Overcome Vendor Lock-In by Integrating Already Available Container Technologies Towards Transferability in Cloud Computing for SMEs

Peter-Christian Quint, Nane Kratzke
Lübeck University of Applied Sciences, Center of Excellence CoSA
Lübeck, Germany
email: {peter-christian.quint, nane.kratzke}@fh-luebeck.de

***Abstract*—Container clusters have an inherent complexity. A distributed container application in the cloud can be complex at planning, installation and configuration, maintenance and search for failures. Small and medium enterprises (SMEs) are mostly limited by their personnel and financial restrictions. Using advanced cloud technologies like a container cluster often requires high personnel expenses or the use of an external system builder. In addition to economical, security- and governance issues there is also the concern of technical vendor lock-ins. This paper introduces C^4S, an open source system for SMEs to deploy and operate their container application with features like elasticity, auto-scaling and load balancing. The system also supports transferability features for migrating container between different Infrastructure as a Service (IaaS) platforms. This paper presents a solution for SMEs to use the benefits of cloud computing without the disadvantages of vendor lock-in.**

Keywords–Microservice; Container; Docker; Container Cluster; Software Defined Network; Cloud Computing; SME

I. INTRODUCTION

Infrastructure as a service (IaaS) enables companies to get resources like computational power, storage and network connectivity on demand. IaaS can be obtained on public or private clouds. Public clouds are provided by third parties for general public use. Type representatives are Amazon's Elastic Compute Cloud (EC2) and Google Compute Engine (GCE). Private Cloud are intended for the exclusive use by a single organization [1]. They are mostly installed on the respective companies own infrastructure. OpenStack is a cloud platform for providing (not exclusively) private clouds. One big benefit using cloud computing is the elastic scaling. Elasticity means the possibility to match available resources with the current demands as closely as possible [2]. Scalability is the ability of the system to accommodate larger loads by adding resources or accommodate weakening loads by removing resources [3]. With autoscaling, resources can be added automatically when they are needed and removed when they are not in use [4]. The resources are allocated on demand and the customer only has to pay for resources he really used. The system described in this paper will support several, public and private, cloud environments. Features like elastic scaling and transferability will also be available. The authors define transferability as the possibility to migrate some or all containers between different cloud platforms. This is needed to avoid vendor lock-in by the cloud providers, which is a major obstacle for small and medium enterprises (SMEs) in cloud computing [5]. Only a few research projects deal with the specific needs of SMEs in cloud computing [6].

In the last few years, container technologies like Docker got more and more common. Docker is an open source and lightweight virtualization solution to provide an application deployment without having the overhead of virtual machines [7]. With Docker, applications can be easily deployed on several machine types. This makes it possible to launch containers from the same application (image), e.g., on a personal computer or a datacenter server.

Container clusters like Kubernetes (arose from Google Borg [8]) and Mesos [9] can deploy a huge number of containers on private and public clouds. A big benefit of cluster technologies is the horizontal scalability of the containers, the fast development and the contained software defined network, which is often necessary for distributed container applications. Container and container cluster software are mostly open source and free to use.

SMEs are mostly financially and personnel-wise restricted (see the European definition of SME [10]). Since the management of container cluster applications with features like transferability and elasticity is complex, it can be very difficult to achieve for a small (maybe only one person size) IT department. Getting started using services like Infrastructure as a Service (IaaS) might be very simple. But the use of advanced cloud technologies like clusters, containers and cloud benefits like auto-scaling and load balancing can quickly grow into complex technical solutions. The cloud provider supplied services (i.e., auto-scaling) might pose another issue due to often having non-standardized service APIs. This is often resulting in inherent vendor lock-in [11]. However, there are products and services to manage these technologies like the T-Systems Cloud Broker and Amazon EC2 Container Service (ECS). These management solutions also have disadvantages. For example, the Cloud Broker is a commercial product, which is inherently designed for very big companies. This kind of cloud broker services move the dependencies from the cloud provider to the system/service provider like T-Systems. ECS works only with Amazon EC2-instances, which means there is still a vendor lock-in. Both solutions just shift vendor lock-in to another company. Creating an open source system for easy deployment and managing of cloud applications in a container cluster would support SMEs using these technologies without worries about vendor lock-in.

