applied sciences

Article

CloudOps: Towards the Operationalization of the Cloud Continuum: Concepts, Challenges and a Reference Framework

Juncal Alonso [1,*], Leire Orue-Echevarria [1] and Maider Huarte [2]

[1] TECNALIA, Basque Research and Technology Alliance (BRTA), Parque Científico y Tecnológico de Bizkaia, 48160 Derio, Spain; leire.orue-echevarria@tecnalia.com
[2] Department of Communications Engineering, Faculty of Engineering, University of the Basque Country UPV/EHU, Alda. Urquijo S/N, 48013 Bilbao, Spain; maider.huarte@ehu.eus
* Correspondence: juncal.alonso@tecnalia.com

Abstract: The current trend of developing highly distributed, context aware, heterogeneous computing intense and data-sensitive applications is changing the boundaries of cloud computing. Encouraged by the growing IoT paradigm and with flexible edge devices available, an ecosystem of a combination of resources, ranging from high density compute and storage to very lightweight embedded computers running on batteries or solar power, is available for DevOps teams from what is known as the Cloud Continuum. In this dynamic context, manageability is key, as well as controlled operations and resources monitoring for handling anomalies. Unfortunately, the operation and management of such heterogeneous computing environments (including edge, cloud and network services) is complex and operators face challenges such as the continuous optimization and autonomous (re-)deployment of context-aware stateless and stateful applications where, however, they must ensure service continuity while anticipating potential failures in the underlying infrastructure. In this paper, we propose a novel CloudOps workflow (extending the traditional DevOps pipeline), proposing techniques and methods for applications' operators to fully embrace the possibilities of the Cloud Continuum. Our approach will support DevOps teams in the operationalization of the Cloud Continuum. Secondly, we provide an extensive explanation of the scope, possibilities and future of the CloudOps.

Keywords: DevOps; cloud continuum; deployment; multi-cloud; hybrid-cloud; self-healing

check for updates

Citation: Alonso, J.; Orue-Echevarria, L.; Huarte, M. CloudOps: Towards the Operationalization of the Cloud Continuum: Concepts, Challenges and a Reference Framework. *Appl. Sci.* **2022**, *12*, 4347. https://doi.org/10.3390/app12094347

Academic Editor: Eui-Nam Huh

Received: 25 February 2022
Accepted: 23 April 2022
Published: 25 April 2022

1. Introduction

Cloud Computing evolution in the last decade and its transformation into a service utility has promoted a wide adoption by the industry for applications in general to store and process data. With the expansion of the IoT paradigm [1], the need for computational and storage services is expected to grow in the next few years, as well as the amount of data generated at the edge of the network. While cloud computing has been an effective way of acquiring computation and storage as a service to many applications, it may not be suitable to handle the endless data from IoT devices and fulfil the largely heterogeneous application requirements [2]. To this extent, some of the limitations of the traditional Cloud Paradigm particularly applies to applications that need a real time response, low latencies or those giving support to critical infrastructures. The centralized nature of traditional cloud services poses some limitations implying communication and data transfers to traverse multiple hops, which introduces delays and consumes network bandwidth of edge and core networks [3]. Computing capacity at the edge increased with the hardware evolution of personal devices and, as a result, proposed the utilization of edge devices to run applications and store data, bringing a new player: Edge Computing.

As a result, new approaches that effectively leverage distributed computational and storage infrastructure and services are necessary. These approaches must seamlessly combine resources and services at the edge (edge computing), in the core (cloud computing) and along the data path (fog computing) as needed, through the Cloud Continuum.

Thus, for application developers and operators to fully embrace this new paradigm, specific and in-depth knowledge on the plethora of underlying techniques and technologies is needed. Furthermore, the operation of such heterogeneous computing environments poses complex tasks for the operators of the applications as they must configure, plan, prepare and execute them, facing new challenges in all stages of the operation phase of the application:

- **Optimization in resource allocation**, which becomes more challenging as the number of variables (e.g., heterogeneous resources) increase as well as their dynamics (e.g., these variables change more often over time). The composition of the infrastructural services in the Cloud Continuum brings new variables as the heterogeneity of services and applications' components reach ground-breaking levels. This dynamic nature of the system, along with high levels of heterogeneity, creates a demand for dynamic, multi-criteria resource allocation strategies that can cope with the constantly changing environment;
- **Microservices management** throughout the Cloud Continuum stack presents challenges associated to the movement of services among the different levels (sensors, edge and cloud computing). The automatic adaptation of the execution of microservices must consider deployment location and context, but should also not neglect resource constraints that may exist at each level of the Cloud Continuum. To achieve this automatic and transparent adaptation, services' reconfiguration that considers quality of service requirements should be considered;
- **The heterogeneity of networks** across the Cloud Continuum ecosystem in regard to microservices' deployment and reconfiguration is also challenging. Standalone services can have network requirements of the data sources, which can be achieved through network technologies such as network virtualization and software defined networks (SDN). In this case, the need for a reconfiguration of services includes a reconfiguration of the network in order to ensure requirements' consistency. On the other hand, the composition of services with different requirements can also be enacted vertically in the hierarchy (from the network to the cloud through the edge), where a reconfiguration of services (and network, if necessary) is even more complex due to the heterogeneity of services in terms of computing needs and requirements (e.g., latency). Network topology is expected to constantly change along with device mobility and variable application requirements, introducing a more dynamic behavior in the system;
- **Data management and portability** at run time involves the design and deployment of policies, architectures and procedures, allowing the accurate management of the full data lifecycle, including strategies for placement and accessing it (e.g., measuring and quantifying the trade-off between placing data and services at the cloud or edge level, etc.) as well as performing the appropriate actions when data need to be ported to assure service continuity. Even if containers-based technologies can support and ease the portability of stateless components, complexity rises when addressing stateful components, where integrity needs to be maintained during the porting process while, at the same time, ensuring business continuity;
- Applying **federation concepts in the Cloud and in the Edge**: The pervasiveness of the technologies introduces a need to manage shared resources in a more intelligent, comprehensive manner that is less ad hoc. This has given rise to the concept of federation [4]. The Cloud Continuum will ultimately need some type of federation to manage how different sets of data producers (e.g., sensors, edge nodes) and data consumers (e.g., application components running on traditional cloud resources) can collaborate and share data. This will also imply the federation of the resources where these different elements are running;

- **Deployment and governance** properties will take on additional dimensions when considering heterogeneous distributed environments. While sensors and IoT devices will typically not have any extra capacity for hosting the federation properties, edge nodes could host these functions, depending on their actual capacity and the number of sensors associated with them. Scalability is an issue as an edge node will have a finite capacity that will define how many IoT devices it can manage, and how it can manage the sharing of data with external data consumers;
- **Security Assurance in Cloud/Edge/IoT:** Security Teams and Operators in the Cloud Continuum are challenged today with the task of providing and consuming services in a secure way, but often lack the proper tools to quickly identify, choose and compose the most suitable set of technologies and platforms that offer the ideal trade-off between functionality and security or privacy guarantees. Additionally, they are in a scenario where data or services might be migrated between IoT, Edge and Cloud when security implications (security, privacy, physical tampering or legal policies) are relevant and need to be addressed. Due to the peculiar features of edge computing architectures (i.e., heterogeneity, distributed architecture, massive data processing, location-awareness and volatility), the traditional data security and privacy-preserving mechanisms in cloud computing are no longer suitable. In particular, data security and privacy-preserving in edge computing have to be lightweight, fine-grained and distributed [5].

Key Takeaway: *Application developers and operators in the Cloud Continuum (CloudOps) face today the challenge of embracing the new paradigm but often lack the proper tools and mechanisms to configure, plan, prepare and execute these heterogeneous computational environments. Moreover, they are faced with the tasks of continuous optimization and autonomous (re-)deployment of complex context-aware stateless and stateful applications and data in a federated environment (including edge, cloud and network services) assuring service continuity and anticipating potential failures in the underlying infrastructure, especially in critical systems that must be resilient and whose response time becomes vital.*

The rest of the paper is structured in four main sections. Section 2 is dedicated to presenting the requirements and challenges in the operationalization of complex applications in the Cloud Continuum through the analysis of the related work and, in particular, in the fields that are most relevant for the article: benchmarking of infrastructural resources and application classification and profiling; deployment orchestration and optimization; self-learning through monitoring in hybrid environments and self-healing mechanisms for corrective actions and data portability strategies. Section 3 describes the CloudOps concept and workflow and presents the CloudOps reference framework. The main components of the proposed solution are defined, namely, application components classification and infrastructural requirements specification, resource discovery, optimized deployment configuration, deployment and self-healing, continuous monitoring and self-learning. An overview of how each of them works is provided. Section 4 describes the applicability of the solution and relates the advantages brought to specific application domains. Finally, Section 5 provides the conclusions, a general overview of the research and future work.

2. Requirements and Challenges in the Operationalization of the Cloud Continuum

Delivering innovative software faster gives companies a competitive advantage. Software delivery will continue increasing in the upcoming years, led by the giants such as Amazon who allegedly deploy new code every 11.7 s [6]. Microservices applications, containers, cloud infrastructure and edge computing, DevOps (Continuous Integration/Continuous Development), artificial intelligence driven software development and cloud agnosticism and cybersecurity have been identified among the trends in the current years [7–10]. Achieving excellence in continuous delivery and continuous deployment is one of the main reasons for software companies when putting in place a DevOps strategy. DevOps requires agile work processes and automated workflows which can only be achieved through the assurance of readily available IT infrastructure which is needed to continuously run and deploy the developed code. This can only happen within an automated workflow.

Among the challenges (Figure 1) reported when deploying edge applications, the following stand out [11,12]: (1) management of a large number of endpoints, that can be solved with an automated approach at operation time; (2) lack of skills or knowledge to perform said automation; (3) deployment strategy: deciding which components go on the edge and which ones go in the centralized node, usually, the cloud, to have an optimized application in terms of the non-functional requirements (e.g., latency, performance) and how the deployment is actually made, which needs to be simpler; (4) recovery when there is a failure should also be easier and more straightforward; (5) monitoring resource utilization across all nodes simultaneously to understand the overall Quality of Service (QoS) of the application; (6) orchestration tools that manage and coordinate many edge sites and workloads, eventually developing into a self-managed edge application; (7) tools to manage cloud and edge application life cycles, including: "the definition of advanced placement constraints in order to cope with latency requirements of application components, the provisioning/scheduling of applications in order to satisfy placement requirements (initial placement), and data and workload relocations according to internal/external events (mobility use-cases, failures, performance considerations, and so forth)".

