
Scalable Computing: Practice and Experience, ISSN 1895-1767, http://www.scpe.org
© 2021 SCPE. Volume 22, Issues 3, pp. 313–320, DOI 10.12694:/scpe.v22i3.1941

SERVICE DEPLOYMENT CHALLENGES IN CLOUD-TO-EDGE CONTINUUM

DANA PETCU*

Abstract. This position paper aims to identify the current and future challenges in application, workload or service deployment mechanisms in Cloud-to-Edge environments. We argue that the adoption of the microservices and unikernels on large scale is adding new entries on the list of requirements of a deployment mechanism, but offers an opportunity to decentralize the associated processes and improve the scalability of the applications. Moreover, the deployment in Cloud-to-Edge environment needs the support of federated machine learning.

Key words: Cloud computing, Edge computing, Resource management

AMS subject classifications. 68M14

1. Introduction. The requirements of a deployment process in current distributed environments (listed in [1], for example) are currently far from being satisfied by the available support tools. The main concerns are related to contextualization, data transfer, negotiation, deployment optimization, service manifest construction, discovery of IPs, or even filtering available IPs. Moreover, the Cloud is bringing new items on the requirement lists, like the elasticity: the deployment service should add or remove Cloud resources in a dynamical way [2]). Furthermore, recent studies on automated or semi-automated deployment processes of distributed software systems have revealed the low efficiency of the deployment of distributed software systems. In particular, this is the case when scalability, dynamics, openness, distribution or heterogeneity are primary concerns [3]. Such concerns are often encountered in Cloud computing.

The emergence and widespread adoption of Cloud computing, mobile technologies, social media and big data analytics is transforming how society operates and interacts and is heralding the so-called Industry 4.0. In Cloud computing, processing takes place within the clear boundaries on its underlying infrastructure. Cloud computing was not designed to deal with the scale of geographically dispersed, heterogeneous end points and low latency required for many Industry 4.0 use cases. Therefore, conventional paradigms of computing need to be rethought to deal with the scale of data processing needed to support the requirements of the devices from the edge of the network to work in a distributed and coordinated way at minimum latency. Fog and Edge computing are two paradigms of computing that have been proposed to address these challenges. Fog computing is seen as a horizontal, physical or virtual resource paradigm that resides between smart end-devices and traditional Cloud or data centers. Edge computing, in contrast, is local computing at the network layer encompassing the smart end-devices and their users. Fog and Edge computing provides significant advantages for processing data closer to the source and thus mitigate latency issues, lower costs of data transmission, and reduce network congestion [29]. Despite the flexibility and advantages offered by this distributed architecture, it poses substantial challenges related to resource management, infrastructure or application orchestration. These challenges are exacerbated when considering deployments spanning Cloud-to-Edge continuum (denoted by C2E).

In this position paper we try to identify the main challenges for the near future research and development activities dedicated to the application or service deployments in C2E. The paper has three parts. The first deals with the current limitations of the available deployment mechanisms. The second discusses the expected effects of adopting lightweight virtualisation techniques on the deployment mechanisms. The third part deals with the particular issues of C2E.

*West University of Timioara, Romania (dana.pecu@e-uvt.ro)

2. Solutions for Application Deployments in Clouds and their Limitations. In the first section we are studying the deployment mechanisms classified as application-oriented (according [4]), i.e. following an application-specific deployment plan that is portable as enabling infrastructure abstracting and not bounded to a specific e-infrastructure.

Automation with deployment management tools. Current extensively used configuration management tools, like Chef or Puppet, support the automation of complex deployments, even on heterogeneous systems. However, they employ proprietary domain specific languages allowing to abstractly describe the workload creation process. The problem is getting exponentially more complicated when Cloud services of different providers are expected to be simultaneously accessed (Multi-Cloud case). Multi-Cloud deployment services exposed by commercial providers, as Rightscale, support the application deployment processes across multiple Clouds through a simple web-based user interfaces as well as a RESTful API, both aiming to hide the complexity of their automation tasks. Moreover, the proof-of-the-concept prototype, autoJuJu [5], designed as an autonomic Cloud deployment manager for Ubuntu virtual machines, has shown how autonomous decisions about when to scale-in or out or to deploy various architectural components can be taken in order to improve performance. However, above enumerated solutions are complex and not designed to enable the flexibility required to choose a certain provider, despite their goal to make deployments faster and easier as well to be compatible with the Cloud service interfaces of a certain number of selected providers.

