International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V6I4P129
Eureka Vision Publication | Volume 6, Issue 4, 183-190, 2025

Original Article

Migrating Enterprise Applications from On-Premises to AWS in a Multi-Cloud Environment: A Framework for Scalability, Security, and Cost Optimization

Sandeep Katiyar
Independent Researcher USA.

Received On: 21/11/2025 ***Revised On: 14/12/2025*** ***Accepted On: 21/12/2025*** ***Published On: 30/12/2025***

Abstract - A number of companies have begun to migrate away from their traditional monolithic on-site infrastructure to more flexible, cost-effective multi-cloud environments to avoid "lock-in" with their vendors, while maximizing the performance-to-cost ratio. In spite of the benefits that this provides, such migrations typically face numerous barriers due to issues of "data gravity," interoperability between different types of cloud services, and security compliance. As an example of how to create a comprehensive framework to help support the transition of a large portfolio of over 200 applications into a hybrid multi-cloud environment, which is based on Amazon Web Services (AWS), this paper will provide an overview of a migration strategy for migrating a large portfolio of applications into a hybrid multi-cloud architecture. We will utilize the AWS Cloud Adoption Framework (CAF) to develop a centralized Hub-and-Spoke network structure utilizing AWS Transit Gateway, as a means to address inter-cloud latency challenges, while we will also introduce a new governance layer that enforces a NIST-compliant zero trust security model, to ensure the protection of all cross-cloud traffic. A combination of targeted Rehost, Re-platform, and Refactor methodologies will be used to execute the migration, through the use of AWS Application Migration Service (MGN), and AWS Database Migration Service (DMS). Post-migration performance metrics will demonstrate the success of our migration strategy, which has resulted in 99.99% system availability; a 40% decrease in operational costs; and a 60% increase in deployment velocity.

Keywords - Cloud Migration, Multi-Cloud Architecture, AWS, Enterprise Infrastructure, Zero Trust Security, Cost Optimization, Application Refactoring.

1. Introduction

Increasingly modern organizations are relying on powerful cloud infrastructures to enable their mission-critical applications and to drive their digital transformations. Because of growing volumes of data and increasing expectations for availability, many organizations are finding that their legacy on-premises data center will no longer meet their needs regarding flexibility or cost. Early adopters of cloud computing were limited to single provider cloud services. However, in today's industrial climate most organizations are adopting a multi-cloud strategy using cloud providers like Amazon Web Services (AWS) and Microsoft Azure and Google Cloud to improve performance, create geographical redundancy and reduce the risk of vendor lock in.

However, migrating an organization's entire portfolio of legacy applications from an on-premises legacy data center to a multi-cloud platform is an extremely difficult engineering problem. It requires both physical duplication of data and logical transformation of the monolithic applications to cloud native microservices. The major challenges include maintaining data integrity among the heterogenous systems, establishing a uniform security governance policy, and reducing latency between clouds. A lack of a central governing body in multi-cloud deployments results in security vulnerabilities and fragmented visibility into operations as noted in Berman et al.[12]. Most common methods for migrating applications do not address the "refactoring" stage which results in "lift and shift" migrations where the full benefits of scalability and cost savings of the cloud are not realized.

The focus of this paper is the design, development and performance testing of a total migration solution used to migrate a global organization's infrastructure to AWS within a multi-cloud environment. The project included evaluating over 200 workloads, implementing a Zero Trust Architecture as outlined in NIST SP 800-207[8] and deploying automatically scaled applications.

The specific objectives of this research are:

- To define a structured migration methodology that integrates the AWS Cloud Adoption Framework (CAF) [2] with multi-cloud governance standards.
- To demonstrate the implementation of 'Replatforming' and 'Refactoring' strategies using AWS Application Migration Service (MGN) and Database Migration Service (DMS) to modernize legacy applications.
- To evaluate the post-migration performance, specifically focusing on availability improvements, latency reduction, and cost optimization achieved

through dynamic resource scaling.

