International Journal of Emerging Research in Engineering and Technology
Pearl Blue Research Group| Volume 5, Issue 1, 117-125, 2024
ISSN: 3050-922X | https://doi.org/10.63282/3050-922X.IJERET-V5I1P114

Original Article

Architecture Led Cloud Modernization: A Framework for Enterprise Migration from VMware to OpenShift and AWS

Siva Kantha Rao Vanama
Cloud Solution Architect, Mphasis Corporation Tampa, Florida, USA.

Abstract *- Enterprises are under sustained pressure to deliver software faster while improving reliability security and cost efficiency. Virtual machine centric platforms such as VMware vSphere remain highly capable for many workloads yet they often reinforce operating models built around static capacity manual change control and infrastructure coupling that slows product delivery. Container platforms and managed cloud services address these constraints by shifting elasticity resiliency and automation into the platform layer. This paper proposes an Architecture Led Cloud Modernization Framework (ALCMF) for migrating complex enterprise workloads from VMware based on premises environments to Red Hat OpenShift running on Amazon Web Services using Red Hat OpenShift Service on AWS (ROSA). Unlike tactical rehosting approaches, the framework starts with architecture and quality attributes and then selects a modernization strategy per workload. It combines migration process guidance from cloud migration literature with cloud native design principles including microservices container isolation policy driven security and automated delivery pipelines. The framework decomposes migration into five phases: discovery and dependency mapping target state design strategy selection execution and continuous optimization. For each phase the paper details practical technical decisions in networking storage identity governance observability and security controls. We also provide reference architecture for ROSA on AWS that supports multi–Availability Zone resiliency least privilege access network policy enforcement service mesh-based traffic security and measurable operational outcomes. A case study style evaluation illustrates how architecture led decisions reduce downtime, improve deployment frequency and enable systematic cost management through autoscaling and rightsizing. The result is a repeatable modernization approach that reduces migration risk while enabling a durable shift toward platform engineering and product-oriented delivery.*

Keywords *- Cloud Modernization, VMware Migration, Red Hat OpenShift, AWS Enterprise Migration, Kubernetes, ROSA, Microservices Architecture, Application Modernization.*

1. Introduction

Cloud adoption has shifted from experimentation to core enterprise strategy. The shift is not only about where workloads run but also how software is built released operated and secured. Traditional virtualization platforms provide strong isolation and operational tooling, but they encourage patterns such as long-lived virtual machines static resource sizing and centrally managed change queues. These patterns can be effective for predictable monolithic systems, yet they become limiting when organizations adopt continuous delivery and frequent product change. In parallel container orchestration platforms have matured and made it practical to run distributed systems with standardized deployment primitives and automated recovery behaviors. Kubernetes based orchestration has been widely studied as a foundation for scheduling scaling and lifecycle automation in modern platforms [1].

However, migration from VMware to containers and public cloud is rarely straightforward. Most enterprise portfolios include heterogeneous systems with varied compliance constraints, latency sensitivity data gravity concerns licensing limitations and operational ownership models. Industry experience shows that purely tactical migration can reproduce legacy inefficiencies in a new hosting environment. Cloud migration research similarly emphasizes that successful migration depends on systematic process models assessment and governance rather than only tool selection [2] [3].

This paper presents an Architecture Led Cloud Modernization Framework for migrating from VMware to OpenShift on AWS. The central premise is that architecture decisions should precede movement decisions. Each workload should be evaluated for business criticality, technical fitness operational risk and quality attributes such as availability, security, performance and cost. The framework then guides teams to choose an appropriate modernization path such as rehost replatform refactor repurchase retire or retain. The framework is designed for ROSA because it combines managed OpenShift operations with AWS primitives for networking identity and storage which helps enterprises reduce undifferentiated operational work while retaining platform control boundaries appropriate for regulated environments.