The software presented in this paper is called C^4S. The name is an acronym for Container Cluster in Cloud Computing System. C^4S is designed to (automatically) deploy an operating container cluster application without vendor lock-in. Moreover, the system will be able to monitor the cloud platform, the container cluster and the containers themselves. Beside bare reporting, the system will offer methods to keep the application running in most failure states. Altogether, the C^4S can make container cluster cloud computing technologies usable for SMEs without large and highly specialized IT departments.

 ISBN: 978-1-61208-460-2

C^4S is not exclusively designed for SMEs. The user group of C4S is not limited to specific company types, so (e.g., big international) companies with small and specific in-house-departments can use it, too.

Section II describes the features and the requirements of the C^4S system. An overview about the C^4S architecture is given in Section III. The intended validation of concept, which is performed at the final phase, is presented in Section IV. Related work, several business services and products are described in Section V. Finally, the conclusion is presented in SectionVI.

II. GENERAL REQUIREMENTS OF C^4S

C^4S is designed to handle the high complexity of a container cluster with benefits like elasticity, auto-scaling and transferability. Feature requirements and the technical specifications are explained below. By designing and developing a generic cloud service description language, a solution to define secure, transferable and elastic services of typical complexity will be provided. Thus, they are deployable to any IaaS cloud infrastructure. This work promotes the implementation of easy-to-handle, elastic and transferable cloud applications.

A. Deploying and Controlling Applications

The basic feature of C^4S is to deploy a distributed container application on cloud environments. Therefore, the user can easily configure the needed containers, the interfaces and the cloud environments. According to the user set configuration, the system will automatically deploy the application on the container cluster and the cloud platforms. Overall, there are three controlling levels the management solution C^4S has to support:

The **Container Application** can be configured, launched, controlled, changed and stopped. Application parts (e.g., container types) must be replaceable (e.g., changing of container images to keep the application up to date). Details like the status and the configuration of every single container and an overview of all running containers should also be visible for the customers.

The **Container Cluster** is used for automatic deployment of the container on the available virtual machines (running on IaaS platforms). Therefore, the cluster solution can create, terminate and transfer containers. The user is able to set values like the deployment limitations and restrictions in the container host selection.

Virtual Machines on IaaS platforms form the host system of the container cluster with the containers. The system should support a management solution for monitoring the status as well as creating and terminating virtual machines on several cloud platforms. The user is able to set values like the virtual machine limitations and the favored platform for each container type.

A view with all used machines, their type, running time, and other data is also necessary to observe, e.g., economic information like actual costs. The system has to be able to monitor on all controlling levels. In case of failure, the monitoring system should trigger reports and automatic actions to keep the application running.

B. Usage of Cloud Features

As described in Section I, cloud computing enables features like **elasticity**, **scalability** and **load balancing**. C^4S enables the user to handle the inherent complexity of these features in an easy way. **Auto-Scaling** containers on the cluster and virtual machines on IaaS platforms is also featured.

C. Prevent Dependencies

To avoid **vendor lock-in by the cloud provider**, the system can install a multi-cloud container cluster with transferability features. On demand, some or all containers can migrate from one cloud provider to another. Accordingly, the user has full control over where the containers are running.

To **prevent dependencies by used software and services**, the C^4S will be published under MIT license. It is recommended that all third-party parts like the cluster software are also open source. Thus, the consuming companies avoid dependencies to the C^4S system and adapt the source code to their special needs. The system has to be designed generic for several cloud platforms. A modular architecture enables extensions for other platforms. Beside the cloud platforms, the users should not be limited by the choice of the container cluster. The modular architecture enables later extensions for missing cluster connectivity.

III. C^4S ARCHITECTURE

The architecture is divided into four layers. The core of C^4S is the deployment and the monitoring engine. The user can manage the deployment and get the monitoring events over two interfaces. The other two parts are the container cluster and the IaaS environments.

