ISSN 0361-7688, Programming and Computer Software, 2022, Vol. 48, No. 8, pp. 632–645. © Pleiades Publishing, Ltd., 2022.

A Microservice Deployment Guide

V. M. Niño-Martínez[a],*, J. O. Ocharán-Hernández[a],, X. Limón[a],***, and J. C. Pérez-Arriaga[a],****

[a]*School of Statistics and Informatics, Universidad Veracruzana, Av. Xalapa, Obrero Campesino, Xalapa-Enríquez, Ver., 91020 Mexico*

**e-mail: ninomtz.victor@gmail.com*

***e-mail: jocharan@uv.mx*

****e-mail: hlimon@uv.mx*

*****e-mail: juaperez@uv.mx*

Received June 14, 2022; revised July 12, 2022; accepted August 2, 2022

Abstract—Modern software development requires agile methods to deploy and scale increasingly demanded distributed systems. Practitioners have adopted the microservices architecture to cope with the challenges posed by modern software demands. However, the adoption and deployment of this architecture also creates technical and organizational challenges, potentially slowing down the development and operation teams, which require more time and effort to implement a quality deployment process that allows them to constantly release new features to production. The adoption of a DevOps culture, along with its practices and tools, alleviates some of these new challenges. In this paper we propose a guide for the deployment of systems with a microservices architecture, considering the practices of a DevOps culture, providing practitioners with a base path to start implementing the necessary platform for this architecture. We conducted this work following the Design Science Research Methodology for Information Systems (DSRM). In this way, we identified the problem, and also defined the solution objectives through the execution of a Systematic Literature Mapping and a Gray Literature Review, having as a result the proposed guide. This work can be summarized as follows: (I) Identification of practices and technologies that support the deployment of microservices. (II) Identification of recommendations, challenges, and best practices for the deployment process. (III) Modeling of the microservices deployment process using SPEM. (IV) Integration of the knowledge in a guide to deploying microservices by adopting DevOps practices.

DOI: 10.1134/S0361768822080151

1. INTRODUCTION

In the 1990s, the popularization of the World Wide Web (WWW) and the subsequent dot-com gold rush introduced the world to software as a service (SaaS), leading to entire industries built on this SaaS model. This motivated the development of applications that required more resources, making them more complex to develop, maintain and deploy. Nowadays, enterprise systems need to transfer information with other systems, internal or external to the organization, even at a global scale. Companies such as Amazon, Netflix [1], Uber [2], LinkedIn [3] and, SoundCloud [4], among others, found the need to migrate to a software architecture that allows them to undertake the complexity and constant need of evolution of their systems. To this end, they chose to adopt a Microservices Architecture (MSA). Not only have these large companies migrated to an MSA, but small and medium-sized companies have also done so, all of them seeking the benefits that this architecture brings, such as scalability, heterogeneity, and extensibility, among others.

A MSA is an approach to developing a distributed system as a set of small services. Each of these services runs in its process and communicates using lightweight mechanisms, like an HTTP resource API [5]. One of the characteristics that make this architecture different is the granularity of the services, which must be small and highly cohesive. Microservices adopt the single responsibility principle approach, which states "Gather together the things that change for the same reasons, separate those things that change for different reasons" [6], focusing the service boundaries on the business boundaries, in this way, preventing services from growing too large as well as the difficulties that this may introduce. The key benefits that microservices architecture offers over conventional architectural patterns are: the heterogeneity of technologies, fault tolerance, agile deployment, scalability, alignment with organizational structure, replaceability, and agile development of business functionality [7], [8].

Software deployment is a stage of the software development life cycle in which a system is put into operation and transition issues are resolved [9].

Deployment combines two closely related concepts, the first one is the deployment process, which consists of a series of steps that must be executed by the developers or those in charge of managing the system infrastructure to put the software into a production environment, and the second is the deployment architecture, which defines the structure of the software execution environment [10]. An application is only useful when deployed to users. Mature deployment practices are crucial to building reliable and stable microservices.

Unlike a monolithic system, optimized for a single-use case, microservices deployment practices need to scale to multiple services; it is possible to have tens or hundreds of microservices, written in different programming languages and frameworks. Each microservice is a small application with a specific process and architecture, which operators and developers need to deploy in production. If operators and developers are not able to quickly and reliably deploy microservices, then the added development speed gained from microservices would be useless. Therefore, a mature deployment process and automated deployments are essential for developing microservices at scale.

When migrating from a monolithic approach to deploy microservices, the main challenges are the familiarization with the variety of technologies and tools, the automation of the process, and the implementation of a pipeline to continuously deploy [11]. In addition, among the most important challenges related to the deployment of this type of architecture are: 1) maintaining stability for a large volume of releases and component changes; 2) avoiding coupling between components, leading to dependencies in the build or release times; 3) managing changes in the service API, as changes could negatively affect the clients; and 4) removing and updating production services [12]. The practices found in DevOps aid to alleviate the mentioned challenges, these practices include: Continuous Integration (CI), Continuous Delivery (CD), Configuration Management (CM), and monitoring, among others. The implementation of these practices generates new challenges regarding: communication and coordination between teams; lack of investment in costs; lack of experience and skills; conflict management; design and code dependencies between components; implementation and release of software to customers [13].

To help developers and people in charge of creating a stable infrastructure to deploy microservices, we decided to elaborate a guide for the deployment of microservices-based systems, considering DevOps culture practices. The goal of the guide is to reduce the effort associated with creating an ecosystem for the microservices architecture. The guide integrates different organizational technical decisions, technologies, and tools successfully used by organizations, as well as the associated DevOps practices. The guide helps all related parties in the process of adopting a microservices architecture.

In order to create the guide, we followed the Design Science Research Methodology (DSRM) methodology [14], consisting of six phases. We have already completed the following phases: identification of practices, technologies, tools, activities, and recommendations for the deployment of microservices, through a previous work [15] consisting of a systematic mapping of the literature and a review of gray literature; classification and grouping of the information found; MSA process adoption modeling; and the selection and integration of related activities according to the adoption process. With these phases covered, it is possible to have a first version of the microservices deployment guide, leaving the demonstration and evaluation as future work.

This paper is organized as follows. Section 2 gives an overview of some studies focused on the deployment of microservices and the adoption of DevOps practices. Section 3 presents the followed method to develop our microservices deployment guide, based on the DSRM methodology [14]. Section 4 describes the proposed deployment guide and its structure. Finally, Section 5 features the conclusion and future work.

2. RELATED WORK

We found in the literature numerous studies about microservices, however, few of them focus on microservice deployment. In the reviewed literature, we found practices adopted by practitioners, technologies, tools experiences, and recommendations for the deployment of microservices.

We found a series of works including case studies [16]– [24], regarding the implementation of a microservices architecture in conjunction with practices of DevOps, such as CI/CD Continuous Integration and Delivery. As a result of these studies, we obtained the tools and technologies used for microservices deployment, as well as their rationale. We also identified studies concerning the migration to MSA applying DevOps practices [25]– [29], these studies include mistakes made by organizations, tools, adoption processes, and recommendations by practitioners. In addition to experiences and case studies, we identified some methods, guidelines, or classifications for continuous deployment, monitoring, and DevOps tactics at an architectural level [30]– [33]. Two aspects that stand out in these studies are the lack of depth in the implementation process to adopt DevOps practices, and the lack of details for tasks related to each activity.

