International Journal of Computer Trends and Technology
ISSN: 2231–2803 / https://doi.org/10.14445/22312803/IJCTT-V72I2P114

Volume 72 Issue 2, 75-81, February 2024
© 2024 Seventh Sense Research Group®

Original Article

Overcoming Challenges in Deploying Large Language Models for Generative AI Use Cases: The Role of Containers and Orchestration

Sriramaraju Sagi

NetApp Inc, US.

Corresponding Author : sagi.sriram@gmail.com

Received: 07 January 2024 Revised: 07 February 2024 Accepted: 19 February 2024 Published: 29 February 2024

***Abstract** - This research delves into using Language Models (LLMs) in converged infrastructure, specifically focusing on container technologies like Kubernetes and OpenShift for orchestration purposes. The passage discusses the challenges involved in implementing LLMs, including scalability, performance issues and security considerations. It suggests that containers can effectively address these challenges. Additionally, it explores the benefits of using containers to deploy LLMs, such as scalability, optimized resource utilization, enhanced flexibility, increased portability, and strengthened security measures. Furthermore, it examines how Suse Rancher plays a role in managing applications that are containerized to ensure both security and scalability. The validation and analysis section provides an assessment of a study that utilizes an infrastructure platform called FlexPod to evaluate LLM models across container orchestration platforms, demonstrating the practicality and advantages of integrating FlexPod Datacenter.*

***Keywords** - Large Language Models (LLM), Containerization, Scalability, Datacenter, Kubernetes.*

1. Introduction

Sophisticated artificial intelligence systems, known as Large Language Models (LLMs), have undergone training using vast amounts of written material. This training allows them to understand and generate language that closely resembles communication. By utilizing deep learning algorithms, LLMs can. Comprehend language patterns and structures, enabling them to produce coherent and contextually appropriate text. These models possess a range of capabilities, such as answering questions, completing sentences and even generating documents. LLMs find applications in domains like language translation, content creation and chatbot development. As researchers continuously refine their training methods and algorithms for LLMs, these models continue to evolve and improve.

The deployment of language models for AI purposes has become increasingly essential in the field of artificial intelligence. These models have the ability to generate text that sounds natural and human, making them useful in applications such as chatbots, language translation and content creation. However, it is crucial to use LLMs with caution since they are powerful tools. Proper training and fine-tuning are necessary to avoid biases and inaccuracies in the generated text. Moreover, it is essential to take into account the concerns associated with the utilization of LLMs. This includes assessing the possibility of misuse and the potential effects on employment within particular industries. Fundamentally employing LLMs for AI purposes holds importance because they have the ability to improve and automate language related tasks while ensuring a high standard of excellence and precision.

Deploying language models (LLMs) presents challenges, including scalability, performance and security. One of the difficulties in LLM deployment revolves around scalability since these models require computational resources to function effectively. Moreover, response times can be slow, leading to an impact on user experience. To overcome scalability and performance concerns, organizations may need to invest in hardware and optimize their software to handle the demands of LLMs efficiently. Another challenge lies in ensuring the security of deployed LLMs, as they can be vulnerable to attacks that compromise integrity and output, resulting in consequences. Organizations must implement access controls, monitor activity, and regularly conduct security audits to mitigate these risks.

Furthermore, regulatory compliance is crucial when deploying LLMs due to data privacy and security requirements. Organizations must guarantee adherence to all regulations before deploying an LLM, as noncompliance can lead to fines and legal repercussions. Despite these challenges, successful deployment of LLMs is achievable with strategies and investments. By addressing scalability concerns, optimizing performance factors, and prioritizing security measures while also ensuring compliance at all levels, organizations can ensure effective usage of their LLMs while maintaining secure operations within the boundaries set by relevant regulations.

