Database Modernizer Assessment Tool

Executive Report

Source database wordpress

Approve wave 1: move wordpress to Aurora MySQL

Wave 1: Suggested, 2 warnings to accept

Assessment id 8dd35fce · Generated 2026-11-18

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Built from code commit 8f99214b1709.

The recommendation

Hybrid architecture: Aurora MySQL retained as the relational core (66% of queries); DynamoDB for key-value access; ElastiCache for caching hot data.



Engine	Role	Workload	Scope	Est. monthly
Aurora MySQL	Retained	66.3%	source schema retained	$497.25
DynamoDB	Migration target	33.7%	20 source tables served: 19 migrate, 1 kept current by CDC	$618.61
ElastiCache	Cache layer	38 designed cached reads · 82.1% of calls	31 key designs	$184.59
Total		**100%**	**19 tables migrate**	**$1,300.45**

Full-modernization estimate: all waves done, every kept engine running. Estimates from public on-demand list prices for the stated Region. Check them with your own pricing in the AWS Pricing Calculator (https://calculator.aws/).

Aurora MySQL is retained as the relational core (source-compatible, no migration). The migration moves the 19 tables assigned to DynamoDB. In the Scope column, “served” is the source tables a migration target’s queries read. The tables it owns migrate, and the Total counts only those. A table another engine of the same wave owns moves to that engine. A table an engine outside the wave owns stays there and is kept current by CDC from it, as in the wave plan.

ElastiCache is recommended as a cache-aside layer in front of the engine that owns each cached read: the cached reads are hot, small and can be rebuilt from that engine, so it stays the system of record and the cache is sized for the hot reads, not the whole dataset. ElastiCache for Valkey 9.0 and later offers durability on Multi-AZ clusters; this assessment does not use it as a system of record today.

39 cache candidates, 38 designed; it stays on its owner engine.

Table owners: Of the 23 mapped tables, DynamoDB owns 19 and Aurora MySQL owns 4; 20 tables are also read by another engine through CDC replicas; 17 tables are cached by ElastiCache.
Each table has one owner, as in the wave plan: the engine that takes its writes; a table no query writes stays on the plan's relational engine, or on the source database when the plan has none. A cache never owns a table. Every other engine that reads a table reads a copy kept current from its owner via CDC.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

The numbers

$1,300.45
Projected monthly cost
Full modernization (all waves done)
List prices: AWS Price List, us-east-1 on-demand, retrieved 2026-11-04

3
Engines in the target architecture
Aurora MySQL, DynamoDB, ElastiCache

DocumentDB, OpenSearch
Removed by the reality check
DocumentDB: queries moved to Aurora MySQL, DynamoDB
OpenSearch: queries moved to Aurora MySQL

55
of 163 queries move to a new engine
19 tables migrate
163 queries in 149 distinct query patterns

COST BY ENGINE

Engine	Role	Est. monthly	Sized for
Aurora MySQL	Retained	$497.25	108 queries, 647.74 calls/s, plus 27 tables no query reads
DynamoDB	Migration target	$618.61	55 queries, 923.66 calls/s
ElastiCache	Cache layer	$184.59	38 cached reads, 1,289.35 calls/s, also billed on their owners
Total		**$1,300.45**	

Assumptions: On-demand capacity mode for all target databases; Does not include data transfer, backups, or global tables; Reads the cache layer fronts are also billed on their owner engines, as if every one missed the cache (conservative); 27 source tables no in-scope query reads (<0.01 GB) are stored on Aurora MySQL, where wave 1 moves them; The DynamoDB figure includes $122.12/mo for the design's 5 global secondary indexes at on-demand prices: $0.05 of storage (each index's item count times its projected item size plus 100 bytes of index overhead per item) and $122.07 of index writes (one for each put or delete whose item may set the index's keys, two for an update that changes an index key or does not say what it changes, one for an update that changes only a projected attribute, billed at the standard rate per 1 KB of the projected item, on average load with the collected rows per call, an average of exactly zero counted as one row). Index reads are not added, since each is one of its query's reads and already priced, and the free tier is left out; DynamoDB reads are priced eventually consistent, except transactions, which are priced at transactional rates..

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

The numbers (continued)

Call rates: execution counts are divided by each statement's own observed window (163 statements), the server uptime (0 statements), or an assumed 24 hours (0 statements); costs assume that rate runs continuously.

