An Overview of OpenShift Architecture and Enterprise Container Platform Management

Dharmendra Ahuja
DevOps Engineer
IBM
dharmendradevops11@gmail.com

To Cite this Article
Dharmendra Ahuj, "An Overview of OpenShift Architecture and Enterprise Container Platform Management", Journal of Science Engineering Technology and Management Science, Vol. 01, Issue 1, December 2024,pp: 224-232, DOI: http://doi.org/10.64771/jsetms.2024.v01.i01.pp224-232
Submitted: 09-11-2024 *Accepted: 17-12-2024* *Published: 24-12-2024*

***Abstract*—OpenShift has become one of the enterprise container platforms since it has developed Kubernetes to support security, automation, and lifecycle management. The paper is a detailed introduction to OpenShift architecture and its application in the management of enterprise container platforms. It analyses the heart of OpenShift architecture, such as the control plane, worker nodes and their role in enabling scalable, resilient and production-ready containerized environments, along with networking and storage, as well as security functions. The paper also covers the various deployment models such as public, private, hybrid and community cloud, and examines how OpenShift can be used to sustain various requirements of an enterprise within each model. The most important parts of enterprise container platform management, including orchestration, monitoring, observability, application lifecycle management, and automation, are discussed to present efficiency and reliability in operations. Lastly, the paper discusses the enterprise adoption situations and applications, highlighting the role of OpenShift in contemporary application development, microservice scalability, cost-effectiveness, and system reliability in cloud-native and hybrid cloud environments.**

***Keywords*—OpenShift, Kubernetes, Containerization, Enterprise Container Platform, Cloud Deployment Models, Monitoring and Observability.**

I. INTRODUCTION

Containerization has become a powerful technology to construct practicable, transportable and safe computing infrastructures in the cloud, edge, and enterprise space [1][2]. Its natural benefits, which include lightweight virtualization, fault isolation, portability, and enhanced security, have given it a choice of deployment in contemporary distributed systems including, including smart cities, intelligent buildings, industrial automation, and smart grids [3]. Containers allow unrestricted consistency in the deployment of applications and their dependencies by isolating them into their own runtime environments and running them on multiple hardware and operating system platforms.

An effective implementation and management of containerized applications, in its turn, entails the use of sophisticated orchestration systems with the ability to manage resource distribution, scalability, high availability, fault resilience, and load balancing [4]. Out of the existing container orchestrator options, Kubernetes (K8S) has become the new standard, thanks to its scalability, versatile scheduling, and large-scale distributed workloads [5][6]. Kubernetes makes it easy to deploy applications and provide lifecycle management, scale up dynamically and service resilience, which is appropriate in enterprise and edge computing infrastructure.

Enterprise systems in Industry 4.0 are becoming more and more heterogeneous in terms of computing devices with different operating systems, runtime environments, and resource constraints [7]. The proposed solution to these problems is containerization technologies that allow isolated and standardized execution environments on distributed nodes. Software containers started as a large-scale cloud computing concept have been valuable in the management of complex cyber physical systems (CPSs) through an increase in modularity, interoperability, and concurrent development of sub systems [8][9]. The specific features are significant to the enterprise platforms that should be able to host both legacy and cloud-native applications.

Although Kubernetes offers the fundamental orchestration capability, enterprise environments require winning features like built-in security measures, multi-tenancy, automatic lifecycle management, observability and control [10]. Red Hat OpenShift builds on Kubernetes by providing an enterprise-ready container platform that brings together orchestration, DevOps preservation, security execution, networking, storage and observation together in a single architecture[11]. OpenShift is compatible with hybrid and multi-cloud deployments, which allows organizations to operate containerized workloads uniformly across on-premises, cloud and edge networks.

Enterprise container platform management has developed observability as a pressing necessity. Cloud native Computing Foundation (CNCF) defines observability and analysis as the key elements of the maturity of cloud-native systems [12]. Observability can be described as a property based on control theory, and it is the property to deduce the internal state of a system based on the external outputs of the system [13][14]. Observability is achieved in the cloud-native platforms by default, consisting of metrics, logging, and distributed tracing, which would allow real-time monitoring, conduct performance analysis, and diagnose

faults. OpenShift has the in-built observability tooling, which facilitates proactive system administration and operational reliability on a large scale.

A. Structure of the Paper

This paper is structured in the following way. Section II provides an overview of OpenShift architecture, including its components, control plane, worker nodes, and deployment models. Section III discusses enterprise container platform management, covering container technology principles, container orchestration, platform architecture design, and application lifecycle management. Section IV presents use cases and enterprise adoption scenarios, emphasizing the impact of OpenShift on application performance, scalability, reliability, and cost efficiency. The literature related to the study is reviewed in Section V. Finally, Section VI outlines conclusions and future research directions, focusing on advancements in multi-cloud integration, AI-driven automation, and performance optimization for enterprise container platforms.

II. OpenShift Architecture Overview

Red Hat created a suite of containerization software tools called OpenShift. It is the cloud development platform as a service, which is hosted on RedHat Enterprise Linux (RHEL). RedHat offers production ready OpenShift where it can get support from RedHat, and there is another open-source community version of OpenShift called OKD.