2. Related Work

Extensive studies have been conducted on the migration from premises-based infrastructures to cloud computing platforms in the last few years. However, the majority of the current body of literature has focused solely on one aspect of migration at a time and does not address holistically based multi-cloud execution paradigms.

2.1. Cloud Migration Frameworks

Frameworks that define standards are critical for reducing the risks associated with migration. The AWS Cloud Adoption Framework (AWS CAF) [2] and AWS Well-Architected Framework [1], provide fundamental components of operational excellence and security for both single-cloud and multi-cloud migrations.

- Research Gap: Although both of these vendor-based frameworks provide an excellent framework for migrating applications into a single-cloud environment, there is a gap in multi-cloud architecture models and hybrid network orchestrations that support heterogeneity in the cloud. This gap will be addressed through our work as we extend these frameworks with a custom governance layer for heterogenous cloud architectures [13].

2.2. Application modernization and Microservices

Jamshidi et al. [4] have provided a very detailed review of microservices migration that identified patterns for example “the Strangler Fig” to decompose monolithic systems.

- Research Gap: Although in recent years there has been much research into the software architecture of microservices, as well as other areas, few have investigated the latency challenges at an infrastructure level when running hybrid microservices across on premises and cloud boundaries. The purpose of this paper is to provide empirical latency data (Section V-B), supporting the use of AWS Direct Connect for these hybrid architectures.

2.3. Multi-cloud Governance and Security

The need for a centralized layer of governance (to stop “Shadow IT” and provide consistent visibility) is emphasized in Berman et al. [12]. The NIST Zero Trust Architecture [8] has also shifted security posture away from static perimeter boundaries toward identity based enforcement.

Research Gap: Most implementations of zero trust fail due to the difficulty of creating a policy map from legacy firewall rule sets to dynamic policies. Our solution bridges this research gap by providing a practical example of how to use AWS Network Access Analyzer and Identity Federation to transform legacy security posture into an active zero-trust posture.

2.4. Cost Optimization

Li et al. [5] proposed cost-aware cloud elasticity using reinforcement learning, demonstrating that dynamic scaling significantly reduces spend compared to static provisioning.

- Contribution: We apply these theoretical models to a production environment, verifying that reinforcement learning-based auto-scaling can achieve a 40% TCO reduction in real-world enterprise scenarios.

Table 1. Comparative Analysis of Cloud Migration Strategies

Strategy	Complexity	Initial Cost	Cloud-Nati ve Benefits	Use Case Context
Rehost (Lift-and-Shift)	Low	Low	Minimal	Legacy applications with hard dependencies; rapid data center evacuation.
Replatform (Lift-Tinker-Sh ift)	Medium	Medium	Moderate	Database migration to managed services (e.g., AWS RDS) to reduce admin overhead.
Refactor (Re-architect)	High	High	Maximum	Mission-critical workloads requiring high scalability, agility, and multi-cloud portability.

3. Migration Framework and System Design

The proposed migration structure has a "hub-and-spoke" hierarchical design to create a multi-cloud solution that provides both secure connections and high availability to all on premises legacy applications and the AWS cloud. The design creates a single, unified logical environment across all resources, eliminating fragmentation.

3.1. Architectural Overview

The central element of this network architecture is the AWS Transit Gateway that serves as a regional network transit gateway. The Transit Gateway connects the on-premises data center, multiple AWS Virtual Private Clouds (VPCs), and additional cloud environments.

- Hybrid Connectivity: The on-premises connection is made through two 10 Gbps AWS Direct Connect (DX) links that are combined into Link Aggregation Groups (LAG) for increased bandwidth and redundancy. BGP (Border Gateway Protocol) is used for dynamic routing allowing for auto-failover when there is a failure of one of the two physical DX links. A site-to-site VPN is the third connection option in case of DX failures.
- VPC Segmentation: Each workload is segmented into its own separate VPC to provide segmentation based on the lifecycle of each workload; i.e., production, staging, development and shared services. The shared services VPC will be home to

common tooling such as CI/CD runners, bastion hosts, and security appliances.