In [34] authors identify the different principles and patterns of a microservices architectural style, mapping the existing tools and techniques in the context of DevOps. This study focuses on the advantages and disadvantages of the microservices architecture and its

patterns, however, the study only partially covers the implementation of a DevOps workflow. On the other hand, in [13], through a systematic review focused on continuous practices, the authors made a classification of tools, identified challenges and some other related practices, showing some gaps for further research.

Bolscher and Daneva [35] performed a systematic mapping of the literature, to provide a summary of the issues and requirements for the design of an architecture that supports continuous integration and DevOps practices. This study does not mention in-depth information on DevOps practices, nor mentions an implementation of these practices in a real environment. However, the study provides an explanation of the importance of implementing a DevOps culture in the adoption of a microservices architecture. Finally, in the context of microservices and DevOps, [36] presents a classification of the problems reported by practitioners, including some related solutions, giving also a summary of the tools and challenges.

In summary, each of the resources and papers found present valuable information regarding the process of deploying a microservices architecture, however, we found few papers that made an intersection of knowledge on how to implement DevOps practices in the context of a microservices architecture. The few studies that did so, only focus on a single topic, lacking a broader scope. For this reason, the proposed guide develops and compiles all the information identified, seeking to deepen the architectural and deployment topics related to achieve a mature process, supporting the needs of distributed systems and a microservices architecture in particular.

3. RESEARCH METHOD

We followed the Design Science Research Methodology (DSRM) [14], establishing the recognition and legitimization of aims, processes, and investigation outputs, and helping researchers to present their work according to a common framework. The methodology incorporates principles, practices, and procedures required to carry out such research, meeting three objectives: consistency with prior literature, providing a nominal process model for doing Design Science (DS) research, and providing a mental model for presenting and evaluating DS research. Several studies have used this methodology to develop artifacts and validate its process, for example [37], [38]. DSRM includes six steps: problem identification and motivation, the definition of the objectives, design, and development, demonstration, evaluation, and communication. We detail these phases in the following subsections.

3.1. Problem Identification and Motivation

For the identification of the problem related to microservices deployment and its importance, we performed a preliminary literature review. The concepts and topics analyzed were the microservice architecture style; advantages and drawbacks of its use; processes to deploy microservices; aspects that affect the deployment; and DevOps culture and its practices.

One of the main challenges we found, is the familiarization with the variety of technologies and tools, as well as the automation and implementation of a pipeline to deploy continuously [11]. Moreover, the implementation of practices such as Continuous Integration (CI), Continuous Delivery (CD), Configuration Management (CM), and monitoring; bring new chal-

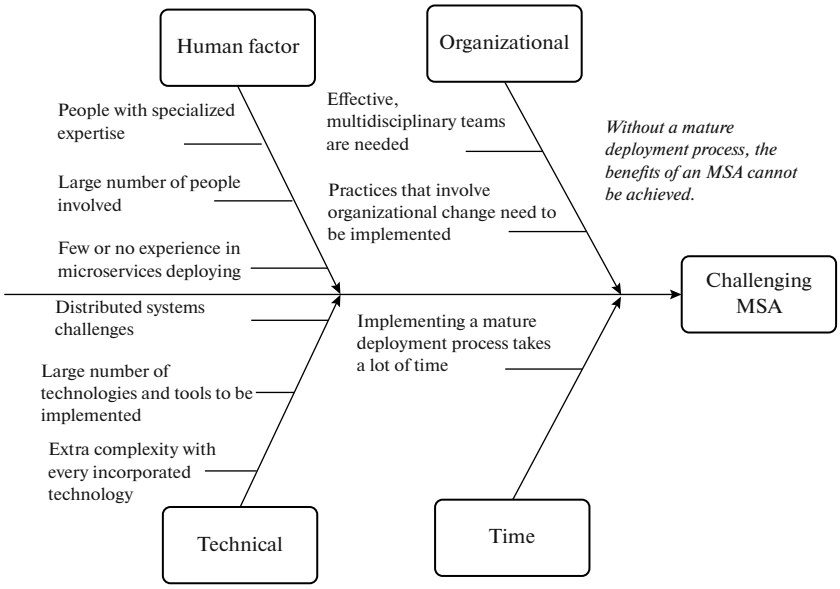

Fig. 1. Cause-effect diagram of a microservice deployment.

lenges, such as communication and coordination between teams; lack of investment in costs; lack of experience and skills; conflict management; design and code dependencies between components; challenges in the implementation and release of software to customers [13], [39].

With our literature review, we developed a cause-effect diagram to reflect the factors that impact the deployment of microservices and convert it into a challenging process. Figure 1 shows the cause-effect diagram.

3.2. Design the Objectives for a Solution

Once we identified the problems, we concluded that a guide to deploy a microservice architecture could help to solve the problems. To know the state of the art, and the possible solutions, we performed a Systematic Mapping Study and A Gray Literature Review, both with the aim to identify practices, processes, technologies, recommendations, and lessons learned and reported by practitioners.

3.2.1. Systematic Mapping Study. We conducted the study following the guidelines of Kitchenham, Budgen, and Brereton [40], the guidelines describe a process to perform the mapping in Software Engineering. The objective of a mapping study is to survey the available knowledge about a topic. It is possible to synthesize information by categorization, identify "clusters" of studies that could form the basis of a fuller review, and also identify "gaps", indicating the need for more primary studies. We executed the mapping study in three main phases: planning, conduction, and results report. Some activities carried out within these phases were: a preliminary literature review; definition of the research questions and search keywords; database selection; inclusion, and exclusion criteria; methods for the data extraction and analysis.

3.2.1.1. Planning. **Research questions.** Derived from the objective of the work, we formulated four research questions (RQ). The questions compiled the state of the art, showing us the techniques and technologies that researches, and practitioners use to deploy microservices, along with the related DevOps practices. The RQs and their motivation are shown in Table 1.

Research process. We performed a preliminary literature review, identifying a series of articles that helped us to define a set of keywords representing the main concepts around the research questions and, some of their related concepts. In the end, we decided to run an automated search for selecting primary studies. We constructed a base string with the search terms identified, refined, and validated using the Recall and Precision techniques. The generated string is the following:

(microservices OR "microservice architecture" OR micro-services OR "architecting microservices") AND (DevOps OR development OR operations OR "continuous integration" OR CI OR "continuous deployment" OR "continuous delivery" OR CD OR migration OR automation OR tools OR adoption OR monitoring OR cloud).

Table 2 shows the selected databases that to conduct the search. We chose these databases because they compile the most significant number of works related to Software Engineering. In addition, in a previous manual review, we found results in the mentioned sources. ACM Digital Library and Elsevier Science Direct repositories have some considerations in their search engines, so we adjusted the search string. Due to the large number of results obtained in ACM, we decided to search only using the title as the indexer. In the Wiley repository, we used the exact string as in Science Direct, because, in the first tests, we observed that it performed better. We present the search string of each database in Table 3. We only covered the last five years in the study, in these years the topics of DevOps and Microservices had more relevance in research articles. We have also observed in these years an increase in popularity of the topics of interest, and therefore it is of relevance for the study. We defined a list of inclusion and exclusion criteria for the studies, presented in Table 4.

Data extraction. We defined a template to extract the necessary information from each article to answer the research questions. Data D1-D10 contains the general information of each study, and data D11-D16 helped to extract qualitative data that answers the research questions. We used a spreadsheet to collect the information.

