CHAPTER 4

Orchestration from the Cloud to the Edge

Sergej Svorobej, Malika Bendechache, Frank Griesinger, and Jörg Domaschka

Abstract The effective management of complex and heterogeneous computing environments is one of the biggest challenges that service and infrastructure providers are facing in the cloud-to-thing continuum era. Advanced orchestration systems are required to support the resource management of large-scale cloud data centres integrated with the big data generation of IoT devices. The orchestration system should be aware about all available resources and their current status in order to perform dynamic allocations and enable short time deployment of applications. This chapter will review the state of the art with regards to orchestration along the cloud-to-thing continuum with a specific emphasis on container-based orchestration (e.g. Docker Swarm and Kubernetes) and fog-specific orchestration architectures (e.g. SORTS, SOAFI, ETSI IGS MEC, and CONCERT).

S. Svorobej (✉) • M. Bendechache
Irish Institute of Digital Business, Dublin City University, Dublin, Ireland
e-mail: sergej.svorobej@dcu.ie; malika.bendechache@dcu.ie

F. Griesinger • J. Domaschka
Institute for Organisation and Management of Information Systems (OMI), University of Ulm, Ulm, Germany
e-mail: frank.griesinger@uni-ulm.de; joerg.domaschka@uni-ulm.de

T. Lynn et al. (eds.), *The Cloud-to-Thing Continuum*, Palgrave Studies in Digital Business & Enabling Technologies,
https://doi.org/10.1007/978-3-030-41110-7_4

Keywords Cloud computing • Edge computing • Fog computing • Orchestration • Management • Container orchestration • Orchestration tools • Orchestration standards • Orchestration challenges • Orchestration architectures

4.1 Introduction

The inarguable success of cloud computing combined with rapid growth in adoption of Internet services is resulting in an unprecedented demand for computing resources. However, cloud computing performance for many applications depends closely on the network latency. In particular, the strength of network connectivity is crucial for large data sets. As more and more data is generated by enterprises and consumers, particularly with the adoption of the Internet of Things (IoT), traditional cloud connectivity may not be sufficient (Carnevale et al. 2018). To make up for the lack of speed and connectivity with conventional cloud computing, processing for mission-critical applications will need to occur closer to the data source. Processing the data close to where it originated is referred to as *edge computing* and *fog computing*.

Edge computing is pushing computing applications, data, and services away from centralised cloud data centre architectures to the edges of the underlying network (Barika et al. 2019). It is defined by NIST (Iorga et al. 2018) "*as a local computing at the network layer encompassing the smart end-devices and their users. It runs specific applications in a fixed logic location and provides a direct transmission service.*" It promises to reduce the amount of data pushed to centralised cloud data centres avoiding load on the network and therefore is beneficial for analytics and knowledge-based services. Edge computing also leads to lower latencies, hence increasing communication velocity, reducing wider network footprints and avoiding congestion. As it reduces the distance the data must travel, it boosts the performance and reliability of latency-critical applications and services.

Service orchestration is an arrangement of auxiliary system components that cloud providers can use for coordination and management of computing resources to ensure service provision to cloud consumers (Bohn et al. 2011). Orchestration can also be defined as the use of programming technology to manage the interconnections and interactions among

workloads on distributed edge-cloud infrastructure (Mahmoudi et al. 2018). This is accomplished through three main attributes of orchestration, which are closely related: service orchestration, workload orchestration, and resource orchestration. An orchestration platform usually integrates permission, checks for security, and compliance (Ranjan et al. 2015). Orchestration may also integrate components from various domains, for example provide connections between network-deployed components and fixed applications and resources. For some applications, the integration of virtualised components with the data centre is the only needed orchestration type.

Cloud-to-Edge orchestration is a crucial feature for many IT organisations and DevOps adopters as a way to speed the delivery of services, simplify optimisation, and reduce costs (Nygren et al. 2010). A cloud orchestrator automates the management, coordination, and organisation of distributed computer systems, services, and middleware. In addition to reduced personnel involvement, orchestration eliminates the potential for errors introduced into provisioning, scaling, or other cloud processes. Orchestration supports the delivery of cloud resources to customers and end users, including in a self-service model where users request resources without IT's involvement (Carnevale et al. 2018).

