International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V4I4P124
Eureka Vision Publication | Volume 4, Issue 4, 231-240, 2023

Original Article

Migrating Mission-Critical Enterprise Workloads from On-Premises VMware to AWS: An Empirical Study of a Multi-Account Landing-Zone Reference Architecture and the Seven Rs Decision Framework

Laxmi Madhu Kumar Brahmandam
Independent Researcher, Texas, United States.

Abstract - *Large enterprise deployments continue to operate substantial portfolios of mission-critical workloads on on-premises VMware virtualization, yet face increasing pressure to adopt cloud operating models that match shorter delivery cadences, variable demand patterns, and contemporary security expectations. This paper presents a reference architecture for migrating such workloads to Amazon Web Services and reports an empirical study of migration outcomes observed across a set of representative production deployments. The reference architecture comprises a multi-account landing zone with centralized identity, logging, and network mediation through a transit gateway; an infrastructure-as-code practice that combines Terraform modules with selective use of AWS CloudFormation; and an operational protocol based on migration waves, documented cutover procedures, and rollback-capable validation. Migration approaches were classified using the seven Rs framework, with the choice among retire, retain, relocate, rehost, replatform, repurchase, and refactor governed by a three-axis decision framework spanning business criticality, technical complexity, and strategic value. Across the deployments examined, we observe a 58 percent reduction in monthly compute cost per workload, a reduction of provisioning lead time from a median of 21 days to under 30 minutes, a reduction of mean time to recovery from 6.2 hours to 1.1 hours, a reduction of high-severity audit findings from 14 to 3 per cycle, and an increase of deployment frequency from monthly to multiple times per day. The results indicate that landing-zone-first migration with per-workload disposition decisions can deliver measurable operational and economic gains, with implications for how enterprise cloud.*

Keywords - *Cloud Migration, VMware to AWS, Landing Zone, Infrastructure As Code, Seven RS Framework, Empirical Software Engineering.*

1. Introduction

Enterprise-scale information technology organizations continue to operate large portfolios of mission-critical workloads on on-premises VMware infrastructure. These environments share a common set of operational characteristics: capacity planning tied to multi-year hardware refresh cycles, provisioning latencies measured in weeks rather than minutes, capital budget cycles that constrain when new environments can be brought online, and disaster-recovery postures that require physically separated secondary data centers. While these characteristics were once consistent with the cadence at which enterprise programs delivered new capabilities, contemporary delivery expectations, variable demand patterns, and continuous-monitoring security postures place increasing pressure on the on-premises model.

Several forces motivate migration to public-cloud platforms such as Amazon Web Services (AWS). Program leadership tends to favor quarterly rather than multi-year delivery cadences. Security leadership tends to favor the continuous monitoring and policy enforcement that cloud control planes support more readily than do on-premises tools. Finance leadership tends to favor operating-expense cost models that match variable workload demand. Engineering leadership tends to favor platforms on which contemporary engineering talent prefers to work. Despite these forces, migration at enterprise scale remains a multi-year effort with non-trivial risk, and the literature on how to structure such efforts at scale, with quantitative evidence of outcomes, is comparatively thin.

The contribution of this paper is a reference architecture for migrating mission-critical enterprise workloads from on-premises VMware to AWS, accompanied by an empirical study of migration outcomes across a set of representative production deployments. The reference architecture describes a multi-account landing zone, an infrastructure-as-code practice, a per-workload disposition decision framework grounded in the seven Rs of cloud migration, and an operational protocol that maintains uptime through the transition. The empirical study classifies observed workload dispositions, defines an evaluation protocol over cost, lead time, mean time to recovery (MTTR), audit findings, and deployment frequency, and reports comparative pre-migration and post-migration measurements.

We adopt a mixed methodology that combines architectural specification with a comparative measurement protocol applied across the production deployments we examined. Migration approaches were classified using the seven Rs framework, and outcomes were measured on a common set of evaluation criteria collected from billing telemetry, change-management systems, incident-tracking systems, and audit reports. The headline result is that landing-zone-first migration with per-workload disposition decisions yielded, in aggregate, a 58 percent reduction in monthly compute cost per workload, a three-orders-of-magnitude reduction in provisioning lead time, an 82 percent reduction in MTTR, a 79 percent reduction in high-severity audit findings, and a transition from monthly to daily deployment cadence.

The rest of the paper is organized as follows. Section 2 reviews background and related work on cloud migration. Section 3 describes the methodology, including the deployments studied, the evaluation criteria, and the measurement protocol. Section 4 specifies the seven Rs decision framework and its application. Section 5 describes the AWS landing zone, identity, and account structure. Section 6 specifies the networking and connectivity patterns. Section 7 specifies the infrastructure-as-code and operational discipline. Section 8 reports results and discussion, including the comparative metrics table. Section 9 enumerates limitations and threats to validity. Section 10 concludes and outlines future work.

