AWS CLOUD MIGRATION

Enterprise Db2 Modernization:

IBM Db2 to Amazon Aurora on AWS

How a leading U.S. ultra-low-cost carrier retired seven mission-critical on-premises IBM Db2 databases and migrated to Amazon Aurora PostgreSQL — achieving 45% cost reduction, sub-30-second automated failover, and 65% DBA maintenance reduction in 12 weeks.

45%
Total Cost Reduction
Fully-loaded 3-yr TCO vs on-prem Db2

<30 sec
Automated Failover RTO
Down from 2–4 hr manual recovery

65%
DBA Maintenance Reduction
30 hrs/week → 11 hrs/week

Engagement Summary

A renowned U.S. ultra-low-cost carrier — one of the fastest-expanding airlines in the country — migrated seven mission-critical IBM Db2 databases spanning reservations, pricing, scheduling, and passenger data management to Amazon Aurora PostgreSQL. The on-premises source environment was fully decommissioned following completion of the final overnight cutover. The engagement was completed in 12 weeks using a structured six-phase methodology, with zero data loss confirmed across all seven databases.

The Challenge

Customer Background

A renowned U.S. ultra-low-cost carrier — one of the fastest-expanding airlines in the country — ran its most critical airline operations on aging on-premises IBM Db2 databases. Reservations, pricing, scheduling, and passenger data management all depended on this legacy infrastructure across seven mission-critical database workloads.

Operational Constraints

The cost and complexity of maintaining on-premises Db2 infrastructure had reached a tipping point. IBM Db2 licensing, hardware refresh cycles, and ongoing maintenance obligations consumed a disproportionate share of IT budget. The shrinking pool of qualified Db2 specialists created concentration risk and elevated retention costs. Approximately 30 DBA hours per week were consumed by patching, backup monitoring, storage management, and incident response — delivering operational continuity rather than business value.

Technical Constraints

The monolithic on-premises Db2 architecture could not scale dynamically during peak travel periods — holiday surges, flash sale events, and high-volume booking windows. Fixed-capacity infrastructure created performance bottlenecks at precisely the moments of highest business and customer impact. Reservation throughput, pricing query response times, and passenger data access were all constrained by infrastructure that could not flex with business demand.

Strategic Constraints

The legacy Db2 environment had become a blocker on the airline's broader cloud adoption roadmap. Downstream initiatives in real-time operations analytics, dynamic pricing intelligence, and customer experience modernization all depended on a modern, cloud-native data layer the on-premises architecture could not support. Every quarter on legacy infrastructure represented compounding technical debt across the entire digital transformation programme.

Migration Risk as a Barrier

For an airline operating at scale, database availability is a direct operational dependency. Even brief periods of unavailability or data integrity issues during migration could cascade into reservation failures, pricing disruptions, scheduling errors, and passenger-facing breakdowns — each carrying regulatory, financial, and reputational consequences. The absence of a proven, low-risk migration pathway had historically made modernisation feel more threatening than remaining on aging infrastructure.

> **The Core Challenge**
>
> *Seven mission-critical Db2 databases. Aging on-premises infrastructure. No automated failover. Fixed capacity unable to scale with peak demand. A proven, low-disruption migration pathway was required — one that eliminated operational risk without putting airline operations at stake.*

Why Amazon Web Services?

Amazon Aurora PostgreSQL was selected as the target database platform based on four strategic criteria:

Lift-and-Shift Compatible	Managed Operations	Elastic Scalability	Open-Source Foundation
AWS SCT and DMS minimise application code changes, reducing migration risk across seven interconnected workloads.	Automated patching, Multi-AZ failover, continuous backups, and elastic storage eliminate the operational overhead driving the airline's DBA burden.	Aurora Read Replicas and storage autoscaling deliver the dynamic capacity the airline needed during peak travel periods and flash sale events.	PostgreSQL-compatible Aurora eliminates proprietary Db2 licensing constraints and opens access to a broad, growing engineering talent pool.

The Solution

Migration Methodology

The AWS Partner designed and delivered a structured six-phase migration framework to safely move all seven mission-critical Db2 databases to Amazon Aurora PostgreSQL — balancing speed with risk management across a regulated aviation environment where database availability is a direct operational dependency.

Phase 1	Phase 2	Phase 3	Phase 4	Phase 5	Phase 6
Assessment	**Schema Conversion**	**Pilot Migration**	**Full Migration**	**Cutover**	**Optimisation**
Inventory, SCT analysis, risk register	SCT automation + Db2 remediation	Non-prod to Aurora, perf baseline	DMS full load + CDC replication	Overnight windows, zero data loss	Right-sizing, tuning, Reserved Instances

Phase Detail

Phase 1 — Assessment

A comprehensive inventory of all seven Db2 databases was conducted — cataloguing schema complexity, stored procedure dependencies, data volumes, application integration points, and downstream system dependencies. AWS SCT was run against all seven environments to produce the conversion assessment report that governed remediation workplan and timeline commitment.

