Comparison of container orchestration platforms

Craig Dillon
School of Enterprise Computing and Digital Transformation
Technological University Dublin
Dublin, Ireland
x00205790@mytudublin.ie

***Abstract*—Containerisation has become a prominent tool in modern software development and deployment by enabling streamlined application lifecycles. Kubernetes has become the dominant orchestration platform due to its extensive feature set. However, as the containerisation landscape matures, lightweight alternatives have gained traction providing developers with alternatives to traditional Kubernetes deployments.**

This literature review compares Kubernetes and lightweight distributions, analysing their differences and proposes topics for future research. The study aims to provide valuable insights for organisations or teams seeking to implement a container orchestration platform by providing a brief overview of Kubernetes and further comparing some key differences between several distributions. Through analysis of existing literature, this review identifies gaps in existing literature and suggests potential avenues for further research, contributing to the ongoing development of container orchestration platforms.

Keywords— DevOps, Kubernetes, K3s, Cloud-native, Containerisation, Minikube

I. Introduction

Containerisation has become an industry standard in modern software development and deployment, allowing for streamlined application lifecycles. Kubernetes has emerged as the primary orchestration platform seeing widespread adoption as a result of a mature and robust feature set. However, as the technology evolves, lightweight alternatives such as Minikube and k3s have come to the fore, offering developers efficient alternatives to traditional Kubernetes deployments. This literature review aims to compare Kubernetes and lightweight alternatives, exploring their strengths, weaknesses and suitable use cases. As organisations aim to find the right balance between ease of use, scalability, efficiency and acceptable costs, understanding the differences between these orchestrators can provide valuable information when deciding on an orchestrator for containerised workflows. Through an examination of key features, performance metrics, ease of use and real world implementations, this review will analyse the practicalities of implementing Kubernetes or lightweight counterparts.

A. Motivation

The adoption of container orchestration platforms by organisations is growing, driven by a shift towards cloud-native computing. This literature review will explore the nuances of different container orchestration platforms and their role in modern DevOps operations. By focusing on the comparison of these platforms, their features and capabilities, this paper aims to provide insight on the potential advantages and challenges associated with their adoption. Key questions to be addressed include;

- What are the distinct differences between these platforms?
- What are the resource requirements?
- What are the key features of a container orchestration platform?

B. Contribution

This review serves as a consolidation of existing literature container orchestration platforms, drawing from various journals and conference proceedings to provide an overview of the similarities and differences of several orchestration platforms. Furthermore, by identifying gaps in the current literature and suggesting potential future research topics, this review seeks to contribute to ongoing research and innovation in the field of container orchestration.

II. Theoretical background

A. Overview of Kubernetes

Kubernetes is an open-source orchestration platform designed to streamline the deployment, scaling and management of containerised applications. Kubernetes was released in 2014, is a component of the Cloud Native Computing Foundation (CNCF) and has seen widespread adoption, *Kjorveziroski & Filiposka (2022), Ding et al. (2022), Turin et al. (2023)*. Core concepts of Kubernetes are;

- Pods - the fundamental deployment unit in Kubernetes. A pod consists of one or more co-located containers that share resources and can communicate directly with one another.
- Nodes - Kubernetes has two types of nodes.
  - Master Node - The central control point, overseeing cluster management, API exposure and is responsible for allocation of pods and communication within the cluster.
  - Worker Nodes - Make up the larger part of a cluster and host pods. Worker nodes have a defined CPU and memory allocation.
- Containers - Encapsulated applications in self-contained binaries. Containers are used by Kubernetes to run pods.
- Services - Services act as internal load balancers and proxies for pods, creating logical groups of pods performing the same function.

- The Scheduler - Monitors and tracks node capacity and resource allocation, allowing for proper deployment of pods to nodes with sufficient resources available.

Turin et al. (2023).

