

— GUIDE · RABBITMQ

RabbitMQ support readiness self-assessment

These are the 24 questions AceMQ asks on every RabbitMQ support scoping call, with a score per answer and a table of score bands. Answer them yourself first and you will know whether you need a support contract, a health check, or a fix before either, without speaking to anyone.



By **Tyler Eastridge**, Head of Operations

LinkedIn · Updated October 29, 2026 · 12 min read · 10 sections

— HOW THE SCORE WORKS

Score the 19 questions in sections 2 to 5 from 0 to 2 each (38 points at most), then read the band: 31 to 38 means stable and wanting a safety net, 16 to 30 means gaps you already suspected, and 0 to 15 means mostly unknowns. The five questions in section 1 are not scored: any one of three triggers (an open incident, weekly incidents, or lost messages with no fix) moves you to fixing first, whatever your score. The bands are AceMQ's rule of thumb, not a benchmark against other estates. Checked against rabbitmq.com on 29 October 2026.

This page is for finding out where you stand. For choosing a support model and reading an SLA, see [choosing a RabbitMQ support model and SLA](#); for community against commercial support, see the [comparison](#).

How to score your answers

Give each question in sections 2 to 5 a 2 if you can answer it specifically and the answer is current, a 1 if you can answer it partly or only from memory, and a 0 if you

do not know. A 2 should survive one test: a second person on your team would give the same answer. The questions you cannot answer matter more than the ones you can, because each 0 is a place where an incident will start with a search for information instead of a fix.

Section 1 is not scored. It asks what is happening now, and the answer changes what you buy first, so it works as a set of triggers.

SCORE	WHAT IT MEANS	TEST
2	Specific and current	A second person on your team gives the same answer, and it is written down
1	Partial	You answer from memory, or for some clusters but not all, or it was true at the last review
0	Unknown or no	You would have to find out, or the honest answer is that nothing is in place

Table • Scoring rubric for sections 2 to 5. Scroll sideways on small screens.

What is happening now (not scored)

If any one of three answers below is true, fix the cluster before you buy support, because a support contract on an unstable cluster buys a faster response to the same recurring problem. Record the other two answers anyway: they set the response time you will need to ask for.

QUESTION	WHAT TO RECORD	TRIGGER
1. Do you have an open production incident involving the broker right now?	Yes or no, and the ticket reference	Yes
2. How often do incidents occur: daily, weekly, monthly, or rarely?	A frequency from the last twelve months, not a feeling	Daily or weekly
3. How long does a P1 typically last, in minutes, from detection to resolution?	The median from your last few incidents	No trigger. A long figure is the case for a faster response commitment.
4. Has an outage in the last year been traced to the messaging layer?	Yes or no, and the root cause if it was found	No trigger on its own
5. Have you lost messages, or had a service affected, because of a broker problem?	Yes or no, and whether the cause was found and fixed	Yes, and the cause is not fixed

Table • Section 1: triggers, not scores. Scroll sideways on small screens.

What you run (questions 6 to 10)

You can score a 2 here only if you can list every cluster, its version and where it runs, because support scope, patch exposure and failure tolerance all follow from that list. RabbitMQ 4.4 is the current series, with community support until 31 July 2027; the project's own table gives a different end date for each series, so the version answer is the one to get exactly right.

QUESTION	A SCORE OF 2 LOOKS LIKE	WHY IT MATTERS
6. Which RabbitMQ and Erlang versions are in production?	Exact versions on every node, and the end-of-support dates for those series	Support and CVE exposure follow the series. Community support for 4.2 ended 31 July 2026 and for 3.13 ended 30 September 2024; commercial support for 4.0 ended 30 September 2026.
7. How many clusters, and how many nodes in each?	A list: cluster, node count, hosts and failure domain	Three nodes survive one failure and five survive two. Even counts and two-node clusters do not follow that rule.
8. Are you on VMs, bare metal, containers or a managed service?	The platform named for each cluster	Failure modes, storage and restarts differ by platform, and so does who can fix what.
9. How many environments need covering: production only, or staging and QA too?	Every environment named, with its RabbitMQ version	A licence counts every environment where RabbitMQ runs, and the incidents that reach production often start in staging.
10. Open-source RabbitMQ, Tanzu RabbitMQ, or a vendor image such as Bitnami?	The distribution named, and who builds the image you pull	Patch cadence is set by whoever builds the image, so a support team needs to know which one it is.

Table • Section 2: what you run. Scroll sideways on small screens.

How it survives failure (questions 11 to 15)

This is the section that turns a routine failure into either a ten-minute event or a long night, and it is the one most often scored 0 or 1. A 2 on each question means a written, tested answer, not an intention. Two questions changed with RabbitMQ 4.x: classic queue mirroring was removed in 4.0, and 4.3 removed partition handling strategies along with Mnesia.

