Survey on Microservice Architecture - Security, Privacy and Standardization on Cloud Computing Environment

Washington Henrique Carvalho Almeida, Luciano de Aguiar Monteiro, Raphael Rodrigues Hazin, Anderson Cavalcanti de Lima and Felipe Silva Ferraz
Center of Advanced Studies and Systems of Recife
Recife, Brazil
E-mail: {washington.hc.almeida, lucianoaguiarthe, raphaelhazin, andclima}@gmail.com
E-mail: {fsf}@cesar.org.br

***Abstract* — Microservices have been adopted as a natural solution for the replacement of monolithic systems. Some technologies and standards have been adopted for the development of microservices in the cloud environment; API and REST have been adopted on a large scale for their implementation. The purpose of the present work is to carry out a bibliographic survey on the microservice architecture focusing mainly on security, privacy and standardization aspects on cloud computing environments. This paper presents a bundle of elements that must be considered for the construction of solutions based on microservices.**

Keywords-Microservice; Cloud; Architecture; API; REST

I. INTRODUCTION

Migration of the monolithic architecture to the cloud has been a major problem. In this paper a research was carried out on the topic of microservices that have been adopted as a natural solution in the replacement of monolithic systems. The main question lies in how its architecture has been used and issues of security and privacy keys in a cloud computing environment. The motivation for this collection was the fact that more and more microservices have been found as a solution for applications in the cloud. Cloud computing provides a centralized pool of configurable computing resources and computing outsourcing mechanisms that enable different computing services to different people in a way similar to utility-based systems, such as electricity, water, and sewage.

For the recent advances of cloud computing technologies, the use of microservices on applications has been more widely addressed due to the rich set of features in such architecture. These applications can be deployed on clouds that make users use it at low cost, threshold, and risk. Therefore, their practical use in business can be expected as a trend for the next generation of business applications [1].

Scaling monolithic applications is a challenge because they commonly offer a lot of services. Some of them are more popular than others. If popular services need to be scaled because they are highly demanded, the whole set of services will also be scaled at the same time, which implies that unpopular services will consume a large amount of server resources even when they are not going to be used [2].

The architecture based on microservices has emerged to simplify this reality and are a natural evolution to application models.

Microservices are a software oriented entity, which have the following features [3]:

Isolation from other microservices, as well as from the execution environment based on a virtualized container;

Autonomy – microservices can be deployed, destroyed, moved or duplicated independently. Thus, microservices cannot be bound to any local resource because microservice environment can create more than one instance of the same microservice;

Open and standardized interface that describes all specific goals with effectiveness, efficiency and available communication methods (either API or GUI);

Microservice is fine-grained – each microservice should handle its own task.

The microservice architecture is a cloud application design pattern that implies that the application is divided into a number of small independent services, each of which is responsible for implementing a certain feature, as noted in Figure 1.

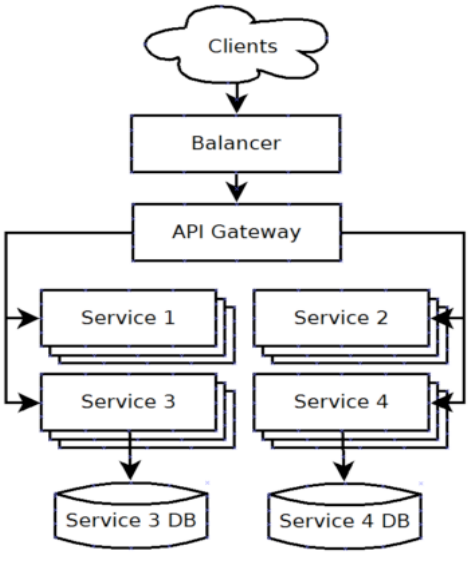

Figure 1. Microservice system architecture[3].

 ISBN: 978-1-61208-590-6

Microservices can be considered meta-processes in a Meta operating system (OS); they are independent, they can communicate with each other using messages and they can be duplicated, suspended or moved to any computational resource and so on [3].

The remainder of this article is structured as follows: Section II introduces the architecture of implemented microservices. Section III presents security in the cloud computing environment and Section IV shows the privacy model adopted in the cloud applications for microservices. In Section V, we present the standards of cloud environment and then conclude and summarize all the results of that exercise in Section VI.