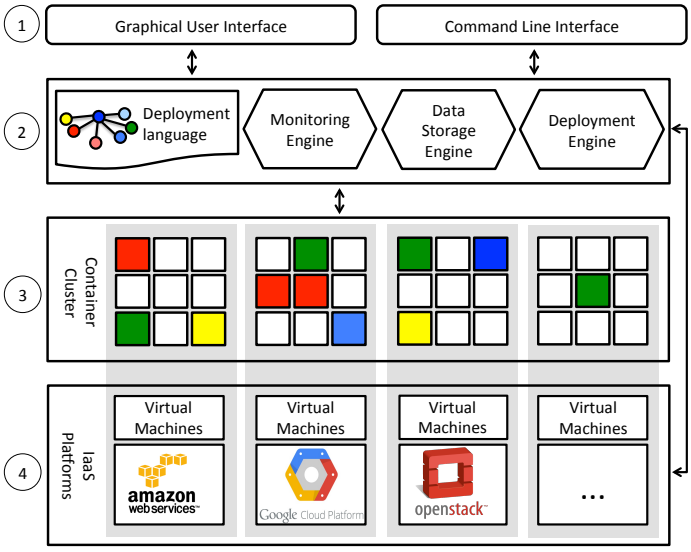

Figure 1. C^4S architecture overview

A. Interfaces ①

The management system will provide a web-based graphical user interface (GUI) and a command line interface (CLI), see Figure 1, ①. Here the user can set the the account data and limits of the IaaS platforms, the configuration for scaling and also set transfer orders (e.g., moving containers to another

cloud platform). It is also possible to set rules for creating, terminating and moving containers on a special event. By setting these rules, the system can deal with failures like a cloud platform interrupt. Other features like the automatic container cluster software installation can be started using the interfaces. Status information about the containers, the cluster and the cloud platform are visible, monitoring events and the triggered actions are also reported.

B. Deployment and Monitoring Engines ②

This subsection describes the engines shown at Figure 1, ②. The deployment engine is responsible for creating and deleting instances on the cloud platforms, managing the container cluster and deployment of the containers on it, making it the main core of $\boldsymbol{C}^4\boldsymbol{S}$. Features like load balancing and container transfer are also controlled by this engine. The deployment language is designed for stating the constraints of the configuration. All needed informations about the container application, the cluster and the IaaS platforms can be described by the deployment language. The monitoring engine observes the containers on the cluster and virtual machines on the IaaS platform. Additional to reporting states and failures, actions can be triggered by events. For example, if the engine registers an exceptionally high workload of a container, it reports to the deployment engine to scale out containers, and vice versa. The engine can deploy the application on different cloud platforms and is able to transfer containers between them. The data storage engine is compatible with several block and object storage systems to avoid vendor lock-in. The engine also enables scalability and security features for data storage.

C. Container Cluster ③

The container cluster deploys the containers on the virtual cloud machines (see Figure 1, ③). The management system can terminate and create the containers in the cluster network. The system can also transfer a container from one virtual machine to another, which is not even necessarily running on the same cloud platform. These actions are controlled by the deployment engine and supervised by the monitoring engine. In combination with the container cluster, the engines make it possible to migrate services from one private or public cloud infrastructure to another (not necessarily compatible) cloud infrastructure.

D. IaaS Platforms ④

The $\boldsymbol{C}^4\boldsymbol{S}$ can manage several cloud platforms (see Figure 1, ④). It is possible to create and terminate IaaS instances accordingly on demand. Hence, it has to communicate with the cloud platforms. Because of missing standardization [12] for every provider API a special driver has to be designed. These require a modular and easy to adapt software architecture. The deployment language is designed for informations like access data, limits and all other settings which can be set over the described interfaces.

IV. INTENDED VALIDATION OF CONCEPT

It will be shown that SMEs can manage a container cluster over (multi) cloud platforms. At first it will be demonstrated that building a system, which fits all the required features, is possible. Therefore, a working, open source $\boldsymbol{C}^4\boldsymbol{S}$ prototype, which conforms the requirements set in Section II, will be developed. The system has to be implemented in a modular and extendable way. As cluster platform, $\boldsymbol{C}^4\boldsymbol{S}$ will support Kubernetes first, other cluster environments will follow. Presenting interchangeability and the open source type of $\boldsymbol{C}^4\boldsymbol{S}$ will show that dependencies by the used software can be prevented. To avoid vendor lock-in by the cloud provider, the prototype must be able to install a multi-cloud container cluster. First, the system will be compatible with the IaaS cloud platform type representatives Amazon EC2, Google GCE and OpenStack. To support other platforms, appropriate drivers can be implemented. Transferability features like moving all containers from one cloud platform to another will be implemented. Terminating all containers and virtual machines on one provider and creating them on another at the same time, without changes in features like elasticity and auto-scaling, will proof that $\boldsymbol{C}^4\boldsymbol{S}$ is preventing vendor lock-in. The software will also manage container application deployment. It will deploy a container cluster, create and terminate containers and is usable for deploying applications. Also, workloads will be created to test the autoscaling features. With enforced failure states, the robustness of the system will be demonstrated. It will be shown that the system is able to keep the applications running even when containers and virtual machines get disconnected. In the second part of the proof of concept, a company will employ the software. Thus, the expense for a small business using the container cluster manager will be evaluated. Finally, a proof of concept will be realized by several business companies. These companies will use the $\boldsymbol{C}^4\boldsymbol{S}$ system on their own for testing a productive application deployment with real workload. Load balancing, elasticity, auto-scaling and transferability features will be applied in production. This way it will be shown that SMEs can handle the complexity of a container cluster application running on multiple cloud platforms without vendor lock-in or dispensing with features like auto-scaling.

