The hypervisor runs directly on the hardware of the machine without any host OS in between. This is more efficient, but requires special device drivers to be executed. The other type is the *hosted virtualization*. Unlike the first type the VMM run as a host OS process and the VMs as a process supported by the VMM. No special drivers are needed for these type of virtualization, but by comparison the overhead is much bigger. For both types, the performance limitation remains. Each VM need a full guest OS image in addition to binaries and libraries which are necessary for the application to be executed.[cf. PL15, p. 381] If only a single application, which only needs a few binaries and libraries, is needed to be virtualized, VMs are too bloated.

2.2.2 Container Virtualization

Container virtualization which is also known as Operating System-level virtualization, is the second virtualization mechanism. It based on fast and lightweight process virtualization to encapsulate an entire application with its dependencies into a ready-to-deploy virtual container.[cf. TRA15, p. 72] Such a container can be executed on the host OS which allows an application to run as a sand-boxed user-space instance.[cf. AHA+16, p. 1] All containers share a single OS kernel, so the isolation supposed to be weaker compared to hypervisor based virtualization.[cf. CMF+16, p. 2] Compared to VMs, the number of containers on the same physical host can be much higher, because the overhead of a full OS virtualization is eliminated.[cf. CMF+16, p. 2]

2.2.3 Container Orchestration

Containers by itself helps to develop and deploy applications, but containers release their full potential only when they are used together with an orchestration engine. Before orchestration engines, the deployment of an application or service was realized via Continuous Integration (CI) and deployment tools like Vagrant or Ansible. Deployment scripts or plans was created and be executed every time an application changed or should be scaled up on a new machine. This was less flexible and error-prone. Orchestration engines cover these needs by automatically choosing new machines, deploying containers, handle the lifecycle of them and monitor the system. These flexibility enables a new level of abstraction and automatization of deployment. There are a bunch of orchestration engines out there. For Docker, Kubernetes and Docker Swarm are the most popular at the moment.

2.2.4 Network Function Virtualization

NFV is an architectural framework to provide a methodology for the design, implementation, and deployment of Network Functions (NFs) through software virtualization.[cf. ETS13, p. 8][Riv14, cf.] "These NFs are referred as Virtual Network Functions (VNFs)."[ETS13, p. 8] It takes into consideration Software Defined Networking (SDN) and preparing for the use of non-proprietary software to hardware integration instead of multiple vendor specific devices for each function, e.g. routers, firewalls, storages, switches, etc.[Riv14, cf.] Now high-performance firewalls and load balancing software for example can run on commodity PC hardware and traffic can be off-loaded onto inexpensive programmable switches.[Nob15, cf.]

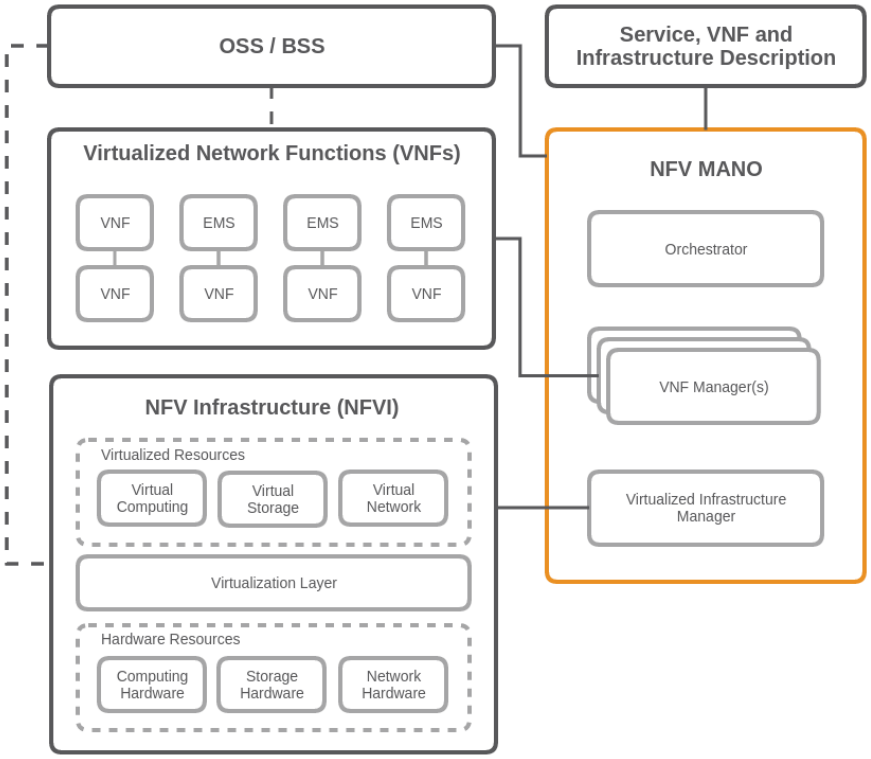

Figure 4: NFV architecture. Adapted from: [NFV]

