AMERICAN ACADEMIC PUBLISHER

INTERNATIONAL JOURNAL OF DATA SCIENCE AND MACHINE LEARNING (ISSN: 2692-5141)

Volume 05, Issue 02, 2025, pages 556-563
Published Date: - 31-12-2025

A Migration Framework for Transforming OpenStack Cloud Infrastructure to OpenShift-Based Enterprise Cloud Platforms

Jeyakumar Ramachandran
Independent Researcher, USA

ABSTRACT

Enterprise organizations are increasingly modernizing OpenStack-based private cloud environments to adopt Kubernetes and OpenShift platforms for cloud-native application delivery. However, migrating workloads from OpenStack to OpenShift introduces challenges related to application dependencies, networking, storage, security, workload compatibility, operational processes, and infrastructure governance. This paper proposes a structured migration framework for transforming OpenStack cloud infrastructure into an OpenShift-based enterprise cloud platform. The framework covers workload discovery and classification, dependency assessment, application containerization, storage and networking migration, identity and security integration, CI/CD transformation, operational monitoring, and workload validation. The proposed approach uses phased migration strategies to reduce service disruption and operational risk while enabling organizations to gradually transition from traditional infrastructure-oriented cloud operations to Kubernetes-based platform engineering. The framework also addresses coexistence between OpenStack and OpenShift during the migration period and provides criteria for workload prioritization, migration readiness, validation, rollback, and post-migration optimization. The proposed methodology provides enterprises with a practical approach for modernizing OpenStack environments while improving application portability, deployment agility, scalability, and operational consistency.

KEYWORDS

OpenStack, OpenShift, Kubernetes, Cloud Modernization, Cloud Migration, Enterprise Cloud, Containerization, Platform Engineering, Infrastructure Automation, Workload Migration, Hybrid Cloud, DevOps

1. Introduction

Enterprise cloud infrastructures have evolved from conventional virtual-machine-oriented environments toward platforms capable of supporting containers, microservices, automated deployment, elastic scaling, and policy-driven operations. OpenStack has played an important role in establishing programmable infrastructure environments in which computer, networking, and storage resources can be managed through software-defined mechanisms. OpenShift, by contrast, builds its operational model around Kubernetes and containerized workloads, emphasizing application-centric deployment, orchestration, automation, and platform-level governance. Consequently, organizations seeking to transform an OpenStack infrastructure into an OpenShift-based enterprise platform face a transition between two related but substantially different operational paradigms.

The fundamental problem is not merely how to install OpenShift on infrastructure previously managed through OpenStack. The more difficult question is how existing workloads, network policies, resource allocations, automation processes, security controls, and operational responsibilities can be transformed without

compromising availability, performance, or governance. Research on Kubernetes demonstrates that networking policies can have measurable performance and security implications, indicating that network migration must be treated as an architectural concern rather than as a configuration task (Budigiri et al., 2021). Similarly, research into Kubernetes orchestration demonstrates that heterogeneous resources create important scheduling and cost-efficiency considerations (Zhong & Buyya, 2020).

The relevance of this transformation is strengthened by the increasing dependence of enterprise applications on distributed services. Service-mesh research illustrates the need to manage performance objectives across multiple services rather than treating application components independently (Samani & Stadler, 2022). Autoscaling research further demonstrates that responsive resource adaptation is an important component of containerized application management (Huo et al., 2022; Xie et al., 2023).

This study therefore proposes a migration framework that treats OpenStack-to-OpenShift transformation as a multi-dimensional modernization process. The objectives are to identify the major migration dimensions, establish a phased transformation methodology, analyze the technical dependencies among infrastructure, networking, security, automation, and workloads, and identify the expected outcomes and limitations of the proposed approach. The scope is conceptual and analytical rather than empirical; the framework is derived exclusively from the supplied literature.

2. Literature Review

The literature establishes several theoretical foundations relevant to OpenStack-to-OpenShift transformation. First, cloud infrastructure transformation must account for heterogeneous resource environments. Zhong and Buyya (2020) emphasize cost-efficient container orchestration in Kubernetes-based infrastructures containing heterogeneous resources. Their work is particularly relevant because migration from OpenStack may expose differences in CPU, memory, storage, and network capabilities across infrastructure pools. A migration strategy that ignores these differences could produce technically functional but operationally inefficient deployments.