Developers can create, test, deploy, and use their applications on-premises or in the cloud with the aid of OpenShift, which also includes a full-stack automated software life cycle solution [15]. The core of OpenShift is based on Kubernetes to manage the containers so Kubernetes is the kernel and OpenShift is the distribution with some additional functionality and features.

A. OpenShift vs Kubernetes

The main contrast between OpenShift and Kubernetes is that OpenShift is a “product” with a service level agreement (SLA) that allows us to receive enterprise support from RedHat, whereas Kubernetes is an open source “project” that receives contributions from a community of developers. The main differences between OpenShift and Kubernetes are as follows.

1) Support

RedHat provides enterprise support for OpenShift for Enterprise OpenShift, however, there is relatively little community support for OpenShift (OKD) in general. As opposed to this, the developer community for Kubernetes is large and active, and it is actively working together to improve the platform.

2) Security

While Kubernetes users must perform the authentication procedure manually, OpenShift offers a robust security policy and built-in authentication mechanisms. OpenShift offers a secure-by-default option that prevents us from running a container as root by default, improving security.

3) Runtime

There is no need for docker daemons in master or worker nodes in OpenShift version 4 (v4). It increases the cluster’s security. Whereas Docker is being dropped, Kubernetes 1.24 and later use Containerd as a runtime instead.

4) Image Registry

Kubernetes does not have its image registry, it can use a private registry or create their docker registry. OpenShift, on the other hand, has its built-in image registry, and it can manage container images with “ImageStreams” on OpenShift.

5) Networking

The key parameter determining the range of security is networking. OpenShift has its networking solution, Open vSwitch, which includes three plug-ins. The three plug-ins are as follows:

- OVS- subnet,
- OVS – multitenant,
- OVS – network policy.

As opposed to Kubernetes, which relies on external networking products such as Calico and WaveNet.

6) Templates

Kubernetes uses Helm templates, which are adaptable and simple to deploy and upgrade. While OpenShift templates are somewhat complex, it lacks package versioning.

7) Releases and Updates

OpenShift has three releases per year, whereas Kubernetes has four releases each year on average. Multiple concurrent updates are supported by Kubernetes, but not by OpenShift DeploymentConfig.

8) Graphical User Interface

Kubernetes has a complicated web interface, and users must install the GUI for Kubernetes elements. This may be burdensome for new users, and users must perform the manual authentication process. OpenShift, on the other hand, gives a simple login-style console to graphically control and monitor the cluster. OpenShift Hybrid Cloud Interface, which allows us to create clusters in various cloud native environments or on-premises environments, is the key feature of the OpenShift console. From there, it can manage and monitor the resources.

B. Deployment Models

1) Public Cloud

The term "public cloud" refers to a specific deployment architecture wherein several businesses share and access cloud services over the public internet. Third-party cloud service providers are in charge of managing the platforms, apps, and infrastructure. There are a number of examples, such as AWS, Azure, and GCP (Google Cloud Platform). One advantage of public cloud computing is ability to host websites and handle large amounts of data using services like Google BigQuery and Amazon Redshift [16]. Additionally, it offers content delivery networks such as Azure FrontDoor and AWS CloudFront. This enables effortless scalability to accommodate demand, is cost-effective due to its pay-as-you-go model, and is maintenance-free.

2) Private Cloud

A private cloud is one that is created within a company and made accessible solely to its employees, giving them sole access to certain computer resources [17]. Additionally, A private cloud might be on-site or hosted by a third party. Resources are allocated to a particular company and developed to satisfy highly specific organizational needs. Delivers enhanced regulatory compliance and security. VMware vSphere is an example of a comprehensive suite for administering virtualized environments.

3) Hybrid Cloud

The term "hybrid cloud" refers to a system that allows data and applications to move freely between public. This approach has additional alternatives for deployment and maximum flexibility. Together, public and private clouds are better served by this paradigm [18][19]. Together, on-premises and cloud resources may be seamlessly integrated. Provides access to elastic public cloud resources even as it keeps mission-critical applications hosted on private clouds. Achieving the necessary balance between security and scalability is aided by storing sensitive data on private clouds and less sensitive data on public clouds.

4) Community Cloud

Organizations that use same cloud infrastructure and have comparable concerns, such security, compliance, or jurisdiction, share community clouds. It entails the collective of organizations with comparable objectives and the sharing of infrastructure. This enables for cooperation across the various companies and share the expenses which will lessen the burden of each entity [20]. Common regulatory needs are taken into account while designing the compliance requirements.

C. Architecture of OpenShift

Red Hat OpenShift is a container orchestration enterprise-grade platform created and supported by Red Hat based on Kubernetes. It also adds infrastructure management, security controls, networking and operational tooling to the core Kubernetes features in a single platform, thus giving developers and operations-users a whole and production-ready experience. The general structure of OpenShift is depicted in Fig. 1, as it shows the layered and modular nature of the system in terms of control, compute, networking, and storage units. The OpenShift architecture revolves around the control plane, and it is based on the concept of the Kubernetes master node and serves as the central management point within the cluster [21]. The control plane performs vital orchestration, such as allocation of workloads, cluster resource management, service delivery, and load balancing. The API server is at center of the control plane and it serves as the main entryway of both internal and external communications [22]. It authenticates requests using authentication and authorization policies and permits any operation in the cluster. Another crucial element of the control plane is the distributed key-value store, etc. It is in charge of maintaining the cluster's configuration and status information and offers consistency, fault tolerance, and high availability. It is the backbone of dependable container orchestration. It is critical to the integrity of the cluster operation of the state persistence and synchronization of state across the control plane components.