Major cloud providers, such as Microsoft and Google, as well as third-party vendors, provide tools for orchestration as part of their services (e.g. AWS Cloud Formation, Google Cloud Composer, Azure Automation). With orchestration, the overall goal is to ensure successful hosting and delivery of applications. Currently provided functionality is still lacking focus on Quality of Service (QoS) requirements, however meeting the QoS objectives of users will gain further importance in the future. Examples of QoS functional and non-functional attributes includes performance statistics, consistency, security, integrity, reliability, renting cost, scalability, availability, legal, and regulatory concerns (Pahl et al. 2019).

The rest of the chapter is organised as follows. The next section provides a summary overview of day to day challenges in Cloud-to-Edge orchestration. Next, we outline current industry standards for orchestration architectures and orchestration tools within their respective subsections. Finally, we conclude with some closing remarks on the topic.

4.2 Orchestration Challenges

The orchestration of virtualised environments is challenging due to the scale, heterogeneity, and diversity of resource types and the uncertainties of the underlying cloud environment. The uncertainties arise from a number of factors including resource capacity demand (e.g. bandwidth and memory), failures (e.g. failure of a network link), user access pattern (e.g. number of users and location) and lifecycle activities of applications. In particular, cloud resource orchestration is challenging because applications are composed of multiple, heterogeneous software and hardware resources, which may have integration and interoperation dependencies (Barika et al. 2019).

Orchestration along the Cloud-to-Edge continuum adds another layer of complexity and challenges. In the cloud-to-thing era, applications as well as storage are geo-distributed. Therefore, applications will need to be restructured to distribute logic across the network. Storage will likewise need to be decentralised. This creates new issues of reliability and data integrity that are inherent in broadly decentralised networks. Cloud servers become control nodes for intelligent edge devices, performing summary analytics while leaving real-time decision making to edge servers (Jiang et al. 2018). Therefore, there is a need for comprehensive orchestration techniques that can coordinate and schedule network services simultaneously through different technologies across the Cloud-to-Edge network (Vaquero et al. 2019). Table 4.1 summarises the emerging orchestration needs in the edge/fog computing technologies and the corresponding requirements for each need.

In order to orchestrate distributed system as cloud-to-thing computing, new architecture needs to be defined taking into account the above edge orchestration needs and requirements.

The orchestration and management of a cloud-to-thing architecture is mostly realised through virtualisation. As discussed in Chap. 2, the evolution of virtualisation has moved away from virtual machines towards more lightweight solutions such as containers. This is specifically relevant for application packaging at a software platform and application level. Different application packages such as containers have been proposed to cluster Cloud-to-Edge and solutions such as Docker container and Kubernetes architectures. Yet, there is still a need for a topology specification and a derived orchestration plan for cloud edge computing.

Table 4.1 Emerging orchestration needs in edge/fog computing. (Adapted from Vaquero et al. (2019))

Functional orchestration needs	*Requirements per need*
Dynamic coalitions of edge devices and cloudlets	• Locality-awareness • Dynamism • Churn • Scalability • Replacement • Recovery
Going beyond shadow devices for reliability.	• Device churn
Dynamic end-to-end service availability.	• Multi-tenant • Multi-domain
Smaller execution units, smaller state	• Larger scale • Finer grain
Diversity	• Heterogeneity
M2M confidentiality, wireless-based attacks, trust management	• Security
AAA, privacy-leakage	• Privacy
Ensure quality-of-service on a variety of infrastructure elements	• Heterogeneity • Multi-domain

4.3 Industry Standards

4.3.1 Network Function Virtualisation

Network Function Virtualisation (NFV) is a constantly evolving paradigm which enables the virtualisation of chains of communication services thus replacing purpose-built hardware appliances. With the increase in network traffic diversity and capacity growth in 5G NFV concept offers greater degree of flexibility for network, cloud, and mobile service providers (Barakabitze et al. 2019). The benefits of virtualisation include scalability, elasticity, and cost savings to the service; however the management of NFV chains becomes a challenge. The European Telecommunications Standards Institute (ETSI) Industry Specification Group for NFV (ETSI ISG NFV) has proposed an Open Source NFV Management and Orchestration (MANO) framework which provides NFV operators with the standard tools and framework for NFV orchestration (ETSI 2019). The NFV-MANO architecture is defined by three main functional blocks (ETSI 2014):