Some benefits are speed, agility and cost reduction in deployment as well as execution manner.[Nob15, cf.] Using homogeneous hardware simplifies the process of planning and reduces power, cooling and space needs.[Nob15, cf.] Through virtualization providers can utilize resources more effectively, by allocating only the necessary resources for a specific functionality.[Nob15, cf.] Overall NFV can reduce Operating Expense (OpEx) as well as Capital Expenditure (CapEx) and can decreasing the time necessary to deploy new services to the network.[Nob15, cf.] To achieve NFV the European Telecommunications Standards Institute (ETSI) has defined a framework the Network Function Virtualistion Management And Orchestration (NFV-MANO)[8] and the Organization for the Advancement of Structured Information Standards (OASIS) created Topology and Orchestration Specification for Cloud Applications (TOSCA) a NFV specific data model and templates to coordinate and orchestrate the NF into the cloud.

OSS / BSS refers to the Operations Support Systems (OSS) and the Business Support Systems (BSS) of a telecommunication operator.[Kah15, cf.] The OSS is responsible for the underlaying soft- and hardware system, for example for network and fault management. The BSS on the other hand is responsible for the business handling, for example customer and product management. Both can be integrated with the NFV-MANO.[Kah15, cf.]

VNFs are the virtualized network elements, for example a virtualized router or virtualized firewall. Even if only a sub-functions or a sub-components of a hardware element is virtualized, it is called VNF.[Kah15, cf.] Multiple sub-functions can act together as one VNF.

Element Management System (EMS) is responsible for the functional management of a single or multiple VNFs.[Kah15, cf.] This includes fault, configuration, accounting, perfomance and security management.[Kah15, cf.] Furthermore the EMS itself can be a VNF or it can handle a VNF through proprietary interfaces.[Kah15, cf.]

Network Function Virtualization Infrastructure (NFVI) is the environment where VNFs are executed. This includes physical resources as well as virtual resources and the virtualization layer. The physical resources could be a commodity switch, a server or a storage device. These physical resources can be abstracted into virtual resources through the virtualization layer which is normally a hypervisor. If the virtualization part is missing, the software runs natively on the hardware and the entity is no longer a VNF it is then a Physical Network Function (PNF).[Kah15, cf.]

NFV-MANO consists of three main parts. The Virtual Infrastructure Manager (VIM) is "responsible for controlling and managing the NFVI compute, network and storage resources within one operator's infrastructure domain"[Kah15]. The Virtual Network Function Manager (VNFM) manages one or multiple VNFs. This includes the life cycle management of the VNF instances, such as instantiate, edit or shut down an VNF instance.[Tos, cf.] In contrast to the EMS, the VNFM handles the virtual part of the VNF, for example instantiate an instance, while the EMS handles the functional part of an VNF, such as issue handling for a VNF. The orchestrator as the third component in the NFV-MANO block manage network services of VNFs. It is responsible for the global resources management, such as computing and networking resources among multiple VIMs.[Kah15, cf.] The orchestrator interacts with the VNFM to perform actions, but not with the VNFs directly.[Kah15, cf.] TOSCA is often used with NFV-MANO frameworks like Cloudify or Open Baton.[Tos, cf.]

[8] `http://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_NFV-MAN001v010101p.pdf`

TOSCA is developed by the OASIS to deliver a declarative description of a NFV application topology for network or cloud environments.[Tos, cf.] In figure 4 it is represented by the *Service, VNF and Infrastructure Description* block. Beside that, it can also be used to define workflows which should be automated in a virtualized environment.[Tos, cf.] The TOSCA modeling language can specify ndoes, whereby a node can be a network, a subnet or only a server software component, and it also handles relationships between the nodes and also services.[Tos, cf.] To define schemas, relationships and the configuration of such an infrastructure, it uses YAML Ainť Markup Language (YAML) files for ease the usage.[Tos, cf.] TOSCA works pretty well with NFV-MANO components to automate the deployment and management of NFs and services.

2.3 Existing tools and frameworks

There are several advantages of using frameworks and sophisticated tools, for example they reduces the time and energy in developing any software, they are more secure, well tested and they provides a standardized system through which users can develop applications. They also allow to create a prototype of an application in a short amount of time. To achieve the benefits the user has to spend some time to learn the concepts, functions and how to use a framework.

