

— GUIDE · RABBITMQ

Migrate to RabbitMQ: Migration Checklist and Intake Worksheet for IBM MQ, ActiveMQ and TIBCO EMS

A migration to RabbitMQ can be scoped once a team has answered about thirty questions on workload, JMS features in use, coexistence, the target platform and compliance. This worksheet lists them, with the reason each answer moves the estimate. It is the intake, not the how-to: the sequence is in the migration guide and the concept mapping tables are on the migration services page.



By **Scott Sternloff**, Senior Enterprise Architect

LinkedIn • Updated October 29, 2026 • 14 min read • 12 sections

— THE ANSWERS THAT MOVE A MIGRATION ESTIMATE MOST

Broker migrations rarely overrun because of the broker. They overrun on application semantics nobody wrote down. Five answers decide most of the estimate: whether any application uses **queue selectors**, whether any flow depends on **XA or syncpoint transactions**, whether request-reply runs on **temporary queues**, which **client languages** are in use, and whether both brokers must **run side by side** while applications move one at a time.

Gather the other answers too, because they size the target cluster and the support scope. As at 29 October 2026, build the target on RabbitMQ 4.4 (the current series, released 20 October 2026); 4.3 is the previous series.

What this worksheet is for, and what to read instead

This page is for the person assembling the facts that let a migration be scoped, whether the scoping is done by your team, by us or by another provider. It does not explain how to migrate. For that, use the existing material:

- **The sequence:** the RabbitMQ migration guide covers decide, inventory, map, build, cut over, in order.
- **The concept mapping:** the RabbitMQ migrations service page holds the IBM MQ, ActiveMQ and Azure Service Bus mapping tables (queue manager to cluster and vhost, JMS topic to topic exchange, and so on).
- **The pre-cutover gate:** the migration readiness checklist covers rollback planning and load validation once the target is built.

Use the worksheet first, the guide to plan from it, and the checklist before cutover. TIBCO EMS is covered here only as a source of questions: nothing below makes a claim about how EMS behaves, so confirm EMS-specific features against TIBCO's own documentation.

What the current broker actually does

Write these numbers down before anything else, because sizing and cutover risk both come from them and almost nobody has them to hand.

- **Messages per second, average and peak, for queues and topics separately.** Peak, not average, sizes the target cluster, and topic fan-out multiplies the delivery rate on the RabbitMQ side.
- **Typical and worst-case payload size.** Large payloads change the disk, network and memory plan, and the worst case matters more than the typical one.
- **Queues holding large durable backlogs.** The RabbitMQ documentation points very long backlogs (5 million messages or more) at streams rather than quorum

queues, so a deep backlog can change the target queue type for that flow.

- **TTLs on messages or queues, and what happens when they expire.** RabbitMQ supports per-message and per-queue TTL on classic and quorum queues, but quorum queue TTL has a caveat (the queue lease is not renewed on re-declaration), so expiry behaviour has to be re-tested rather than assumed.
- **Concurrent client connections, and which are production.** The documented guideline for the open file handle limit is twice the 95th percentile of concurrent connections plus the total number of queues, with at least 50,000 as a floor.
- **Client languages, frameworks and versions.** This decides the client path: the RabbitMQ JMS client for Java JMS applications, a native AMQP 0-9-1 client, or an AMQP 1.0 client (the RabbitMQ team maintains AMQP 1.0 clients for Java and .NET). Every non-JMS client is a code change.

Features that do not map cleanly

This group is where migrations overrun: the broker swap is straightforward and the semantics are not. For each item, find the applications that use it.

- **Message selectors.** The RabbitMQ JMS client supports selectors for topics, through the RabbitMQ JMS Topic Exchange plugin (which must be enabled on the server), and does not support selectors for queues. The JMS client documentation names the Qpid JMS client over AMQP 1.0 as the way to get queue selectors, so a queue-selector estate chooses between a different client and moving the filter into routing.
- **Transactions.** The RabbitMQ JMS client documents that local transactions do not guarantee atomicity and that XA interfaces are not implemented. AMQP 1.0 in RabbitMQ does not support transactions. Any flow that relies on XA or on a syncpoint needs an idempotent consumer or an outbox pattern, and should be scoped first because it carries the longest lead time.