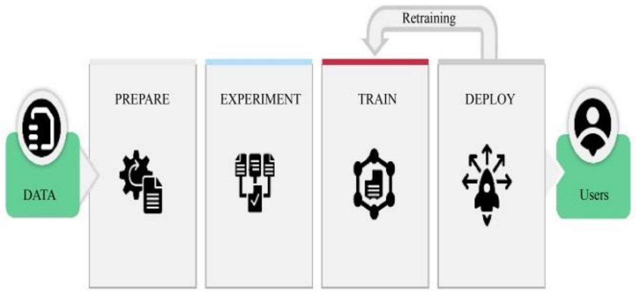

Fig. 1 Typical LLM lifecycle

2. Containers for AI LLM Models

The use of language models (LLMs) in training, fine-tuning and inference tasks presents a significant challenge in the field of artificial intelligence (AI) and machine learning (ML) due to their complex nature, computational requirements and scalability needs. Containers, those managed by platforms like Kubernetes and OpenShift, play a role in overcoming these challenges.

The advent of LLMs has brought about a transformation in AI and ML, offering capabilities in natural language processing, text generation and comprehension. However, implementing these models comes with its set of hurdles, such as the need for computing resources, efficient workload scaling and the demand for resilient, adaptable and secure development and deployment environments. Containers managed by platforms like Kubernetes and OpenShift have emerged as solutions to address these challenges.

Containers offer an advantage when it comes to scalability in LLM deployment. They bring together the application and its dependencies into an entity, making scalability easier and more efficient. Kubernetes and OpenShift take this advantage further by automating the deployment, scaling and management of applications. This is particularly beneficial for LLMs because their resource requirements can vary significantly between the training phase, which demands processing power and inference, which may require scaling out across instances to handle simultaneous requests. One of the benefits of using containers is improved resource efficiency. By allowing containerized applications to share the operating system, kernel overhead is minimized. This is crucial for LLMs as they have demanding memory needs. Kubernetes and OpenShift enhance resource utilization by utilizing scheduling algorithms that effectively distribute workloads across the cluster to make the most out of resources.

The containerized approach provides flexibility and ease of use. Developers can package LLMs (Logical Link Managers) and their dependencies into containers, ensuring consistency across environments and simplifying the transition between development, testing and production setups. Kubernetes and OpenShift add a layer of abstraction that hides the underlying infrastructure details, making it easy to deploy LLMs on premises in the cloud or systems without any complications.

Containers offer a way to iterate and develop LLMs. Developers can make changes within containers without affecting the application, allowing for the adoption of continuous integration and continuous deployment (CI/CD) practices. Kubernetes and OpenShift further enhance this capability by automating deployment procedures, facilitating testing, and implementing upgrades or new features.

Training and fine-tuning language models (LLMs) require computational resources, often leading to the need for distributed computing strategies. Kubernetes or OpenShift-managed containers simplify the distribution of these tasks across nodes, effectively managing the workload and optimizing hardware utilization. Managing iterations and versions of LLMs can be complex due to model development and training. However, containers streamline version control by encapsulating model versions, while Kubernetes and OpenShift provide rollback capabilities to revert to versions if necessary, enhancing reliability and ease of management.

When deploying LLMs in tenant environments, ensuring security is crucial. Containers offer a level of isolation between applications, minimizing the risk of interference or breaches. Kubernetes and OpenShift further enhance security through features like namespace isolation, network policies and secure access controls. This ensures that LLM deployments are safeguarded at levels. The deployment of language models presents challenges that require robust, scalable and efficient solutions. Containers orchestrated by systems such as Kubernetes and OpenShift provide a solution by offering scalability, resource efficiency, flexibility and enhanced security as AI and ML progress containers will continue to play a role in deploying and managing LLMs. They are essential for enabling dependable and efficient AI applications.

Suse Rancher is a container management system that can handle the complexities of deploying language models (LLMs). It provides a scalable infrastructure for running containerized applications, including LLMs. Suse Rancher ensures operations with features like orchestration, resource management, and high availability. To enhance security, it incorporates measures such as access controls and vulnerability scanning to protect against access and potential attacks on LLM deployments. Integrating with Kubernetes Suse Rancher enables scaling and efficient utilization of resources based on workload demands. Additionally, it ensures compatibility with cloud providers to deploy LLMs in hybrid or multi-cloud environments while maintaining unified management and security protocols.

