Facing Big Data System Architecture Deployments: Towards an Automated Approach Using Container Technologies for Rapid Prototyping

Matthias Volk, Daniel Staegemann, Ashraful Islam, Klaus Turowski
Otto-von-Guericke University Magdeburg, Germany
{matthias.volk, daniel.staegemann, ashraful.islam, klaus.turowski}@ovgu.de

Abstract

Within the last decade, big data became a promising trend for many application areas, offering immense potential and a competitive edge for various organizations. As the technical foundation for most of today´s data-intensive projects, not only corresponding infrastructures and facilities but also the appropriate knowledge is required. Currently, several projects and services exist that not only allow enterprises to utilize but also to deploy related technologies and systems. However, at the same time, the use of these is accompanied by various challenges that may result in huge monetary expenditures, a lack of modifiability, or the risk of vendor lock-ins. To overcome these shortcomings, in the contribution at hand, modern container and task automation technologies are used to wrap complex big data technologies into re-usable and portable resources. Those are subsequently incorporated in a framework to automate the deployment of big data architectures in private and limited resources.

1. Introduction

Within the last decade, the storage, management, and processing of huge amounts of differently structured data have become more important than ever. With the advent of new trends, such as social media, the internet of things, and other data-intensive application scenarios, the necessity to handle those became ubiquitous [1]. As a result, apart from new technologies, sophisticated architectural concepts were required that provide a scalable and robust framework for the current and future development. However, at the same time, this reinforced the complexity of the engineering of the related systems and, thus, potentiated the lack of qualified staff [2]. Consequently, several deployment solutions and commercial services came up that promise potential users to *easily* realize their data-intensive endeavors and thus facilitate the rapid prototyping of novel ideas and testbeds. However, the convenience to have everything at *one* place comes in most of the cases with the cost of technology and platform-specific knowledge. Prominent examples, such as Amazon Web Services or IBM BigInsight offer a broad range of potential technologies, functionalities, and sophisticated pay-per-use models. However, not only the use of the platform-specific technologies themselves but also the cost estimation of their usage can be sometimes cumbersome. Currently, the maintenance of internal software, external expert knowledge, and customization efforts denote oftentimes the biggest sources of unplanned costs [3]. As a result, the combined use of multiple technologies and services in parallel may not only be challenging to realize, sometimes it can be even prevented, due to a lack of existing connectors or interfaces. Ultimately this could not only result in high monetary expenditures, extensive knowledge required to handle big data projects but also a potential vendor lock-in effect, through which the user is forced to solely stick to the services offered by the provider. Especially the latter was noted for many different providers in several research studies, such as [4–6].

In contrast to this, free and open-source solutions such as the *Developing Data-Intensive Applications with Iterative Quality Enhancements* (DICE) framework [7], Apache BigTop, or the Cloudera Hortonworks project, which partially attempt to overcome the referred problems, are in many cases limited in terms of their applicability. For instance, Cloudera offers inter alia their distribution as a sandbox that comes with an extensive collection of big data technologies, however, in many cases they exceed what the user needs, which in turn results in the necessity for a potentially complicated and cumbersome customization and configuration. Generally speaking for these solutions, there is no opportunity to extend or reduce the setup to its required technologies. In the flux of big data, a multitude of technologies is constantly emerging or changing [1]. Therefore, compared to currently existing solutions, a lightweight and modifiable approach that relies on open-source technologies, allowing an automated system architecture deployment, appears to be highly aspirational. To facilitate bridging this gap, the following research question shall be answered throughout this work:

How could a modifiable and open-source-based approach for automated big data system architecture deployments be facilitated and designed?

Resulting from this question, the main purpose of this work is to provide a convenient and low complexity solution, where individual components can be offered as re-usable and re-configurable packages. This also includes their combined use, allowing for components to be added or removed to complex architectures with minimal effort and without specialized knowledge for the deployment to eventually facilitate rapid prototyping setups. The desired solution should be platform agnostic, extendable, and adjustable to suit the available computing resources.

To find a suitable answer to the aforementioned research question, the constructive design science research (DSR) methodology is followed [8] and the six-stepped workflow as recommend by Peffers et al. [9] is implicitly employed. This leads to the publication being structured as follows. After giving an initial motivation and definition of the main objectives an overview of the existing theory is needed. This is realized through the presentation of theoretical background information as well as a structured literature review. The latter is used to identify existing approaches, container technologies, and other guidelines for the intended artifact. Eventually, the obtained findings are used for the design and development of the artifact. Afterward, everything is demonstrated using one of the most prominent big data architectures for real-time stream processing. In the end, a thorough evaluation is performed at which the created solution is compared to similar existing approaches. Concluding remarks will end the paper.

2. Theoretical Foundation

In recent years, big data became one of the most promising trends. One common way to facilitate the deployment of the related technologies and architectures is the use of container technologies.

