



# Modernization is a behavioral shift



A practical guide for finance leaders  
looking to leverage the cloud



# Content

00

Introduction

01

Modernization Readiness Assessment

02

Modernization Planning

03

Implementation: Building the Modernized Environment

04

Improvement: Driving Team Enablement & New Workflows

05

Support & Continuous Evolution

06

Rambunct Case Study



## Introduction

# Too often, cloud migration is sold as a clean, predictable journey.

But once the work starts, the gaps appear: systems that resist change, data that refuses to cooperate, and teams stretched between keeping the lights on and moving everything forward.

Teams suddenly face old systems that don't want to move and timelines that tighten instead of stretch .

And that's usually the moment when the whole effort shifts tone. Every solved problem can uncover another dependency, every step forward exposes something the plan didn't account for. And slowly, the real challenge becomes keeping the business stable while navigating a landscape that keeps changing under your feet.





Financial organizations depend on predictable systems, clean data, and strong regulatory alignment. When migration begins, these are precisely the areas where fragility becomes most visible.

### Across Accenture's surveys on financial services IT leaders:

**70%**

report that legacy systems are the main barrier to real-time reporting and regulatory agility.

**Over 60%**

cite data fragmentation as a source of operational risk.

**Most modernization**

come from hidden dependencies between financial applications.

# What this means for the finance sector

Cloud migration pressures existing systems, and, more often than not, can expose how dependent everything is on data that moves cleanly and on time.

**Regulatory expectations** keep tightening and operational risk rises when teams are forced to improvise. What this ultimately reveals is a sector that needs stronger foundations to keep pace with change, not just new infrastructure, but stability around it.



# A note from Neo

Over the past three years at Rambunct, we've guided more than **60 organizations** through their AWS journeys—from initial migrations to full-scale cloud transformations.

*What strikes me most is how similar the challenges are across different industries, yet how unique each solution needs to be. This guide distills the patterns we've seen work consistently, and more importantly, the mistakes we've learned to avoid.*

*The AWS landscape changes rapidly, but the fundamentals of good architecture remain constant: understand your dependencies, instrument everything, automate ruthlessly, and never assume your mental model matches production reality. Let's dive in.*

Neo Lee  
Founder & Cloud Architecture Lead, Rambunct



# A practical path to modernization

From first assessment to steady-state evolution.

