Automating the Provisioning and Integration of Analytics Tools with Data Resources in Industrial Environments using OpenTOSCA

Michael Zimmermann, Michael Falkenthal,
Frank Leymann

Institute of Architecture of Application Systems,
University of Stuttgart, Stuttgart, Germany
{lastname}@iaas.uni-stuttgart.de

Felix W. Baumann, Ulrich Odefey

TWT GmbH Science & Innovation,
Stuttgart, Germany
{firstname}.{lastname}@twt-gmbh.de

BibTeX:

```
@inproceedings{Zimmermann2017_ProvisioningAnalytics,
  author    = {Michael Zimmermann and Felix W. Baumann and Michael Falkenthal
               and Frank Leymann and Ulrich Odefey},
  title     = {Automating the Provisioning and Integration of Analytics Tools
               with Data Resources in Industrial Environments using OpenTOSCA},
  booktitle = {Proceedings of the 2017 IEEE 21st International Enterprise
               Distributed Object Computing Conference Workshops and
               Demonstrations (EDOCW 2017)},
  year      = {2017},
  pages     = {3--7},
  doi       = {10.1109/EDOCW.2017.10},
  publisher = {IEEE Computer Society}
}
```

Automating the Provisioning and Integration of Analytics Tools with Data Resources in Industrial Environments using OpenTOSCA

Michael Zimmermann,
Michael Falkenthal, Frank Leymann
Institute of Architecture of Application Systems
University of Stuttgart
70569 Stuttgart, Germany
Email: {lastname}@iaas.uni-stuttgart.de

Felix W. Baumann, Ulrich Odefey
TWT GmbH Science & Innovation
70565 Stuttgart, Germany
Email: {firstname}.{lastname}@twt-gmbh.de

***Abstract*—The fourth industrial revolution is driven by the integration and analysis of a vast amount of diverse data. Thereby, data about production steps, overall manufacturing processes, and also supporting processes is gathered to enable holistic analysis approaches. These approaches promise to provide new insights and knowledge by revealing cost saving possibilities and also automated adjustments of production processes. However, such scenarios typically require analytics services and data integration stacks since algorithms have to be developed, executed and therefore be wired with the data to be processed. This leads to complex setups of overall analytics environments that have to be installed, configured and managed according to the needs of different analysis scenarios and setups. The manual execution of such installations is time-consuming and error-prone. Therefore, we demonstrate how the different components of such combined integration and analytics scenarios can be modelled in order to be reused in different settings, while enabling the fully automated provisioning of overall analytics stacks and services.**

I. Introduction

The technological evolutions of cloud computing, the Internet of Things (IoT) and smart data analytics approaches from the fields of data mining and artificial intelligence have pioneered the 4th industrial revolution, which is also known as Industry 4.0 [1]. Industry 4.0 endeavours concern, among others, the optimization of production steps, whole production lines and processes along with proactive optimizations, such as machinery maintenance, based on the forecasting of failures and wear and tear. To support these goals, vast amounts of data must be collected and organized by data processing frameworks. This data is required to enable the application of machine learning and analytics algorithms to indicate and leverage opportunities, e.g., in the aforementioned disciplines and service areas. The combination of analytics algorithms, the data to be analyzed and the deployment models of the underlying infrastructures and components are called smart services [2].

However, since the data to be processed originates from manufacturing processes and production steps, it is typically of vital importance for the data creating and owning company. Furthermore, this data is commonly proprietary and classified, as it can contain sensitive information on business processes, customers, or equipment. Thus, access to this data has to be restricted, which can be achieved by specifying and enforcing appropriate data policies [3]. Such policies often prohibit that data is allowed to leave the company, implying that analytics algorithms have to be provisioned locally to the data. In addition, for accessing data from different data sources, such as manufacturing execution systems, production scheduling systems, or even machinery directly, a data integration middleware has to be employed. As a result, installation and configuration of the overall analytics stack is typically complex, time-consuming, and requires immense expertise, especially if it is performed manually every time when new analyses have to be executed [4].

Therefore, in this demonstration, we show how the OASIS standard *Topology and Orchestration Specification for Cloud Applications* (TOSCA) [5], [6] can be used to model analytics algorithms in association with their hosting execution environments as smart services. Thereby, we emphasize the separation of these components into deployment model fragments, which can be reused in arbitrary new configurations for automatic provisioning of entire analytics stacks. In addition, we demonstrate how data integration services can be employed into this deployment automation to enable wiring analytics services with different types of data sources. Moreover, we show how the specified deployment model containing these components can be provisioned fully automatic by means of OpenTOSCA, a standard-compliant runtime for deploying and managing TOSCA-based applications. As an exemplary setup of an analytics environment we use Apache Flink [7] as hosting environment for analytics services.