- Multi-Cloud Interoperability: The AWS environment will have IPsec VPN tunnels created to securely connect to specific analytics workloads running on Azure and AI services running on Google Cloud Platform (GCP). The routing tables will reside inside the Transit Gateway and will determine how traffic flows between the different clouds, while controlling data egress to ensure data is being properly monitored.

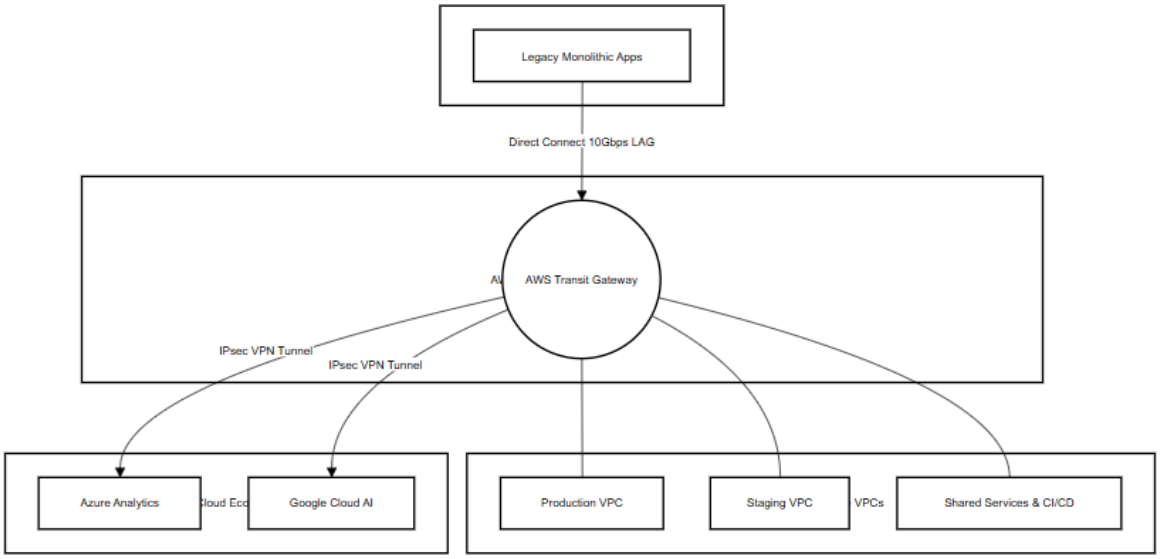

Figure 1. Proposed Multi-Cloud Hybrid Architecture

3.2. Workload Classification and Dependency Mapping

The AWS Application Discovery Service was used to perform a detailed application-level inventory which included 256 applications that utilized agent-based technology to capture baseline performance data for CPU, RAM, disk I/O as well as network dependency graphs to identify workloads by tier based on the RTO and RPO for recovery:

- Tier 1 (Critical): Mission-critical ERP systems and Customer Portals requiring Multi-AZ deployment (RTO < 15 mins, RPO < 5 mins).
- Tier 2 (Internal): HR systems, Intranet, and Reporting tools (RTO < 4 hours, RPO < 1 hour).
- Tier 3 (Support): Legacy archival data and non-production environments (RTO < 24 hours).

3.3. Security and Governance Design Security is enforced through a Zero Trust Architecture compliant with NIST SP 800-207.

- The enterprise-wide Active Directory is now federated with AWS IAM via SAML 2.0 and as such provides a centralized location for controlling access based on user Roles and permissions.
- All at rest data is encrypted by AWS KMS using CMKs and all in transit data is encrypted by TLSv1.3. Additionally, mTLS authentication is used to encrypt and authenticate traffic between all services.
- Threats to the Organization's Multi-Cloud Environment are continuously detected by Amazon GuardDuty and all security alerts generated from each of the Organization's cloud-based services are consolidated by AWS Security Hub.