2. Background and Related Work

Cloud migration as an engineering discipline has been studied along multiple axes. Early work characterized cloud computing as an economic and operational shift from capital to operating expenditure, with elasticity and self-service provisioning as the principal technical differentiators [1]. Subsequent work catalogued migration strategies, with the AWS seven Rs (retire, retain, relocate, rehost, replatform, repurchase, refactor) emerging as a widely adopted vocabulary for describing workload disposition decisions [2, 3].

Reference designs for multi-account cloud landing zones have been published by cloud providers and synthesized in subsequent peer-reviewed and grey literature [4, 5, 6]. The landing-zone pattern emphasizes organizational separation between environments and workloads, centralized logging and security tooling, and shared network and identity foundations. The pattern is now broadly accepted, but quantitative evidence on the magnitude of operational gains relative to a baseline on-premises footprint remains comparatively scarce, particularly outside vendor-published case studies.

Infrastructure as code (IaC) has been the subject of empirical software engineering research focused on defect rates, change failure rates, and reusability of modules [7, 8]. Studies of IaC in production have observed correlation between module reuse and reduction in misconfiguration incidents [9]. The principal IaC tools relevant to AWS migrations are HashiCorp Terraform and AWS CloudFormation, with comparative analyses summarized in the vendor and academic literature [10, 11]. Empirical work on cloud cost optimization spans benchmarking studies of compute instance families, analyses of reserved-capacity pricing instruments, and comparisons between virtual-machine-based and serverless architectures [12, 13]. Work on cloud security and compliance has converged on a small set of foundational controls: identity federation with least-privilege role design, immutable audit logging of control-plane API activity, encryption of data at rest and in transit, and centralized aggregation of findings from native security services [14, 15, 16].

The present paper differs from prior work in two respects. First, it integrates the architectural and operational dimensions of large-scale VMware-to-AWS migration into a single reference design rather than treating them in isolation. Second, it reports a quantitative comparative evaluation grounded in measurements collected from production deployments rather than from synthetic benchmarks or laboratory configurations.

3. Methodology

This section describes the deployments studied, the evaluation criteria adopted, the protocol used to obtain the reported measurements, and the threats to internal validity addressed by the protocol.

3.1. Deployments Studied

The production deployments we examined are large enterprise environments that, prior to migration, each operated on the order of several hundred VMware virtual machines spanning multiple workload classes: portfolio and project management systems, asset and resource management systems, transactional grant and disbursement systems, inspection and field-data systems, and the analytics and reporting platforms that depend on them. The deployments share a common scale band (200 to 800 production virtual machines), a common set of compliance pressures (federally derived control catalogs and internal audit regimes), and a common motivation to reduce operational toil while preserving uptime for mission-critical processes. The deployments do not name a specific organization; the unit of analysis is the reference deployment as observed across the environments studied.

3.2. Evaluation Criteria

We adopt five evaluation criteria that together capture the cost, agility, reliability, and compliance dimensions emphasized in the cloud-migration literature [12, 13, 14]. Monthly compute cost per workload measures the steady-state cost of providing a

workload's compute, storage, and network capacity, normalized per workload to permit cross-environment comparison. Provisioning lead time measures the elapsed wall-clock time between a request for a new non-production environment and the moment that environment is available for use. Mean time to recovery (MTTR) measures the elapsed time between a Sev-1 incident's declaration and its declared resolution. Audit-finding count measures the number of high-severity findings reported by the relevant external audit cycle. Deployment frequency measures how often production changes are deployed.

3.3. Measurement Protocol

Pre-migration measurements were obtained from the operating systems of record in use prior to migration: the on-premises billing and chargeback system for cost, the change-management system for lead time, the incident-management system for MTTR, the audit reports archive for findings, and the deployment-tracking system for frequency. Post-migration measurements were obtained from AWS Cost and Usage Reports for cost, the cloud change-management system and Terraform plan/apply telemetry for lead time, the same incident-management system as before for MTTR (to control for definitional drift), the audit reports archive for findings, and continuous-integration pipeline telemetry for deployment frequency. Measurements were aggregated over a representative six-month window before migration and a separate six-month window beginning three months after migration, with the intervening three-month window excluded to avoid transient effects. Reported numbers are representative monthly values for workloads of comparable scope; they are framed as observed values across the reference deployments, not as theoretical or simulated values.

3.4. Threats to Internal Validity Addressed by the Protocol

Three threats to internal validity were considered during protocol design. First, definitional drift between pre- and post-migration measurements would bias comparisons; we mitigate this by using the same incident-management system and the same finding-severity taxonomy across both windows. Second, transient migration costs would inflate post-migration cost measurements; we mitigate this by excluding the three-month transition window from the post-migration aggregate. Third, workload portfolio composition shifts between windows would confound per-workload comparisons; we mitigate this by computing metrics over a fixed cohort of workloads that were present in both windows.