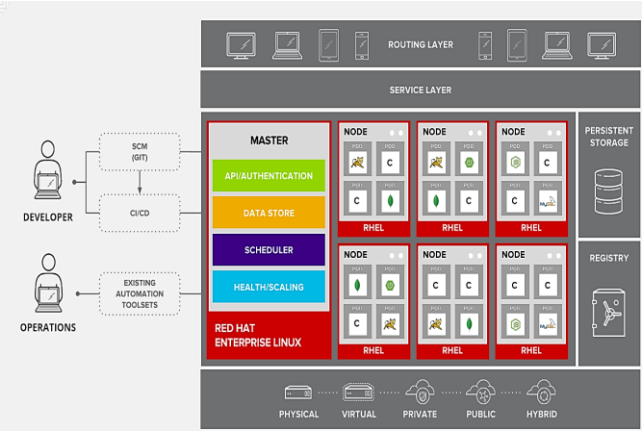

Fig. 1. Architecture of OpenShift.

There are containerized apps and services running on these worker nodes. The control plane manages each worker node, which has application pods. The OpenShift platform upgrades scalability and adds or removes worker nodes to handle the dynamic workload demand, whether in physical, virtual, private, public or hybrid environments. The kubelet is a little agent that executes on each worker node; it allows the control plane to communicate with the worker nodes. As a part of control plane, the kubelet is in charge of making the containers operate as expected. The nodes also have a container runtime engine. This runtime

complies with the standards of the Open Container Initiative (OCI), which means that containers are portable, secure, and interoperable regardless of the environment.

D. Security and Compliance Features

Security is a fundamental component of OpenShift, which includes secure container image scanning, SELinux enforcement, and Role-Based Access Control (RBAC). The platform guarantees adherence to regulatory mandates in the finance, healthcare, and government sectors. Vulnerability management, automated policy enforcement, and network segmentation are features that provide better security with less operational overhead. Scalability is made possible by OpenShift's integrated authentication and encryption technologies, which allow businesses to retain uniform security standards across hybrid cloud infrastructures while protecting critical data and tasks.

III. ENTERPRISE CONTAINER PLATFORM MANAGEMENT

Enterprise container platforms must have extensive management functions, which can guarantee scalability, security, availability, and operational efficiency. Red Hat OpenShift offers built-in administration capabilities that assist the complete life cycle of clusters, applications, and distributed systems on hybrid and multi-cloud systems.

A. Container Technology Principles

Container technology allows applications to be bundled as deployable, self-contained and ready-to-run units which include application code, necessary libraries, binaries and, where needed, middleware and business logic. Docker and other container engines give an application the ability to move across the infrastructure and remain consistent. Containers in multi-tier and distributed applications have to have dependencies explicitly controlled [23]. This is done in terms of orchestration mechanisms that characterize the application components, dependencies and lifecycle policies. PaaS environments use those orchestration plans to assemble, scale, and manage containerized applications automatically and therefore provide dependable and repeatable application delivery.

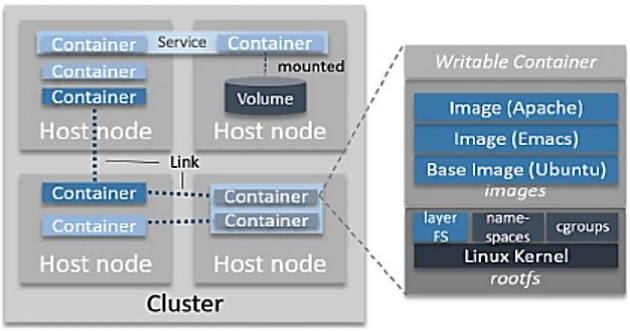

Fig. 2. Container Cluster Architectures.

The majority of solutions based on containers use Linux container (LXC) technologies, which rely on the capabilities of key features of the kernel, namespaces and control groups, to isolate processes on a common operating system. Docker which is one of the most widely used container platforms uses a layered image architecture that stacks file systems with union file systems, and this is shown in Fig. 2. Base image layers can be read only and reused by multiple containers, with a writable layer being added at runtime to each instance of a container [24]. Containerization also enhances the shift towards single-host deployments being clustered so that a group of container hosts is linked together to create scalable clusters. Such clusters contain application services made of logically grouped containers that can be moved to nodes, with persistent volumes that offer long-lived storage of stateful workloads. Service discovery, inter-container communication, scaling, and lifecycle coordination should be orchestrated in order to manage such clustered environments effectively.