This paper aims to explore infrastructure solutions for deploying LLMs using Kubernetes clusters. It specifically examines the benefits of utilizing a converged infrastructure solution like FlexPod Datacenter in conjunction with Kubernetes.

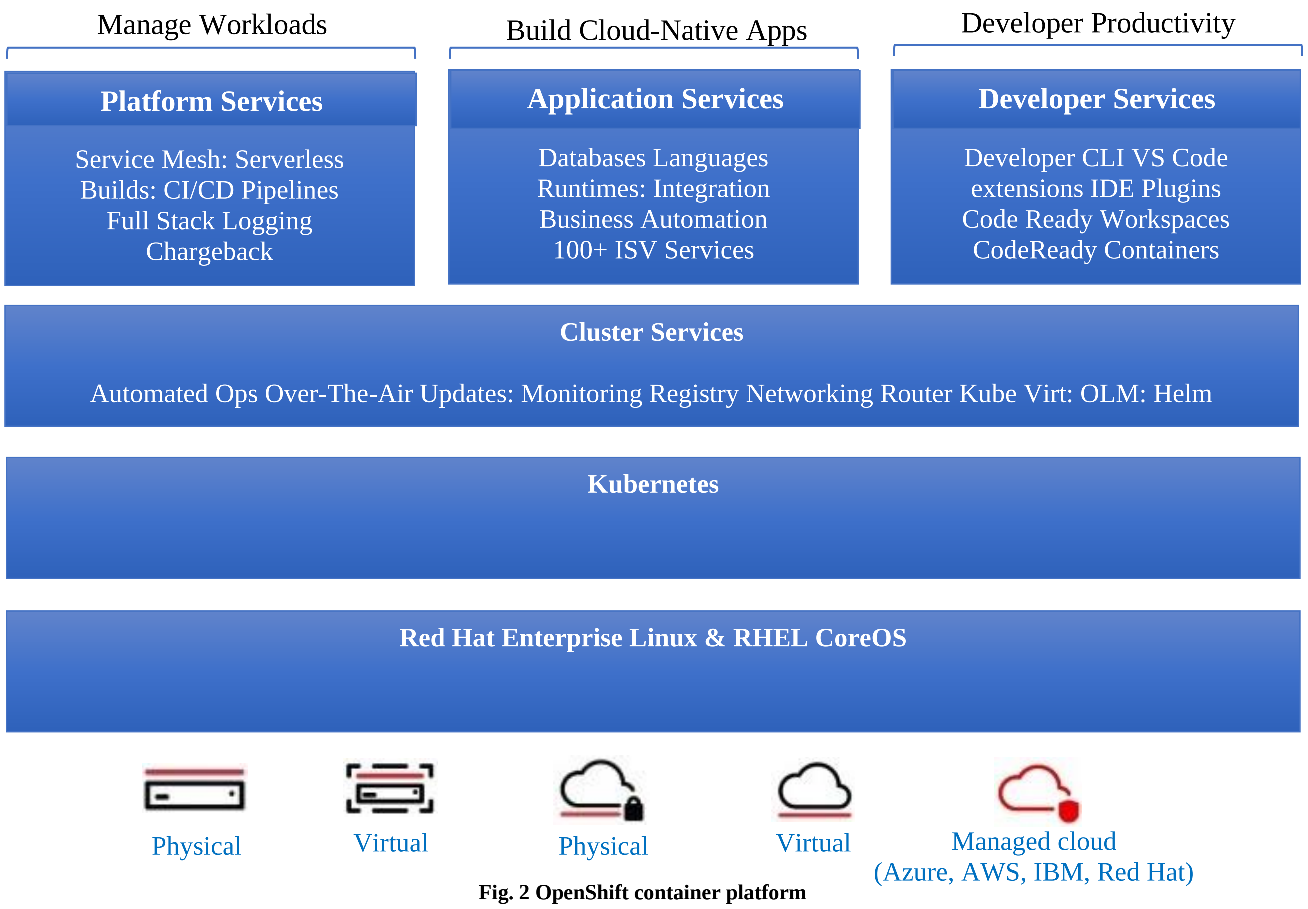
Manage Workloads
Build Cloud-Native Apps
Developer Productivity
Platform Services
Service Mesh: Serverless Builds: CI/CD Pipelines Full Stack Logging Chargeback
Application Services
Databases Languages Runtimes: Integration Business Automation 100+ ISV Services
Developer Services
Developer CLI VS Code extensions IDE Plugins Code Ready Workspaces CodeReady Containers
Cluster Services
Automated Ops Over-The-Air Updates: Monitoring Registry Networking Router Kube Virt: OLM: Helm
Kubernetes
Red Hat Enterprise Linux & RHEL CoreOS
Physical
Virtual
Physical
Virtual
Managed cloud (Azure, AWS, IBM, Red Hat)