2.1. Big Data

With the increasing volume and complexity of data produced in today's society, which are addressed by the term big data, traditional techniques for managing and processing data are oftentimes no longer sufficient [10]. As a result, new approaches have emerged to deal with those challenges. Even though they are amalgamated under one umbrella term, the corresponding endeavors comprise a variety of highly different use cases [1]. Common to them, however, is a strong focus on the scalability and portability of the deployed solutions [11–13]. Furthermore, in many scenarios, a high degree of flexibility is desirable [1], which also affects the design and development of those solutions.

Mandatory for the implementation of big data projects is the utilization of highly sophisticated and scalable technologies, which can cope with the challenges resulting from the big data characteristics. In general those "*summarize technological developments and techniques in the area of data storage and data processing that allow the handling of exponential increases of data in terms of volume, variety, velocity, value and veracity"* [14]. Many of those technologies are widely known today. Some of the most prominent representatives include solutions from the Apache Foundation, such as Hadoop, Spark, Zookeeper, or Hive. While some bring a broad range of functionalities, other technologies are only intended for one specific purpose. Besides a surge in interest, as of today, also a lack of comprehensive knowledge prevails [2]. This is largely due to the ever-growing market of technologies and tools that renders it nearly impossible to always stay up to date concerning its development. This was also thoroughly described and investigated in [15]. There, a comprehensive big data technology ontology (BDTOnto) was introduced that comprises existing properties, required knowledge, and relations to other technologies. By facing such kind of technology mapping concept to the entirety[1] of big data technologies, it becomes apparent that the selection, combination, and implementation depicts a complex undertaking, at which numerous steps need to be performed.

The activities related to the planning, design, and development of big data systems are oftentimes consolidated under the term big data engineering [16]. Based on a structured analysis and planning of the targeted project, requirements, specifications, system design decisions, tests, and deployments are determined and carried out. Eventually, this leads to a purposeful composition of big data technologies, a big data architecture. More precisely, it can be defined as an "*architecture that provides the framework for reasoning with all forms of data. Thus, it is a logical structure of core elements used to store, access and manage the big data*" [17].

[1] http://dfkoz.com/ai-data-landscape/, accessed on 15-06-2021

In many cases, those can be not only very complex to be constructed but also to be deployed and managed. Hence, with regard to the dynamic nature of this domain, reflected by the continuous emergence of new application areas, technologies, or architectures, it appears to be beneficial to automatize leastwise the latter, the time-intensive configuration and deployment steps.

2.2. Container Technologies

Since big data applications are usually highly suited to be deployed in the cloud [18] and need to be highly scalable, it is common to use container technologies for their implementation. Containers are virtualized, lightweight operating system (OS) processes that provide portable runtime environments independent of the underlying hardware [19]. Thereby, they help in dealing with issues like dependency conflicts, missing dependencies, and platform differences [20]. There are numerous technologies, which can be drawn upon. For instance, Docker is a container-based technology that offers a user-friendly application programming interface (API) that is unified across platforms. It uses namespaces to completely isolate an application's view of the underlying OS and environment, including process trees, network, user IDs, and file systems.

Furthermore, to reduce the complexity and effort when dealing with dependencies, it packages each component and its dependencies. Ansible is a simple automation engine to automate cloud provisioning, configuration management, application deployment, intra-service orchestration, and other needs. When utilized, it connects to nodes (servers, containers, or VMs) and creates small programs called "Ansible Modules". These programs are resource models of the desired state, the system has to be in. Ansible then executes these modules and removes them once the task is completed. Compared to other similar tools, such as Puppet or Chef, Ansible is efficient and lean, due to not requiring an active server, daemon, or database to run specific modules or keep states.

2.3. Available Non-Commercial Deployment Solutions for Big Data Technologies

Commercial service providers such as Amazon and Google that partially allow (semi-) automated deployments of prominent big data technologies in their cloud environments provide in many cases proprietary, self-developed big data solutions that can be *used* with relatively little effort. However, those come with expenses and obligations as mentioned before. To our knowledge, only a few non-commercial solutions exist that allow an automated deployment of well-known big data technologies. As briefly described at the beginning of this contribution, these are Apache BigTop, the DICE framework [7], and the Cloudera distribution. While the latter depicts rather a multifunctional suite and provides a variety of well-known tools, the other two approaches are deployment solutions that allow the provision of targeted tools. BigTop "*is an Apache Foundation project for Infrastructure Engineers and Data Scientists looking for comprehensive packaging, testing, and configuration of the leading open source big data components"* [21].

The offered components are packaged, delivered, and maintained by the community behind the project. According to its declarations, the scope of this solution mostly covers but is not exclusively limited to, big data technologies from the Hadoop ecosystem. For the actual deployment of the components, Docker is used, and for their internal configuration Puppet. While BigTop offers a wide range of functionalities, configurations as well as testing capabilities, many technologies outside the Hadoop ecosystem are excluded here, presumably, due to complexity and integration efforts.

