2025 12th International Conference on Future Internet of Things and Cloud (FiCloud) | 979-8-3315-5437-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/FiCloud66139.2025.00014

Survey of Container Orchestration Distributions in Cloud-to-Edge Continuum

Sercan Sari
Research and Development, Siemens
Istanbul, Turkey
sari.sercan@siemens.com

***Abstract*—The shift in industrial paradigms from 4.0 to 5.0 has accelerated the convergence of operational technology and information technology across the cloud-to-edge continuum. This brings forth orchestration solutions with a capability to work in constrained resource edge deployments. While Kubernetes (K8s) is the de facto platform for cloud-scale container orchestration, its demanding resource profile runs the risk of overloading edge device capabilities. Consequently, some lightweight K8s distributions have emerged to address the unique constraints of edge computing, for example, limited resources, intermittent connectivity, and hardware heterogeneity. This survey examines how these lightweight K8s distributions are compared in current literature, with particular focus on K3s, MicroK8s, K0s, MicroShift, and KubeEdge. We integrate methodologies, evaluation criteria, and results from comparative studies. Our analysis demonstrates substantial diversity in testing methods and emphasizes the context-dependent nature of distribution performance. Through aggregating evidence-based results from previous studies, this survey provides guidance to practitioners in making appropriate orchestration technologies on the cloud-to-edge spectrum and establishing foundations for future directions of research in this evolving field.**

***Index Terms*—edge computing, internet of things (iot), containers, virtualization**

I. Introduction

The cloud-to-edge continuum represents a paradigm of computing that stretches computation from centralized cloud platforms to decentralized edge platforms [1]. It affects the industrial internet of things (IIoT), as the the gap between information technology (IT) and operational technology (OT) continues to shrink day by day. IIoT enables remote monitoring and autonomous operations and makes industrial areas more responsive, safer and powering Industry 4.0 initiatives globally [2].

As industrial paradigms evolve from Industry 4.0's focus on automation and connectivity to Industry 5.0's focus on human-machine collaboration, sustainability, and resilience [3], [4], the underlying technological infrastructure needs to evolve as well. This change requires solutions capable of managing complex distributed applications across the cloud-to-edge continuum. The human-centricity of Industry 5.0 requires not only automated mechanisms but also intelligent management of computational resources that can adapt to changing human requirements without compromising on operational effectiveness. Additionally, the merging of traditional OT and modern IT systems needs robust platforms that can bridge these historically disparate domains while addressing their requirements.

Kubernetes (K8s) [5] has started emerging as an orchestration platform easing dynamic application management within such industrialized complex environments. Traditional OT solutions are being redesigned through containerized technologies, with K8s offering opportunities to optimize resource allocation and improve operational efficiency. However, since the original K8s is meant to work in IT native environments, it is needed to be tailored for different requirements in different edge environments.

Edge and lightweight distributions of K8s have emerged to address such constraints and requirements of native edge environments that traditional K8s deployments are unable to successfully accommodate. Unlike cloud environments, edge computing needs to deal with gigantic challenges, such as constrained resources, intermittent network connectivity, and heterogeneous hardware configurations in multiple locations. Specialized distributions like K3s, MicroK8s, and K0s offer light-weighted offerings with reduced resource footprints, support for offline operation, and tuned communication protocols for high-latency networks.

The aim of this study is to investigate how lightweight distributions of K8s are compared and studied in current literature, providing criteria of comparison, and findings of comparative studies and paving the way for related future work. Through synthesizing studies of distributions including K3s, MicroK8s, K0s, MicroShift, and KubeEdge, we intend to find research designs and limitations in current evaluation approaches. This survey addresses the growing need for evidence-based guidance on selecting the appropriate orchestration technologies across the cloud-to-edge continuum, where deployment environments vary significantly in terms of resource availability, network conditions, and management requirements.

The remainder of the paper is organized as follows. Section 2 outlines the background information. Section 3 then gives details about lightweight K8s distributions. In Section 4, the focus shifts to comparative studies of Kubernetes distributions. Section 5 gives details about the proposed evaluation framework for lightweight K8s distributions. Finally, Section 6 summarizes the study's conclusions and offers suggestions for future research.