Automated deployment versus flexibility. Introducing autonomy in deployment is expected to enable the automatic elasticity in terms of number and type of the deployed application or service components, in order to meet performance requirements or energy consumption constraints. Flexible deployment service that support the specification and execution of deployment plans are now technically possible, but not yet implemented on a large scale. The application can be described through a pattern reflecting a logical view of the application together with its components mapping into the Cloud resources. Following this idea, the service described in [2], based on Chef, and accessible through a RESTful interface, is able to instantiate the pattern and update at runtime the deployment. A similar RESTful accessible service is described in [6] and unattended deployment is tested for security platform deployments.

Assisted deployment. Low-level scripts that are hands-one designed to use specific service APIs and tools are the opposite solution to the automation scripts. However, their development is a complex task requiring the deep knowledge of the services that are used, is error prone and should be repeated when the interfaces are changed. Guided writing can help the script development process to a certain extent. For example, the deployment documents proposed by Contrail [7] allows to specify how many and which kind of resources are needed to be deployed (but the resources should use the OVF format). Most of the proof-of-concept toolsets aiming to assist in the application deployment are dependent on the Cloud services of certain providers. Disnix [8], for example, is a toolset for automatic deployment in a network of machines of a distributed system; it uses declarative specifications for properties such as rollbacks, dependencies, or upgrades, but is not portable to all operating systems as being dependent on certain Unix packages. The deployment framework described in [9] and designed for analytics is based on customization workspaces and Chef. Another approach using deployment specification for component-based applications is proved to be able to automate the deployment of scientific applications in Azure Cloud[10]. High level domain specific language are in this context useful for the definition in a provider-independent manner of the deployment process of an application with its resource requirements, as well the definition for the process that transforms the definitions into deployment scripts for a large variety of services of different providers. A such domain specific language is proposed in [11]; a drawback is its incompatibility with Chef or Puppet. A Ruby-based domain specific language was also proposed [12] to automate the deployment of software frameworks, but only for AppScale platforms.

Deployment plan synthesis. Automatic synthesis of deployment plans are the done in a step before the actual deployment. A deployment plan synthesis process was proposed, for example, in[14]. Multi-objective optimization, as proposed in [13], are used often to establish a deployment plan. Another paper [15] proposed an approach for deployment mechanism based on requirements decomposition and clustering techniques. A clustering-based method for the optimal selection of Cloud nodes on which the application should be deployed can be used, as reported in [16], in the particular case of a communication-intensive application. The communication traffic between services and its performance are taken into account in the deployments reported

in [17].

Multi-Cloud deployment requirements. The modern deployment system should work together a service broker enhanced with different scheduling strategies for near-optimal deployments based on individual or multiple criteria like placement constraints, cost, energy consumption, performance, instance types, service loads [18]. The Chef cookbooks are not bound to a specific platform, but they are implemented with a specific domain specific language.

Application deployment versus their dependencies. Deploying an application is often requiring the deployment of several software components associated with that application. Chef and Puppet are requiring the creation of application-specific deployment plans that are usually not reusable when the application dependencies are changed. To mitigate this problem a middleware-oriented deployment approach can be used, according [19], to offer software components as building blocks for configurable and composable deployment plans.

Continuous deployment. The introduction of DevOps concepts in the Cloud context is justified by the need of application continuous deployment both in design and production phases. Proof-of-concept tools, such as the ones described in [20] or [21], are allowing to close the loop between the application design, deployment and execution through the intelligent interpretation of the monitoring process results. A large scale of this idea take-up has not been reached yet.

Standards and blueprints. TOSCA is the OASIS standard that is designed to describe composite Cloud applications and their management in a portable way. The structure of a composite application is described as a topology that includes components and relationships. However, the deployment procedures are not defined in this topology. Instead, deployment artifacts are specified for the actual implementation (for example, the OS type may have as artifact an image). A protocol for deploying application blueprints on the Cloud is CAMP, which uses YAML language to describe the application deployment plans. Apache Brooklyn is an example of an open-source engine able to deploy blueprints specified in TOSCA and CAMP. The use on a large scale of TOSCA, CAMP, Brooklyn is not yet registered.

