

— GUIDE · RABBITMQ

Message Broker Production Readiness Checklist: RabbitMQ, Kafka and Redis Quick Check

Six questions, asked once for each broker, tell you in about ten minutes where a mixed RabbitMQ, Kafka and Redis estate is weakest. Each answer is a yes or a no, and each no points to the part of that broker's full readiness checklist to run next.



By **Scott Sternloff**, Senior Enterprise Architect

LinkedIn • Updated October 29, 2026 • 8 min read • 9 sections

— THE QUICK CHECK IN ONE PARAGRAPH

An estate is production-ready when, for every broker, you can answer yes to six questions: it runs a release that still gets fixes, its replicas sit in separate failure domains, alerts fire before a limit blocks clients, a restore has been run and timed, no client uses a shared or default identity, and the next upgrade has a written path. Answer them in the tables below. A no on version or restore is the one to fix first, because neither has a quick way back once something fails.

How to use this quick check

Answer each question for each broker you run, yes or no, from what you can verify today, not from what the design document says. This page is the cross-broker triage. It does not replace the full checklists, which hold the commands, defaults and pass tests per broker: the Kafka production readiness checklist, the Redis production readiness checklist and the RabbitMQ production readiness checklist.

Each table cell is kept to the one fact that decides the answer, and each command in the last column is a read-only query of the running broker. RabbitMQ rows were checked against rabbitmq.com on 29 October 2026; Kafka rows follow the Apache Kafka 4.4 documentation and Redis rows the redis.io documentation, both as read for AceMQ's full checklists on 28 October 2026. Redis rows apply to Valkey as well.

Does every broker run a release that still gets fixes?

Yes only if each broker is on a release line its project or a vendor still patches. A broker past that line has no route to the next security fix, so this is the question to answer first.

BROKER	YES MEANS	WHERE TO LOOK
RabbitMQ	4.4 (community support to 31 July 2027) or 4.3 (community support to 31 January 2027), or an older series under a commercial or extended support contract.	rabbitmq-diagnostics server_version on every node
Kafka	Apache Kafka 4.4, 4.3 or 4.2: the project fixes only its last three releases. A ZooKeeper cluster has a dated plan to move to KRaft.	bin/kafka-topics.sh --version; the controller setup in server.properties
Redis or Valkey	A version that still receives fixes on the Redis end-of-life table.	INFO server, field redis_version

Table • Question 1: supported release, per broker. Scroll sideways on small screens.

Do replicas sit in separate failure domains?

Yes only if losing one host or one zone leaves a working majority and a copy of the data. Replicas that share a rack, host or zone behave like one replica on the day that domain fails.

BROKER	YES MEANS	WHERE TO LOOK
RabbitMQ	An odd node count (3 or 5, not 2), with replicated data on quorum queues or streams. Classic queue mirroring was removed in 4.0. Zones within one cloud region work with default settings.	rabbitmqctl cluster_status; rabbitmqctl list_queues name type
Kafka	Replication factor 3 and min.insync.replicas=2, broker.rack set on every broker, and 3 or 5 dedicated controllers.	kafka-topics.sh --describe; kafka-metadata-quorum.sh describe --status
Redis or Valkey	Sentinel: three Sentinels on independent hosts or zones. Cluster: at least three primaries, each with a replica. Replication is asynchronous, so a failover can still lose acknowledged writes.	SENTINEL masters or CLUSTER NODES; INFO replication

Table • Question 2: failure-domain layout, per broker. Scroll sideways on small screens.

Do alerts fire before a limit blocks clients?

Yes only if each broker alerts on the trend toward its limit, not only on the limit itself. All three brokers stop serving some clients when they hit a resource limit, and an alert at that point is an outage report.

Configure each alert with a threshold, a severity and an on-call owner, and put each threshold on a dashboard so the trend is visible before the alert fires. Metrics, dashboards and logs together are the observability the on-call engineer works from. RabbitMQ's HTTP API also exposes health checks, such as `/api/health/checks/local-alarms`, that a load balancer or monitoring probe can call in addition to the metrics.

BROKER	YES MEANS	WHERE TO LOOK
RabbitMQ	Metrics scraped from the rabbitmq_prometheus plugin. Alerts warn before the memory alarm (publishers block at 60% of RAM by default) and the disk alarm, whose low default needs raising for production.	Port 15692 on each node; rabbitmq-diagnostics check_local_alarms
Kafka	Alerts on under-replicated, under-min-ISR and offline partitions (all should be 0), exactly one active controller, sustained consumer lag growth and disk time-to-full.	Broker JMX metrics; kafka-consumer-groups.sh --describe
Redis or Valkey	Alerts on used_memory against maxmemory, rising evicted_keys, failed saves, master_link_status down and connected_clients against maxclients. The latency monitor is off by default.	INFO; CONFIG GET latency-monitor-threshold

Table • Question 3: monitoring and alerts, per broker. Scroll sideways on small screens.

Has a restore been run and timed?

Yes only if someone has restored from the backup or failed over to the recovery copy and written down how long it took. A backup that has never been restored is an assumption. It matters most where the broker holds the only copy of the data, as when Redis or Valkey is used as a primary database rather than a cache.

BROKER	YES MEANS	WHERE TO LOOK
RabbitMQ	Definitions (users, permissions, policies, queues, exchanges and the bindings that define message routing) are exported to JSON on a schedule and have been imported into a scratch cluster. Messages are not part of a routine backup: the docs advise against copying the message store from a running node.	rabbitmqctl export_definitions; the last import test and its date
Kafka	A failover to the disaster recovery cluster with consumers attached, with the recovery point measured, and topic configuration restorable from saved output.	kafka-configs.sh --describe output kept outside the cluster; the drill record
Redis or Valkey	An RDB snapshot leaves the machine every day and has been restored to a scratch instance with the expected key count.	Backup job logs; DBSIZE on the restored instance

Table • Question 4: backup and recovery rehearsal, per broker. Scroll sideways on small screens.