In contrast to the aforementioned solution, the DICE framework originated from an EU project funded under the Horizon 2020 program, which seeks to "*to deliver a quality-driver DevOps toolchain for Big data applications that natively support these Big data technologies*" [7]. In doing so, a comprehensive plugin for the integrated development environment (IDE) Eclipse is provided that helps step-by-step with the implementation of data-intensive applications. Through the chained integration of UML diagram profiles and technology-specific peculiarities, comprehensive and detailed activities of big data engineering can be performed, including the planning, design and development, testing, and deployment. Here, the deployment of related technologies is realized using the configuration management tool Chef, which fulfills similar functionalities as Ansible and Puppet. Through the additional use of the cloud industry-standard *Topology and Orchestration Specification for Cloud Application* (TOSCA), cloud deployments of related prototypes, as well as continuous delivery and testing, are facilitated here. Unfortunately, the DICE project ended in 2018. Since then, no major extensions or updates have been performed. Hence, long-lasting usage cannot be recommended, because changes in the big data ecosystem will no longer be considered.

This circumstance, again, reinforces the necessity to provide such a solution that remains usable in the long term. Notwithstanding that, as one may note, the presented examples comprise just a small excerpt of currently existing approaches. Hence, in-depth observation of the current state of the art is required that shall cover the research conducted in this domain.

3. State of the Art

To obtain an overview of the current state of the art, a structured literature review according to the recommended workflow of Levy and Ellis [22] was conducted that further relies on the approach presented by Webster and Watson [23]. In the following, the review protocol as well as the results are presented.

3.1. Review Protocol

For the identification of relevant research articles, which incorporate deployment technologies in the field of big data, suitable keywords were defined, logically connected, and applied in various scientific literature databases. Those are namely, IEEE, ScienceDirect, Scopus, and CiteSeerx. Depending on each query engine, the following search term was applied on title, abstract, and keywords to find only relevant articles: "*big data" AND (architecture OR application) AND (DevOps or deployment) AND (strategy OR framework OR practice OR method OR survey)*".

To cover only articles that were proposed after the early hype of big data, no papers published before 2014 were considered. This resulted in a total of 2988 unique articles that were manually checked. Those are distributed as follows: 93 (IEEE), 2237 (ScienceDirect), 367 (Scopus) and 291 (CiteSeerx). To refine the overall amount of articles, several inclusion and exclusion criteria were used as proposed in [22]. As soon as one of the latter was valid, the paper was rejected, the same applies to those who did not fulfill all inclusion criteria. A list of the inclusion and exclusion criteria is given in Table 1.

Table 1. Inclusion and exclusion criteria

Inclusion Criteria	Exclusion Criteria
Discusses big-data/ large-scale deployments	Published before 2014
Incorporates non-commercial technology	Vague to no deployment information
Presents enough information for replication	Focus on proprietary technologies
Use of modern technologies	Focused on a very specific application
Written in English	Use of outdated or unmaintained technologies
Peer-reviewed publication	

3.1 Literature Review Results

During further examination, it was noticed that a large number of research articles were focused on smart-cities, smart-grid, large sensor networks, or relying heavily on commercial infrastructures with little to no information about the deployment. Eventually, a number of eight papers remained that appeared promising for consideration.

Feller et al. [24] discuss in their article how Hadoop clusters are deployed in the cloud. In their deployment discussion, they note that Hadoop was not designed to be deployed in VMs as it expects data and compute nodes to coexist and there is also no concept of elasticity. A potential solution for the deployment of Hadoop workers in the cloud was given in [25]. Here, the authors highlight that in cloud environments the computing VMs are typically running full OSs. However, the hypervisor in VMs often degrades the performance of the virtual OS. To overcome this issue, an approach is proposed that harnesses the capabilities of Docker. Wu et al. [26] propose in their work the *YZStack* architecture, where big data tools are implemented in separated layers. The deployment of those is performed using an *adaptive image*. In the infrastructure layer, they pre-generate a virtual server image that includes the OS and minimum required modules that are commonly used. The intended big data tools are then built onto these images with all configurations happening in an ad-hoc manner.

In the work of [27], an automated deployment model for high-performance clusters (HPC) is described. The presented solution focuses on the complexity of deployment automation and configuration management. Especially container technologies are highlighted here as the key technology for HPC cases. Specifically, Ansible was mentioned as one of the most important deployments automation engines. This is due to the reason that, inter alia, common standards such as SSH are used and no dedicated daemons on each node are required, which effectively reduces the overall overhead. Apart from that, all configurations can be done using YAML (Yet Another Markup Language). The authors highlight that configurations for deploying a component can be abstracted into roles, which consist of several tasks [27]. The flexibility of running a task or role on specific nodes using inventories offers complete convenience and freedom for system administrators to define and maintain re-usable scripts, called *playbooks*, that can be used to take a node into the desired state for a specific package or technology. Docker images were also of major interest in other research articles, such as in [28]. Within this article, the authors propose a deployment method, which is based on a general docker workflow, where individual components are packaged into Docker images and deployed in container engines as necessary. Beyond that, they highlight the benefits of using this approach compared to classic VMs. Morabito et al. [29] performed a comparison between the hypervisor and container-based virtualization technologies. In doing so, various strengths and weaknesses of each type were highlighted. The work presented by Felter et al. [30] denotes another comparison of virtual machines and *Linux containers*. In particular, KVM as hypervisor and Docker as container engine were used. They came to similar conclusions as Morabito et al. [29] and concluded that generally speaking, both solutions achieved a mature status. However, container deployments using Docker still outperform the KVM deployments in terms of all tested metrics. Nevertheless, both solutions have their advantages and disadvantages. Lastly, in [31], an approach to deploy large-scale datasets in cloud environments is presented. Using configuration management tools and a modified version of BitTorrent, automation of their deployment is achieved.