Fig. 2 OpenShift container platform

3. Literature Review

Several studies have explored the utilization of intelligence (AI) in Kubernetes with a focus on improving container scheduling (Jo Martinez, 2021). The research paper introduces a strategy called Kubernetes Container Scheduling Strategy (KCSS), which leverages AI to enhance scheduling efficiency and minimize costs for workloads and services. KCSS utilizes AI to enable multicriteria node selection, allowing the scheduler to gain an understanding of both the state of the cloud and the user's requirements. Additionally, the AI Scheduler empowers customers to effectively utilize GPUs, a whole number of GPUs, and multiple GPU nodes for distributed training on Kubernetes.

In another study by Toka (2020), a scaling engine for Kubernetes that offers auto-scaling decisions specifically designed to handle requests is presented. The effectiveness of this forecast scaling engine and its proposed management parameter is evaluated through simulations and measurements using collected web traces. The results showcase improved resource allocation in response to service demand. By implementing this auto-scaling engine in Kubernetes, there is a reduction in dropped requests while requiring higher resource allocation compared to the default baseline.

In a paper written by Thurgood (2019), the focus is on discussing how well Kubernetes can handle workloads that rely on AI applications. The paper also introduces a method that uses Free and Open-Source Software (FOSS) to expand Kubernetes worker nodes, making them more efficient in handling these workloads. The challenges related to scalability and security are also addressed in the paper.

The main findings from Dang Quang's research (2021) suggest using a custom autoscaler, which proves to be more accurate and faster than the default HPA. Additionally, their Bi LSTM prediction model outperforms others.

Wus study (2020) reveals that humans' manual allocation of GPU resources is inefficient. To address this issue, an automatic machine-learning platform was developed to handle resource management and job allocation. Experiments were conducted to validate the feasibility of this platform.

Lees research (2020) highlights discoveries. These include proposing a flexible and scalable machine learning framework built on Kubernetes, which effectively utilizes resources. The framework also incorporates functionalities that streamline the modeling process for machine learning developers. Empirical case studies provide evidence of the platform's effectiveness in overcoming obstacles in machine learning.

Various research studies have been conducted to explore methods in the realm of AI and Kubernetes. These include using AI-driven forecasting techniques for auto-scaling, implementing an Open-Source Software (FOSS) solution to adjust the number of Kubernetes worker nodes and developing a custom autoscaler that employs a deep neural network model. One proposed framework called KubeEdge.AI aims to leverage KubeEdge for edge AI applications. Additionally, the KFServing project is being considered as a serverless option for machine learning inference. Collectively, these studies demonstrate how AI can significantly improve resource utilization, performance, and scalability on Kubernetes. By utilizing platforms like KubeEdge.AI and exploring options such as the KFServing project, organizations can further enhance their capabilities in edge AI. Streamline machine learning inference processes. These studies emphasize the evolution of AI technologies within the Kubernetes ecosystem, paving the way for efficient and intelligent applications across various industries.