Second, networking represents a critical transformation layer. Budigiri et al. (2021) examine Kubernetes network policies from both performance and security perspectives, demonstrating that network-policy implementation is not isolated from system behavior. Minna, Maffeis, and Polverini (2021) similarly highlight security implications associated with Kubernetes networking. These studies imply that migration requires explicit translation of existing OpenStack network segmentation, security groups, routing assumptions, and access policies into the networking and policy model of the target platform.

Third, infrastructure automation constitutes an essential foundation. Staron, Bosch, and Runeson (2022) examine Infrastructure as Code and its relationship to contemporary research and industrial practice. For migration, this perspective suggests that manually reconstructing the target environment introduces unnecessary configuration inconsistency. Infrastructure and platform definitions should instead be represented as reproducible artifacts wherever possible.

Fourth, application-level performance must be considered alongside infrastructure transformation. Samani and Stadler (2022) demonstrate the relevance of dynamically meeting performance objectives across multiple services in a service-mesh environment. Huo, Li, and Chen (2022) investigate fast-response horizontal pod autoscaling, while Xie et al. (2023) propose bottleneck-aware autoscaling for microservice-based applications. Together, these studies indicate that migration cannot be considered successful merely because applications start successfully on the target platform. Resource responsiveness, service dependencies, bottlenecks, and scaling behavior must also be validated.

The distributed nature of enterprise migration introduces an organizational dimension. Prikladnicki, Audy, and Evaristo (2010) identify challenges associated with distributed software development project management, which are relevant to migration programs involving infrastructure teams, application teams, security groups, and operations personnel. Calefato et al. (2021) similarly identify challenges in global software engineering, supporting the proposition that migration governance must address coordination and communication in addition to technical implementation.

Miorandi et al. (2012) provide a broader perspective on Internet of Things architectures and distributed systems, reinforcing the importance of scalable and interconnected computing environments. Gunda et al. (2023) contribute an architecture-centered governance perspective through their discussion of decision intelligence and agile software lifecycle governance.

Despite these contributions, the supplied literature does not provide a unified framework specifically addressing transformation from OpenStack infrastructure to an OpenShift-based enterprise platform. Existing studies concentrate on individual dimensions such as Kubernetes networking, orchestration efficiency, autoscaling, infrastructure automation, distributed software development, or governance. The research gap therefore concerns integration: a migration framework must connect infrastructure assessment, architectural transformation, networking, security, automation, workload migration, performance validation, and organizational governance into a single lifecycle.

3. Methodology

3.1 Research Design

This study adopts a conceptual framework-development methodology based exclusively on the supplied references. The methodology synthesizes recurring technical and organizational dimensions across the literature and translates them into a structured migration lifecycle. The resulting framework is not presented as an experimentally validated migration procedure; rather, it functions as a research-driven architectural model for planning and controlling OpenStack-to-OpenShift transformation.

The framework consists of six sequential but partially iterative phases: **(1) discovery and assessment, (2) target architecture design, (3) workload and resource mapping, (4) infrastructure and platform transformation, (5) validation and optimization, and (6) operational governance and continuous improvement**.

3.2 Phase I: Discovery and Assessment

The first phase establishes a baseline of the existing OpenStack environment. Assessment should identify compute nodes, memory availability, storage classes, virtual networks, security groups, workload dependencies, availability requirements, deployment mechanisms, monitoring practices, and automation maturity.

Workloads should then be classified according to their architectural characteristics. Stateless services generally represent lower migration complexity because they can be redeployed with comparatively limited state-management concerns. Stateful workloads require additional analysis of storage, persistence, backup, recovery, and data-consistency requirements.

Resource heterogeneity must also be documented. Zhong and Buyya (2020) demonstrate why heterogeneous infrastructure is important for container orchestration efficiency. Therefore, migration assessment should not treat all OpenStack resources as interchangeable. Instead, resource pools should be characterized according to

computational capacity, storage performance, network characteristics, and workload suitability.

A practical enterprise example would involve an OpenStack environment containing web applications, databases, message-processing services, and internal APIs. The assessment could identify the web and API tiers as initial candidates for containerization while treating databases and highly stateful services as later migration candidates.