Growth: 5× (default), shown as a separate what-if scenario. The costs and risks in this report use the collected traffic.

Peak: not measured (the collection has no CloudWatch metrics covering 7 to 14 days): the rates are used as measured (default).

Region and prices: us-east-1, on-demand prices from the AWS Price List retrieved 2026-11-04 (default).

Growth scenario: 1× $1,300.45/mo (the collected traffic), 5× $3,774.37/mo (the default growth), 10× $6,866.77/mo; a what-if only, not part of the costs and risks in this report.

Busiest estimated partition key value (reads), share of its limit: 1× 0.15%, 5× 0.76%, 10× 1.5%; that key value can take 656× its collected traffic before it reaches its limit.

Scenario at the measured rate: the rate window is not representative (under a day; short against the server's uptime; counted from a reset or a restart; well above the server's own rate), so the costs and the rate-driven risks assume that window's load runs all day, every day. Collect again over a longer window (at least 7 days of usual load, without resetting the statement statistics first) for figures that represent normal traffic.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Why this architecture

Engine	Role	Why
Aurora MySQL	Retained	75% mean fit across 90 queries (16 rated tables; 1 query without table-level evidence), kept relational by aggregation, full-text or pattern search and multi-table joins (42 of all 108 routed); 18 more queries are pinned (utility), not in the mean. Schema design: 22 target tables, 108 in-scope access patterns.
DynamoDB	Migration target	86% mean fit across 55 queries (20 rated tables), led by key-value lookups (15 of 55). Schema design: 17 target tables, 48 in-scope access patterns.
ElastiCache	Cache layer	Cache layer for 38 designed cached reads (82.1% of calls), mostly point lookups, cache-aside in front of the engines that own these reads, in the order the wave plan gives, which stay the system of record; 37% mean cache fit across 38 cached reads (17 rated tables). Schema design: 31 key designs, 34 in-scope access patterns. These reads total 1289.35 calls/s, above the 1 calls/s hot-read floor.

Workload fit: how closely this engine's capabilities match the access patterns of the queries routed to it. It is not a probability that the migration will succeed.

Cache fit: how closely the cache layer's capabilities match the reads it fronts. The cache layer owns no queries, so this is not a workload fit and is not compared with the owning engines' figures.

Reality-check decisions
Consolidated 15 queries from DocumentDB to Aurora MySQL: DocumentDB provides no unique capabilities over Aurora MySQL. No saving is claimed: DocumentDB does not run today; it was a candidate this plan considered and does not recommend.
Consolidated 4 queries from DocumentDB to DynamoDB: DocumentDB provides no unique capabilities over DynamoDB. DocumentDB does not run today, so no saving is claimed.
Consolidated 1 query from OpenSearch to Aurora MySQL: OpenSearch provides no unique capabilities over Aurora MySQL. No saving is claimed: OpenSearch was never justified by the workload. None of its first-assignment queries clear the real-search-depth plus traffic-or-result-size floor a standing deployment needs, so no infrastructure cost was ever justified for it.
Recommended pattern: Event-Driven Sync: Use database change streams or event logs to propagate changes between databases asynchronously. Each database maintains its own optimized representation of the data.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Migration waves

This is one suggested adoption path. To modernize in one step instead, adopt the target architecture directly. The cutover is your responsibility.

A wave is suggested only when its design exists, was reviewed, and no check holds it: an open critical or high risk on it, an access pattern its design leaves unserved, a design review that did not approve its design, a plan that costs more than the source while the wave carries an open performance risk, or a wave 1 that is not suggested yet.

WAVE 1

Move to Aurora MySQL

all 163 queries at cutover; 108 (66.3%) remain on Aurora MySQL at the end, 50 source tables and views

Engines: Aurora MySQL

Moves from: the source MySQL database

Why: The whole source database (MySQL 8.0.45) moves 1:1 to Aurora MySQL: same engine family. The collector reports engine and version only; feature compatibility is not assessed here. 108 queries (66.3%) remain on Aurora MySQL once later waves have moved their share; later waves move subsets of these 50 tables and views out of Aurora MySQL. 23 of these tables and views are mapped to a target engine in the migration map; later waves move 19 of the 50 off Aurora MySQL, and the other 31 stay on Aurora MySQL: 4 migration-map rows that do not migrate and 27 tables and views the map does not list.