979-8-3315-5437-8/25/$31.00 ©2025 IEEE
DOI: 10.1109/FiCloud66139.2025.00014

II. Background

The cloud-to-edge continuum is a distributed computing model that spans hierarchical computational resources from centralized cloud data centers to in-between processing nodes and edge devices at the network [1]. Compared to traditional cloud-centric models, which concentrate on processing in remote facilities, this spectrum establishes a gradient of computing capacity instantiated across several physical and logical distances from data providers and consumers [6].

The idea of edge computing has evolved from earlier paradigms such as distributed computing, grid computing, and ubiquitous computing, but with a particular emphasis on uninterrupted utilization of resources on diverse infrastructure [7]. This shift is driven by new application requirements that cannot be adequately addressed by standalone centralized cloud models, including latency-sensitive applications, bandwidth-constrained situations, and privacy-sensitive use cases requiring on-premises data processing [8]. The continuum model supports context-based deployment decisions, with workloads best located on the continuum based on their specific requirements and environmental constraints, as compared to a one-size-fits-all approach to infrastructure [9].

A. Computing Resource Spectrum: Cloud, Fog, Edge

When we look at the computing resource spectrum across this continuum, we can see that it divides into multiple tiers, each with its own distinct characteristics and purposes. At one end, cloud computing represents the centralized powerhouse of this spectrum, and it's characterized by virtually unlimited computational resources and high reliability [10]. Not only do cloud environments offer comprehensive management capabilities, but they also provide sophisticated orchestration tools. They introduce relatively high latency for remote users and devices, which can be problematic in certain scenarios [11]. Although traditional cloud providers have established global infrastructure footprints to reduce these latencies, both physical and network constraints still impose fundamental limits on responsiveness, especially for time-critical applications where every millisecond counts [12].

Fog computing is a layer that lies in the middle of this spectrum, between the cloud and the edge, where distributed computing nodes bring cloud capabilities closer to data sources. It builds a link between fully distributed and centralized computing [13]. Since these fog resources are typically deployed at strategic locations such as local data centers, network aggregation points, or cellular base stations, they provide moderate processing power with lower latency compared to centralized clouds [14]. Bonomi et al. [15] state that fog computing is characterized by a number of unique features, such as heterogeneity, real-time interactions, mobility support, and geographic distribution, all of which add to its special place in the continuum. Fog nodes are appropriate for many latency-sensitive applications because they can provide more predictable performance and resource availability because they usually operate in more controlled environments than edge devices.

At the far end of the spectrum, edge computing represents the distributed boundary of the continuum, where computational resources are either embedded in or positioned very close to endpoint devices and data sources. By this way, it minimizes the distance data must travel [16]. This level includes a diverse range of device categories, including IoT gateways, industrial controllers, network equipment with computational capabilities, and increasingly powerful endpoint devices that can process data locally [17]. While edge resources are certainly characterized by significant constraints in processing power, memory, storage, energy availability, and network connectivity compared to cloud or fog tiers [18]. For many use cases, they offer crucial features like minimal latency for local operations and the ability to continue functioning even in the event of network outages [19]. According to Shi and Dustdar's analysis, edge computing has several important characteristics that make it appropriate for latency-sensitive and locally autonomous applications, such as closeness to data sources, dense geographical distribution, mobility support, and context awareness [20].

B. Container Technology and Orchestration Principles

Containers represent a virtualization technology that includes applications and their dependencies in isolated, portable units. Unlike traditional virtual machines (VMs) that include entire operating systems, containers share the host system's kernel while maintaining isolation through Linux namespaces and control groups (cgroups) [21]. The revolution of containers began with Docker's introduction, which consolidated container formats and simplified their creation and deployment [22]. Containers provide homogeneous environments during development, test, and deployment stages, essentially eliminating the "it works on my machine" syndrome that has plagued software development for so long. This technology enables microservices architectures with the capability of complex applications to be decomposed into simpler, independently deployable services, improving modularity, scalability, and development velocity [23], [24].

The fundamental components of container technology are images, registries , and runtimes [25]. Open Container Initiative (OCI) has established industry standards for container formats and runtimes to ensure compatibility and prevent vendor lock-in. Aside from Docker, there are container technologies like containerd, CRI-O, and rkt, each having specific characteristics better suited to a specific deployment use case [26].