Phase 2 — Schema Conversion

AWS Schema Conversion Tool automated the majority of schema conversion across all seven databases. The AWS Partner's database engineers remediated Db2-specific constructs — proprietary SQL extensions, Db2 data type mappings, and application-specific stored

Phase 4 — Full Data Migration

AWS DMS executed the full data migration using an initial full load followed by continuous CDC replication — allowing all seven source Db2 databases to remain fully operational throughout the migration period. CDC lag was monitored continuously against defined thresholds. A parallel run period validated data integrity, application behaviour, and performance against the production workload profile before cutover approval was granted for each database.

Phase 5 — Cutover

Each of the seven database cutovers was executed during overnight off-peak travel windows — periods of minimum reservation transaction volume and lowest passenger-facing system dependency. A documented abort criteria

procedure logic — that had no direct Aurora PostgreSQL equivalent. Each remediated object was validated against functional parity criteria before advancing.

Phase 3 — Pilot Migration

A non-production replica of the reservations database — the highest-complexity and most integration-dependent workload — was migrated to Aurora PostgreSQL to establish a validated performance baseline, confirm application compatibility across core operational integrations, and produce the cutover runbook template applied to all subsequent database migrations.

framework defined conditions, decision authority, and rollback procedures for each window. Zero data loss was confirmed across all seven cutover events.

Phase 6 — Post-Migration Optimisation

Following source decommission, the AWS Partner conducted Aurora instance right-sizing based on 30 days of live production data, Read Replica count optimisation, RDS Proxy connection pool tuning, and S3 lifecycle policy configuration for backup retention tiering. Reserved Instance coverage was locked in post-right-sizing to ensure the three-year cost model reflected actual production sizing.

Solution Architecture

The target architecture is centred on Amazon Aurora PostgreSQL as the fully managed, cloud-native database platform for all seven migrated workloads — deployed across multiple AWS Availability Zones for full high availability with automatic failover. Aurora Read Replicas provide elastic read scaling for the highest-throughput workloads. Amazon RDS Proxy manages connection pooling for the airline's high-concurrency application tier. All database instances operate within private VPC subnets with no public exposure, secured by AWS KMS encryption at rest and TLS in transit.

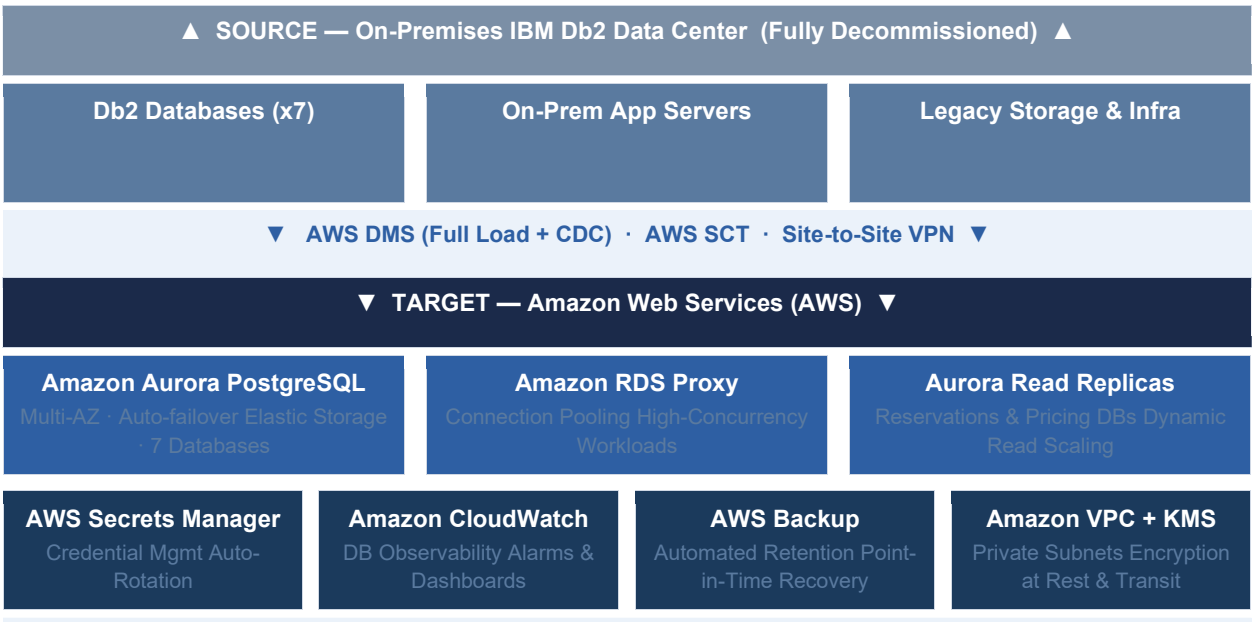

Figure 1: Multi-tier AWS Target Architecture — On-Premises Db2 Source Fully Decommissioned