4. Validation and Analysis

In this study, we used an infrastructure platform called FlexPod. It combines computing resources from Cisco and storage resources from NetApp. We tested LLM models on container orchestration platforms like RedHat OpenShift and Suse Rancher. During our testing, we provided customers with ways to incorporate the core infrastructure layer and its automation, as shown in the accompanying image. We demonstrated the deployment of the Kubernetes container platform along with integrating software using AI framework management tools for its lifecycle management on top of the infrastructure layer. The FlexPod Datacenter solution is an approved approach for implementing Cisco and NetApp technology and products to build a shared private and public cloud infrastructure.

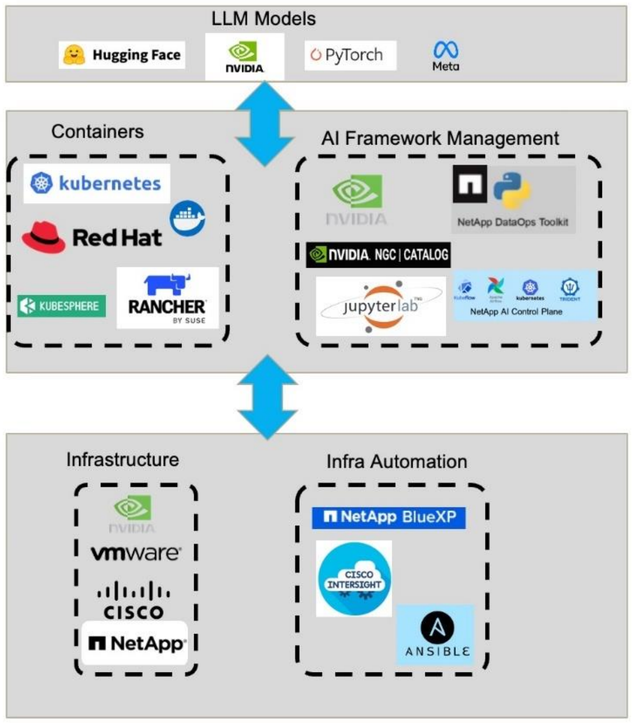

Fig. 3 Typical End-to-End Infrastructure with containers

The FlexPod Datacenter solution is an authorized method for implementing Cisco and NetApp technology and products to construct a communal private and public cloud infrastructure. Cisco and NetApp have collaborated to provide a range of FlexPod solutions that empower important data center platforms. The integration of SUSE Rancher ECM with the FlexPod solution streamlines the process of deploying and managing the container system. The integration of Ansible with the FlexPod solution automates the process of deploying the FlexPod infrastructure and installing SUSE Rancher. This allows customers to efficiently program and automate the infrastructure on a large scale, providing flexibility and extending the advantages of automation to the entire system. The FlexPod Datacenter for the SUSE Rancher Enterprise Container Management system provides the following significant advantages for customers:

- An all-encompassing solution that facilitates the complete SUSE software-defined Linux and Kubernetes stack for containerized and AI/ML applications.

- A standardized framework that enables fast, consistent, and error-free deployments of workload domains based on FlexPod.
- Automated life cycle management ensures that all system components are kept up to date.
- Streamlined cloud-based administration of diverse FlexPod components.
- The design is modular and can easily integrate with hybrid cloud environments. It is also driven by policies.
- The FlexPod architecture is designed to be highly available, adaptable, and scalable.
- The cooperative support model and Cisco Solution Support
- This design is straightforward to implement, use, and oversee, following the recommended practices and compatibility criteria of Cisco, NetApp, and SUSE.
- Support monitoring individual components, automating and orchestrating solutions, and optimizing workloads.