The remainder of this work is structured as following: in Section II we give a more comprehensive motivation for the automation of analytics deployments in Industry 4.0 endeavours. To understand the demonstrated modelling and automation approach we discuss the fundamental concepts of TOSCA in Section III. Finally, we describe the required setup and the demonstration itself in Section IV, then we conclude this work in Section VI by also giving directions for future work.

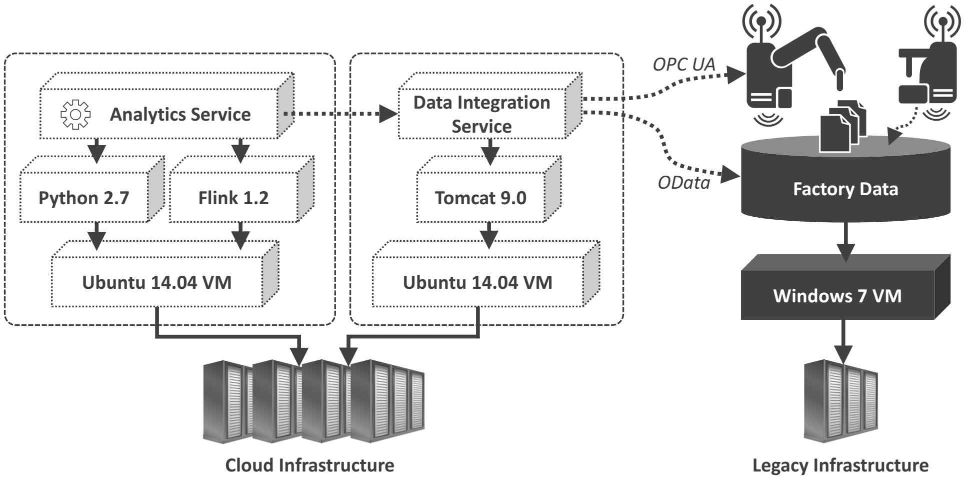

Figure 1. Motivating Scenario: Both, analytics stack as well as data integration stack is hosted on a cloud infrastructure and access different data sources with factory and machine data that are available and stored only in the manufacturing environment.

II. Motivation & Background

In the field of Industry 4.0, the analysis of data gathered, e.g., in manufacturing environments, enables value-adding opportunities, such as predictive maintenance or optimization of production lines. However, the development of such analysis algorithms requires a lot of expert knowledge in implementing machine-learning algorithms, both in general and domain-specific knowledge. Since typically the development of analytics services is not the core competency of manufacturing companies, the task of developing these algorithms and services is commonly out-sourced to professional data scientists. However, due to data privacy and security reasons, the data to be analyzed must not leave the company and thus, needs to be processed locally. Therefore, the analytics service, together with the whole analytics stack needs to be rolled out in the manufacturing environment as close to the data as possible. Moreover, for accessing different types of data resources and unifying different types of data formats and protocols, some kind of data integration middleware is required additionally. However, the manual installation, configuration and adaption of the whole analytics stack, as well as the data integration middleware is commonly complex, time-consuming, and requires vast know-how. The requirements are especially high in case of manual cases. In order to resolve this, all required components of the analytics stack as well as the data integration stack need to be provisioned automatically in the manufacturing environment. Furthermore, the wiring with the data resources containing the data to be analyzed is required to be performed automatically.

