



Enterprise Application Migration & Modernisation

Amazon Web Services Playbook

Table of Contents

- 1 Introduction
- 2 Strategic Assessment and Migration & Modernisation Strategy
- 3 Application Preparation and Migration & Modernisation
- 4 Continuous Integration and Continuous Deployment (CI/CD)
Pipeline and Deployment
- 5 Multi-Environment Configuration and Certificates
- 6 Resilience, Scale and Disaster Recovery
- 7 Security, Access and Governance

Introduction

An authoritative transition blueprint designed to bridge the gap between technical execution and business strategy, ensuring every local asset adopts enterprise-grade security and automated operational patterns.



Why a Migration Framework



Standardised Guardrails

Eradicates configuration drift, guarantees container optimization, and ensures all migrated assets align directly with the target stack.



Enterprise Scalability

Replaces isolated, low-availability local environments with dynamically scalable container pods backed by Multi-AZ persistent relational engines.



Early Risk Integration

Shifts security validation left, embedding automated checks, secrets masking, and mandatory compliance validation into early stages of engineering.



Purpose & Strategic Value Pillars



Standardised Delivery

Translates cloud engineering mechanics directly into business goals, keeping metrics, tracking logs, and audit pathways transparent across all tiers.



Rigid Compliance

Hardens the host border from Day 1, ensuring IAM authentications, certificate tracking, and secret synchronization pipelines are natively unified.



Operational Parity

Ensures immediate code execution parity across all development, testing, and staging ranges by standardizing multi-stage build pipelines.



Strategic Assessment, Modernisation and Migration Strategy

Aligning client business objectives with technical feasibility to ensure the transition path supports growth, speed -to-market, and institutional security

Strategic Assessment, Modernisation and Migration Strategy



What Are We Doing?

Performing deep-dive discovery sessions to define project requirements and assessing the legacy footprint to determine the most cost-effective and optimal transition model.



How It's Done

Infrastructure as Code: Using Terraform templates to build identical environments reliably every time

Version Control: Leveraging Git as the "Time Machine" for all system and code changes to ensure absolute version control

Continuous Assembly & Auditability: Testing every developer contribution for security and bugs before it reaches users, ensuring every change is recorded, reviewed, and reversible for banking audits.



Roles

DevOps Engineer

Cloud Engineer



Key Outputs

- Terraform IaC templates
- GitOps automated assembly line
- Reversible audit trails



Application Preparation, Modernisation and Migration

Deconstructing application code structures and translating data persistence engines to support stateless, horizontal scaling within containerised cluster resources.

Application and Pipeline Preparation



What Are We Doing?

Decoupling existing application layers from underlying OS bindings and packaging code into high-security, minimal-footprint container images tailored for orchestration.



How It's Done

Multi-Stage Build Pattern: Separating build-stage dependencies from final runtime containers using minimised ECR-mirrored base slim images.

Environment Sanitisation: Declaring environment-specific runtime configurations strictly outside of built container binaries.

Orchestrated Local Parity: Utilizing Docker Compose during early development phases to replicate multi-service production network.



Tools Needed



Docker



Visual Studio



Github



Key Outputs

- Optimised Multi-Stage Dockerfile
- Local compose configurations
- Minimized runtime container

Modernisation and Migration



What Are We Doing?

Migrating localised, file-based data stores (such as SQLite) to managed AWS RDS instances configured for high availability.



How It's Done

Zero-Static-Password Mandate: Decommissioning static database connection strings in favour of short-lived AWS IAM database tokens generated on runtime.

High-Availability Configurations: Deploying database engines inside Multi-AZ configurations across physical availability boundaries.

Automated Protection: Enforcing institutional 30-day automated backup retainment parameters and point-in-time recovery points.



Tools Needed



PostgreSQL



IAM



Key Outputs

- Data schema refactoring scripts
- AWS IAM DB connection policies
- Managed RDS production instances

CI/CD Pipeline and Deployment

Standardising the continuous verification of codebase assemblies and mapping automated delivery routes using verified pipeline configurations.

CI/CD Pipeline and Deployment



What Are We Doing?

Implementing a modern delivery pipeline with GitOps integration, ensuring that the cloud environment automatically syncs to match the approved state in the Git repository .



How It's Done

Constraint & Persona Analysis: Analysing legacy infrastructure constraints while identifying primary user personas and load profiles.

