Solution Architecture Design Proposal: Migrating a legacy on-premise system to AWS

*Author: Tatenda Manyepa*



Table of Contents

[Executive Summary 3](#_Toc192885388)

[Business Context and Objectives 3](#_Toc192885389)

[Business Context 3](#_Toc192885390)

[Key Challenges 3](#_Toc192885391)

[Objectives of the AWS Migration 4](#_Toc192885392)

[Solution Overview 5](#_Toc192885393)

[Current System Architecture Diagram 5](#_Toc192885394)

[Proposed Cloud Architecture Diagram 5](#_Toc192885395)

[Flow Desription 6](#_Toc192885396)

[Technical Implementation 7](#_Toc192885397)

[Migration Approach 7](#_Toc192885398)

[Security & Compliance Strategy 9](#_Toc192885399)

[Cost Optimisation Strategy 10](#_Toc192885400)

[Expected Business Impact 11](#_Toc192885401)

[Implementation Roadmap 12](#_Toc192885402)

[Conclusion 12](#_Toc192885403)

# Executive Summary

This Solution Architecture Design Proposal outlines the migration of an existing on-premises e-commerce storage system to **Amazon Web Services (AWS)** to improve **scalability, cost efficiency, operational reliability, and security**. The current infrastructure consists of **application servers, automation control servers , and database servers** that face challenges such as **high operational costs, manual maintenance, performance bottlenecks, and limited scalability**.

By leveraging AWS cloud services, this proposal aims to establish a **highly available, scalable, and automated** architecture that ensures **optimal system performance, reduced downtime, and improved security** while lowering infrastructure costs.

# Business Context and Objectives

##

## Business Context

The company operates an **automated e-commerce storage management system** integrated with an automated task coordination and execution system for stock movement and order fulfilment. The existing on-premises servers support:

* **Order processing and stock management** (Application Servers)
* **Automated task coordination and execution** (Automation Control Servers)
* **Inventory and task tracking** (Database Servers)
* **Testing and development environments** (Test Servers)

## Key Challenges

The current infrastructure presents the following issues:

1. **Limited Scalability** – On-premises hardware constraints limit performance and expansion.
2. **High Maintenance Effort** – Manual patching, monitoring, and upgrades increase operational burden.
3. **Performance Bottlenecks** – Communication latency between servers slows business operations.
4. **Lack of Automation** – Routine maintenance (e.g., backups, health checks) is manual and inefficient.
5. **Security and Compliance Risks** – On-premises security is harder to enforce at scale.

## Objectives of the AWS Migration

This proposal aims to:

* **Enhance Scalability** – Migrate infrastructure to AWS to support workload fluctuations.
* **Optimise Costs** – Reduce hardware expenses and operational overhead.
* **Improve Performance** – Lower latency and optimise query processing times.
* **Automate Infrastructure Management** – Implement **AWS Systems Manager** and **CloudWatch** for monitoring, patching, and health checks.
* **Strengthen Security and Compliance** – Leverage **AWS IAM, KMS encryption, GuardDuty, and WAF**.
* **Ensure High Availability** – Implement failover mechanisms using **AWS Multi-AZ and Auto Scaling**.

# Solution Overview

## Current System Architecture Diagram



*This architecture was created for educational and design demonstration purposes, based on general industry patterns.*

## Proposed Cloud Architecture Diagram

The diagram illustrates the **proposed AWS cloud architecture** for migrating the **automated e-commerce storage management system** from an on-premises setup to **AWS**. It highlights key components categorised into **compute, networking, database, monitoring, and security layers**.



## Flow Description

This architecture supports the migration of an automated e-commerce system from on-premises infrastructure to AWS. The system is designed for scalability, high availability, and operational automation.

**Traffic Flow:**

External users access the system via an AWS Application Load Balancer (ALB) located in the public subnet. The ALB distributes traffic to EC2-based Application Servers running in Auto Scaling Groups within private subnets across two Availability Zones. These servers handle front-end logic, business workflows, and interact with Amazon RDS and ElastiCache for data and caching operations.

**Automation Layer:**

Automation Control Servers, deployed on separate EC2 instances in their own Auto Scaling Groups, manage the storage control logic. These are also placed in private subnets for enhanced security and are optimised for internal task coordination rather than user-facing workloads.

**Monitoring & Security:**

AWS CloudWatch and Systems Manager oversee monitoring, patching, and health checks. AWS Config tracks configuration changes to ensure compliance. GuardDuty and Security Hub continuously monitor for threats and anomalies. IAM, KMS, WAF, Security Groups, and NACLs enforce layered security.

**Backup & Recovery:**

AWS Backup handles scheduled backups of EC2 and RDS data. Backups are archived to Amazon S3 Glacier for long-term storage.

**Migration & Integration:**

Database migration is handled by AWS DMS, which transfers data from the legacy on-premises database to Amazon RDS with minimal downtime. The architecture supports secure integration with the Customer's Host System through clearly defined VPC connectivity.

# Technical Implementation

## Migration Approach

The migration will be performed in stages using a **lift-and-shift strategy** followed by **optimisation and refactoring**.

**Phase 1: Assessment and Planning**

* Evaluate current on-premises workloads and dependencies.
* Define security and compliance requirements.
* Plan network and access control configurations.