Figure 1 abstractly depicts a motivating scenario, showing the provisioning of an analytics stack, as well as a data integration middleware for gathering and processing machine and factory data. Thus, the presented figure illustrates a typical Industry 4.0 scenario. On the left side, the *Analytics Service* and its required analytics stack is depicted. The analytics stack consists of a Python installation, Apache Flink as well as some additional Python-based analysis libraries. Furthermore, this stack is running on an Ubuntu virtual machine, which is hosted on a cloud infrastructure. The data processing platform Apache Flink can be operated on a cluster of several compute nodes, or, like in this scenario, on one virtual machine. It allows batch, as well as stream processing to analyze, e.g., aggregated machine data and thus, enables the optimization of production lines and processes, or predictive maintenance of manufacturing systems. Moreover, it allows the deployment and execution of analytics and machine-learning algorithms implemented, for example in Python, Scala or Java. In this scenario, the *Analytics Service* analyzing the data from the *Data Integration Service* is implemented using Python. In the middle of the figure, the *Data Integration Service* together with its stack is shown. Here, the stack is based on an Apache Tomcat, also running on an Ubuntu virtual machine and hosted on the same cloud infrastructure as the analytics stack. The *Data Integration Service* enables the unified access of multiple and diverse data resources. For example, in the depicted scenario, the integration of a machine using OPC UA [8], as well as a database using OData [9] is shown. Therefore, the *Data Integration Service* is based on different adapters supporting different types of data formats and protocols, similar to an abstract application programming interface (API) [10]. Furthermore, it provides transformation and conversion capabilities to make the accessed machine and factory data processable by the *Analytics Service*. Thus, it enables the integration of various types of data resources to enable the processing and analysis of different kinds of data, such as machine or scheduling data. Both, the depicted machine as well as the database are operated in a factory of the manufacturing company. Furthermore, the database storing metering data of the machines is running on a Windows-based virtual machine, which is hosted on a legacy infrastructure, again operated within the manufacturing environment due to data security reasons.

III. Overview on TOSCA

The OASIS standard Topology and Orchestration Specification for Cloud Applications (TOSCA) [5], [6], [11] enables the automatic provisioning and management of cloud and IoT applications. The specified TOSCA meta-model allows the definition of the structure of such applications in form of *Topology Templates*. A *Topology Template* is a directed graph, consisting of typed nodes and edges. The nodes represent the components of the specified application and are called *Node Templates*. The edges represent the relationships between these *Node Templates* and are called *Relationship Templates*. For example, between two *Node Templates*, a relationship “hostedOn” may be defined, specifying that one *Node Template* is hosted on the other on. Both, *Node Templates* as well as *Relationship Templates* are typed by *Node Types* and *Relationship Types* respectively. These types are specifying the semantics of the templates. *Node Types* as well as *Relationship Types* define *Properties*, enabling the configuration of instances of these types. Furthermore, *Node Types* define *Management Operations* for managing the instances of these types. For example, an OpenStack *Node Type* may define both “startVM” and “stopVM” operations to start and stop a virtual machine.

Two types of artifacts are define by TOSCA: *Implementation Artifacts (IAs)* as well as *Deployment Artifacts (DAs)*. *Implementation Artifacts* are used for implementing the *Management Operations* provided by *Node Types*. *Deployment Artifacts*, on the other hand, implement the business functionality of *Node Templates*. For example, the *Deployment Artifact* of the *Analytics Service Node Template* may be a Python file, implementing the algorithm of the *Analysis Service* to be deployed on Apache Flink. The orchestration of *Management Operations* is realized by using *Management Plans*. These plans are executable workflow models, specifying which operations are executed in which order. All TOSCA elements, artifacts and plans can be bundled in a *Cloud Service Archive (CSAR)*. A *CSAR* is a self-contained archive format defined by TOSCA, which can be executed by standard-compliant runtimes in order to provision and manage the modelled application.

IV. Demonstration: Setup and Description

The TOSCA *Topology Template* describing the provisioning of the analytics stack as well as the data integration middleware is depicted in Figure 2. In the shown topology, the analytics stack and the data integration middleware is provisioned on a virtual machine, running an Ubuntu 14.04, hosted on OpenStack [12]. Instead of OpenStack, of course, any other hypervisor or cloud platform can be used by changing the corresponding *Node Template*. On the virtual machine, the data processing platform Apache Flink as well as Python 2.7 should be installed. Python 2.7 is a required dependency of the *Analytics Service* modelled in the topology, and thus, needs to be installed before the *Analytics Service* can be deployed on Flink. Besides Flink and Python, Tomcat 8 is required to be installed on the virtual machine to enable deployment of the *Data Integration Service*. The *Data Integration Service* is implemented as a Java application and packaged as Web application Archive (WAR), which is why an application server, such as Tomcat is required.

