A comparison of AKS and k3s running in Azure

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

***Abstract*—This research aims to conduct a comprehensive comparative analysis of k3s running on a Virtual Machine Scale Set and AKS. Both AKS and k3s are prominent container orchestration platforms used for deploying, managing and scaling containerised applications. AKS is a fully managed service offering by Azure and simplifies the management of Kubernetes clusters by automating critical tasks such as control plane management, monitoring and security. In contrast, k3s is a lightweight Kubernetes distribution optimised for edge-computing and resource constrained environments, which offers flexibility and customisation by using add-ons instead of built-in components.**

The primary objective of this study is to evaluate the performance, scalability, ease of deployment and cost efficiency of these platforms. This will involve conducting a series of quantitative tests under a variety of load conditions and scenarios. To ensure a fair comparison, the research will standardise key aspects of the test environment, including using etcd for cluster state management, Cilium for networking and consistent node specifications across both platforms.

Keywords— DevOps, Kubernetes, K3s, Cloud-native, Containerisation, k3s, Microservices

I. Introduction

While Azure Kubernetes Service (AKS) and k3s are well documented in a range of academic literature, there is a notable lack of research comparing these platforms. Most existing studies relating to the performance of k3s is in the context of edge or low resource-computing, overlooking the potential benefits of using a lightweight Kubernetes platform in an enterprise cloud environment. This research aims to fill this gap by providing a detailed comparative analysis of k3s running on Azure Virtual Machine Scale Sets (VMSS) and AKS, assessing their performance, scalablilty, ease of deployment and cost efficiency.

The study will rely on a consistent testing environment to ensure fairness in the comparison, providing valuable insights for teams who are considering these platforms. By addressing the research gap, this study aims to provide in depth metrics and knowledge relating to these two platforms and highlight their strengths and limitations. Performance metrics such as response time, throughput and resource utlisiation will be measured. Scalability will be assessed by simulating traffic spikes and targeting specific services to analyse each platforms ability to scale up or down dynamically. Cost analysis will be performed using Azure's cost management tools, and ease of deployment will be evaluated based on setup complexity, configuration and ongoing management requirements. Additionally, monitoring tools such as Prometheus, Grafana, Hubble and Goldpinger will be used to gather and analyse data.

This research aims to provide insights for teams deciding between AKS and k3s for their container orchestration needs, highlighting the strengths and limitations of each platform.

A. Overview of AKS

Azure Kubernetes Service (AKS) is a fully managed container orchestration service offered by Azure. The goal of AKS is to reduce the complexity and overhead of managing a Kubernetes cluster by removing the need to manually create and configure components such as control planes, cluster networking and monitoring solutions, *Sethy et al. (2022)*. Further to this, AKS also offers robust security features and compliance certifications meeting the security and regulatory requirements of customers, *Sindhu & Pavithra (2020), Goedtel (2024)*. As of writing AKS deploys with Ubuntu 22.04 LTS or Azure Linux 2.0 as the operating system, Kubernetes version 1.28.9 by default (although other versions are also available) and has a number of service mesh solutions on offer. When creating the AKS cluster, customers can choose from a variety of virtual machine SKUs for their application node pools and can also choose to create several node pools of varying sizes. The control plane and system node pool are created and managed entirely by Azure, while etcd is deployed as the keystore for state management and containerd as the container runtime, *Goedtel (2024)*.

B. Overview of k3s

k3s is a lightweight Cloud Native Computing Foundation (CNCF) compliant Kubernetes distribution designed to run using as small a binary as possible, *Füstös et al. (2022)*, *Telenyk et al. (2021)*. k3s is often associated with edge computing and resource limited devices *Füstös et al. (2022)*, however, it can still provide the core functionality of Kubernetes by removing the need for drivers and instead relying on the concept of add-ons in their place *Telenyk et al. (2021)*, reducing footprint and providing greater customisation. By default, platform deploys with sqlite3 as the datastore for state management, Flannel Container Network Interface (CNI) and Traefik as the ingress controller, although these can be customised, *Rancher (2023)*.

II. Objectives

The primary objective of this research is to conduct a comprehensive comparative analysis of Azure Kubernetes Service (AKS) and k3s running on Azure Virtual Machine Scale Sets (VMSS). The study aims to provide valuable insights into the performance, scalability, ease of deployment and cost-efficiency of both platforms in a cloud native environment. The specific objectives for this research are as follows;

- Performance evaluation - Assess and compare the performance of AKS and k3s under a variety of load conditions by measuring key performance metrics such as response time, throughput and resource ultilisation
- Scalability analysis - Analyse the scalability of both platforms by conducting tests that simulate varying traffic levels and workload intensity by measuring their ability to handle sudden spikes in demand
- Ease of deployment - Consider factors such as setup time, configuration complexity and management requirements of both platforms
- Cost comparison - Making use of Azure cost analysis compare the operational costs of both platforms

III. Limitations

While this research aims to provide valuable insights into the comparative analysis of AKS and k3s running on VMSS, it is important to acknowledge that there are limitations within this research.

A. Scope limitations

This research is focused on specific aspects of AKS and k3s, including performance, scalability, ease of deployment and cost efficiency. Certain features or functionalities of the platforms may not be fully explored within this study.