2.3.1 Linux Containers

When we talk about container virtualization nowadays, Docker have to become one of the most famous tools out there. It based on Linux Containers (LXC)[9] a technology which uses kernel mechanisms like *namespaces* or *cgroups* to isolate processes on a shared OS.[cf. PL15, p. 381] Namespaces for example are used to isolate groups of processes whereas cgroups are used to manage and limit resources access just like restricting the memory, disc space or Central Processing Unit (CPU) usage.[cf. PL15, p. 381] "The goal of LXC is to create an environment as close as possible to a standard Linux installation but without the need for a separate kernel."[TRA15, p. 72] There are several other container virtualization tools out there like OpenVZ[10] or Linux-VServer[11]. In contrast to them, an advantage of LXC is that it runs on an unmodified Linux kernel. This means that LXC can be executed in most of the popular Linux distributions these days.

2.3.2 Docker

As mentioned before, Docker based on LXC. This allows the Docker Engine to build, deploy and run containers in an easy and customizable way. Similar to VMs, containers are executed from images. A mayor benefit of Docker is the fact, that Docker images can be combined like build blocks. Each image can build on top of another. Figure 5 illustrates the concept for the image of the pretty famous Django[12] web framework. In that case, the Django image, which

[9] `https://linuxcontainers.org/`
[10] `https://openvz.org/Main_Page`
[11] `http://www.linux-vserver.org`
[12] `https://www.djangoproject.com/`

is the resulting image, based on the Python 3.4 image which again based on the Debian Jessy image. All of them are read-only, but Docker adds a writable layer, also known as *container layer*, on top of the images as soon as the container will be created. File system operations such as creating new files, modifying or deleting existing files are written directly to these layer.[Docb, cf.] The other images dont́ get involved. These chaining mechanism of images allows Docker to ease the use of dependencies and administrative overhead.

Beside that, Docker is split up in several components, such as the Docker Engine, the Docker Registries and Docker Compose to mention only a few. The Docker Engine is a client-server application which can be distinguished by the Docker client, a Representational State Transfer (REST) Application Programming Interface (API) and the Docker server. Latter is a daemon process which "creates and manages Docker objects, such as images, containers, networks, and data volumes"[Doca]. The client is a Command Line Interface (CLI), which interact via a REST API with the daemon.[Doca, cf.]

Another component, the Docker Registry, is basically a library of Docker images. They can be public or private available, as well as on the same machine like the Docker daemon or an external server.[Doca, cf.] The most popular one it the official Docker Hub[13]. There is also an Docker Store[14], where customers can buy and sell trusted and enterprise ready containers and plugins. With the Docker client it is pretty easy to *search* for new containers and to *pull* containers from or *push* containers to a specific repository.

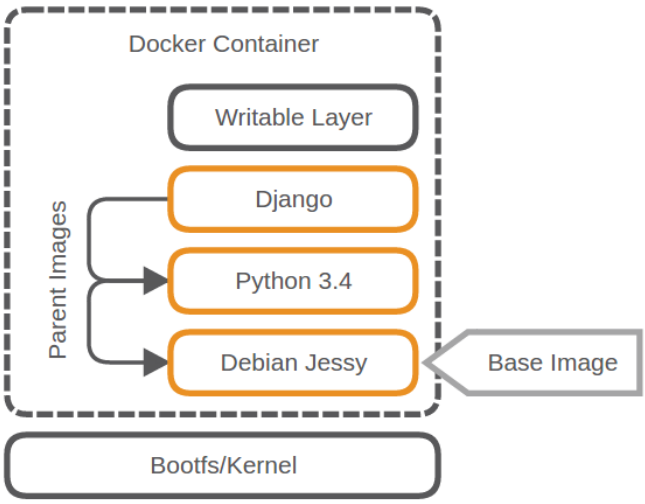

Figure 5: Docker container structure.

With Docker Compose multiple Docker Containers can be executed as a single application. Therefore YAML compose file will be used to configure and combine the services. For example the already mentioned Django image can be executed and linked together with a MongoDB[15] images. The main benefit is the ease of configure dependencies between several containers and configuration steps. These concept is similar to deployment tools like Vagrant[16], Ansible[17] or

[13] `https://hub.docker.com`
[14] `https://store.docker.com`
[15] `https://www.mongodb.com`
[16] `https://www.vagrantup.com`
[17] `https://www.ansible.com`

Puppet[18].