In today's world, the number of containerized applications is increasing to make them run in different environments without needing any dependency. As the complexity of containerized applications increases, orchestration can no longer remain a choice but a necessity. Container orchestration platforms deploy, scale, network, and control containerized applications in distributed systems [27] Kubernetes, a Google-designed one that now has the Cloud Native Computing Foundation (CNCF) running it, has become the de facto solution for container orchestrating [21]. Some of the other well-known orchestrators

are Docker Swarm, Amazon ECS, and HashiCorp Nomad that vary in terms of architecture as well as design philosophy.

Container orchestration platforms address several critical requirements. They make decisions on scheduling the optimal placement of the containers over infrastructures at their disposal based on resource constraints, and priority settings [28]. On the other hand, they provide service discovery and internal load balancing so that containers can discover and communicate with one another even if their existence is transient [29]. The other important aspect is that they gain self-healing capability through health checks and auto-restarts, which improve system robustness [30].

C. Edge Computing and Container Orchestration

Edge computing is a paradigm that brings computation and data storage closer to where it is needed to avoid latency limitations and bandwidth expenses inherent in cloud-based models [16]. This paradigm emerges as a result of increased number of IoT devices, which by 2025 will be generating 80B zettabytes of data [31]. Edge computing complements rather than replaces cloud infrastructure, creating a continuum of computing resources distributed from centralized data centers to the network edge [32].

Cloud-edge continuum provides many architectural models that include multi-tier edge deployments, cloudlets, fog computing, and mobile edge computing. The models have differences in how far they stretch from end devices, resource capability, and administration strategies but they all aim at conserving on latency and network bandwidth [15], [33]. Applications with timing constraints such as autonomous cars, industrial automation, and augmented reality are particularly well-suited to edge processing, which has the capability to reduce round-trip latencies from hundreds of milliseconds to single-digit milliseconds [34].

Container technology has become important in such distributed computing scenarios due to its deployment architecture. However, traditional container orchestration platforms designed for data center infrastructures face challenges when deployed at edge locations, including resource constraints, intermittent connectivity, heterogeneous hardware, and physical security concerns [35]. These challenges have led to the development of optimized orchestration distributions for the cloud-to-edge continuum with a balance between control and edge autonomy.

III. Lightweight Distributions of Kubernetes

In the scope of this survey, we focus on five lightweight K8s distributions and current studies in the literature that compare these distributions in terms of performance, availability, usability and some other metrics. In Table I, we summarize the comparative analysis of lightweight Kubernetes distributions, highlighting their resource requirements, architectural features, and suitability for different deployment scenarios across the cloud-to-edge continuum.

A. K3s

K3s, initially developed by Rancher Labs and now part of SUSE, was one of the earliest lightweight K8s distributions and has evolved into a reference implementation in this category. Through careful packaging of all K8s necessities into a compact single binary weighing less than 100MB, K3s achieves substantial resource conservation while maintaining CNCF conformance.One of its design decisions is replacing the resource-intensive etcd database with SQLite as the default data store This reduces memory usage by an enormous degree without compromising essential functionality. By integrating components in a strategic manner and removing extraneous features, K3s requires just 512MB RAM to operate effectively—75% less than standard K8s deployments [36]. This release has been one of the most effective in retail environments, shop floors, and hardware-constrained IoT deployments where its minimal footprint enables consistent orchestration patterns regardless of hardware limitations.

B. MicroK8s

Canonical's MicroK8s approaches the lightweight K8s challenge through an innovative packaging perspective, leveraging the company's snap package system to create an isolated, self-healing distribution with distinctive operational characteristics. Unlike other distributions that focus primarily on binary size reduction, MicroK8s emphasizes a modular addon architecture that permits users to selectively enable only necessary components, thereby optimizing resource utilization for specific deployment scenarios. Although MicroK8s usually requires slightly more resources than K3s, it compensates for this through robust high-availability features and auto-recovery machanisms that continuously monitor and correct cluster components independently of human involvement [37].

C. K0s

K0s, developed by Mirantis, is one of the newer distributions in the lightweight Kubernetes ecosystem. It is well known for minimal dependencies and a philosophy of simplicity. This distribution provides a complete K8s distribution packaged as a single binary that has no external dependencies except the Linux kernel itself [38].