**Phase 2: Infrastructure Setup on AWS**

* **Define Terraform configurations** for AWS resources, including **VPC, subnets, security groups, and EC2 instances.**
* **Deploy Amazon EC2 instances** for application and automation control servers using Terraform.
* **Provision Amazon RDS with Multi-AZ deployment** via Terraform.
* **Set up Amazon ElastiCache** for database performance optimisation.
* **Implement IAM roles, policies, and security best practices** as code.
* **Enable AWS CloudWatch, Systems Manager, and GuardDuty** for monitoring and automation.
* **Use Terraform state management and version control** for deployment consistency.

**Phase 3: Data and Application Migration**

* Migrate databases using AWS DMS (Database Migration Service).
* Deploy application workloads and test connectivity.
* Optimise network and application configurations.

**Phase 4: Testing & Optimisation**

* Conduct performance testing and fine-tune EC2 and RDS instances.
* Simulate failover scenarios to validate high availability.
* Enable auto-scaling and adjust configurations as needed.

**Phase 5: Go-Live & Monitoring**

* Gradual traffic shift to AWS environment.
* Enable full monitoring and alerting mechanisms.
* Conduct post-migration assessment and optimisation.

# Security & Compliance Strategy

| The **Security & Compliance Strategy** ensures the **automated e-commerce storage management system** is protected against threats, meets compliance standards, and enforces best security practices. By implementing AWS-native security services, this architecture safeguards **data integrity, application security, and access control** while enabling continuous monitoring and threat detection.

|  |  |
| --- | --- |
| Security Feature | Implementation |
| IAM Access Control | Role-based policies that enforce **least privilege policies** to limit user and service access to necessary resources only. |
| Data Encryption | Ensures data security **at rest and in transit** using AWS KMS (Key Management Service). |
| API Protection | AWS WAF (Web Application Firewall) mitigates risks such as **SQL injection, cross-site scripting (XSS), and DDoS attacks**. |
| Threat Detection | AWS GuardDuty and AWS Security Hub provide continuous **real-time monitoring and anomaly detection** to identify security threats. |
| Compliance | AWS Config ensures **continuous compliance tracking** by auditing changes and maintaining security posture. |

By integrating these security measures, this architecture aligns **with AWS Well-Architected Framework** security best practices, ensuring resilience, compliance, and proactive threat defence. |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Cost Optimisation Strategy

| To ensure **cost efficiency** in the **automated e-commerce storage management system**, a combination of **Auto Scaling, Reserved Instances, AWS Savings Plans, and optimised storage solutions** can be used. These strategies help **reduce unnecessary expenses, maximise resource utilisation, and enhance performance** while maintaining **high availability and reliability**.

|  |  |
| --- | --- |
| Optimisation Approach | Expected Savings |
| Auto Scaling | Dynamically adjusts compute resources based on demand, **preventing over-provisioning** and reducing operational costs. |
| Reserved Instances | Offers savings of **up to 72%** on long-term EC2 and RDS workloads compared to on-demand pricing. |
| Amazon S3 Lifecycle Policies | Automatically **archives infrequently accessed data**, lowering storage costs while maintaining accessibility. |
| AWS Savings Plans | Provides a **flexible, cost-effective pricing model** for consistent cloud workloads, reducing long-term expenditure. |
| Amazon RDS Read Replicas | Offloads **read-heavy database queries**, **optimising performance** and **minimising primary database costs**. |

 |  |
| By implementing these **cost-saving measures**, the AWS migration ensures a **financially optimised cloud environment** without compromising **performance, security, or scalability.** |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Expected Business Impact

Migrating the **automated e-commerce storage management system** to AWS brings **significant business benefits**, improving **scalability, efficiency, reliability, and cost-effectiveness**. This transformation enables **automated operations, optimised performance, and reduced infrastructure costs**.

|  |  |
| --- | --- |
| Impact Area | Benefit  |
| Scalability  | Auto Scaling dynamically adjusts resources **based on workload demand**, ensuring optimal performance during peak operations. |
| Operational Efficiency  | Automates key tasks like **patching, monitoring, and security enforcement**, minimising manual effort and human error. |
| System Reliability | **Multi-AZ deployments and failover mechanisms** ensure **continuous system availability** and minimal downtime. |
| Cost Reduction | Migrating to AWS eliminates **expensive on-premises infrastructure costs**, reducing maintenance overhead. |
| Performance Improvement | **Database query optimisations** and caching mechanisms **reduce latency**, enhancing response times and business operations. |

This **modernised cloud-based infrastructure** empowers the company to scale effortlessly, optimise costs, and improve overall automation efficiency.

# Implementation Roadmap

|

|  |  |  |
| --- | --- | --- |
| Phase | Timeframe | Key Activities |
| Assessment and Planning | Week 1-2 | Define migration strategy, security requirements |
| Infrastructure Setup | Week 3-4 | Deploy AWS resources via Terraform |
| Data & Application Migration | Week 5-6 | Migrate workloads, test connectivity |
| Optimisation and Testing | Week 7-8 | Fine-tune configurations, simulate failover scenarios |
| Deployment & Monitoring | Week 9-10 | Gradual traffic cutover, enable full monitoring |

 |

# Conclusion

Migrating the **e-commerce storage management system** to AWS will **enhance scalability, reduce operational overhead, and improve security and performance**. This solution will transform infrastructure management from a **manual, on-premises setup** to an **automated, cloud-based architecture**, ensuring the company can support future growth efficiently and cost-effectively.