

RabbitMQ Troubleshooting Decision Tree

Use this page to troubleshoot RabbitMQ issues: find the symptom, run the one check listed under it, and follow the row that matches what you see. This page is a lookup for someone with a problem right now: it tells you what to check and what the answer means. For the full command sequence once you know what you are dealing with, use the outage runbook. Commands checked against the RabbitMQ 4.4 CLI manuals and rabbitmq.com on 29 October 2026.



By **Scott Sternloff**, Senior Enterprise Architect

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

— RUN TWO COMMANDS BEFORE FORMING A THEORY

`rabbitmq-diagnostics alarms` tells you whether the broker has stopped accepting publishes, and `rabbitmqctl list_queues -p <vhost> name messages consumers` tells you whether anything is reading. A queue backing up with zero consumers is the single most common "RabbitMQ is broken" ticket we receive, and the fix is almost always in the consumer application. Then find your symptom:

SYMPTOM	FIRST CHECK
Queue growing, not draining	<code>rabbitmqctl list_queues</code> with consumers and unacked
Memory alarm, publishers blocked	<code>rabbitmq-diagnostics memory_breakdown</code>
Disk alarm	<code>rabbitmq-diagnostics status</code> , free disk against the limit
Partition or unreachable node	<code>rabbitmq-diagnostics cluster_status</code>
Node will not start or rejoin	<code>rabbitmq-diagnostics log_tail</code>
Clients cannot connect	<code>rabbitmq-diagnostics listeners</code>
Federation or shovel link down	<code>rabbitmqctl federation_status --only-down</code>
Messages published, never arrive	<code>rabbitmqctl list_bindings</code>
Pods stuck on Kubernetes	<code>kubectl describe pod</code>

What this page is for, and what it assumes

This is a symptom-to-check lookup, not a runbook: each section gives one first check and a table of what the result means. The outage runbook has the full, ordered command sequences for disk and memory alarms, partitions, quorum loss and datacenter recovery, and the failure patterns guide explains why each pattern happens. Use this page to decide which of those you need.

It assumes RabbitMQ 4.x. From 4.3.0 the metadata store is Khepri only and the old partition-handling strategies are gone, so the partition branches below are written for Raft; where a 4.2 or older cluster behaves differently, the row says so. The symptoms are the ones that actually reach AceMQ support, drawn from our tickets rather than a generic list.

A queue is growing and not draining

Check the consumer count on that queue before anything else: it decides whether you are looking at the broker or the application.

```
rabbitmqctl list_queues -p <vhost> name type messages messages_unacknowledged cons
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
Consumers is 0	Nothing is reading. The consumer application is down, crashed, or failed to reconnect after a restart.	Check the application, not the cluster. The broker is doing its job.
Consumers above 0, unacked high and not moving	Consumers hold messages they never acknowledge, usually stuck on a downstream call with no timeout, or a code path that never acks.	Restart the consumers to release the messages, then add the missing timeout or ack. Quorum queues return them after the consumer timeout (30 minutes by default).
Unacked near 0, messages still climbing	Consumers keep up per message but there are not enough of them, or prefetch is too low to keep each one busy.	Add consumers and raise prefetch before touching the broker. The docs put the usual throughput sweet spot at 100 to 300.
Consumers keep disconnecting and reconnecting	A load balancer or firewall idle timeout shorter than the heartbeat, or a channel closed with PRECONDITION_FAILED by the delivery acknowledgement timeout.	Set the heartbeat below every idle timeout on the path, and look in the client log for the timeout error before raising <code>consumer_timeout</code> .

Table • Queue backlog: read the consumers and unacked columns. Scroll sideways on small screens.

A memory alarm has fired and publishing has stopped

Confirm the alarm and find what is holding the memory: the breakdown tells you whether this is a backlog, a connection problem or a sizing problem.