- **Request-reply and temporary queues.** The JMS client documents temporary reply queues and a direct reply-to option. Quorum queues cannot be exclusive and are not meant for temporary queues, so reply queues stay on classic queues or direct reply-to, which affects the queue-type plan.
- **Message priority.** Classic queues support priorities from 0 to 255 (a single-digit number is recommended). Quorum queues support strict priority with 32 levels (0 to 31) from RabbitMQ 4.3; before 4.3 they mapped priorities onto two buckets. If priorities are in use, the target series matters.
- **Ordering and message groups.** RabbitMQ delivers in order within a queue, but competing consumers and redelivery break strict sequence. The documented option for ordered processing is single active consumer. List every flow where order across consumers matters.
- **Redelivery and poison messages.** Quorum queues have a delivery limit, which defaults to 20 from RabbitMQ 4.0, with dead lettering after it. Map the source broker's backout or redelivery policy onto that limit and a dead letter exchange.
- **Exclusive queues and other broker-side features in use** (store settings, flow control, exclusive consumers). Quorum queues ignore the exclusive consumer flag; single active consumer is the replacement.
- **Client wire protocols.** IBM MQ's native protocol and ActiveMQ's OpenWire are not spoken by RabbitMQ. Applications on those protocols move to the JMS client or an AMQP client.
- **Other products bridging into the broker.** For EMS and IBM MQ estates, check for integration products that connect to the broker. They are outside the broker swap and need their own plan.

Running both brokers during the move

Plan for both brokers to be live at once, because almost every migration of any size does this and the bridge is the part nobody budgets for.

- **Can applications move one at a time, or do groups have to move together?** Groups that share a queue or a topic move together, so this sets the number and order of phases.
- **Is a bridge needed so a message published on the old broker reaches a consumer on the new one?** RabbitMQ shovels support AMQP 0.9.1 and AMQP 1.0 sources and destinations, so a source broker that can be reached over AMQP 1.0 can be shoveled into RabbitMQ directly; confirm this against the source broker's own documentation. Otherwise the bridge is an application-level relay.
- **How is duplicate delivery prevented while both are live?** RabbitMQ is at-least-once: after a failover or reconnect a consumer can receive the same message twice, so consumers need to tolerate duplicates before a bridge is added.
- **What is the rollback position for a single application after it has moved?** While the bridge runs, moving back is a consumer rejoin. Decide when the bridge is switched off, because that is when the rollback closes.

The target platform

Decide the target RabbitMQ series, queue type default and protocol per application during intake, not after the first application has moved. The version rows below are read from the release table in this site's data module and were checked against rabbitmq.com on 29 October 2026.

- **RabbitMQ version to target.** RabbitMQ 4.4.0 is current, released 20 October 2026; 4.3 is the previous series. We recommend a new target start on the current series, because starting a migration on an older one adds an upgrade project on top of it.
- **Erlang/OTP.** RabbitMQ 4.4.0 runs on Erlang 27.0 up to 29.x. 4.3.6 runs on 27.0 up to 28.x. Erlang 26 has reached end of life.
- **Metadata store.** On 4.3 and later, Khepri is the only metadata store. Mnesia was removed and so were the partition handling strategies, so a source-side question

such as "which partition mode do you use" has no equivalent. Cluster size is still planned around Raft majority, with an odd number of nodes.

- **Default queue type.** Queue type cannot be changed after declaration. Set `default_queue_type` for the node or per virtual host before applications declare their first queues, so durable queues land as quorum queues by default.
- **Protocol per application.** AMQP 1.0 is supported natively since RabbitMQ 4.0, with no plugin, alongside AMQP 0-9-1 on port 5672. A JMS application can use the RabbitMQ JMS client (JMS 2.0 and 3.0 editions exist) or a JMS client over AMQP 1.0. From 4.4, RabbitMQ can enforce a single connection per AMQP 1.0 container ID, behind a feature flag, which JMS-over-AMQP-1.0 clients use for client ID uniqueness.
- **Platform.** Kubernetes with the Cluster Operator, virtual machines, or bare metal. This sets storage, network policy for inter-node traffic, and who patches the operating system.

SERIES	LATEST PATCH	RELEASED	COMMUNITY SUPPORT TO	COMMERCIAL SUPPORT TO	ROLE MIGR
4.4	4.4.0	20 October 2026	31 July 2027	31 October 2028	Curre defau for a clust
4.3	4.3.6	23 April 2026	31 January 2027	30 April 2028	Previ acce when depe not y quali 4.4

Table • RabbitMQ series to choose between, checked 29 October 2026. Scroll sideways on small screens.

Operations and compliance

These are the facts that are discovered late and priced badly, so ask for them on day one.