Decoupling the software deployment from virtual machine deployment. Such decoupling as proposed in [22] is improving the deployment flexibility. The virtual machine size on storage is smaller as the software is not pre-stored in it. The software is streamed on demand from an application repository as soon as the user request it to run in a virtual machine. The performance tests are in favor of such approach in the case that multiple similar virtual machines are activated in a cluster of a particular Cloud. Moreover, a thrird decoupled layer, as we proposed in [6] (mOS Enabling Platform), can provide further benefits without a significant overhead: the third layer is the middleware that allows the on-demand software to be deployed is view as independent from the operating system and allowing a supplementary independence from the Cloud services.

Self-organisation. CloudLightning [23] proposes a delivery architecture based on the principles of self-organisation and management that addresses the challenges of complexity introduced by heterogeneous resources (a market like approach rather than client-server approach). This approach can be beneficial by shifting the deployment effort from the consumer to the software stack running on the e-infrastructure while reducing power consumption and improving service delivery. The overall goal is to address inefficient use of resources through over-provisioning and to deliver savings to the Cloud provider and consumer in terms of reduced power consumption and improved service delivery, with hyperscale systems particularly in mind. A proof-of-concept implementation has been recently finalized, including a simulator for large and heterrogeneous e-infrastructure services [24].

Serveless computing. In this novel approach (named also Function-as-a-Service or event-based programming), parts (modules, functions) of the application are executed when necessary without requiring the application to be running all the time. Examples of platforms suppiring the concept are AWS Lambda, OpenLambda, Google Cloud Functions or Lambda@Edge. The current challenge is to develop programming models that allow high-level abstractions to facilitate serveless computing [25]. The infrastructure is expected to be abstracted away from the user who can focus the attention to control, cost and flexibility rather than application deployment or over-or under- provisioning of resources for the application. The trade-offs in using Cloud services along with serverless computing services need further investigation.

Deployments in Micro Clouds. Small sized and low power computing processors like Raspeberry PIs co-located with switches or routers are now used to develop Micro Clouds. However, networking Micro Cloud

installations over multiple sites, integration with existing computing ecosystem, or just-in-time deployment of workloads in Micro Clouds are not yet possible.

Bare-metal services require guided deployments. The HPC-as-a-service (HPCaaS) concept makes a compromise between the common understanding of distributed Cloud services and the HPC requirements. It refers to a bare-metal compute model similar to an in-house cluster (cluster-on-demand). Several services are available from large companies like Amazon and SGI. Open source applications that are often used by the HPC community are ready to be deployed in the acquired environments. However, theirand application deployment process is not automated. The scale-up and scale-out options are rarely encountered (a counterexample is the offer of Cyclone). Self-configuration is not yet implemented and the vision of an autonomic HPC Cloud is far from being achieved [26].

3. Services Deployment Challenges following the Lightweight Virtualisation Approaches. Beyond the hypervisor approach to virtualization in which context the above described solutions were intensively tested, a new virtualization approach has gained a large community interest in the last years, the one based on containers. Furthermore, the newest approach, that of unikernels is now raising the community interest. New virtualization techniques for HPC applications like Singularity are also emerging. Additionally, smaller virtualisation overhead provided by containers and unikernels enables to run significantly more workloads than in case of using heavier virtual machines and thus results in greater energy efficiency.

All these new virtualization approaches are changing the order of magnitude of the deployed artifacts, the hardware devices that can host the services, as well as the deployment time and therefore they are generating to new challenges for the deployment mechanisms.