1

Modernization  
Readiness  
Assessment

2

Modernization  
Planning

3

Implementation:  
Building the  
Modernized  
Environment

4

Improvement:  
Driving Team  
Enablement & New  
Workflows

5

Support & Continuous  
Evolution

1

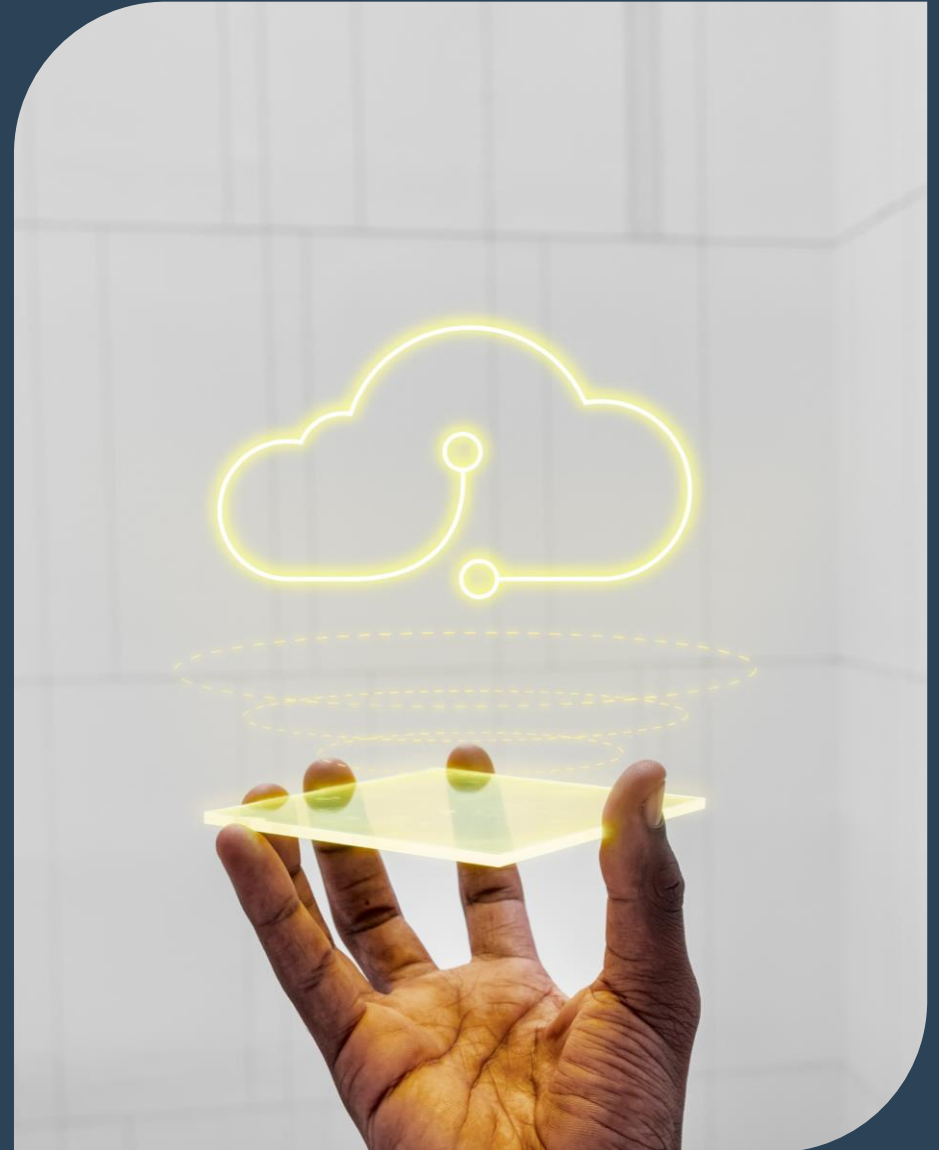
# Modernization Readiness Assessment



# A full discovery of your technical, operational, and **behavioral reality**.

Most modernization failures come from misjudging the starting point.

A **Readiness Assessment** clarifies what's actually running, how it behaves, and where the real constraints sit, before migration decisions lock in.



# A Readiness Assessment creates a realistic baseline by analyzing four areas:



## 1. Technical landscape

- Legacy components that survived previous migrations.
- Workloads still operating in near-lift-and-shift mode.
- Performance bottlenecks tied to architecture rather than hardware.
- Hard-coded integrations that break once moved.



## 2. Data flows & integration fragilities

Financial systems are interconnected by design, core banking, risk engines, reporting modules, batch pipelines.

Across case studies, migrations break not because the workload fails, but because the data feeding it doesn't arrive cleanly or on time.



## 3. Behavioral patterns

Teams often cling to old workflows even when systems have evolved, this is the silent killer of modernization.



## 4. Operational capability & skills

Cloud-native skills, adoption appetite, support structures, and maturity gaps.

# Clarifying what is stable, what is fragile, and what will break the moment modernization begins

This baseline prevents avoidable surprises later in the program.

Once the real starting point is visible, priorities become clearer: which workloads can move first, which integrations need reinforcement, and where operational capabilities require support.

It doesn't delay progress; it prevents rework and keeps the path clean.



The assessment highlights early wins, reveals hidden blockers, and gives the modernization effort a grounded foundation.

With this clarity in place, the next step shifts toward design. The focus moves from “What do we have?” to “How do we sequence modernization in a way that delivers value early and reduces risk?” This is where the roadmap begins.