A major benefit of Docker is that the execution environment of an application is completely the same on a local machine as on the production environment.[cf. Gal15, p. 2] There is no need to do things differently when switching from a development environment like a local machine, to a production environment like a server.[cf. Gal15, p. 2]

2.3.3 Kubernetes

Kubernetes is an open source container cluster manager which was released 2014 by Google. It is "a platform for automating deployment, scaling, and operations"[Gra15, p. 1] of containers. Therefore a cluster of containers can be created and managed. The system can for example schedule on which node a container should be executed, handle node failures, can scale the cluster by adding or removing nodes or enable rolling updates.[Gra15, p. 5 f.]

Figure 6 illustrates the basic architecture of Kubernetes. The user can interact with the Kubernetes system via a CLI, an User Interface (UI) or a third party application, over a REST API to the Kubernetes Master or more specifically the API Server in the master. The master himself controls the one or multiple nodes, monitor the system, schedule resources or pull new images from the repository, to name only a few tasks.

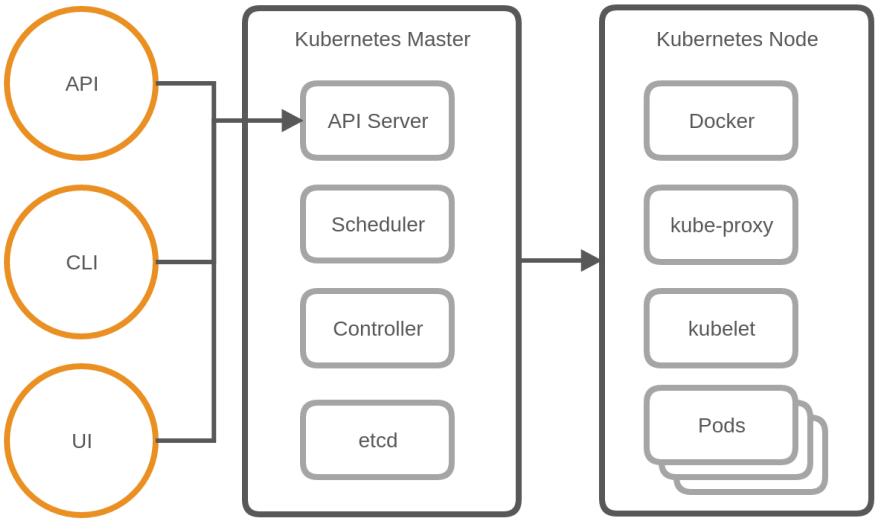

Figure 6: Kubernetes architecture. Adapted from: [MSV16, p. 4]

Each node has a two way communication with the master via a kubelet. In addition each node has the services necessary to run container applications like Docker. Furthermore Kubernetes can combine one or multiple containers into one so called Pod.[cf. Mul16, p. 7] "Pods are always co-located and co-scheduled, and run in a shared context."[Kub16c] One node again can execute multiple Pods. Pods are only temporary grouped containers with a non-stable Internet Protocol (IP) address. After a Pod is destroyed it can never be resurrected. Pods can also share functionality to other Pods inside a Kubernetes cluster. A logical set of Pods and the access policy of them is called a Kubernetes service. Such a service can abstract multiple

[18] `https://puppet.com`

Pod replicas and manage them. A frontend which have access to the service do not care about changes in the service. Any change, be it a down scale or an up scale of the system, remains unseen for the frontend. They are exposed through internal or external endpoints to the users or the cluster.[cf. MSV16, p. 11] Labels can be used to organize and to select subsets of Kubernetes Objects, such as Pods or Services.[Kub16b, cf.] They are simply key/value pairs which and should be meaningful and relevant to users, but do not imply semantics to the core system.[Kub16b, cf.]

The kube-proxy is a network proxy and load balancer which is accessible from the outside of the system via a Kubernetes service.[cf. Mul16, p. 7] "Each node and can do simple TCP,UDP stream forwarding or round robin TCP,UDP forwarding across a set of backends."[Kub16a] The Replication Controller is one of the mayor controllers in a Kubernetes System. It ensures that a specified number of pod replicas are running and available at any time.[Kub16d, cf.] If for example one node disappear because of connection issues, the Replication Controller will start a new one. If the disappeared node is available back again it will kill a node. These functionality increases the stability, the availability and the scalability of the system in an autonomous manner. The last important component in Kubernetes are rolling updates. With rolling updates the system can update one pod at a time, rather than taking down the entire service and update the whole system.[Kub16e, cf.] This also increases the stability and availability of the system and eases the managing of container clusters.