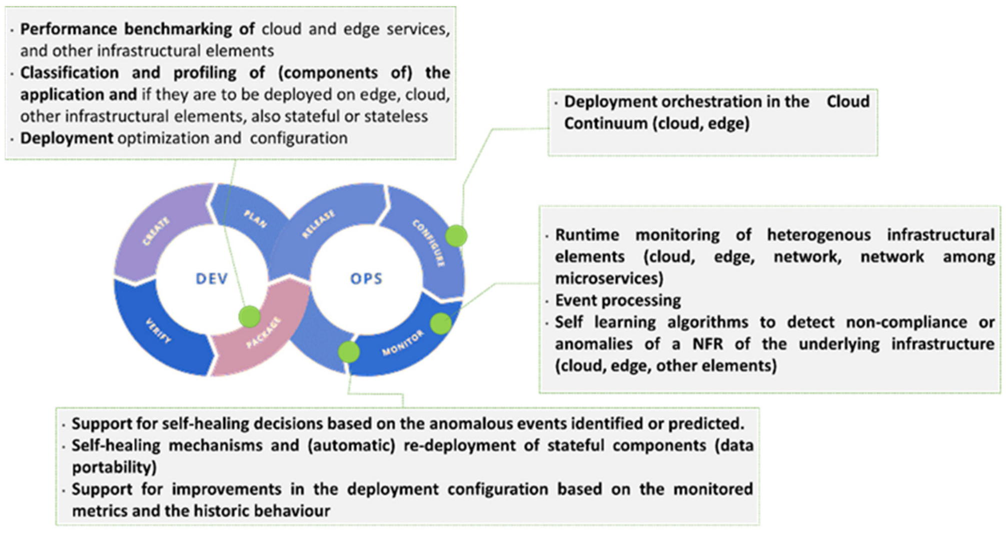

· Performance benchmarking of cloud and edge services, and other infrastructural elements
· Classification and profiling of (components of) the application and if they are to be deployed on edge, cloud, other infrastructural elements, also stateful or stateless
· Deployment optimization and configuration
· Deployment orchestration in the Cloud Continuum (cloud, edge)
· Runtime monitoring of heterogenous infrastructural elements (cloud, edge, network, network among microservices)
· Event processing
· Self learning algorithms to detect non-compliance or anomalies of a NFR of the underlying infrastructure (cloud, edge, other elements)
· Support for self-healing decisions based on the anomalous events identified or predicted.
· Self-healing mechanisms and (automatic) re-deployment of stateful components (data portability)
· Support for improvements in the deployment configuration based on the monitored metrics and the historic behaviour
CREATE
PLAN
RELEASE
CONFIGURE
DEV
OPS
VERIFY
PACKAGE
MONITOR

Figure 1. Challenges in the Operationalization (Ops) of applications in the Cloud Continuum.

The analysis of the related work in the topics relevant to the main challenges (Table 1) presented is detailed next.

2.1. Benchmark of Resources and Application Classification

Given the heterogeneity of the cloud resources and the types and complexity of the applications it is not trivial to select the optimal resources (or combination of resources) for each application (or application component) in order to maximize its performance [13], or to choose the best performance/cost trade-off [14]. That is why benchmarking is a widely used technique in cloud computing. Moreover, when shifting from the cloud to the edge computing, benchmarking techniques are seen as even more valuable because of the high diversity and variability of edge nodes in terms of hardware, software stacks, energy consumption or connectivity capabilities. A common practice used in the cloud [15] is to rely on specialized benchmarking tools [16–18] to simulate how an application would behave when deployed on different cloud options. The extension of these techniques to the edge, however, still poses some new challenges [19] such as the limited capabilities of edge nodes needing more lightweight benchmarking tools, specific workloads running in the edge (e.g., virtual reality, image processing) being different from the workloads in the cloud (e.g., data analysis, searching, video streaming), and benchmarking composition (cloud + edge nodes) in order to provide information on complex deployment options. These challenges are currently being tackled by different initiatives through different ongoing works: characterization of the most common edge use cases and their workloads [20], selection of metrics to define their performance [21] and the implementation of new suites of benchmarking tools designed to run in the edge for different use cases such as speech decoding and face recognition [22], surveillance cameras and autonomous vehicles [23], etc. A big challenge today in existing service catalogues is that, in particular, the information about security attributes, such as certificates or compliance to standards, is not available. In [24], a solution is proposed that allows the discovery and monitoring of security-related aspects on what it is called the Legal Level. This Legal Level assesses the regulation and legislation supported by the different Cloud Services through the verification against evidences included in CSPs contract forms. Nevertheless, new metrics and aspects related to privacy and security and other infrastructural elements (i.e., edge nodes, IoT agents) need to be addressed.

Table 1. Current unsolved challenges in the Operationalization of applications in the Cloud Continuum.

Topic	Current Challenges
(Dev)Ops	• Partial automation of the DevOps cycle especially in heterogenous environments • Context (cloud, edge, network) specific tools • Specific skills needed, especially for "combined" environments, with cloud services, edge nodes and network elements
Monitoring	• Inefficiencies mainly due to lack of dynamicity of the distributed proposed models • Lack of holistic approaches covering not only the monitoring of traditional cloud resources but also the integrated monitoring of edge nodes and network communications • Communication channels between application elements (e.g., microservices) are usually considered at device level but not at software components level (even inside the same device) • Lacks the precision or speed needed for high performance architectures as existing solutions for forecasting methods are based on basic offline algorithms such as linear regression or nearest neighbors adapted to online environments
Benchmarking	• Benchmarking of edge resources is incipient and existing solutions are platform or scenario specific • Existing tools usually involve inefficient or ineffective tests as the complete benchmarking process is not automated • Current approaches for application classification are usually focused on specific applications types (e.g., data streaming applications) and not suitable for other types • The characterization and benchmarking of networking and communication have been barely addressed
Self-learning, self-healing	• Few available solutions tackle the automatic self-healing of applications based on failure prediction • Existing self-healing works do not consider stateful components' needs (e.g., data portability issues) • Current approaches focus on certain aspects of the self-healing process, usually the re-deployment of the application, but do not offer integrated frameworks covering previous or subsequent tasks such as failure prediction, new resources contracting or "old" devices and services shut down

Table 1. *Cont.*

Topic	Current Challenges
Cloud Continuum	• Few to no available approaches supporting the Cloud Continuum at pre-deployment and operation time
Security Assurance in Cloud/Edge/IoT	• Services and tools to implement SecDevOps as a whole in a Cloud Continuum are missing • Security and privacy assurance in Cloud/Edge/IoT • Active Data Protection in the Cloud Continuum

Besides, application and application components need to be classified (e.g., stateful, stateless) and profiled (e.g., computation, data, serverless) in an agnostic manner, without requiring a priori knowledge about the application internals so that the deployment needs can be inferred and deduced. However, most of the available applications profiles are platform- and scenario-dependent and hardly reusable in other contexts. The NAMB project has started to work in the direction of a generic approach that is context and platform agnostic [25,26]. Moreover, benchmarking of the whole stack, including the network resources, in particular the paths that interconnect the physical or virtual machines hosting application components, has not yet been deeply addressed. In that respect, the profiling of the paths within the cloud and the edge will provide relevant information which can impact on the decision on the selection of the best combination of resources for the deployment. The understanding of these paths is not trivial because public cloud service providers such as Amazon Web Services and Microsoft Azure provide links that interconnect data centers with high performance [27] opposite to typical best-effort Internet links [28]. However, those links present complex traffic engineering techniques, which require careful design and tuning of monitoring tools in order to avoid biased measurements. Additionally, the paths that extend from the cloud up to the edge expectedly offer less predictable performance than cloud links [29].

Osmotic computing [30] is another approach which "aims to decompose applications into microservices without degradation of QoS and perform dynamic tailoring of microservices in smart environments exploiting resources in edge and cloud infrastructures".

Unfortunately, decomposition of applications into microservices is not always possible. Therefore, stateless and stateful components will coexist in the Cloud Continuum. The benchmark of stateful applications is based on latency, scalability and elasticity, and developers on the cloud try to incorporate these parameters into their designs [31]. On the other hand, stateless applications are assessed in terms related to the modularity of the application, such as control and flexibility.

2.2. Deployment Orchestration and Optimization

The Cloud Continuum concept encompasses the idea of processing each application and/or application component with the most appropriate resource, from cloud computing systems to IoT devices, depending on different factors: the need for resources, proximity, etc. In this complex scenario, the selection of the optimized deployment configuration is still a challenge to be solved. The selection of the best combination of resources can be treated as an optimization problem which can be solved through different types of techniques from the three main categories: exact methods [32]; heuristics [33] and metaheuristics [34], the last ones enjoying greater popularity because of their adaptability and efficiency [35]. The literature is scarce in studies centered on the optimal deployment of microservice over heterogeneous cloud services. In [36], a NSGA-II algorithm was used to match the specific requirements of a certain Big Data application to the capabilities provided by an IaaS infrastructure and the Big Data platform deployed therein against three design aspects for the infrastructure of the Big Data application: cost, reliability and net computing capacity. Similarly, in [37], two main objectives were considered for optimization: (1) fulfilment of the microservices' Non-Functional-Requirements (NFRs) (location, availability, cost, performance and legal level), and (2) meeting of the characteristics set by the developers of these microservices (classification, public IP, disk space, RAM and number of cores). The optimization process described above can be enriched by having information about the behavior of the execution environment and the underlying resources at run-time, so that the optimization function can also consider this information as another dimension of incoming data.

The deployment of applications in general has benefitted in the last decade from the Infrastructure-as-Code (IaC) [7] concept which aims to automate the provisioning, configuration and deployment of infrastructure resources following a machine-readable file approach. Tools for IaC process configuration files developed by Ops teams for different purposes include: Provisioning; Configuration; Deployment and Orchestration of the resources. However, there are several open issues that still need to be solved, especially when talking about automatic deployment and orchestration in the Cloud Continuum:

- Tools [38–41] for virtual and physical resources provisioning are platform dependent lacking interoperability and portability and mainly focused on cloud resources and do not consider edge nodes;
- Some Continuous Delivery and Continuous Deployment supporting tools are specific for deploying containers-based applications (Rancher [42]) while others are more generic (e.g., Apache Brooklyn, Spinnaker [43], Alien4Cloud [44], Cloudify [45]) and can deploy both traditional and container-based applications;
- Kubernetes [46], the leading, most used and advanced orchestration platform today has been used for the orchestration of traditional cloud resources. When it comes to the edge though, new features such as low network latency [47] and self-healing need to be considered due to the edge's higher proneness to failures that impact on maintenance costs. Emerging Kubernetes on the edge open source solutions such as K3S, MicroK8S and KubeEdge are getting huge interest by the Kubernetes' edge community and aim to leverage on the existing ecosystem through official Cloud Native Computing Foundation certification, while at the same time are light enough to run on low-cost boards;
- The management of IoT devices, especially those with high constraints (i.e., not always connected, lack of containerization support), needs to be addressed so that the deployment can be done supporting key edge characteristics, such as mobility, heterogeneity and volatility [48].

2.3. Cloud Resources Monitoring and Self Learning in Hybrid Environments

Cloud resources' monitoring is a challenging task when applied to hybrid environments. Cloud providers claim to support extensive monitoring mechanisms to aid in controlling application performance and working conditions but they barely offer transparent access to actual law level metrics (e.g., Mean Time To Recover (MTTR), Mean Time Between Failures (MTBF), real time CPU consumption, etc.) nor provide common or standardized metrics that can be compared among different cloud resources types or providers. In [49], the authors identified the QoS and Service Level Agreement (SLA) assessment in complex hybrid scenarios as one of the four clear gaps of the cloud computing management, due to the lack of mechanisms to address the particularities of large-scale cloud setups with more complex environments in terms of resource heterogeneity and distribution, such as hybrid and multi-cloud scenarios. In [50], the authors proposed an approach to adapt the planning in Complex Service-Based Systems, should an SLA violation occur. However, in this dissertation the authors assumed that a violation occurs but do not support the actual monitoring nor go in depth into how the NFRs are monitored or calculated so that the violation can be identified.