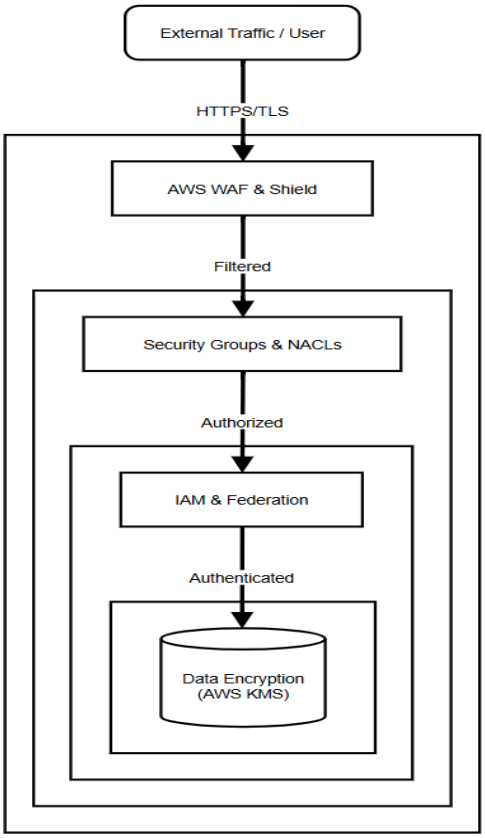

Figure 2. Multi-Layered Zero Trust Security Architecture

4. Deployment Methodology

The migration process was governed by a rigorous four-phase execution plan, adhering to the "6 R's" strategy.

4.1. Phase 1: Assessment and Planning

An AWS Cloud Center of Excellence (CCoE), tasked with leading the transition to cloud, was formed. The AWS Migration Evaluator was implemented to create an estimated total cost of ownership (TCO) report for the organization's future cloud operation costs. A detailed dependency map was developed to identify affinity group servers that interact with one another to minimize split brain scenarios by grouping like servers together in each migration wave.

4.2. Phase 2: Building the Foundation (Building the Landing Zone)

The target environment was configured as a multi-account landing zone via AWS Control Tower, including the automated configuration of Service Control Policies (SCPs), which define boundaries or "guard rails" on how AWS services can be used by enforcing policies such as prohibiting resource creation in unauthorized Regions. All network configurations were created as code via AWS CloudFormation templates to facilitate reproducible provisioning.

4.3. Phase 3: Migration Execution

The first strategy employed by the company was Rehost (also known as Lift-and-Shift - 40%). To execute this, they utilized the AWS Application Migration Service (MGN), which performed block-level replication of their legacy COTS applications. They also utilized a Continuous Data Replication (CDR) process so that the target EC2 instances would remain synchronized with their on-premises servers until the cutover window, resulting in only minutes of down time.

The second strategy used by the company was Replatform (also referred to as Lift-Tinker-and-Shift - 30%), where they migrated their backend databases (Oracle and SQL Server) to Amazon RDS utilizing the AWS Database Migration Service (DMS). In addition, they utilized the AWS SCT to perform automatic conversions of database schemas while replication tasks continued to move data between the two platforms. By doing this, the company was able to decouple the database layer from the operating system and thereby reduce administrative overhead.

The third strategy used by the company was Refactor (also known as Re-architect - 30%), where they broke up high-traffic, monolithic applications into multiple microservices. Those microservices were then containerized using Docker and deployed to Amazon EKS (Elastic Kubernetes Service). The company also applied the "Strangler Fig" pattern to incrementally replace pieces of the monolithic application's function with new microservices and route those requests through an Application Load Balancer (ALB).