2.3.4 Docker Swarm

The basic functionality of Docker Swarm is pretty similar to Kubernetes: It is possible to create, manage and monitor a cluster of multiple machines running Docker on it. Before Docker version 1.12.0, Docker Swarm was an independent tool, which is now integrated in the Docker Engine.[Docc, cf.] No additional software is necessary to have a bunch of machines work together as a so called swarm. Similar to Kubernetes, Docker Swarm needs a master node called manager and several worker nodes. The manager for example keep track of the nodes and their lifecycle and it can start new instances of an image if one or multiple nodes disappear. Furthermore Docker Swarm has a build in proxy and load balancer, which can redirect requests to the node with the necessary container running on it or redirect requests based on the workload of the machines. Compared to Kubernetes Docker Swarm is more lightweight, but misses some features like the label functionality or the schema definition of a pod. But as mentioned before, both tools are pretty similar and aim for the same goal.

2.3.5 Open Baton

Open Baton is an open source ETSI NFV compliant Management And Orchestration (MANO) Framework[Opea, cf.]. "It enables virtual Network Services deployments on top of heterogeneous NFV Infrastructures."[Opea] It works together with OpenStack and provides a plugin mechanism which allows to add additional VIMs.[Opea, cf.] "OpenStack is a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter"[Opeb] In Open Baton it is implemented as the VIM as first Point of Presence (PoP).[Opea] All the resources in the NFVI are controlled by the VIM, in this case OpenStack.

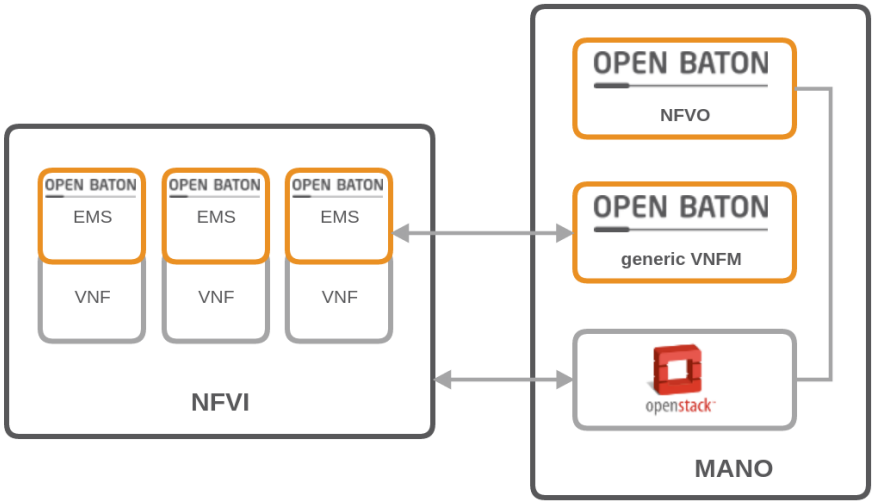

Figure 7: Open Baton abstract architecture. Adapted from: [Opea]

In the basic configuration Open Baton provides a generic VNFM, but it can also be replaced with a custom VNFM. The manager can use a REST API or an Advanced Message Queuing Protocol (AMQP) message queue to communicate with the system. Additional components, such as autoscaling and fault management, are provided to manage to the runtime of a network service.[Opea] The necessary informations are deliverd from the monitoring system available at the NFVI level, which can also be extended or replaced by a custom monitoring system.[Opea] Figure 7 illustrates the abstract architecture of Open Baten together with OpenStack and a generic VNFM.

The VIM Driver mechanism allow to integrate external heterogeneous PoPs without modification to the orchestration logic. The provided OpenStack driver uses the OpenStack APIs for requesting virtual compute and networking resources. Following the ETSI NFV specification the OpenStack API is just one of the VIM interfaces implementation.[Opea]

Zombie ipsum reversus ab viral inferno nam rick grimes malum cerebro. De carne lumbering animata corpora quaeritis. Summus brains sit morbo vel maleficia? De apocalypsi gorger omero undead survivor dictum mauris. Hi mindless mortuis soulless creaturas, imo evil stalking monstra adventus resi dentevil vultus comedat cerebella viventium. Qui animated corpse, cricket bat max brucks terribilem incessu zomby. The voodoo sacerdos flesh eater, suscitat mortuos comedere carnem virus. Zonbi tattered for solum oculi eorum defunctis go lum cerebro. Nescio brains an Undead zombies. Sicut malus putrid voodoo horror. Nigh tofth eliv ingdead.