Data synthesis. For information synthesis, we used the meta-aggregation method [41]. The synthesis brings together the study findings, communicated as themes, metaphors, categories, or concepts; and grouped by further aggregation based on similarity of meaning [41]. This method helped us to identify lessons learned, common mistakes and understand why the literature reports certain technologies a higher number of times. Moreover, with the information classified and grouped, its analysis becomes a more straightforward process.

3.2.1.2. Conduction. We conducted the selection process in three stages, implementing the inclusion and exclusion of the strings in the different sources, and using the filters provided by each of them, the CI-1 and CI-2 criteria corresponding to the years of publication and their language were applied. In addition, in databases such as Science Direct, Springer Link, and ACM Digital Library, we used filters to only include research articles and not book chapters or lecture notes, thus applying the execution criteria CI-3 as well as the exclusion criteria CE-1 and CE-2. In the third stage, we read the full text, and the inclusion and exclusion criteria CI-4, and CE-3 were applied. Figure 2 shows the results after applying the inclusion ad

Table 1. Research Questions and Motivation

Questions	Motivation
RQ-1: What DevOps practices and approaches support the deployment of Microservices?	Identify the practices and approaches used in the DevOps culture and classify the technologies needed for each practice
RQ-2: What technologies do DevOps practices use to deploy Microservices?	It is important to identify the technologies that are used in each DevOps practice, to understand which are the most suitable for a given situation
RQ-3: What challenges does the literature report regarding the adoption of DevOps practices in the deployment of microservices?	Many problems can emerge in the implementation of the practices and this question aims to know what they are and how often they are reported.
RQ-4: What lessons does the literature report for successful microservices deployment?	This question aims to identify the processes, best practices, and recommendations that practitioners implemented in the deployment of their systems and serve as a guide for those in the same situation.

exclusion criteria by stage and database. At the end of the third stage, we obtained a total of 21 primary studies.

Data extraction and analysis. Once we selected the primary studies, we created a spreadsheet in which each column presents the to be extracted data. We performed a complete reading of each article, highlighting the information that answered the research questions and capturing this information in the spreadsheet; we performed this process for each of the primary studies selected. With the extracted information, we proceeded to apply the meta-aggregation method. This method has three main steps: (I) Identify and assemble findings from all included studies; (II) Aggregate well-founded and explicit findings; (III) Synthesis of findings implications. We also captured the findings in a spreadsheet, and with all the findings identified, we iteratively created categories and grouped findings on them .

3.2.1.3. Results. **Meta-aggregation results.** After the application of the method, we extracted classified 43 findings into seven categories. These categories were grouped into three synthesized findings Considerations for Deployment Microservices, Precautions when Deploying Microservices, and Deployment Technologies. Figure 3 shows the associations between categories.

Table 2. Selected Electronic Databases

Database	Link
IEEE Xplore Digital Library	https://ieeexplore.ieee.org
Elsevier Science Direct	https://www.sciencedirect.com
Springer Link	https://link.springer.com
Wiley Online Library	https://onlinelibrary.wiley.com
ACM Digital Library	https://dl.acm.org

Microservices deployment requirements: In this category, we identified requirements that practitioners, from their experience in the area, considered necessary for a successful microservices deployment. We found that architectural support is crucial for the adoption of DevOps practices, as well as having a mature operations team, to allow continuous deployment of numerous microservices. Furthermore, developers need to consider microservices' backward compatibility, and microservices upgrading with minimum effort and application downtime. Flexible and maintainable delivery systems support these needs.

Characteristics of DevOps practices: We grouped in this category, requirements, tips, and lessons learned by practitioners when implementing DevOps practices as well as deployment pipelines. The practitioners agree that pipelines are one of the key parts in the deployment of microservices because without good construction of pipelines, long wait times for releases and builds occur. To prevent it, it is necessary to apply DevOps principles in building CI/CD pipelines, automation is paramount to successful deployment.

Microservices deployment challenges: The findings related to this category are challenges those practitioners identified when adopting a microservices-based architecture. One of the challenges identified is the release of a new version of a microservice, because one or more microservices may depend on it. In addition, when adopting this architecture, there is a great effort in the context of new tools and frameworks. Microservices configuration is essential to achieve the expected results.

Challenges of DevOps practices: In this category, we grouped a set of challenges related to practices and technologies related to DevOps practices. The constant updating of tools and libraries makes development difficult, as well as the lack of tools for specific tasks that developers need to automate. For example, monitoring has several challenges such as lack of commercial options, lack of standardization, and lack of faster learning curves.

Characteristics of building technologies: Technologies are an important part of software deployment and construction; therefore, it is an aspect that practitioners pay particular attention to. In this category, we

Table 3. String Adjusted to each database

Source	String
ACM Digital Library	[microservice* OR "microservice architecture" OR "architecting microservices"] AND [DevOps OR development OR operations OR "continuous integration" OR CI OR "continuous deployment" OR "continuous delivery" OR CD OR migration]
Elsevier Science Direct	(microservice OR "microservice architecture") AND (devops OR development OR operations OR "continuous integration" OR "continuous deployment" OR "continuous delivery" OR migration)
Springer Link	(devops OR development OR operations OR "continuous integration" OR "continuous deployment" OR "continuous delivery" OR migration)
Wiley Online Library	(devops OR development OR operations OR "continuous integration" OR "continuous deployment" OR "continuous delivery" OR migration)

Table 4. Inclusion and Exclusion Criteria

Database	Link
IC-1: Studies published between 2015 and 2020.	EC-1: It is an abstract, workshop, opinion article, presentations, book chapters, or conference notes.
CI-2: Articles written in English	EC-2: The study does not focus on Microservices and DevOps process deployment
IC-3: The title and abstract contain information indicating that the full text could answer at least one research question.	EC-3: The study is an earlier version of more recent work
IC-4: The full text answers at least one research question	

gathered characteristics mentioned by practitioners for these technologies. Some examples are integration servers such as Jenkins, GitLab CI, and Travis CI. Also, as part of the findings of the category, we made a comparison and characteristics of usage of each technology.

Characteristics of containerization technologies: The use of containerization technologies, such as Docker, is one of the characteristics that popularized the microservices architecture. Many studies recommend the use of this technology, contrasting the advantages with respect to virtual machines. Deploying microservices using containers takes significantly less time than using virtual machines. The use of containers makes deployment a simple, fast, and platform-independent process. The mentioned benefits come from the fact that developers can automate the construction and provisioning of containers using scripts.

Characteristics of orchestration technologies: As a result of the wide adoption of containerization technologies, solutions for their orchestration have emerged. Technologies such as Kubernetes, Docker Swarm, and Docker-compose, among others, provide practitioners with various deployment benefits. This category presents a comparison in terms effectiveness of these technologies, and also compiles the experiences that developers had with their adoption. Kubernetes for container orchestration is the most suitable method for deploying microservices when the application demands high availability and scalability, however when it comes to security Kubernetes and Docker Swarm do not provide complete isolation between deployed containers, which introduces security issues.

Answer to research questions. *What DevOps practices and approaches support the deployment of Microservices?* The studies mentioned the DevOps practices of Continuous Integration (CI), Continuous Delivery (CD), Continuous Deployment and Monitoring. However, some studies did not directly mention the use of DevOps practices but used the processes and activities of these practices. Fig 4 shows the practices reported and related articles. It is worth noting that some studies mentioned more than one practice.