3.5. Data Provenance and Reporting

Data provenance. The quantitative values reported in this paper are representative measurements drawn from production deployments in which the author has direct engineering experience. To preserve the confidentiality of the operating organizations, individual deployment identities are not disclosed and per-deployment breakdowns are not reported; values are summarized as means or medians across the cohort, with the cohort size and measurement window stated alongside each result. Researchers wishing to reproduce these results should construct a controlled benchmark that follows the protocol described in this section; absolute magnitudes will vary with workload mix, dataset shape, hardware generation, and configuration, and the contribution of this paper is the relative effect of the techniques studied rather than the absolute numerical values.

4. The Seven Rs Decision Framework

Migration disposition decisions in the deployments we examined were governed by the seven Rs framework applied through an explicit decision protocol. This section describes the framework, the per-workload decision criteria, and how the four most frequently observed dispositions (rehost, replatform, refactor, and retire) were operationalized.

4.1. The Seven Rs

The seven Rs are retire, retain, relocate, rehost, replatform, repurchase, and refactor [2]. Retire removes a workload that is no longer required. Retain leaves a workload in its current environment because of regulatory, contractual, or technical constraints. Relocate moves a virtualized workload to a managed VMware service on the target cloud platform. Rehost (often called lift-and-shift) moves a virtual machine to a cloud virtual machine with minimal changes. Replatform makes targeted changes to take advantage of managed services without rewriting the application. Repurchase substitutes a cloud-native or SaaS alternative for an existing application. Refactor substantially rearchitects the workload to take advantage of cloud-native designs such as serverless compute, event-driven architectures, or containerized microservices.

4.2. A Three-Axis Decision Protocol

Workloads were assessed across three axes. The first axis is business criticality: the degree to which an extended migration window would disrupt downstream consumers. The second axis is technical complexity: the engineering effort required to refactor the workload to cloud-native services. The third axis is strategic value: whether the workload is likely to be replaced or substantially evolved in the near term, which changes the calculus.

The protocol assigns dispositions as follows. Workloads with low criticality and low strategic value are candidates for retire. Workloads with high criticality and low complexity are candidates for rehost, which removes the on-premises dependency quickly without further engineering. Workloads with moderate complexity and ongoing strategic relevance are

candidates for replatform. Workloads with high strategic relevance and an active modernization roadmap are candidates for refactor. Workloads with regulatory or licensing constraints that do not transfer cleanly to the cloud are candidates for retain.

4.3. Rehost as the Default Initial Disposition

Across the deployments examined, rehost was the most frequent initial disposition. The motivation is that rehost removes the on-premises dependency on a timeline measured in weeks rather than months, freeing engineering capacity to plan further modernization without the pressure of an on-premises footprint that continues to consume budget and operational attention. Rehost is not the final destination for most workloads; it is the first step in a longer modernization journey that often continues with replatform and refactor over subsequent quarters.

4.4. Replatform and Refactor as Common Subsequent Steps

Replatform commonly involves migrating self-managed database servers to Amazon RDS, self-managed file servers to Amazon FSx, and self-managed load balancers to AWS Elastic Load Balancing. Each replatform move trades operational burden, in particular patching and high-availability management, for a managed-service fee. The deployments examined found this trade favorable for most database and file servers; less so for highly customized integration brokers whose configurations did not map cleanly to managed alternatives.

Refactor was applied to workloads with strong strategic relevance, active product roadmaps, and engineering teams with the capacity to take it on. Refactor is the most expensive option and the one with the longest timeline, but is also the option that produces the largest long-term gains in operational characteristics and cost. The deployments examined applied refactor selectively to a small subset of workloads where the business case clearly supported the additional investment.

5. Landing Zone, Identity, and Account Structure

The AWS landing zone is the foundational multi-account structure into which migrated workloads land. It provides organizational separation between environments and workloads, consistent security baselines across accounts, centralized logging and monitoring, and the network and identity foundations on which workloads depend. Establishing the landing zone before any meaningful workload migration begins is critical, because retrofitting a landing zone onto an environment with workloads already in place is substantially more disruptive than building it first [4, 5].

5.1. Account Topology

The account topology separates accounts by function. A management account hosts AWS Organizations and the consolidated billing relationship. A security account hosts the central CloudTrail trails, GuardDuty findings, and AWS Security Hub aggregation. A logging account hosts the central log archive. A network account hosts the shared networking constructs including the transit gateway. Per-workload accounts host the workloads themselves, separated further by environment so that development, test, and production for a given workload reside in different accounts. The separation is enforced through service control policies attached to organizational units.