- **Which environments exist, and does licensing or support have to cover all of them?** Non-production environments are rebuilt too, and they change both effort and the support scope.
- **What monitors the current broker, and what has to exist on day one for the new one?** Monitoring should be live before the first message arrives, not added after the first incident.
- **PCI, PII or other regimes that apply to message content.** They decide TLS requirements, audit trail and where the cluster can run.
- **Audit or change-control requirements that constrain the cutover window.** A change board that meets monthly sets the pace more than the engineering does.
- **Significant outages in the last two years, and whether the broker was the cause.** This separates a broker problem from an application problem you would otherwise import into the new platform.
- **Who runs it afterwards.** Decide whether your team, a support provider or a managed service operates the new cluster before the build, because it changes the monitoring and runbook scope.

How the answers change the plan

Each answer maps to a specific change in scope, so a short answer sheet is enough to see where the estimate will move.

ANSWER	WHAT CHANGES IN THE PLAN
Queue selectors in use	Per-application decision: a different client, or the filter moves into routing. Goes on the application change list.
XA or syncpoint transactions	Idempotent consumers or an outbox pattern, designed and tested first because it has the longest lead time.
Request-reply on temporary queues	Reply queues stay classic or use direct reply-to, so the target runs more than one queue type.
Message priorities in use	Choose queue type by priority need: classic 0 to 255, or strict 32-level priority on quorum queues from 4.3.
Deep durable backlog	A drain plan before cutover, and streams considered for the deepest flows.
Clients are not Java or not JMS	The JMS client does not apply. Each application needs a native AMQP client and a code change.
Both brokers must run side by side	A bridge design (shovel or relay), duplicate-tolerant consumers and a rollback date.
Several non-production environments	Build and test effort is repeated per environment, and support scope grows.
Regulated message content	TLS, audit trail and compliance review become gates on the cutover window.

Table • If the answer is this, the plan changes like this. Scroll sideways on small screens.

Plan in phases, with both brokers live

For an estate of any size, plan in phases with both brokers live rather than a single cutover. In the migrations we have run, teams that tried to move everything in one window either moved a fraction of it or spent the window rolling back. The applications, not the broker, set the pace, which is why the first sections of this worksheet are about the applications.

Once the new cluster takes traffic it inherits none of the old operational habits. The production readiness checklist covers what has to be true before it takes real traffic, and the queue type worksheet covers the choice that most often gets made by accident during a migration.

Pre-migration planning to post-migration optimization: the checklist by phase

Once the worksheet is answered, this is the checklist for the migration process, from planning to the point where the old broker is switched off. It is the outline of the step-by-step guide linked above. The phases are AceMQ practice for a staged move, not a RabbitMQ requirement, and each row ends with the evidence that says the phase is done.

PHASE	CHECKLIST	DONE WHEN
1. Pre-migration planning	Answer sections 1 to 5 of this worksheet. Inventory every queue, topic and client, record each application's tech stack and client library, and draw the data flows between them. List the queue selectors, XA transactions and temporary-queue request-reply flows. Agree the rollback plan and the date it closes. Agree the KPIs below and record a baseline from the source system.	Every application has an owner and a named client path.
2. Build the target	Provision RabbitMQ 4.4 and set the default queue type. Automate the build: keep the topology (virtual hosts, users, policies, queues) in a definitions file or infrastructure code so it can be rebuilt, and load it with <code>rabbitmqctl import_definitions</code> . Turn on TLS, give each application its own user with permissions limited to its own virtual host, and have the monitoring dashboard live before the first message.	The production readiness checklist passes.
3. Parallel run	Connect the two brokers with a shovel or an application relay. A shovel's default ack-mode, on-confirm, acknowledges the source only after the destination confirms, which protects against data loss. Make consumers tolerate duplicates. Move one application group at a time.	The first group runs on the new platform with KPIs at or better than baseline.
4. Cutover, application by application	Repoint the application and compare message counts at both brokers to confirm nothing was lost. Keep the rollback open while the bridge runs, and record the change for the change board.	Counts reconcile and the application's owner signs off.
5. Post-migration optimization	Switch the bridge off and decommission the source system after the agreed period. Review queue types and consumer prefetch against observed traffic, tune alerts from real KPI data, and hand the runbooks to your team or your managed service provider.	The old broker is retired and the new cluster is on a support plan.

Table • Pre-migration to post-migration checklist for a move to RabbitMQ. Scroll sideways on small screens.

Key performance indicators (KPIs) to record before the migration and compare after

A migration is only successful if it can be measured against what came before, so record these key performance indicators from the source system before the first application moves, along with usage patterns such as the peak hour, bursts and batch windows.