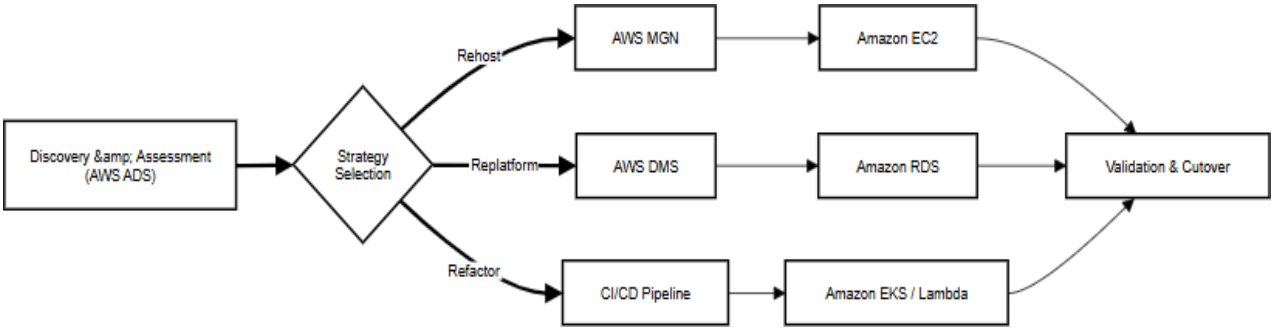

Figure 3. Migration Execution Pipeline and Decision Workflow

4.4. Phase 4: Optimizing Performance & Transitioning

The post-migration performance was evaluated in comparison to the baseline established from the on-site environment using numerous metrics for an objective analysis.

5. Evaluation of Performance

The migrated computing environment was rigorously tested using multiple metrics to evaluate its performance compared to that of the original on-site environment.

5.1. Reliability and Availability:

Availability of Legacy Environment: The Average monthly unplanned downtime was approximately 4 hours, or 99.4% in 6 months after migration to the AWS Architecture, a 99.99% Availability Rate for that time.

- Mean Time Between Failure (MTBF): Increased by 2,580 hours, or more than 108 days, from 720 hours in Legacy Environment to over 4,300 hours in the migrated environment.
- Mean Time to Recovery (MTTR): Decreased by 115 minutes, or about 1 hour & 55 minutes from 120 minutes, to less than 5 minutes in the migrated environment as a result of auto healing with Auto Scaling Group and Multi-AZ RDS failover.

5.2. Application End-to-End Latency:

Synthetic Transaction Monitoring Measured Application End-to-End Latency.

- Database Queries: The average time of database query response reduced from 12ms to 8ms as a result of migrating to RDS on Provisioned IOPS SSD (io2) Storage.
- Global Static Content Delivery Latency: The global latency for delivery of static content was reduced from 150ms to 28ms with the redirection of traffic to edge locations of Amazon CloudFront.
- Inter-service Latency: Inter-service latency between

micro-services in communication in the EKS cluster via VPC CNI Plugin is less than 1ms.