What technologies do DevOps practices use to deploy Microservices? We found several technologies for the construction and deployment of microservices. Figure 5 presents the ten most frequently reported technologies.

Studies mentioned Docker, a containerization technology, 16 times. The literature compares containers with other similar technologies such as Virtual Machines (VM), and in each comparison, the studies concluded that the former provided more significant benefits. The literature also highlights DockerHub as a repository for container images.

Another important technology is Jenkins, a building technology used in CI/CD practices, mentioned in the literature eight times. In contrast, the literature only mentions once Circle CI and Travis-CI, which are similar to Jenkins.

Among deployment and orchestration technologies, the literature mentions Kubernetes, Docker-compose, and Docker Swarm. Kubernetes was the

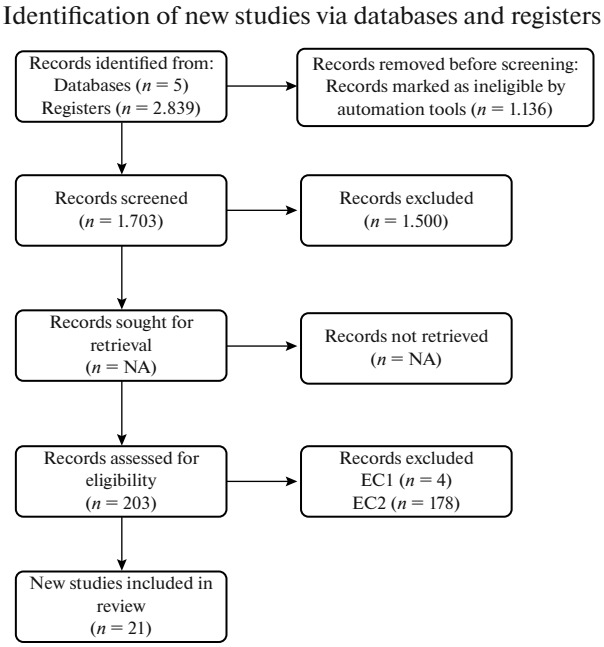

Fig. 2. Selection process.

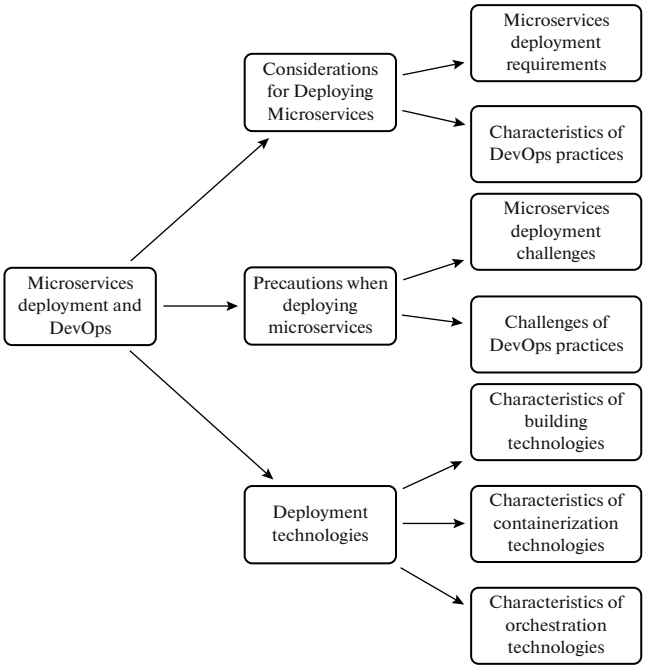

Fig. 3. Meta-aggregation classification.

most used because it provides significant benefits in systems with many microservices. Finally, the literature also mentions GitHub and Gitlab four and three times, respectively.

What challenges does the literature report regarding the adoption of DevOps practices in the deployment of microservices?

Publishing and upgrading microservices: Updating and publishing a new microservice version is a significant challenge, developers have to be careful since a microservice may depend on many others [21]. In addition, service discovery is a challenging aspect affected by upgrading a new version of a microservice and deploying it [24].

Technologies and tools required for building and deploying microservices: Developers make a great effort to adopt new tools and frameworks for each practice that they implement [27]. It is crucial to choose the right tools to protect the DevOps approach; otherwise, the rollback or tool change is very costly in time and effort [28]. Developers must perform careful initial configuration of the tools as this will allow correct automation [18]. Constant updates of libraries and tools make development and maintenance difficult.

Monitoring of a microservices architecture: The challenges that practitioners must face are the lack of commercial monitoring options, lack of standardization, and lack of faster learning curves [42].

What lessons does the literature report for successful microservices deployment? We grouped the lessons learned into two main topics: Solid architectural foundations and Attention to DevOps principles.

Solid architectural baseline: A long and scalable system requires a good architectural foundation that supports DevOps [31]. Every change in the architecture imposes new requirements on the delivery system and the implementation of new components and technologies [33]. Backward compatibility between microservices, separation of domains, and responsibilities for each service helps to prevent cross-configuration and keep services running smoothly.

Attention to DevOps principles: Applying DevOps principles in building CI/CD pipelines makes them leaner and more robust. Principles such as automation in all processes (integration, testing, deployment, analysis, and monitoring) are key to ensuring system reliability [26]. Good design and implementation of deployment pipelines allow rapid error detection [28]. Maintenance and updating of pipelines should take priority over code development. When problems arise, it is important to centralize error handling, in order to reduce the work of developers and operators. System monitoring should be flexible and scalable.

3.2.2. Gray literature review. We conducted a gray literature review to complement the mapping findings. For the review, we considered books, and electronic resources focused on the topics of DevOps, microservices deployment, and associated technologies. We searched the resources using the search engines Google Scholar, Google Books, and Google. We used these three since we aimed to have as much information as possible. In addition, we applied the snowball-

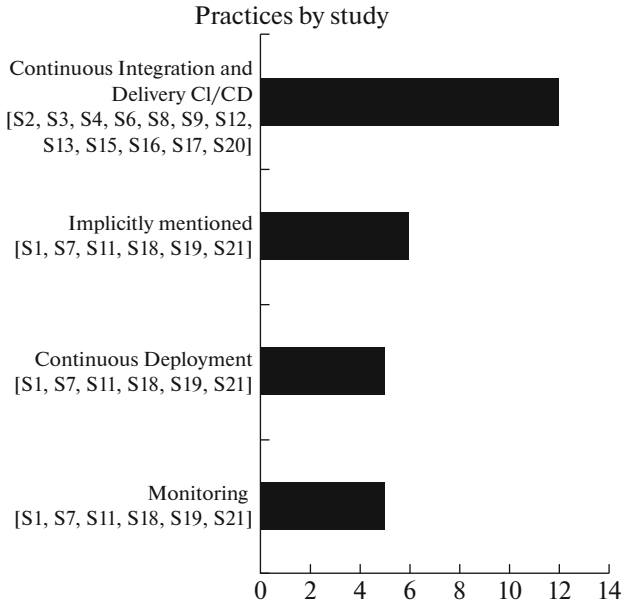

Fig. 4. Practices mentioned by study.