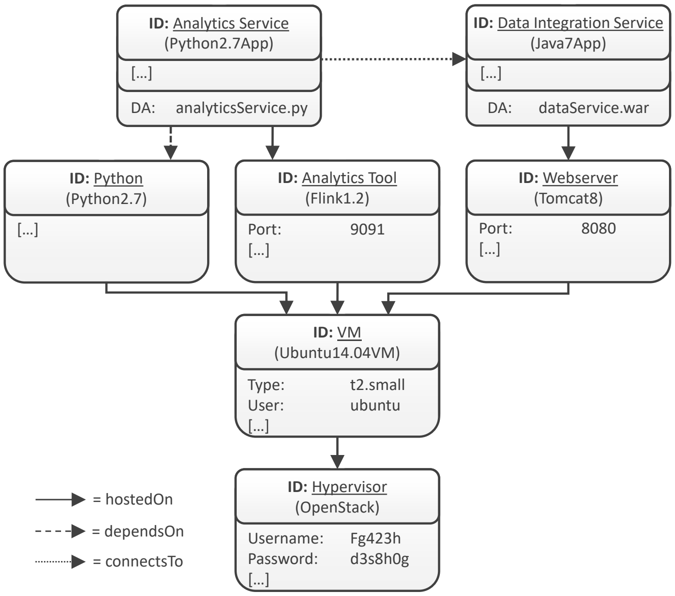

Figure 2. TOSCA Topology Template describing the deployment of *Analytics Service* as well as the *Data Integration Service*.

For creating the virtual machine and installing the components on it different *Implementation Artifacts* are used. For example, the OpenStack *Node Type* has an *Implementation Artifact* implemented as a WAR, providing *Management Operations*, such as “createVM” for creating the virtual machine. Furthermore, the Ubuntu *Node Type* provides a *Management Operation* for transferring files to the virtual machine or to execute scripts on the virtual machine by using a secure shell (SSH) connection. As in our scenario Flink is installed on a single virtual machine, the installation type called *local setup* is used. Therefore, in order to install Flink on the virtual machine the Flink *Node Type* provides a *Management Operation* “install”, which is implemented using Ansible [13]. For simplification purposes, also some common analysis libraries, for instance sklearn [14], are already installed by the Ansible script, together with Flink. Alternatively, these required libraries could also be modelled as separate nodes within the topology.

Required information for executing *Management Operations*, like for example the IP-address of the virtual machine or the SSH credentials required for connecting to the virtual machine are either specified directly in the topology model by defining *Properties*, provided by the user starting the provisioning or gathered during runtime. In TOSCA, *Implementation Artifacts* implementing the *Management Operations* can be realized using various technologies. For example, while in our scenario the *Implementation Artifact* of the Flink *Node Type* is implemented using Ansible, in contrast the “install” operation of Tomcat is implemented using a simple bash script. The *Analytics Service Node Template* for analyzing the data has a *Deployment Artifact* attached to it, allowing to easily exchange the implementation of the *Analytics Service*. Likewise, the *Data Integration Service Node Template* has a *Deployment Artifact* implementing the data integration capabilities. The *Analytics Service Node Template* and the *Data Integration Service Node Template* are connected by a relationship "connectsTo". Thus, the *Analytics Service* only needs to communicate with the *Data Integration Service* to acquire the data that has to be analyzed. The communication with different data resources, such as machines or databases as well as transforming and converting of the data is done by the *Data Integration Service*. Therefore, the *Data Integration Service* implements different adapters to support various types of data formats and protocols, such as OData or OPC UA. Again, the endpoints of these data resources can be either specified directly in the model or provided during provisioning.

For deploying the presented topology model, the OpenTOSCA ecosystem, a standards-based open-source TOSCA runtime environment is used. It enables the automatically provisioning and management of TOSCA-based applications. The ecosystem consists of the three main components: (i) Winery [15], (ii) OpenTOSCA container [16], and (iii) Vinothek [17]. Winery is a graphical modelling tool for creating TOSCA *Topology Templates* of the applications that should be deployed using the OpenTOSCA container. The OpenTOSCA container is the engine processing these *Topology Templates* and, thus, enabling the provisioning and managing of the described applications. The Vinothek is a graphical self-service portal, enabling the end user to select available applications and initiate the provisioning of them. The *Management Plans* required for the provisioning of the applications can be generated by the plan generator [18], a component of the OpenTOSCA container, by analysis of the application topology model. As workflow language the Business Process Execution Language (BPEL) [19] is used. The generated *Management Plans* are deployed and executed by the OpenTOSCA container on a locally running workflow engine, here the WSO2 Business Process Server (BPS). The source code of OpenTOSCA is publicly available on GitHub[1].

V. Related Work

In this section, we present different works that are related to automated provisioning and integration of components.