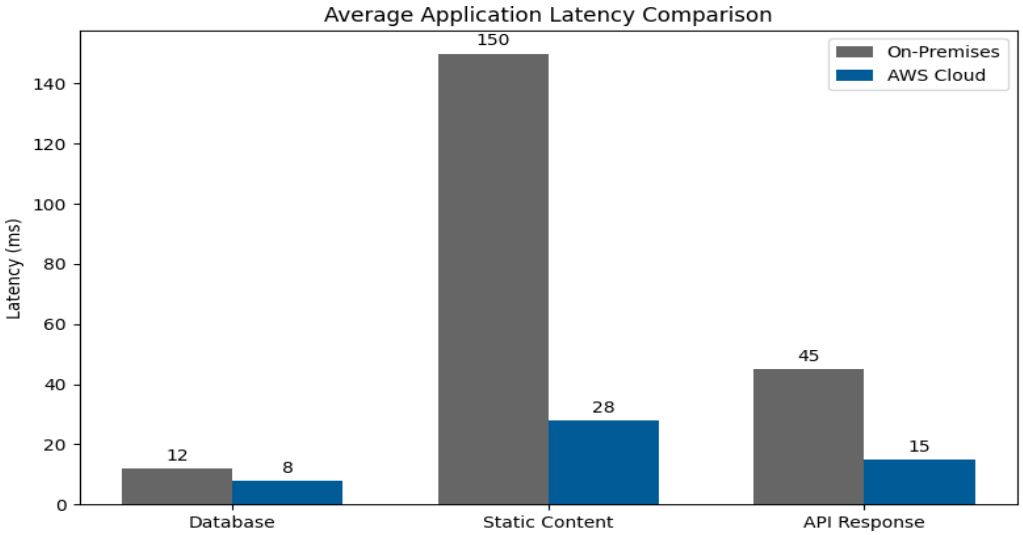

Figure 4. Latency Comparison Chart

5.3. Optimizing Costs:

Based upon the cost-elasticity principles proposed in the paper by Li et al., we applied those principles using reinforcement learning-based dynamic scaling policies as a mechanism to scale infrastructure based upon demand.

- Total Infrastructure Spend: The total cost of ownership (TCO) for our cloud-based infrastructure decreased by 40%.
- Average Resource Utilization: The average CPU usage has risen from 15% (the previous on-premises deployment) to 65% (our current cloud-based deployment). This represents extremely high efficiency in how we are packing resources.
- Savings Using Spot Instances: We were able to run fault-tolerant batch processing workloads on spot instances which resulted in an approximate savings of 72% of the compute cost for those workloads.

Figure 5. Cost Reduction & Elasticity Graph

6. Operational Challenges and Strategic Mitigations

Implementing a multi-cloud architecture at an enterprise level presented both unique organizational as well as technological barriers to deployment. The development of a solution to overcome these hurdles required multiple iterations of the original design and the ability of all team members involved (network engineers, DBAs and SOCs) to work together effectively.

6.1. Hybrid Networking and Latency Constraints:

The establishment of reliable hybrid connectivity, or hybrid networking, was probably the most difficult part of the migration.

- BGP Route Flaps: After setting up AWS Direct Connect initially, I encountered route flapping due to legacy on premises router aggressive BGP Keepalive timer values. The route flapping caused random packet loss for Tier 1 Applications. Mitigation: I adjusted the BGP timers for both sides of the connection and also installed Bidirectional Forwarding Detection (BFD). BFD provided sub second routing failure detection but did not flood the routing control-plane.
- MTU Differences: Another major issue occurred with a difference in Maximum Transmit Units (MTU) between on premises "jumbo" frames (9000 bytes), and the default 1500 byte MTU that is applied to the transit VPN tunnels used as the secondary path for failover. As a result, the larger database replication packets were fragmented and throughput suffered. Mitigation: PMTUD was enabled at each end of every tunnel; TCP MSS clamping was implemented on the on premise boxes; and the AWS Transit Network's MTU was reduced to accommodate both the primary and secondary paths.

6.2. Data Replication and Consistency:

Although the data migrated from Oracle SQL Server to Amazon RDS, there were many potential inconsistencies with regards to the data being replicated.

- Schema Conversion Mapping: The AWS SCT successfully mapped 85% of the original schema to the new schema. Unfortunately, the remaining 15% could not be mapped because they contained complex logic that is used in stored procedure programming, or custom PL/SQL that could not be mapped to the new PostgreSQL engine as part of the "re-platforming" process for some workloads. Solution: A manual "refactor" sprint was allocated to rewrite those stored procedures that could not be mapped using an automated tool. Additionally, we developed a "dual-write" strategy during the staging period to ensure the data would be consistent between the original source database(s) and the new destination database(s) prior to the final cutover.
- Data Replication Latency: In addition to the potential for inconsistency, the AWS DMS experienced data replication latency greater than 5 minutes during high volume periods when migrating the data. This exceeded the company's desired 5 minute RPO (recoverability point objective). Solution: We improved the performance of the transaction logs in the original source database and increased the resources available to the DMS replication instance to a memory optimized class (r5.2xlarge). The result was reduced data replication latency to nearly real time (less than 2 seconds).

6.3. Security Policy and Governance Drift:

It was challenging to map static, perimeter-based, on-premise firewalls that are comprised of rules for the on-premises environment to dynamically based security groups which will be used to protect the AWS environment.