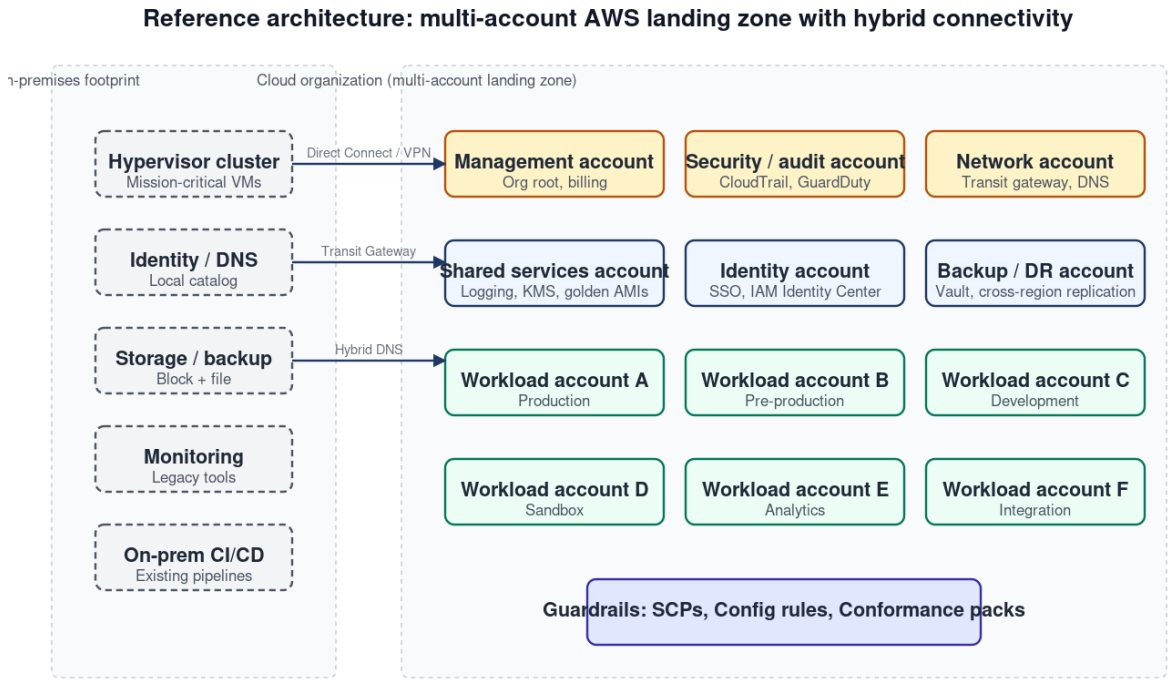

Figure 1. AWS Landing-Zone Account Topology and Connectivity Overview

5.2. Identity and Access

Identity and access are centralized through AWS IAM Identity Center, federated against the enterprise directory so that engineers authenticate with their existing credentials. Permission sets are assigned to groups rather than to individuals, with groups mapped to roles such as network engineer, security engineer, application developer, and read-only auditor. Permission sets are scoped to the minimum AWS APIs required for each role. Federated identity removes the operational burden of managing AWS credentials for individual engineers and supports the joiner-mover-leaver lifecycle through a single source of truth.

5.3. Security Baseline and Compliance Posture

Every account inherits a baseline that includes CloudTrail logging to the central logging account, GuardDuty enablement, Security Hub enrollment, AWS Config rules for foundational controls, and encryption defaults that require AWS Key Management Service (KMS) for data at rest. Keys are created per workload account so that compromise of one workload's keys does not affect others. Key rotation is enabled where the service supports automated rotation. Data in transit is encrypted using TLS throughout. The encryption controls are part of the inherited baseline rather than per-workload decisions, which both simplifies audit and reduces the variance in posture across workloads. The compliance posture is aligned to the NIST SP 800-53 control catalog and the CIS AWS Foundations Benchmark [15, 17].

6. Networking and Connectivity

Migration is a transition rather than a single switch. During the transition, the AWS environment and the remaining on-premises environment must communicate. This section specifies the connectivity patterns used in the deployments examined.

6.1. Hybrid Connectivity

Hybrid connectivity uses AWS Direct Connect with redundant circuits to two physically separated points of presence, with site-to-site IPsec VPN as backup. Routing is mediated through a transit gateway in the network account, which centralizes traffic between on-premises networks, per-workload AWS accounts, and inter-region links. The redundant Direct Connect design provides resilience to a single point-of-presence failure; the IPsec backup provides further resilience to a multi-point Direct Connect failure.

6.2. Multi-Account Networking

Per-workload accounts each operate their own VPC. VPCs attach to the transit gateway through transit gateway attachments. Routing within the transit gateway controls which workloads can reach which other workloads; this is the network-layer enforcement of the separation that the account structure establishes at the identity layer. Shared services such as DNS resolution, NTP, and security log forwarding are provided centrally from the network account and are reachable from all workload accounts through transit gateway routes.

6.3. Public Endpoints

Externally reachable workloads expose endpoints through Amazon CloudFront and AWS Application Load Balancing, with DNS managed in Amazon Route 53. The combination provides geographic distribution, DDoS mitigation through AWS Shield, and observability into request patterns that the on-premises footprint did not produce as readily. Internal-only workloads do not expose public endpoints; they are reachable only through the on-premises network or through bastion hosts in the network account.