AWS Services Used

Amazon Aurora PostgreSQL	**AWS Database Migration Service**	**AWS Schema Conversion Tool**
Amazon RDS Proxy	**AWS Secrets Manager**	**Amazon CloudWatch**
AWS Backup	**Amazon VPC**	**AWS IAM**
AWS KMS	**AWS Site-to-Site VPN**	

Business Benefits & Outcomes

Operational Outcomes

Automated patching, automated backups, storage autoscaling, and Multi-AZ failover — delivered natively by Amazon Aurora PostgreSQL — removed the primary categories of recurring DBA overhead that characterised the on-premises Db2 environment. Weekly database administration effort across the seven migrated workloads was reduced by 65%, from approximately 30 hours per week to under 11 hours per week, with freed capacity redirected to strategic cloud adoption initiatives. The concentration risk associated with specialised Db2 expertise was eliminated entirely, replaced by the broad and growing Aurora PostgreSQL engineering talent pool.

Performance & Scalability Outcomes

Aurora Read Replicas provisioned for the reservations and pricing databases delivered dynamic read scaling capability that was structurally unavailable in the on-premises Db2 architecture. During the first peak travel period following go-live, the platform scaled read capacity automatically to handle the surge in reservation queries and pricing engine lookups without impacting write performance. Query response times for the reservations database improved by approximately 40% under peak load conditions. The flash sale event profile — previously one of the highest-risk operational scenarios for the airline's database infrastructure — was handled without performance degradation or manual intervention for the first time following migration.

Availability & Recovery Outcomes

Aurora Multi-AZ deployment delivered automated failover with a recovery time objective of under 30 seconds — replacing an on-premises environment with no automated failover and an estimated manual recovery time of 2–4 hours. Recovery point objective improved from daily backup cycles carrying up to 24 hours of potential data loss exposure to continuous automated backup with point-in-time recovery to within 5 minutes across all seven databases. Zero unplanned database outages were recorded across all seven migrated workloads during the 90-day post-migration production period.

Cost Outcomes

IBM Db2 licensing and on-premises hardware maintenance obligations were fully eliminated following source decommission. The consumption-based Aurora pricing model replaced fixed infrastructure costs with variable spend that scaled directly with actual workload demand — delivering cost efficiency during off-peak travel periods unavailable in the fixed-capacity on-premises environment. Total database infrastructure cost was reduced by approximately 45% on a fully-loaded three-year basis. Hardware refresh capital expenditure of $600K–$800K — previously a pending obligation — was avoided entirely.

Strategic Outcomes

The migration to Aurora PostgreSQL removed the primary blocker on the airline's broader cloud adoption roadmap. Downstream initiatives in real-time operations analytics, dynamic pricing intelligence, and customer experience personalisation — each dependent on a modern, cloud-native data layer — were unblocked following completion of the Db2-to-Aurora migration. The open-source PostgreSQL-compatible platform established by Aurora provides a foundation for future modernisation initiatives that the proprietary Db2 environment could not have supported.

Before vs. After

Metric	Before (On-Premises Db2)	After (Amazon Aurora PostgreSQL)
Database Platform	IBM Db2 — on-premises, manual ops	**Amazon Aurora PostgreSQL — fully managed**
Automated Failover / RTO	None — 2–4 hour manual recovery	**Multi-AZ auto-failover — under 30 seconds**
Recovery Point Objective	24 hours — daily backup cycle	**Under 5 minutes — continuous backup**
DBA Maintenance Effort	~30 hrs/week — patching, backups, ops	**~11 hrs/week — 65% reduction**
Peak Query Response	Bottlenecks at peak travel periods	**~40% improvement under peak load**
Read Scalability	Fixed capacity — no dynamic scaling	**Aurora Read Replicas — elastic on demand**
Unplanned Outages	Recurring — storage & patching events	**0 — across 90-day post-migration period**
Total Cost Reduction	—	**~45% on fully-loaded 3-year basis**
Capex Avoided	$600K–$800K hardware refresh pending	**Eliminated — no on-prem refresh required**
Engagement Timeline	—	**12 weeks — discovery to decommission**
Data Loss on Cutover	—	**Zero — confirmed across all 7 databases**

TCO Analysis

The AWS Partner conducted a structured Total Cost of Ownership analysis comparing the airline's fully-loaded on-premises IBM Db2 environment against the projected Amazon Aurora PostgreSQL target state over a three-year horizon across five cost categories.