In [37] monitoring of cloud resources was identified as one of the missing functionalities in current solutions and approaches. This conclusion was extracted from an analysis of different multi-cloud abstraction solutions, as cloud brokers, cloud intermediators or cloud marketplaces which automate the assembly of complex applications, its deployment and operation on one or multiple cloud infrastructures. This approach proposes using a combination of push and pull monitoring [51] for relevant NFRs such as performance, cost, availability of location and supporting the ISO 19086.

In [52] a cloud monitoring stack is proposed for different Cloud Services (virtual machines, databases as a service, etc.) from different providers (Amazon, Azure, ARSYS). Furthermore, it focuses the monitoring of the resources on the lower level, considering Fowler's [53] definition. Nevertheless, it doesn't tackle the communication layer, which in some specific scenarios can pose relevant challenges [54]. The approach proposed in this paper tries to extend it to support the monitoring of distributed and heterogeneous (different nature) infrastructural elements. These metrics will be provided both to the self-learning and self-healing algorithms as extra input data to enhance their results.

Monitoring and preventing QoS violations are a critical aspect of the design and planning of infrastructural services, hence for the operationalization of applications, in which predictive models play an essential role. Initial *post-mortem* analysis techniques [55] applied to the management of these violations have made way for new approaches focused on the anticipation of the failures, first as an offline process [56] and more recently at run time [57,58]. More recently, and due to the nature of heterogeneous distributed services especially in highly demanding environments, the Internet of Things (IoT) has become one of the main applications of stream learning [59], since it is composed of sensors and actuators connected by networks to computing systems, which manage the health and actions of connected objects or machines in real-time. One of the main problems of the Cloud Continuum paradigm referring to the prediction of QoS failures is that they are subject to configuration drifts, unauthorized changes, missing dependencies or any invisible or not monitorable environment variations that have arisen in the services. However, techniques such as concept drift [60,61] or anomalies detection [62,63], that could provide relevant benefits, have not attracted much attention yet in the research community.

2.4. Self-Healing and Data Portability

The application of self-healing, refers to the autonomous and responsible behavior of the applications to changes in the execution environment, and has been a challenge since the cloud and distributed computing paradigms appeared on the software operation gameboard. Several solutions have been proposed for specific scenarios such as IoT [64,65] or traditional cloud environments [66]. Other solutions are focused on specific steps in the self-healing, self-configuration process [67], or in the resolution of specific problems such as scalability [68] or trust enforcement [69]. The cross/multi-layer and the networking aspects are challenges that have not yet been addressed, nor has the face of the problem in a generic way, covering the whole self-healing process from the discovery and configuration of the resources to the network preparation and deployment of all software layers.

Self-healing mechanisms can include several actions, going from the starting up of the failing resource with a new one, to the re-configuration of the network or the orchestration of portability workflows so that business continuity is ensured. Several approaches have been applied to automatically create computational resources (usually virtual machines) but few initiatives can be found when addressing heterogeneous environments of different natures (edge and cloud) and networking set ups. In such a complex environment, self-healing actions may need dynamic orchestration of the services used for re-deployment following the proper workflow, as well as network resources' re-allocation and setting up.

This process becomes even more complex when addressing not only the portability of the computational components (stateless components) but also the portability of data, or stateful components. One of the enablers for such portability can be adopted from the application point of view, through the adoption of containers-based technologies, such as Docker. However, containerization per se does not solve the portability problem. When porting components between two cloud providers, data need to be moved and kept synchronized (at three different levels—blocks, files or transactions) and most container-based platforms don't handle this. Some containers' orchestration systems, such as Kubernetes, provide convenient abstractions for application resource requirements and endpoints, namely persistent volumes, ingress rules and ingress controllers [70]. However, manual configuration is still needed, and stateless components are decoupled from the data that have to be stored in a database or other type of storage. Another important concept to be considered when porting data components from one resource to another is the "data gravity", a term coined by Dave McCrory [71], who explained that, because of network bandwidth limits, latency, costs and other considerations, data "want" to be near the applications analyzing, transforming or otherwise working on it. Consequently, data portability introduces new requirements added to the already time consuming, error prone and mainly manual activity of porting application components over different infrastructural elements [72], such as: (1) Establishing the right networking conditions, so that data can be accessed from the required microservice with the required (network) conditions; (2) Handling persistent data storage, during a redeployment; (3) Data Base automatic configuration so that it can be re-deployed without manual intervention. Security is also to be addressed when selecting a porting strategy. Mechanisms for automatic execution of secure data portability in the Cloud Continuum, preserving data integrity and assuring that the data are not being corrupted need to be considered.

3. CloudOps: Concepts and a Reference Framework

3.1. CloudOps Concept

The current IT market is more and more dominated by the "Cloud Continuum". Increasing ubiquity and the pervasiveness of compute capabilities and data availability have resulted in the proliferation of complex applications which effectively process data from heterogenous digital sources in a timely manner [73]. As the types and volumes of available data grow, new applications are implemented that seamlessly combine real-time data with complex models and data analytics to monitor and manage systems of interest. The data have to be processed promptly to extract insights that can drive decision making. Traditional approaches that rely on moving data to remote data centers for processing are no longer feasible. Instead, new approaches that effectively leverage distributed computational infrastructure and services are necessary. Specifically, these approaches must seamlessly combine resources and services at the edge (edge computing), in the core (cloud computing) and along the data path (fog computing) as needed, through the Cloud Continuum. At runtime, applications can choose to execute parts of their logic on different infrastructures that constitute the continuum, with the goal of minimizing latency, energy consumption and maximizing availability or performance [74]. This requires novel techniques and methods for federating infrastructure, programming applications and services and composing dynamic workflows, which are capable of reacting in real-time to unpredictable data sizes, availability, locations and rates [75].

The heterogeneity of the "computing continuum" is broad and multistage. In the "traditional" cloud, computing resources are typically provided through virtualization and containerization [74], with "infinite" resource availability thanks to horizontal scaling. In contrast, in edge computing, computational resources are scarce and must be managed very efficiently due to battery constraints or other limitations [76].

The combination of such technologies implies that the DevOps teams need to have a deep knowledge of the underlying techniques and technologies and need to be able to work with several of them, seamlessly integrated. These kinds of heterogeneous computing environments pose complex tasks for the operationalization of the applications by the DevOps teams as they must configure, plan, prepare and execute them. When applying it to dynamic, changeable execution environments where the applications need to be reconfigured under real time tight requirements, this process needs to be short, repeatable and (semi-)automatic. Application operators need to: (1) have real time updated information on the network conditions, computing resources available and data infrastructures and services requirements and (2) be able to seamlessly and transparently adapt the deployment configurations of the applications to respond to the execution environment situation. In effect, applications' operators need to seamlessly select, combine, configure and adapt computation resources all along the data path and support the complete service lifecycle, covering: (1) distributed application deployment over heterogenous computing resources; (2) monitoring of execution platforms in real-time; (3) application deployment and adaptation while optimizing the execution and (4) application of self-healing to avoid compromising situations that may lead to an unexpected failure.

In Figure 2, the proposed CloudOps workflow is presented. It includes the extension and customization of the Ops cycle with activities and supporting techniques addressing the specific needs and challenges of the applications in the Cloud Continuum:

- Complexity in the discovery, selection and configuration of diverse and heterogeneous IT infrastructural components (cloud, edge and fog computing) to select the optimal deployment configuration in terms of availability, rates or other;
- Implementation of consolidated architectures and automatic deployment mechanisms for both the edge deployment targets, as well as traditional cloud computing services;
- Operation and proactive monitoring of the dynamic and heterogenous Cloud Continuum to be aware of new available resources, bottlenecks situations or network reconfiguration needs;
- Failure prediction based on historic data supporting the automatic reconfiguration of the applications that need to respond intelligently to changes, avoiding compromising situations;
- Self-healing mechanisms for applications to be capable of being re-deployed given the new environmental conditions;
- Seamless components' portability, without affecting the service continuity and addressing special needs for data and stateful components' migration over different service providers.

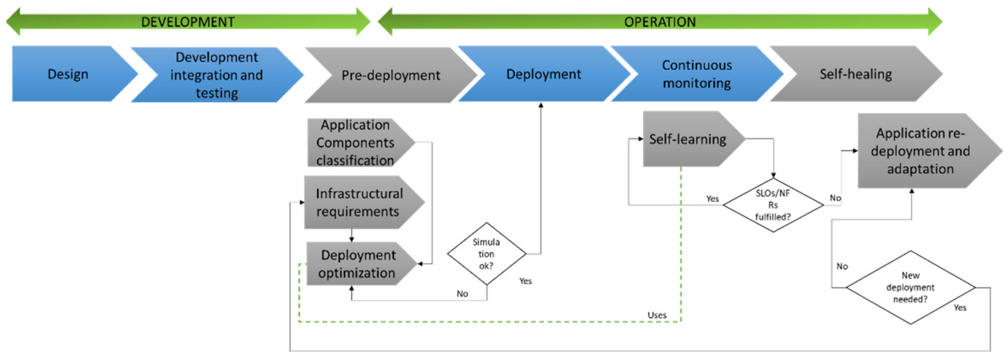


Figure 2. CloudOps workflow for the operationalization in the Cloud Continuum.

The traditional Operation workflow is extended and customized with new activities supporting the specific needs of applications sunning in the Cloud Continuum (gray boxes in Figure 2). These activities are detailed along with the supporting proposed techniques in Section 3.2.

3.2. CloudOps: A Reference Framework for Operationalization in the Cloud Continuum

This section presents the CloudOps reference framework. The main goal of the CloudOps reference framework is to support the operation of complex applications in heterogenous execution environments. To achieve this, several practices and techniques are suggested: increase the optimization, adaptability and portability of services to different resources and usage contexts while ensuring business continuity; provision of self-healing mechanisms to anticipate failures and violations and self-learning from the conditions that triggered such re-adaptations.

As illustrated in Figure 3, the proposed referenced framework is composed of six functional blocks to support the operation lifecycle of the applications in the Cloud Continuum.