II. MICROSERVICE ARCHITECTURE

The microservice architecture has become a dominant architectural style choice in the service oriented software industry. Microservice is a style of architecture that puts the emphasis on dividing the system into small and lightweight services that are purposely built to perform a very cohesive business function, and is an evolution of the traditional service oriented architecture style [4].

The idea of splitting an application into a set of smaller and interconnected services (microservice) is currently getting many interests from application developers and service providers (e.g., Amazon [5][6], Netflix [7][8], eBay [9][10]).

A Microservice based architecture has a pattern for development of distributed applications, where the application is composed of a number of smaller "independent" components; these components are small applications in themselves [11].

A microservice normally comprises three layers as a typical 3-tiered application [12], consisting of an interface layer [13], a business logic layer [9] and a data persistence layer, but within a much smaller bounded context. This sets a broad scope of the technical capabilities that a microservice could possess. However, not every microservice provides all capabilities. This would vary depending on how the function provided is meant to be consumed. For example, a microservice used primarily by providers of API's would have a communications interface layer, business logic and data persistence layers but not necessarily have user interfaces [11].

We are considering a reference architecture model of microservices, demonstrating the main components and elements of this standard [11]. Table 1 presents a comparison between monolithic architecture and microservice architecture

TABLE I. COMPARING MONOLITHIC AND MICROSERVICE ARCHITECTURE [14]

Category	Monolithic Architecture	Microservice Architecture
Code	A single code base for the entire application.	Multiple code bases. Each microservice has its own code base.
Understandability	Often confusing and hard to maintain.	Much better readability and much easier to maintain.
Deployment	Complex deployments with maintenance windows and schedules downtimes.	Simple deployment as each microservice can be deployed individually, with minimal if not zero downtime.
Language	Typically, entirely developed in one programing language.	Each microservice can be developed in a different programing language.
Scaling	Requires you to scale the entire application even though bottlenecks are localized.	Enables you to scale bottle-necked services without scaling the entire application.

In this paper, we will cover the following main elements:

A. API Proxy

To "de-couple" the microservice from its consumers, this proxy pattern is applied at the microservice interface level, regardless of the "API proxy" component. Organizations will provide API's to different consumers, some of whom are within and others outside the enterprise. These microservices would differ in service level agreements (SLA), security requirements, access levels, etc [11].

B. Enterprise API Registry

The "discovery" requirements of the microservices are met through the use of the API registry service. Its purpose is to make the interfaces exposed by the microservice visible to consumers of the services both within and outside the enterprise. An "Enterprise API registry" is a shared component across the enterprise, whose location must be well known and accessible. Its information content is published in a standard format, information should be in consistent and human readable format, and must have controlled access. It must have search and retrieval capabilities to allow users to look up details on available API specifications at design time [11].

C. Enterprise Microservice Repository

The "enterprise microservice repository" would be a shared repository for storing information about microservices. It provides information such as microservice lifecycle status, versions, business and development ownership, detailed information like its purpose, how it achieves the purpose, tools, technologies, architecture, the service it provides, any API's it consumes, data persisted and queried and any specific non-functional requirements. In the absence of well-defined repository standards, the enterprise must define its own standard specification artefacts for microservices [5].

 ISBN: 978-1-61208-590-6

These elements are fundamental to the organized implementation of microservices and have been considered in this survey.

III. SECURITY ON CLOUD COMPUTING

Switching from a monolithic or centralized architecture to a decentralized architecture requires some care. In the past, security was focused on a single point [15], responsible for receiving all service requests. In the microservice-based architecture, the resources are offered through several points of access that interconnect each other, forming a unique solution.

Monolithic security services are relatively easier to implement than microservices. Monolithic services have a clear boundary and encapsulate their intercommunications. This will obscure security vulnerabilities [16][17] within the inner layers of the system. A microservice also encapsulates its communications. Both microservices and services are based upon clear requirements.

In a microservice-based system a simple routine completion requires the microservices to communicate with each other over network, for example. This will expose more data and information (endpoints) about the system and thus it expands the attack surface [8]. Some care must be taken in the communication between other services in the same network, and this is one of the major challenges [13][15][18] in this approach.