D. MicroShift

MicroShift, developed by Red Hat, takes a different approach among light-weight K8s distributions. It adapts OpenShift capabilities for edge computing rather than adjusting the core Kubernetes code. This approach supports compatibility with Red Hat's enterprise OpenShift platform while keeping resource demands significantly low through clever component design, streamlined control processes, and tailored interfaces. Seamless integration of MicroShift into OpenShift management tools and security infrastructures, MicroShift empowers organizations to embrace uniform practice across their infrastructure from huge data centers all the way down to tiny edge devices [39].

TABLE I
COMPARATIVE ANALYSIS OF LIGHTWEIGHT DISTRIBUTIONS OF KUBERNETES

Feature/Aspect	K3s [36]	MicroK8s [37]	K0s [38]	MicroShift [39]	KubeEdge [40]	Standard Kubernetes [5]
Origin	Rancher Labs (SUSE)	Canonical	Mirantis	Red Hat	Huawei, now CNCF project	Google, now CNCF
Implementation Approach	Single binary replacing components with lightweight alternatives SQLite instead of etcd	Snap-packaged K8s with selective component enablement	Zero dependencies beyond Linux kernel Single binary Containerd-in-containerd support	OpenShift components optimized for edge deployment Reduced control plane	Two-tier architecture with separate cloud and edge components	Full component architecture
Connectivity Model	Server-agent architecture with websocket tunneling Embedded flannel CNI Reduced port requirements	Standard K8s model	Similar to standard Kubernetes with optimizations	OpenShift-compatible with edge optimizations	Two-tier architecture with edge autonomy	Direct connection to API server required
Security Features	Automated certificate management Standard K8s RBAC	Standard K8s security with built-in Calico network policies	Strong security defaults Simplified certificate management	Inherits OpenShift security model Adapted for edge environments	Extended model with IoT device identity management	Comprehensive security model with external authentication integration
Management Complexity	1-3 commands for installation	Simple snap installation Addon-based feature management	Simple installation Automated deployment Controller isolation	Simplified compared to OpenShift OpenShift-compatible management	Complex setup with cloud/edge components Specialized edge management interfaces	15-25 discrete steps Extensive API surface
Key Innovations	Consolidated components Minimal resource footprint Production-ready simplicity	Snap packaging Modular addons Automatic recovery	Zero external dependencies Containerd-in-containerd support Worker and controller isolation	OpenShift compatibility at the edge Optimized control loops Enterprise integration	Device twin functionality Metadata synchronization MQTT integration	Comprehensive orchestration model Extensive ecosystem

E. KubeEdge

KubeEdge, originally created by Huawei and later contributed to the Cloud Native Computing Foundation, introduces a different architectural approach among the distributions we have examined. Its two-tier design divides responsibilities between cloud and edge environments: the CloudCore component works with standard K8s control systems, while the streamlined EdgeCore component runs efficiently on edge devices with limited resources. Connecting these components, KubeEdge employs a metadata synchronization system specifically built for unreliable networks, which reduces management traffic by up to 90% compared to regular K8s communications. The platform enhances K8s with robust IoT device management features, including an inventive "Device Twin" capability that creates virtual representations of physical devices that can be managed through familiar K8s interfaces [40].

IV. COMPARATIVE STUDIES OF KUBERNETES DISTRIBUTIONS

Some studies in the literature compare and analyze different lightweight distributions of Kubernetes against different benchmarks in different setups.

Fathoni et al. [41] conducted one of the earliest systematic comparisons of lightweight distributions designed specifically for resource-limited edge devices. The authors executed KubeEdge and K3s on Raspberry Pi 3+ Model B hardware (1GB RAM, Quad Core 1.2 GHz) to evaluate their performance in actual edge computing environments. Using a custom GPS location simulator container for load testing, they measured CPU and memory usage with htop. Their findings indicated that KubeEdge performed slightly more efficiently in idle resource usage with 10% CPU versus K3s at 14%.

Böhm et al. [42] extended the evaluation beyond basic resource metrics to include operational scenarios. Using virtualized environments (4 Ubuntu VMs, 2 vCPUs, and 4GB RAM each), they tested standard Kubernetes, MicroK8s, and K3s on several operations like cluster management and application deployment. From their extensive monitoring using netdata, the researchers noted that MicroK8s consumed slightly more resources than standard Kubernetes and K3s for most operations. Significantly, the study included sophisticated workflows like worker node draining, providing operational performance insight that extended beyond simple idle measurements. The findings quantified K3s's efficiency advantages in virtualized environments typical of small deployment scales.