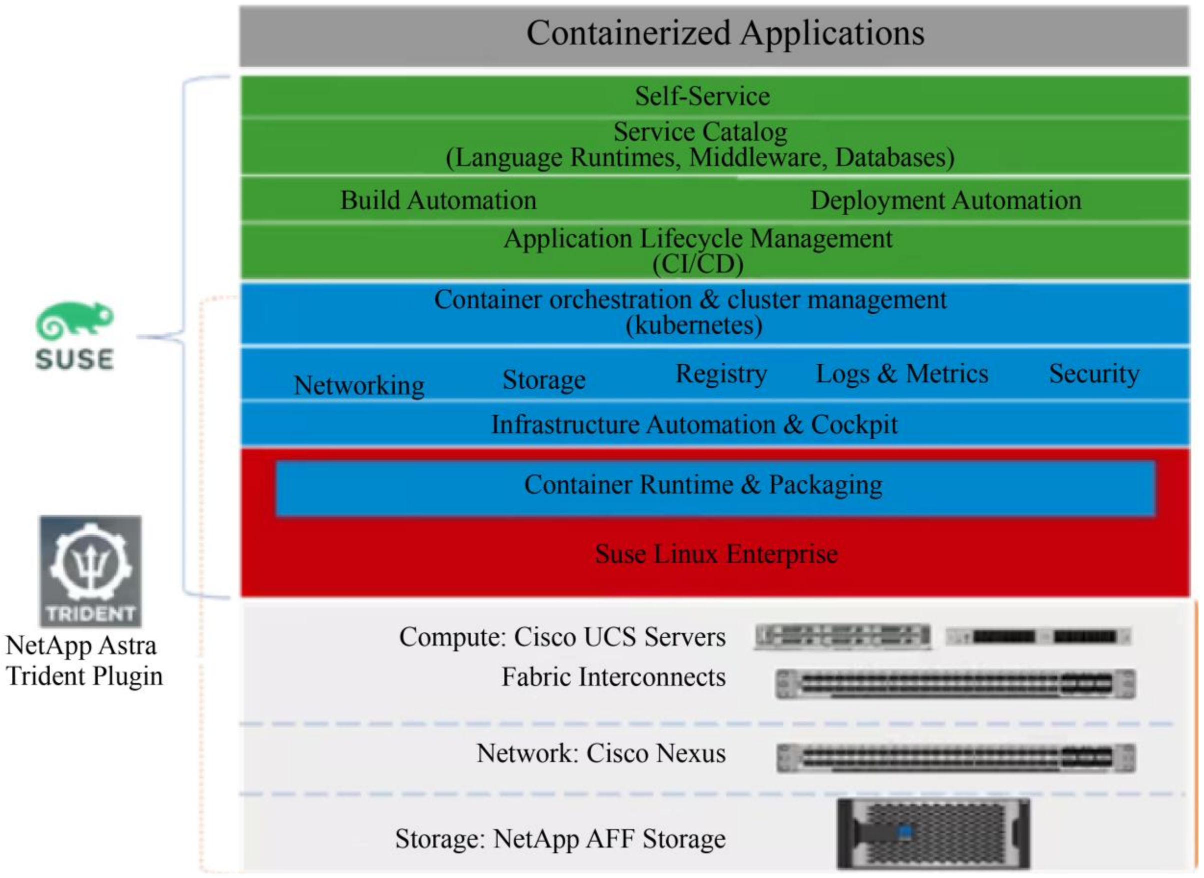
Containerized Applications
Self-Service
Service Catalog
(Language Runtimes, Middleware, Databases)
Build Automation
Deployment Automation
Application Lifecycle Management
(CI/CD)
Container orchestration & cluster management
(kubernetes)
Networking
Storage
Registry
Logs & Metrics
Security
Infrastructure Automation & Cockpit
Container Runtime & Packaging
Suse Linux Enterprise
SUSE
TRIDENT
NetApp Astra
Trident Plugin
Compute: Cisco UCS Servers
Fabric Interconnects
Network: Cisco Nexus
Storage: NetApp AFF Storage

Fig. 4 Container architecture with Cisco and NetApp Infrastructure

Figure 5 below depicts the logical structure of the FlexPod Datacenter with SUSE Rancher, showcasing the utilization of computing, network, and storage resources on FlexPod through RKE2 components. The Cisco Nexus 9000 series switches within FlexPod provide the necessary storage and network connection for the SUSE RKE2 Cluster nodes running on Cisco UCS servers.