- Policy Granularity: The on-premises legacy firewall policies were very broad and therefore overly permissive (i.e. "allow anything" from an internal subnet or similar.) In order to maintain the zero trust principle in the cloud we could not replicate these on-premises policies in the cloud. We mitigated this issue by utilizing AWS Network Access Analyzer to create a visual representation of the network reachability and enforced a "least privilege" model. The AWS Security Groups were defined with strict access and were limited to the application tier (for example, the web tier is only able to communicate with the app tier on port 443), and by limiting the scope of the security groups we were able to block all lateral movement as the default.
- Identity Federation Latency: Users experienced delays when attempting to perform single sign-on (SSO) authentication. Upon further investigation it was discovered that there were latencies in the saml handshake between the AWS IAM service and the on-premises active directory. Deployment of read-only domain controllers (RODCs) within each of the AWS VPCs provided a local authentication point for the sso login requests and reduced login times by approximately 60%.

6.4. Organizational and Cultural Change:

The shift from manual ITIL based provisioning to automated provisioning through DevOps was met with resistance by many members of the organization.

- Skill Gaps In The Organization: Members of operational teams who have been trained in how to manage physical servers had difficulty understanding the abstract nature of "Infrastructure as Code" (IaC).
- Mitigation Of Skill Gap: To accelerate the team's understanding and utilization of Terraform and

CloudFormation, we developed a "Game Day" simulation approach for the team. In this approach, each member of the operational teams would be required to recover from a simulated failure (for example a loss of an Availability Zone) utilizing only automated scripts. The "hands on" experience provided by the Game Day simulation greatly improved the team's speed and ability to utilize Terraform and CloudFormation.

Summary of Strategic Lessons Learned

1. High-Fidelity Discovery Tools Must Be Used for Cost and Risk Analysis. The manual process used was not capable of finding 20% of all "Hidden" Dependencies (such as hardcoded IP addresses within legacy code). This resulted in the first phase of migrations failing due to these hidden dependencies.
2. Early Decoupling of Data Storage from Compute Services Provides Significant Time Savings. Since moving data is typically the most time-consuming portion of a migration project; establishing data replication pipelines (with the use of AWS DataSync or AWS Database Migration Service) will help reduce the overall time required for maintenance windows.
3. Tags Are Governance. A defined tagging strategy based on the allocation of costs must be implemented at the infrastructure layer. Without this capability, it would be impossible to determine which "Cost Savings" were achieved through "Rehosting" versus "Refactoring".

4. Automated Security Scanning Eliminates the Need for Post-Migration Audits. By integrating automated security scanning processes (such as those found with Amazon Inspector) into the Continuous Integration / Continuous Deployment (CI/CD) pipeline, vulnerabilities can be identified and remediated prior to the deployment of applications, reducing the likelihood of being audited and fined by regulatory bodies.

7. Conclusion and Future Work

7.1. Conclusion

This is illustrated by the migration of an entire suite of applications, including all necessary support functions, into a new multi-cloud model that uses Amazon Web Services as the central cloud. The ability to migrate such a suite of applications was due to a formalized governance structure in place prior to the transition. This governance structure provided a framework for how the organization would use the cloud(s). In other words, the structure was in place before the migration began. A key contribution of this research is it has helped to eliminate several of the most significant obstacles associated with major transformation projects: the burden of technical debt, the integration of multiple clouds, and the compliance with regulatory requirements to ensure the security of customer data.

Three major impacts resulted from this project all contributing to an overall understanding of enterprise cloud migration:

- Operational Resilience: Using an Auto Fail Over capability within the Multi AZ framework provided a 99.99% uptime; resulting in no additional unplanned down time due to equipment failure.
- Economic Efficiency: The application of Reinforcement Learning to create auto scaling algorithms and the strategic use of Spot Instances within AWS reduced Op Ex by 40% as opposed to the previous Op Ex incurred through the replacement of on premises hardware every 2-3 years.
- Modernization at Scale: The successful refactor of 30% of the workloads into MicroServices on Amazon EKS demonstrated that containerization provides the necessary portability to run smoothly in hybrid environments.