Automated deployment of microservices. The total time to provision a virtual machine is a burden for the current Cloud applications resulting in pre- or over-provisioning of resources. A solution to reduce the deployment time is the use of microservices. While a microservice virtual machines contains the core OS kernel, runtime, frameworks, libraries, code, configuration, and any other application's dependencies, the container eliminates the need of packaging the OS Kernel. Consequently, the microservices are quick to start, launching and being able to respond to a request in milliseconds. The usage of containers that controls the execution of microservices make them easily portable and re-deployable in different context (containers are tehrefore seen as an alternate virtualisation technology). Docker and LXC are helping the implementation of this vision. In particular, Docker offers facilities to package a service in a container, creating a lower footprint on the overall execution environment and increasing the service portability. A proof-of-concept engine that is able to deploy automatically a container-based distributed system acting as a monitoring platform was build and reported for example in [27]. Container-as-a-Service, like Google Kubernetes or Docker Swarm, offers the deployment and management of containers, allowing workload execution in ad hoc Clouds and Micro Clouds enabling Fog computing. Container monitoring, live migration and portability remains opens issues [25].

Dynamically tailored deployment engines. The approach to generate dynamically deployment engines tailored for specific application stacks was presented first in [28]. The generated engines should be portable i.e. able to run them on various e-infrastructures. APIs are expected to be generated for particular scripts and plans that perform specific jobs in an automated deployment process. Then the deployment tasks are triggered through the generated API endpoints (offering an abstraction layer detached from details of various deployment automation tools). The proposal is fitting well with the novel microservice architectures, as a tailored engine can be dynamically generated for each component of the application.

Edge computing new requirements for deployment mechanisms. Being able to build and manage on demand a hosted computing environment as close as possible to the Edge is advantageous in services that benefit from locality of data production, processing and consumption. When deploying the service several specific requirements, like an software-defined network based fabric, federation of the devices, remote installation of Edge small artifacts, are emerging. A complex task should be depicted by a configuration engine in order to select a deployment blueprint suitable for the on-line devices as well as the scripts stating what is needed to be deployed in each individual Edge node. This is a complex task and a moving target as the amount of computing power at the edge of the Internet is increasingly very rapidly, thanks to the constant deployment and churn of multitudes of heterogeneous devices (personal, embedded, or environment-situated, with ever-increasing computing and communication capacity and capabilities). Furthermore, responds to issues related

to the adaptation to an dynamic and open computing environment, like the policy for re-deployments, are still not available.

Unikernels usage. Unikernels are fixed-purpose images that run directly on a hypervisor without a guest operating systems. Unikernels run a single application, relying on the underlying hypervisor to isolate each unikernel system it is hosting. They are constructed using library operating systems having a minimal set of services that are required for an application to run. An unikernel is a virtual machine specialised and optimised for the particular application. Unikernels meet the requirement of the fast deployment as do traditional containers. However, they do not have most of the multi-process and user handling, unneeded device drivers and unnecessary system services and are able to start on bar metal. Therefore they start much faster, have a smaller storage footprint and allow an just-in-time spawning of new services. The reduced size of unikernel allow Cloud providers to increase their hardware utilisation rates, increasing margins and opening scope for reduced service development costs. However, unikernels deployment tools are still in infancy phase.

Distributed Clouds supplementary requirements. The usage of lightweight virtualisation solutions that can be deployed anywhere in the network raises another problem for the deployment mechanism. While the unikernel images or microservices can boot in microseconds the supporting software frameworks are not designed for such rapid and large-scale deployments, with the major bottleneck being the centralized framework. Moreover, assuming that Edge, Fog and Cloud resources are integrated, the deployment mechanism need to be aware of the resource configurations available at each location and on each special device. Furthermore, the small, rapid-response workloads images should be small enough to be moved quickly through the network.

Service dependency versus deployment complexity. Many current deployment frameworks rely on having declared the dependency chain for each of the services to be deployed (e.g. service X cannot be deployed until service Y is deployed). Assuming that the services can start in microseconds, the dependency can be solved instantaneous (if the service Y is not started, X requests service Y deployment and start). This offers the opportunity to have a light deployment mechanism that is built within the service that is deployed.

Compute-on-Demand. The concept refers to the creation and start of a virtual machine only when is needed. Jitsu is a prototype DNS server which is able to boot virtual machines on demand when a query is received. Today only single atomic services can be delivered in this fashion using Jitsu. However, using techniques to parallelise the creation of unikernels with the supporting network through orchestration, the same functionality for composed services is also possible, but not yet implemented.