QUESTION	A SCORE OF 2 LOOKS LIKE	WHY IT MATTERS
11. Are you using quorum queues, classic queues, or still classic mirrored queues?	Quorum queues or streams for data that must survive a node loss; classic only for transient traffic; no mirrored queues	Mirroring was removed in RabbitMQ 4.0, so mirrored queues mean you are on 3.x with a migration to plan.
12. What is your partition handling set to?	On 4.2 and earlier, a strategy chosen on purpose with an odd node count. On 4.3 and later there is no setting, so the answer is your node count and failure domains.	RabbitMQ 4.3 removed the partition handling strategies. A stale cluster_partition_handling key left in the config does nothing.
13. Do you have a disaster recovery position, and has it been tested?	A second site or cluster, a written failover procedure and the date it was last tested	Quorum queues give high availability inside one cluster. They are not disaster recovery.
14. Are definitions exported on a schedule and stored off the node?	A scheduled export held away from the node, and a restore you have tried	Definitions are the topology: without them a rebuilt cluster is empty.
15. Could you rebuild the cluster from documentation if you lost it tonight?	A runbook that someone other than its author has followed	The first loss of a cluster is the worst time to find the steps are in one person's head.

Table • Section 3: how it survives failure. Scroll sideways on small screens.

What you can see (questions 16 to 19)

You can score a 2 here only if someone is paged for the right alerts and can tell which application is causing a problem. Monitoring that nobody reads scores 1 at best, and a dashboard that is not wired to a pager scores 0 for question 17.

QUESTION	A SCORE OF 2 LOOKS LIKE	WHY IT MATTERS
16. What monitors the broker, and who receives the alerts?	A named tool, a named recipient or rota, and alerts for node down, memory alarm and disk alarm	Memory and disk alarms block publishers, and they should reach a person before the application team notices.
17. Which alerts actually page someone, versus going to a dashboard nobody watches?	A short written list of paging alerts, each with an owner	An alert nobody is paged for is a log line.
18. Can you tell which application opened a given connection?	Applications set a client-provided connection name, and you can look one up	Most broker problems start in a client, and attributing them is the first step of a fix.
19. Do you have a ticketing system, and does it connect to your monitoring?	Alerts create tickets, and the ticket history for the broker is searchable	A support team works from the history, and a ticket without it starts from zero.

Table • Section 4: what you can see. Scroll sideways on small screens.

Who runs it (questions 20 to 24)

A 2 here means more than one person can operate the broker unsupervised and the cover you need is written down. The commonest low score is an inherited cluster, where the people who built it have moved on and the documentation did not.

QUESTION	A SCORE OF 2 LOOKS LIKE	WHY IT MATTERS
20. How many people are comfortable operating the broker unsupervised?	At least two, named	One person is a single point of failure that holidays and resignations find.
21. Is there written documentation for this deployment, and is it current?	A document reviewed since the last topology or version change	Documentation older than the cluster describes a cluster that no longer exists.
22. Did the people who built it still work here?	Yes, or the build is fully documented and handed over	Design intent leaves with the people, and an inherited cluster is the usual reason for a health check.
23. What are your business hours and time zone, and do you need cover outside them?	Hours and time zone stated, and the out-of-hours decision made and agreed	It sets whether you need 24/7 cover or a business-hours escalation route.
24. What response time would you need contractually for a production outage?	A number for each severity, agreed with the business that depends on the broker	Response and resolution are different promises, so ask for both numbers.

Table • Section 5: who runs it. Scroll sideways on small screens.

What to note about your applications before the first call

The 24 questions are about the broker, but most problems that reach it start in a client, so a support engineer will also ask about the applications around it. None of this is scored. Write it down once, keep it with the answers above, and attach it to your first support request: it saves the first round of questions during an incident.