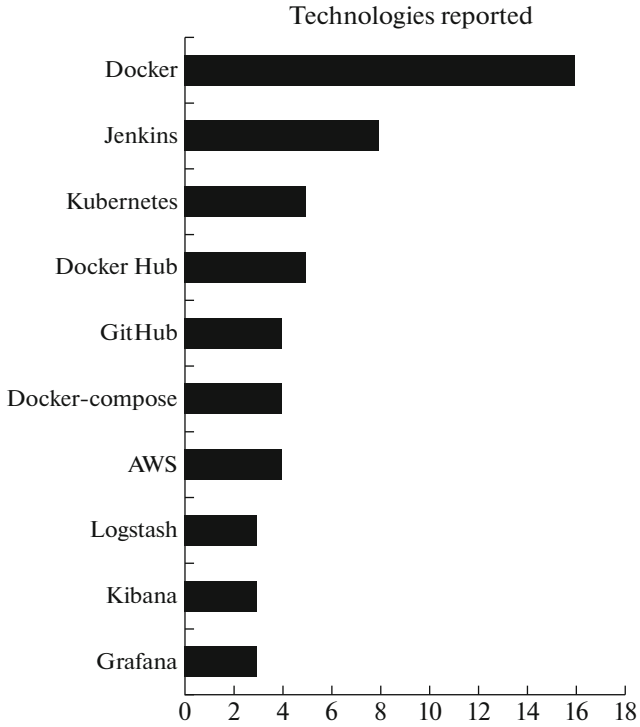

Fig. 5. Technologies reported by the studies.

ing method [43], which consists of searching for the material cited or referenced in the mapping articles. The steps carried out for the selection of the resources were as follows: For the selection of books, the process consisted of reading the table of contents, and the chapters that corresponded to the deployment of microservices or some DevOps practice related to microservices. For the selection of electronic resources such as company blogs, standards, and technical documentation, we read the content to determine if it would be useful. We investigated each of the resources to answer the research questions formulated in the MSL, or at least to find information that contributes to the findings.

Once we identified the resources, we continued with the reading of the most relevant aspects. Following a process similar to the meta-aggregation method used in the mapping, we identified important ideas or findings, and classify them according to their type. Among the types identified are deployment patterns, principles, practices, advantages, and disadvantages of technologies and resources.

3.3. Design and Development

In the design and development phase, we performed a series of activities, these activities consisted of grouping and classifying the information obtained from the white and gray literature reviews. We focus on the implementation modeling process of a microservices architecture, aiming to provide an order to the set of tasks and activities that we identified in previous phases. Finally, using the modeling and the information obtained, we integrated the microservices deployment guide, which we structured according to the modeling phases, having as content the related activities in each phase.

3.4. Demonstration and Evaluation

The demonstration aims to use the artifact to solve one or more instances of the problem. To achieve it, the authors propose certain approaches such as experimentation, simulation, case study, or other appropriate activity. Once performed the demonstration is needed to observe and measure how well the artifact supports a solution to the problem. However, given the complexity, the amount of time, personnel, and resources involved in building a microservices architecture large enough to be applied as a case study, as well as the number of case studies that would be needed to have deterministic results, it was decided not to include this phase in the scope of this work. For the evaluation of the guide, we decided to use another approach and analyze the evaluation method that best suits our problem, so far, we are considering using the work of Garousi et al. [44] and focusing on the evaluation of quality for technical software documents, thus the application of the evaluation is planned as future work.

3.5. Communication

As a part of the communication phase, we communicated the importance of the problem through the

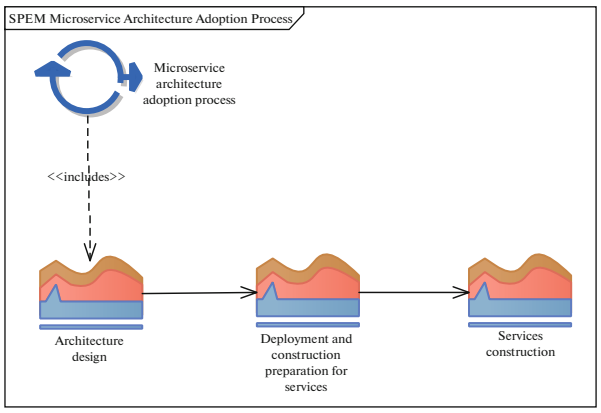

Fig. 6. Microservice architecture adoption process.

paper publication Microservice Deployment: A Systematic Mapping Study [15]. For the artifact communication, its utility, and effectiveness we present the current paper, and we are developing a website to publish the guide so it could be accessible for the practitioners.

4. PROPOSED DEPLOYMENT GUIDE

The guide works as a path where practitioners can identify their starting point and gradually adopt practices and strategies for microservices deployment. The guide includes practices, patterns [45], technologies, and tips found in the literature. The guide organizes possible decisions according to the phases of the microservices deployment process. Organizations interested in adopting the MSA can follow the guide, in this way, the person in charge of design or deployment can consult the practices and strategies recommended for each specific phase. The intention of showing the decisions in a modular way is that the managers can consult the parts they need, without the need to read the whole guide, or if practitioners have already managed to adopt some practices, they can find additional information that allows them to improve their current process.

We used SPEM 2.0 (Software & Systems Process Engineering Metamodel) for the modeling of the guide, it is a standard for defining software processes. SPEM uses the UML (Unified Modeling Language) notation, which provides components that allow the standardized representation of methods, life cycles, roles, activities, tasks, and work products used in Software Engineering. The main process consists of three phases. Each phase can have different iterations, an iteration is a set of activities performed iteratively, and each activity has one or many tasks needed to complete the activity. Due to the time involved in having a platform that supports the microservices architecture, practitioners can perform all these activities iteratively and incrementally as the project develops, thus adding value to the deployment process as the project and its needs grow.

The first phase corresponds to the architectural design, separating the problem domain, identifying the required microservices, the communication style between them, and the deployment method for orchestrating the microservices. The second phase presents the preparation of the development environment for each microservice; the related activities in the construction; integration and delivery of each service; and finally, the strategies for delivery and observability of the microservices in the production environment. The third and last phase, covers microservice construction, following the design and platform created in the previous phases. Figure 6 describes the relations between these phases. The following is a description of the sections that make up the guide as well as the related activities and tasks.

4.1. Deployment Design

This section of the guide covers the design and deployment planning iteration, which has four main activities for those responsible for the design and implementation of the system. Each activity has an output that serves as input for the next task, the first activity is the selection of the deployment strategy, followed by the selection of technologies, and finally, the last two activities, possibly executed in parallel, corresponding to the design of configurable services, and the design of observable services can.

The activities described in this section contain the following information: Name, Roles in charge, Description, List of identified methods or patterns, and Recommendations. Each identified pattern has the following properties: Characteristics, Advantages, Disadvantages, and Technology. These activities are described in figure 7.