As one may note, according to the given summaries, it becomes apparent that different approaches exist that attempt to provide suitable solutions for the deployment of technologies in resource-limited infrastructures, including also big data tools. For instance, while in [26] a pre-packaged VM was used to offer a complete solution, another approach used configuration and management tools to allow automation for the configuration and deployments of various components, such as Ansible [27]. Despite the great acceptance of classical VM-based approaches, including not only the OS but also multiple preinstalled functionalities, it was noted that in many cases container technologies delivered better results in a direct comparison [29]. Thus, Docker and Ansible appeared to be desirable solutions for further deployment and configuration management.

4. Design and Development

Emerging from these considerations, in the following design and development section, the intended artifact of this work is presented. In particular, a convenient, platform-agnostic, and low complexity concept is proposed that allows the deployment of individual components and re-configurable packages. As found out during the investigation of existing theory and the performed literature review, container technologies are a widely acknowledged solution when it comes to the deployment of a large number of components in limited-resource environments. They offer many different advantages, such as dependency management and conflict mitigation [20] as well as a high degree of portability of the created solution that allows, in turn, easy migration from e.g. public to private cloud deployment models [27].

4.1. Preliminary Considerations

After investigating the recommended container and automation from the literature review in more detail, general steps were identified, which are required for the basic implementation and application of a potential solution. This includes the setup of related deployment and management nodes. While the first is used for the actual deployment of the targeted technologies and architectures, the latter is utilized to manage and handle all required implementations. After that, in case that available registries do not already provide them, the components for each technology need to be created. Consequently, for each component, a base image is used and extended, following the required container technology component creation guidelines and the idea of the *adaptive image*. For the automation, then, a deployment management framework is required that converts manual process steps into automated small scripted steps. Those are predominantly important in complex environments, as it is the case for the big data domain. Hence, the capability to deploy a large number of isolated or compound components does not only allow the provision of single big data tools but also complete architectures. To cover the information required for such a sophisticated artifact, a suitable concept needs to be utilized that delivers an all-encompassing overview in terms of existing technologies, their fulfilled functionalities, implementation details as well as relations between each of those. When looking at other approaches, such as the DICE framework [7] or Apache BigTop, this shall offer an opportunity to easily extend or reduce the planned setup to its required technologies.

4.2. A Basic Framework for the Automated Deployment of Big Data System Architectures

By taking all of the aforementioned information into consideration, a hybrid framework was derived, where big data components, as well as their combination, configuration, management, and deployment, are prepared via machine-readable format, to achieve increased automation, portability, and reusability. Compared to other existing solutions, the focus was on non-commercial, low complex, resource conservative, and easily extendable elements. For an improved (re-) usability, a sophisticated concept is utilized that allows potential users to discover, identify and keep track of interdependencies between single big data technologies as well as complex architectures. Eventually, for the developed artifact, the BDTOnto [15], Docker, and Ansible were used, most of all due to their prevailing benefits compared to alternative solutions, which were presented in the aforementioned sections. An overview of the framework is depicted in Figure 1.

All required information, which is relevant for the general understanding of the technologies and their relation to each other, are stored in the ontology [15]. This includes not only single technologies and their general compatibilities, version information, provided functionalities, and deployment details, but also in which way they can be composed to specific architectures, such as in [32, 33]. Since Docker containers are used for the packaging of big data technologies, essential information for the construction of those or even the used deployment files can be linked within the ontology.

In case that a container for a specific technology is neither created nor available in openly accessible registries, an initial creation needs to be performed. Once a container image is created, it can be persisted and distributed for later reuse through a private or public Docker registry, such as DockerHub. There, a multitude of publicly available big data implementations is already provided. Generally, an adhering deployment of single components and generic architectures can already be performed through docker-compose files, created using YAML.

To have all information in one place, the created files and images can be gathered via the linkage with the specific entries for each big data technology, within the ontology. For multiple components, different tasks are required, such as the structuring, copying, managing, or changing of configurations, not only regarding the aimed destination but also in terms of the component interaction. Those tasks are automated through the use of Ansible and logically structured by utilizing the *role* concept. These roles, in turn, can be used within *playbooks*, which define, similar to docker-compose files, the structure of potential architectures. All used images of relevant big data technology components need to be either build prior or directly pulled from an existing registry. The persistency and linkage of the created playbooks can be achieved for complex architectures in the same way as the Docker components, through a registry and the used ontology. To reduce the effort of manual settings and frequent interactions during the deployment process, various configuration information are required within the playbooks, such as the specific endpoints. This information has to be declared in *inventory* files. After the successful creation of a playbook, the deployment of the big data architecture can be executed. Ansible autonomously performs all steps required for the deployment to the desired host in sequential order. Again, to provide a global source of information, these files are then linked to the ontology, similar to the Docker container information. For multi-user management and user-specific endpoint declarations, the inventory files are stored in separate data storage.