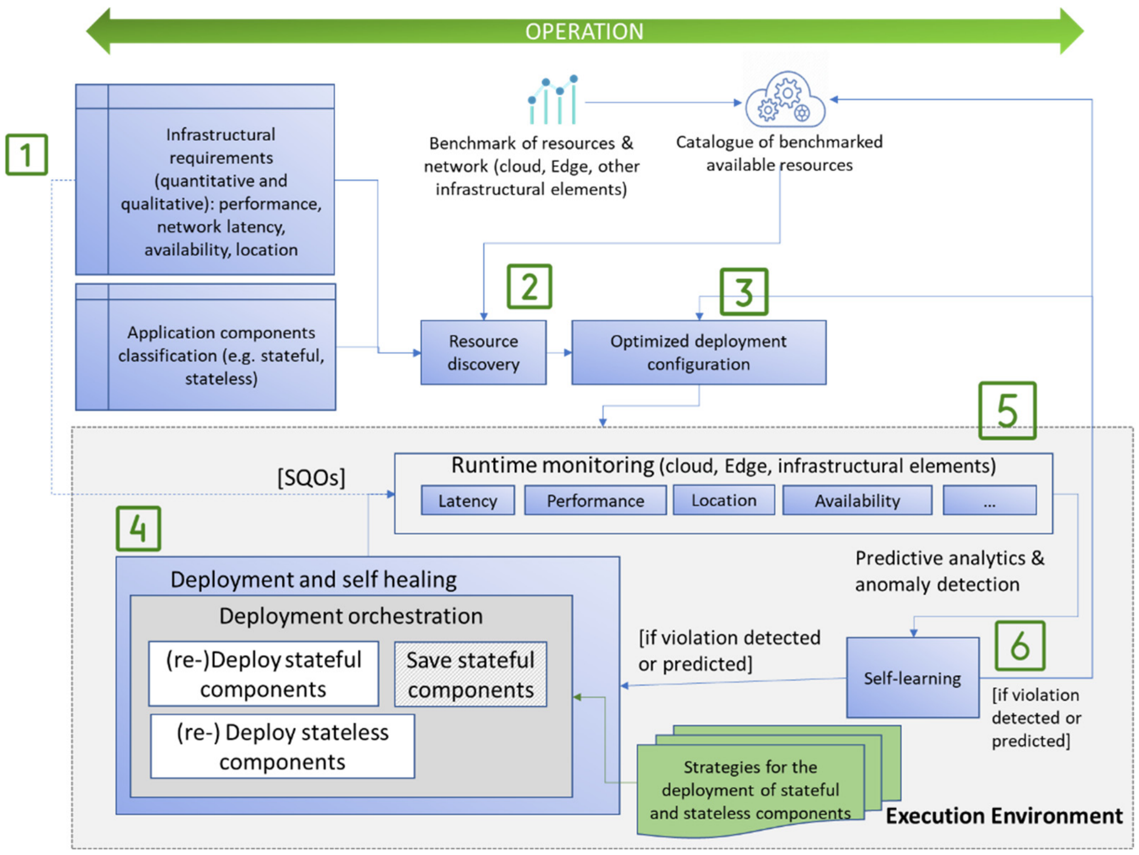

OPERATION
1
Infrastructural requirements (quantitative and qualitative): performance, network latency, availability, location
Application components classification (e.g. stateful, stateless)
Benchmark of resources & network (cloud, Edge, other infrastructural elements)
Catalogue of benchmarked available resources
2
Resource discovery
3
Optimized deployment configuration
5
[SQOs]
Runtime monitoring (cloud, Edge, infrastructural elements)
Latency
Performance
Location
Availability
...
4
Deployment and self healing
Deployment orchestration
(re-)Deploy stateful components
Save stateful components
(re-) Deploy stateless components
Predictive analytics & anomaly detection
[if violation detected or predicted]
Self-learning
6
[if violation detected or predicted]
Strategies for the deployment of stateful and stateless components
Execution Environment

Figure 3. CloudOps reference framework for the Operationalization of applications in the Cloud Continuum.

1. **Application Components classification and infrastructural requirements specification** This component will support the specification of both the application components' characteristics which needs to be considered for their correct operation in the Cloud Continuum avoiding incompatibilities and the definition of the qualitative and quantitative infrastructural components needed to configure the heterogeneous (cloud, edge, network) execution environment. In the same way, the operators of the application need to identify the infrastructural and communications' requirements. Among other things, they identify the need for provisioning certain services, for configuring certain network functions, security requirements, network latency values or performance objectives for the infrastructural environment. Semantic techniques will be considered to enable the classification (e.g., stateful, stateless), profiling (e.g., computation, data, serverless) and categorization (e.g., nature, underlying technology, security constraints) of the components of the applications to be deployed. The usage of a taxonomy and the corresponding ontology can support the automatization of this with tools to automatically make an initial identification and classification of the components and application architecture, considering various sources such as deployment scripts (e.g., Kubernetes YAML files, Dockerfiles) and annotations in the source code. The user will then have to refine and further detail the application classification and to express other deployment requirements that will be input for the deployment orchestrator to improve the deployment configuration. The user will also have the option to exploit the application classification to generate mock-applications which can be deployed and monitored (through the benchmarking process) to have a good estimate of the performance of a set of resources. This approach will enable the user to assess the suitability of a set of candidate resources (e.g., a specific combination of cloud and edge devices) on which its final application could be deployed, without having to fully specify the application.

2. **The Resource discovery** component will assist the operators in the discovery and benchmarking of the available infrastructural resources, services and network elements. To do so it will provide: (1) support for infrastructural and network benchmarking with respect to relevant non-functional characteristics, such as performance, location, availability or others and the network, so that the different available network elements can be measured against established Service Level Objectives (e.g., network performance, availability, latency); (2) a catalogue of classified and categorized available benchmarked infrastructural elements, networks, services and resources. These elements will conform to a heterogeneous federated execution environment covering all the paths from the distributed edge nodes to the central cloud computing infrastructures, including the network elements which interconnect the other execution resources. The volatility of the resources, especially the IoT agents, needs to be also considered by this component. The catalogue of available benchmarked infrastructural elements, networks, services and resources requires updated information of the status of the catalogue and its available resources to accommodate the deployment to resource volatility and real-time service demands. Therefore, specific mechanisms to manage the huge, heterogeneous, volatile and dynamic set of resources from the edge up to the cloud need to be incorporated [49].
3. **The Optimized deployment configuration** component will gather information coming from both the "Application Components classification and infrastructural requirements" for the components' classification and the "Resource discovery" for the benchmarked catalogue of resources and services, to provide the best combination of those infrastructural and networking elements. The obtained optimized deployment configuration schema, based on the needs expressed by the operators both for the application components and for the infrastructural elements, will then be applied in the Execution Environment. Using optimization algorithms, it will seek an optimized deployment configuration of the application on heterogeneous resources that best meet the predefined constraints (e.g., types of infrastructural elements, NFRs, classification of the components of the application). The application of machine learning and swarm intelligence, such as multi-objective meta-heuristics algorithms (e.g., NGSA-II, MOCell or SMPSO), can support the operators in the automatic identification of the optimized configuration in any of these two situations: (1) Whenever a first deployment configuration of the Cloud Continuum is required; (2) Once the potential situation of risk in relation to an SLA violation is detected. To this end it must be able to run in a dynamic-definition environment where the optimization parameters (resources to be configured), parameters' vocabulary (resource configuration candidates) and the fitness/objective function to optimize (measuring predetermined NFRs' accomplishment as well as solution robustness with regard to real-time monitoring metrics) are likely to mutate as a result of each use case specificity.
4. **Deployment of the self-healing** component seeks to automate the configuration and orchestration of the different elements conforming to the selected infrastructure, as well as the deployment of the software elements needed for the application to run and deployment of the components themselves. It will support the operators in automating the orchestration of the main tasks of the Execution Environment, creating a deployment plan composed of a set of tasks and distributing these tasks among the different subsystems in charge of the actual provisioning of the resources (e.g., creation of the virtual machines using proper IaaS connectors, installation of specific software packages or configuration of network nodes). It will distribute applications among the continuum of devices (IoT devices, on-premises devices and on cloud resources) in order to perform as required. The proposed approach should take into account the heterogeneous hardware capabilities of the hardware devices present in the Edge and Cloud Continuum, e.g., by exploiting GPU capacities in those devices which have it present. In addition, an execution environment is needed to reproduce any kind of production environment by creating arbitrary infrastructures

and deploying any kind of software over it. Containerization will allow applications to be executed and moved throughout the Cloud Continuum. This component will also support the automatic re-deployment of the components into a new Execution Environment to ensure that the infrastructural and network elements are always conforming to the SLAs committed with the end-user, even if the environmental situation changes. The self-healing process will include the selection of the different and most adequate strategies to (re-)deploy the different components based on their nature. Special focus will be put on the provision of a set of strategies and techniques to ensure the integrity of stateful application components when porting data from one resource to another. The Deployment and Self-Healing component will support the operators in the implementation of these strategies so the actual portability of the application components from the running resources to the new optimized deployment configuration is successfully completed.

5. **Run-time monitoring component** has as main goal to continuously monitor the functioning conditions of the federated resources from Service Level Objective (SLO) and NFR accomplishment perspective. Once all the components are running in the different resources the it will log the working conditions of the execution environment at different levels, including all the underlying computational and network elements, (e.g., network access of virtual and physical machines, and continuous micro-service level response monitoring). The Run-time monitoring component will gather metadata and metrics about the selected Non-functional/QoS properties (latency, performance, location, availability, etc.), so that the business continuity of the application in the Cloud Continuum is always guaranteed. This system will ensure not only that the conditions are always met but will also be able to provide all the run-time monitoring information to the self-learning component so that it can exploit this data to predict potential failures. Different monitoring strategies shall be combined as the monitoring requirements may vary according to the exact objective, which might be either to monitor the performance of the hosting components of a micro-service (VMs, pods, containers) or a micro-service itself. The network level also needs to be considered gathering either measurements out of the micro-services traffic itself are derived or measurement probes on the network. The actual choice depends on the allowed level of intrusiveness and/or the monitoring budget, which will be a function of the location of the component, edge or cloud. This shall involve techniques for monitoring data aspects (e.g., new data sources, throughput changes of incoming data, etc.) and data operations (e.g., distributed query processing time, generated outputs of different application components, etc.) that might trigger runtime adaptations of the infrastructure. Events processing functionality can help coping with heterogenous and dispersed monitoring data that highlight health-status and the performance aspects of applications, data and hosting infrastructure. The event processing needs to be context-sensitive by dynamically engaging different monitoring probes and processing functions according to the applications' classification aspects (e.g., stateful vs. stateless applications, data stream processing vs. batch processing etc.) and deployed in a distributed manner (e.g., region, availability zone, VM and/or edge group level) for avoiding the aggregation of all the monitoring data in a centralized manner. With this approach, event processing will result in decision making and event propagation and persistence in time-series database containing only the most valuable monitoring information.
6. **The self-learning component,** based on the processed monitoring information, will be able to detect if a failure or non-compliance of the NFRs is likely to occur, thus enhancing the platform's failure detection functionalities. This enhancement refers to the fact that it is unrealistic for a DevOps team to predefine, at the design phase, all the possible latent factors that may lead to SQO or SLO violations. The Ops team will be then informed, and a redeployment will be automatically triggered. The prediction process will leverage on advanced learning strategies such as Concept

Drift and Incremental Learning to detect that a failure, or non-compliance, is about to occur based on the analysis of application health-status events. Such alerts, along with other monitoring data and prediction probabilities, will be visualized accordingly, to empower DevOps teams with final decisions on configuration and prevalence of predictions that may bypass or reconfigure in real-time the automatic decisions of the system.

Figures 4–6 present an overview of how these six functional blocks work together and depicts their main interactions through the CloudOps workflows.

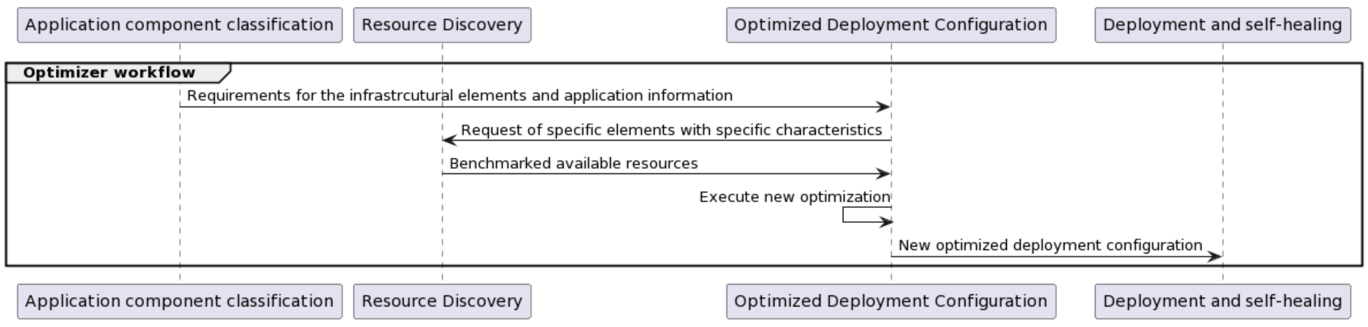

Application component classification
Resource Discovery
Optimized Deployment Configuration
Deployment and self-healing
Optimizer workflow
Requirements for the infrastrcutural elements and application information
Request of specific elements with specific characteristics
Benchmarked available resources
Execute new optimization
New optimized deployment configuration

Figure 4. Sequence diagram of the proposed Cloud Ops Optimizer workflow through the reference framework.