4.2. Configuration Management and Development Environment

This section encompasses the Iteration Delivery Environment Preparation activity for the preparation of the deployment pipeline. This activity is very important since it is the basis that will allow the implementation of a deployment pipeline, the person in charge of the deployment has the task of implementing a set of practices and technologies that allow the control of the changes made in the service's code, as well as the automation of the processes for the construction of services. Figure 8 shows the activities implemented, among these are the Implementation of version control, Establishment of development guidelines, Implementation of patterns for source code branch manage-

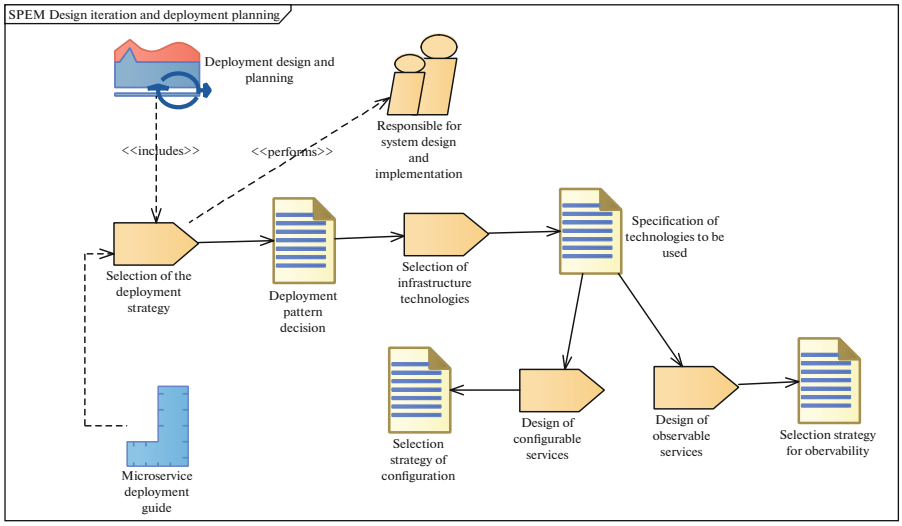

Fig. 7. Design iteration and deployment planning.

ment, implementation of unit tests, and automation of the build and test processes.

4.3. Deployment Pipeline

This section of the guide presents DevOps activities related to Continuous Integration and Continuous Delivery practices. The section incorporates two activities from the iteration phase of the deployment pipeline: the preparation of the built environment, and the preparation of the delivery environment. These activities are fundamental to constantly building and releasing microservices, a key aspect of successfully implementing MSA. The section features recommendations, technologies, and features for each task. The first activity corresponds to the practice of Continuous Integration, this activity concerns the implementation of a continuous integration system; automation of the compilation process; implementation of unit and acceptance tenting; implementation of code analysis and generation of binaries; and packaging artifacts. Figure 9 presents the relations between the tasks. The second activity, focused on the Continuous Delivery practice, concerns the tasks of environment configuration; implementation of smoke tests; implementation of manual tests; acceptance or performance tests; and deployment and release to a production environment. These tasks are shown in Fig. 10.

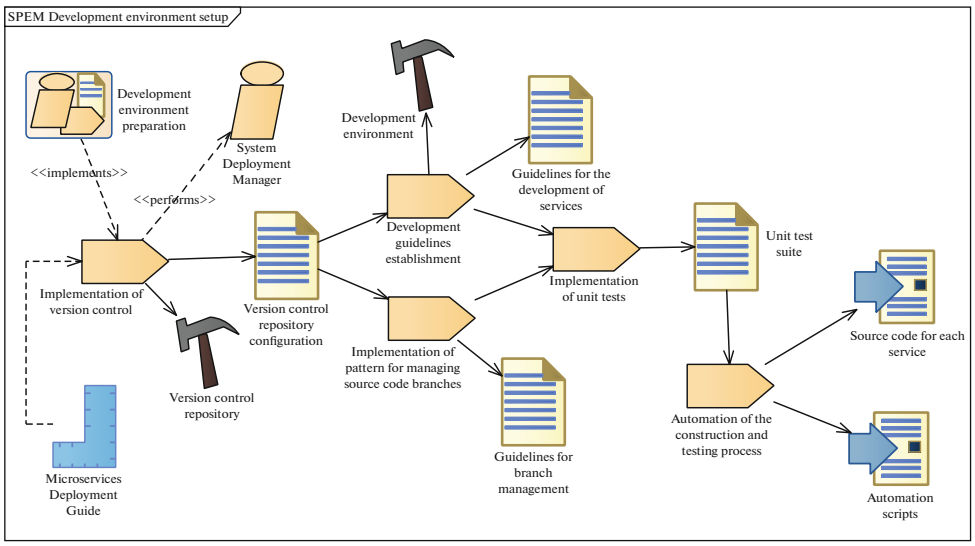

Fig. 8. Development environment setup.

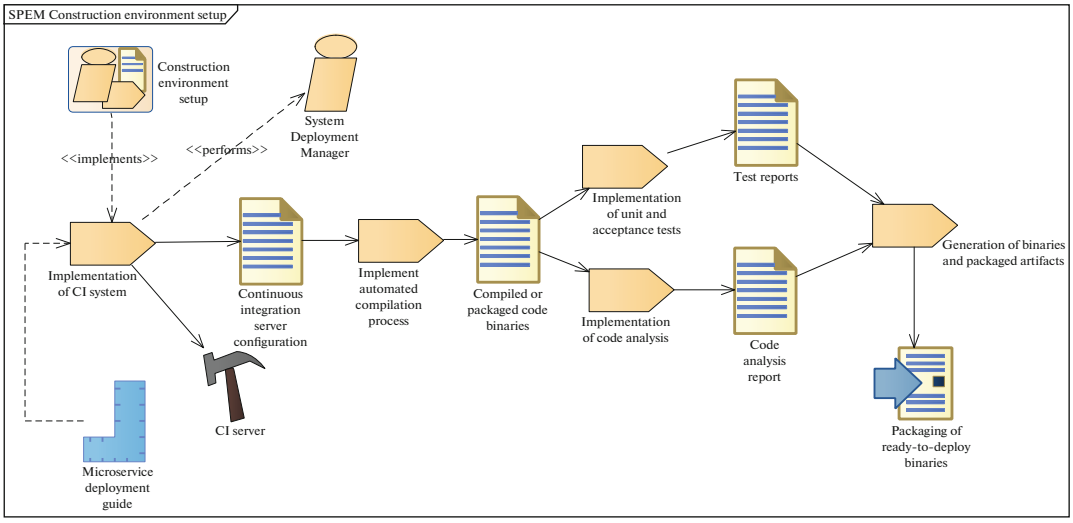

Fig. 9. Construction environment setup.

4.3. Infrastructure Management and System Observability

This section presents the tasks that correspond to DevOps culture practices, such as Infrastructure as Code and GitOps. Here we present the description of these practices, the description of the existing technologies, as well as good practices found in the literature for their correct implementation. In addition, the last section presents the practices we found in the literature to achieve adequate observability of the services deployed in a production environment. Figure 11 introduced the tasks of the section.

5. CONCLUSION AND FUTURE WORK

This paper presented the current results of a project to build a deployment guide for applications with a microservices architectural style. To this end, we conducted a systematic mapping study to identify the practices, tools, technologies, activities, and recommendations used in microservices deployment, we also complemented the information found with a gray literature review. We integrated into the guide all the elements and models found.