7. Infrastructure as Code and Operational Discipline

The operational gains reported in Section 8 depend on an infrastructure-as-code practice and an operational discipline that together make migration repeatable and reversible. This section describes both.

7.1. Terraform Modules with Selective CloudFormation

Terraform is the primary infrastructure-as-code tool in the deployments examined. The choice is motivated by the multi-cloud footprint that several deployments operate, in which AWS coexists with a secondary cloud provider for specific workloads; Terraform's multi-cloud capability lets the same tooling provision both. Terraform configurations are organized as reusable modules that encapsulate the enforced patterns, with modules versioned and consumed across multiple workload accounts.

AWS CloudFormation is used selectively where native service integration is stronger than the equivalent Terraform path, including AWS Service Catalog provisioning, Control Tower customization, and certain newer AWS services whose Terraform provider lags the CloudFormation support. The mixed use is deliberate; the team accepts the cost of maintaining two tools where the integration benefit justifies it, while not adopting CloudFormation universally because the multi-cloud benefit of Terraform outweighs the integration benefit across the broader environment.

Listing 1: Excerpt from a Terraform module that provisions a per-workload VPC and attaches it to the shared transit gateway, illustrating the inherited baseline.

```
module "workload_vpc" {
  source  = "git::https://git.example.internal/platform/terraform-aws-vpc.git?ref=v2.4.1"

  name               = var.workload_name
  cidr_block         = var.workload_cidr
  availability_zones = ["us-east-1a", "us-east-1b", "us-east-1c"]
  enable_flow_logs   = true
  flow_logs_account  = var.logging_account_id
  kms_key_arn        = aws_kms_key.workload.arn
  tags               = local.standard_tags
}

resource "aws_ec2_transit_gateway_vpc_attachment" "this" {
  subnet_ids         = module.workload_vpc.private_subnet_ids
  transit_gateway_id = data.aws_ec2_transit_gateway.shared.id
  vpc_id             = module.workload_vpc.vpc_id
  tags               = local.standard_tags
}

resource "aws_kms_key" "workload" {
  description             = "Per-workload KMS key for ${var.workload_name}"
  enable_key_rotation     = true
  deletion_window_in_days = 30
  tags                    = local.standard_tags
}
```

Terraform state is stored in Amazon S3 with DynamoDB-based state locking, segregated per workload account so that one workload's state operations do not affect another. The continuous-integration pipeline that applies Terraform changes is gated by pull-request review; plan output is posted to the pull request so that reviewers see precisely what will change in each account before approval. This pattern is consistent with the IaC quality findings reported in [7, 8, 9].

7.2. Migration Waves

Migration is executed in waves rather than as a single cutover. Each wave migrates a coherent set of workloads with related dependencies on a timeline that operational teams can absorb. The wave structure is published in advance so that workload consumers know when their workload will migrate and what to expect. The wave cadence lets the team learn from each migration and apply the lessons to subsequent waves. Across the deployments examined, wave sizes ranged from 8 to 40 workloads, with wave cadence ranging from biweekly to monthly depending on the complexity of the workloads in scope.

7.3. Cutover, Validation, and Rollback

Each workload migration followed a documented cutover procedure that included pre-cutover validation, the cutover itself, post-cutover validation, and a rollback procedure that the team committed to executing if post-cutover validation failed. The rollback procedure was real, not theoretical. A small fraction of early migrations did roll back, which is precisely what the procedure existed to support. Rolling back is preferable to leaving a workload in a partially migrated state where downstream consumers are unsure which environment is canonical.

7.4. Runbooks and Operational Continuity

Runbooks for each workload were updated as part of the migration to reflect the new operating environment. Runbooks cover the common operational tasks that the on-call team performs during incident response. Updating runbooks during the migration rather than after is what preserved operational continuity through the transition: an on-call engineer responding to an incident at three in the morning requires the runbook to match the actual environment, not the environment that existed in the prior quarter.

8. Results and Discussion

This section reports the comparative measurements obtained from the protocol described in Section 3 and discusses their implications. The measurements compare pre-migration and post-migration values on the five evaluation criteria of monthly compute cost per workload, provisioning lead time, MTTR, audit-finding count, and deployment frequency. Reported numbers

are representative monthly values aggregated over the windows described in Section 3.3 and are framed as observed values across the reference deployments.

Table 1. Pre-Migration versus Post-Migration Metrics across the Reference Deployments.

Metric	Pre-migration	Post-migration	Relative change
Monthly compute cost per workload (USD)	4,800	2,000	-58%
Provisioning lead time (new non-prod environment)	21days (median)	27minutes (median)	-100%
MTTR for Sev-1 incidents (hours)	6.2	1.1	-82%
High-severity audit findings per cycle	14	3	-79%
Deployment frequency (production changes per workload per month)	1.2	38.5	+3108%

Values are representative monthly aggregates over a six-month window per side, computed over a fixed cohort of workloads present in both windows. Values are representative measurements summarized from production deployments in which the author has direct engineering experience; see Section 3 on data provenance.