Deployment versus configuration. Leveraging light virtualisation resources such as unikernels or containers, the boundary between service deployment and its configuration vanishes. While initial deployment is only one concern, once the service has been deployed it needs to be (re-)configured, managed and updated. Deployments of updates should be facilitated via gateway APIs allowing new code to be uploaded to a single virtual or physical node. This node can then ensures that the update is propagated to others ensuring that the easy upgrade deployments are possible.

4. Challenges in Cloud-to-Edge continuum. The discovery and provisioning of cross-layer resources in a fragmented landscape, in terms of heterogeneity of hardware, in an automated and unified manner cannot be performed seamlessly. More specifically several different frameworks need to be invoked manually, each one with a different scope, goals and algorithms, to obtain a coalition of resources that could host an end-to-end deployment. These different frameworks have diverse goals when allocating resources that in many cases result in suboptimal allocation of resources with respect to resource efficiency and energy consumption since over-provisioning is utilized to avoid Service Level Agreement violation, since application-level requirements are not considered. This procedure introduces significant obstacles from both the side of infrastructure providers as well as the side of potential users. Even mature orchestration tools such as OpenStack Heat, Kubernetes, Docker Swarm, are designed with a conventional Cloud architecture in mind and are not ready or sufficient for the scale and complexity of the C2E paradigm. Open projects trying to address handling and scheduling of microservices in a fragmented heterogeneous environment such as EdgeX Foundry Project and OpenFog Consortium have been also presented, however they are targeted primarily to IoT devices.

The C2E has been foreseen in early 2016, even in a paper of [30], at that moment under the name "CloudIoT paradigm", but identifying part of the problems encountered today. C2E challenges have been formulated in [31]. Under investigation C2E solutions referring to C2E resources are focusing today on costs [32], workflows

[33], adaptive placement [34], network [35, 36] or provisioning [37].

The majority of related work of resource management (like compute nodes, devices etc.) in Edge computing assumes that these entities are already discovered and integrated into the system. The hierarchical architecture is the most adopted solution using services and devices divided into layers. Due to this aspect the approaches that make use of this architectural model are focusing less on how the resources are discovered and integrated into the model. Some research directions are already investigating the possibility to switch computations from Cloud to Edge [41] and partially tackles the resource discovery problem in Edge environments [38, 39, 40], but without emphasizing the case were devices or resources are dynamically added to the ecosystem and how are integrated and related to proper Edge computational nodes.

Transprecise computing breaks away from the traditional notion that computing must be exact and can be applied with success in C2E environments. It proposes to perform computations at a reduced precision while reducing accuracy of the computation in a controlled way. Here, precision indicates the level of detail in the computation (e.g., the bit width of a number) while accuracy indicates the fidelity of the computed values to the exact values. Many computations inherently allow a reduction in accuracy [42, 43] or even allow approximation [44, 45].

Machine learning (ML) algorithms that apply best-effort algorithms to inherently noisy data are prime candidates for transprecise operation [51]. Operating at reduced precision can significantly improve speed of training and prediction. Moreover, hyper-parameter selection can compensate for lack of precision in machine arithmetic [46, 47, 52].

Monitoring of distributed systems is a challenging prospect, more so on a system based on the C2E, specifically in Fog computing which lies between IoT and Cloud, where any device that has computation, networking and storage capabilities can act as a processing node [48]. These types of devices have an inherent high rate of failure which can be caused by any number of factors [49, 50] like wireless connectivity issues, power failure, lack of computational availability. It is easy to see how this has a major impact on scheduling but there are also opportunities when considering failure as a special case of runtime behavior in a transprecision framework. Furthermore, monitoring appropriate metrics that are able to create the proper context entails complementing traditional system metric monitoring with specialized metrics related to identifying and reporting anomalous behavior especially for Infrastructure-as-a-Code (IaC) scenarios [48]. IaC anomalies and anti-patterns can include but are not limited to: golden image - bespoke infrastructure server image overuse; postponing secret isolation - information that should be secret but is included in the code; massive file copying - all configuration files for a particular package copied onto target by provisioning tool; data as code - data included in IaC. In some circumstances it is not desirable to send the collected monitoring data to a central location where predictive ML models can be trained. In these scenarios an adaptive federated training model [53] is preferred. These more localized models can be used to update a global predictive model via some form of global parameter aggregation. This approach minimizes both network traffic and also helps in predictive model specialization.