- VNF Manager (VNFM)
- VNF Orchestrator (VNFO)
- Virtualised Infrastructure Manager (VIM)

The VNFM is responsible for the lifecycle management of the VNF instances such as image template instantiation, software upgrades, scaling, and instance termination. The VNFO is responsible for orchestrating numerous VIMs to fulfil more complex function objectives across multiple VNF groups. Finally, VIM is an interface for a single infrastructure domain that is responsible for control and management of resources such as computation, storage, and network at that particular location. The latest implementation of Open Source Mano (OSM) Release 6 deploys the framework as a cohort of configurable Docker containers which provide VNF management capabilities and can integrate with multiple VIMs using plugins.

In a bid to bring unity to the NFV environment, the Open Platform for NFV (OPNFV) was launched through Linux Foundation (OPNFV 2019). The OPNFV project goal is to establish an ecosystem for NFV solutions that integrates through joint collaboration of development and testing. The OPNFV is a midstream project that drives new features based on the upstream user feedback, and also ensures component continuous integration downstream through composition deployment and testing (Brockners 2016).

4.3.2 OpenFog Reference Architecture

To standardise and promote the use of the fog computing paradigm across various disciplines the OpenFog consortium[1] was founded by the industry and academia in the telecommunication field. The OpenFog consortium working group created the OpenFog Reference Architecture (RA) for fog computing which was adopted by the IEEE Standards Association (OpenFog Consortium 2018). The reference architecture provides an overview of fog opportunity areas, use cases, and introduces eight pillars of OpenFog RA:

- Security—trust, attestation, privacy
- Scalability—localised command control and processing, orchestration and analytics, avoidance of network taxes

[1] Merged with Industrial Internet Consortium in January 2019

- Openness—resource visibility and control, white box decision making, interop and data normalisation
- Autonomy—flexible, cognition and agility, value of data
- Programmability—programmable SW/HW, virtualisation and multi-tenant, app fluidity
- RAS—Reliability, Availability, and Serviceability
- Agility—tactical and strategic decision making, data to wisdom
- Hierarchy—fully cloud enabled, computational and system, autonomy at all levels

The pillars provide guidance and describe requirements for hardware manufacturers, software developers, system vendors, and other parties in the fog supply chain. This view aligns well with the ISO/IEC CD 30141 that defines an Internet of Things RA (International Organization for Standardization 2018). It points out several necessary capabilities of IoT systems including the realisation of automated network management; ensuring of maintainability over long periods of time and large geographical region, including the need for configuration changes; reliability, and resilience of the system; and the need for availability and therefore scalability. The realisation of all of these capabilities requires a huge degree of automation and hence, are well-suited for the use of an orchestrator.

4.3.3 Orchestration Architectures

Multiple resource orchestration and provisioning architectures were developed to take advantage of Cloud-to-Edge infrastructure and its features. Munoz et al. (2015) present a management and orchestration architecture based on Software Defined Networking (SDN) and NFV. This architecture allows dynamic deployment of virtual tenant networks (VTN) and required corresponding SDN controllers in distributed data centre network as NFVs. The proposed solution is compatible with NFV MANO and consists of the following main functional blocks: *Multidomain SDN Orchestrator, Multidomain Network Hypervisor, Intra-DC Cloud and Network Orchestrator* and *Global Cloud and Network Orchestrator*.

The *Multidomain SDN Orchestrator* mechanism acts as a "controller of the controllers" of end-to-end provisioning services using Control Orchestration Protocol (COP). It orchestrates services across heterogeneous network layer components at a higher abstraction level thus supporting multiple lower level technologies. The *Multidomain Network*

Hypervisor aggregates and partitions physical network resources into virtual resources forming multiple connections among VTNs. The network hypervisor can dynamically create, change, and delete network resources based on matrix of QoS requirements. The *Intra-DC Cloud and Network Orchestrator* is responsible for VM lifecycle management, that is creation, migration, and deletion within a data centre. In a distributed data centre network, there is a need for an integrated orchestration. The *Global Cloud and Network Orchestrator* architecture component is responsible for global network and resource provisioning. It ensures VM migration and end-to-end connectivity links setup between distributed data centre site locations. The Integrated SDN/NFV Management and Orchestration Architecture was validated by an implementation that was deployed across three data centres in Spain and Poland.