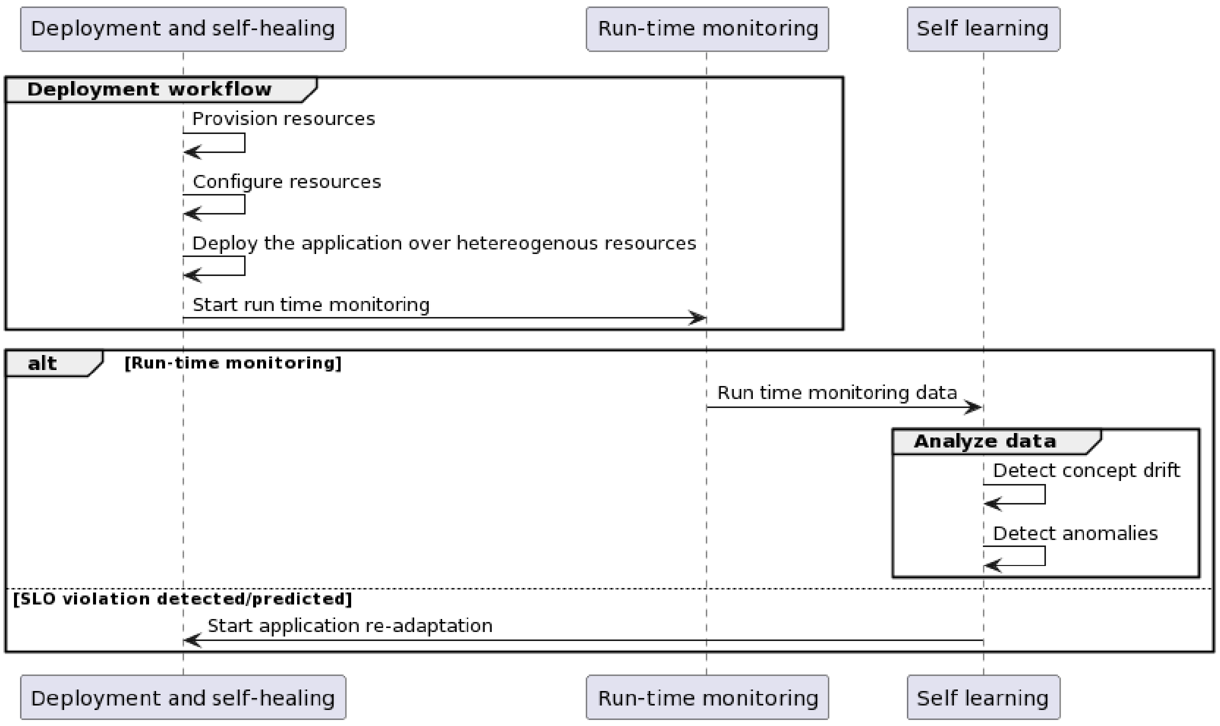


Figure 5. Sequence diagram of the proposed Cloud Deployment workflow through the reference framework.

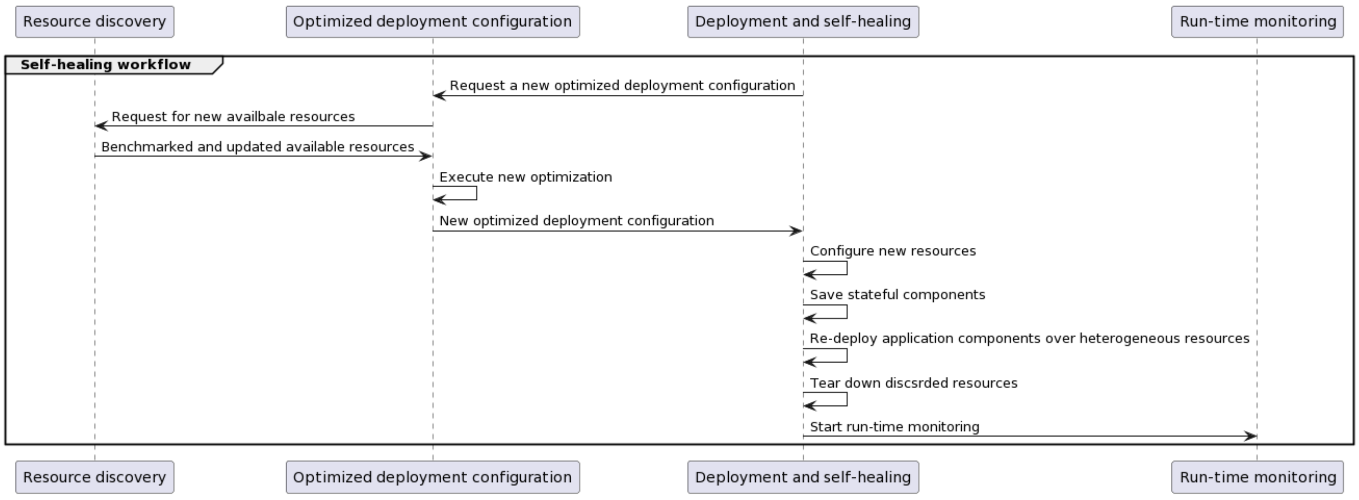

Resource discovery
Optimized deployment configuration
Deployment and self-healing
Run-time monitoring
Self-healing workflow
Request a new optimized deployment configuration
Request for new availbale resources
Benchmarked and updated available resources
Execute new optimization
New optimized deployment configuration
Configure new resources
Save stateful components
Re-deploy application components over heterogeneous resources
Tear down discsrded resources
Start run-time monitoring

Figure 6. Sequence diagram of the proposed Cloud Self-healing workflow through the reference framework.

4. Scope and Applicability

The presented CloudOps approach along with the CloudOps reference framework will assist (Dev)Ops teams in the operation of new cloud-based services, able to exploit the advantages of multiple and heterogeneous clouds and execution environments. This CloudOps framework leads to provision of a set of advantages for the end users that are summarized in Table 2.

Table 2. Expected benefits for the end users of the CloudOps reference framework.

Expected Benefits for the End Users of the CloudOps Reference Framework
Decrease the time needed to select the most appropriate resources and services in respect to the application needs. Have a catalogue where services can be compared with respect to certain non-functional characteristics through benchmarking.
Understand better the characteristics of the application and the computational and storage needs that it might have, as well as its condition (stateful or stateless), needed to define data portability strategies.
Be able to select the optimized deployment configuration out of a predefined set of constraints, using optimization algorithms.
Be able to optimally execute, orchestrate and deploy complex Cloud Continuum compliant applications, planning, provisioning and preparing both the infrastructure and the application in an automatic way.
Gather and process metrics that are able to predict anomalies and detect failures before they happen, so as to ensure the business continuity of the application.
Improve the quality of service of the Cloud Continuum compliant applications through the provisioning of self-learning mechanisms that, by the gathering of metrics, are able to predict anomalies, detect failures and trigger self-healing strategies that will allow the reconfiguration and re-deployment, wholly or partially, of the application in an improved configuration.
Maintain, preserve and port automatically the data of a stateful component once a redeployment activity is launched.
Improve the operation of Cloud Continuum compliant applications, ensuring business continuity, QoS and integrity of the ported data.

The application of the CloudOps approach and the supporting reference framework will bring benefits to the software industry operating in complex Cloud-edge environments. Nevertheless, some of the limitations of the traditional centralized Cloud paradigm applies in particular to applications that need real time, low latency or those giving support to critical infrastructures.

In particular, the presented CloudOps approach can be of special relevance when tackling critical infrastructures or business-critical applications where the response time becomes vital and the system must be resilient and able to operate in different scenarios, including lack of connectivity that can be replaced by alternative communication technologies (e.g., new edge nodes). Critical infrastructure is the body of systems, networks and assets that are so essential that their continued operation is required to ensure the security of a given city or region, its economy and the public's health and/or safety. Those critical infrastructures must have a very high availability rate, even higher than five nines. Following on from that, how the operation of critical infrastructures-based systems can benefit from the approach and the reference framework proposed in this article is explained through two specific examples.

4.1. *Critical Infrastructures for Safety and Security*

The advent of smart cities is fostering the deployment of different types of infrastructures to complement and provide public security services with data to detect threatening situations. Systems such as smart street lighting infrastructures supporting the security and safety of citizens and goods in public spaces are gaining momentum. However, these systems need to manage the large quantity of data generated, e.g., by surveillance cameras, therefore, AI-based systems are required to automatically analyze and filter such data to report only on potentially threatening situations. The large variety of possible situations and the large quantity of data generated implies an obligation for the systems to be able to adapt to these continuously changing situations, which may also be completely different depending on the city neighborhood, seasons of the year or specific events., Machine learning technologies will be required to take advantage of algorithms that may need to be adapted and redeployed depending on the detected situations. In the case of smart lighting infrastructures, the CloudOps reference framework can provide support to the optimization of the deployment configuration in an environment with continuously changing requirements and a limited processing capacity. Additionally, the algorithms that analyze the data may suddenly need high computational capacity in specific locations and times, and permanently providing such computational capacity up to the edge would be neither affordable nor sustainable, so there is also the need of flexibility to take advantage of the computational power of other edge nodes or the cloud. To achieve this, runtime monitoring and self-healing mechanisms can support the operators in the operation of such complex environments, providing them with the flexibility to execute code at any time in the most convenient computing resource, the capacity to update and to re-deploy algorithms whenever required and keeping investment and operational costs affordable.

4.2. *Critical Infrastructures for Emergency Management*

The management of public events and crowd control is a typical example of mission critical services; a service downtime could undermine citizen security and create problems in the service continuity leading to very serious issues. This type of system collects data from people using in-place nodes, e.g., Wi-Fi/Bluetooth. With this information the system is able to estimate the number of people in a specific area and, taking advantage of the bandwidth and low latency features offered by 5G connectivity, send these data to the edge nodes so that they are able to determine if a critical condition is going to happen, for example, if the flow of people is coherent with the data collected on the buses or in other public transportation systems. In this situation, if an edge node goes down, fault-tolerance mechanisms need to be put in place. However, while in many other cases it is possible to use fault-tolerance features of many of the most common orchestrators (such as Kubernetes) this is not possible for stateful services. In this respect, the CloudOps self-healing approach, including automatic data portability techniques, can provide support to the Operators of the system. Similarly, instead of managing the service continuity by having two edge nodes, it could be possible to move computation across the continuum moving from the edge to the cloud. In both cases, using stateful objects, the self-healing mechanisms will ensure that in the event of a fault, operations continue without interruption by providing data on the movement of the crowd.