Script-centric configuration management technologies, such as Chef[2], Puppet[3], or Juju[4] enable the wiring and configuration of components by writing deep technical scripts. However, manually writing low-level scripts for integrating different components is not trivial and thus, an error-prone task. Furthermore, these technologies are mainly used to install or configure components on a target infrastructure, but are not directly able to integrate proprietary management APIs, for example, to provision a new virtual machine. In contrast, using TOSCA as a high-level modeling language enables the modelling of the provisioning, configuration, and wiring of virtual machines as well as software components.

There are also different related works [20]–[22] available, discussing about using container technologies, for example, Docker Compose[5], Docker Swarm[6], and Kubernetes[7] for a fast and automated deployment of applications as well as orchestrating them. Using such a container technology enables the creation of images containing whole application stacks as well as transferring them between different environments. Thus, also enabling the reusability of the created images. However, the container-based approaches only consider the orchestration of containerized components, whereas TOSCA provides a flexible and generic, infrastructure and container independent orchestration approach. Especially if multiple heterogeneous components or physical devices needs to be integrated.

In [23], Breitenbücher et al. discuss problems and challenges occurring when integrating different components and technologies. For example, they state that most of the cloud providers' available APIs and web services are not standardized, thus, preventing the fully automated provisioning of components and applications. In their work, they present an approach for integrating different script- and service-centric provisioning and configuration technologies. In [24], Eilam et al. discuss the challenges of deploying and configuring web applications in the context of data centers. In their work, they state that the deployment as well as the configuration of applications are complex and error-prone tasks and thus, model-driven approaches should be favored over low-level and error-prone script-based technologies.

Besides TOSCA, there are other alternatives for modelling cloud applications, such as Blueprints [25], CloudML [26], and enterprise topology graphs [27]. However, because of the tooling support available, we decided to use TOSCA as interoperable modelling language to realize our demonstration.

VI. Conclusion

In this paper we showed how analytics algorithms, analytics stacks and data integration services can be bundled as smart services by using TOSCA. We showed how the modelling capabilities of TOSCA can be used to create reusable topology fragments of the overall service, which can be rearranged and configured for particular use cases and analysis scenarios at hand. In future works, we plan to enrich the data integration service in order to enforce policies attached to data to assure security and compliance aspects for business critical manufacturing data.

Acknowledgment

This work is partially funded by the project SePiA.Pro (01MD16013F) as part of the BMWi program Smart Service World.

[1] https://github.com/OpenTOSCA

[2] http://www.chef.io/chef/

[3] http://puppet.com/

[4] http://jujucharms.com/

[5] https://www.docker.com/products/docker-compose

[6] https://www.docker.com/products/docker-swarm

[7] http://kubernetes.io/

References

[1] M. Hermann, T. Pentek, and B. Otto, “Design Principles for Industrie 4.0 Scenarios,” in *Proceedings of the 49th Hawaii International Conference on System Sciences (HICSS)*, 2016, pp. 3928–3937.

[2] G. Allmendinger and R. Lombreglia, “Four strategies for the age of smart services,” *Harvard Business Review*, vol. 83, no. 10, p. 131, 2005.

[3] M. Falkenthal, U. Breitenbücher, M. Christ, C. Endres, A. W. Kempa-Liehr, F. Leymann, and M. Zimmermann, “Towards Function and Data Shipping in Manufacturing Environments : How Cloud Technologies leverage the 4th Industrial Revolution,” in *Proceedings of the 10th Advanced Summer School on Service Oriented Computing*. IBM Research Division, 2016, pp. 16–25.

[4] M. Falkenthal, U. Breitenbücher, K. Képes, F. Leymann, M. Zimmermann, M. Christ, J. Neuffer, N. Braun, and A. W. Kempa-Liehr, “OpenTOSCA for the 4th Industrial Revolution: Automating the Provisioning of Analytics Tools Based on Apache Flink,” in *Proceedings of the 6th International Conference on the Internet of Things*, ser. IoT’16. ACM, 2016, pp. 179–180.

[5] OASIS, *Topology and Orchestration Specification for Cloud Applications (TOSCA) Version 1.0*, Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[6] ——, *Topology and Orchestration Specification for Cloud Applications (TOSCA) Primer Version 1.0*, Organization for the Advancement of Structured Information Standards (OASIS), 2013.