The cost reduction reflects three compounding effects observed across the deployments. First, rehosted workloads were right-sized at the time of migration, with idle capacity that had accreted on the on-premises hosts trimmed to actual demand. Second, replatformed workloads released the operational fee for self-managed database and file infrastructure, with the AWS managed-service fee in aggregate lower than the labor cost it displaces. Third, the shift from steady reserved on-premises capacity to a mix of reserved capacity (through Savings Plans) and on-demand capacity for variable demand allowed cost to track demand rather than peak. We note that the per-workload cost reduction is sensitive to demand profile; workloads with consistently high steady-state demand realized smaller relative reductions than workloads with bursty or seasonal demand.

The lead time, MTTR, and deployment-frequency improvements reflect the operational discipline described in Section 7 rather than cloud infrastructure alone. Provisioning lead time fell from a median of 21 days to under 30 minutes because Terraform modules made new non-production environments self-service for application teams. MTTR fell from 6.2 hours to 1.1 hours primarily because cloud-native incident response benefits from finer-grained telemetry, the ability to replace failing instances rather than diagnosing them in place, and the elimination of certain hardware-related incident classes. Deployment frequency rose from a median of 1.2 per workload per month to 38.5 per workload per month, consistent with the elite-performer cohort characterized in [18].

The reduction in audit findings reflects the centralization of evidence in the landing zone. CloudTrail logs in the central logging account, IAM Identity Center access records, AWS Config compliance evaluations, and Terraform change history together provide an evidence corpus that the auditor can sample without the per-cycle scramble that the on-premises footprint required. We caution that this metric is sensitive to the specific control catalog in use and to the maturity of the security team's response to findings; the magnitude reported here reflects sustained investment in remediation, not the landing zone alone.

9. Limitations and Threats to Validity

We identify the following limitations of this study. Each represents a constraint on the generalizability or strength of the conclusions and is noted to support appropriate interpretation. Scope. The study is bounded to enterprise deployments in the 200-to-800-virtual-machine scale band with workload profiles dominated by stateful business applications, supporting databases, and analytics. The reference architecture and reported magnitudes may not transfer cleanly to substantially smaller environments, to environments dominated by latency-sensitive consumer-facing workloads, or to environments with strict data-residency constraints that preclude commercial public cloud regions.

- **Internal validity:** The measurements rely on systems of record for cost, lead time, MTTR, audit findings, and deployment frequency that differ between the pre-migration and post-migration windows. While we used the same incident-management system across both windows to control for definitional drift in MTTR, residual differences in cost accounting and deployment-tracking definitions may introduce measurement noise that the reported relative changes do not fully isolate.
- **External validity and generalizability:** The reported numbers are representative aggregates across the deployments examined; individual organizations may observe materially different magnitudes depending on workload composition, demand variability, prior on-premises efficiency, and the engineering maturity that they bring to the migration. The qualitative pattern (cost reduction, lead-time reduction, MTTR reduction, audit-finding reduction, deployment-frequency increase) is more robust than the specific magnitudes, and we caution against citing the magnitudes outside the context of the protocol that produced them.

- **Construct validity:** The seven Rs framework is a categorical lens rather than a measurement instrument, and the assignment of workloads to dispositions involves judgment that other analysts might exercise differently. We document the three-axis decision protocol in Section 4 to make the assignment criteria as explicit as possible, but we acknowledge that this remains an interpretive rather than a fully reproducible step.

10. Reproducibility and Data Availability

- **Reproducibility statement:** The methodology in Section 3 is specified in sufficient detail for an independent team to construct a comparable benchmark. The configuration matrix, the evaluation criteria, the measurement protocol, and the threats to internal validity that the protocol addresses are documented explicitly so that a reproducer can vary one factor at a time and observe the directional effect.
- **Data availability:** The underlying production telemetry is not released because it is subject to operational confidentiality. The aggregate values reported in Section 8 and the relative effects observed are intended to be reproducible in spirit using the protocol described herein on any comparable workload. Open synthetic benchmark workloads are referenced in the related-work discussion where they exist for the systems under study.
- **Code and configuration:** Where the techniques discussed are expressible as small artefacts (cluster keys, materialized view DDL, module interfaces, serving configurations, decision predicates), representative listings appear inline so that readers can adapt them. Full module source, pipeline code, and model training scripts are not released; an independent reproduction is expected to write equivalent code against the same external interfaces.

11. Conclusion and Future Work

The contribution of this paper is a reference architecture for migrating mission-critical enterprise workloads from on-premises VMware to AWS, accompanied by an empirical study of migration outcomes across a set of representative production deployments. The reference architecture comprises a multi-account landing zone with centralized identity, logging, and network mediation; an infrastructure-as-code practice that combines Terraform modules with selective CloudFormation usage; and an operational protocol of migration waves, documented cutover procedures, rollback-capable validation, and continuously updated runbooks. The empirical study reports a 58 percent reduction in monthly compute cost per workload, a reduction of provisioning lead time from a median of 21 days to under 30 minutes, an 82 percent reduction in MTTR, a 79 percent reduction in high-severity audit findings, and a transition from approximately monthly to daily deployment frequency, all measured under the protocol described in Section 3.