```
rabbitmq-diagnostics alarms
rabbitmq-diagnostics memory_breakdown --unit "MB"
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
<code>queue_procs</code> or quorum queue memory dominates	A backlog is the cause and the alarm is the symptom.	Work section 1. The alarm clears when the queues drain.
<code>connection_*</code> or <code>binary</code> dominates	Many connections, large messages, or a client opening a connection per operation.	Compare <code>rabbitmqctl list_connections</code> name <code>user</code> against what you expect and find the client that churns.
Nothing dominates and the limit looks low	The watermark was lowered below the 0.6 default, or the node is sized too small for the workload.	Check the limit in <code>rabbitmq-diagnostics status</code> . The docs recommend 0.4 to 0.7 and at least 30% left to the OS; going above 0.7 needs memory monitoring in place, so resize before you raise it.
Memory climbs steadily with no queue growth	An unbounded queue nobody is watching, or quorum queue write-ahead log memory.	Bound the queues. On quorum-queue nodes the docs say to allocate at least three times the WAL size limit, 512 MiB by default.

Table • Memory alarm: read the largest categories in the breakdown. Scroll sideways on small screens.

A disk alarm has fired

Check free space against the configured limit, on the volume that actually holds the data directory.

```
rabbitmq-diagnostics alarms
rabbitmq-diagnostics status | grep -A 2 -B 2 "Node data directory"
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
The disk is genuinely full	A backlog on quorum queues or streams, which keep everything on disk and may not reclaim space from consumed messages straight away.	Drain or drop the largest queue, then grow the volume. The docs say to overprovision disk when in doubt.
Free space looks fine but the alarm persists	The limit is higher than you think, or the data directory is on a different volume from the one you checked.	Confirm the data directory path from the command above before changing anything.

Table • Disk alarm. Scroll sideways on small screens.

A node is unreachable or the cluster has partitioned

Establish which side holds the majority before touching anything: on Raft, the majority side keeps working and the minority side waits.

```
rabbitmq-diagnostics cluster_status
rabbitmq-queues quorum_status <queue>
rabbitmq-diagnostics check_if_node_is_quorum_critical
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
One node missing from the running nodes, the rest reachable	Working as designed. The majority elected new leaders within seconds; quorum queue deliveries pause only on the cut-off side.	Restore the network. The returning member catches up from the leader and cannot become leader until it has. Do not reset it.
No side holds a majority	Publishes, acknowledgements and metadata changes cannot make progress; they wait for a majority or time out.	Bring nodes back rather than resetting any of them. If the only way forward involves deleting a node's data, stop and escalate.
The isolated node was a VM that was suspended or migrated	A suspend/resume partition: the node does not think it failed, its peers do.	Treated like any single-node partition; short suspensions are handled. Check the hypervisor's migration and snapshot schedule.
Partitions recur on a cluster stretched across regions or zones	The topology, not a fault. Inter-node traffic does not tolerate high latency or jitter well.	Run separate clusters joined by federation or shovel for cross-site traffic.
4.2 or older, with <code>autoheal</code> or <code>ignore</code>	Both sides may have accepted writes, so there is divergence to reconcile.	Keep the majority side with the more complete data, restart the other side's nodes so they rejoin, then switch to <code>pause_minority</code> .

Table • Partitions on RabbitMQ 4.3 and later (Khepri and quorum queues). Scroll sideways on small screens.

A node will not start or will not rejoin

Read the log before restarting anything a second time: the reason is almost always in the first error.

```
rabbitmq-diagnostics log_tail -N 200
rabbitmq-diagnostics check_running
rabbitmq-diagnostics cluster_status
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
Cookie or authentication errors	The shared Erlang cookie differs between nodes or CLI and node.	Compare <code>rabbitmq-diagnostics erlang_cookie_hash</code> on each node, then fix the file, its owner and its permissions.
<code>epmd</code> or <code>nxdomain</code> errors	Name resolution. On Kubernetes, the operator docs point to CoreDNS caching; we also see it on IPv6-only clusters where Erlang was not configured for IPv6.	Fix DNS or the node's hostname resolution, not RabbitMQ. See section 9 for the Kubernetes case.
The node is on a different RabbitMQ version from the rest	It is out of step mid-upgrade.	Bring its software to the majority version and let it rejoin. Do not roll the cluster back to meet it: the upgrade guide says downgrades are not supported.
File descriptor or <code>emfile</code> errors	The open-file limit is too low. Connections, queues and quorum queue segment files all count against it.	Raise the limit for the service. The docs ask for at least 50,000, or twice the 95th-percentile connection count plus the number of queues.

Table • Node start and rejoin failures. Scroll sideways on small screens.

Clients cannot connect or fail authentication

Establish whether it is the listener, TLS or credentials, in that order.