3.3 Phase II: Target Architecture Design

The second phase defines the OpenShift target architecture. The architecture should establish the relationship between underlying infrastructure and the Kubernetes/OpenShift control plane. The objective is not to reproduce the OpenStack environment exactly but to preserve business capabilities while adopting a more application-centric operating model.

The target architecture should define cluster topology, compute allocation, networking, storage integration, identity management, security policies, container registries, deployment mechanisms, monitoring, and disaster-recovery requirements.

The architectural design should explicitly account for the security and performance implications of Kubernetes networking. Budigiri et al. (2021) demonstrate that network policies can influence both security and performance, while Minna et al. (2021) emphasize security considerations in Kubernetes networking. Therefore, network migration should include policy translation and validation rather than simple connectivity testing.

3.4 Phase III: Workload and Resource Mapping

Workload mapping establishes how existing OpenStack resources and applications correspond to target OpenShift constructs. Virtual-machine-based applications must be evaluated for container suitability. Where direct containerization is impractical, transitional deployment strategies may be required.

Resource mapping should examine CPU and memory requests, resource limits, storage requirements, network dependencies, service discovery, ingress requirements, and scaling characteristics. This is particularly significant because inefficient resource placement can undermine the expected benefits of container orchestration. Zhong and Buyya (2020) provide a theoretical basis for considering resource heterogeneity when designing orchestration strategies.

Microservice applications should additionally be evaluated for inter-service dependencies. Samani and Stadler (2022) indicate the importance of dynamically managing performance objectives across multiple services. Therefore, migration planning should identify critical service paths rather than evaluating every application component independently.

3.5 Phase IV: Infrastructure and Platform Transformation

The fourth phase implements the target environment. Infrastructure provisioning should be automated wherever practical. Infrastructure as Code provides a mechanism for representing infrastructure configuration as reproducible and reviewable artifacts (Staron et al., 2022).

The transformation process should proceed incrementally. A representative sequence could begin with development workloads, proceed to non-critical enterprise applications, and finally address highly critical production systems. Such sequencing reduces blast radius and creates opportunities to validate architectural assumptions.

Network policies should be implemented according to the target security model. Existing access relationships should be translated into explicit application-to-application communication requirements. Security validation should verify both permitted and denied traffic patterns.

Application deployment should subsequently be transformed into container-native processes. Continuous deployment mechanisms, configuration management, health checks, resource policies, and scaling rules should be incorporated into the target platform.

3.6 Phase V: Validation and Optimization

Validation should occur at infrastructure, platform, application, security, and operational levels. Infrastructure validation should verify resource availability and cluster stability. Platform validation should examine scheduling, networking, storage, service discovery, and application lifecycle management.

Application validation should include functional correctness, latency, throughput, failure recovery, and scaling behavior. Autoscaling is particularly important for workloads with variable demand. Huo et al. (2022) demonstrate the importance of responsive horizontal pod autoscaling, while Xie et al. (2023) emphasize bottleneck awareness in microservice autoscaling.

Security validation should include policy enforcement and isolation testing. Performance validation should compare representative workloads against established baseline measurements from the OpenStack environment. The migration should be considered successful only when functional requirements and relevant non-functional requirements are maintained or improved.

3.7 Phase VI: Governance and Continuous Improvement

The final phase converts migration from a one-time technical project into an operational capability. Governance should establish responsibility for platform architecture, security policy, deployment standards, resource management, and application lifecycle processes.

Gunda et al. (2023) support the relevance of architecture-centered governance and decision intelligence in managing software lifecycle activities. Similarly, the distributed coordination challenges identified by Prikladnicki et al. (2010) and Calefato et al. (2021) suggest that technical transformation requires clear communication structures and cross-functional accountability.

Continuous improvement should use operational measurements to refine resource allocation, scaling policies, security rules, and deployment processes. The migration framework therefore concludes not with platform activation but with establishment of a measurable operating model.

4. Results

The analytical synthesis produces five principal findings concerning OpenStack-to-OpenShift transformation.