- **Message rates in and out**, average and peak, per queue or topic.
- **End-to-end latency** from publish to consume for the flows that have a deadline.
- **Queue depth and consumer lag** at the busiest hour, and how long a backlog takes to clear.
- **Redelivery and dead-letter counts**, which show the poison-message behaviour you are about to re-map.
- **Connection and authentication errors**, which are the first sign of a client that was not fully ported.
- **Incidents attributed to the broker** over the last two years, to compare with the first months on the new platform.

Compare the same figures after each application group moves, and again once the bridge is off. A flow that is slower or loses messages is the edge case to fix before the next group moves.

Migration strategies: big-bang, phased or parallel run

The migration approach decides how much of the estate is exposed at once, so choose it before the build, not after the first cutover.

- **Big-bang cutover.** Every application moves in one change window. It has the shortest bridge and the most that can go wrong at once, and it is the approach most likely to disrupt operations when a flow behaves differently on RabbitMQ.

- **Phased, with a bridge.** Application groups move one at a time while a shovel or relay carries messages between the brokers. This is the approach in sections 3 and 7.
- **Parallel run.** Both brokers carry the same flow for a period and the outputs are compared. It gives the best evidence and doubles the consumer and reconciliation work, so keep it for the flows where a wrong answer is expensive.

Whichever you choose, put it through the change management process early: application owners, the change board and whoever supports the platform afterwards need the dates and the rollback plan before the first group moves.

Compatibility, integration issues and data quality checks

Most defects found after a cutover were findable before it, in a test environment that runs the real workflow rather than a unit test of the client.

- **Compatibility.** Run each application's client library against the target version and queue type it will use in production. A client that connects is not yet a client that behaves the same.
- **Integration issues.** List every product, file transfer or scheduler that publishes to or reads from the source system, because each one needs its own plan and owner.
- **Data quality.** Compare a sample of message bodies, headers and properties at both brokers, including the oldest and the largest messages, to catch anything a conversion step dropped or changed.
- **Data loss.** Reconcile message counts in and out per queue at every cutover, using publisher confirms and consumer acknowledgements on the RabbitMQ side.
- **Workflow tests.** Replay the end-to-end business workflows that cross several queues, and include the failure paths: a poison message, a stopped consumer, a

broker restart.

Frequently asked questions

Does RabbitMQ support JMS?

Yes, through the RabbitMQ JMS client, which implements JMS 2.0 (and a JMS 3.0 edition) on top of the RabbitMQ Java client. It has documented limits: selectors for queues are not supported, local transactions do not guarantee atomicity and XA interfaces are not implemented. Topic selectors work with the RabbitMQ JMS Topic Exchange plugin enabled.

Does RabbitMQ support AMQP 1.0?

Yes, natively since RabbitMQ 4.0, with no plugin, on the same port as AMQP 0-9-1. It does not support transactions or resuming links, and it does not provide exactly-once delivery.

Which RabbitMQ version should a migration target?

RabbitMQ 4.4, the current series, released on 20 October 2026. The previous series is 4.3. Starting on an older series means a version upgrade is added to the migration project.

Can IBM MQ or ActiveMQ clients connect to RabbitMQ without code changes?

Generally not. RabbitMQ does not speak IBM MQ's native protocol or ActiveMQ's OpenWire. JMS applications can move to the RabbitMQ JMS client with configuration and some code changes, and others move to a native AMQP client. Discovery should specify the change per application.

What is the most common reason a broker migration overruns?

Application semantics that nobody documented: queue selectors, XA transactions, temporary-queue request-reply, or ordering assumptions. In our experience these, rather than the broker swap, move the estimate, which is why this worksheet starts with the applications.

What should a RabbitMQ migration checklist include?

An inventory of queues, topics and clients; the features that do not map cleanly (queue selectors, XA transactions, temporary-queue request-reply, priorities); a coexistence and rollback plan; the target RabbitMQ version, queue type and client path; operations and compliance requirements; and KPIs recorded before the move. The worksheet above covers each, and section 8 turns them into a phase-by-phase checklist.

How do you migrate from IBM MQ or ActiveMQ to RabbitMQ without losing messages?

Run both brokers in parallel and move one application group at a time. A shovel can bridge them when the source broker can be reached over AMQP 0.9.1 or AMQP 1.0, and its default on-confirm ack-mode acknowledges the source only after the destination confirms. Because RabbitMQ is at-least-once, consumers must tolerate duplicates, and message counts at both brokers should reconcile before the rollback closes.

Related