V. RELATED WORK

There are several solutions with overlapping features and/or usage scenarios available. However, a system which fits all requirements and features set in Section II for the $\boldsymbol{C}^4\boldsymbol{S}$ deployment manager does not exist.

A. Container Cluster, Load Balancing and Scaling

A Container Cluster should run on homogeneous machine types to provide fine-grained resource allocation capabilities [9]. In previous work, the similarity of different cloud provider instance types was analyzed. It was concluded that only a few instance pairs are really similar. There are just a few virtual machine types, which should be used when running an application on a multi cloud platform environment [13]. Another issue to consider is the network performance impacts using technologies like container and software defined networks (SDN). Previous investigations found that performance impacts depends i.e., on the used machine types and the message sizes [14]. Using a (encrypted) cross-provider SDN also causes performance impacts, especially when using low-core machines [15].

B. IaaS Management and Transferability

Container migration from one cloud provider to another is an important feature of $\boldsymbol{C}^4\boldsymbol{S}$. Vendor lock-in is caused, i.a., by a lack of standards [12]. Currently the proprietary EC2 is the de-facto standard specification for managing cloud infrastructure.

 ISBN: 978-1-61208-460-2

However, open standards like OCCI and CIMI are important to reduce vendor lock-in situations [16]. C^4S includes a special IaaS driver for each supported cloud provider. Other research approaches in cloud migration can be reviewed under [17]. There are several solutions like Apache Libcloud, KOALA [18], Scalr, Apache jclouds and deltacloud and T-Systems Cloud Broker for managing and deploying virtual machines on IaaS platforms. Except for the T-Systems Cloud Broker, the solutions are open source but have mostly payable services, reduced functionality or limited virtual machine quantities. These systems support features like creating, stopping and scaling virtual machines on IaaS cloud platforms. Some of them like the T-Systems Cloud Broker, Scalr and Apache jclouds are designed for cross-platform IaaS deployment. In contrast to the C^4S requirements, the presented cloud managers are limited to IaaS managing and do not offer container deploying services. Some of them do not prevent vendor lock-in by cloud providers or create new dependencies by itself (e.g., T-System Cloud Broker, KOALA is limited to Amazon AWS API compatible services).

C. Application Deployment

Peinl et al. [19] has defined requirements for a container application deploying system. These coincide strongly with the requirements for the C^4S system, which have been discussed in Section II-A. He also gives an overview about container cluster managing. For easy deployment a container application with monitoring, scaling and controlling benefits, there exist several commercial solutions like the Amazon EC2 Container Service (ECS), Microsoft Azure Container Service and Giant Swarm. Limited to the providers own IaaS infrastructure, these solutions are not designed for multi-cloud platform usages, especially between public clouds (a requirement of C^4S). Open source cluster managers are Apache Mesos and Kubernetes. These systems are designed to run workloads across tens of thousands of machines. The benefits and issues using cluster technologies are very high reliability, availability and scalability [9] [8]. However, they are not designed to create and terminate virtual machines (like AWS instances), but to deploy applications on given resources. So, they cannot prevent cloud provider dependencies on their own, but provide essential ingredients to do so. Another cluster management tool for increasing the efficiency of datacenter servers is called Quasar, which is developed by the Stanford University and designed for maximizing resource utilization. The system performs coordinated resource allocation. Several techniques analyze performance interferences, scaling (up and out) and resource heterogeneity [20].