The organization of teams for the development of a system based on microservices are generally subdivided into teams and services, and these teams are generally responsible for the implementation and delivery of services. For this type of implementation, the teams have to be aligned in the purposes of the microservices and the interconnection between them, thus also synchronizing the protocol [19] used to carry out the communication, thus respecting a standard for access protection or improper interception. Defining the way services are interconnected and interacting is the key point of security [20].

The security challenge brought by such network complexity is the ever-increasing difficulty in debugging, monitoring, auditing and forensic analysis of the entire application [21]. Since microservices are often deployed in a cloud that the application owners do not control, it is difficult for them to construct a global view of the entire application [10].

In microservice architecture, an application is essentially a collection of workflows. These workflows can compose many levels of services, each processing and modifying the data before its final destination. What we need is a way to certify the metadata related to a data stream and manage its validity during time and re-elaboration [22].

Security is a major challenge that must be carefully thought of in microservices architecture. Services communicate with each other in various ways creating a trust relationship. For some systems, it is vital that a user is identified in all the chains of a service communication happening between microservices. OAuth and OAuth2 are well-known solutions that are employed by designers to handle security challenges [4].

Although the microservices are independent and do not cause dependencies among the modules, the biggest challenge nowadays is to guarantee availability [23]. The DevOps movement (set of practices to integrate the software development to IT operations) is currently collaborating with cloud environments and microservice architecture, providing continuous integration from the code compilation to the availability of the test and production environment, making it a facilitator for systems implementation utilizing microservices.

Ensuring the availability of services is presented as a security requirement facilitated by the use of the microservice architecture. This approach usually works by fragmenting the entire solution in smaller pieces [24]. Considering that these fragments are parts of the code with specific functions (microservices), in the event of a fragment failure, it would not result in the unavailability of all system resources. Availability has some critical points as they are bound to be observed such as: implementing software versions, software crash recovery, invasions, unavailability of infra features beyond points.

In a microservice architecture, it is typical for many instances of a particular service to be running at any one time and for these instances to stop and start over time [25]. The problem of service discovery is to enable service consumers to locate service providers in real time to facilitate communication [26]. Docker Containers have been gaining a lot of hard work because of their agility and ease of making new services available [23]. The containers allow the microservices to be packaged [27] and available next to their dependencies in a single image, thus facilitating the availability of the service in a timely manner, minimizing downtime. This mode is called code portability [28]. In the context of microservices, the use of docker containers for service delivery has resulted in benefits under various aspects, such as: automation, independence, portability and security, especially when considering ease of management, creation and continuous integration of environments systems offered by the docker platform. In Docker, each container consists of only the application and the dependencies that the application needs to run, ideally no more and no less [28].

IV. PRIVACY MODEL

Privacy has been a barrier to adoption of cloud computing [24][29]. The migration to microservices has helped overcome this obstacle due to the scale gains proposed in this architecture.

In general, privacy refers the condition or state of hiding the presence or view [30]. There is a need to attain this state in the places where confidential things are used such as data and files. In cloud data storage privacy is needed to attain the data, user identity and controls [31]

 ISBN: 978-1-61208-590-6

The exchange of sensitive data is intense in large-scale scenarios of cloud computing, with several federations, where multiple Identity Providers (IdP) and Service Providers (SP) work together to provide services. Therefore, identity management should provide models and privacy mechanisms in order to manage the sensitive data of its users [15].

Cloud service provides various options to the business customers to choose the level of protection needed for their data. The most common of these approaches is encryption. The customer chooses the type of encryption that they prefer and store the encryption key in a safe place under their control [19].

To ensure privacy, a well referenced model is used. This model is presented in Figure 2.

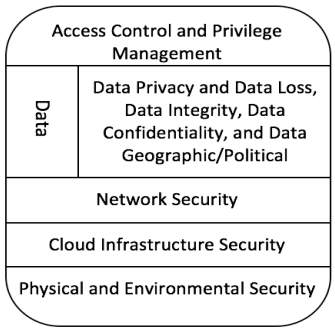

Figure 2. Cloud security and privacy model [29].

According to the proposed model in [29], a secure and private cloud model is divided into five layers: Physical and Environmental Security, Cloud Infrastructure Security, Network Security, Data and Access Control and Privilege Management.