SUSE Rancher with Kubernetes simplifies storage management by abstracting the intricacies of storage provisioning and consumption, making persistent storage an essential component for operating stateful containers. The persistent volumes used for containers can either be static or dynamically provisioned. In this particular scenario, the persistent volumes are dynamically provisioned using FlexPod and are facilitated by the NetApp Astra Trident CSI Driver. Dynamic volume provisioning enables the creation of storage volumes as needed. NetApp Astra Trident CSI eliminates the requirement to pre-allocate storage for containers and enables the provisioning of persistent storage during container deployment. This solution employed NFS and iSCSI storage to facilitate dynamic storage provisioning.

SUSE Rancher employs a software-defined networking (SDN) methodology to establish a cohesive cluster network, facilitating communication between pods throughout the SUSE RKE2 cluster. The SUSE RKE2 SDN establishes and maintains this pod network, configuring an overlay network with Open vSwitch (OVS). The RKE2 SDN system utilizes

Open vSwitch (OVS) as its default framework. RKE2 offers the cluster administrator the option to deploy using either one of the native SDN plug-ins provided by RKE2 or a third-party SDN from the supported ecosystem, such as Cisco ACI. We utilized the RKE2 native SDN plug-in, specifically the OVN-Kubernetes, to implement this solution.

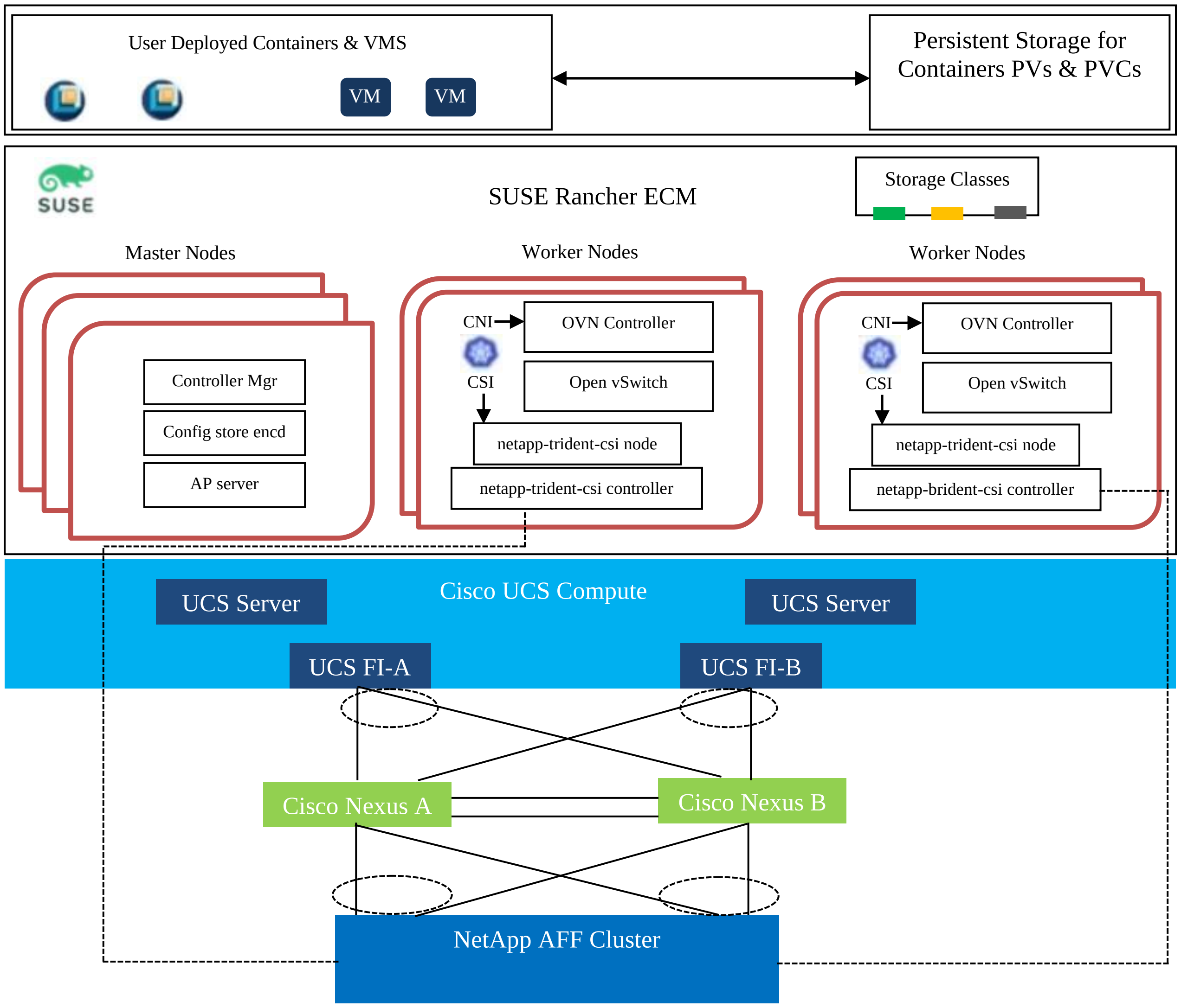

Fig. 5 FlexPod Topology diagram with SUSE Rancher

5. Conclusion

This article delves deeply into the potential of containerization technology in software deployment. The research extensively explores the architecture of containers, compares them to virtualization, and rigorously analyzes their performance, scalability and security. The findings strongly support the adoption of containerization, highlighting its ability to enhance deployment processes and operational flexibility. The study also emphasizes the importance of optimizing and exploring container ecosystems to address emerging challenges.