Telenyk et al. [43] utilized cloud infrastructure to experiment with performance in such environments. With standard Kubernetes, MicroK8s, and K3s on Google Cloud instances (2 vCPUs, 8GB RAM), they performed comparative analysis between typical cluster operations. Contrary to some other research studies, their findings were opposite, where standard Kubernetes was found to be better than K3s and MicroK8s in their test cases. This unexpected result highlighted the relevance of deployment context to distribution performance and suggested that light distributions will optimize for alternative operating conditions than those which apply in homogenous cloud environments.

In [44], the authors focused specifically on serverless computing performance. They deployed OpenFaaS across standard Kubernetes (installed via KubeSpray), MicroK8s, and K3s on substantial server hardware (Intel Xeon processors with 8GB RAM). Using 14 benchmarking functions, they measured application-level performance metrics including response times and throughput using the hey benchmarking tool. Their results showed very similar performance statistics for all three distributions, and they suggested that for serverless workloads on high-end hardware, the distribution choice could

TABLE II
EVALUATION FRAMEWORK FOR LIGHTWEIGHT KUBERNETES DISTRIBUTIONS

Dimension	Metrics	Measurement Approach
Performance	Resource Utilization	Memory footprint (idle, load, peak) CPU utilization patterns Storage requirements Network bandwidth consumption
	Operational Performance	Control plane responsiveness Deployment operation timing Recovery timing Scheduling efficiency
	Scalability & Resilience	Maximum pods per node Node failure recovery Network partition tolerance Control plane scalability
Human-Centric	Operational Complexity	Installation complexity score Management operation complexity Observability quality Error handling clarity
	Integration Capability	OT system integration effort Learning curve Documentation completeness Ecosystem compatibility
Sustainability	Resource Efficiency	Energy consumption profile Hardware utilization efficiency Long-term stability indicators
	Industrial Adaptability	Legacy system compatibility Security compliance capabilities Brownfield deployment suitability Lifecycle management

have minimal impact on application-level performance.

In [45], the authors utilized Microsoft Azure VMs (Intel Xeon E5-2674 v3 CPUs at 2.4. GHz, 2 cores and 8 GB RAM each). They compared the features and performance of MicroK8s, K3s, K0s, and MicroShift by applying stress scenarios. Their study is the first one putting stress on lightweight K8s distributions with more than 100 pods while analyzing the k0s and MicroShift as well. Their results showed that K3s and K0s have more control plane throughput.

Usman et al. [46], proposed a comprehensive model for evaluating the performance of lightweight container orchestration distributions. They focused on resource usage, control-plane, data-plane performance metrics. Their results showed that K3s is the overall best performance amongs the distributions.

V. EVALUATION FRAMEWORK FOR LIGHTWEIGHT KUBERNETES DISTRIBUTIONS

We have observed that previous comparative studies of lightweight Kubernetes distributions have suffered from significant methodological inconsistencies that limit their collective value for industrial decision-making. In [41], they focused narrowly on resource utilization metrics using Raspberry Pi hardware, while Böhm et al. [42] prioritized operational scenarios in virtualized environments. Telenyk et al. [43] contradicted these findings through cloud-based testing, revealing how deployment context fundamentally alters distribution performance. Later studies by Kjorveziroski [44] emphasized application-level metrics for serverless computing, while Koziolek [45] introduced stress testing with higher pod densities. This fragmentation of approaches such as varying in hardware platforms, metrics, workloads, and methodologies has created a disconnected rather than a unified understanding of distribution performance.

Our proposed framework in Table 2 addresses these limitations by integrating the strengths of previous approaches while extending evaluation criteria to encompass the human-centric and sustainability dimensions critical for Industry 5.0 environments. By standardizing hardware tiers, network configurations, workload profiles, and measurement methodologies, this framework enables consistent, reproducible evaluations that can be meaningfully compared across studies. This harmonization represents a significant advancement over the current state of research, where contradictory findings often result from methodological differences rather than actual distribution characteristics.