Cost Category	On-Premises Db2 Baseline	AWS Aurora Target State
Database Licensing	IBM Db2 enterprise licensing + support — fixed annual cost regardless of utilisation	**Eliminated — Aurora pricing is consumption-based and inclusive of managed service overhead**
Infrastructure & Hardware	$600K–$800K hardware refresh pending over 3-year horizon; SAN, server, and network maintenance	**Capex obligation fully eliminated — Aurora storage autoscaling replaces fixed hardware provisioning**
Operational Labour	~30 DBA hrs/week — patching, backups, monitoring, capacity planning; specialised Db2 skill premium	**~11 DBA hrs/week — 65% reduction; freed capacity redirected to cloud adoption roadmap initiatives**
Availability Risk	No automated failover — 2–4 hour manual RTO; peak-period outage estimated at $500K–$1M per incident	**Aurora Multi-AZ — sub-30-second automated failover; risk-adjusted exposure materially eliminated**
3-Year TCO Outcome	Baseline — full cost model including licensing, hardware, labour, and risk-adjusted availability	**~45% total cost reduction on a fully-loaded 3-year basis**

Lessons Learned

The engagement delivered successful outcomes across all defined success criteria. Six key lessons were identified that strengthen the repeatable framework applied to future Db2 modernisation workloads.

1. Front-Load SCT Analysis — Schema Complexity Is Consistently Underestimated

IBM Db2 environments accumulate proprietary SQL extensions, platform-specific data type implementations, and stored procedure logic with no direct Aurora PostgreSQL equivalent. Running AWS SCT against the actual environment — not documentation — as the first executable activity in the Assessment phase is mandatory. The SCT assessment report must be the primary input to timeline commitment. Three of the seven databases required timeline adjustment after SCT revealed higher-than-anticipated remediation scope.

2. Multi-Database Cutover Sequencing Requires an Inter-Database Dependency Map

Application-level dependencies between databases — particularly between reservations and pricing, and between scheduling and passenger data management — meant cutover sequencing had to account for dependency chains,

not just individual database readiness. A formal inter-database dependency map, produced during Assessment and validated with application owners, is a mandatory deliverable for all multi-database migration engagements.

3. Overnight Cutover Windows Should Align to Airline Schedule Boundaries

Transaction volume alone is an incomplete proxy for cutover risk in the aviation vertical. Aligning cutover windows to the overnight period between the last departure of the day and the first departure of the following morning — rather than to transaction volume troughs — minimises downstream operational dependencies across check-in, gate management, and crew scheduling systems simultaneously.

4. Aurora PostgreSQL Query Performance Requires Active Tuning for Db2-Originated Workloads

A subset of high-complexity analytical queries in the pricing and scheduling databases exhibited performance regression on Aurora PostgreSQL that was not apparent during pilot testing against non-production data volumes. Query plan differences between the Db2 and PostgreSQL optimisers — around join ordering, index utilisation, and statistics freshness — required targeted tuning. A structured query performance validation protocol covering the top 50 queries by frequency and top 20 by execution time is now a mandatory parallel run gate criterion.

5. RDS Proxy Connection Pool Sizing Must Be Based on Peak Production Data

Initial RDS Proxy configuration based on application documentation estimates was exceeded during the first peak travel period post-launch. For airline reservation workloads, flash sale event connection profiles must be the primary sizing input — not average or documented connection counts — with a defined headroom buffer above the highest observed peak.

6. Application Team Engagement Is the Critical Path Item

The most consistent source of schedule risk across the 12-week engagement was not database complexity — it was application team availability for integration validation and cutover sign-off. Application team validation activities sat on the critical path for every database cutover. Establishing formal engagement commitments — named validation owners, defined review SLAs, and scheduled milestones — at project kickoff is mandatory for all multi-database migration engagements.

About the Customer

The customer is a renowned U.S. ultra-low-cost carrier and one of the fastest-expanding airlines in the country. Operating across a rapidly growing domestic and international route network, the airline serves millions of passengers annually across its reservation, pricing, scheduling, and passenger management platforms. The airline operates in a highly regulated aviation environment under FAA oversight and state-level consumer protection requirements, with database availability representing a direct operational and compliance dependency across all passenger-facing and back-office systems.

About the AWS Partner

Agateverse is an AWS Partner specializing in cloud migration, database modernization, and managed services for enterprise customers across regulated industries. With deep expertise in AWS migration tooling — including AWS DMS, AWS SCT, and Amazon Aurora — the partner delivers structured, low-risk migration frameworks that minimize operational disruption while accelerating cloud adoption. The partner's database engineering team has executed complex database migrations across aviation, financial services, healthcare, and public sector verticals, with a proven methodology validated against the most demanding mission-critical workload profiles.