4.3. Business Critical Applications: Initial Results

The proposed CloudOps workflow and framework have been partially validated through an implementation of an MVP (Minimum Viable Product) in an e-health scenario. In this case, an initial version of the proposed CloudOps framework was used to deploy and operate a multi-cloud e-health .NET application. The solution implemented included all the described functional components from Figure 3, with limited features (i.e., only cloud infrastructural elements were considered, a limited set of NFRs were monitored and stateless components were not supported). The different components from Figure 3 have been inserted into the actual framework and the main sub-components for each of the functional blocks are represented in Figure 7. Figure 7 depicts the architecture of the initial MVP of the proposed CloudOps framework validated in the e-health scenario. It also shows how the additional activities proposed in the CloudOps concept in Figure 2 (i.e., pre-deployment and self-healing) are supported by specific tools in this test case. To this end, the **pre-deployment phase** is materialized in two new tool supported activities:

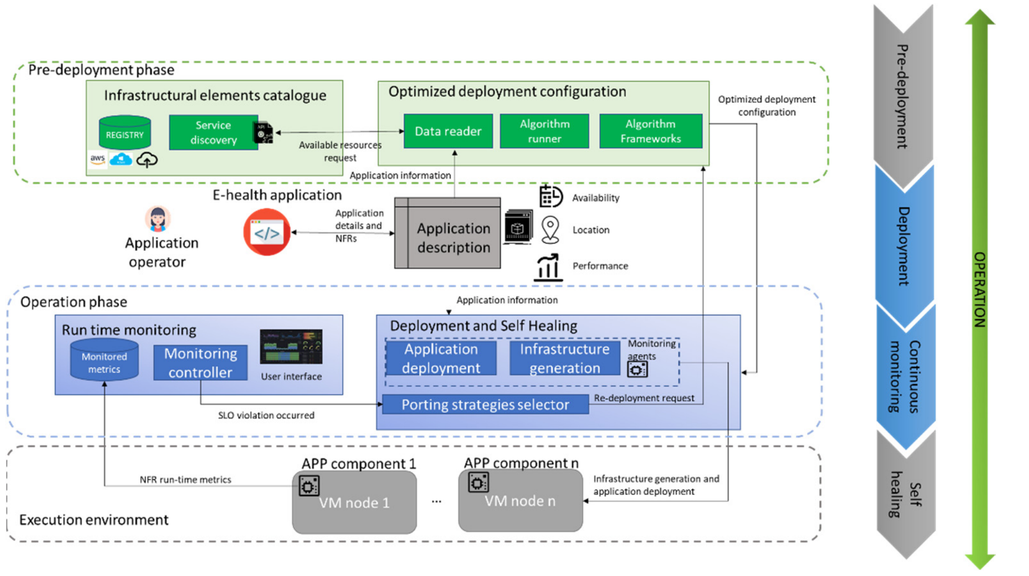
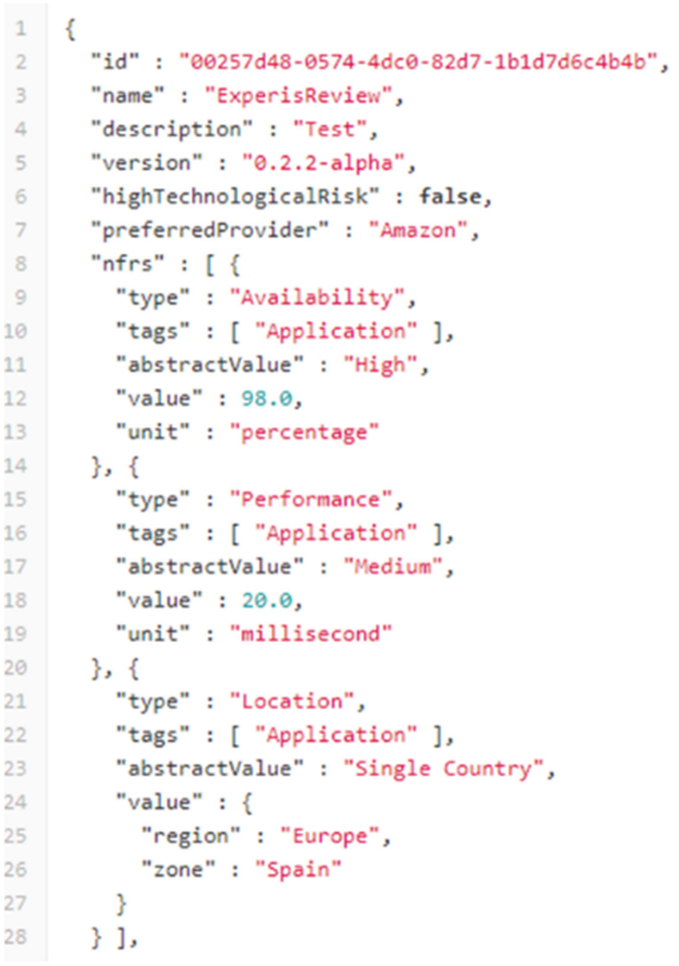


Figure 7. Architecture of the MVP of the proposed CloudOps framework validated in an e-health scenario.

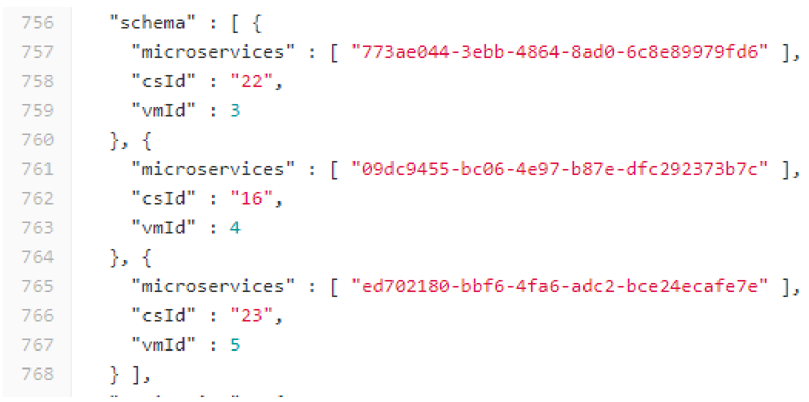
1. Available cloud resources discovery through the "Infrastructural elements catalogue" (Resource discovery functional block in Figure 3): This catalogue provides a unique registry for all the infrastructural elements available along with their relevant characteristics. In this concrete test case, cloud services from Amazon, Azure and Arsys were included. The operators of the tested e-health application were able to discover and filter the different resources to their own convenience. As reported by the operators, this supposed a great advantage and relevant time savings as they did not need to dig into the web pages of each of the cloud providers looking for the best combination of cloud resources. The catalogue has been developed with the JHipster [77] frontend and backend generator;
2. "Optimized deployment configuration" provisioning (the Optimized Deployment Configuration building block in Figure 3): This component gathers the information from the application (available in the application description file shown in Figures 8 and 9) and collects the required information from the Infrastructural elements' catalogue (through a REST API). From this, it creates the best combination of

available infrastructural elements fulfilling the application components types' needs (i.e., computing capabilities, storage capabilities, etc.) and the SLOs (i.e., location, performance and availability). It uses the NSGA-II Genetic Algorithm [78] implemented in MOEA Framework [79], an open source Java library. These components provide the application operator team with a tool to perform several tests before the application is deployed. Furthermore, it eases the selection of the services against the selected NFRs, in a multi-objective optimization problem.



```
{
  "id" : "00257d48-0574-4dc0-82d7-1b1d7d6c4b4b",
  "name" : "ExperisReview",
  "description" : "Test",
  "version" : "0.2.2-alpha",
  "highTechnologicalRisk" : false,
  "preferredProvider" : "Amazon",
  "nfrs" : [ {
    "type" : "Availability",
    "tags" : [ "Application" ],
    "abstractValue" : "High",
    "value" : 98.0,
    "unit" : "percentage"
  }, {
    "type" : "Performance",
    "tags" : [ "Application" ],
    "abstractValue" : "Medium",
    "value" : 20.0,
    "unit" : "millisecond"
  }, {
    "type" : "Location",
    "tags" : [ "Application" ],
    "abstractValue" : "Single Country",
    "value" : {
      "region" : "Europe",
      "zone" : "Spain"
    }
  } ],
```

Figure 8. Snippet of the application description JSON file, where the application's NFRs are described.



```
  "schema" : [ {
    "microservices" : [ "773ae044-3ebb-4864-8ad0-6c8e89979fd6" ],
    "csId" : "22",
    "vmId" : 3
  }, {
    "microservices" : [ "09dc9455-bc06-4e97-b87e-dfc292373b7c" ],
    "csId" : "16",
    "vmId" : 4
  }, {
    "microservices" : [ "ed702180-bbf6-4fa6-adc2-bce24ecafe7e" ],
    "csId" : "23",
    "vmId" : 5
  } ],
```

Figure 9. Snippet of the application description JSON file, where the best combination of infrastructural elements are suggested in through the "schema" element.

As explained, the microservices-based e-health application's main characteristics were maintained in the "Application Description" JSON file which was used by the different components to gather and exchange relevant information. Consequently, the infrastructural requirements and application components' description (Figure 3) are collected in the application description file as presented in Figure 7.
The **operation phase** is also enriched with a set of new tool-supported activities specially for **the self-healing**;

3. Automatic deployment of the application occurs through the "Infrastructure generation" and the "Application Deployment" sub-modules (Deployment and Self-Healing in Figure 3). They create the needed infrastructural resources and deploy the application through Terraform and Ansible templates. Besides the deployment of the application, they also deploy a set of monitoring agents (Telegraf-based [80]) in the infrastructure to gather the corresponding runtime metrics aligned with the selected NFRs;
4. "Automatic run-time monitoring" of all the infrastructural elements (Figure 3): It gathers in a single place all the monitored metrics from the different providers' resources. It comprises a full monitoring stack, based on Telegraf agents, the InfluxDB [81] time-series database as the persistence layer and Grafana dashboards [82] for the graphical interface. In this component, the "Monitoring controller" is also in charge of detecting SLO violations in the infrastructural elements and informing both the Operator of the application (through an email) and the "Deployment and Self-Healing" module (through a REST API) about these violations;
5. Finally, the "Porting strategies selector" sub-module (as part of the Self-learning building block in Figure 3) receives the requests from the "Monitoring controller" once a violation occurs and re-deploys the application, if needed. If this is the case, the "Optimized deployment configuration" is called and the cycle starts again with the optimization of the deployment configuration. It is to be noted that the "Infrastructural Elements catalogue" is also updated with the information of the violations that have occurred on the used infrastructural elements. As a result, when the next set of available resources are requested the information of the violations is also considered so that the opinionated selection of infrastructural elements can be provided by the "Optimized deployment configuration".

As anticipated, the test case only covered some of the main functionalities for the CloudOps framework. Nevertheless, it showed positive impacts and improvements of the Operationalization of the referenced application both qualitative, presented in the description of the Figure 7 components, and quantitative, presented in Table 3. Table 3 shows the quantitative improvements in the form of comparative costs (measured in effort, PM-Person Month and PH-Person Hour) between using the CloudOps framework and not using it. These figures have been reported by the operators of the e-health application. The different phases of the CloudOps workflow were implemented by the operators manually when using the CloudOps framework.