VI. CONCLUSION AND FUTURE WORK

When we look at the current studies, one of the most significant result is that the context-dependent performance variations across distributions. While K3s demonstrated resource efficiency advantages in constrained environments, standard Kubernetes sometimes outperformed lightweight alternatives in cloud settings, as shown in Telenyk's findings [43]. This

contextual performance variation suggests that distribution selection can be environment-specific rather than generic.

Additionally, these studies show the changes of evaluation methodologies. Earlier research like Fathoni's [41] focused primarily on basic resource metrics (CPU, memory) on single-node configurations, while later studies incorporated operational scenarios, stress testing, and multi-node deployments [42]. Koziolek's work particularly brings more realistic experiments to the field by examining scalability under significant load [45].

The contradictory findings across studies using different hardware profiles—from Raspberry Pi devices to cloud VMs—emphasizes that distribution optimization strategies may target specific hardware environments. This insight requires more standardized testing across representative hardware configurations spanning the continuum. Different industrial PCs can be used as a bare metal to setup and monitor different distributions.

Not only hardware environments, but also different cluster layers and configurations can affect the overall performance of the system. For the future research standardized hardware profiles can be helpful to identify optimization patterns across the spectrum. On the other hand, benchmark and workload patterns can affect the reproducibility of comparative studies. Furthermore, sustainability considerations remain largely unexplored in the current research studies. Future studies should include energy consumption profiling that quantifies power usage in different operational states and patterns of workload. It should be also assessed the efficiency of resource utilization to establish the extent to which distributions optimize hardware utilization while conserving resources an important factor in resource-scarce edge ecosystems.

Although edge environments introduce new security issues like physical access risksand heterogeneous attack surfaces, few studies have addressed cybersecurity issues in lightweight distributions. Similarly, legacy system support, one of the central pillars of IT/OT convergence success, has not been explored in comparison research, although industrial environments usually consist of equipment with lifecycles that range from decades and proprietary protocols.

In future work we plan to compare some of the different distributions with the proposed framework. This will give a systematic comparison chance of different flavors with the specific measurements. On the other hand, future research can also examine security features beyond basic requirements, including runtime security enforcement and industrial security standards compatibility. Additional research should examine integration features with legacy systems through industrial protocols (OPC Unified Architecture, Modbus, PROFINET), hardware interfaces, and brownfield deployment patterns.

These are critical for Industry 5.0 environments where human-centric, resilient systems must operate securely within existing industrial environments while being able to sustain operational integrity through generations of technology.

REFERENCES

[1] M. Villari, M. Fazio, S. Dustdar, O. Rana, and R. Ranjan, "Osmotic computing: A new paradigm for edge/cloud integration," *IEEE Cloud Computing*, vol. 3, no. 6, pp. 76–83, 2016.

[2] S. F. Ahmed, M. S. B. Alam, M. Hoque, A. Lameesa, S. Afrin, T. Farah, M. Kabir, G. Shafiullah, and S. Muyeen, "Industrial internet of things enabled technologies, challenges, and future directions," *Computers and Electrical Engineering*, vol. 110, p. 108847, 2023.

[3] S. Nahavandi, "Industry 5.0—a human-centric solution," *Sustainability*, vol. 11, no. 16, p. 4371, 2019.

[4] X. Xu, Y. Lu, B. Vogel-Heuser, and L. Wang, "Industry 4.0 and industry 5.0—inception, conception and perception," *Journal of manufacturing systems*, vol. 61, pp. 530–535, 2021.

[5] Kubernetes. Production-grade container orchestration. [Online]. Available: https://kubernetes.io/

[6] T. Lynn, P. Rosati, A. Lejeune, and V. Emeakaroha, "A preliminary review of enterprise serverless cloud computing (function-as-a-service) platforms," in *2017 IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2017, pp. 162–169.

[7] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[8] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[9] 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*, vol. 3, pp. 134–155, 2018.

[10] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing," 2011.

[11] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. Netto *et al.*, "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM computing surveys (CSUR)*, vol. 51, no. 5, pp. 1–38, 2018.

[12] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, pp. 7–18, 2010.

[13] S. Yi, C. Li, and Q. Li, "A survey of fog computing: concepts, applications and issues," in *Proceedings of the 2015 workshop on mobile big data*, 2015, pp. 37–42.

[14] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, 2016.

[15] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13–16.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[17] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere, "Edge-centric computing: Vision and challenges," pp. 37–42, 2015.