Gate before the next wave: Schema and data parity validated against the source database. Later waves move from Aurora MySQL, not from the source MySQL database. The cutover is your responsibility. Decommissioning the source database is your responsibility. Close RISK-005 and RISK-006 before this gate (source objects' database logic and design trade-offs).

Stages: 1 stage (the whole database at once)

Open critical or high risks: 0

Readiness: Suggested, with 2 design review warnings to accept (listed in the engineering report). If you do not accept them: revise the Aurora MySQL schema design in your modernization session and review it again.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Migration waves (continued)

WAVE 2

Cache hot reads with ElastiCache

38 queries (82.1% of calls)

Engines: ElastiCache

Fronts: Aurora MySQL

Why: 38 hot reads (82.1% of calls), cache-aside in front of Aurora MySQL in wave 2, then DynamoDB once wave 3 moves these reads: no data migration, fully reversible. It takes the read pressure off before the data migrations that follow. These reads belong to queries that move in wave 1 (Aurora MySQL) and wave 3 (DynamoDB); this share overlaps the owner shares, which alone sum to 100%.

Gate before the next wave: Cache hit rate and invalidation verified against Aurora MySQL. Close RISK-003, RISK-004 and RISK-008 before this gate (dropped cache reads, unsupported cache patterns and design trade-offs).

Stages: 1 stage (the reads the cache fronts)

Open critical or high risks: 0

Readiness: Suggested, with 3 design review warnings to accept (listed in the engineering report). If you do not accept them: revise the ElastiCache schema design in your modernization session and review it again.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Migration waves (continued)

WAVE 3

Move key-value and point-lookup queries to DynamoDB

55 queries (33.7% of queries), 20 source tables: 19 to move, 1 kept current by CDC from its owner

Engines: DynamoDB

Moves from: Aurora MySQL

Why: 55 key-value and point-lookup queries (33.7% of queries) across 1 table group, where co-dependent tables move together: the pattern DynamoDB fits best, and the smallest-blast-radius data migration available once the cache wave has absorbed the read pressure.

Gate before the next wave: Dual-write/backfill validated and query parity confirmed per table group. All 19 tables moving to DynamoDB stay dual-read by Aurora MySQL queries, which remain on Aurora MySQL; DynamoDB owns them, so keep the Aurora MySQL copy current from DynamoDB via CDC. 1 query read 1 table Aurora MySQL owns (wave 1) and keeps owning; keep a DynamoDB copy of them current via CDC. Close RISK-001, RISK-002, RISK-007 and RISK-009 before this gate (tables kept current by CDC, close calls, design trade-offs and redesigns routing relies on).

Stages: 2 stages (reads on a CDC copy, ownership with a CDC copy back to an earlier engine)

Open critical or high risks: 0

Readiness: Suggested, with 4 design review warnings to accept (listed in the engineering report). If you do not accept them: revise the DynamoDB schema design in your modernization session and review it again.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Ways to adopt

Each path is this plan stopped after one of its steps: it takes steps 1 to that step, and the queries of every later step stay on Aurora MySQL. The staged full target is the default; an earlier stop is a smaller end state.

Path	Steps	Gates	Queries off Aurora MySQL	Monthly cost	Against the relational move only	Open critical and high risks	Engines
Relational move only	2	2	0	$681.84	-	0	Aurora MySQL, ElastiCache
Full target, the default	3	4	55	$1,300.45	$618.61 more	0	Aurora MySQL, ElastiCache, DynamoDB

Each path's cost adds up its engines' estimates for the queries that path places on each of them, so the relational engine is estimated for every query the path leaves on it.

ONE STEP

The same end state in one step, in place of 3 steps and 4 stages.

Its one gate holds all 4 stage gates.

It needs none of the 19 CDC copies back to an earlier engine that the staged form carries.

These checks describe the shape of this workload. Not every check holds, so it does not suit a one-step move.

Check	Observed	At most	Holds
No open critical or high risk	0	0	yes
At most one purpose-built target	1	1	yes
No stage that needs a CDC copy back to an earlier engine	1	0	no

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Risks

Medium
Overall risk

9
Open risks
9 medium

4
Resolved by the assignment
1 high, 3 medium

The rate-driven risks are a scenario at the measured rate too: the rate window is not representative (under a day; short against the server's uptime; counted from a reset or a restart; well above the server's own rate).