B. Overview of lightweight orchestration platforms

There are several lightweight orchestration platforms available for anyone who has a requirement for deployments with low resource requirements. Some examples of Kubernetes based platforms are K3s, MicroK8s and Kubespray *Kjorveziroski & Filiposka (2022)*, with ioFog, *Čilić et al. (2023)* and Nomad, *Bahy et al. (2023)* serving as alternatives to Kubernetes based platforms. Kubernetes is feature rich, which allows it to be deployed in a wide range of environments, at a cost of size, resource requirements and deployment complexity *Kjorveziroski & Filiposka (2022)*. As a result, alternative platforms have been developed that strip back specialised features and components, focusing on reduced resource requirements and simple deployments *Kjorveziroski & Filiposka (2022)*.

C. Key features of orchestration platforms

Container orchestration systems provide a platform to deploy containerised services providing scalability, load balancing, deployment patterns through declarative configuration *Turin et al. (2023)* and are responsible for managing the lifecycle of containerised services *Bahy et al. (2023)*. Containers running on these platforms share the operating system with the host machine as opposed to running a complete operating system within a hypervisor as is the case with traditional virtual machines, *Ding et al. (2022)*. Containers running on a node can share dependency libraries, and as a result of these features containers require a much lower resource footprint and are quicker to start than virtual machines, *Ding et al. (2022)*.

III. Technical comparison

The underlying methodologies for container orchestration platforms remain the same, where containers share the underlying hardware and operating system, *Turin et al. (2023)*, *Ding et al. (2022)*. Differences between these orchestration tools come in the form of resource requirements, bundled packages and limitations where certain features have been removed to improve efficiency, *Kjorveziroski & Filiposka (2022)*.

A. Resource utilisation

Resource utilisation and efficiency are key metrics when comparing container orchestration platforms. When testing a Kubernetes scheduler to reduce CO_2 emissions, *Piontek et al. (2024)* simulated a server that had 36 cores and 768GB RAM to reflect a real world scenario. However, Kubernetes appears to be a versatile service that *"has an unexpectedly low memory footprint on a resource-constrained node"* and *"Kubernetes performed excellently in both cluster deployment and performance evaluation"*, *Čilić et al. (2023)*. One drawback noted was the requirement of a VPN to connect nodes in remote private networks, noting that K3s did not have this particular disadvantage as it supplies a tunnel proxy that can perform bidirectional data transfers between remote nodes and the controller. In comparison to the aforementioned large Kubernetes deployment, *Bahy et al. (2023)* deployed KubeEdge, K3s and Nomad on an 8 core CPU with 3.5GB RAM hosted on Azure. These orchestration platforms were also tested on a quad-core ARM processor with 1GB RAM showing great versatility in the ability to deploy container orchestration services to a variety of architectures, *Bahy et al. (2023)*.

B. Variations in datastores

While these distributions share a similar end goal, that is to provide an orchestration tool for containers, they make use of different modules and components to achieve their goal. For example, Kubernetes uses etcd, a key-value store that provides cluster state persistence, *Piontek et al. (2024)*, *Lai et al. (2023)*. K3s uses an adapter to allow for the use of different database backends with SQLite serving as the default option, *Kjorveziroski & Filiposka (2022)*. When performing tests, *Čilić et al. (2023)* noted that while SQLite reduced the size of the footprint of the controller this came at a performance cost. MicroK8s uses Dqlite for this purpose and Kubespray relies on etcd, *Kjorveziroski & Filiposka (2022)*.

C. Differences in networking

While many lightweight platforms are built from Kubernetes, they tend to undergo customisation to meet certain demands. Networking is a core component of an orchestration platform and can differ between distributions. K3s, for example utilises Flannel for container networking with a thin wrapper layer that uses unique shell commands, *Bahy et al. (2023)*. Flannel creates a layer 3 IPv4 network between nodes within a cluster, *Subratie et al. (2023)*. Cloud providers are also able to offer customised networking layers. Route Controller is an optional add-on for GCP (Google Compute Engine) clusters to allow communication between containers on different nodes, *Piontek et al. (2024)*. Calico is another networking resource that virtualises the IP layer and avoids tunneling where endpoints are addressable, *Subratie et al. (2023)*, and can be deployed on Kubernetes and several alternative distributions such as Kubespray, K3s and microk8s *Kjorveziroski & Filiposka (2022)*. KubeEdge also supports the MQTT protocol which allows for customised logic and facilitates communication with IoT devices, *Čilić et al. (2023)*.