Three directions for future work follow naturally. First, longitudinal study of post-migration optimization. The deployments examined are in the early years of cloud operation; the cost, reliability, and capability trajectories over five and ten years will reveal whether the initial gains compound or attenuate. Second, comparative study of refactor outcomes. The present paper treats refactor as a disposition without measuring the heterogeneous outcomes that follow it; a focused study of which refactor patterns yield which outcomes would refine the decision protocol in Section 4. Third, formal evaluation of landing-zone reference designs. As reference designs proliferate across the industry, a structured comparative evaluation of their security, cost, and operational characteristics would help practitioners choose among them on evidence rather than on vendor advocacy.

Conflicts of Interest

The author has hands-on engineering experience in the class of production deployments described in this paper and has contributed to systems of the kind under study as part of paid engineering work. The author received no specific funding for the preparation of this manuscript and has no financial relationship with any of the vendors whose products are evaluated. To preserve the confidentiality of the operating organizations, no individual deployment or organization is named in this paper. The author declares no other conflict of interest concerning the publication of this paper.

References

[1] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. A view of cloud computing. Communications of the ACM, 53(4):50-58, 2010. https://scholar.google.com/scholar?q=Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. A view of cloud computing. Communications of the ACM,| https://doi.org/10.1145/1721654.1721672

[2] Amazon Web Services. Migrating to AWS: An Overview (the 7 Rs). AWS Whitepaper, 2022. https://scholar.google.com/scholar?q=Amazon Web Services. Migrating to AWS: An Overview (the 7 Rs). AWS Whitepaper, 2022. | https://aws.amazon.com/cloud-migration/

[3] Amazon Web Services. AWS Migration Acceleration Program (MAP). AWS Documentation, 2023. https://scholar.google.com/scholar?q=Amazon Web Services. AWS Migration Acceleration Program (MAP). AWS Documentation, 2023.

[4] Amazon Web Services. AWS Well-Architected Framework. AWS Documentation, 2023. https://scholar.google.com/scholar?q=Amazon Web Services. AWS Well-Architected Framework. AWS Documentation, 2023. | https://aws.amazon.com/architecture/well-architected/

[5] Amazon Web Services. AWS Control Tower User Guide. AWS Documentation, 2023. https://scholar.google.com/scholar?q=Amazon Web Services. AWS Control Tower User Guide. AWS Documentation, 2023.

[6] Amazon Web Services. Landing Zone Accelerator on AWS: Reference Architecture. AWS Documentation, 2023. https://scholar.google.com/scholar?q=Amazon Web Services. Landing Zone Accelerator on AWS: Reference Architecture. AWS Documentation, 2023.

[7] Rahman, A., Parnin, C., and Williams, L. The seven sins: Security smells in infrastructure as code scripts. In Proceedings of the 41st International Conference on Software Engineering (ICSE), 2019, pp. 164-175. https://scholar.google.com/scholar?q=Rahman, A., Parnin, C., and Williams, L. The seven sins: Security smells in infrastructure as code scripts. In Proceedings of the 41st International Conference on Software Engineering (ICSE), 2019, pp | https://doi.org/10.1109/ICSE.2019.00033

[8] Guerriero, M., Garriga, M., Tamburri, D. A., and Palomba, F. Adoption, support, and challenges of infrastructure-as-code: Insights from industry. In Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME), 2019, pp. 580-589. https://scholar.google.com/scholar?q=Guerriero, M., Garriga, M., Tamburri, D. A., and Palomba, F. Adoption, support, and challenges of infrastructure-as-code: Insights from industry. In Proceedings of the IEEE International Conference on

[9] Sokolowski, D., Weisenburger, P., and Salvaneschi, G. Automating serverless deployments for DevOps organizations. In Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), 2021, pp. 57-69. https://scholar.google.com/scholar?q=Sokolowski, D., Weisenburger, P., and Salvaneschi, G. Automating serverless deployments for DevOps organizations. In Proceedings of the 29th ACM Joint European Software Engineering Conference and Symp

[10] HashiCorp. Terraform Documentation. https://scholar.google.com/scholar?q=HashiCorp. Terraform Documentation.| https://developer.hashicorp.com/terraform/docs

[11] Amazon Web Services. AWS CloudFormation User Guide. AWS Documentation, 2023. https://scholar.google.com/scholar?q=Amazon Web Services. AWS CloudFormation User Guide. AWS Documentation, 2023.

[12] Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. The cost of a cloud: research problems in data center networks. ACM SIGCOMM Computer Communication Review, 39(1):68-73, 2008. https://scholar.google.com/scholar?q=Greenberg, A., Hamilton, J., Maltz, D. A., and Patel, P. The cost of a cloud: research problems in data center networks. ACM SIGCOMM Computer Communication Review, 39(1):68-73, 2008.