B. Container Platform and Enterprise Environment

A container platform, which is an implementation of LXC, offers an environment in which numerous programs may run independently on a single machine, as seen in Fig. 2. It accommodates the tasks, responsibilities, and needs that numerous LXC containers may perform. Data and applications may be contained in a static file called a container image. It might be user-generated or based on an existing container. The execution environment, which is a namespace, is comprised of containers, which are created when container images are executed via the Container Engine. As a result, files or application processes running in a container are separate from those running in other containers or hosts [25]. The Container Engine transforms the user-specified access file path from a virtual to a real path and initiates input/output processing in the kernel whenever a user opens a container file. It can statically assign resources if needed, but the Container Engine does a good job of dynamically allocating system resources to containers by group. A host application and a container application both make use of the same set of system resources when the container platform is utilized in the way that is explained here.

A container-based information service is easy to construct and provides a substantial advantage over legacy systems in terms of management. The application can be utilized without the need for an installation procedure when a container image is executed with an application installed, thereby reducing the cost of construction [26]. Additionally, a single host executes multiple containers, and the application services operating in each container share the host's resources, as dynamic allocation of system resources is employed for these services. This enhances the efficacy of resource utilization.

C. Building Container Management Platform (CMP)

Container orchestration is the automated management of the entire lifecycle of containers. Scaling, health monitoring, resource sharing, load balancing, deployment, and provisioning of hosts are all topics that are being discussed. All of these little container management duties build up as the study expands, and before realizes it, a really complicated situation. That is the difficulty that container orchestration solutions are attempting to resolve [27]. An organization's CNCF architecture and tools for cloud workload deployment and administration need the development of a Container management platform so that teams can efficiently oversee the applications. CMP necessitates the integration of container cluster, monitoring, CMDB, change management, and security scanning tools in order to boost developer and operator efficiency. A container management platform's data flow architecture design is shown in Fig. 3 [28]. The platform includes an automated process for resolving issues, executive dashboards, self-service dashboards, application administration user interface, SRE metrics monitoring, alarms, and ML and AI capabilities. The data is transmitted to the central CMP Database from a variety of container orchestration servers, including Kubernetes, Mesos, Docker Swarm, and ECS. In the CMP database, information regarding Code Pull Requests from the Source Code repository, Change Management tools, Container Scanning tools, and CMDB is stored.

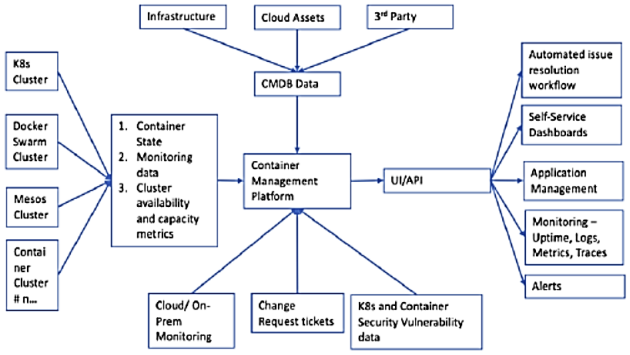

Fig. 3. Container Management Platform.

Data is sent to Datalake for further analysis and longer-term storage once it is accessible in the database. As seen in Fig. 4 below, datalike may be used by ML and LLMs to provide significant insights:

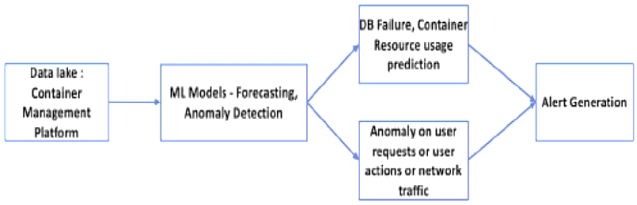

Fig. 4. ML models usage with CMP.

D. Platform Architecture Design

The platform uses a microservices architecture and is built on Docker container technology and Kubernetes container scheduling platform. The whole platform is divided into three main layers: infrastructure layer, service layer and application layer.

The infrastructure layer is the foundation of the entire platform, including computing resources, storage resources, and network resources. This layer provides virtualization technology to convert physical resources into virtual resources, improving resource utilization. At the same time, this layer also provides security mechanisms to ensure data security and privacy.

The service layer is core of the whole platform, including container scheduling, container orchestration, container service management and container network management modules. Container scheduling module is responsible for container scheduling and allocation, dynamically allocating containers according to application requirements and resource conditions [29]. The Container orchestration module is responsible for the orchestration of containers and dynamically generates container instances based on deployment configuration and running status of application. The container service management module is responsible for container service governance, including container monitoring, logging, configuration management, etc. The container network management module is responsible for container network management, including network planning, configuration, and monitoring.

The application layer includes various power enterprise applications, such as power marketing, power production, and power management. These applications can be deployed, scaled, and managed through a container cloud platform to meet the needs of different business scenarios.

E. Application Lifecycle Management

Modern applications encompass a variety of components, such as databases, caching systems, monitoring tools, event brokers, and more. Managing these subsystems demands specialized knowledge of performance tuning, security hardening, backup, and recovery procedures. Manual processes risk introducing human error and inconsistency [30]. Moreover, DevOps

practices frequently require continuous integration and deployment (CI/CD), further emphasizing the need for repeatable and automated processes. Operators fulfill this need by embedding subject-matter expertise directly into the cluster's control plane, monitoring application health, and taking corrective actions autonomously. This "intelligent automation" not only accelerates deployment but also ensures compliance with best practices.