5. Evaluation of the Developed Artifact

For the evaluation of the proposed concept, experimental implementation and application of a potential big data architecture was realized. The ascertained complexity was afterward compared to the DICE framework [7] and Apache BigTop, based on various criteria. As one of the most prominent approaches, the Kappa architecture was used [33] and tested with an openly accessible dataset of green taxi trips in New York City[2]. Generally speaking, this architecture presents an *answer* to the Lambda

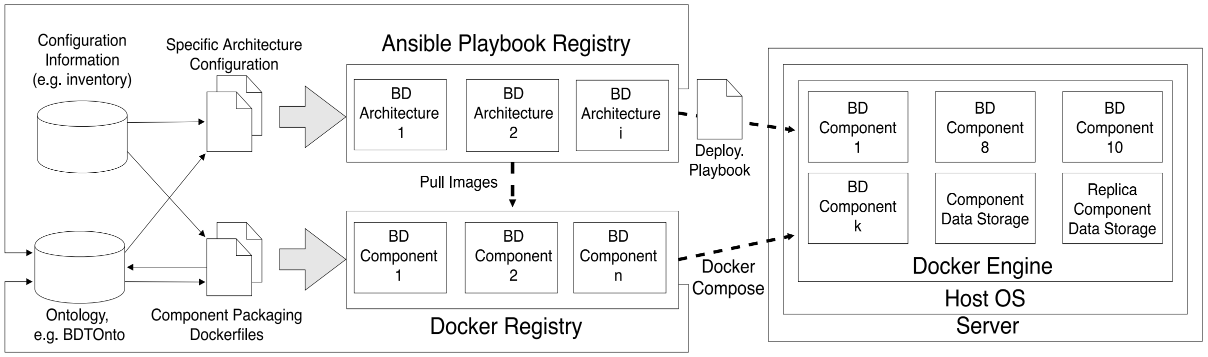

Figure 1. Architectural Setup

2 https://data.cityofnewyork.us/Transportation/2017-Green-Taxi-Trip-Data/5gj9-2kzx, accessed on 15-06-2021

architecture [32]. Compared to the two required systems of the Lamba approach, the Kappa architecture requires only a stream processing system through which the data is incoming and transformed. Afterward, everything is stored within an analytical database [33]. The streaming layer, as the *heart* of the system, is constituted by a messaging system, in that case, Apache Kafka. Further technologies that are frequently used in the context of this are the data storage Apache Cassandra, as the serving layer, as well as Apache ZooKeeper, for cluster state management [33]. An overview of the architecture can be seen in Figure 2.

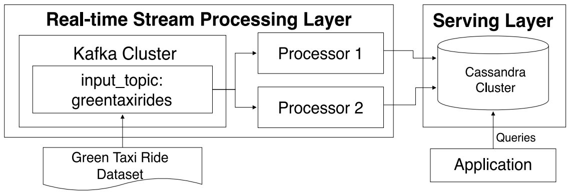

Figure 2. Kappa Architecture

5.1. Prototypical Implementation of the Artifact

For the implementation of the architecture, a multi-node setup was chosen, for which the preparation of the management and deployment nodes are required. On both machines, Docker is required as well as for the manager node additionally Ansible. Here, two VMs are used, each of them with 2x2.2GHz cores, 4GB RAM, a 50 GB disk storage, and Ubuntu Linux 18.04.1 LTS as the distribution. The first VM contains the processing layer and cluster management, represented by Kafka and Zookeeper, along with the evaluation data as the test workload. In the second VM, a Cassandra cluster and the stream processors were deployed. Both VM1 and VM2 were connected using an overlay Docker network that can be accessed from both nodes, allowing communication between them. For the deployment of the architecture, each component was prepared and defined in an independent cluster configuration, using a docker-compose file. The initially required information about general relations and dependencies of the architecture as well as the specific technologies were already included within the used BDTOnto [15]. However, by using additional classes as well as data and object properties, further extensions were performed after the successful deployment. This includes general implementation information, such as the specific runtime environment, and also the linkage to the related files for the deployment. After the Docker-compose files were created for each component, those have been used in Ansible. In particular, different tasks were defined and combined into roles that dealt with the setup of the specific component. Then, each role was put together into one single playbook that is in charge of the automated deployment of the architecture. The needed inventory information for the used machines were defined in the user-specific inventory file. Thereupon, the deployed architecture was successfully tested, using the exemplary dataset as well as some simple data analysis methods. Eventually, all created files were linked to the related big data technology classes within the ontology. The same applies to the user-specific information in the user data storage, regarding the connection endpoints of the used machines. Through the use of a computer-supported solution that gathers all required files and information, the deployment is afterward automated and made executable via *one-click*.