Does every client have its own identity, over an encrypted link?

Yes only if no application connects with a default or shared account and traffic that crosses a network you do not control is encrypted. A shared identity makes an incident impossible to trace and a leaked credential impossible to revoke cleanly.

Access control has to hold at every endpoint a client or an operator can reach: the client listener, any management or admin interface, and the links between brokers. Check each one, not only the endpoint the applications use.

BROKER	YES MEANS	WHERE TO LOOK
RabbitMQ	The default guest user is deleted, each application has its own user with permissions on its own virtual host, and client listeners use TLS.	rabbitmqctl list_users; rabbitmqctl list_permissions --vhost name
Kafka	Every listener, including the inter-broker one, uses SSL or SASL_SSL. An authorizer is set, allow.everyone.if.no.acl.found stays false and super.users is short.	Listener and authorizer settings in server.properties
Redis or Valkey	Not reachable from the internet, the open default user is closed, each application has an ACL user, and application users cannot run dangerous commands.	ACL LIST; firewall or security-group rules

Table • Question 5: access and identity, per broker. Scroll sideways on small screens.

Is the next upgrade written down?

Yes only if the next version is named and the route to it is written and tried on a copy. Each broker has one step that blocks a later upgrade when it is skipped, listed below. Each is a dependency: the later step cannot start until the earlier one is done.

BROKER	YES MEANS	WHERE TO LOOK
RabbitMQ	All feature flags are enabled after each rolling upgrade, because each series requires the previous series' flags (4.3 requires every 4.2 flag). From 3.13 with mirrored classic queues, the route to 4.x is a blue-green move to quorum queues.	rabbitmqctl list_feature_flags
Kafka	A ZooKeeper cluster is migrated to KRaft first, because only KRaft clusters can reach any 4.x release.	The KRaft migration plan, with dates
Redis or Valkey	Replicas are upgraded one after another and an upgraded replica is promoted with a manual failover, rehearsed on a copy.	The upgrade runbook and its last test

Table • Question 6: upgrade path, per broker. Scroll sideways on small screens.

Run the check as an audit and a deployment gate

The six questions work as a production readiness audit when they run on a schedule, and as a gate before a deployment or an upgrade. A workflow that holds up in practice has five steps.

- 1. Audit.** Answer all 18 cells (six questions for each of three brokers) once a quarter and before any change window. Record the answer, the owner and the date.
- 2. Give every threshold an owner.** A threshold from question 3 is only useful if it pages someone. Name the on-call engineer for each broker and check that the on-call runbook holds the commands from the tables above.
- 3. Gate the deployment.** Before you deploy, upgrade or reconfigure a broker, answer questions 1, 4 and 6 again for it. The change goes ahead only with a restore that has been rehearsed end-to-end and a written upgrade path.
- 4. Write the rollback before the change.** For RabbitMQ, enabling a feature flag cannot be undone and rules out downgrading the nodes, so after that step the

way back is a restore, not a rollback of the version.

5. **Automate the commands.** Automation of the read-only commands from the tables, as a scheduled job that stores the output with its date, means a person reviews what changed instead of collecting the data by hand.

What to do with the answers

Fix the noes in this order: question 1, then 4, then 2, then the rest. That order is AceMQ's practice, not a documented rule: an unsupported release and an untested restore are the two gaps with no quick recovery once something fails, and a shared failure domain is the next most likely cause of a full outage. For each no, run the matching group in that broker's full checklist.

Most teams can close one or two noes themselves. If the noes cluster on one broker nobody on the team owns, a health check reads that broker's configuration and returns a ranked fix list. AceMQ supports RabbitMQ, Kafka and Redis, and covering several technologies under one contract earns a multi-technology discount. AceMQ has 160+ customers and handles 40+ Sev 1 incidents a year, and we reply to quote requests within 1 hour.

Frequently asked questions

What should I check first in a mixed RabbitMQ, Kafka and Redis estate?

That each broker runs a release that still gets fixes, then that a restore has been run and timed. Those are the two gaps with no quick recovery once something fails. The six questions on this page cover both, plus failure domains, alerts, access and the upgrade path.

Which Kafka versions are supported now?

Apache Kafka 4.4, 4.3 and 4.2, the three releases the project maintains as of 28 October 2026. The project fixes only its last three releases, and only KRaft clusters can reach any 4.x release.

Which RabbitMQ versions get community fixes?

RabbitMQ 4.4 (community support to 31 July 2027) and 4.3 (community support to 31 January 2027), checked against rabbitmq.com on 29 October 2026. Older series get fixes only through commercial or extended support.

What is a message broker production readiness checklist?

A short list of yes or no questions that tell you whether a broker is safe to run in production: a supported release, replicas in separate failure domains, alerts that fire early, a restore that has been run, per-client access control and a written upgrade path. This page asks the same six questions of RabbitMQ, Kafka and Redis.

How often should the check be run?

Once a quarter and before every deployment or upgrade window. For a change, re-answer the support, restore and upgrade questions for the broker involved, and record the answers with the date.

Is this a replacement for a full readiness checklist?

No. It is a ten-minute triage that tells you which broker and which area to look at. Each no points to a group of checks in the full Kafka, Redis or RabbitMQ readiness checklist, which hold the commands and pass tests.

Related