Yannuzzi et al. (2017) propose a novel converged architecture called the Digital IoT Fabric, that complies with both OpenFog and MANO standards. The design of the Digital IoT Fabric aims to deliver a uniform management to NFV and IoT services with the deployment options from cloud to edge. The architecture is logically separated into four components:

1. the sensors, actuators, and control layer;
2. the system view of hardware resources and software view of virtualisation layer components;
3. five perspectives that comprise of platform capabilities i.e. manageability, security, performance and scale, data analytics and control, IT business and cross-fog applications; and
4. the User Interface (UI) and cloud and OpenFog services layer.

The logical connection between OpenFog and VNF MANO is achieved through the link between the OpenFog Node Management component and the MANO VIM component, both of which manage virtual functions and virtual infrastructures. Yannuzzi et al. (2017) argue that such architecture allows automated orchestration across the Edge-to-Cloud continuum and can play a key role in merging of operational technology and information technology.

SmartFog is another novel fog architecture which was designed to resemble human brain functions, where fog devices and network communication channels are analogous to neurons and synapses (Kimovski et al. 2018). This nature-inspired fog architecture makes use of graph theory, machine learning, and multi-criteria decision making to make fast

decisions and architecture structuring. The architecture enables self-clustering of fog devices based on functional areas, further extending parallels with nature, for example temperature sensors forming a group of thermoreceptors or camera sensors forming a group of photoreceptors. The proposed architectural model can be logically divided into three distinctive layers: cloud layer, fog layer, and IoT layer. The cloud layer is the top layer where IoT application components are deployed and governed by functional requirements. The fog layer is the intermediary tier between the cloud and IoT layers, where the SmartFog architecture evolves around. SmartFog manages fog layer resources available within fog devices to create data transmission and processing paths through establishing communication gateways and assigning resources needed to host IoT application components and temporary storage blocks. The spectral clustering approach is applied to the lower IoT layer to classify and group fog devices based on their functional resemblance. Such groups are then connected to cloud applications in the upper cloud layer through dynamic communication gateways in the intermediary tier of the fog layer. The SmartFog architecture concept was validated via simulation only and as such, remains only a theoretical contribution.

Velasquez et al. (2017) recognise the difference between cloud computing and fog computing requirements and propose the Supporting the Orchestration of Resilient and Trustworthy Fog Services (SORTS) framework which introduces new mechanisms for services and resources orchestration specifically in fog environment. SORTS aims to maintain acceptable levels of QoS through ensuring resilience, trustworthiness and low latency within the dynamicity of a fog environment. The framework proposes a hybrid approach by using service orchestration and choreography management approaches. The orchestration is defined as a centralised management mechanism for cloud level resource management in the upper tier of the architecture. While the choreography mechanism is dedicated to the lower architecture tier covering management of IoT device virtual clusters and fog instances. Such operational decoupling in management levels allows quicker reaction to the changes to virtual clusters without intervention of higher level service management.

A Service Orchestration Architecture for Fog-enabled Infrastructures (SOAFI) is proposed by de Brito et al. (2017) which is based on the core requirements of fog computing focusing on heterogeneity and dynamics of IoT devices. Authors of SOAFI consider every exposed computer interface as a resource, and therefore in control by a resource manager: resource

examples include microservices, sensors CPU, memory, network, VMs, accelerators. The framework itself is split in two tiers *Fog Orchestrator* (FO) and *Fog Agent (FA)*. The FO manages the infrastructure of connected fog nodes; it keeps a database of available resources through a built discovery service. The FA is running on a fog node and provides monitoring and local access to resource management through the interface to the FO. The authors were successful in implementing and initial working prototype of SOAFI which was deployed in their IoT testbed.