The contributions of this paper are fourfold. First it synthesizes migration process research into a concrete enterprise-oriented modernization playbook. Second it provides a target architecture blueprint for ROSA that maps VMware constructs to AWS and OpenShift primitives. Third, it embeds security and governance patterns such as least privilege access and policy

enforcement into each phase to reduce late-stage rework. Fourth, it offers evaluation metrics and a case study style assessment approach that allows organizations to measure outcomes beyond the initial cutover.

2. Background and Related Work

2.1. Constraints of VM centric enterprise estates

VMware vSphere environments are typically optimized around host and cluster utilization lifecycle patching and centralized operational controls. They often rely on capacity forecasts and reserved headroom for peak loads which can lead to chronic overprovisioning. Resource management at the hypervisor level has been studied for decades including memory reclamation and policy mechanisms that improve consolidation efficiency [4]. Those mechanisms help but they do not eliminate the organizational tendency to size for worst case and keep systems running permanently.

Operationally many enterprises treat virtual machines as durable assets. Configuration drift accumulates over time and patch cycles compete with delivery cycles. Recovery often depends on runbooks and ticket driven intervention. This slows incident remediation and makes change risk perception higher which can further reduce delivery speed.

2.2. Container based virtualization and platform scheduling

Containers improve packaging consistency and reduce overhead compared to many virtual machine footprints. Empirical studies have evaluated container-based virtualization performance in high performance environments and show that containers can provide efficient isolation with favorable overhead characteristics for many workloads [5]. Additional comparative work evaluating containers versus virtual machines demonstrates that container-based infrastructure can achieve comparable performance with lower overhead in several scenarios [6]. These characteristics make containers attractive for cost efficiency and elastic scaling, but they also introduce new risks such as multi-tenant kernel sharing and configuration missteps.

At the platform layer Kubernetes provides declarative state reconciliation and supports automated scheduling self-healing and horizontal scaling behaviors. Research on Kubernetes orchestration highlights its role in standardizing deployment and enabling dynamic scheduling in heterogeneous infrastructure [1]. In enterprise settings OpenShift extends this foundation with integrated build delivery security and policy capabilities that reduce the integration burden for internal platform teams.

2.3. Microservices and distributed system complexity

Modernization often aims to decompose monoliths into smaller services to enable independent deployment and scaling. Microservices can improve organizational alignment and delivery velocity, but they also increase the complexity of inter service communication, traffic management and observability. Studies on microservice architectures discuss the reliability and security implications of distributed service interactions and the resulting need for standardized governance mechanisms [7]. In practice an architecture led approach treats microservices as an option rather than a requirement and applies them where domain boundaries latency budgets and team ownership support the shift.

2.4. Cloud migration process models and taxonomy

Cloud migration literature emphasizes that migration is a multi-activity process spanning assessment planning, design, execution validation and optimization. Surveys provide evaluation frameworks and taxonomies that classify migration approaches and highlight recurring challenges such as dependency discovery data movement and organizational readiness [2] [3]. Additional taxonomy work classifies migration research and identifies gaps that are often operational or governance related rather than purely technical [8]. These findings support the need for a structured framework that integrates architecture quality attributes with process discipline.

2.5. Security governance and least privilege in public cloud

Security in cloud native environments is not only perimeter defense. It is identity centric policy driven and continuously validated. Research surveys cloud security issues and mitigation approaches including identity risks misconfiguration and multi tenancy threats [9] [10]. Least privilege access remains a critical control, yet it is difficult to maintain at scale due to evolving workloads and permission sprawl. Research on least privilege calls in AWS analyzes policy requirements and highlights the complexity of achieving minimal permissions for real systems [11]. These findings motivate integrating IAM and RBAC mapping early in migration design rather than treating it as a post cutover hardening activity.

Zero trust approaches also influence enterprise modernization. Work on zero trust architectures in next generation networks emphasizes continuous verification fine grained access and segmentation principles that align with Kubernetes network policy models and service mesh encryption [12] [13].