Acknolewdgment. This work was supported by a grant of the Romanian Ministry of Education and Research, CNCS - UEFISCDI, project number PN-III-P4-ID-PCE-2020-0407, within PNCDI III.

REFERENCES

[1] W. Li, P. Svard, J. Tordsson, E. Elmroth, A general approach to service deployment in cloud environments, in: 2nd International Conference on Cloud and Green Computing (CGC), 2012, pp. 17–24.

[2] H. Lu, M. Shtern, B. Simmons, M. Smit, M. Litoiu, Pattern-based deployment service for next generation clouds, in: IEEE 9th World Congress on Services (SERVICES), 2013, pp. 464–471.

[3] J.-P. Arcangeli, R. Boujbel, S. Leriche, Automatic deployment of distributed software systems: Definitions and state of the art, Journal of Systems and Software 103 (2015) 198 – 218.

[4] J. Wettinger, V. Andrikopoulos, S. Strauch, F. Leymann, Characterizing and evaluating different deployment approaches for cloud applications, in: IEEE International Conference on Cloud Engineering (IC2E), 2014, pp. 205–214.

[5] B. Karakostas, Towards autonomic cloud configuration and deployment environments, in: 2014 International Conference on Cloud and Autonomic Computing (ICCAC), 2014, pp. 93–96.

[6] S. Panica, D. Petcu, Unattended deployment of enabling platforms for cloud-based applications, in: 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2016, pp. 144–149.