A Cloud-Based Architecture for Next-Generation Cellular Systems, named CONCERT is proposed by Jingchu Liu et al. (2015). As the name suggests the architecture is targeted for management of cellular edge infrastructure embracing NFV services. The CONCERT approach is based on the concept of control and data plane decoupling where data plane embodies physical resources of edge infrastructure and the control plane coordinates physical resources through virtualisation. In addition, CONCERT allows physical resource placement and task scheduling in a bit for better service orchestration. The control plane entity, called *Conductor*, is at the centre of the proposed architecture design. It orchestrates and virtualises data plane resources as well as controlling software defined switches through centralised packet forwarding tables. This way the *Conductor* manages physical data plane resources as a central entity by provisioning them to a required VNF.

In an effort to bridge the gap between theory and practice, Santos et al. (2017) propose a container-based fog computing orchestration architecture. The proposed architecture was implemented using Kubernetes, an open source management solution for containerised applications, which was extended with network-aware scheduling (NAS) (Santos et al. 2019) and Integer Linear Programming (ILP) decision support for IoT service placement (Santos et al. 2017). The network-aware scheduling makes resource provisioning decisions by taking in consideration current load status of available network infrastructure and the target location of service. The ILP ensures close placement proximity of IoT application services to the end devices which use these services. The smart city scenario-based experiments show a 70% network latency reduction compared to the default Kubernetes scheduling setup with 1.22 ms scheduling decision time overhead.

4.3.4 Orchestration Tools

Industry standard and proposed orchestration architectures define high level system design best practices for multiple integrated functional components. However, in order to use any of the system design features in real world scenarios, an actual implementation has to take place. A wide range of resource management tools are available to orchestrate Cloud-to-Edge infrastructure which are outlined below.

Since edge site resources are considered to be limited due to constraints in physical hosting space, we focus primarily on container-supporting tools as containers have leaner resource overhead profiles when compared to virtual machines. A container is technology that provides lightweight virtualisation at the kernel level. It is a packaged, self-contained, ready-to-deploy set of parts of applications, that might include middleware and business logic in the form of binaries and libraries to run the applications (Pahl and Lee 2015). Containers address concerns at the cloud PaaS level allowing to spawn self-contained applications on demand. Containers are often called building blocks of PaaS due to flexibility to be spawned on both physical and virtual infrastructures. Containers also relate to the IaaS level through sharing and isolation aspects that exemplify the evolution of OS and virtualisation technology. Docker is one of the most popular container tools for the Linux and Windows operating system with about 83% of market share followed by CoreOS rkt (12%), Mesos Containeriser (4%) and Li Linux Containers (LXC) (1%) (Carter 2018). Dockers are frameworks built around container engines (Turnbull 2014). They make containers a portable way to package applications to run in containers. The Open Container Initiative[2] is making a push to create de-facto standards for container runtime and image formats. In terms of a tiered application, a tier can be represented by a single container or a number of containers depending on application design and requirements.

Kubernetes is a popular open-source platform for managing containers and their hosted services. Kubernetes was initially developed by Google and open sourced in 2014; it is maintained by the Cloud Native Computing Foundation. Kubernetes is a modular platform that focuses on automation of container management tasks such as service discovery and load balancing, storage orchestration, deployment roll outs and rollbacks, container bin packing, self-healing, secret and configuration management. The

[2] https://www.opencontainers.org/

architecture of Kubernetes is divided into three distinct component areas—Master Components, Node Components, and Add-ons. The Master Components form the control plane of the cluster making global decisions on scheduling, backup and ensuring node pod deployments to hardware nodes. The Node Components form and maintain usable Kubernetes environment on a hardware node. These components are deployed on each individual node in the data centre that are zoned to be used for container hosting providing network proxy features, healthy container state and enable container runtime features. Add-ons are an optional component group that complements the Master and Node Components by providing additional DNS, web UI and resource monitoring features (Kubernetes 2019). Kubernetes is used as a base platform for Red Hat OpenShift[3] and Rancher,[4] which provide additional features for Kubernetes cluster management, resource provisioning, monitoring and security. Recently, Rancher released K3s specifically tailored towards low-end infrastructure such as IoT gateways and extreme edge devices.