# Outcome

A clear, honest understanding of:

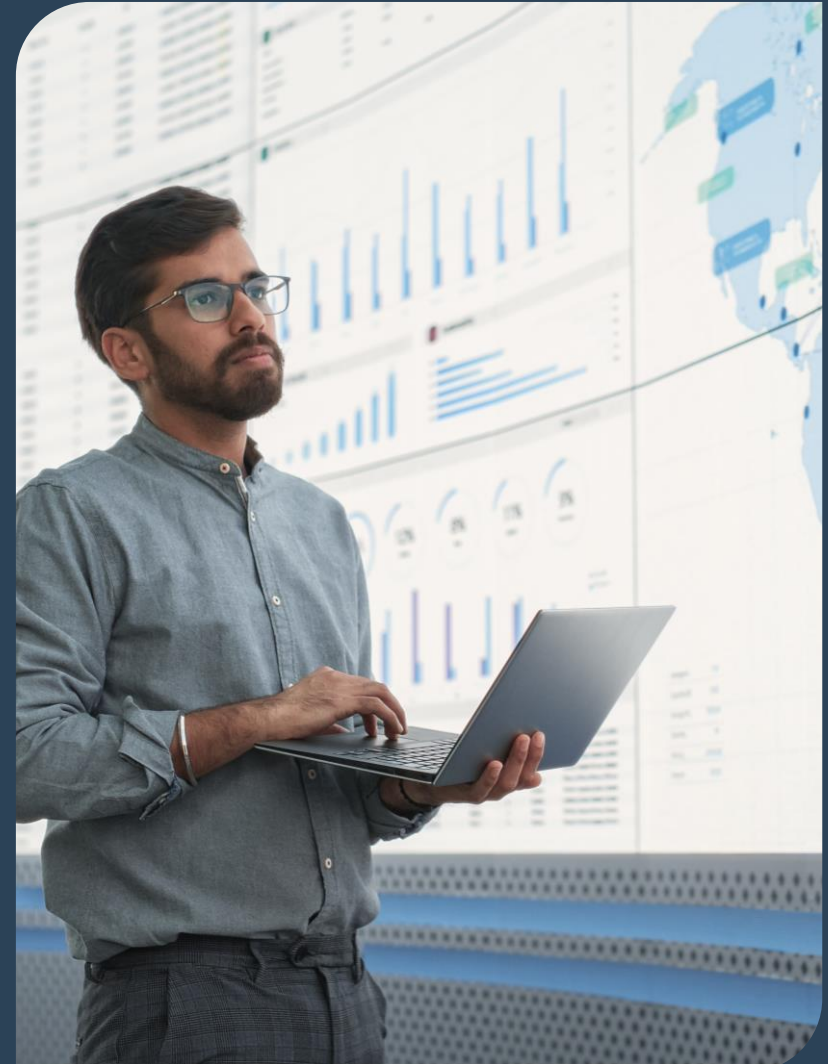
**What you have**

**vs.**

**What needs to change**

**vs.**

**What is blocking modernization today**



2

# Modernization Planning

# Key objectives of this stage

With the assessment complete, the next step is to shape the direction of the modernization effort.

## Setting priorities

The organization decides which initiatives will create the strongest impact. Some workloads require optimization, others benefit from containerization, and certain systems need deeper restructuring.

## Selecting the right approach for each workload

The organization decides which initiatives will create the strongest impact. Some workloads require optimization, others benefit from containerization, and certain systems need deeper restructuring.



## Mapping dependencies and sequencing

Modernization works best when changes follow a logical order.

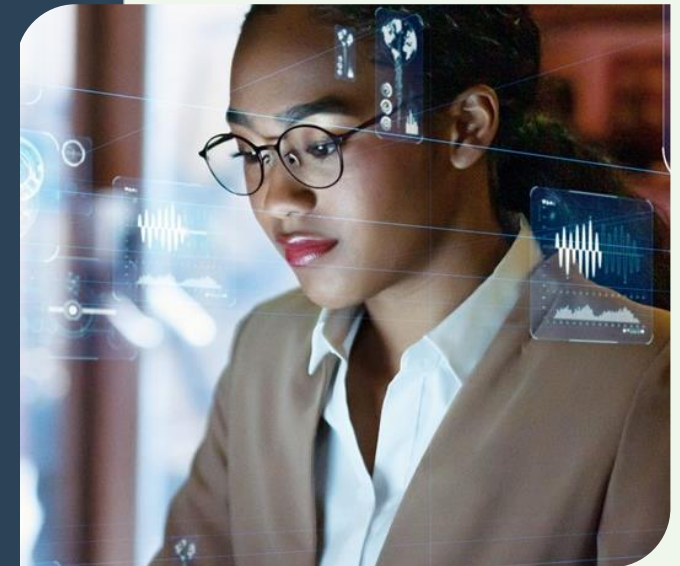
This sequencing reduces rework, avoids disruptions, and provides a clear path toward milestones.