As for future work, we plan to perform the evaluation phase of the DSRM methodology. This phase is for analyzing the guide and related artifacts, to know if they meet the intended objectives. To perform the

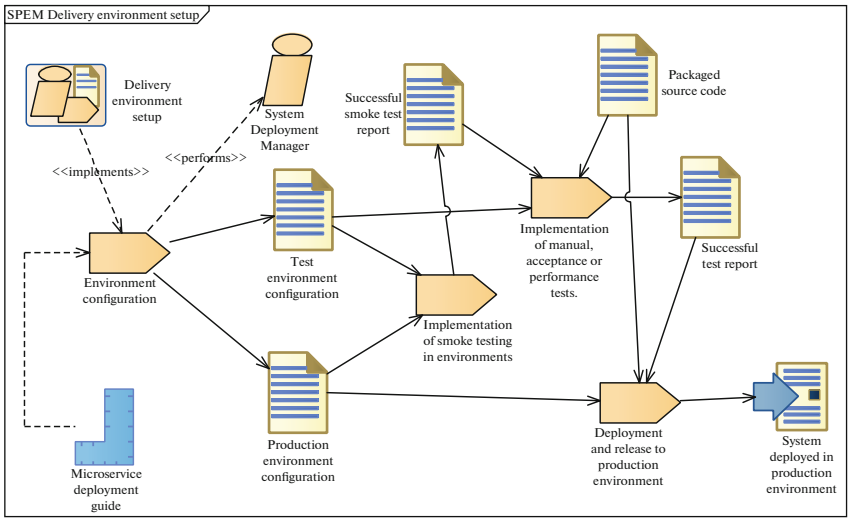

Fig. 10. Delivery environment setup.

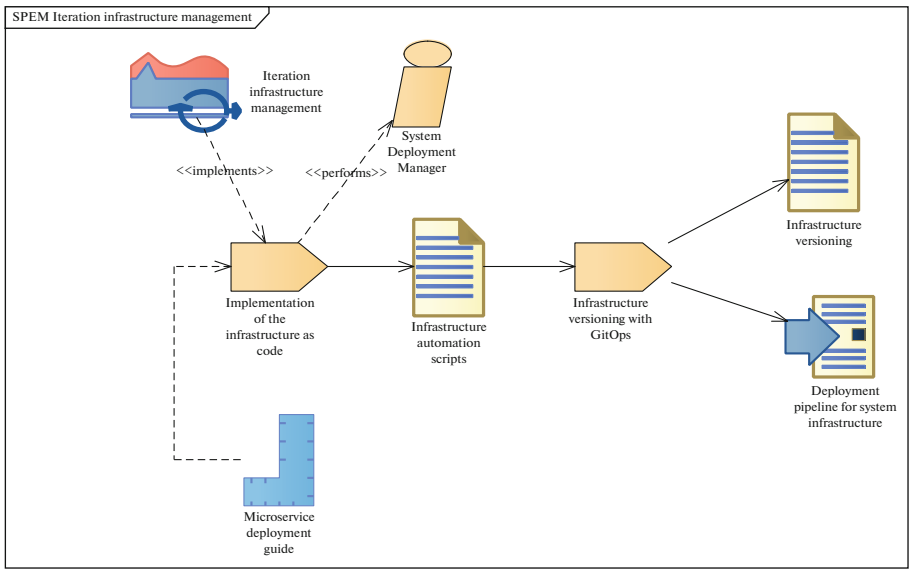

Fig. 11. Infrastructure management.

evaluation of the guide we intend to use the work of Garousi et al. [44] for the evaluation of the use and quality of software technical documentation.

The present version of the artifact does not cover organizational aspects of the DevOps culture. To obtain the benefits of a DevOps culture, organizations not only have to adopt technologies and practices, but they also have to adopt an organizational and cultural base, driven by the highest levels of the organization. Therefore, as future work, the guide will incorporate the organization of effective teams for microservices deployment. In this way, the work would bring additional value to organizations and to all those who seek to adopt a DevOps culture.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. Mauro, T., Adopting microservices at Netflix: lessons for architectural design, nginx blog, 2015. https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/. Accessed May 10, 2021.
2. Rewriting Uber Engineering: the opportunities microservices provide Uber Engineering Blog. https://eng.uber.com/building-tincup-microservice-implementation/. Accessed May 10, 2021.
3. Ihde, S., From a monolith to microservices + REST: the evolution of LinkedIn's service architecture, Mar. 2015. https://www.infoq.com/presentations/linkedin-microservices-urn/. Accessed Mar. 22, 2022.
4. Calcado, P., Building products at SoundCloud – part I: dealing with the monolith SoundCloud backstage blog, June 11, 2014. https://developers.soundcloud.com/blog/building-products-at-soundcloud-part-1-dealing-with-the-monolith. Accessed Mar. 22, 2022.
5. Lewis, J. and Fowler, M., Microservices, Mar. 25, 2014. https://martinfowler.com/articles/microservices.html. Accessed Nov. 16, 2021.
6. Martin, R.C., Clean coder blog, May 8, 2014. https://blog.cleancoder.com/uncle-bob/2014/05/08/SingleReponsibilityPrinciple.html. Accessed Jan. 26, 2022.
7. Newman, S., *Building Microservices,* O'Reilly Media, 2015.
8. Indrasiri, K. and Siriwardena, P., *Microservices for the Enterprise,* Apress, 2018.
9. Olszewska, J.I., *IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment: IEEE Standard 2675-2021,* 2021.
10. Richardson, C., *Microservices Patterns: with Examples in Java,* Simon and Schuster, 2018.
11. Fritzsch, J., Bogner, J., Wagner, S., and Zimmermann, A., Microservices migration in industry: intentions, strategies, and challenges, *Proc. IEEE Int. Conf. Software Maintenance and Evolution ICSME 2019,* Cleveland, 2019, pp. 481–490. https://doi.org/10.1109/ICSME.2019.00081
12. Bruce, M. and Pereira, P.A., *Microservices in Action,* Manning Publications Co., 2018.
13. Shahin, M., Ali Babar, M., and Zhu, L., Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices, *IEEE Access,* 2017, vol. 5, pp. 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629
14. Peffers, K., Tuunanen, T., Rothenberger, M.A., and Chatterjee, S., A design science research methodology for information systems research, *Manag, J. Inf. Syst.,* 2007, vol. 24, no. 3, pp. 45–77. https://doi.org/10.2753/MIS0742-1222240302