3. Problem Statement and Design Goals

3.1. Problem statement

Enterprises migrating from VMware to ROSA on AWS face a combined technical and organizational challenge: they must move workloads while changing the operating model. If they migrate without architecture redesign, they risk recreating static

capacity patterns operational silos and security gaps in the cloud. If they attempt full refactoring for the entire portfolio, they risk cost overruns timeline failures and business disruption. The challenge is to create a repeatable framework that selects the right modernization action per workload and provides a target architecture that is secure resilient observable and cost controllable.

3.2. Design goals

The proposed framework is guided by five goals.

- Portfolio aware decision-making modernization strategy should be selected per workload based on architecture fit and business value.
- Measurable outcomes: each phase should produce artifacts and metrics that allow progress tracking and value verification.
- Security by design: identity segmentation and policy enforcement should be designed early and validated continuously.
- Resiliency by default: target architecture should support failure isolation multi–Availability Zone patterns and automated recovery.
- Operational sustainability: migration should leave behind a platform and operating model that teams can run with realistic skills and staffing.

4. The Architecture Led Cloud Modernization Framework

The Architecture Led Cloud Modernization Framework consists of five integrated phases. Each phase produces specific deliverables and establishes decision gates that reduce downstream rework.

4.1. Phase 1: Holistic discovery and inventory mapping

Discovery is not a simple virtual machine list. It is a system understanding exercise that combines technical topology with business context. This phase includes four workstreams.

- Application and service catalog Teams build an application catalog that includes owners criticality service level objectives compliance tier and release cadence. This catalog becomes the anchor for later strategy selection.
- Dependency mapping Enterprise estates contain hidden dependencies such as shared file shares legacy DNS assumptions embedded IP allowlists and non-obvious batch schedules. Dependency discovery should combine static analysis traffic observation and interview-based validation. Migration process research identifies dependency discovery as a recurring source of risk because missing dependencies create cutover failures and extended stabilization windows [2] [3].
- Resource utilization and performance baselining Workloads often run over provisioned on premises. Accurate baselines require CPU memory storage IOPS network throughput latency and peak variability. This baseline supports rightsizing and informs whether pods can scale safely. Studies comparing virtualization methods highlight that performance characteristics differ across container and VM based deployments which makes baselining critical when moving latency sensitive systems [6].
- Data classification and residency constraints Data classification drives region selection encryption requirements and audit controls. The output is a placement policy that determines which systems can move immediately and which require transitional hybrid patterns.

Deliverables from Phase 1 include application catalog dependency diagrams, a baseline metrics report and a risk register.

4.2. Phase 2: Target state architecture design

Target architecture design translates quality attributes into concrete platform decisions. This phase produces a reference architecture tailored to the enterprise rather than a generic cloud diagram.

- Network and connectivity design A ROSA based design typically uses a multi-tier VPC layout with segmentation between ingress egress application nodes and data services. The design must include routing strategies for hybrid connectivity during transition and DNS strategies for service discovery. Latency variability and availability characteristics in public cloud networks have been empirically studied and demonstrate that tenants should design for variation rather than assuming uniform performance [14]. This motivates multi–Availability Zone architecture and careful placement of latency sensitive components.
- Cluster topology and isolation boundaries Enterprises must decide between one large cluster and multiple smaller clusters. Multiple clusters reduce blast radius and align with compliance boundaries, but they increase management overhead and require multi cluster governance. The decision should be driven by threat model operational ownership and scaling boundaries.
- Identity integration and access model Target state design must map VMware operational roles to AWS IAM and OpenShift RBAC. Research on least privilege in AWS highlights that permissions must reflect observed API usage

and evolve as systems change [11]. In practice this means designing IAM roles per workload class and integrating them with Kubernetes service accounts and OpenShift RBAC for platform operations.