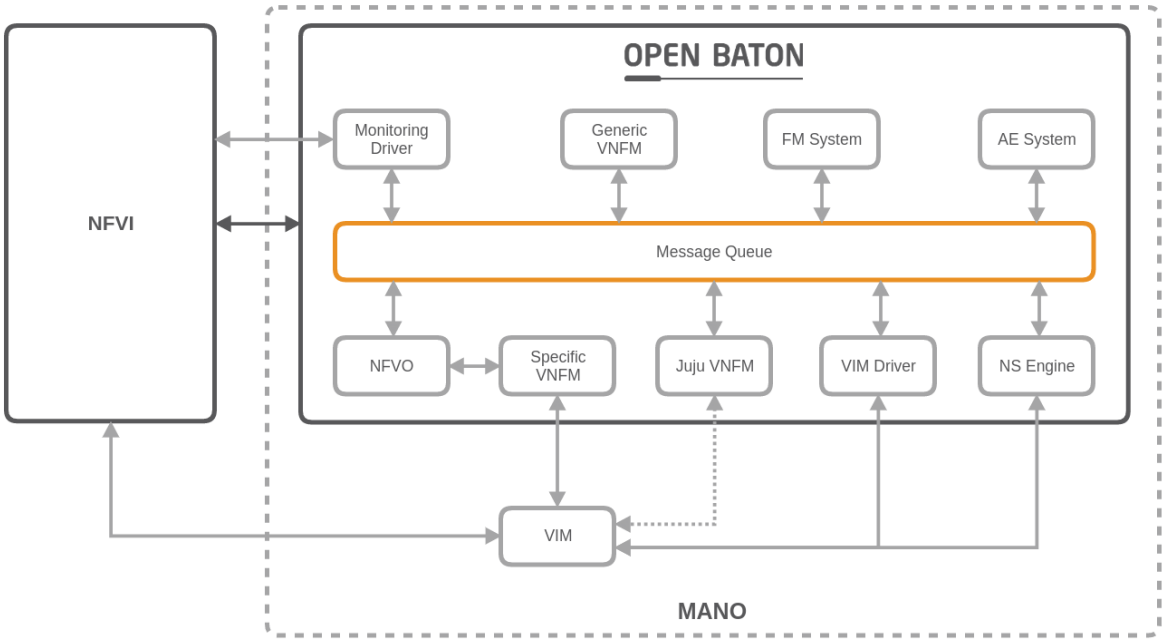

Figure 8: Open Baton detailed architecture. Adapted from: [Opea]

Basic function: Dashboard -> user input (instanciate network service) -> submited to NFVO -> NFVO request instanciation of NS at VIM (OpenStack) -> VIM starts virtual machines for the VNFs -> after booting, EMS is installed -> EMS (by OpenBaton) communicate with VNFM -> Open Baton sends lifecylce events to all VNFMs responsible of the VNFs part of the NS -> VNFM processes the VNFs via the EMS on to the given resources of the VNFI on the Datacenter -> Services started

2.4 Conclusion

Zombie ipsum reversus ab viral inferno nam rick grimes malum cerebro. De carne lumbering animata corpora quaeritis. Summus brains sit morbo vel maleficia? De apocalypsi gorger omero undead survivor dictum mauris. Hi mindless mortuis soulless creaturas, imo evil stalking monstra adventus resi dentevil vultus comedat cerebella viventium. Qui animated corpse, cricket bat max brucks terribilem incessu zomby. The voodoo sacerdos flesh eater, suscitat mortuos comedere carnem virus. Zonbi tattered for solum oculi eorum defunctis go lum cerebro. Nescio brains an Undead zombies. Sicut malus putrid voodoo horror. Nigh tofth eliv ingdead. Zombie ipsum reversus ab viral inferno nam rick grimes malum cerebro. De carne lumbering animata corpora quaeritis. Summus brains sit morbo vel maleficia? De apocalypsi gorger omero undead survivor dictum mauris. Hi mindless mortuis soulless creaturas, imo evil stalking monstra adventus resi dentevil vultus comedat cerebella viventium. Qui animated corpse, cricket bat max brucks terribilem incessu zomby. The voodoo sacerdos flesh eater, suscitat mortuos comedere carnem virus. Zonbi tattered for solum oculi eorum defunctis go lum cerebro. Nescio brains an Undead zombies. Sicut malus putrid voodoo horror. Nigh tofth eliv ingdead.