NOTE DOWN	WHY IT HELPS
Producers: which applications publish, at what peak rate, and whether they use publisher confirms	Without confirms a producer cannot know a message reached the broker, so lost messages are hard to trace. The peak rate is the baseline for throughput questions.
Consumers: which applications consume, how long a message takes to process, the prefetch value, and whether they acknowledge manually or use automatic acknowledgement	A consumer on automatic acknowledgement can lose messages if it crashes mid-processing, and prefetch and processing time explain most backlogs.
Delivery handling: retry behaviour, how redelivery and duplicates are tolerated, and what happens to a message that keeps failing	RabbitMQ delivers at least once when consumers acknowledge, so a consumer has to cope with seeing a message twice. A retry loop with no limit is a common cause of queue growth.
Protocol and client library: AMQP 0-9-1, AMQP 1.0, MQTT or STOMP, and the library or framework in use (for example Spring AMQP on Java)	MQTT and STOMP run through plugins, and AMQP 1.0 has been native since RabbitMQ 4.0, so the protocol decides which settings and logs matter.
Topology: each exchange, its queues, bindings and routing keys, ideally from a definitions export	It shows how a message is routed and which queue types are in use, and it is the same export question 14 asks about.
Integrations: federation or shovel links to other clusters, and any system that reads from or writes to the broker	Links and integrations add failure modes that sit outside the cluster.
Recent errors: the client and broker error messages, with timestamps	Matching a client error to the broker log line at the same moment is often the fastest route to the cause.
Planned updates: any RabbitMQ, Erlang or client library update in the next quarter	Support is easier to scope around a known update, and an update in flight changes what to check first.

Table • Application details a support engineer asks for. Scroll sideways on small screens.

Reading your score

Add sections 2 to 5 for a total out of 38, check section 1 for triggers first, then read the band: a trigger overrides the score, 31 to 38 means stable, 16 to 30 means gaps, and 0 to 15 means mostly unknowns. The cut-offs sit at roughly 80% and 40% of the available points. They are AceMQ's rule of thumb, chosen to match how an engineer sorts a discovery call, and they are not benchmarked against other estates.

Example: a team scores 8 on what it runs, 4 on failure, 4 on visibility and 6 on team, which is 22 out of 38. That is the gaps band, and the weakest section, failure, is where a health check starts.

RESULT	WHAT IT MEANS	DO FIRST
Any section 1 trigger	Something needs fixing before support makes sense	Root cause first, then support from a cluster that behaves. If something is broken now, use emergency support; under a support contract a senior engineer is on a P1 within 15 minutes.
31 to 38	Stable, wanting a safety net	An onboarding assessment that documents the environment, then support. Support on an environment nobody has seen is a worse deal for both sides.
16 to 30	Gaps you already suspected	A health check that ranks the gaps from P1 to P3 and prices them. You are free to close them yourself.
0 to 15	Mostly unknowns, which is an inherited cluster in practice	Document the environment and run a health check before you choose a support model, because you cannot yet say whether the cluster is stable.

Table • Score bands and what to do first. Scroll sideways on small screens.

You may not need a support contract at all

If you scored 31 or more and nothing is broken, you may need someone to call on the day something unusual happens, which is a smaller purchase than a 24/7 contract. Say so on the first call: the point of the assessment is to find the smallest thing that covers your risk. Open-source RabbitMQ does not need a commercial licence to be supported, and the licensing question is separate from the support one.

Where to go with your result

Match the weakest section to the page that works it in detail. If question 6 was the uncomfortable one, the version and Erlang support matrix shows what your series still receives. If it was section 3, the production readiness checklist covers the same ground and you can work it yourself. If you could not answer the estate questions at all, start with the environment one-sheet. To talk it through, contact AceMQ support: enquiries get a reply within 1 hour.

Frequently asked questions

Do I need RabbitMQ support?

It depends on whether anyone on your team can restore the cluster and name its failure modes at 3am. If a section 1 trigger is true you need a fix first, and if you score 31 or more on the other 19 questions you may only need someone to call. Use the score bands on this page to decide before you speak to a vendor.

What should I know before I buy RabbitMQ support?

Your exact versions and end-of-support dates, how many clusters and nodes you run, how it survives a failure, what you monitor and who is paged, who can operate it, and the response time the business needs. Those are the 24 questions on this page, and a provider will ask all of them.

What is a RabbitMQ health check?

A senior engineer's review of a cluster against known failure patterns, ending in a written report with findings ranked from P1 to P3. It suits the gaps and unknowns bands, because it prices and prioritises what you do not yet know before you commit to a support contract.

Does a support contract fix an unstable cluster?

No. It gets a faster response to the same recurring problem. If incidents are weekly or an outage was never root-caused, find and fix the cause first, then start support from a cluster that behaves.

What should I tell a RabbitMQ support provider about my applications?

Which applications publish and consume, whether producers use publisher confirms, how consumers acknowledge and what prefetch they use, which protocol and client library they speak, how failed messages are retried, and any recent error messages with timestamps. It is not scored, but it shortens the first round of questions on a support request.

Is RabbitMQ partition handling still a setting on 4.x?

No, not from 4.3. RabbitMQ 4.3.0 removed Mnesia and with it the partition handling strategies, so the metadata store, quorum queues and streams all rely on Raft majority. On 4.2 and earlier you still choose a strategy, and the docs advise `pause_minority` with an odd number of nodes when in doubt.

Related