[7] R. Cascella, L. Blasi, Y. Jegou, M. Coppola, C. Morin, Contrail: Distributed application deployment under sla in federated heterogeneous clouds, in: A. Galis, A. Gavras (Eds.), The Future Internet, Vol. 7858 of Lecture Notes in Computer Science, 2013, pp. 91–103.
[8] S. van der Burg, E. Dolstra, Disnix: A toolset for distributed deployment, Science of Computer Programming 79 (2014) 52 – 69.
[9] C. Jin, W. Wu, H. Zhang, Automating deployment of customized scientific data analytic environments on clouds, in: IEEE 4th International Conference on Big Data and Cloud Computing (BdCloud), 2014, pp. 41–48.
[10] J. Cala, P. Watson, Automatic software deployment in the azure cloud, in: F. Eliassen, R. Kapitza (Eds.), Distributed Applications and Interoperable Systems, Vol. 6115 of Lecture Notes in Computer Science, 2010, pp. 155–168.
[11] A. Thiery, T. Cerqueus, C. Thorpe, G. Sunye, J. Murphy, A dsl for deployment and testing in the cloud, in: 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2014, pp. 376–382.
[12] C. Bunch, B. Drawert, N. Chohan, C. Krintz, L. Petzold, K. Shams, Language and runtime support for automatic configuration and deployment of scientific computing software over cloud fabrics, Journal of Grid Computing 10 (1) (2012) 23–46.
[13] B. Xu, Z. Peng, F. Xiao, A. Gates, J.-P. Yu, Dynamic deployment of virtual machines in cloud computing using multi-objective optimization, Soft Computing 19 (8) (2015) 2265–2273.
[14] T. A. Lascu, J. Mauro, G. Zavattaro, Automatic deployment of component-based applications, Science of Computer Programming 113, Part 3 (2015) 261 – 284.
[15] A. Alkhalid, C.-H. Lung, S. Ajila, Towards efficient software deployment in the cloud using requirements decomposition, in: IEEE 5th International Conference on Cloud Computing Technology and Science (CloudCom), Vol. 2, 2013, pp. 100–105.
[16] P. Fan, J. Wang, Z. Zheng, M. Lyu, Toward optimal deployment of communication-intensive cloud applications, in: IEEE International Conference on Cloud Computing (CLOUD), 2011, pp. 460–467.
[17] J. Yang, C. Liu, Y. Shang, Z. Mao, J. Chen, A communication-aware deployment method for communication-intensive applications in service clouds, in: International Conference on Cloud Computing and Big Data (CloudCom-Asia), 2013, pp. 111–118.
[18] J. L. Lucas-Simarro, R. Moreno-Vozmediano, R. S. Montero, I. M. Llorente, Scheduling strategies for optimal service deployment across multiple clouds, Future Generation Computer Systems 29 (6) (2013) 1431 – 1441.
[19] J. Wettinger, V. Andrikopoulos, S. Strauch, F. Leymann, Enabling dynamic deployment of cloud applications using a modular and extensible paas environment, in: IEEE 6th International Conference on Cloud Computing (CLOUD), 2013, pp. 478–485.
[20] E. Di Nitto, G. Casale, D. Petcu, On modaclouds' toolkit support for devops, in: 4th European Conference on Service Oriented and Cloud Computing Workshops (ESOCC), 2016, pp. 430–431.
[21] M. Artac, T. Borovšak, E. Di Nitto, M. Guerriero, D.A. Tamburri, Model-driven continuous deployment for quality DevOps, in: 2nd International Workshop on Quality-Aware DevOps (QUDOS), 2016, pp. 40–41.
[22] Y. Zhang, Y. Li, W. Zheng, Automatic software deployment using user-level virtualization for cloud-computing, Future Generation Computer Systems 29 (1) (2013) 323 – 329.
[23] T. Lynn, H. Xiong, D. Dong, et al, Cloudlightning: A framework for a self-organising and self-managing heterogeneous cloud, in: 6th International Conference on Cloud Computing and Services Science (CLOSER 2016), 2016, pp. 333–338.
[24] C.K. Filelis-Papadopoulos, K.M. Giannoutakis, G.A. Gravvanis, D. Tzovaras, Large-scale simulation of a self-organizing self-management cloud computing framework, The Journal of Supercomputing 74 (2) (2018) 530 – 550.
[25] B. Varghese, R. Buyya, Next generation cloud computing: New trends and research directions, Future Generation Computer Systems 79 (3) (2018) 849 – 861.
[26] D. Petcu, On autonomic hpc clouds, in: 2nd Int.Workshop Sustainable Ultrascale Computing Systems (NESUS 2015), 2015, pp. 29–40.
[27] A. Ciuffoletti, Automated deployment of a microservice-based monitoring infrastructure, Procedia Computer Science 68 (2015) 163 – 172, 1st International Conference on Cloud Forward: From Distributed to Complete Computing.
[28] J. Wettinger, U. Breitenbucher, F. Leymann, Dyn tail - dynamically tailored deployment engines for cloud applications, in: IEEE 8th International Conference on Cloud Computing (CLOUD), 2015, pp. 421–428.
[29] S. Svorobej, P.E.Takako, M.Bendechache, C.Filelis-Papadopoulos, K.M.Giannoutakis, G.A.Gravvanis, D.Tzovaras, J.Byrne, T.Lynn, Simulating Fog and Edge Computing Scenarios: An Overview and Research Challenges. Future Internet 11, 55, 2019.
[30] D. Petcu, M. Fazio, R. Prodan, Z. Zhao and M. Rak, On the Next Generations of Infrastructure-as-a-Services, In 6th International Conference on Cloud Computing and Services Science (CLOSER), 320-326, 2016.
[31] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, and O. Rana, The Internet of Things, Fog and Cloud continuum: Integration and challenges, Internet of Things, vols. 3–4, 134-155, 2018.
[32] C. Li, H. Sun, Y. Chen, and Y. Luo, Edge cloud resource expansion and shrinkage based on workload for minimizing the cost, Future Generation Computer Systems, Vol.101, 327-340, 2019.
[33] D. Balouek-Thomert, E. G. Renart, A. R. Zamani, A. Simonet, and M. Parashar. Towards a computing continuum: Enabling edge-to-cloud integration for data-driven workflows. The International Journal of High Performance Computing Applications, vol. 33, issue 6, 1159–1174, 2019.
[34] Y. Xu, J. Li, Z. Lu, J. Wu, P. C.K. Hung, A. Alelaiwi, ARVMEC: Adaptive Recommendation of Virtual Machines for IoT in Edge–Cloud Environment, Journal of Parallel and Distributed Computing, Vol. 141, 23-34, 2020.