```
rabbitmq-diagnostics listeners
rabbitmq-diagnostics check_port_connectivity
rabbitmq-diagnostics check_certificate_expiration
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
Failures on 5671 that mention certificates	The certificate name does not match the hostname clients dial, it has expired, or a load balancer terminates TLS while clients expect the broker's certificate.	Test with <code>openssl s_client -connect <host>;5671 -servername <host>;</code> using the name the client uses.
Federation, shovel or LDAP TLS fails after an Erlang upgrade	From Erlang 26, TLS client peer verification is on by default, so those links fail without a client certificate and key.	Configure the certificate pair, or disable peer verification deliberately.
Authentication fails for one application only	Its user has no permissions on the vhost it opens, even though the password is right.	Check <code>rabbitmqctl list_user_permissions <user>;</code> , not just the password.
Connection refused right after a restart	The node is still booting, or an alarm is blocking.	Run <code>rabbitmq-diagnostics alarms</code> before assuming the network.

Table • Connection failures. Scroll sideways on small screens.

Federation or shovel links are down

List the links and their state, and compare against a written list of what should exist.

```
rabbitmqctl federation_status --only-down
rabbitmqctl shovel_status
rabbitmqctl list_parameters -p <vhost>
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
Every link restarted at once	In our experience, expected after a cluster membership change such as adding or replacing a node.	The only question is whether they all came back, which needs a list of what should exist.
One link will not establish	Credentials, the network path to the upstream, or an upstream vhost that no longer exists. These failures are usually upstream-side and look like downstream problems.	Check the upstream URI and user, test the path, and confirm the vhost. For TLS links, see the Erlang 26 row in section 6.

Table • Federation and shovel links. Scroll sideways on small screens.

Messages are published but never arrive

Confirm the message is being routed at all: an exchange with no matching binding drops messages without an error by default.

```
rabbitmqctl list_exchanges -p <vhost> name type
rabbitmqctl list_bindings -p <vhost>
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
No binding matches the routing key	The message is unroutable. With <code>mandatory</code> false, the default, it is discarded unless an alternate exchange is set.	Add an alternate exchange so unroutable messages are visible, and watch <code>rabbitmq_global_messages_unroutable_</code>
Bindings look right but the wrong consumer gets messages	Topic wildcard overlap: two bindings match and both queues get a copy.	Tighten the binding patterns.
Messages turn up in the dead-letter queue	Something rejected or expired them: TTL, a length limit, a reject without requeue, or the quorum queue delivery limit (20 by default from 4.0).	Read the <code>x-death</code> header on a dead message; it records the reason and the queue.

Table • Lost or misrouted messages. Scroll sideways on small screens.

Pods will not start or will not terminate on Kubernetes

Read the pod events first: most causes are outside RabbitMQ.

```
kubectl describe pod <pod> -n <namespace>
kubectl logs <pod> -n <namespace>
kubectl describe rmq <cluster> -n <namespace>
```

IF YOU SEE	IT USUALLY MEANS	DO THIS
<code>ImagePullBackOff</code>	Registry credentials.	The secret must exist in the pod's namespace and be listed under <code>imagePullSecrets</code> in the <code>RabbitmqCluster</code> spec.
Pods stuck terminating	The operator's preStop hook is holding a pod that is quorum critical, usually because of a leftover quorum queue.	Check quorum status first. Force-delete only once you know losing that member's data is acceptable.
Pods restart with an <code>epmd</code> <code>nxdomain</code> error	Nodes resolve peer hostnames at boot and the DNS cache is stale.	The operator docs suggest lowering the CoreDNS cache from 30 seconds to 5 to 10.
Container fails under an enforced non-root policy	Security context settings do not match the image.	Set <code>runAsUser</code> , <code>runAsGroup</code> and <code>fsGroup</code> consistently; <code>fsGroup</code> decides whether the broker can write its volume.

Table • Kubernetes pod problems. Scroll sideways on small screens.

Troubleshoot RabbitMQ with the management UI, log files and metrics

The sections above start from a symptom. When none of them explains the issue, these are the places to gather evidence before you change anything. Each answers

a different question, and none replaces the first two commands at the top of the page.