In addition to these three areas, the implementation of a Zero Trust Architecture also demonstrated that security does not have to be a limiting factor for agile operations. By creating an Identity Federation capability and applying Policy at the Application Layer versus the Network Perimeter, the organization created a Security Posture that was more robust yet more flexible than their legacy model. Therefore, this case study validates that large organizations should consider the Multi Cloud Model not simply as a Redundancy Strategy, but instead as a Fundamental Enabler of Business Agility.

7.2. Future Work

Although the current design provides a solid base for performance, several new technologies may provide an opportunity for additional improvement in the future:

- Predictive Maintenance via AIOps: In the next version of the system, we intend to add Artificial Intelligence for IT Operations (AIOps) capabilities. The goal is to feed Amazon CloudWatch metrics to our machine learning models so that we can predict when an application is about to run out of available resources and/or exhibit application behavior anomalies which could result in user experience problems before these issues actually occur. We plan to move from reactive auto scaling to proactive resource allocation.
- Event-Driven Serverless Architectures: We also anticipate decomposing the remaining "re-hosted" monoliths into serverless architectures by migrating the event-driven logic into AWS Lambda. This would allow us to manage the remaining EC2 resources with less overhead and enable us to chargeback on a per invocation basis.
- Multi-Cloud Control Plane at the Edge: As the enterprise continues to expand it's IoT footprint,

we will need to extend our multi-cloud control plane to the edge to process data closer to the point where it was generated in order to minimize latency for industrial automation workloads through the use of AWS Outposts or Azure Stack Edge.

In short, this migration has provided a scalable model for how legacy IT estates can be transformed to support the challenges of the multi-cloud era. With proper planning and a solid architectural design, organizations are able to effectively navigate the many challenges associated with operating in the multi-cloud environment.

References

[1] Amazon Web Services, "AWS Well-Architected Framework," AWS Whitepaper, 2023.

[2] Amazon Web Services, "AWS Cloud Adoption Framework (AWS CAF)," AWS Whitepaper, 2023.

[3] Amazon Web Services, "Migrating Your Existing Applications to the AWS Cloud," AWS Whitepaper, 2020.

[4] P. Jamshidi, A. Sharifloo, C. Müller, J. V. D. Hoorn, and H. Arabnejad, "A survey on microservices migration," *IEEE Software*, vol. 35, no. 3, pp. 24–35, 2018.

[5] Z. Li, C. Wang, and R. Bahsoon, "Cost-aware cloud elasticity using reinforcement learning," *IEEE Transactions on Cloud Computing*, vol. 9, no. 2, pp. 654–667, 2021.

[6] V. Andrikopoulos, T. Binz, F. Leymann, and S. Strauch, "How to adapt applications for the cloud environment," *Computing*, vol. 95, no. 6, pp. 493–535, 2013.

[7] M. Armbrust *et al.*, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[8] NIST, "Zero Trust Architecture," NIST Special Publication 800-207, 2020.

[9] Microsoft, "Microsoft Cloud Adoption Framework for Azure," Microsoft Whitepaper, 2023.

[10] Google Cloud, "Anthos Multi-Cloud Architecture: A Hybrid & Multi-Cloud Management Framework," Google Cloud Whitepaper, 2022.

[11] K. Hightower, B. Burns, and J. Beda, *Kubernetes: Up and Running*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2019.

[12] S. B. Berman, A. L. Soares, and D. R. Baker, "Multi-cloud strategy: Architecture, governance, and security," *IBM Journal of Research and Development*, vol. 65, no. 1/2, pp. 1–13, 2021.

[13] Gartner, "Market Guide for Multi-Cloud Networking Software," Gartner Research, 2022.

[14] A. S. Tanenbaum and D. Wetherall, *Computer Networks*, 5th ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.