The Nebula Container Orchestrator[5] is an open-source project designed to manage large-scale clusters of Docker containers. The solution is specifically targeted at large-scale scenarios such as IoT devices or virtual Content Delivery Networks (vCDN) running Docker containers. The Nebula Container Orchestrator provides a REST API that can be used for sending management instructions to the deployed container groups such as rolling out updates, mounting volumes, changing images, monitoring health and performance and adjusting resource allocation. The architecture consists of two core components, *Manager* and *Worker*, and an optional monitoring component, *Reporter*. First the IoT device connects to the *manager* to retrieve group configuration information, and after configuration obtained the *worker* is handling device further. All of the architecture components are using a single scalable backend database (i.e. MongoDB[6]) to store configuration states and monitoring data. *Manager* is a fully stateless component that serves as an API endpoint to control the system. *Worker* is running on the remote container and manages the *Worker* by periodically pulling instructions from manager components. The *Reporter* component is used for collecting the data from the

[3] https://www.openshift.com/
[4] https://rancher.com/
[5] https://nebula-orchestrator.github.io/
[6] https://www.mongodb.com/

individual containers in the group to provide monitoring data to the system administrator. The Nebula Container Orchestrator is designed with scale in mind, and each component can be scaled to meet the demands of the system. Stress test results suggest a linear increase in number of IoT devices a single *Manager* component can handle from 7780 devices checking manager every 10 seconds to 466,800 devices checking every 600 seconds (Nebula 2019). Since multiple *Manager* components can be dynamically deployed the orchestrator provides a flexible solution for large-scale containerised service deployments.

Swarm is an open-source native container orchestration engine build for the Docker platform. The Docker integration allows Docker Engine CLI to be used directly to issue Swarm commands providing streamlined Docker container cluster management experience. Since the Swarm mode is already a part of the Docker engine, no other additional orchestration tools are needed when working with Docker-based containers. The Swarm architecture consists of manager nodes, distributed state store, and worker nodes (Docker Inc. 2019a). The manager nodes are responsible for maintaining cluster state, schedule services and provide access to the functionality over web API endpoints. It is recommended to run multiple managers as a safeguard against failures in order to maintain consistency of the entire swarm. The manager nodes use Raft (Ongaro and Ousterhout 2014) consensus algorithm for managing replicated logs via the internal distributed state store where each manager is connected. The worker nodes' sole purpose is to execute containers. Worker nodes do not use distributed state storage and don't provide services of manager nodes; however, a worker can be promoted to a manager with a single "*promote*" command as they are also instances of Docker Engine. This functionality is useful for node maintenance and failure recovery scenarios. Swarm includes features as incremental node updates, TLS-based authentication and traffic encryption, internal load balancing specification, and an API to connect external load balancers with support for the overlay networking and scaling (Docker Inc. 2019b).

4.4 Conclusion

The last decade has brought a rapid emergence of smart devices which encouraged the development of cloud computing, hardware, networks, and mobility. Both the enterprise and consumer landscape are seeing an increase in these device numbers as the value of rapid information access is being realised in day-to-day scenarios. The devices are used in geographically remote locations where access to Internet connection as well as the

remote cloud services, is not stable, hence the need to process generated data locally. This introduces an additional unique layer of heterogeneity with physical form factor variability as well as unique network data transfer capability. Meanwhile the increasing consumer and enterprise service demand is creating significant strain on the telecommunication compute and network infrastructure. Hardware heterogeneity, scalability and latency are some of the main challenges that Cloud-to-Edge infrastructure providers are facing on a day-to-day basis in order to uphold QoS that are expected by customers.

Orchestrators on the other hand have emerged together with cloud computing and provide a mature approach to coordinate the automated managing tasks for distributed applications running on IaaS or container-based environments. The resource orchestration approach and tools stack play an important role in distributed service delivery. Considering that edge and fog applications need to deal with more dynamic and less predictable environments, their operators are even more dependent on reliable and efficient orchestrators that need to handle the new challenges: the use of geo-distributed infrastructure demands for more detailed understanding of application behaviour; support for federation, as there is a high chance that edge environments will span multiple providers.