F. Monitoring, Logging, and Observability

The reliability, performance, and availability of enterprise container platforms are critical, and with the assistance of monitoring, logging, and observability, they are guaranteed. These capabilities are offered by using built-in monitoring and logging systems in OpenShift environments that allow real-time access to containerized applications, components of a cluster, and resources of an infrastructure. Inbuilt metrics collection, log aggregation and alerting functionalities enable organizations identify anomalies, identify failures and improve the efficiency of the system throughout the application lifecycle [31]. Effective observability must also be able to integrate well with CI/CD pipelines to be able to correlate deployment activities with runtime behavior [32]. Nevertheless, monitoring and logging solutions effectiveness relies on the extent of integration, configuration and tool support which means that observability strategies need to be balanced with platform-native capabilities and organizational operational needs.

IV. Use Cases and Enterprise Adoption Scenarios

OpenShift is adopted by enterprises due to its capacity to support diverse workloads and also guarantee scalability, security and consistency when running. The platform is extensively utilized in the industries to facilitate cloud-native development of applications, legacy modernization, and implementation of data-intensive and smart workloads.

A. Impact on Enterprise Applications

Understanding the implications of expanding microservices on corporate applications is critical for businesses intending to embrace this architectural style. It investigates how scale impacts:

- **performance:** The capacity of the application to manage a higher load while remaining responsive.
- **Reliability:** The robustness of the system to errors and its capacity to bounce back from disturbances.
- **Cost Efficiency:** The financial consequences of scaling, which encompass operational costs and resource utilization.
- **Development and Maintenance:** The effects on iterative development, deployment, and maintenance procedures.

By analyzing these characteristics, the article intends to give a full knowledge of the advantages and trade-offs involved with expanding microservices in business contexts.

B. Use Cases and Adoption Scenarios

One factor strongly influencing enterprises' adoption of container platforms is the ability to support a wide range of scalability needs for both modern and legacy applications.

Horizontal scaling is especially desirable to applications that have unpredictable or a rapidly changing workload. Horizontal scaling is used to provide normal demand management in cloud-native web applications, microservices-based systems, and large-scale data processing platforms. One example is the seasonal or promotional traffic surge [33] on an e-commerce platform, which can dynamically scale containerized services by creating or destroying instances in response to real-time load to maintain performance stability and cost efficiency. This process is automated by container orchestration platforms like OpenShift by being able to support such features as horizontal pod autoscaling and cluster autoscaling. Vertical scaling, conversely, is more appropriate to enterprise applications whose growth patterns or workloads do not exhibit any predictable patterns and whose independent components are hard to identify.

Vertical scaling is commonly useful with legacy enterprise systems, databases that maintain state, and applications that require high inter-process communication by adding more CPU, memory or storage resources to the existing instances. As an example, a vertically scaled resource might be needed by an enterprise database that has payroll or financial reporting systems to ensure a steady level of performance and transactional integrity. Container platforms provide support to such uses by resource allocation controls and node-level scaling, allowing organizations to tradeoff between modernization and operation constraints.

V. Literature Review

As explained in Table I, the literature shows that the practices of containerization and orchestration technologies are essential to providing scalable, efficient, and reliable cloud, edge, and enterprise computing environments. The research results all point to Kubernetes-based platforms being robust and rich in features, which also note continued issues with performance optimization, automation, and management at scale.

Vanama (2024) an Architecture Led Cloud Modernization Framework (ALCMF) for transferring sophisticated business workloads from on-premises VMware infrastructures to Red Hat OpenShift operating on Amazon Web Services via Red Hat OpenShift Service on AWS (ROSA). The framework selects a modernization plan for each workload based on architectural and quality factors, rather than tactical rehosting techniques. It integrates cloud-native design principles, such as microservices, container isolation, policy-driven security, and automated delivery pipelines, with migration process guidelines from the cloud migration literature. Target state design, strategy selection, execution, discovery and dependency mapping, and ongoing optimization are the five stages that the framework breaks down migration into [34].

Bondarenko et al. (2023) the potential and suitability of container management systems for developing distributed systems with a microservice architecture, as programmers and architects are increasingly favoring this design these days. The widespread

usage of cloud technologies, which are getting easier to set up and maintain every year, is the cause. To accomplish the study's objective, the authors considered the key components of the microservice architecture and assessed whether a Kubernetes/OpenShift-based container management system could be used to build distributed systems with this design. Additionally, an experiment was carried out to show that one of the most crucial prerequisites for the produced systems of this kind is the efficient administration of the network topology [35].

Zhong et al. (2022) provide an extensive analysis of the current state of container orchestration methods that rely on ML. The present studies are to be categorized according to their shared characteristics using detailed taxonomies. In addition, measurements and goals have guided the development of container orchestration systems that use ML from 2016 to 2021. A comparative study of the examined methodologies is undertaken according to the suggested taxonomies, with a focus on their main properties. Lastly, a number of unresolved research issues and possible future paths are highlighted [23].