1. *Physical and Environmental Security*
   Layer of policies adopted with the objective of protecting physical access to the cloud provider [5].
2. *Cloud Infrastructure Security*
   Addresses issues with cloud infrastructure security, but specifically with the virtualization environment [32].
3. *Network Security*
   Specifies the medium to which the end user connects to the cloud, comprising browsers and their connection [9].
4. *Data*
   Layer covers data privacy, integrity, confidentiality, and geographic location [22].
5. *Access Control and Privilege Management*
   Policies and processes used by cloud services provider to ensure that only the users granted appropriate privileges can use or modify data. It includes identification, authentication [33] and authorization issues [29].

V. MICROSERVICE STANDARDS AND SOLUTIONS

In the centralized structure, the standardization becomes almost a natural way, but in the implementation of microservices this philosophy changes.

Teams building microservices prefer a different approach to standards too. Rather than using a set of defined standards, written down somewhere on paper, they prefer the idea of producing useful tools that other developers can use to solve similar problems to the ones they are facing. These tools are usually harvested from implementations and shared with a wider group, sometimes, but not exclusively, using a git and github has become the de facto version control system of choice. Open source practices are becoming more and more common in-house [34].

A microservice is an application on its own to perform the functions required. It evolves independently and can choose its own architecture, technology, platform, and can be managed, deployed and scaled independently with its own release lifecycle and development methodology. This approach takes away the construct of the SOA and ESB and the accompanying challenges by making "smart endpoints" and treating the intermediate layers as network resources whose function is that of data transfer [11].

The applications that expose interfaces that can be used by other applications to interact with are defined as "application programming interfaces" (API) [5]. Microservice API's which are built using internet communication protocols like HTTP, adhere to open standards like REST [35][36] and SOAP [2] and use data exchange technologies like XML [18] and JSON [5].

Applications developed in a monolithic architecture perform multiple functions such as providing address validation, product catalogue, customer credit check, etc. When using the microservice based architecture pattern, applications are created for specific functions, such as address validation, customer credit check and online ordering; these applications are cobbled together to provide the entire capability for the proposed service. The approach to application development based on microservice architecture addresses the challenges of "monolithic" application and services [11].

In the research undertaken in this paper, the microservices are implemented and documented as follows:

A. Architectural views/diagrams [4]
- UML
- Standard modeling languages, e.g. RAML and YAML.
- Specifically designed modeling languages, e.g. CAMLE.
- Standard specification languages, e.g. Javascript (Node.js), JSON and Ruby.
- Specifically designed specification languages, e.g. Jolie.
- Pseudocode for algorithms.

B. REST
REpresentational State Transfer (REST) consisting of a set of architectural principles that, when followed, allows a well-defined interface design to be created. Applications that use REST principles are called RESTFul. REST [10][18][36][37] is often applied to provide services to other services (web services) and to the same full use of messages. To better understand the architectural style, it is

important to highlight three important concepts: (i) feature; (ii) operations and (iii) representations. Resource is any information that is made available to customers through a unique identifier (URI). We can also define resource as being the source of representations. The representations are a set of data that explains the state of the requested resource. URIs must have a notation pattern, be descriptive, and have a previously defined hierarchy. The same resource can be identified by one or more URIs, but a URI [38] [39] identifies only one resource.

C. API

Application Program Interface (API) are (a) basic authentication, including API user registration with strong password protection, (b) modern security mechanisms such as message level security, web signature and web encryption, and (c) security mechanism within API and its backend services as a third security factor such as token based API for backend authentication, public key infrastructure and transport layer handshake protocol [13].

REST APIs [7] are developed in many technologies and microservices developed using different types of programing languages (Java, .NET, PHP, Ruby, Phyton, Scala, NodeJs,etc.) and persistent technologies (SQL, No-SQL, etc.) [2][28]. They can be managed and exposed to web clients, who can then access the microservices and receive their responses through a "livequery" mechanism whereby updates to database data are instantly communicated to subscribing clients [18]. Figure 3 best presents categories of practices for designing REST-based web services.