There is currently a strong movement to establish cloud and fog computing as business models and a movement towards fog orchestrators. Also, multiple active standardisation initiatives exist. Nevertheless, this chapter showed that the current state of the art in Cloud-to-Edge orchestration does not address all challenges and that more work in research and standardisation needs to be done. Just as the paradigm of cloud-native applications has given momentum to the development of cloud orchestrators, establishing a commonly accepted definition of fog-native applications might accelerate the evolvement of fog orchestrators.

References

Barakabitze, Alcardo Alex, Lingfen Sun, Is-Haka Mkwawa, and Emmanuel Ifeachor. 2019. A Novel QoE-Aware SDN-Enabled, NFV-based Management Architecture for Future Multimedia Applications on 5G Systems. *arXiv preprint arXiv:1904.09917*, April. http://arxiv.org/abs/1904.09917.

Barika, Mutaz, Saurabh Garg, Albert Y. Zomaya, Lizhe Wang, A. van Moorsel, and Rajiv Ranjan. 2019. Orchestrating Big Data Analysis Workflows in the Cloud: Research Challenges, Survey, and Future Directions. *ACM Computing Surveys* 52: 1–37.

Bohn, Robert B., John Messina, Fang Liu, Jin Tong, and Jian Mao. 2011. *NIST Cloud Computing Reference Architecture*. 2011 IEEE World Congress on Services, 594–596. IEEE. https://doi.org/10.1109/SERVICES.2011.105.

Brito, Mathias Santos de, Saiful Hoque, Thomas Magedanz, Ronald Steinke, Alexander Willner, Daniel Nehls, Oliver Keils, and Florian Schreiner. 2017. *A Service Orchestration Architecture for Fog-Enabled Infrastructures*. 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), 127–32. IEEE. https://doi.org/10.1109/FMEC.2017.7946419.

Brockners, Frank. 2016. What is OPNFV? OPNFV Summit.

Carnevale, Lorenzo, Antonio Celesti, Antonino Galletta, Schahram Dustdar, and Massimo Villari. 2018. *From the Cloud to Edge and IoT: A Smart Orchestration Architecture for Enabling Osmotic Computing*. 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2018-January, 419–424. IEEE. https://doi.org/10.1109/WAINA.2018.00122.

Carter, Eric. 2018. Docker Usage Report. https://sysdig.com/blog/2018-docker-usage-report/.

Docker Inc. 2019a. How Nodes Work | Docker Documentation. https://docs.docker.com/engine/swarm/how-swarm-mode-works/nodes/.

———. 2019b. Swarm Mode Overview | Docker Documentation. https://docs.docker.com/engine/swarm/.

ETSI. 2014. Network Functions Virtualisation (NFV); Management and Orchestration. *Etsi*, vol. 2. http://www.embase.com/search/results?subaction=viewrecord&from=export&id=L34467660.

———. 2019. ETSI—Open Source Mano | Open Source Solutions | Mano NFV. https://www.etsi.org/technologies/nfv/open-source-mano.

International Organization for Standardization. 2018. *Information Technology—Internet of Things Reference Architecture (IoT RA)*. ISO/IEC130141:20182018, Geneva. https://www.iso.org/standard/65695.html.

Iorga, Michaela, Larry Feldman, Robert Barton, Michael J. Martin, Nedim S. Goren, and Charif Mahmoudi. 2018. Fog Computing Conceptual Model.

Jiang, Yuxuan, Zhe Huang, and Danny H.K. Tsang. 2018. Challenges and Solutions in Fog Computing Orchestration. *IEEE Network* 32 (3): 122–129. https://doi.org/10.1109/MNET.2017.1700271.

Kimovski, Dragi, Humaira Ijaz, Nishant Saurabh, and Radu Prodan. 2018. *Adaptive Nature-Inspired Fog Architecture*. 2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC), 1–8. IEEE. https://doi.org/10.1109/CFEC.2018.8358723.

Kubernetes. 2019. Kubernetes. https://kubernetes.io.

Liu, Jingchu, Tao Zhao, Sheng Zhou, Cheng Yu, and Zhisheng Niu. 2015. CONCERT: A Cloud-based Architecture for Next-Generation Cellular Systems. *IEEE Wireless Communications* 21 (6): 14–22. https://doi.org/10.1109/mwc.2014.7000967.