Liu et al. (2021) provide a technique for evaluating performance from the standpoint of industrial clients and apply it to cutting-edge edge-cloud computing infrastructures that rely on containerization. They evaluate the impact of message sending interval, payload, network bandwidth, and concurrent devices on overall stack latency as well as the computing power required to complete ML tasks. It suggests that containerization on the edge does not significantly impair communication, processing, or intelligence capabilities, which makes it a viable remedy for the edge-cloud computing paradigm. Nevertheless, the current implementation of the edge-cloud infrastructure still has a significant amount of space for performance enhancement in order to meet the rigorous requirements of time-critical industrial applications [36].

Shih et al. (2021) researched the performance of three environments (virtual machines, Docker containers, and bare-metal) in order to comprehend the differences in the features of each environment. The inquiry of whether container-based virtualization can mitigate the difficulties of traditional virtualization was also covered. Additionally, they developed a framework for managing containers by combining OpenStack and Docker. At last, they used Hadoop deployment to test Docker's ability to speed up and eliminate deployment issues. Virtualization is a key element of next-generation data centers. However, the problem with virtualization technology is that each instance has to run many programs and a client operating system [37].

Chinamanagonda (2020) teams may decrease human error, minimize downtime, and speed up time-to-market by using advanced technologies and approaches like infrastructure as code, AI-driven testing, and automated security scanning. Furthermore, embracing sophisticated automation provides continuous feedback and improvement, enabling developers to adapt swiftly to changes and maintain a competitive advantage. Automation is moving from being a supplemental tool to a crucial component in the development lifecycle, increasing efficiency and innovation, as CI/CD becomes mainstream. This article delves into the revolutionary effects of advanced automation on CI/CD pipelines, emphasizing essential tactics and best practices for harnessing automation to accomplish durable, scalable, and seamless software delivery [38].

TABLE I. OPENSHIFT ARCHITECTURE AND ENTERPRISE CONTAINER PLATFORM MANAGEMENT

Authors	Focus Area	Objectives	Approach	Key Findings	Future Work
Vanama (2024)	OpenShift-based cloud modernization	Design an architecture-led migration framework	Multi-phase cloud-native framework with microservices and automation	Improves scalability, security, and structured migration process	Needs broader validation and quantitative performance evaluation
Bondarenko et al. (2023)	Container orchestration for microservices	Assess Kubernetes/OpenShift in distributed systems	Analytical study with experimental validation	Effective for managing microservices and network topology	Lacks focus on large-scale deployment, security, and performance optimization
Zhong et al. (2022)	ML-based container orchestration	Review existing ML-based container orchestration approaches, classify research, and analyze evolution from 2016–2021	Literature review with taxonomies and comparative analysis	Detailed taxonomies for research classification; evolution trend analysis; comparison of techniques	Highlighted potential future possibilities for ML-based orchestration as well as open research issues.
Liu et al. (2021)	Edge-cloud containerization	Analyze containerized edge-cloud infrastructure performance from the standpoint of an industrial customer.	Benchmarking latency, message intervals, payload, bandwidth, and concurrent device processing	Containerization on edge introduces negligible performance overhead; suitable for edge-cloud paradigm	Optimization of edge-cloud infrastructure for time-critical industrial applications
Shih et al. (2021)	Container vs. traditional virtualization	Understand differences between bare-metal, Docker containers, and VMs; evaluate deployment efficiency	Experimental evaluation and Docker + OpenStack implementation; Hadoop deployment test	Containers reduce deployment complexity and time; address some limitations of traditional virtualization	Further research on container orchestration with complex applications
Chinamanagonda (2020)	CI/CD automation	Examine impact of advanced automation tools in software delivery	Review of AI-driven testing, IaC, automated security scans	Automation reduces human error, downtime; accelerates time-to-market; enables continuous improvement	Integration of advanced automation with container orchestration for enterprise pipelines

VI. CONCLUSION AND FUTURE WORK

The authors have provided an in-depth profile of OpenShift architecture and enterprise container platform management, as it has become an important requirement of cloud-native and hybrid enterprise environments. With built-in security, automated working procedures, and commercial-grade tooling, OpenShift eases the installation, administration, and development of

containerized applications. The paper has studied some of the major architectural elements and deployment patterns and management features like orchestration, monitoring, observability, and application lifecycle automation. These characteristics help organizations to give better scalability, reliability and operational efficiency and have good security and compliance controls. Moreover, the enterprise adoption and use cases discussion showed that OpenShift can provide support to microservices-based architectures, modernization of legacy applications, and data-intensive applications. Altogether, OpenShift offers a powerful and production-ready system that helps to manage the complexity of enterprise container management and meet the needs of the digital transformation that is constantly changing.

The performance benchmarking of OpenShift in multi-cloud environments, its integration with AI-based automation and security analytics, and the comparison with new container platforms can be considered in future research. Also, there is potential research in assessing energy efficiency, sustainability and cost minimization plans in OpenShift large-scale deployments.

REFERENCES

[1] A. Parupalli and H. Kali, "An In-Depth Review of Cost Optimization Tactics in Multi-Cloud Frameworks," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 3, no. 5, pp. 1043–1052, Jun. 2023, doi: 10.48175/IJARSCT-11937Q.