[35] A. Carrega and M. Repetto, A network-centric architecture for building the cloud continuum, In 2017 International Conference on Computing, Networking and Communications (ICNC), pp. 701-705, IEEE, 2017.
[36] E. Qafzezi, K. Bylykbashi, M. Ikeda, K. Matsuo, and L. Barolli, Coordination and management of cloud, fog and edge resources in SDN-VANETs using fuzzy logic: A comparison study for two fuzzy-based systems, Internet of Things, Vol. 11, 100169, 2020.
[37] J. Guo, C. Li, Y. Chen, and Y. Luo, On-demand resource provision based on load estimation and service expenditure in edge cloud environment, Journal of Network and Computer Applications, Vol. 151, 102506, 2020.
[38] N. Tomar and R. Matam. Optimal query-processing-node discovery in iot-fog computing environment. In 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 237–241, IEEE, 2018.
[39] R. Venanzi, B. Kantarci, L. Foschini, and P. Bellavista. MQTT-driven node discovery for integrated IoT-fog settings revisited: The impact of advertiser dynamicity. In Symposium on Service-Oriented System Engineering (SOSE), 31–39, IEEE, 2018.
[40] N. Desai and S. Punnekkat. Safety of fog-based industrial automation systems. In Workshop on Fog Computing and the IoT (IoT-Fog), 6–10, ACM, 2019.
[41] R. Kolcun, D. Boyle and J. A McCann. Optimal processing node discovery algorithm for distributed computing in IoT. In 2015 5th International Conference on the Internet of Things (IOT), pages 72–79, IEEE, 2015.
[42] A. C. I. Malossi et al, The transprecision computing paradigm: Concept, design, and applications. In 2018 Design, Automation Test in Europe Conference Exhibition (DATE), 1105-1110, 2018.
[43] J. Lee, H. Vandierendonck, M. Arif, G. D. Peterson and D. S. Nikolopoulos, Energy- Efficient Iterative Refinement Using Dynamic Precision, in IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 8, no. 4, 722-735, 2018.
[44] G. L. Steele, Jr. and J.-B. Tristan, Adding approximate counters. In 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP). ACM, Article 15, 2016.
[45] S. Mittal. A Survey of Techniques for Approximate Computing. ACM Comput. Surv. 48, 4, Article 62, 2016.
[46] M. Blott, T. B. Preusser, N. J. Fraser, G. Gambardella, K. O'Brien, Y. Umuroglu, M. Leeser, and K. Vissers. 2018. FINN-R: An End-to-End Deep-Learning Framework for Fast Exploration of Quantized Neural Networks. ACM Trans. Reconfigurable Technol. Syst. 11, 3, Article 16, 2018.
[47] J. Lee, H. Vandierendonck and D. S. Nikolopoulos, Adaptive Mixed Precision Kernel Recursive Least Squares. In Adaptive Many-Core Architectures and Systems Workshop, 2018.
[48] R. K. Naha et al, Fog Computing: Survey of Trends, Architectures, Requirements, and Research Directions, IEEE Access, vol. 6, 47980-48009, 2018.
[49] Y. Wang, T. Uehara, and R. Sasaki, Fog computing: Issues and challenges in security and forensics, in IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), vol. 3, 53–59, IEEE, 2015.
[50] S. Yi, Z. Hao, Z. Qin, and Q. Li, Fog computing: Platform and applications, in 3rd IEEE Workshop on Hot Topics in Web Systems and Technologies (HotWeb), 73–78, IEEE, 2015
[51] A. Borghesi, F. Baldo, M. Lombardi, M. Milano, Injective Domain Knowledge in Neural Networks for Transprecision Computing. CoRR abs/2002.10214, 2020.
[52] A. Borghesi, G. Tagliavini, M. Lombardi, L. Benini, M. Milano, Combining learning and optimization for transprecision computing. 17th ACM International Conference on Computing Frontiers (CF), 10-18, 2020.
[53] S. Wang et al., Adaptive Federated Learning in Resource Constrained Edge Computing Systems, IEEE Journal on Selected Areas in Communications, vol. 37, no. 6, 1205-1221, 2019.