Mahmoudi, Charif, Fabrice Mourlin, and Abdella Battou. 2018. *Formal Definition of Edge Computing: An Emphasis on Mobile Cloud and IoT Composition*. 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), 34–42. IEEE. https://doi.org/10.1109/FMEC.2018.8364042.

Muñoz, Raul, Ricard Vilalta, Ramon Casellas, Ricardo Martinez, Thomas Szyrkowiec, Achim Autenrieth, Víctor López, and Diego López. "Integrated SDN/NFV management and orchestration architecture for dynamic deployment of virtual SDN control instances for virtual tenant networks." Journal of Optical Communications and Networking 7, no. 11 (2015): B62-B70.

Nebula. 2019. Scaling—Nebula Container Orchestrator. https://nebula.readthedocs.io/en/latest/scaling/.

Nygren, Erik, Ramesh K. Sitaraman, and Jennifer Sun. 2010. The Akamai Network. *ACM SIGOPS Operating Systems Review* 44 (3): 2. https://doi.org/10.1145/1842733.1842736.

Ongaro, Diego, and John Ousterhout. 2014. *In Search of an Understandable Consensus Algorithm*. 2014 Annual Technical Conference, 305–319.

OpenFog Consortium. 2018. *IEEE Standard for Adoption of OpenFog Reference Architecture for Fog Computing*. IEEE Std 1934–2018, August, 1–176. https://doi.org/10.1109/IEEESTD.2018.8423800.

OPNFV. 2019. Software—OPNFV. https://www.opnfv.org/software.

Pahl, Claus, and Brian Lee. 2015. *Containers and Clusters for Edge Cloud Architectures—A Technology Review*. 2015 3rd International Conference on Future Internet of Things and Cloud, 379–386. IEEE. https://doi.org/10.1109/FiCloud.2015.35.

Pahl, Claus, Antonio Brogi, Jacopo Soldani, and Pooyan Jamshidi. 2019. Cloud Container Technologies: A State-of-the-Art Review. *IEEE Transactions on Cloud Computing* 7 (3): 677–692. https://doi.org/10.1109/TCC.2017.2702586.

Ranjan, Rajiv, Boualem Benatallah, Schahram Dustdar, and Michael P. Papazoglou. 2015. Cloud Resource Orchestration Programming: Overview, Issues, and Directions. *IEEE Internet Computing* 19 (5): 46–56. https://doi.org/10.1109/MIC.2015.20.

Santos, Jose, Tim Wauters, Bruno Volckaert, and Filip De Turck. 2017. *Resource Provisioning for IoT Application Services in Smart Cities*. 2017 13th International Conference on Network and Service Management (CNSM), 2018-January, 1–9. IEEE. https://doi.org/10.23919/CNSM.2017.8255974.

———. 2019. *Towards Network-Aware Resource Provisioning in Kubernetes for Fog Computing Applications*. IEEE Conference on Network Softwarization (NETSOFT), Paris.

Turnbull, James. 2014. *The Docker Book: Containerization Is the New Virtualization*. James Turnbull.

Vaquero, Luis M., Felix Cuadrado, Yehia Elkhatib, Jorge Bernal-Bernabe, Satish N. Srirama, and Mohamed Faten Zhani. 2019. Research Challenges in Nextgen Service Orchestration. *Future Generation Computer Systems* 90: 20–38. https://doi.org/10.1016/j.future.2018.07.039.

Velasquez, Karima, David Perez Abreu, Diogo Goncalves, Luiz Bittencourt, Marilia Curado, Edmundo Monteiro, and Edmundo Madeira. 2017. *Service Orchestration in Fog Environments.* Proceedings—2017 IEEE 5th International Conference on Future Internet of Things and Cloud, FiCloud 2017, 2017-January, 329–336. https://doi.org/10.1109/FiCloud.2017.49.

Yannuzzi, M., R. Irons-Mclean, F. van Lingen, S. Raghav, A. Somaraju, C. Byers, T. Zhang, et al. 2017. *Toward a Converged OpenFog and ETSI MANO Architecture.* 2017 IEEE Fog World Congress (FWC), 1–6. IEEE. https://doi.org/10.1109/FWC.2017.8368535.