## Governance and ownership

The plan defines who approves changes, how progress is communicated, and how the organization will maintain consistency across teams.

## Training and enablement

Teams receive clarity on the support they will have. This prevents confusion and reduces resistance once implementation begins. A solid plan gives the modernization effort momentum and keeps all stakeholders aligned.



3



# Implementation: Building the Modernized Environment

This stage includes the technical and operational work needed to bring modernization to life.

## Technical enhancements



Introducing containerization for workloads that benefit from modularity.



Improving performance with optimized configurations.



Implementing infrastructure-as-code, CI/CD pipelines, and automated deployments.



Strengthening identity, monitoring, and observability.



Updating integrations and removing legacy dependencies.



Introducing containerization for workloads that benefit from modularity.



## Workflow alignment

As the modernized environment takes shape, the way teams collaborate needs to evolve with it.

This phase focuses on establishing clear ownership, consistent release practices, and dependable handoffs across engineering, operations, and security.



The goal is to bring structure to how changes move from idea to deployment: reducing rework, eliminating bottlenecks, and giving teams a predictable rhythm for delivery. **Well-aligned workflows** ensure that the capabilities built into the new environment are actually used and sustained.

- ✓ Clear ownership
- ✓ Consistent release practices
- ✓ Dependable handoffs

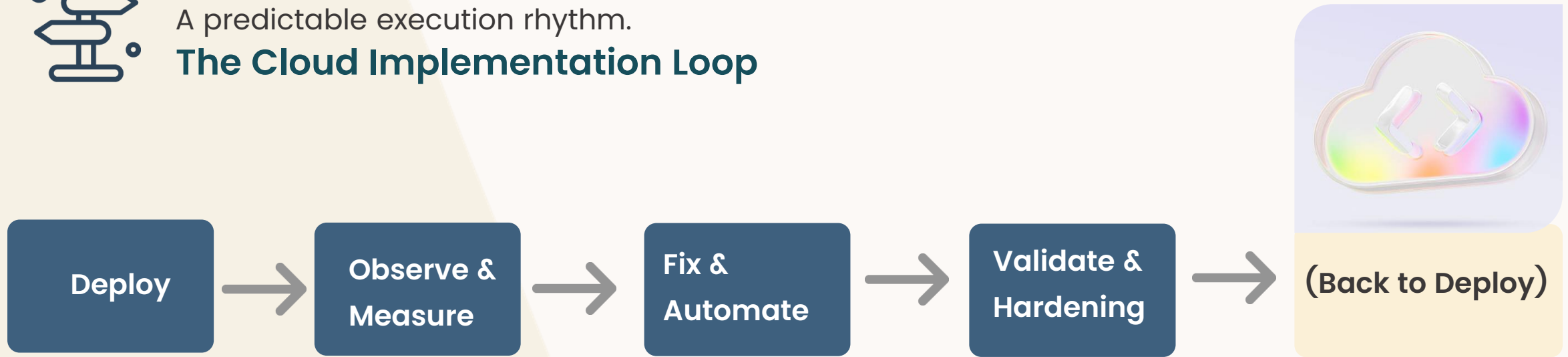
### Structure





A predictable execution rhythm.

## The Cloud Implementation Loop



**Incremental changes create predictability**, surface issues early, and reduce risk before wider adoption.

This cycle reflects the way teams work when systems begin shifting to the cloud. Instead of relying on one large migration moment, the environment matures through a sequence of small, controlled steps. It creates predictability, helps surface issues early, and allows the team to refine the system before broader rollout.

## Field Note



*The most successful AWS migrations happen when technical architecture and organizational workflows evolve together. We've seen projects stall not because the infrastructure failed, but because teams continued working in patterns designed for their old environment.*

*Before moving the first workload, map **how code moves from development to production**, who approves what, and where manual handoffs slow things down. Your AWS architecture should eliminate friction, not recreate it in the cloud. When workflow and infrastructure align, deployments accelerate and incidents decrease.*

Neo Lee

Founder & Cloud Architecture Lead, Rambunct

4

# Improvement: Driving Team Enablement & New Workflows



Organizations often assume modernization value appears the moment workloads land in the cloud. In practice, the majority of value emerges after go-live.