Chapter 3

Requirements Analysis

3.1 Overview

3.2 Technical requirements

3.3 Technologies

3.4 Use-Case-Analysis

3.5 Delineation from existing solutions

3.6 Conclusion

Chapter 4

Concept

4.1 Overview

4.2 Development environment

4.3 Architecture of the system

4.3.1 Orchestration layer

4.3.2 Constraint layer

4.3.3 User interface

4.4 Conclusion

Chapter 5

Implementation

5.1 Environment

5.2 Project structure

5.3 Used external libraries

5.4 Important Implementation Aspects

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 Test Environment

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

Conclusion

7.1 Summary

7.2 Dissemination

7.3 Impact

7.4 Outlook

Acronyms

AMQP Advanced Message Queuing Protocol
API Application Programming Interface
BSS Business Support Systems
CapEx Capital Expenditure
CI Continuous Integration
CLI Command Line Interface
CPU Central Processing Unit
CPS Cyber-Physical System
CS Cyber System
EMS Element Management System
ETSI European Telecommunications Standards Institute
H2H Human-to-Human
H2M Human-to-Machine
IERC European Research Cluster on the Internet of Things
IoS Internet of Services
IoT Internet of Things
IP Internet Protocol
IT Information Technology
JVM Java Virtual Machine
LXC Linux Containers
M2M Machine-to-Machine
MANO Management And Orchestration
NF Network Function
NFV Network Function Virtualistion
NFVI Network Function Virtualization Infrastructure
NFV-MANO Network Function Virtualistion Management And Orchestration
NFVO Network Function Virtualistion Orchestrator
NIST National Institute of Standards and Technology
OASIS Organization for the Advancement of Structured Information Standards
OpEx Operating Expense
OS Operating System
OSS Operations Support Systems
PNF Physical Network Function
PoP Point of Presence
RAM Random Access Memory
REST Representational State Transfer
RFID Radio Frequency Identification
SDN Software Defined Networking
TOSCA Topology and Orchestration Specification for Cloud Applications
UI User Interface
VIM Virtual Infrastructure Manager
VM Virtual Machine
VMM Virtual Machine Monitor
VNF Virtual Network Function
VNFM Virtual Network Function Manager
YAML YAML Ainť Markup Language

Glossary

Algorithmus a

Chiffrierung a

Dechiffrierung a

Bibliography

[AHA+16] Anderson, J. ; Hu, H. ; Agarwal, U. ; Lowery, C. ; Li, H. ; Apon, A.: Performance considerations of network functions virtualization using containers. In: *2016 International Conference on Computing, Networking and Communications (ICNC)*, 2016, S. 1–7

[BCJ+16] Bosche, A. ; Crawford, D. ; Jackson, D. ; Schallehn, M. ; Smith, P.: *How Providers Can Succeed in the Internet of Things*. http://bain.com/publications/articles/how-providers-can-succeed-in-the-internet-of-things.aspx. Version: Aug 2016. – Accessed: 2017-02-20

[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

[CMF+16] Celesti, A. ; Mulfari, D. ; Fazio, M. ; Villari, M. ; Puliafito, A.: Exploring Container Virtualization in IoT Clouds. In: *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2016, S. 1–6

[Doca] *Docker Overview - Docker*. https://docs.docker.com/engine/understanding-docker/, . – Accessed: 2017-02-24

[Docb] *Understand images, containers, and storage drivers - Docker*. https://docs.docker.com/engine/userguide/storagedriver/imagesandcontainers/, . – Accessed: 2017-02-24

[Docc] *Docker Swarm Documentation*. https://docs.docker.com/engine/swarm, . – Accessed: 2017-03-18

[ETS13] ETSI - European Telecommunications Standards Institute: Network Function Virtualistion (NFV); Architectural Framework. Version: Oct 2013. http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf. 2013. – Forschungsbericht. – Accessed: 2017-02-27

[Eva11] Evans, D.: The Internet of Things: How the Next Evolution of the Internet Is Changing Everything / Cisco Internet Business Solutions Group (IBSG). Version: April 2011. http://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. 2011. – Forschungsbericht. – Accessed: 2017-

02-12

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

[Gra15] Grant, B.: *Kubernetes: a platform for automating deployment, scaling, and operations*. https://www.slideshare.net/BrianGrant11/wso2con-us-2015-kubernetes-a-platform-for-automating-deployment-scaling-and-oper Version: Nov 2015. – Accessed: 2017-02-27

[HPO15] Hermann, M. ; Pentek, T. ; Otto, B.: Design Principles for Industrie 4.0 Scenarios: A Literature Review / Technische Universität Dortmund - Fakultät Maschinenbau. Version: Jan 2015. http://www.thiagobranquinho.com/wp-content/uploads/2016/11/Design-Principles-for-Industrie-4_0-Scenarios.pdf. 2015. – Forschungsbericht. – Accessed: 2017-02-12