[7] The Apache Software Foundation, “Apache Flink: Scalable Stream and Batch Data Processing.” [Online]. Available: https://flink.apache.org

[8] OPC Foundation, “Unified Architecture - OPC Foundation.” [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua

[9] OASIS, “OASIS Open Data Protocol (OData) TC.” [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=odata

[10] F. W. Baumann, O. Kopp, and D. Roller, “Abstract API for 3D printing hardware and software resources,” *The International Journal of Advanced Manufacturing Technology*, 4 2017. [Online]. Available: http://dx.doi.org/10.1007/s00170-017-0260-y

[11] T. Binz, U. Breitenbücher, O. Kopp, and F. Leymann, *TOSCA: Portable Automated Deployment and Management of Cloud Applications*, ser. Advanced Web Services. Springer, Jan. 2014, pp. 527–549.

[12] O. Sefraoui, M. Aissaoui, and M. Eleuldj, “Article: OpenStack: Toward an Open-source Solution for Cloud Computing,” *International Journal of Computer Applications*, vol. 55, no. 3, pp. 38–42, 10 2012.

[13] M. Mohaan and R. Raithatha, *Learning Ansible*. Packt Publishing, Nov. 2014.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, V. D. Ron Weiss, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011. [Online]. Available: http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

[15] O. Kopp, T. Binz, U. Breitenbücher, and F. Leymann, “Winery – A Modeling Tool for TOSCA-based Cloud Applications,” in *Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC 2013)*. Springer, Dec. 2013, pp. 700–704.

[16] T. Binz, U. Breitenbücher, F. Haupt, O. Kopp, F. Leymann, A. Nowak, and S. Wagner, “OpenTOSCA - A Runtime for TOSCA-based Cloud Applications,” in *Proceedings of the 11th International Conference on Service-Oriented Computing (ICSOC 2013)*. Springer, Dec. 2013, pp. 692–695.

[17] U. Breitenbücher, T. Binz, O. Kopp, and F. Leymann, “Vinothek - A Self-Service Portal for TOSCA,” in *Proceedings of the 6th Central-European Workshop on Services and their Composition (ZEUS 2014)*. CEUR-WS.org, Feb. 2014, Demonstration, pp. 69–72.

[18] U. Breitenbücher, T. Binz, K. Képes, O. Kopp, F. Leymann, and J. Wettinger, “Combining Declarative and Imperative Cloud Application Provisioning based on TOSCA,” in *International Conference on Cloud Engineering (IC2E 2014)*. IEEE, Mar. 2014, pp. 87–96.

[19] OASIS, *Web Services Business Process Execution Language (WS-BPEL) Version 2.0*, Organization for the Advancement of Structured Information Standards (OASIS), 2007.

[20] C. Pahl, “Containerisation and the PaaS Cloud,” *IEEE Cloud Computing*, vol. 2, no. 3, pp. 24–31, 2015.

[21] A. Tosatto, P. Ruiu, and A. Attanasio, “Container-Based Orchestration in Cloud: State of the Art and Challenges,” in *Ninth International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS 2015)*. IEEE, Jul. 2015, pp. 70–75.

[22] D. Bernstein, “Containers and Cloud: From LXC to Docker to Kubernetes,” *IEEE Cloud Computing*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[23] U. Breitenbücher, T. Binz, O. Kopp, F. Leymann, and J. Wettinger, “Integrated Cloud Application Provisioning: Interconnecting Service-Centric and Script-Centric Management Technologies,” in *On the Move to Meaningful Internet Systems: OTM 2013 Conferences (CoopIS 2013)*. Springer, Sep. 2013, pp. 130–148.

[24] T. Eilam, M. Kalantar, A. Konstantinou, G. Pacifici, J. Pershing, and A. Agrawal, “Managing the configuration complexity of distributed applications in Internet data centers,” *Communications Magazine*, vol. 44, no. 3, pp. 166–177, Mar. 2006.

[25] M. P. Papazoglou and W.-J. van den Heuvel, “Blueprinting the cloud,” *IEEE Internet Computing*, vol. 15, no. 6, pp. 74–79, 2011.

[26] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, “Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-cloud Systems,” in *Proceedings of the 2013 IEEE Sixth International Conference on Cloud Computing (CLOUD 2013)*. IEEE, Jul. 2013, pp. 887–894.

[27] T. Binz, C. Fehling, F. Leymann, A. Nowak, and D. Schumm, “Formalizing the Cloud through Enterprise Topology Graphs,” in *Proceedings of 2012 IEEE International Conference on Cloud Computing (CLOUD 2012)*. IEEE, Jun. 2012, pp. 742–749.

All links were last accessed on June, 29th 2017.