5.2. Framework Comparison

After the successful implementation and evaluation of the artifact, for the identification of its usefulness, an additional comparison to non-commercial solutions was performed. In particular, this comprises the previously described DICE framework [7] as well as Apache BigTop. All required information for the comparison were either directly tested or extracted from the related documentation. The criteria that were employed for this step are derived from the objective of this work, as well as influenced by non-functional requirements from [34] as they define "*constraints on the services or functions offered by the system*" [35]. Particularly, the usability, portability, reproducibility, resources requirements, flexibility, and scalability were observed and compared to each of the evaluated solutions.

The *usability* does not only focus on the required technical knowledge but also the ease of operation. For the proposed approach, the difficulty lies in preparing each component for deployment and composing all required elements for the final architecture. However, once an architecture is prepared, it can be deployed with little technical knowledge. Therefore, the user only needs to execute the specific playbook against deployment nodes and the whole process afterward is automated. To set up Apache BigTop on deployment nodes, a specific shell script is provided, which prepares each node for the package-based deployment that is provided by the community contributors. For Docker-based deployments, the focus is mainly put on Hadoop and the related ecosystem. Additionally, the deployment management is realized through Puppet. DICE, in turn, requires for most of the provided tools only the Eclipse plugin. With the aid of the IDE, the user is guided through different stages of the development and deployment of the data-intensive application (DIA), covering everything from the modeling up to the initial implementation of the prototype. However, a multitude of information is required during the complete setup and use of each DIA that prevents a simple quickstart, or a rather rapid deployment, of a system. The user needs, similar to BigTop, specific details about underlying principles before using the proposed big data technologies.

With the *portability*, the effort of operating and migrating the same architecture to different resources was evaluated. By following a container-based approach, increased portability can be ensured in the proposed artifact. For the validation of this criterion, the destinations of the targeted machines were changed. With minor changes in the configuration, the system was easily deployable to similar pre-configured environments. For BigTop's package-based deployment, a portability is not easily achievable, since a given installation cannot be migrated to a different resource without re-configuring everything from scratch. However, BigTop's Docker deployments reveal similar results to the proposed approach. DICE intends to interact with various cloud platforms, at which the DIA can be deployed. By using reconfigurable files and the provided IDE portability can be achieved to a certain degree.

To investigate the *reproducibility* in more detail and find out whether the developed approach always delivers the same architecture for the same container images and configuration files, the deployment playbook of the architecture was run multiple times. As already presumed, in each run, the architecture was deployed in the same formation and a stable state. Once everything is set up and correctly configured, similar results should be achievable with the DICE framework. However, compared to the proposed solution, those configurations will be presumably a bit more complex, due to the given configuration options. For Apache BigTop, reproducibility is possible, but again, new configurations for each deployed component can be are required.

Regarding the *resource requirements*, in the experimental setup for the proposed solution, the deployment architecture generated individual clusters on two VMs. Using the idle state resource, the additional main memory usage for deploying the ZooKeeper and Kafka cluster was 1.95GB on VM1 and 1.2GB on VM2 for Cassandra and the stream engine. Similar resource utilization is expected for BigTop's Docker-based deployment, due to the same underlying technology. To harness the basic functionalities of the DICE framework, the Eclipse IDE, as well as the plugin and some further tools, are required. The deployment itself is performed through the use of the configuration management tool Chef and cloud environments. In general, it can be expected that the workload will be relatively equal compared to the other solutions.

The *reuse* focuses on the single components, which can be deployed by each of the approaches. The proposed solution includes a combination of an ontology and container-based approach to allow potential users to easily deploy and extend single big data technologies and complex architectures to their desired environment. The configuration of the deployment is performed within the respective solution. As a result, the components and architectures can be reused as individual container images, extended with custom configurations, and easily shared. For BigTop's Docker-based deployments, individual components are deployed as independent packages. It creates a generic container, installs system-level packages of the available components, and additionally uses a dynamically generated configuration for each container. This approach differs from the presented one in the sense that there are no shareable containers in the end. DICE itself sticks to container-based deployments. However, the relatively static inputs for the configurations prevent quick reuse of the developed components. The same applies to complex architectures.

The *flexibility* of each solution was investigated and compared through the examination of the modifiability and extendibility. In terms of the developed artifact, the user can easily add or remove new components, as well as seamlessly integrate completely new technologies, without any further changes on the core. Especially through a self-creation or use of open access repositories, thorough extensions are imaginable in a short time (cf. Docker repository). BigTop, on the other hand, only allows to build and install specific container images. As a result, the user is only able to use components that are provided by the team behind the tool. DICE, in turn, does not intend to provide further big data technologies, since the tools, configurations, and functionalities are very complex and tailored for each of them. In combination with the discontinued development, this circumstance acts as the greatest counterargument for a potential application.