McKinsey estimates:



**42% of the ROI**

of cloud migration occurs post-migration, through optimization, tuning, and operational improvements.

This stage focuses on:

### a) Hands-on support

Teams need support as they adapt, especially finance operations. Uncertainty is normal, this is where behavioral change solidifies.

## b) User feedback loops

Real usage reveals issues planning doesn't catch: unexpected latency, missing automations, misunderstood permissions, model reporting lags

## c) Incremental training

Once teams gain confidence: introduce advanced automation, opportunities to expand monitoring, refine analytics, improve pipelines (batch → streaming where relevant)

## d) Performance tuning

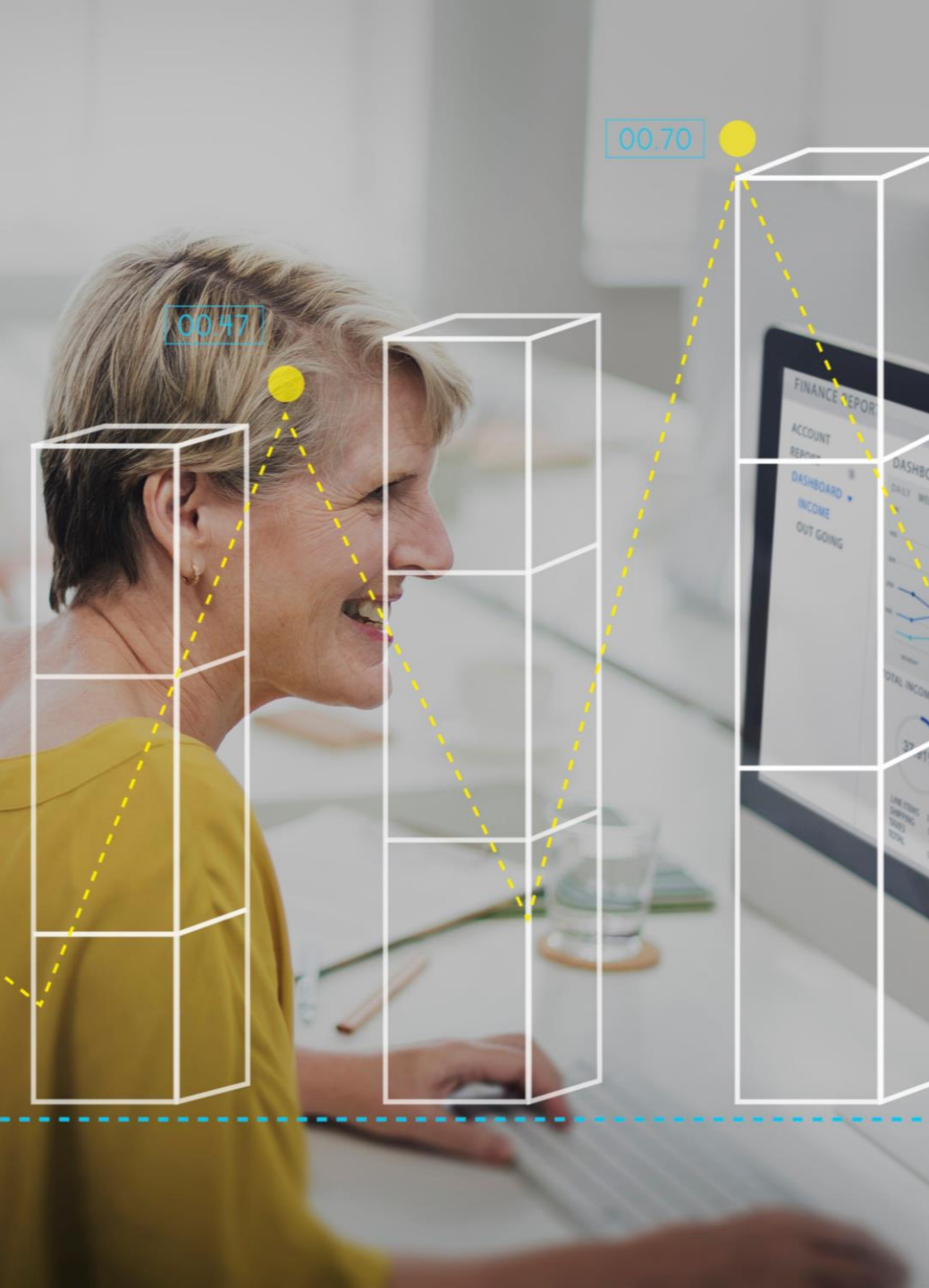
Performance tuning relies on real operational metrics such as latency, error frequency, scaling behavior, resource consumption, and cost anomalies. Monitoring and adjusting these signals over time helps stabilize the environment and turn the cloud into a lasting operational advantage.