15. Niño-Martínez, V.M., Ocharán-Hernández, J.O., Limón, X., and Pérez-Arriaga, J.C., Microservices deployment: a systematic mapping study, *Proc. 9th Int. Conf. in Software Engineering Research and Innovation (CONISOFT),* 2021, pp. 24–33.
16. Debroy,V., Miller, S., and Brimble, L., Building lean continuous integration and delivery pipelines by applying devops principles: a case study at varidesk, *Proc. 26th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering ESEC/FSE 2018,* Lake Buena Vista, FL, 2018, pp. 851–856. https://doi.org/10.1145/3236024.3275528
17. Eismann, S., Kistowski, J.V., Grohmann, J., Bauer, A., Schmitt, N., and Kounev, S., TeaStore – a micro-service reference application, *Proc. 4th IEEE Int. Workshops on Foundations and Applications of Self* Systems (FAS*W) 2019,* Umea, 2019, pp. 263–264. https://doi.org/10.1109/FAS-W.2019.00073
18. Fan, C.Y. and Ma, S.P., Migrating monolithic mobile application to microservice architecture: an experiment report, *Proc. 6th IEEE Int. Conf. on AI & Mobile Services (AIMS),* Honolulu, 2017, pp. 109–112. https://doi.org/10.1109/AIMS.2017.23
19. Grobmann, M. and Ioannidis, C., Continuous integration of applications for onos, *Proc. IEEE Conf. on Network Softwarization (NetSoft),* Paris, 2019, pp. 213–217. https://doi.org/10.1109/NETSOFT.2019.8806696
20. Kang, H., Le, M., and Tao, S., Container and microservice driven design for cloud infrastructure DevOps, *Proc. IEEE Int. Conf. Cloud Engineering IC2E 2016 Colocated with 1st IEEE Int. Conf. Internet-of-Things Design Implementation, IoTDI 2016,* Brilin, 2016, pp. 202–211. https://doi.org/10.1109/IC2E.2016.26
21. Kargar, M.J. and Hanifizade, A., Automation of regression test in microservice architecture, *Proc. 4th Int. Conf. on Water Research ICWR 2018,* Tehran, 2018, pp. 133–137. https://doi.org/10.1109/ICWR.2018.8387249
22. Rajavaram, H., Rajula, V., and Thangaraju, B., Automation of microservices application deployment made easy by rundeck and kubernetes, *Proc. IEEE Int. Conf. Electron. Comput. Commun. Technol. CONECCT 2019,* Bangalore, 2019, pp. 3–5. https://doi.org/10.1109/CONECCT47791.2019.9012811
23. Sarita, N. and Sunil, S., Transform monolith into microservices using docker, *Proc. Int. Conf. Int. Conf. on Computing, Communication, Control and Automation (ICCUBEA) ICCUBEA 2017,* Pune, 2017, pp. 1–5. https://doi.org/10.1109/ICCUBEA.2017.8463820
24. Singh, V. and Peddoju, S.K., Container-based microservice architecture for cloud applications, *Proc. IEEE Int. Conf. on Computing, Communication and Automation ICCCA 2017,* Greater Noida, 2017, pp. 847–852. https://doi.org/10.1109/CCAA.2017.8229914
25. Hakli, A., Taibi, D., and Systa, K., Towards cloud native continuous delivery: an industrial experience report, *Proc. 11th IEEE/ACM Int. Conf. on Utility and Cloud Computing Companion, UCC Companion 2018,* Zurich, 2018, pp. 314–320. https://doi.org/10.1109/UCC-Companion.2018.00074
26. Hasselbring, W. and Steinacker, G., Microservice architectures for scalability, agility and reliability in e-commerce, *Proc. IEEE Int. Conf. on Software Architecture Workshops ICSAW 2017,* Gothenburg, 2017, pp. 243–246. https://doi.org/10.1109/ICSAW.2017.11
27. Richter, D., Konrad, M., Utecht, K., and Polze, A., Highly-available applications on unreliable infrastructure: microservice architectures in practice, *Proc. IEEE Int. Conf. Software Quality, Reliability and Security Companion, QRS-C 2017,* Prague, 2017, pp. 130–137. https://doi.org/10.1109/QRS-C.2017.28
28. Soenen, T., et al., Insights from SONATA: implementing and integrating a microservice-based NFV service platform with a DevOps methodology, *Proc. IEEE/IFIP Network Operations and Management Symp. NOMS 2018,* Taipei, 2018, pp. 1–6. https://doi.org/10.1109/NOMS.2018.8406139
29. Yang, D., et al., DevOps in practice for education management information system at ECNU, *Procedia Comput. Sci.,* 2020, vol. 176, pp. 1382–1391. https://doi.org/10.1016/j.procs.2020.09.148
30. Barna, C., Khazaei, H., Fokaefs, M., and Litoiu, M., Delivering elastic containerized cloud applications to enable DevOps, *Proc. 12th IEEE/ACM Int. Symp. on Software Engineering for Adaptive and Self-Managing Systems SEAMS 2017,* Buenos Aires, 2017, pp. 65–75. https://doi.org/10.1109/SEAMS.2017.12
31. Chen, H.M., Kazman, R., Haziyev, S., Kropov, V., and Chtchourov, D., Architectural support for DevOps in a neo-metropolis BDaaS platform, *Proc. 34th IEEE Symp. on Reliable Distributed Systems Workshop (SRDSW),* Montreal, 2015, vol. 2016, pp. 25–30. https://doi.org/10.1109/SRDSW.2015.14
32. Haselbock, S. and Weinreich, R., Decision guidance models for microservice monitoring, *Proc. IEEE Int. Conf. on Software Architecture Workshops ICSAW 2017,* Gothenburg, 2017, pp. 54–61. https://doi.org/10.1109/ICSAW.2017.31
33. Steffens, A., Lichter, H., and Döring, J.S., Designing a next-generation continuous software delivery system: concepts and architecture, *Proc. Int. Conf. on Software Engineering,* Gothenburg, 2018, pp. 1–7. https://doi.org/10.1145/3194760.3194768
34. Taibi, D., Lenarduzzi, V., and Pahl, C., Continuous architecting with microservices and DevOps: a systematic mapping study, *Commun. Comput. Inf. Sci.,* 2019, vol. 1073, pp. 126–151.
35. Bolscher, R. and Daneva, M., Designing software architecture to support continuous delivery and DevOps: a systematic literature review, *Proc. 14th Int. Conf. on Software Technologies,* Prague, 2019, pp. 27–39. https://doi.org/10.5220/0007837000270039
36. Waseem, M., Liang, P., and Shahin, M., A systematic mapping study on microservices architecture in DevOps, *J. Syst. Software,* 2020, vol. 170, p. 110798. https://doi.org/10.1016/j.jss.2020.110798
37. Hyvärinen, H., Risius, M., and Friis, G., A blockchain-based approach towards overcoming financial fraud in public sector services, *Bus. Inf. Syst. Eng.,* 2017, vol. 59, no. 6, pp. 441–456. https://doi.org/10.1007/s12599-017-0502-4

38. Tello-Rodríguez, M., Ocharán-Hernández, J.O., Pérez-Arriaga, J.C., Limón, X., and Sánchez-García, Á.J., A design guide for usable web APIs, *Program. Comput. Software,* 2020, vol. 46, no. 8, pp. 584–593. https://doi.org/10.1134/S0361768820080241

39. Chen, L., Continuous delivery: overcoming adoption challenges, *J. Syst. Software,* 2017, vol. 128, pp. 72–86. https://doi.org/10.1016/j.jss.2017.02.013

40. Kitchenham,B., Budgen, D., and Brereton, P., *Evidence-Based Software Engineering and Systematic Reviews,* CRC Press, 2015.

41. Pearson, A., Robertson-Malt, S., and Rittenmeyer, L., *Synthesizing Qualitative Evidence,* Wolters Kluwer, 2011.

42. Tamburri, D.A., Miglierina, M., and Di Nitto, E., Cloud applications monitoring: an industrial study, *Inf. Software Technol.*, 2019, vol. 127, p. 106376. https://doi.org/10.1016/j.infsof.2020.106376

43. Wohlin, C., Guidelines for snowballing in systematic literature studies and a replication in software engineering, *Proc. 18th Int. Conf. on Evaluation and Assessment in Software Engineering EASE'14,* London, 2014. https://doi.org/10.1145/2601248.2601268

44. Garousi, G., Garousi, V., Moussavi, M., Ruhe, G., and Smith, B., Evaluating usage and quality of technical software documentation: an empirical study, *Proc. 17th Int. Conf. on Evaluation and Assessment in Software Engineering,* Porto de Galinhas, 2013, pp. 24–35. https://doi.org/10.1145/2460999.2461003

45. Valdivia, J.A., Lora-González, A., Limón, X., Cortes-Verdin, K., and Ocharán-Hernández, J.O., Patterns related to microservice architecture: a multivocal literature review, *Program. Comput. Software,* 2020, vol. 46, no. 8, pp. 594–608.