In terms of scalability, the proposed framework currently uses a pre-defined cluster size and configuration. To scale to a large cluster or more complex architecture, additional effort from the user is required to update the deployment configuration accordingly. This severely limits the scalability of the deployment operation. With the integration into specialized workflows or complex deployment systems, this could be again automated using configurations that are passed through. This was already done in the related tools of the DICE framework that attempt to reduce the effort for manual settings as much as possible. In combination with the used cloud platform, scaling for the deployment is partially possible. Nevertheless, an initial configuration has to be performed. For BigTop, similar limits are applicable and the user has to configure additional components. However, it allows a dynamic configuration for deployed components, which results in much easier and less complex scaling.

5.3. Discussion

By summarizing the outlined aspects, it becomes apparent that the developed solution outperforms Apache BigTop and the DICE framework in multiple aspects. In such a fast-changing environment like big data, a long-lasting and adaptable solution was proposed that allows system engineers to rapidly deploy even the most recent big data technologies for their application scenarios. However, the benefits of this solution come at the expense of the level of detail for configuration, which needs to be invested before the initial deployment. While both of the investigated approaches deliver numerous additional functionalities, configurations, and supplementary material, it was intended to develop a *convenient and low complexity solution, where individual components can be offered as re-usable and re-configurable packages*. Depending on the role of the user that either creates or uses the big data technologies, only basic knowledge is required. Nevertheless, for further configuration management and other specifications, additional effort needs to be put into it. Especially the current scalability should be aimed in the future. For now, stress testing or the setup of turnkey solutions may only be feasible in a limited way. However, with the combination and use of the well-known open-source technologies Docker as well as Ansible, an integration in cloud environments, such as the Google Cloud Platform (GCP) are imaginable without fearing a potential vendor lock-in effect. Therefore, in future work, it is planned to facilitate hybrid or multi-cloud integration to overcome those shortcomings. The further integration and extension are also intended in other workflows and systems. For instance, as proposed by [36], the connection to a technology selection decision support system for big data projects appears to be sensible. Decision-makers that not only want to identify potential technologies but also determine, in which way those could be deployed may greatly benefit from such a solution. Through the use of the ontology, a related setup could greatly increase the level of automation in the way that decision-makers may either deploy single technologies or recommended combinations by *one-click*.

6. Conclusion

In this work, a lightweight, flexible and automated framework was proposed that allows researchers and practitioners to deploy their big data architectures in various environments. By investigating the current state of the art, essential concepts and technologies were discovered. In doing so, not only a conceptual framework was designed and developed as an answer to the aforementioned formulated research question, but also a prototypical implementation performed and presented. In particular, a Kappa architecture was constructed, deployed, and automated. Additionally, for an adhering evaluation with existing concepts that act towards our proposed idea, a comparison to those was performed. As an essential element in continuous integration and continuous delivery pipelines or decision support and decision-making systems, this approach may help future users with the rapid deployment of their big data environments. Through the interconnection with the ontology, no all-encompassing knowledge in all domains is required. As a benefit, prospective big data technology users can easily include and use their desired technologies in the ontology, facilitating their automated deployment at a low cost. Through the future extension to cloud environments, their respective cost models could also be incorporated, when designing new applications, facilitating an even more elaborated decision making. Especially in case, if no internal resources are existing and everything is already located in cloud environments.

7. References

[1] Davoudian, A. and M. Liu, "Big Data Systems", ACM Computing Surveys, 53(5), 2020, pp. 1–39.

[2] Lee, I., "Big data: Dimensions, evolution, impacts, and challenges", Business Horizons, 60(3), 2017, pp. 293–303.

[3] Mccafferty, D., "How Unexpected Costs Create a ‘Cloud Hangover’", CIO Insight, 16.12.2015.

[4] Anthony Jnr, B., S. Abbas Petersen, D. Ahlers, and J. Krogstie, "Big data driven multi-tier architecture for electric mobility as a service in smart cities", International Journal of Energy Sector Management, 14(5), 2020, pp. 1023–1047.

[5] Han, J., S. Park, and J. Kim, "Dynamic OverCloud: Realizing Microservices-Based IoT-Cloud Service Composition over Multiple Clouds", Electronics, 9(6), 2020, p. 969.

[6] Castellanos, C., B. Perez, D. Correal, and C.A. Varela, "A Model-Driven Architectural Design Method for Big

Data Analytics Applications", 7281-7415, 2020, pp. 89–94.

[7] Casale, G. and C. Li, "Enhancing Big Data Application Design with the DICE Framework", in Advances in Service-Oriented and Cloud Computing, Z.Á. Mann and V. Stolz, Editors. 2018. Springer International Publishing: Cham.

[8] Hevner, A.R., S.T. March, J. Park, and S. Ram, "Design Science in Information Systems Research", MIS Quarterly(28), 2004, pp. 75–105.

[9] Peffers, K., T. Tuunanen, M.A. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research", Journal of Management Information Systems, 24(3), 2007, pp. 45–77.

[10] Chang, W.L. and N. Grady, "NIST Big Data Interoperability Framework: Volume 1, Definitions", 2019.