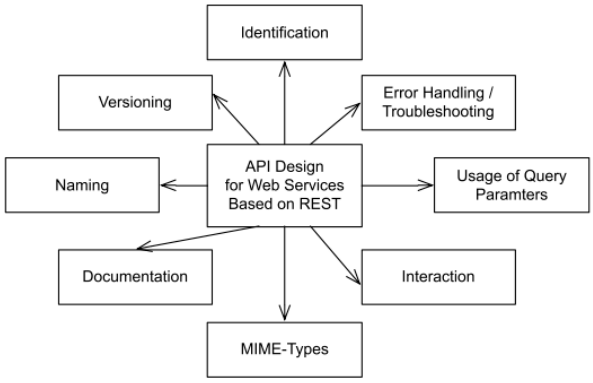

Figure 3. Categories of best practices for designing REST-based web services [40].

NoSQL database are used in these implementations [18][41][42][43]. The NoSQL nature of the database is essential for providing the scaling, sharding and replication functionality expected from modern architectures, as well as to better support hierarchical data required for collaborative document editing [18].

The popularity of the architecture based on microservices is evident from the report by the popular jobs portal *indeed.com,* in which the number of job openings on microservices-related technologies, such as JSON [10][20][39] and REST [2][18][36] has grown more than 100 times in the last six years, whereas jobs in similar technology areas like SOAP and XML have remained nearly identical [10].

Solutions for microservices seek to implement simple algorithms that meet specific needs with the elements presented in this section.

VI. CONCLUSIONS AND FUTURE WORK

Microservice-based architecture has been a growing choice as an architectural style for software development. In this architectural style, the services provided by software solutions are divided into smaller parts and focused on the specific service of some functionalities. The approach of developing microservices with the construction of smaller software components has a number of advantages over the traditional monolithic architecture, such as increasing the resilience of the software implemented as a microservice and the ease of scaling the solution implemented through the microservices.

The development of software using the microservice-based architecture comprises important aspects that must be observed in order to obtain good results. The objective of this article is to present the elements that should be considered for the development of solutions based on microservices, describing how the architecture based on microservices is defined, identifying the elements related to their implementation in the cloud computing environment, explaining the privacy model applicable and relating the elements that integrate the standards and solutions linked to the architecture based on microservices.

Future work can be developed to present case studies demonstrating the implementation of the microservice architecture in a cloud computing environment with the use of docker containers for its construction.

REFERENCES

[1] J. Lin, L. Chaoyu, and S. Huang, "Migrating Web Applications to Clouds with Microservices Architectures," *Int. Conf. Appl. Syst. Innov.*, pp. 1–4, 2016.

[2] M. Villamizar and et al, "Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud Evaluando el Patrón de Arquitectura Monolítica y de Micro Servicios Para Desplegar Aplicaciones en la Nube," *10th Comput. Colomb. Conf.*, pp. 583–590, 2015.

[3] D. I. Savchenko, G. I. Radchenko, and O. Taipale, "Microservices validation: Mjolnirr platform case study," *2015 38th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2015 - Proc.*, no. May, pp. 235–240, 2015.

[4] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," *Proc. - 2016 IEEE 9th Int. Conf. Serv. Comput. Appl. SOCA 2016*, pp. 44–51, 2016.

[5] A. Krylovskiy, M. Jahn, and E. Patti, "Designing a Smart City Internet of Things Platform with Microservice Architecture," *Proc. - 2015 Int. Conf. Futur. Internet Things Cloud, FiCloud 2015 2015 Int. Conf. Open Big Data, OBD 2015*, pp. 25–30, 2015.

 ISBN: 978-1-61208-590-6

[6] H. Khazaei, C. Barna, N. Beigi-Mohammadi, and M. Litoiu, "Efficiency analysis of provisioning microservices," *Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom*, pp. 261–268, 2017.

[7] R. Heinrich *et al.*, "Performance Engineering for Microservices: Research Challenges and Directions," *Proc. 8th ACM/SPEC Int. Conf. Perform. Eng. Companion*, pp. 223–226, 2017.

[8] M. Ahmadvand and A. Ibrahim, "Requirements reconciliation for scalable and secure microservice (de)composition," *Proc. - 2016 IEEE 24th Int. Requir. Eng. Conf. Work. REW 2016*, pp. 68–73, 2017.

[9] T. Q. Thanh, S. Covaci, T. Magedanz, P. Gouvas, and A. Zafeiropoulos, "Embedding security and privacy into the development and operation of cloud applications and services," *2016 17th Int. Telecommun. Netw. Strateg. Plan. Symp.*, pp. 31–36, 2016.