- Storage and data services mapping VMware often uses large shared datastores and LUN based patterns. In OpenShift storage is typically requested via Persistent Volume Claims and backed by cloud block or file systems. Migration design must select storage classes performance tiers backup patterns and encryption options.
- Observability and audit design Containers increase the need for distributed tracing structured logging and standardized metrics. Monitoring approaches for Kubernetes using Prometheus and Grafana have been explored for cloud native deployments and highlight the importance of consistent metric collection and alerting [15]. Target design should include log retention trace correlation and audit event pipelines.

Deliverables from Phase 2 include reference architecture diagrams, security model documentation, a cluster topology decision record and an observability design.

4.3. Phase 3: Strategy selection using an expanded 6 Rs

Strategy selection determines how each workload will move. A portfolio rarely benefits from applying one method across all systems. The framework uses six categories.

- Rehost: The workload moves largely unchanged into cloud based virtual machines. This is appropriate for systems that cannot be containerized due to vendor constraints or tightly coupled OS requirements. Rehost can provide quick data center exit value but it does not deliver the full cloud native benefit.
- Replatform: The workload is containerized with minimal code changes. Common examples include packaging an application server into a container image externalizing configuration and shifting state to managed data services.
- Refactor: The workload is redesigned into smaller services or adopts managed platform services to improve scalability and release independence. Microservices research emphasizes both the benefits and operational challenges of this approach which is why the decision must be architecture driven rather than trend driven [7].
- Repurchase: A managed SaaS alternative replaces the system. This can simplify operations and reduce technical debt, but it introduces vendor dependency and integration change.
- Retire: The system is decommissioned. Portfolio discovery often finds unused applications that still consume capacity and increase security exposure.
- Retain: The system stays on premises temporarily due to compliance latency or contractual constraints. Retain decisions require a hybrid integration plan.

Selection criteria include business value modernization effort change risk data complexity and security posture. The output is a modernization backlog prioritized by value and feasibility.

4.4. Phase 4: Migration execution and data synchronization

Execution is implemented as a factory model with repeatable pipelines not as one off projects. This phase includes building standardization of data movement and validation.

- Container build and image governance Container images must be reproducible versioned and scanned. Container security surveys show that vulnerabilities and misconfigurations remain common and advocate for defense in depth controls across build deploy and runtime [16] [17]. The framework therefore requires image signing vulnerability scanning and base image standardization before promoting to production.
- CI CD integration and delivery automation Modern delivery requires pipeline automation across build test security checks and deployment promotion. Research on continuous delivery practices highlights how automation reduces lead time and change failure rates when paired with disciplined validation gates [18]. In OpenShift pipelines such as Tekton can implement this pattern while integrating policy checks.
- Data migration and cutover planning Stateful workloads require careful planning for replication latency and rollback. Cloud migration process research identifies data movement and synchronization as a top challenge because data gravity increases cutover risk [2] [3]. The framework recommends staged replication with controlled cutovers and clear decision points for rollback.
- Incremental rollout and traffic control For services with customer traffic the framework recommends incremental rollout using blue green or canary techniques. Service mesh approaches provide traffic shaping mutual TLS and policy enforcement which can reduce release risk by controlling the blast radius of new versions. Frameworks for managing services on service mesh platforms emphasize end to end objectives and highlight the role of standardized control planes in complex environments [19].
- Verification and operational readiness Each migrated workload must pass functional tests performance regression tests security verification and operational readiness checks. Operational readiness includes alert coverage, runbooks on call ownership and capacity rules.

Deliverables from Phase 4 include container repositories pipeline definitions migration runbooks validation reports and cutover records.

4.5. Phase 5: Continuous optimization and modernization closure

Modernization does not end at cutover. Post migration optimization ensures the enterprise captures financial and operational benefits.