[11] McSherry, F., M. Isard, and D.G. Murray, "Scalability! But at What Cost?", in Proceedings of the 15th USENIX Conference on Hot Topics in Operating Systems. 2015. USENIX Association: USA.

[12] Roy, C., S. Swarup Rautaray, and M. Pandey, "Big Data Optimization Techniques: A Survey", International Journal of Information Engineering and Electronic Business, 10(4), 2018, pp. 41–48.

[13] Günther, W.A., M.H. Rezazade Mehrizi, M. Huysman, and F. Feldberg, "Debating big data: A literature review on realizing value from big data", The Journal of Strategic Information Systems, 26(3), 2017, pp. 191–209.

[14] Schermann, M., H. Hemsen, C. Buchmüller, T. Bitter, H. Krcmar, V. Markl, and T. Hoeren, "Big Data", Business & Information Systems Engineering, 6(5), 2014, pp. 261–266.

[15] Volk, M., D. Staegemann, N. Jamous, M. Pohl, and K. Turowski, "Providing Clarity on Big Data Technologies", International Journal of Intelligent Information Technologies, 16(2), 2020, pp. 49–73.

[16] Volk, M., D. Staegemann, S. Bosse, R. Häusler, and K. Turowski, "Approaching the (Big) Data Science Engineering Process", in Proceedings, 5th International Conference on Internet of Things, Big Data and Security, Prague, Czech Republic. 2020. SCITEPRESS.

[17] Immonen, A., P. Paakkonen, and E. Ovaska, "Evaluating the Quality of Social Media Data in Big Data Architecture", IEEE Access, 3, 2015, pp. 2028–2043.

[18] Kune, R., P.K. Konugurthi, A. Agarwal, R.R. Chillarige, and R. Buyya, "The anatomy of big data computing", Software: Practice and Experience, 46(1), 2016, pp. 79–105.

[19] Sebastio, S., R. Ghosh, and T. Mukherjee, "An Availability Analysis Approach for Deployment Configurations of Containers", IEEE Transactions on Services Computing, 2018, p. 1.

[20] Merkel, D., "Docker: lightweight Linux containers for consistent development and deployment", Linux Journal(239), 2014.

[21] https://bigtop.apache.org/, accessed 6-15-2021.

[22] Levy, Y. and T.J. Ellis, "A Systems Approach to Conduct an Effective Literature Review in Support of Information Systems Research", Informing Science: The International Journal of an Emerging Transdiscipline, 9, 2006, pp. 181–212.

[23] Webster, J. and R.T. Watson, "Analyzing the Past to Prepare for the Future: Writing a Literature Review", undefined, 2002.

[24] Feller, E., L. Ramakrishnan, and C. Morin, "Performance and energy efficiency of big data applications in cloud environments: A Hadoop case study", Journal of Parallel and Distributed Computing, 79-80, 2015, pp. 80–89.

[25] Tan, Y., W. Wang, Q. Wu, and J. Lin, "An implementation of heterogeneous architecture based MapReduce in the clouds", in Proceedings of 2016 2nd International Conference on Cloud Computing and Internet of Things, CCIOT. 2016. IEEE: Piscataway, NJ.

[26] Wu, S., C. Chen, G. Chen, K. Chen, L. Shou, H. Cao, and H. Bai, "YZStack", Proceedings of the VLDB Endowment, 7(13), 2014, pp. 1778–1783.

[27] Higgins, J., T. Al-Jody, and V. Holmes, Rapid Deployment of Bare-Metal and In-Container HPC Clusters Using OpenHPC playbooks, 2018.

[28] Tihfon, G.M., S. Park, J. Kim, and Y.-M. Kim, "An efficient multi-task PaaS cloud infrastructure based on docker and AWS ECS for application deployment", Cluster Computing, 19(3), 2016, pp. 1585–1597.

[29] Morabito, R., J. Kjallman, and M. Komu, "Hypervisors vs. Lightweight Virtualization: A Performance Comparison", in Proceedings, 2015 IEEE International Conference on Cloud Engineering (IC2E). 2015. IEEE: Piscataway, NJ.

[30] Felter, W., A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and Linux containers", 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). 2015. IEEE: Piscataway, NJ.

[31] Vaquero, L.M., A. Celorio, F. Cuadrado, and R. Cuevas, "Deploying Large-Scale Datasets on-Demand in the Cloud: Treats and Tricks on Data Distribution", IEEE Transactions on Cloud Computing, 3(2), 2015, pp. 132–144.

[32] Marz, N. and J. Warren, Big data: Principles and best practices of scalable real-time data systems, Manning, Shelter Island, NY, 2015.

[33] https://www.oreilly.com/ideas/questioning-the-lambda-architecture, accessed 6-15-2021.

[34] ISO, "Systems and software engineering: Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models", 35.080(25010:2011), 2011.

[35] Sommerville, I., Software engineering, Pearson, Boston, 2016.

[36] Volk, M., D. Staegemann, S. Bosse, A. Nahhas, and K. Turowski, "Towards a Decision Support System for Big Data Projects", in WI2020, N. Gronau, Editor. 2020. GITO Verlag.