Fully leverage the benefits of this technology. In conclusion, this summary effectively captures the essence of the study and provides a looking perspective on containerization technology.

6. Acknowledgements

This article was supported by the technical team from Cisco and NetApp, Ulrich Kleidon, Cisco, Abhinav Sing, and NetApp; we sincerely thank them for their contribution and valuable input.

References

[1] FlexPod Datacenter with SUSE Rancher for AI Workloads Design Guide, NetApp, Cisco, 2023. [Online]. Available: https://www.cisco.com/c/en/us/td/docs/unified_computing/ucs/UCS_CVDs/flexpod_suse_rancher_design.html

[2] Diaz Jorge-Martinez et al., "Artificial Intelligence-based Kubernetes Container for Scheduling Nodes of Energy Composition," *International Journal of System Assurance Engineering and Management*, pp. 1-9, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[3] Laszlo Toka et al., "Adaptive AI-based Auto-Scaling for Kubernetes," *20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*, Melbourne, VIC, Australia, pp. 599-608, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[4] Brandon Thurgood, and Ruth G. Lennon, "Cloud Computing With Kubernetes Cluster Elastic Scaling," *Proceedings of the 3rd International Conference on Future Networks and Distributed Systems*, Paris France, pp. 1-7, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[5] Nhat-Minh Dang-Quang, and Myungsik Yoo, "Deep Learning-Based Autoscaling Using Bidirectional Long Short-Term Memory for Kubernetes," *Applied Sciences*, vol. 11, no. 9, pp. 1-25, 2021. [CrossRef] [Google Scholar] [Publisher Link]

[6] Chaoyu Wu, E Haihong, and Meina Song, "An Automatic Artificial Intelligence Training Platform Based on Kubernetes," *Proceedings of the 2020 2nd International Conference on Big Data Engineering and Technology*, Singapore China, pp. 58-62, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[7] Chun-Hsiang Lee et al., "Multi-Tenant Machine Learning Platform Based on Kubernetes," *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, pp. 5-12, 2020. [CrossRef] [Google Scholar] [Publisher Link]