- Autoscaling and rightsizing Kubernetes autoscaling research proposes methods to improve responsiveness and stability for horizontal scaling under variable load [20]. Additional work studies autoscaling properties and suggests that careful tuning is required to avoid oscillation and performance degradation [21]. The framework therefore mandates load testing and autoscaler policy reviews before enabling aggressive scaling in production.
- Cost and performance optimization Container orchestration cost optimization strategies demonstrate that scheduling decisions and heterogeneous resources influence overall cost and utilization [22]. The framework applies these insights by using node pool segmentation instance selection policies and workload placement rules to align cost with service criticality.
- Security posture management Post migration security includes continuous configuration validation policy drift detection and least privilege refinement. Work on least privilege in AWS suggests that observed behavior should inform policy tightening which aligns with continuous audit and telemetry driven access reviews [11].
- Architecture debt retirement After stabilization teams should complete remaining refactors that were deferred for speed such as decomposing shared databases introducing async messaging or removing legacy runtime dependencies.

Deliverables from Phase 5 include cost optimization reports autoscaling configurations updated threat models and a closure assessment against original business goals.

5. Technical Reference Architecture for Rosa On Aws

5.1. Mapping VMware constructs to AWS and OpenShift

A practical migration requires a clear mapping between source and target constructs.

Table 1: Capability Comparison: Vmware Vs. ROSA On AWS”

Capability	VMware source pattern	ROSA on AWS target pattern
Computer isolation	ESXi virtual machines	Kubernetes pods on EC2 worker nodes with OpenShift scheduling
Storage	Datastores VMDK SAN NAS	Persistent Volume Claims backed by cloud block or file services
Networking	VLANs NSX segments	VPC subnets security groups OpenShift SDN and optional service mesh
Identity	vCenter roles and groups	AWS IAM roles policies and OpenShift RBAC with service accounts
Scaling	Manual VM resize or scripts	Horizontal Pod Autoscaler and cluster autoscaling with node pools
Monitoring	Host and VM metrics	Prometheus metrics Grafana dashboards centralized logging and tracing

This mapping clarifies that the target platform shifts several responsibilities from host management to declarative policy and controllers.

5.2. Network architecture and hybrid connectivity

A ROSA deployment typically uses private subnets for worker nodes and controlled ingress through load balancers and ingress controllers. Hybrid connectivity is needed during migration for data replication and shared identity services. Empirical analysis of AWS tenant network latency and availability shows that architectures should tolerate variability and transient conditions [14]. This supports deploying workloads across multiple Availability Zones and designing retry timeouts circuit breaking and graceful degradation.

During transition enterprises often require dedicated connectivity for predictable throughput and security segmentation. The framework recommends direct connectivity patterns for steady replication workloads and routing policies that restrict lateral movement. Network segmentation should also align with zero trust principles by minimizing implicit trust and requiring explicit policy for service communication [12].

5.3. Service to service security and service mesh

In microservice rich environments east west traffic dominates. Service mesh introduces a standardized layer for mutual TLS identity-based policy and traffic shaping. A service mesh management framework in network and service management research shows how control planes can support management objectives for multiple concurrent services [19]. This aligns with enterprise needs for consistent mTLS enforcement telemetry correlation and progressive delivery.

Service mesh should be applied selectively because it introduces operational overhead and latency. The architecture should define which namespaces or domains require mesh features and which can rely on baseline Kubernetes networking.

5.4. Storage classes backup and stateful patterns

Stateful workloads require careful selection of storage classes and backup strategies. OpenShift storage abstraction via PVCs enables standardized provisioning but performance characteristics depend on underlying cloud storage type. Baseline measurements from Phase 1 should inform whether block storage is needed for low latency databases or whether shared file storage is sufficient for content repositories.

Backup should include both volume snapshots and logical backups for databases. Disaster recovery design should consider region level resilience based on business requirements and should be tested through regular game days.

5.5. Observability and SRE readiness

Observability must be standardized to avoid per team tooling fragmentation. Kubernetes monitoring approaches using Prometheus and Grafana have been used to provide comprehensive metrics and alerting for cloud native deployments [15]. For enterprise scale the architecture should include a metrics federation strategy log aggregation with structured fields and tracing propagation standards.