Migration Path Selection: Choosing between Lift & Shift, Re- platform , or Refactor based on ROI and scalability.



Roles

Business Domain Owners

Technology Analysts



Key Outputs

- Terraform IaC templates
- GitOps automated assembly line
- Reversible audit trails

CI/CD Pipeline and Deployment



What Are We Doing?

Automating structural validation, vulnerability scanning, and secure build compilation workflows directly from corporate code repository platforms.



How It's Done

Continuous Quality Verification: Executing unit test frameworks and static security analysis upon every codebase change.
Secure Compilation: Binding built container artifacts with dynamic, cryptographically verifiable Git SHA tracking variables.



Roles

DevOps Engineer

Cloud Engineer



Key Outputs

- GitHub Action pipeline definitions
- Automated security scan reports
- Version-controlled ECR artifacts

CI/CD Pipeline and Deployment



What Are We Doing?

Enforcing standardised delivery runbooks to deploy finalised container architectures into Kubernetes hosting runtimes securely.



How It's Done

Policy-Driven Execution: Targeting Kubernetes clusters using CD tools (Octopus or Jenkins) worker pools assigned with explicit corporate execution contexts.

Pre-Deployment Database Migrations: Triggering automated hooks inside container environments to apply data schemas before routing live transactions.

API Normalization: Integrating Ingress controllers and proxy routing rules (like path-stripping layers) to align incoming traffic with containers.



Roles

DevOps Engineer

Cloud Engineer



Key Outputs

- Project release runbooks
- Raw Kubernetes manifest setups
- Active application target nodes



Multi-Environment Configuration

Separating developer, testing, and production stages through network limits and securing endpoints..

Multi-Environment Configuration

METRIC / RESOURCE	DEVELOPMENT (DEV)	SYSTEM INTEGRATION TESTING (SIT)	USER ACCEPTANCE TESTING (UAT)
Cluster Allocation	Non-Prod Target A	Non-Prod Target B	Non-Prod Target C
Database Endpoint	Dev-Scoped Managed DB	SIT-Scoped Managed DB	UAT-Scoped Managed DB
Network Control Range	Internal Subnet Boundary	Dedicated Dev/QA Bridged Zone	Dedicated CIDR-restricted Range

Note: Firewall settings and network routing boundaries must be formally requested and authorized independently for each staging environment to prevent cross-tier transaction bleeding.



Resilience, Scale and Disaster Recovery

Building systems capable of surviving hardware failures with zero user impact and defining recovery targets based on criticality.

Disaster Recovery and Resilience



What Are We Doing?

Ensuring the application dynamically grows or shrinks based on transaction volume while maintaining acceptable institutional Recovery Time Objective (RTO) target.



How It's Done

Fault Tolerance: Deploying across Multi-AZ (Availability Zones) so applications continue running automatically if an AWS data center fails.

Scalability: Utilizing Horizontal Pod Autoscaling (HPA) to add "clones" of the app during peak hours and delete them at night to optimize cost.

Tiered Disaster Recovery: Defining Recovery Time Objectives (RTO) and Recovery Point Objectives (RPO) using strategies like Backup & Restore, Pilot Light, or Multi-Site



Roles

Cloud Engineer

Infrastructure Engineers



Key Outputs

- Project release runbooks
- Raw Kubernetes manifest setups
- Active application target nodes



Security, Access and Governance

Enforcing zero-trust configuration architectures across active pipelines, federating credentials via active directories, and satisfying formal change ticketing gates.

Security, Access and Governance



What Are We Doing?

Establishing an end-to-end defence-in-depth model that prevents plaintext database credentials, API keys, and administrative configurations from exposure.



How It's Done

Development Boundary Rules: Enforcing isolated configurations inside developers' environments and blacklisting them globally via gitignore patterns.

Build Automation Isolation: Leveraging GitHub Secrets validation variables to compile dynamic, temporary deployment tokens.

Unified Cloud Storage: Anchoring critical production keys directly inside AWS Secrets Manager, mapped to CD tools dynamically at container runtime.

Environment-scoped sensitive variables to prevent exposed values.



Tools Needed



PostgreSQL



IAM



Key Outputs

- Data schema refactoring scripts
- AWS IAM DB connection policies
- Managed RDS production instances



THANK YOU

Website: www.cooistudios.com

Email: hello@cooistudios.com

138 Jan Smuts Avenue

Parkwood

Johannesburg, 2193

South Africa