[Itu05] International Telecommunication Union: ITU Internet Reports: The Internet of Things. Version: November 2005. https://www.itu.int/net/wsis/tunis/newsroom/stats/The-Internet-of-Things-2005.pdf. 2005. – Forschungsbericht. – Accessed: 2017-02-12

[Jur13] Jurevicius, O.: *Vertical Integration*. https://www.strategicmanagementinsight.com/topics/vertical-integration.html. Version: April 2013. – Accessed: 2017-02-13

[Kah15] Kahn, F.: *A Cheat Sheet for Understanding "NFV Architecture"*. http://www.telecomlighthouse.com/a-cheat-sheet-for-understanding-nfv-architecture. Version: Mar 2015. – Accessed: 2017-03-19

[KKL13] Kramp, T. ; Kranenburg, R. van ; Lange, S.: Introduction to the Internet of Things. In: *Enabling Things to Talk* Bd. 1, Springer-Verlag Berlin Heidelberg, 2013. – ISBN 978–3–642–40402–3, S. 1–10

[Kub16a] *kube-proxy - Kubernetes*. https://kubernetes.io/docs/admin/kube-proxy. Version: Dec 2016. – Accessed: 2017-03-03

[Kub16b] *Labels and Selectors - Kubernetes*. https://kubernetes.io/docs/user-guide/labels. Version: Dec 2016. – Accessed: 2017-03-03

[Kub16c] *Pods - Kubernetes*. https://kubernetes.io/docs/user-guide/pods. Version: Dec 2016. – Accessed: 2017-03-03

[Kub16d] *Replication Controller - Kubernetes*. https://kubernetes.io/docs/user-guide/replication-controller. Version: Dec 2016. – Accessed: 2017-03-03

[Kub16e] *Rolling Updates - Kubernetes*. https://kubernetes.io/docs/user-guide/rolling-updates. Version: Dec 2016. – Accessed: 2017-03-03

[Lee08] Lee, E. A.: Cyber Physical Systems: Design Challenges. In: *2008 11th IEEE International Symposium on Object and Component-Oriented Real-Time Dis-*

tributed Computing (ISORC), 2008. – ISSN 1555–0885, S. 363–369

[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

[Lyd16] Lydon, B.: Industry 4.0: Intelligent and flexible production. In: *InTech Magazine* (2016), May-June. https://www.isa.org/intech/20160601/. – Accessed: 2017-02-13

[MSV16] MSV, J.: *Kubernetes architecture.* https://www.slideshare.net/janakiramm/kubernetes-architecture. Version: Oct 2016. – Accessed: 2017-03-03

[Mul16] Mulyana, E.: *Kubernetes Basics.* https://www.slideshare.net/e2m/kubernetes-basics. Version: May 2016. – Accessed: 2017-03-03

[NFV] *NFV.* https://sdn-wiki.fokus.fraunhofer.de/doku.php?id=nfv. – Accessed: 2017-03-19

[Nob15] Noble, S.: *Network Function Virtualization or NFV Explained.* http://wikibon.com/network-function-virtualization-or-nfv-explained. Version: Apr 2015. – Accessed: 2017-03-03

[Opea] *OpenBaton Documentation.* http://openbaton.github.io/documentation, . – Accessed: 2017-03-18

[Opeb] *Software » OpenStack Open Source Cloud Computing Software.* https://www.openstack.org/software, . – Accessed: 2017-03-21

[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

[Riv14] Rivenes, L.: *What is Network Function Virtualization (NFV)?* https://datapath.io/resources/blog/network-function-virtualization-nfv. Version: Sep 2014. – Accessed: 2017-03-03

[SSH11] Scarfone, K. ; Souppaya, M. ; Hoffman, P.: Guide to Security for Full Virtualization Technologies / National Institute of Standards and Tech-

nology. Version: Jan 2011. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-125.pdf. 2011. – Forschungsbericht. – Accessed: 2017-02-19

[Tos] *Why is TOSCA Relevant to NFV? Explanation.* https://www.sdxcentral.com/nfv/definitions/tosca-nfv-explanation. – Accessed: 2017-03-19

[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

[Vbw14] vbw Vereinigung der Bayerischen Wirtschaft e. V.: Dienstleistungspotenziale im Rahmen von Industrie 4.0. Version: Mar 2014. http://www.forschungsnetzwerk.at/downloadpub/dienstleistungspotenziale-industrie-4.0-mar-2014.pdf. 2014. – Forschungsbericht. – Accessed: 2017-02-12

[Wei91] Weiser, M.: *The Computer for the 21st Century.* http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html. Version: September 1991. – Accessed: 2017-02-12

[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