5

# Support & Continuous Evolution





## Stabilize → Review → Adjust → Extend → Repeat

Once the platform is stable, teams move into a continuous rhythm of review and adjustment. Architecture decisions are revisited using real usage data, services are simplified or extended, and operational patterns are refined as new needs emerge. This prevents complexity from accumulating and keeps the environment aligned with how the business actually operates.

At the same time, automation expands beyond deployment into monitoring, remediation, and policy enforcement. Manual effort gradually decreases, predictability increases, and the platform becomes easier to operate as scale grows.

# How this looks in practice

Through a modernization project led by Rambunct





# The Institute of Ireland

## Oracle to AWS RDS Migration



### Background

The Ireland Institute faced this exact landscape. Their Oracle workloads were stable “until they weren’t,” with hardware failures, manual recovery procedures, and CRM integrations that needed constant babysitting.

### Challenge

Their existing setup presented several roadblocks:

- Aging hardware prone to failure and requiring constant upkeep
- Limited scalability and high downtime risks
- Slow and manual backup and recovery procedures
- Complex CRM integrations needing constant oversight
- The goal was to migrate to a modern database platform
- without disrupting academic operations.



# The Institute of Ireland

## Oracle to AWS RDS Migration



Here's how the structured approach was deployed

### 1. Legacy Landscape

- On-prem Oracle DB
- Aging hardware
- Manual recovery
- Fragile CRM links

### 4. Modern Core on AWS

- RDS Multi-AZ
- Automated backups
- Monitoring + alerts
- CRM continuity

### 2. Readiness Mapping

- Dependency tracing
- Performance baseline
- Integration review
- Risk points flagged

### 5. Operational Impact

- Faster queries
- Lower admin overhead
- Reduced downtime risk
- Foundation for growth

### 3. Migration Path

- Oracle Data Pump (export/import)
- Data Guard sync
- Controlled cutover

## Results

The migration delivered measurable improvements in performance, efficiency, and cost savings, while ensuring a smooth transition with no service disruption.

**60%** improvement in database query performance

**75%** reduction in admin overhead and manual operations

**40%** lower total cost of ownership compared to legacy hosting

### Seamless

cutover with zero data loss and uninterrupted CRM connectivity



# The Institute of Ireland

## Oracle to AWS RDS Migration



Once the landscape was documented, the readiness work exposed the real anchors: dependency chains that were longer than expected, data flows that would not survive a direct lift, and legacy routines that introduced risk into any migration plan.

Without mapping the starting point, teams end up discovering structural issues when it's too late to adjust.

With those insights, the migration path became a sequence rather than a leap. Live replication kept data aligned, export/import controlled the cutover, and contingency points were built into the flow. The goal wasn't speed; it was continuity—moving a core system without disrupting the operations that depended on it.

The result was a new operational baseline on AWS: automated backups, predictable failover, structured monitoring, and CRM integrations that no longer required constant oversight. The shift didn't just improve performance; it stabilized the foundation that everything else relied on.

# About Rambunct

Rambunct is a UK-based cloud and data consultancy founded in 2021, serving clients across the EU and the US. With more than 200 successful AWS launches, Rambunct helps companies migrate, modernize, and operate efficiently on AWS.

The team specializes in legacy-to-cloud transitions, cost and performance optimization, automation, and lightweight DevOps engineering—supporting SMBs that need clarity, stability, and speed without enterprise-level overhead.

**[Book a free consultation](#)**



Thanks!