Overall risk is the highest severity among the open risks; risks the assignment resolved do not count.

RISK-001 · Medium · DynamoDB · data consistency

Wave 3: 104 queries read 20 tables on an engine that does not own them. 103 of them read a copy of 19 tables that CDC keeps current from DynamoDB across 1 engine pair, so a read there can miss a recent write. The other table is written by no captured query; CDC keeps its copy current too, and the copy changes only when something outside the captured workload writes the table.

Mitigation: Before Wave 3 starts, set up CDC for each engine pair the sources of truth section lists, and send any read that must see its own write to its source of truth; set up CDC for the table no captured query writes, and check whether anything outside the captured workload writes it. Compare the row counts on both sides. See the Wave 3 gate in the roadmap.

RISK-002 · Medium · DynamoDB · migration complexity

Wave 3: 29 queries were decided by less than 5 points, and 5 more were exact ties a routing rule settled: Aurora MySQL keeps 19 close calls and 5 ties over DynamoDB; DynamoDB takes 10 close calls over Aurora MySQL. A small change in the scores moves them.

Mitigation: Confirm the engine for these queries in the assignment review before Wave 3 starts, using the close calls section; move any that fit the runner-up better.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Risks (continued)

RISK-003 · Medium · ElastiCache · performance degradation

1 read the plan meant to cache was dropped from the cache layer and goes to Aurora MySQL uncached.

Mitigation: Load-test these reads on Aurora MySQL without the cache, or add a cache design that serves them. See the Wave 2 gate in the roadmap.

RISK-004 · Medium · ElastiCache · migration complexity

The cache design marks 1 pattern unsupported; the engineering report lists each one with its workaround under Target schemas by engine.

Mitigation: Serve this pattern from its owner engine or add it to the cache design, as the design's workaround for each says. See the Wave 2 gate in the roadmap.

RISK-005 · Medium · Aurora MySQL · operational risk

19 source objects need application code on Aurora MySQL: 17 column defaults; 1 deprecated sql; 1 zero date default. The schema carries them over without the database logic.

Mitigation: Move the logic of each one into the application, as the schema design's note for it says. See the Wave 1 gate in the roadmap.

RISK-006 · Medium · Aurora MySQL · migration complexity

The Aurora MySQL schema design records 5 trade-offs to review, listed under Migration trade-offs in the engineering report.

Mitigation: Review these trade-offs with the team that builds Aurora MySQL before Wave 1 starts, and test each one the application depends on.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Risks (continued)

RISK-007 · Medium · DynamoDB · migration complexity

The DynamoDB schema design records 41 trade-offs to review, listed under Migration trade-offs in the engineering report.

Mitigation: Review these trade-offs with the team that builds DynamoDB before Wave 3 starts, and test each one the application depends on.

RISK-008 · Medium · ElastiCache · migration complexity

The ElastiCache schema design records 10 trade-offs to review, listed under Migration trade-offs in the engineering report.

Mitigation: Review these trade-offs with the team that builds ElastiCache before Wave 2 starts, and test each one the application depends on.

RISK-009 · Medium · DynamoDB · migration complexity

Routing did not penalise DynamoDB for the frequent-full-scan finding the source shows on 1 query, because the design removes the read it is about (each query's evidence says how).

Mitigation: Design this query in DynamoDB as each one's evidence says, and serve none of them as a scan. See the Wave 3 gate in the roadmap.

The engineering report lists each one in full.

Resolved by the assignment: 1 high, 3 medium

DynamoDB to Aurora MySQL: resolved because the queries run as SQL on Aurora MySQL.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Decisions needed

No pending engine choices.

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.

Next steps

1. After wave 1 (Aurora MySQL): compare the latency of 163 queries with their source baseline (weighted average 1.33 ms) and check the engine sustains the collected traffic of 1,571.4 requests/s before starting wave 2.

2. After wave 2 (ElastiCache): track the cache hit rate for the 38 hot reads it fronts on Aurora MySQL (82.1% of calls, 1,289.4 calls/s) and compare their latency with the source before starting wave 3.

3. After wave 3 (DynamoDB): compare the latency of 55 queries with their source baseline (weighted average 0.76 ms) and check the engine sustains the collected traffic of 923.7 requests/s.

Full detail: engineering-report.md and index.md

Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved. For informational purposes only.