SOURCE	USE IT FOR	HOW TO GET TO IT
RabbitMQ management UI	Seeing message rates globally and per queue, exchange or channel, and confirming whether a backlog is growing or flat.	Enable the plugin with <code>rabbitmq-plugins enable rabbitmq_management</code> (no restart needed), then open <code>http://&lt;node-hostname>;15672/</code> . Run <code>rabbitmq-plugins list</code> to verify it is on. Do not alert from it: the monitoring guide says the management plugin costs more memory and keeps only hours of history.
Log files	The first error behind a node that will not start, a rejected login or an alarm.	<code>rabbitmq-diagnostics -q log_location</code> prints the path. The default level is <code>info</code> . <code>rabbitmqctl set_log_level debug</code> raises it on a running node but does not survive a restart, so set it back afterwards.
Client connection events	Telling a network problem from an authentication problem.	Nodes log key connection lifecycle events, but only once a TCP connection is established and the peer has sent at least one byte. If nothing is logged for a client, its connection never reached the node: look at the network path and the listener on port 5672 or 5671.
Packet capture	Cases where the logs show nothing and you need to see what is on the wire.	The networking troubleshooting guide names <code>tcpdump</code> and Wireshark, and both understand every protocol RabbitMQ supports. Capture on the node and on the client side and compare.
Prometheus metrics	Trends over days and an alert before the next incident.	Scrape the <code>rabbitmq_prometheus</code> plugin on port 15692. The alert rules pack has the rules.

Table • Evidence sources and what each is good for. Scroll sideways on small screens.

When to stop and escalate

Escalate when the next step is destructive and untested. Three situations are worth someone else's eyes rather than another hour of yours: no side of the cluster has a majority and the way forward involves deleting a node's data, a node will not rejoin after an upgrade, and a 4.2-or-older partition where both sides accepted writes and you have to choose which data survives. All three get worse with improvisation.

Under an AceMQ support contract a senior engineer is on a P1 within 15 minutes, 24/7. Without one, the troubleshooting service starts an emergency engagement, and if the cause turns out to be the consumer application rather than the broker, we will tell you that too.

Frequently asked questions

Why is my RabbitMQ queue not draining?

Check the consumer count on that queue first. Zero consumers means nothing is reading and the problem is in the consumer application. Consumers present with a high, static unacknowledged count means they are holding messages they never acknowledge, usually stuck on a downstream call with no timeout. A near-zero unacknowledged count with a growing queue means there are not enough consumers or prefetch is too low.

What causes a RabbitMQ memory alarm?

Run `rabbitmq-diagnostics memory_breakdown`. If queue memory dominates, a backlog is the cause and the alarm clears when it drains. If connection or binary memory dominates, look for too many connections or large messages. If nothing dominates, check the watermark: the default is 0.6 of available memory, and the docs recommend staying between 0.4 and 0.7.

What should I do about a RabbitMQ network partition on 4.x?

On 4.3 and later there is no partition-handling strategy to choose: Khepri, quorum queues and streams all use Raft, so the side with a majority keeps working and the minority waits. Find the majority with `rabbitmq-diagnostics cluster_status`, restore the network, and let returning nodes catch up. Do not reset nodes to speed it up.

Why are messages published to RabbitMQ never delivered?

Usually because no binding matches the routing key, so the message is unroutable and silently discarded unless the publisher sets mandatory or the exchange has an alternate exchange. Check bindings with `rabbitmqctl list_bindings`, and add an alternate exchange so unroutable messages become visible.

Where are the RabbitMQ log files, and how do I get more detail?

Run `rabbitmq-diagnostics -q log_location` on the node to print the path; the management UI also shows it on the node page. The default log level is info. To get debug output from a running node use `rabbitmqctl set_log_level debug`, remembering that the change is transient and is lost on restart.

How do I troubleshoot RabbitMQ client connection problems?

Run `rabbitmq-diagnostics listeners` to confirm the port is open, then read the node log: it records client connection lifecycle events, but only for connections that completed a TCP handshake and sent data. No log entry means the connection never reached the node, so check the network. A logged authentication failure points to the user, password or vhost permissions instead. If the logs stay silent, capture traffic with `tcpdump` or Wireshark.

Related