[18] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE communications surveys & tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

[19] S. Chen, L. Jiao, L. Wang, and F. Liu, "An online market mechanism for edge emergency demand response via cloudlet control," in *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*. IEEE, 2019, pp. 2566–2574.

[20] W. Shi and S. Dustdar, "The promise of edge computing," *Computer*, vol. 49, no. 5, pp. 78–81, 2016.

[21] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, omega, and kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.

[22] D. Merkel *et al.*, "Docker: lightweight linux containers for consistent development and deployment," *Linux j*, vol. 239, no. 2, p. 2, 2014.

[23] C. Pahl and B. Lee, "Containers and clusters for edge cloud architectures–a technology review," in *2015 3rd international conference on future internet of things and cloud*. IEEE, 2015, pp. 379–386.

[24] S. Newman, *Building microservices: designing fine-grained systems*. " O'Reilly Media, Inc.", 2021.
[25] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," *IEEE cloud computing*, vol. 1, no. 3, pp. 81–84, 2014.
[26] Open Container Initiative. (2020) About OCI. [Online]. Available: https://opencontainers.org/about/overview/
[27] E. A. Brewer, "Kubernetes and the path to cloud native," in *Proceedings of the sixth ACM symposium on cloud computing*, 2015, pp. 167–167.
[28] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at google with borg," in *Proceedings of the tenth european conference on computer systems*, 2015, pp. 1–17.
[29] B. Burns and D. Oppenheimer, "Design patterns for container-based distributed systems," in *8th USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 16)*, 2016.
[30] B. Burns, J. Beda, K. Hightower, and L. Evenson, *Kubernetes: up and running: dive into the future of infrastructure*. " O'Reilly Media, Inc.", 2022.
[31] IDC. Future of industry ecosystems: Shared data and insights. [Online]. Available: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/
[32] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.
[33] R. Vilalta, V. López, A. Giorgetti, S. Peng, V. Orsini, L. Velasco, R. Serral-Gracia, D. Morris, S. De Fina, F. Cugini *et al.*, "Telcofog: A unified flexible fog and cloud computing architecture for 5g networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 36–43, 2017.
[34] S. Chen and J. Zhao, "The requirements, challenges, and technologies for 5g of terrestrial mobile telecommunication," *IEEE communications magazine*, vol. 52, no. 5, pp. 36–43, 2014.
[35] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, and T. Taleb, "Survey on multi-access edge computing for internet of things realization," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 2961–2991, 2018.
[36] SUSE. Lightweight kubernetes. [Online]. Available: https://k3s.io/
[37] Canonical. The effortless kubernetes. [Online]. Available: https://microk8s.io/
[38] Mirantis. Zero friction kubernetes. [Online]. Available: https://k0sproject.io/
[39] MicroShift. Microshift. [Online]. Available: https://microshift.io/
[40] KubeEdge. Kubernetes native edge computing framework. [Online]. Available: https://kubeedge.io/
[41] H. Fathoni, C.-T. Yang, C.-H. Chang, and C.-Y. Huang, "Performance comparison of lightweight kubernetes in edge devices," in *Pervasive Systems, Algorithms and Networks: 16th International Symposium, I-SPAN 2019, Naples, Italy, September 16-20, 2019, Proceedings 16*. Springer, 2019, pp. 304–309.
[42] S. Böhm and G. Wirtz, "Profiling lightweight container platforms: Microk8s and k3s in comparison to kubernetes." in *ZEUS*, 2021, pp. 65–73.
[43] S. Telenyk, O. Sopov, E. Zharikov, and G. Nowakowski, "A comparison of kubernetes and kubernetes-compatible platforms," in *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, vol. 1. IEEE, 2021, pp. 313–317.
[44] V. Kjorveziroski and S. Filiposka, "Kubernetes distributions for the edge: serverless performance evaluation," *The Journal of Supercomputing*, vol. 78, no. 11, pp. 13 728–13 755, 2022.
[45] H. Koziolek and N. Eskandani, "Lightweight kubernetes distributions: A performance comparison of microk8s, k3s, k0s, and microshift," in *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering*, 2023, pp. 17–29.
[46] M. Usman, S. Ferlin, and A. Brunstrom, "Performance analysis of lightweight container orchestration platforms for edge-based iot applications," in *2024 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2024, pp. 321–332.