[10] Y. Sun, S. Nanda, and T. Jaeger, "Security-as-a-service for microservices-based cloud applications," *Proc. - IEEE 7th Int. Conf. Cloud Comput. Technol. Sci. CloudCom 2015*, pp. 50–57, 2016.

[11] Yale Yu, H. Silveira, and M. Sundaram, "A microservice based reference architecture model in the context of enterprise architecture," *2016 IEEE Adv. Inf. Manag. Commun. Electron. Autom. Control Conf.*, pp. 1856–1860, 2016.

[12] J. Rufino, M. Alam, J. Ferreira, A. Rehman, and K. F. Tsang, "Orchestration of Containerized Microservices for IIoT using Docker," pp. 1532–1536, 2017.

[13] M. B. Mollah, M. A. K. Azad, and A. Vasilakos, "Security and privacy challenges in mobile cloud computing: Survey and way ahead," *J. Netw. Comput. Appl.*, vol. 84, pp. 38–54, 2017.

[14] K. Bakshi, "Microservices-based software architecture and approaches," *IEEE Aerosp. Conf. Proc.*, 2017.

[15] J. Werner, C. M. Westphall, and C. B. Westphall, "Cloud identity management: A survey on privacy strategies," *Comput. Networks*, vol. 122, pp. 29–42, 2017.

[16] I. Khalil, A. Khreishah, and M. Azeem, "Cloud Computing Security: A Survey," *Computers*, vol. 3, no. 1, pp. 1–35, 2014.

[17] C. Saravanakumar and C. Arun, "Survey on interoperability, security, trust, privacy standardization of cloud computing," *Proc. 2014 Int. Conf. Contemp. Comput. Informatics, IC3I 2014*, pp. 977–982, 2014.

[18] C. Gadea, M. Trifan, D. Ionescu, and B. Ionescu, "A reference architecture for real-time microservice API consumption," *Proc. 3rd Work. CrossCloud Infrastructures Platforms - CrossCloud '16*, pp. 1–6, 2016.

[19] S. Srinivasan, "Data privacy concerns involving cloud," *2016 11th Int. Conf. Internet Technol. Secur. Trans. ICITST 2016*, pp. 53–56, 2017.

[20] A. Ciuffoletti, "Automated Deployment of a Microservice-based Monitoring Infrastructure," *Procedia Comput. Sci.*, vol. 68, pp. 163–172, 2015.

[21] M. Fazio, A. Celesti, R. Ranjan, C. Liu, L. Chen, and M. Villari, "Open Issues in Scheduling Microservices in the Cloud," *IEEE Cloud Comput.*, vol. 3, no. 5, pp. 81–88, 2016.

[22] F. Callegati, S. Giallorenzo, A. Melis, and M. Prandini, "Data security issues in MaaS-enabling platforms," *2016 IEEE 2nd Int. Forum Res. Technol. Soc. Ind. Leveraging a Better Tomorrow, RTSI 2016*, pp. 0–4, 2016.

[23] H. Kang, M. Le, and S. Tao, "Container and microservice driven design for cloud infrastructure DevOps," *Proc. - 2016 IEEE Int. Conf. Cloud Eng. IC2E 2016 Co-located with 1st IEEE Int. Conf. Internet-of-Things Des. Implementation, IoTDI 2016*, pp. 202–211, 2016.

[24] K. Bao, I. Mauser, S. Kochanneck, H. Xu, and H. Schmeck, "A Microservice Architecture for the Intranet of Things and Energy in Smart Buildings," *Proc. 1st Int. Work. Mashups Things APIs - MOTA '16*, pp. 1–6, 2016.

[25] D. Escobar *et al.*, "Towards the understanding and evolution of monolithic applications as microservices," *Proc. 2016 42nd Lat. Am. Comput. Conf. CLEI 2016*, 2017.

[26] J. Stubbs, W. Moreira, and R. Dooley, "Distributed Systems of Microservices Using Docker and Serfnode," *Proc. - 7th Int. Work. Sci. Gateways, IWSG 2015*, pp. 34–39, 2015.

[27] R. Roostaei and Z. Movahedi, "Mobility and Context-Aware Offloading in Mobile Cloud Computing," *Proc. - 13th IEEE Int. Conf. Ubiquitous Intell. Comput. 13th IEEE Int. Conf. Adv. Trust. Comput. 16th IEEE Int. Conf. Scalable Comput. Commun. IEEE Int.*, pp. 1144–1148, 2017.