Operationally the platform should support SRE style practices including service level indicators service level objectives error budgets and incident response drills. These practices transform modernization from a one-time move into an ongoing reliability program.

5.6. Security controls for cloud native workloads

Security in the target environment includes build time deploy time and runtime controls.

- Image scanning and supply chain controls Container security research identifies exploitation paths across vulnerable base images insecure registries and misconfiguration. It recommends layered defenses including scanning signing and runtime restrictions [16] [17]. Implementations should include admission controls that block untrusted images and enforce minimum security contexts.
- Network policy enforcement Kubernetes networking security research shows that policy enforcement depends on correct integration between Kubernetes components and the network plugin which makes validation essential [23]. Enterprise environments should adopt default deny policies and explicitly allow required communication at namespace and pod levels. Advanced orchestration approaches for network policies further illustrate the importance of dynamic enforcement to keep policy aligned with changing workloads [24].
- IAM least privilege and permission hygiene Least privilege calls in AWS research highlights that permissions should be minimized based on observed API patterns and that achieving correctness requires systematic analysis [11]. ALCMF integrates this by requiring IAM design in Phase 2 and continuous refinement in Phase 5.

6. Addressing the Skills Gap and Operating Model Shift

Technology change without operating model change yields limited benefits. The move from vCenter workflows to declarative Kubernetes manifests demands new skills in YAML policy authoring CI CD pipeline design observability and incident response. The framework recommends establishing a Cloud Center of Excellence or platform engineering team that owns reference architectures reusable templates and guardrails.

Infrastructure as code practices is critical for repeatability and auditability. Research on infrastructure as code identifies practices and challenges and argues that systematic validation reduces defects and drift [25]. Automated testing of infrastructure programs has also been studied and supports the idea that IaC should be treated as software with tests reviews and continuous integration [26].

In practice the operating model should define clear responsibilities: platform team owns cluster lifecycle baseline security and shared services while product teams own application code configuration and service level objectives. This division improves autonomy while preserving governance.

7. Business Value and Roi Measurement

Modernization should be justified with measurable outcomes rather than only qualitative claims. The framework measures value across agility operational excellence cost efficiency and risk reduction.

- Agility metrics Deployment frequency lead time for changes and change failure rate are core indicators. Continuous delivery research links automation and disciplined pipelines with improved flow and quality when implemented with clear validation gates [18].
- Reliability metrics Mean time to recover incident count and error budget burn quantify reliability. Kubernetes self-healing and declarative reconciliation can reduce manual recovery effort but only if probes resource limits and rollout strategies are correctly configured.

- Cost efficiency metrics Cost per transaction resource utilization and idle capacity quantify financial outcomes. Research on cost efficient container orchestration demonstrates that scheduling and heterogeneous resource strategies can reduce total cost and improve utilization compared with default strategies [22]. This supports investing in node pool design placement rules and autoscaler policies rather than treating infrastructure as static.
- Security metrics Policy compliance rate vulnerability remediation time and least privilege drift quantify security posture. Container security research emphasizes that vulnerability management must be continuous due to frequent disclosure and dependency churn [16] [17].

8. Case Study Style Evaluation

To illustrate how ALCMF works we consider a representative enterprise portfolio of 120 applications running on VMware across two data centers. The portfolio includes customer-facing web applications internal APIs batch workloads and several stateful databases.

8.1. Phase 1 outcomes

Discovery identifies that 35 percent of workloads have hidden dependencies on shared NFS mounts and that 20 percent use static IP allowlists for inter application calls. Baseline metrics show that average CPU utilization across the VMware cluster is below 25 percent while memory utilization is moderate which indicates over provisioning. These patterns are consistent with the broader observation that VM estates are often sized for peak and held static due to operational caution.