Assuming a cost of 40 euros/h (calculated by the e-health application operators, based on the costs of a typical project in their company) a total of 7280€ euros was saved. If we assume four health projects of a similar scale per year, then we anticipate a saving of approximately 29,000 Euros, or around 2400 Euros per month (approx. $\frac{1}{2}$ a Person Month.)

The operators of the e-health application also reported qualitative improvements. The most relevant ones were as follows:

1. The **pre-deployment phase** and the related tools ("Infrastructural elements catalogue", "Optimized deployment configuration") provide efficiency for evaluating a user's potential NFR options for each of the service classifications and provide an instant response to illustrate matching based on the user's specified values of each of their NFRs to help evaluate the most effective options when selecting a service solution. This efficiency exists, as without this tool the user would have to search each

CSP individually to identify matches and reach a decision about the best solution on the information that was discovered manually;

2. The **operation and self-healing** phase enhances the fulfilment of the application's Service Level Agreement (SLA). As the SLA, and the underlying SLAs of the running cloud services, can continuously be monitored and the violations are automatically detected, so the application's SLA is hardly broken

Based on these initial results we can envision greater impacts in future implementations of the proposed CloudOps framework, in the two proposed test cases in Sections 4.1 and 4.2. Especially when we will incorporate into the validation process new infrastructural elements (such as edge or IoT agents) and the complete functionality described.

Table 3. Estimation of the savings provided by the Cloud Ops framework in the e-health scenario.

CloudOps Phase	Effort Manual Implementation	Effort CloudOps Framework	Savings
Resources discovery and optimization	0.7 PM	0 PM	0.7 PM
Deployment	0.1 PM	025 PH	0.1 PM
Monitoring (annual projected)	0.25 PM	0.25 PH	0.25 PM
Re-adaptation and self-healing	0.25 PM	0.01 PM	0.24 PM
TOTAL	**1.3 PM**	**0.01 PM**	**1.3 PM**

5. Conclusions

Ecosystems for digitalization require the development of applications (SaaS) often running on heterogeneous resources encompassing infrastructure elements (e.g., sensors), cloud and edge resources and network characteristics. This article has presented research that studies the operationalization of applications in the Cloud Continuum and the corresponding requirements and needs. We have outlined the main challenges through an analysis of the related works, which has served as the basis for the definition of the CloudOps workflow. We have introduced the CloudOps concept and the supporting reference framework which customizes the traditional Ops cycle with activities and supporting techniques addressing the specific needs and challenges of the applications in the Cloud Continuum.

The novel concept of the CloudOps workflow and reference framework are key findings of the work. They focus on facilitating and speeding up deployment and operation of such ecosystems when they exploit heterogeneous cloud resources. To this end, our proposal provides Ops teams with applications under the Cloud Continuum paradigm with a CloudOps reference framework that allows them to deploy in an optimized configuration environment while simplifying its operation. We also incorporated the concept of self-healing and data portability in the Cloud Continuum with special focus on the portability of stateful components, which is an unsolved issue up to date. We have also explained the main benefits of the solution for the target users and especially for the operators of critical infrastructures-based systems. We included the initial results captured with a first implementation of the MVP (minimum viable product) of the presented framework. This will be extended, including a complete structural and behavioral architectural description of the CloudOps' components described in the paper, and also with the development of the corresponding complete POCs (proof of concepts) that will serve to exhaustively validate the presented approach.

The next steps will also include the full validation and verification of the applicability of this novel approach in relevant industrial use cases. As we have already advanced, envisioned candidates for testing the solution are mainly applications to manage critical infrastructures (e.g., safety and security, emergency management).

Author Contributions: Conceptualization, investigation, methodology, J.A. and L.O.-E.; writing—original draft, J.A. and L.O.-E.; writing—review and editing, J.A., L.O.-E. and M.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the European project PIACERE (Horizon 2020 Research and Innovation Programme, under grant agreement No. 101000162).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
2. Bittencourt, L.; Immich, R.; Sakellariou, R.; Fonseca, N.; Madeira, E.; Curado, M.; Villas, L.; DaSilva, L.; Lee, C.; Rana, O. The Internet of Things, Fog and Cloud continuum: Integration and challenges. *Internet Things* **2018**, *3–4*, 134–155. [CrossRef]
3. Bittencourt, L.F.; Diaz-Montes, J.; Buyya, R.; Rana, O.F.; Parashar, M. Mobility-Aware Application Scheduling in Fog Computing. *IEEE Cloud Comput.* **2017**, *4*, 26–35. [CrossRef]
4. Lee, C.A. Cloud Federation Management and Beyond: Requirements, Relevant Standards, and Gaps. *IEEE Cloud Comput.* **2016**, *3*, 42–49. [CrossRef]
5. Ometov, A.; Molua, O.L.; Komarov, M.; Nurmi, J. A Survey of Security in Cloud, Edge, and Fog Computing. *Sensors* **2022**, *22*, 927. [CrossRef]
6. Download Infrastructure as Code: Fueling the Fire for Faster Application Delivery—Forrester TLP from Official Microsoft Download Center. Available online: https://www.microsoft.com/en-us/download/details.aspx?id=46403 (accessed on 11 February 2022).
7. Infrastructure as Code (IaC): The Complete Beginner's Guide—BMC Software Blogs. Available online: https://www.bmc.com/blogs/infrastructure-as-code/ (accessed on 11 February 2022).
8. Survey: DevSecOps Progress Remains Elusive—DevOps.com. Available online: https://devops.com/survey-devsecops-progress-remains-elusive/ (accessed on 11 February 2022).
9. 9 Top Software Development Trends for 2021. Available online: https://www.standardfirms.com/top-software-development-trends/ (accessed on 11 February 2022).
10. Zhou, X.; Peng, X.; Xie, T.; Sun, J.; Ji, C.; Li, W.; Ding, D. Fault Analysis and Debugging of Microservice Systems: Industrial Survey, Benchmark System, and Empirical Study. *IEEE Trans. Softw. Eng.* **2018**, *47*, 243–260. [CrossRef]
11. Shacklett, M. How to Implement Edge Computing—TechRepublic. Available online: https://www.techrepublic.com/article/how-to-implement-edge-computing/ (accessed on 11 February 2022).
12. Cloud Edge Computing: Beyond the Data Center—OpenStack is Open Source Software for Creating Private And Public Clouds. Available online: https://www.openstack.org/use-cases/edge-computing/cloud-edge-computing-beyond-the-data-center/ (accessed on 11 February 2022).
13. Varghese, B.; Akgun, O.; Miguel, I.; Thai, L.; Barker, A. Cloud Benchmarking for Maximising Performance of Scientific Applications. *IEEE Trans. Cloud Comput.* **2016**, *7*, 170–182. [CrossRef]
14. Kousiouris, G.; Aisopos, F.; Psychas, A.; Varvarigou, T.; Domaschka, J.; Baur, D.; Griesinger, F.; Nikolov, V.; Lyberopoulos, G.; Theodoropoulou, E.; et al. A Toolkit Based Architecture for Optimizing Cloud Management, Performance Evaluation and Provider Selection Processes. In Proceedings of the 2017 International Conference on High Performance Computing & Simulation (HPCS), Genoa, Italy, 17–21 July 2017; pp. 224–232.
15. Palit, T.; Shen, Y.; Ferdman, M. Demystifying cloud benchmarking. In Proceedings of the 2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Uppsala, Sweden, 17–19 April 2016; pp. 122–132.
16. CloudSuite—A Benchmark Suite for Cloud Services. Available online: https://www.cloudsuite.ch/ (accessed on 16 February 2022).
17. Round 20 Results—TechEmpower Framework Benchmarks. Available online: https://www.techempower.com/benchmarks/ (accessed on 16 February 2022).
18. Home—Filebench/Filebench Wiki—GitHub. Available online: https://github.com/filebench/filebench/wiki (accessed on 16 February 2022).
19. Hong, C.-H.; Varghese, B. Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms. *ACM Comput. Surv.* **2020**, *52*, 97. [CrossRef]
20. Tocze, K.; Schmitt, N.; Brandic, I.; Aral, A.; Nadjm-Tehrani, S. Towards Edge Benchmarking: A Methodology for Characterizing Edge Workloads. In Proceedings of the 2019 IEEE 4th International Workshops on Foundations and Applications of Self* Systems (FAS*W), Umea, Sweden, 16–20 June 2019; pp. 70–71.

21. Toczé, K.; Nadjm-Tehrani, S. A Taxonomy for Management and Optimization of Multiple Resources in Edge Computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 7476201. [CrossRef]
22. Das, A.; Patterson, S.; Wittie, M. EdgeBench: Benchmarking Edge Computing Platforms. In Proceedings of the 2018 IEEE/ACM International Conference on Utility and Cloud Computing Companion (UCC Companion), Zurich, Switzerland, 17–20 December 2018; pp. 175–180.
23. Zheng, C. Bench 2021: 2021 BenchCouncil International Symposium on Benchmarking, Measuring and Optimizing. Available online: http://wikicfp.com/cfp/servlet/event.showcfp?eventid=132576&copyownerid=168997 (accessed on 16 February 2022).
24. Alonso, J.; Orue-Echevarria Arrieta, L.; Escalante, M.; Benguria, G.; Echevarria, G. Federated Cloud Service Broker (FCSB): An Advanced Cloud Service Intermediator for Public Administrations. In Proceedings of the 7th International Conference on Cloud Computing and Services Science, Porto, Portugal, 24–26 April 2017; p. 391.
25. GitHub—ale93p/namb: Not Only a Micro-Benchmark. Available online: https://github.com/ale93p/namb (accessed on 16 February 2022).
26. Pagliari, A.; Huet, F.; Urvoy-Keller, G. Towards a High-Level Description for Generating Stream Processing Benchmark Applications. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019; pp. 3711–3716.
27. Haq, O.; Raja, M.; Dogar, F.R. Measuring and Improving the Reliability of Wide-Area Cloud Paths. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2017; pp. 253–262.
28. Yeganeh, B.; Durairajan, R.; Rejaie, R.; Willinger, W. A First Comparative Characterization of Multi-cloud Connectivity in Today's Internet. In *Passive and Active Measurement*; Sperotto, A., Dainotti, A., Stiller, B., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2020; Volume 12048, pp. 193–210. ISBN 978-3-030-44080-0.
29. Jin, Y.; Renganathan, S.; Ananthanarayanan, G.; Jiang, J.; Padmanabhan, V.N.; Schroder, M.; Calder, M.; Krishnamurthy, A. Zooming in on wide-area latencies to a global cloud provider. In Proceedings of the ACM Special Interest Group on Data Communication, Beijing, China, 19–23 August 2019; pp. 104–116.
30. Villari, M.; Fazio, M.; Dustdar, S.; Rana, O.; Ranjan, R. Osmotic Computing: A New Paradigm for Edge/Cloud Integration. *IEEE Cloud Comput.* **2016**, *3*, 76–83. [CrossRef]
31. Varghese, B.; Buyya, R. Next generation cloud computing: New trends and research directions. *Future Gener. Comput. Syst.* **2018**, *79*, 849–861. [CrossRef]
32. Jourdan, L.; Basseur, M.; Talbi, E.-G. Hybridizing exact methods and metaheuristics: A taxonomy. *Eur. J. Oper. Res.* **2009**, *199*, 620–629. [CrossRef]
33. Müller-Merbach, H. Heuristics and their design: A survey. *Eur. J. Oper. Res.* **1981**, *8*, 1–23. [CrossRef]
34. Gendreau, M.; Potvin, J.-Y. (Eds.) *Handbook of Metaheuristics*; International Series in Operations Research & Management Science; Springer: Boston, MA, USA, 2010; Volume 146, ISBN 978-1-4419-1663-1.
35. Osaba, E.; Carballedo, R.; Diaz, F.; Onieva, E.; Masegosa, A.D.; Perallos, A. Good practice proposal for the implementation, presentation, and comparison of metaheuristics for solving routing problems. *Neurocomputing* **2018**, *271*, 2–8. [CrossRef]
36. Arostegi, M.; Torre-Bastida, A.; Bilbao, M.N.; Del Ser, J. A heuristic approach to the multicriteria design of IaaS cloud infrastructures for Big Data applications. *Expert Syst.* **2018**, *35*, e12259. [CrossRef]
37. Alonso, J.; Stefanidis, K.; Orue-Echevarria, L.; Blasi, L.; Walker, M.; Escalante, M.; López, M.J.; Dutkowski, S. DECIDE: An Extended DevOps Framework for Multi-cloud Applications. In Proceedings of the 2019 3rd International Conference on Cloud and Big Data Computing, Oxford, UK, 28–30 August 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 43–48.
38. Terraform by HashiCorp. Available online: https://www.terraform.io/ (accessed on 17 February 2022).
39. Heat—OpenStack. Available online: https://wiki.openstack.org/wiki/Heat (accessed on 17 February 2022).
40. Template Structure and Syntax—Azure Resource Manager—Microsoft Docs. Available online: https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/syntax (accessed on 17 February 2022).
41. Google Cloud Deployment Manager Documentation—Cloud Deployment Manager Documentation. Available online: https://cloud.google.com/deployment-manager/docs/ (accessed on 17 February 2022).
42. Enterprise Kubernetes Management—Rancher. Available online: https://rancher.com/ (accessed on 17 February 2022).
43. Spinnaker. Available online: https://spinnaker.io/ (accessed on 17 February 2022).
44. ALIEN 4 Cloud. Available online: https://alien4cloud.github.io/ (accessed on 17 February 2022).
45. Cloudify Orchestration Platform—Multi Cloud, Cloud Native & Edge. Available online: https://cloudify.co/ (accessed on 17 February 2022).
46. Kubernetes. Available online: https://kubernetes.io/ (accessed on 17 February 2022).
47. Haja, D.; Szalay, M.; Sonkoly, B.; Pongracz, G.; Toka, L. Sharpening Kubernetes for the Edge. In Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos on—SIGCOMM Posters and Demos' 19, Beijing, China, 19–23 August 2019; ACM Press: New York, NY, USA; pp. 136–137.
48. Masip-Bruin, X.; Marín-Tordera, E.; Sánchez-López, S.; Garcia, J.; Jukan, A.; Juan Ferrer, A.; Queralt, A.; Salis, A.; Bartoli, A.; Cankar, M.; et al. Managing the Cloud Continuum: Lessons Learnt from a Real Fog-to-Cloud Deployment. *Sensors* **2021**, *21*, 2974. [CrossRef]