[28] D. Jaramillo, D. V. Nguyen, and R. Smart, "Leveraging microservices architecture by using Docker technology," *Conf. Proc. - IEEE SOUTHEASTCON*, vol. 2016–July, pp. 0–4, 2016.

[29] K. El Makkaoui, A. Ezzati, A. Beni-Hssane, and C. Motamed, "Data confidentiality in the world of cloud," *J. Theor. Appl. Inf. Technol.*, vol. 84, no. 3, pp. 305–314, 2016.

[30] C. Perra and S. Member, "A Framework for the Development of Sustainable Urban Mobility Applications," 2016.

[31] M. Thangavel, P. Varalakshmi, and S. Sridhar, "An analysis of privacy preservation schemes in cloud computing," *Proc. 2nd IEEE Int. Conf. Eng. Technol. ICETECH 2016*, no. March, pp. 146–151, 2016.

[32] H. Gebre-amlak, S. Lee, A. M. A. Jabbari, Y. Chen, and B. Choi, "MIST : Mobility-Inspired SofTware-Defined Fog System," 2017.

[33] R. H. Steinegger, D. Deckers, P. Giessler, and S. Abeck, "Risk-based authenticator for web applications," *Proc. 21st Eur. Conf. Pattern Lang. Programs - Eur. '16*, no. February 2017, pp. 1–11, 2016.

[34] J. Fowler, Marthin; Lewis, "Microservices: a definition of this new architectural term," *Microservices:a definition of this new architectural term*, 2014. [Online]. Available: https://martinfowler.com/articles/microservices.ml. [Accessed: 07-May-2017].

[35] S. Yamamoto, S. Matsumoto, and M. Nakamura, "Using cloud technologies for large-scale house data in smart city," *CloudCom 2012 - Proc. 2012 4th IEEE Int. Conf. Cloud Comput. Technol. Sci.*, pp. 141–148, 2012.

[36] J. Bogner and A. Zimmermann, "Towards Integrating Microservices with Adaptable Enterprise Architecture," *Proc. - IEEE Int. Enterp. Distrib. Object Comput. Work. EDOCW*, vol. 2016–Septe, pp. 158–163, 2016.

[37] D. Guo, W. Wang, G. Zeng, and Z. Wei, "Microservices architecture based cloudware deployment platform for service computing," *Proc. - 2016 IEEE Symp. Serv. Syst. Eng. SOSE 2016*, pp. 358–364, 2016.

[38] P. Marchetta, E. Natale, A. Pescape, A. Salvi, and S. Santini, "A map-based platform for smart mobility services," *Proc. - IEEE Symp. Comput. Commun.*, vol. 2016–Febru, pp. 19–24, 2016.

[39] A. de Camargo, I. Salvadori, R. dos S. Mello, and F. Siqueira, "An architecture to automate performance tests on microservices," *Proc. 18th Int. Conf. Inf. Integr. Web-based Appl. Serv. - iiWAS '16*, pp. 422–429, 2016.

[40] P. Giessler, R. Steinegger, S. Abeck, and M. Gebhart, "Checklist for the API Design of Web Services based on REST," vol. 9, no. 3, pp. 41–51, 2016.

[41] A. Gueidi, H. Gharsellaoui, and S. Ben Ahmed, "A NoSQL-based Approach for Real-Time Managing of Embedded Data Bases," *Proc. - 2016 World Symp. Comput. Appl. Res. WSCAR 2016*, pp. 110–115, 2016.

[42] T. I. Damaiyanti, A. Imawan, and J. Kwon, "Extracting trends of traffic congestion using a NoSQL database," *Proc. - 4th IEEE Int. Conf. Big Data Cloud Comput. BDCloud 2014 with 7th IEEE Int. Conf. Soc. Comput. Networking, Soc. 2014 4th Int. Conf.*

Sustain. Comput. C, pp. 209–213, 2015.

[43] R. Simmonds, P. Watson, and J. Halliday, "Antares: A Scalable, Real-Time, Fault Tolerant Data Store for Spatial Analysis," *Proc. - 2015 IEEE World Congr. Serv. Serv. 2015*, pp. 105–112, 2015.