VI. CONCLUSION

The C^4S is in the planning state, although some parts are already implemented, like the cloud platform driver for fast deploying IaaS instances. The next step is the creation of a deployment language for dedicated containers to run on a Kubernetes container cluster, finding solutions for container cluster scaling problems and handling stateful tasks like file storage. The system will be implemented in a modular and generic way to allow an easy adaptation for different cloud platforms and container cluster software. With C^4S SMEs will be able to deploy and operate their container applications on an elastic, auto-scaling and load balancing multi-cloud cluster with transferability features to prevent vendor lock-in.

ACKNOWLEDGMENT

This research is funded by German Federal Ministry of Education and Research (Project Cloud TRANSIT, 03FH021PX4). The author thank the University of Lübeck (Institute of Telematics) and fat IT solution GmbH (Kiel) for their support of Cloud TRANSIT.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing," 2011.

[2] M. Armbrust et al., "A view of cloud computing," Communications of the ACM, vol. 53, no. 4, 2010, pp. 50–58.

[3] L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," ACM SIGCOMM Computer Communication Review, vol. 41, no. 1, 2011, pp. 45–52.

[4] M. Mao and M. Humphrey, "Auto-scaling to minimize cost and meet application deadlines in cloud workflows," in Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis. ACM, 2011, p. 49.

[5] N. Kratzke, "Lightweight virtualization cluster - howto overcome cloud vendor lock-in," Journal of Computer and Communication (JCC), vol. 2, no. 12, oct 2014.

[6] R. Sahandi, A. Alkhalil, and J. Opara-Martins, "Cloud computing from smes perspective: A survey-based investigation," Journal of Information Technology Management, vol. 24, no. 1, 2013, pp. 1–12.

[7] J. Turnbull, The Docker Book: Containerization is the new virtualization. James Turnbull, 2014.

[8] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in Proceedings of the Tenth European Conference on Computer Systems. ACM, 2015, p. 18.

[9] B. Hindman et al., "Mesos: A platform for fine-grained resource sharing in the data center." in NSDI, vol. 11, 2011, pp. 22–22.

[10] Definition recommendation of micro, small and medium-sized enterprises by the european communities. Last access 12th Nov. 2015. [Online]. Available: http://eur-lex.europa.eu/legal-content/EN/TXT/?uri=uriserv:OJ.L_.2003.124.01.0036.01.ENG

[11] N. Kratzke, "A lightweight virtualization cluster reference architecture derived from open source paas platforms," Open J. Mob. Comput. Cloud Comput, vol. 1, 2014, pp. 17–30.

[12] J. Opara-Martins, R. Sahandi, and F. Tian, "Critical review of vendor lock-in and its impact on adoption of cloud computing," in Information Society (i-Society), 2014 International Conference on. IEEE, 2014, pp. 92–97.

[13] N. Kratzke and P.-C. Quint, "How to operate container clusters more efficiently? some insights concerning containers, software-defined-networks, and their sometimes counterintuitive impact on network performance," International Journal on Advances in Networks and Services, vol. 8, no. 3&4, 2015, (in press).

[14] N. Kratzke and P.-C. Quint, "About automatic benchmarking of iaas cloud service providers for a world of container clusters," Journal of Cloud Computing Research, vol. 1, no. 1, 2015, pp. 16–34.

[15] N. Kratzke, "About microservices, containers and their underestimated impact on network performance," CLOUD COMPUTING 2015, 2015, p. 180.

[16] C. Pahl, L. Zhang, and F. Fowley, "Interoperability standards for cloud architecture," in 3rd International Conference on Cloud Computing and Services Science, CLOSER. Dublin City University, 2013, pp. 8–10.

[17] P. Jamshidi, A. Ahmad, and C. Pahl, "Cloud migration research: a systematic review," Cloud Computing, IEEE Transactions on, vol. 1, no. 2, 2013, pp. 142–157.

[18] C. Baun, M. Kunze, and V. Mauch, "The koala cloud manager: Cloud service management the easy way," in Cloud Computing (CLOUD), 2011 IEEE International Conference on. IEEE, 2011, pp. 744–745.

[19] R. Peinl and F. Holzschuher, "The docker ecosystem needs consolidation." in CLOSER, 2015, pp. 535–542.

[20] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and qos-aware cluster management," ACM SIGPLAN Notices, vol. 49, no. 4, 2014, pp. 127–144.

 ISBN: 978-1-61208-460-2