8.2. Phase 2 architecture decisions

The target architecture selects three ROSA clusters separated by compliance tier: general workloads regulated workloads and a shared services cluster for CI CD logging and registry. Network segmentation is implemented through separate VPCs and controlled routing. Multi Availability Zone deployment is mandatory for tier one services in line with research that shows AWS network conditions can vary and that architectures should tolerate variability [14].

Identity design creates dedicated IAM roles per application domain and maps OpenShift service accounts to those roles. Least privilege reviews are scheduled quarterly to refine policies based on observed API calls which align with the findings that least privilege is difficult to maintain without systematic analysis [11].

8.3. Phase 3 strategy selection results

Using the expanded 6 Rs the portfolio is categorized as follows:

- Rehost: 25 applications including vendor locked systems.
- Replatform: 55 applications suitable for container packaging.
- Refactor: 20 applications with high scaling variance and strong business impact.
- Repurchase: 8 applications replaced by SaaS.
- Retire: 12 applications that are unused or duplicated.
- Retain: 0 after compliance mapping identifies that regulated workloads can move with required controls.

This distribution demonstrates why a single approach would be inefficient. Retiring 12 applications immediately reduces migration scope and risk.

8.4. Phase 4 execution and cutover outcomes

A migration factory approach is used with standardized pipelines and templates. Image scanning and policy enforcement are implemented based on container security guidance that stresses defense in depth [16] [17]. For replatform applications, average migration time per service decreases over successive waves as templates stabilize. Canary releases using service mesh traffic shaping are applied to customer facing APIs where rollbacks must be immediate. Service mesh management research supports the role of standardized control planes in meeting management objectives across services [19].

8.5. Phase 5 optimization results

Autoscaling is enabled for 40 services after load testing. HPA tuning follows research that highlights responsiveness and stability tradeoffs and reduces scaling oscillation [20] [21]. Cluster autoscaling is configured with separate node pools for batch and latency sensitive workloads. Cost reports show a reduction in idle capacity and improved utilization as workloads scale down during off-peak periods. These outcomes align with orchestration cost optimization findings that placement and resource heterogeneity influence total spending [22].

Operational metrics after stabilization show improved mean time to recover due to automated pod restart and faster redeploys. Deployment frequency increases because teams adopt pipeline driven releases. These improvements are sustained because IaC practices are tested and versioned which reduces drift and supports audit requirements [25] [26].

9. Limitations and Threats to Validity

The framework is designed to be practical, yet several limitations apply. First, not all workloads are good candidates for containerization due to licensing kernel dependencies or strict latency requirements. Second service mesh adoption can increase complexity and should be justified by clear needs. Third, the case study style evaluation illustrates plausible outcomes, but actual results depend on organizational maturity governance and workload characteristics. Fourth dependency discovery remains imperfect in many enterprises and missed dependencies can still cause cutover failures as highlighted in migration surveys [2] [3].

Despite these limitations the framework reduces risk by enforcing decision gates measurable artifacts and continuous optimization rather than treating migration as a single event.

10. Conclusion and Future Work

Migration from VMware to ROSA on AWS is not only an infrastructure move. It is an operating model transformation that requires architecture led decisions security by design and continuous optimization. The Architecture Led Cloud Modernization Framework presented in this paper decomposes modernization into five phases and provides concrete guidance across discovery design strategy selection execution and optimization. By mapping VMware constructs to OpenShift and AWS primitives and embedding security governance and observability into the target architecture the framework enables enterprises to avoid the pitfalls of lift and shift while still moving at practical speed.

Future work includes quantitative studies across multiple enterprises to validate the framework outcomes at scale deeper evaluation of service mesh performance tradeoffs and automated methods for dependency discovery and least privilege policy refinement using telemetry driven analysis.

References

[1] A. Tesliuk, I. Bobkov and I. Ilyin, “Kubernetes Orchestration as a Basis for Infrastructure Evolution in Cloud Computing,” 2019, DOI: 10.1109/ISPRAS47671.2019.00016.