[2] C. Patel, "A Review of Multi-Channel CRM Strategies Using Big Data and Cloud Integration," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 8, no. 1, pp. 577–588, 2022, [Online]. Available: https://ijsrcseit.com/CSEIT25441

[3] S. Kaiser, M. S. Haq, A. S. Tosun, and T. Korkmaz, "Container Technologies for ARM Architecture: A Comprehensive Survey of the State-of-the-Art," *IEEE Access*, vol. 10, pp. 84853–84881, 2022, doi: 10.1109/ACCESS.2022.3197151.

[4] Z. Zhong and R. Buyya, "A Cost-Efficient Container Orchestration Strategy in Kubernetes-Based Cloud Computing Infrastructures with Heterogeneous Resources," *ACM Trans. Internet Technol.*, vol. 20, no. 2, pp. 1–24, May 2020, doi: 10.1145/3378447.

[5] V. Dakić, M. Kovač, and J. Slovinac, "Evolving High-Performance Computing Data Centers with Kubernetes, Performance Analysis, and Dynamic Workload Placement Based on Machine Learning Scheduling," *Electronics*, vol. 13, no. 13, p. 2651, Jul. 2024, doi: 10.3390/electronics13132651.

[6] Q.-M. Nguyen, L.-A. Phan, and T. Kim, "Load-Balancing of Kubernetes-Based Edge Computing Infrastructure Using Resource Adaptive Proxy," *Sensors*, vol. 22, no. 8, p. 2869, Apr. 2022, doi: 10.3390/s22082869.

[7] A. Nair, "Unlocking Performance Optimizing Hybrid Infrastructure with Oracle Enterprise Linux and Red Hat," *Int. J. Sci. Res. Eng. Trends*, vol. 5, no. 4, pp. 1–9, 2019.

[8] D. Rana, "From AIX to Microservices a Strategic Migration Plan for Legacy Enterprise Applications," *Int. J. Sci. Res. Eng. Trends*, vol. 5, no. 3, pp. 1–6, 2019.

[9] S. Pawar and Y. Ghodke, "Macro Software Integration In Operating System For Office Automation," *IJARIIE*, vol. 4, no. 4, 2018, [Online]. Available: https://ijariie.com/AdminUploadPdf/MACRO_SOFTWARE_INTEGRATION_IN_OPERATING_SYSTEM_FOR_OFFICE_AUTOMATION_ijariie8875.pdf?srsltid=AfmBOopnbZaZgZbTAI_Y_yuWPx8lWNo9gkfy95GWJ49PRPKEBKsECGBD

[10] T. Joshi, "Beyond The Server Red Hat ' s Role in Orchestrating Containers on Hybrid Clouds," *Int. J. Sci. Res. Eng. Trends*, vol. 5, no. 3, pp. 1–7, 2019.

[11] D. P. Guda, "AI-Driven Threat Detection in DevSecOps Pipelines for Insurance Applications," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 23s, pp. 3445–3451, 2024, doi: 10.17762/ijisae.v12i23s.7759.

[12] S. Chatterjee, "A Data Governance Framework for Big Data Pipelines: Integrating Privacy, Security, and Quality in Multitenant Cloud Environments," *Tech. Int. J. Eng. Res.*, vol. 10, no. 5, 2023, doi: 10.56975/tijer.v10i5.158181.

[13] A. Rajana, L. V Vigneshwaran, N. Philip, S. Abdulsalam, and A. B. T. K, "A Comprehensive Proliferation on Cloud Computing with Models," *Int. J. Res. Publ. Rev.*, vol. 2, no. 8, pp. 2760–2766, 2021.

[14] Dhruv Patel and Ritesh Tandon, "Cryptographic Trust Models and Zero-Knowledge Proofs for Secure Cloud Access Control and Authentication," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 2, no. 1, pp. 749–758, 2022, doi: 10.48175/ijarsct-7744d.

[15] S.-H. Han, H.-K. Lee, G.-Y. Gim, and S.-J. Kim, "Empirical Study on Anti-Virus Architecture for Container Platforms," *IEEE Access*, vol. 8, pp. 134940–134949, 2020, doi: 10.1109/ACCESS.2020.3005591.

[16] M. Kothapalli, "Cloud Computing Architectures: Comparing Service Models (IaaS, PaaS, SaaS) and Deployment Models (Public, Private, Hybrid, Community)- Uses and Trade-offs," *J. Sci. Eng. Res.*, vol. 5, no. 10, pp. 334–341, 2018, doi: 10.5281/zenodo.12798259.

[17] G. Sarraf and V. Pal, "Privacy-Preserving Data Processing in Cloud : From Homomorphic Encryption to Federated Analytics," *Int. J. Sci. Res. Comput. Sci. Eng. Inf. Technol.*, vol. 8, no. 2, pp. 735–749, 2022.

[18] A. R. Bairi, V. P. Rambabu, and B. Yakkanti, "AI-Enhanced Test Prioritization in Continuous Integration for SaaS Platforms," *Am. J. Auton. Syst. Robot. Eng.*, vol. 2, pp. 110–145, Aug. 2022, [Online]. Available: https://www.ajasre.org/index.php/publication/article/view/29

[19] C. Tayal, "Designing Hybrid ETL Pipelines for Multi-Cloud Integration," *Int. J. Emerg. Technol. Comput. Sci. Inf. Technol.*, vol. 4, no. 4, pp. 129–134, Dec. 2023, [Online]. Available: https://www.ijetcsit.org/index.php/ijetcsit/article/view/468

[20] S. Raveendran, U. B. Yalamanchi, and N. Raveendran, "Method, apparatus, and computer-readable medium for dynamic binding of tasks in a data exchange," US Patent 11,132,221, 2021

[21] A. Palumbo, "Literature Review on Kubernetes & RedHat Openshift Container Platform," vol. 32, no. 3, pp. 167–186, 2021.