49. Gonzalez, N.M.; de Carvalho, T.C.M.B.; Miers, C.C. Cloud resource management: Towards efficient execution of large-scale scientific applications and workflows on complex infrastructures. *J. Cloud Comput. Adv. Syst. Appl.* **2017**, *6*, 13. [CrossRef]
50. Ismail, A.; Cardellini, V. Towards Self-adaptation Planning for Complex Service-Based Systems. In *Service-Oriented Computing—ICSOC 2013 Workshops*; Lomuscio, A.R., Nepal, S., Patrizi, F., Benatallah, B., Brandić, I., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2014; Volume 8377, pp. 432–444. ISBN 978-3-319-06858-9.
51. Push vs. Pull Monitoring Configs. How Do Monitoring Agents Know—by Steve Mushero—Medium. Available online: https://steve-mushero.medium.com/push-vs-pull-configs-for-monitoring-c541eaf9e927 (accessed on 11 February 2022).
52. Alonso, J.; Orue-Echevarria, L.; Escalante, M. Contribution to the uptake of Cloud Computing solutions: Design of a cloud services intermediator to foster an ecosystem of trusted, interoperable and legal compliant cloud services. Application to Multi-Cloud Aware Software. In Proceedings of the 15th International Conference on Web Information Systems and Technologies (WEBIST), Vienna, Austria, 18–20 September 2019.
53. Fowler, M. Infrastructure as Code. Available online: https://martinfowler.com/bliki/InfrastructureAsCode.html (accessed on 29 March 2021).
54. Natu, M.; Ghosh, R.K.; Shyamsundar, R.K.; Ranjan, R. Holistic Performance Monitoring of Hybrid Clouds: Complexities and Future Directions. *IEEE Cloud Comput.* **2016**, *3*, 72–81. [CrossRef]
55. Dustdar, S.; Schreiner, W. A survey on web services composition. *Int. J. Web Grid Serv.* **2005**, *1*, 1. [CrossRef]
56. Leitner, P.; Michlmayr, A.; Rosenberg, F.; Dustdar, S. Monitoring, Prediction and Prevention of SLA Violations in Composite Services. In Proceedings of the 2010 IEEE International Conference on Web Services, Miami, FL, USA, 5–10 July 2010; pp. 369–376.
57. Ivanović, D.; Carro, M.; Hermenegildo, M. Constraint-Based Runtime Prediction of SLA Violations in Service Orchestrations. In *Service-Oriented Computing*; Kappel, G., Maamar, Z., Motahari-Nezhad, H.R., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 7084, pp. 62–76. ISBN 978-3-642-25534-2.
58. Leitner, P.; Wetzstein, B.; Rosenberg, F.; Michlmayr, A.; Dustdar, S.; Leymann, F. Runtime Prediction of Service Level Agreement Violations for Composite Services. In *Service-Oriented Computing—ICSOC 2007*; Krämer, B.J., Lin, K.-J., Narasimhan, P., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 4749, pp. 176–186. ISBN 978-3-540-74973-8.
59. De Francisci Morales, G.; Bifet, A.; Khan, L.; Gama, J.; Fan, W. IoT Big Data Stream Mining. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 2119–2120.
60. Barros, R.S.M.; Santos, S.G.T.C. A large-scale comparison of concept drift detectors. *Inf. Sci.* **2018**, *451–452*, 348–370. [CrossRef]
61. Lu, J.; Liu, A.; Song, Y.; Zhang, G. Data-driven decision support under concept drift in streamed big data. *Complex Intell. Syst.* **2020**, *6*, 157–163. [CrossRef]
62. Rahman, A.; Elder, S.; Shezan, F.H.; Frost, V.; Stallings, J.; Williams, L. Categorizing Defects in Infrastructure as Code. *arXiv* **2018**, arXiv:1809.07937.
63. Xu, X.; Zhu, L.; Weber, I.; Bass, L.; Sun, D. POD-Diagnosis: Error Diagnosis of Sporadic Operations on Cloud Applications. In Proceedings of the 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, Atlanta, GA, USA, 23–26 June 2014; pp. 252–263.
64. Angarita, R.; Manouvrier, M.; Rukoz, M. An Agent Architecture to Enable Self-healing and Context-aware Web of Things Applications. In Proceedings of the International Conference on Internet of Things and Big Data, Rome, Italy, 23–25 April 2016; Science and and Technology Publications: Setúbal, Portugal, 2016; pp. 82–87.
65. Adel Serhani, M.; El-Kassabi, H.T.; Shuaib, K.; Navaz, A.N.; Benatallah, B.; Beheshti, A. Self-adapting cloud services orchestration for fulfilling intensive sensory data-driven IoT workflows. *Future Gener. Comput. Syst.* **2020**, *108*, 583–597. [CrossRef]
66. Gill, S.S.; Chana, I.; Singh, M.; Buyya, R. RADAR: Self-configuring and self-healing in resource management for enhancing quality of cloud services. *Concurr. Comput. Pract. Exp.* **2019**, *31*, e4834. [CrossRef]
67. Gao, H.; Huang, W.; Yang, X.; Duan, Y.; Yin, Y. Toward service selection for workflow reconfiguration:An interface-based computing solution. *Future Gener. Comput. Syst.* **2018**, *87*, 298–311. [CrossRef]
68. Toffetti, G.; Brunner, S.; Blöchlinger, M.; Spillner, J.; Bohnert, T.M. Self-managing cloud-native applications: Design, implementation, and experience. *Future Gener. Comput. Syst.* **2017**, *72*, 165–179. [CrossRef]
69. El-Kassabi, H.T.; Adel Serhani, M.; Dssouli, R.; Navaz, A.N. Trust enforcement through self-adapting cloud workflow orchestration. *Future Gener. Comput. Syst.* **2019**, *97*, 462–481. [CrossRef]
70. Achieving Application Portability with Kubernetes & Cloud Native. Available online: https://kublr.com/blog/application-portability-with-kubernetes-and-cloud-native/ (accessed on 16 February 2022).
71. Data Gravity—In the Clouds—Data Gravitas. Available online: https://datagravitas.com/2010/12/07/data-gravity-in-the-clouds/ (accessed on 16 February 2022).
72. Stateful Services—The Black Sheep of the Container World | D2iQ. Available online: https://d2iq.com/blog/stateful-services-black-sheep-container-world (accessed on 16 February 2022).
73. Balouek-Thomert, D.; Renart, E.G.; Zamani, A.R.; Simonet, A.; Parashar, M. Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows. *Int. J. High Perform. Comput. Appl.* **2019**, *33*, 1159–1174. [CrossRef]
74. Baresi, L.; Mendonça, D.F.; Garriga, M.; Guinea, S.; Quattrocchi, G. A Unified Model for the Mobile-Edge-Cloud Continuum. *ACM Trans. Internet Technol.* **2019**, *19*, 29. [CrossRef]

75. European Commission. Work Programme 2018–2020. Available online: https://ec.europa.eu/research/participants/data/ref/h2020/wp/2018-2020/main/h2020-wp1820-leit-ict_en.pdf (accessed on 12 February 2022).
76. Shi, W.; Cao, J.; Zhang, Q.; Li, Y.; Xu, L. Edge Computing: Vision and Challenges. *IEEE Internet Things J.* **2016**, *3*, 637–646. [CrossRef]
77. JHipster—Full Stack Platform for the Modern Developer. Available online: https://www.jhipster.tech/ (accessed on 11 March 2022).
78. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. EVol. Comput.* **2002**, *6*, 182–197. [CrossRef]
79. MOEA Framework—A Java Library for Multiobjective Evolutionary Algorithms. Available online: http://moeaframework.org/ (accessed on 11 April 2022).
80. Telegraf Open Source Server Agent—InfluxDB. Available online: https://www.influxdata.com/time-series-platform/telegraf/ (accessed on 11 March 2022).
81. InfluxDB: Open Source Time Series Database. Available online: https://www.influxdata.com/ (accessed on 11 April 2022).
82. Grafana Cloud—Grafana Labs. Available online: https://go2.grafana.com/grafana-cloud.html?src=ggl-s&mdm=cpc&camp=b-grafana-exac-emea&cnt=118483912276&trm=grafana&device=c&gclid=CjwKCAjw6dmSBhBkEiwA_W-EoMb226-I3urBt952aGsX0ocblLsqRjX8W7hia68oxUNGmdBgIdVgBBoCK8cQAvD_BwE (accessed on 13 April 2022).