IV. Limitations

While comparing container orchestration platforms offers valuable insights, it is important to highlight limitations that impact the scope and ability compare like-for-like. The workload scenarios and environments presented in the source papers show a great deal of variance. Some papers focus on edge computing performing a range of tests such as, ease of deployment, networking constraints, memory footprint, startup and performance comparisons, *Čilić et al. (2023)*. Other papers focus on RAM and CPU usage when the orchestration platforms are in different states, *Bahy et al. (2023)* or performing

a range of processes like gzip compression, video processing and model training, *Kjorveziroski & Filiposka (2022)*. Only general conclusions can be drawn from the results in these papers as a result of different testing methodologies and testing environments.

V. Conclusion

Comparing Kubernetes and lightweight alternatives has revealed a diverse landscape of innovation and research, with a notable focus on catering to specific use cases such as edge computing, low energy consumption and niche deployment scenarios. While these studies have contributed valuable insights into the container orchestration ecosystem, this paper has highlighted a critical gap in research relating to the operations of service stacks for end users, customers and enterprise applications. Kubernetes was proven to be quite versatile and in one study performed well on low resource hardware. Another highlight is the fact that many of these distributions can run on a wide range of hardware, from different processor architectures, RAM requirements and storage footprints.

VI. Future work

Existing literature provides valuable insights into the comparison of Kubernetes and related lightweight distributions. However, there are several potential avenues for further research since much of the literature focuses on edge computing when examining features of lightweight distributions. The following proposals provide areas of further research.

A. In depth comparison of state backends

Kubernetes relies on etcd to maintain cluster persistence while other distributions have a range of datastores that can be utilised. An in depth study into the performance and scalability of a single distribution using each available datastore. For example, k3s supports etcd, SQLite, MySQL, MariaDB and PostgresQL, running a variety of performance tests may show benefits of certain datastores depending on the use case.

B. Performance testing between Kubernetes and lightweight alternatives

Kubernetes is often the go-to solution for container orchestration where resource limitations are not as constrained in comparison to edge computing environments. Comparing Kubernetes with lightweight distributions using similar resources for an application or service that relies on several backend functions would be worthwhile especially when comparing scalability, robustness, cost and limitations.

C. End user adoption study

Conducting a user-centric study into the user experiences, operational considerations and challenges related to Kubernetes and related distributions would provide valuable insights into deployment experiences, pain points and best practices. This would be qualitative research by conducting interviews and surveys.

References

Bahy, M. B., Riyanto, N. R. D., Siswantoro, M. Z. F. N. & Santoso, B. J. (2023), Resource utilization comparison of kubeedge, k3s, and nomad for edge computing, *in* '2023 10th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)', IEEE, pp. 321–327.

Čilić, I., Krivić, P., Podnar Žarko, I. & Kušek, M. (2023), 'Performance evaluation of container orchestration tools in edge computing environments', *Sensors* **23**(8), 4008.

Ding, Z., Wang, S. & Jiang, C. (2022), 'Kubernetes-oriented microservice placement with dynamic resource allocation', *IEEE Transactions on Cloud Computing* .

Kjorveziroski, V. & Filiposka, S. (2022), 'Kubernetes distributions for the edge: serverless performance evaluation', *The Journal of Supercomputing* **78**(11), 13728–13755.

Lai, W.-K., Wang, Y.-C. & Wei, S.-C. (2023), 'Delay-aware container scheduling in kubernetes', *IEEE Internet of Things Journal* .

Piontek, T., Haghshenas, K. & Aiello, M. (2024), 'Carbon emission-aware job scheduling for kubernetes deployments', *The Journal of Supercomputing* **80**(1), 549–569.

Subratie, K., Aditya, S. & Figueiredo, R. J. (2023), 'Edgevpn: Self-organizing layer-2 virtual edge networks', *Future Generation Computer Systems* **140**, 104–116.

Turin, G., Borgarelli, A., Donetti, S., Damiani, F., Johnsen, E. B. & Tarifa, S. L. T. (2023), 'Predicting resource consumption of kubernetes container systems using resource models', *Journal of Systems and Software* **203**, 111750.