First, migration should be understood as **architectural transformation rather than infrastructure replacement**. OpenStack and OpenShift operate at different abstraction levels and emphasize different operational models. Consequently, direct one-to-one mapping of virtual-machine infrastructure into containers is insufficient. The migration must identify application requirements and reconstruct appropriate platform services around them.

Second, **resource heterogeneity is a central migration concern**. The effectiveness of container orchestration

depends on matching workload requirements with available resources. Zhong and Buyya (2020) demonstrate the significance of heterogeneous resources for efficient Kubernetes orchestration. This finding implies that migration planning should establish resource profiles before workload placement and should avoid assuming uniform infrastructure capacity.

Third, **networking and security must be migrated together**. The literature indicates that Kubernetes network policies affect both security and performance (Budigiri et al., 2021), while Kubernetes networking introduces security implications that must be explicitly considered (Minna et al., 2021). Accordingly, successful transformation requires policy translation, communication testing, segmentation validation, and performance assessment.

Fourth, **automation represents a foundational migration capability rather than an optional optimization**. Infrastructure as Code provides reproducibility and consistency (Staron et al., 2022), reducing configuration drift during repeated environment construction. Automation is therefore especially valuable when migration requires multiple development, testing, staging, and production environments.

Fifth, **application behavior after migration is as important as infrastructure correctness**. Autoscaling and service-level performance research indicates that containerized applications require dynamic resource management (Huo et al., 2022; Xie et al., 2023). A workload that operates correctly but experiences unstable scaling or unacceptable latency cannot be considered successfully migrated.

The framework consequently indicates that migration success should be measured through a combination of functional continuity, resource efficiency, security-policy correctness, performance stability, deployment reproducibility, and operational governance. These dimensions reinforce the argument that heterogeneous-resource-aware orchestration is fundamental to efficient cloud transformation (Zhong & Buyya, 2020).

5. Discussion

The proposed framework contributes to the literature by integrating technical concerns that are frequently examined separately. Existing research provides valuable evidence regarding Kubernetes orchestration, networking, autoscaling, Infrastructure as Code, service performance, and distributed project management, but these concerns become substantially interdependent during an enterprise platform transformation.

The first major implication is architectural. An organization should resist treating OpenShift as merely a new deployment destination. The target platform changes the relationship between infrastructure teams and application teams by increasing the importance of declarative configuration, container lifecycle management, resource policies, service discovery, and automated deployment. This creates opportunities for operational standardization but also introduces new governance requirements.

The second implication concerns performance. Resource-efficient orchestration is particularly important where OpenStack environments contain heterogeneous hardware. Zhong and Buyya (2020) provide a relevant foundation for understanding this issue. A migration that simply redistributes workloads without considering resource characteristics may produce underutilization in some pools and contention in others. Consequently, resource requests, limits, placement policies, and autoscaling rules should be treated as interconnected mechanisms.

The third implication concerns security. Network-policy transformation creates a potential contradiction between stronger isolation and increased policy complexity. Budigiri et al. (2021) demonstrate the performance dimension of network policies, while Minna et al. (2021) demonstrate their security relevance. Organizations therefore need a policy model that establishes security boundaries without creating unnecessary communication overhead or

operational complexity.

The fourth implication concerns automation and organizational capability. Infrastructure as Code can improve consistency, but automation alone does not eliminate migration risk. Teams must understand the new platform abstractions and establish standardized operational responsibilities. The coordination issues described by Prikladnicki et al. (2010) and Calefato et al. (2021) indicate that distributed transformation projects can encounter organizational bottlenecks even when the underlying technology is appropriate.

The framework also has limitations. Because it is derived exclusively from the supplied literature, it does not provide empirical measurements from a real OpenStack-to-OpenShift migration. The framework therefore cannot establish universal performance improvements or guarantee specific migration timelines. In addition, workload characteristics vary considerably across enterprises, meaning that migration sequencing and validation criteria must be adapted to organizational context.

Another limitation is that the framework assumes sufficient organizational capacity to implement automation, security governance, and continuous validation. Smaller organizations may face resource constraints that make simultaneous transformation across infrastructure, applications, networking, and governance impractical. A phased adoption model is therefore preferable.