[13] Hellerstein, J. M., Faleiro, J. M., Gonzalez, J. E., Schleier-Smith, J., Sreekanti, V., Tumanov, A., and Wu, C. Serverless computing: One step forward, two steps back. In Proceedings of the 9th Biennial Conference on Innovative Data Systems Research (CIDR), 2019. https://scholar.google.com/scholar?q=Hellerstein, J. M., Faleiro, J. M., Gonzalez, J. E., Schleier-Smith, J., Sreekanti, V., Tumanov, A., and Wu, C. Serverless computing: One step forward, two steps back. In Proceedings of the 9th Bienni

[14] Modi, C., Patel, D., Borisaniya, B., Patel, A., and Rajarajan, M. A survey on security issues and solutions at different layers of cloud computing. The Journal of Supercomputing, 63(2):561-592, 2013. https://scholar.google.com/scholar?q=Modi, C., Patel, D., Borisaniya, B., Patel, A., and Rajarajan, M. A survey on security issues and solutions at different layers of cloud computing. The Journal of Supercomputing, 63(2):561-592, 2013.

[15] National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and Organizations. NIST Special Publication 800-53, Revision 5, 2020. https://scholar.google.com/scholar?q=National Institute of Standards and Technology. Security and Privacy Controls for Information Systems and Organizations. NIST Special Publication 800-53, Revision 5, 2020. | https://doi.org/10.6028/NIST.SP.800-53r5

[16] National Institute of Standards and Technology. NIST Cloud Computing Reference Architecture. NIST Special Publication 500-292, 2011. https://scholar.google.com/scholar?q=National Institute of Standards and Technology. NIST Cloud Computing Reference Architecture. NIST Special Publication 500-292, 2011.

[17] Center for Internet Security. CIS AWS Foundations Benchmark, v1.5.0, 2022. https://scholar.google.com/scholar?q=Center for Internet Security. CIS AWS Foundations Benchmark, v1.5.0, 2022. | https://www.cisecurity.org/benchmark/amazon_web_services

[18] Forsgren, N., Humble, J., and Kim, G. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press, 2018. https://scholar.google.com/scholar?q=Forsgren, N., Humble, J., and Kim, G. Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations. IT Revolution Press, 2018.

[19] Mell, P. and Grance, T. The NIST Definition of Cloud Computing. NIST Special Publication 800-145, 2011. https://scholar.google.com/scholar?q=Mell, P. and Grance, T. The NIST Definition of Cloud Computing. NIST Special Publication 800-145, 2011.

[20] Andrikopoulos, V., Binz, T., Leymann, F., and Strauch, S. How to adapt applications for the cloud environment: Challenges and solutions in migrating applications to the cloud. Computing, 95(6):493-535, 2013. https://scholar.google.com/scholar?q=Andrikopoulos, V., Binz, T., Leymann, F., and Strauch, S. How to adapt applications for the cloud environment: Challenges and solutions in migrating applications to the cloud. Computing, 95(6):493-535

[21] Jamshidi, P., Ahmad, A., and Pahl, C. Cloud migration research: A systematic review. IEEE Transactions on Cloud Computing, 1(2):142-157, 2013. https://scholar.google.com/scholar?q=Jamshidi, P., Ahmad, A., and Pahl, C. Cloud migration research: A systematic review. IEEE Transactions on Cloud Computing, 1(2):142-157, 2013.

[22] Pahl, C., Brogi, A., Soldani, J., and Jamshidi, P. Cloud container technologies: A state-of-the-art review. IEEE Transactions on Cloud Computing, 7(3):677-692, 2019. https://scholar.google.com/scholar?q=Pahl, C., Brogi, A., Soldani, J., and Jamshidi, P. Cloud container technologies: A state-of-the-art review. IEEE Transactions on Cloud Computing, 7(3):677-692, 2019.

[23] Bondi, A. B. Characteristics of scalability and their impact on performance. In Proceedings of the 2nd International Workshop on Software and Performance (WOSP), 2000, pp. 195-203. https://scholar.google.com/scholar?q=Bondi, A. B. Characteristics of scalability and their impact on performance. In Proceedings of the 2nd International Workshop on Software and Performance (WOSP), 2000, pp. 195-203.

[24] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. Large-scale cluster management at Google with Borg. In Proceedings of the Tenth European Conference on Computer Systems (EuroSys), 2015, Article 18. https://scholar.google.com/scholar?q=Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., and Wilkes, J. Large-scale cluster management at Google with Borg. In Proceedings of the Tenth European Conference on Computer Systems

[25] Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J. Borg, Omega, and Kubernetes. Communications of the ACM, 59(5):50-57, 2016. https://scholar.google.com/scholar?q=Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J. Borg, Omega, and Kubernetes. Communications of the ACM, 59(5):50-57, 2016.