[22] S. Achouche, U. Bhaskar, Yalamanchi, and N. Raveendran, "Method, apparatus, and computer-readable medium for performing a data exchange on a data exchange framework," 10387195, 2019

[23] Z. Zhong, M. Xu, M. A. Rodriguez, C. Xu, and R. Buyya, "Machine Learning-based Orchestration of Containers: A Taxonomy and Future Directions," *ACM Comput. Surv.*, vol. 54, no. 10, 2022, doi: 10.1145/3510415.

[24] M. N. Chowdary, S. Bussa, C. K. Chowdary, and M. Gupta, "Automated Pipeline for the Deployment using OpenShift," *Procedia Comput. Sci.*, vol. 215, pp. 220–229, 2022, doi: 10.1016/j.procs.2022.12.025.

[25] N. Nikolakis, R. Senington, K. Sipsas, A. Syberfeldt, and S. Makris, "On a containerized approach for the dynamic planning and control of a cyber - physical production system," *Robot. Comput. Integr. Manuf.*, vol. 64, p. 101919, Aug. 2020, doi: 10.1016/j.rcim.2019.101919.

[26] A. Malviya and R. K. Dwivedi, "A Comparative Analysis of Container Orchestration Tools in Cloud Computing," in *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, IEEE, Mar. 2022, pp. 698–703. doi: 10.23919/INDIACom54597.2022.9763171.

[27] A. Chandrasehar, "ML Powered Container Management Platform: Revolutionizing Digital Transformation through Containers and Observability," *J. Artif. Intell. Cloud Comput.*, pp. 1–3, Mar. 2022, doi: 10.47363/JAICC/2023(1)130.

[28] R. Tandon and D. Patel, "Evolution of Microservices Patterns for Designing Hyper- Scalable Cloud-Native Architectures," *ESP J. Eng. Technol. Adv.*, vol. 1, no. 1, pp. 288–297, 2021, doi: 10.56472/25832646/JETA-V1I1P131.

[29] V. S. Thokala, "Utilizing Docker Containers for Reproducible Builds and Scalable Web Application Deployments," *Int. J. Curr. Eng. Technol.*, vol. 11, no. 6, pp. 661–668, 2021.

[30] F. Tapia, M. Á. Mora, W. Fuertes, J. E. Lascano, and T. Toulkeridis, "A Container Orchestration Development that Optimizes the Etherpad Collaborative Editing Tool through a Novel Management System," *Electronics*, vol. 9, no. 5, p. 828, May 2020, doi: 10.3390/electronics9050828.

[31] V. Thakran, "Environmental Sustainability in Piping Systems : Exploring the Impact of Material Selection and Design Optimisation," *Int. J. Curr. Eng. Technol.*, vol. 11, no. 5, pp. 523–528, 2021.

[32] V. Thakran, "Impact of Advanced Materials in Enhancing the Mechanical Properties of Piping Systems for Stress Analysis," *Int. J. Recent Technol. Sci. Manag.*, vol. 7, no. 4, 2022, doi: 10.10206/IJRTSM.2025259794.

[33] 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, Aug. 2015, pp. 379–386. doi: 10.1109/FiCloud.2015.35.

[34] S. K. R. Vanama, "Architecture Led Cloud Modernization: A Framework for Enterprise Migration from VMware to OpenShift and AWS," *Int. J. Emerg. Res. Eng. Technol.*, vol. 5, no. 1, March, pp. 117–125, 2024, doi: 10.63282/3050-922X.IJERET-V5I1P114.

[35] Bondarenko, A. Zaytsev, and KS, "Using container management systems to build distributed cloud information systems with microservice architecture," *Int. J. Open Inf. Technol.*, vol. 11, no. 8, pp. 17--23, 2023.

[36] Y. Liu, D. Lan, Z. Pang, M. Karlsson, and S. Gong, "Performance Evaluation of Containerization in Edge-Cloud Computing Stacks for Industrial Applications: A Client Perspective," *IEEE Open J. Ind. Electron. Soc.*, vol. 2, pp. 153–168, 2021, doi: 10.1109/OJIES.2021.3055901.

[37] W.-C. Shih, C.-T. Yang, R. Ranjan, and C.-I. Chiang, "Implementation and evaluation of a container management platform on Docker: Hadoop deployment as an example," *Cluster Comput.*, vol. 24, no. 4, pp. 3421–3430, Dec. 2021, doi: 10.1007/s10586-021-03337-w.

[38] S. Chinamanagonda, "Enhancing CI/CD Pipelines with Advanced Automation-Continuous integration and delivery becoming mainstream," *J. Innov. Technol.*, vol. 5, no. 4, pp. 1–15, 2020.