[2] M. Fahmideh, G. Low, G. Beydoun and F. Daneshgar, “Cloud migration process: A survey, evaluation framework and open challenges,” Journal of Systems and Software, 2016, DOI: 10.1016/j.jss.2016.06.068.

[3] M. Gholami, F. Daneshgar, G. Low and G. Beydoun, “Cloud migration process: A survey, evaluation framework and open challenges,” Journal of Systems and Software, 2016, DOI: 10.1016/j.jss.2016.06.068.

[4] C. A. Waldspurger, “Memory Resource Management in VMware ESX Server,” 2002, DOI: 10.1145/844128.844146.

[5] M. G. Xavier, M. V. Neves, F. D. Rossi and T. Ferreto, “Performance Evaluation of Container Based Virtualization for High Performance Computing Environments,” 2013, DOI: 10.1109/PDP.2013.41.

[6] S. Shirinbab, A. N. Toosi and R. Buyya, “Performance evaluation of containers and virtual machines in cloud computing,” Concurrency and Computation: Practice and Experience, 2020, DOI: 10.1002/cpe.5693.

[7] X. Lu, J. Yu, Z. Wang and J. Xiong, “Microservices trustworthiness and governance in cloud native systems,” IEEE Access, 2023, DOI: 10.1109/ACCESS.2023.3260147.

[8] Gunda, S. K. G. (2023). The Future of Software Development and the Expanding Role of ML Models. International Journal of Emerging Research in Engineering and Technology, 4(2), 126-129. https://doi.org/10.63282/3050-922X.IJERET-V4I2P113

[9] N. Khalil, M. Khreishah and A. Azeem, “Cloud Security: A Comprehensive Survey,” Computers, 2014, DOI: 10.3390/computers3010001.

[10] R. A. Khan, M. A. Khan, S. U. Khan and M. Ilyas, “A survey of security issues for cloud computing,” Journal of Network and Computer Applications, 2016, DOI: 10.1016/j.jnca.2016.05.010.

[11] P. Gill, W. Dietl and M. Tripunitara, “Least Privilege Calls to Amazon Web Services,” IEEE Transactions on Dependable and Secure Computing, 2023, DOI: 10.1109/TDSC.2022.3171740.

[12] M. Chen, J. Zhou, Y. Li and W. Wang, “Zero Trust Architecture and Security for 6G Networks,” IEEE Network, 2023, DOI: 10.1109/MNET.2023.3326356.

[13] Bicer, C., Murturi, I., Donta, P. K., & Dustdar, S. (2023). Blockchain-based Zero Trust on the edge. arXiv. https://doi.org/10.48550/arXiv.2311.16744

[14] H. Iqbal, A. Singh and M. Shahzad, “Characterizing the Availability and Latency in AWS Network From the Perspective of Tenants,” IEEE ACM Transactions on Networking, 2022, DOI: 10.1109/TNET.2022.3148701.

[15] Shah, H., & Sahatiya, P. (2022). A comparative analysis study of software defect prediction using machine learning algorithms on NASA dataset. Harbin Gongye Daxue Xuebao/Journal of Harbin Institute of Technology, 54(9), 67–76. Retrieved from http://hebgydxxb.periodicales.com/index.php/JHIT/article/view/1301

[16] S. Sultan, I. Ahmad and T. Dimitriou, “Container Security: Issues, Challenges and the Road Ahead,” IEEE Access, 2019, DOI: 10.1109/ACCESS.2019.2911732.

[17] Chekole, E. G., & Ochoa, M. (2023). On the security of containers: Threat modeling, attack analysis, and mitigation strategies. Computers & Security, 128, 103140. https://doi.org/10.1016/j.cose.2023.103140

[18] 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

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

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

[21] 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

[22] 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, DOI: 10.1145/3378447.

[23] F. Minna, S. Maffeis and M. Polverini, "Understanding the Security Implications of Kubernetes Networking," 2021, DOI: 10.1109/EuroSP51992.2021.00029.

[24] 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

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

[26] 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

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

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