B. Testing environment

Despite efforts to standardise the testing environment, variations in network conditions, infrastructure configurations, or cloud provider services may introduce variables that could impact the reliability and reproducibility of the results.

C. Resource constraints

The depth and breadth of this research may be limited by available resources, including time, budget and computing resources. As a result, certain aspects of the research may not be fully explored or may be subject to resource constraints.

D. Vendor specific considerations

The findings of the research are specific to AKS and k3s and may not be directly applicable to other container orchestration platforms. Differences in architecture, features and vendor specific optimisations may impact the analysis.

E. Changes in platforms

The rapid pace of innovation in this space means that AKS and k3s may undergo updates or changes during the course of this research. New features or optimisations introduced after the study will not be captured in the analysis.

IV. Research methodology

This will be a quantitative study aimed at comparing k3s running on Azure Virtual Machine Scale Sets (VMSS) with AKS. The study aims to assess the performance, scalability, cost and ease of deployment of both platforms to provide insights their suitability for various use cases. To ensure the reliability and integrity of the comparison, several key factors will be addressed to maintain comparable operating environments as closely as possible.

A. Cluster configuration

AKS deploys with etcd as the key-value store for cluster state by default and cannot be changed. For consistency, k3s will also be deployed with etcd for cluster state management. Cilium will be used as the Container Network Interface (CNI) which is based on eBPF technology which allows programmable packet filtering and processing at the kernel layer, providing in depth visibility of network traffic and real-time performance metrics, *Magnani et al. (2022)*.

B. Node specifications

While AKS does not allow the end user to modify control plane or system node pools, the application node pool can be deployed with a selection of SKUs and for k3s the same SKU will be used. Node pools will have the same amount of scalable nodes and will be running Ubuntu 22.04 LTS to maintain consistency across environments.

C. Monitoring and metrics

Both AKS and k3s clusters will have the same monitoring and logging solutions to capture performance and monitor cluster health. This will ensure uniform data collection and analysis across environments. A selection of tools is as follows;

- Prometheus - A time series database that gathers metrics from the Kubernetes cluster
- Grafana - A visualisation and analytics platform for creating dashboards and graphs based on time-series data
- Hubble - Network visibility tool for Kubernetes that is part of the Cilium application stack providing insights into network traffic and policy enforcement
- Goldpinger - A Kubernetes liveness probe tool that measures and monitors network connectivity between pods

D. Workload and testing

A freely available microservice application stack will be deployed to both clusters. The application will make use of a variety of microservice features such as individual containers to handle different business logic, database backends and message queuing to replicate a full scale production environment. These deployments will be handled as part of a CI/CD pipeline to ensure consistency and all configuration and deployment data stored as infrastructure as code (IaC). Load testing will be synchronous to reduce the impact of network latency and other unseen factors that may affect the reliability and validity

of results. An example of some testing tools to be used are as follows;

- JMeter - JMeter can be used to target the overall service and individual microservice components to trigger scaling
- JUnit with DBunit - DBunit is a plugin for JUnit that can be used to run performance tests against database instances
- Kube-burner - A Kubernetes specific performance and scale testing suite with Prometheus metric collection

Testing will be repeated multiple times to account for variability in performance and to identify any anomalies. Moving averages will be calculated for each metric to provide a clearer picture of the average performance of both platforms. The study will perform testing over an extended period of time with the aim of minimising the impact of changing network conditions, workload spikes and other transient factors on the data collected.

E. Conclusion and interpretation

Interpretation of the results will be provided, highlighting key differences, strengths and limitations of each platform. The experimental setup aims to provide a fair comparison between the two platforms, creating meaningful insights into their respective abilities.

REFERENCES

Füstös, F., Péter, K., László, N.-P., Mátis, S.-G., Szabó, Z. & Sulyok, C. (2022), Managing a kubernetes cluster on raspberry pi devices, *in* '2022 IEEE 20th Jubilee International Symposium on Intelligent Systems and Informatics (SISY)', IEEE, pp. 133–138.

Goedtel, M. (2024), 'Concepts - kubernetes basics for azure kubernetes services (aks) - azure kubernetes service'. Accessed: 01/05/2024.
URL: *https://learn.microsoft.com/en-us/azure/aks/concepts-clusters-workloads*

Magnani, S., Risso, F. & Siracusa, D. (2022), 'A control plane enabling automated and fully adaptive network traffic monitoring with ebpf', *IEEE Access* **10**, 90778–90791.

Rancher, L. (2023), 'K3s - lightweight kubernetes — k3s'. Accessed: 17/04/2024.
URL: *https://docs.k3s.io/*

Sethy, K. K., Singh, D., Biswal, A. K. & Sahoo, S. (2022), Serverless implementation of data wizard application using azure kubernetes service and docker, *in* '2022 1st IEEE International Conference on Industrial Electronics: Developments & Applications (ICIDeA)', IEEE, pp. 214–219.

Sindhu & Pavithra (2020), 'Deploying a kubernetes cluster with kubernetes-operation (kops) on aws cloud: Experiments and lessons learned', *International Journal of Engineering and Advanced Technology (IJEAT)* pp. 984–989.

Telenyk, S., Sopov, O., Zharikov, E. & Nowakowski, G. (2021), 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, pp. 313–317.