Nevertheless, the integrated framework offers a useful theoretical position: enterprise cloud transformation should be governed as a lifecycle in which infrastructure modernization, application modernization, resource orchestration, networking, security, automation, and organizational coordination are mutually dependent. Service-level management and autoscaling research further suggests that the target environment must be evaluated dynamically rather than only during initial migration (Samani & Stadler, 2022; Huo et al., 2022).

6. Conclusion

This paper proposed a structured framework for transforming OpenStack cloud infrastructure into an OpenShift-based enterprise cloud platform. The analysis established that successful transformation requires substantially more than technical platform installation. It requires systematic discovery of existing infrastructure, target architecture design, workload classification, resource mapping, networking and security transformation, infrastructure automation, application migration, performance validation, and continuous governance.

The principal contribution of the framework is its integrated perspective. Heterogeneous-resource management, emphasized by Zhong and Buyya (2020), is connected with Kubernetes networking, security, Infrastructure as Code, autoscaling, service performance, and organizational governance. This integration provides a stronger conceptual foundation for enterprise migration because migration risks frequently arise at the boundaries between these domains rather than within one domain alone.

For practical implementation, organizations should prioritize workload assessment and dependency mapping before migration, establish automated infrastructure provisioning, translate network and security policies explicitly, adopt incremental workload migration, and validate both functional and non-functional requirements. Highly stateful or business-critical workloads should receive additional validation before production transition.

Future research should empirically evaluate the proposed framework through controlled enterprise migration studies. Such research could measure migration duration, resource utilization, performance variation, policy overhead, operational effort, and failure rates across different workload categories. Comparative studies could also examine alternative migration strategies and determine how workload characteristics influence the optimal

migration sequence. Ultimately, OpenStack-to-OpenShift transformation should be approached as a controlled architectural evolution in which technological modernization and operational governance develop together.

References

1. Budigiri, G., Baumann, C., Mühlberg, J. T., Truyen, E., & Joosen, W. (2021). Network policies in Kubernetes: Performance evaluation and security analysis. In 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G) (pp. 511–516). IEEE. https://doi.org/10.1109/EuCNC/6GSummit51104.2021.9482526

2. F. Calefato, A. Dubey, C. Ebert and P. Tell, "Global Software Engineering: Challenges and solutions," Journal of Systems and Software, 2021, 10.1016/j.jss.2020.110887.

3. Gunda SK, Yettapu SDR, Bodakunti S, Bikki SB. Decision Intelligence Methodology for AI-Driven Agile Software Lifecycle Governance and Architecture-Centered Project Management, 2023 Mar. 30;4(1):102-8. https://doi.org/10.63282/3050-9262.IJAIDSML-V4I1P112

4. W. Huo, X. Li and Y. Chen, "A Method for Horizontal Pod Autoscaling with Fast Response," 2022, 10.1109/AIIPCC57291.2022.00051.

5. Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges. Ad Hoc Networks, 10(7), 1497–1516. https://doi.org/10.1016/j.adhoc.2012.02.016

6. F. Minna, S. Maffeis and M. Polverini, "Understanding the Security Implications of Kubernetes Networking," 2021, 10.1109/EuroSP51992.2021.00029.

7. R. Prikladnicki, J. L. N. Audy and R. Evaristo, "Challenges and Solutions in Distributed Software Development Project Management," 2010, 10.1109/ICGSE.2010.18.

8. Samani, F. S., & Stadler, R. (2022). Dynamically meeting performance objectives for multiple services on a service mesh. In Proceedingsof the 18th International Conference on Network and Service Management (CNSM) (pp. 219–225). IEEE. https://doi.org/10.1109/CNSM54069.2022.9933116

9. C. Staron, J. Bosch and P. Runeson, "Infrastructure as Code: Current Research and Industrial Practice," IEEE Software, 2022, 10.1109/MS.2022.3212035.

10. Xie, S., Wang, J., Li, B., Zhang, Z., Li, D., & Huo, P. C. K. (2023). PBScaler: A bottleneck-aware autoscaling framework for microservice-based applications. arXiv. https://doi.org/10.48550/arXiv.2303.14620

11. Z. Zhong and R. Buyya, "A Cost-Efficient Container Orchestration Strategy in Kubernetes Based Cloud Computing Infrastructures with Heterogeneous Resources," ACM Transactions on Internet Technology, 2020, 10.1145/3378447