

— GUIDE · RABBITMQ

RabbitMQ Environment Documentation Template: the One-Sheet

A RabbitMQ environment one-sheet is a single page, 52 fields across eight sections, that lets an engineer who has never seen your cluster reason about it before the first incident rather than during one. Each field below has an example of the detail wanted and the command or place to get it. Fill in one sheet per cluster.



By **Scott Sternloff**, Senior Enterprise Architect

LinkedIn · Updated October 29, 2026 · 10 min read · 13 sections

— WHAT TO PUT ON THE SHEET

Fill in eight sections: identity, versions, topology, configuration, security, clients, message flows and operations. The fields that matter most in an incident are the **exact RabbitMQ and Erlang versions**, the **metadata store** (Khepri, or Mnesia on 4.2 and earlier), **queue types in use**, **plugins and policies**, **client libraries with prefetch and heartbeat**, and **who is on call**. Record shapes and locations, never secrets.

Checked against rabbitmq.com on 29 October 2026. RabbitMQ 4.4 is current; the previous series is 4.3.

What the one-sheet is for, and where it came from

The one-sheet exists so that the first incident on a cluster is not also the first time anyone has looked at it. For the long-form version of what a healthy production cluster looks like, use the production readiness checklist; this page is only the record of what your cluster actually is.

The format is adapted from a one-page environment document that a customer kept in their own wiki, headed with its purpose: to level-set when working with AceMQ Support. It answered most of the questions we would otherwise ask over the first few tickets, which is why we generalised it. When we take over support of a cluster we did not build, onboarding starts with a health check that establishes the same facts: version, topology, queue types, known faults and patch position.

Collect the facts with these commands first

Run these on a node, as the RabbitMQ user, and paste the output into your working copy; most of the sheet is then a transcription exercise. Every command below is in the RabbitMQ CLI documentation.

```
rabbitmq-diagnostics server_version
rabbitmq-diagnostics erlang_version
rabbitmq-diagnostics status
rabbitmq-diagnostics environment
rabbitmq-diagnostics cluster_status
rabbitmq-diagnostics listeners
rabbitmq-diagnostics check_certificate_expiration
rabbitmq-diagnostics list_deprecated_features
rabbitmq-plugins list
rabbitmqctl list_feature_flags
rabbitmqctl list_vhosts name description tags default_queue_type
rabbitmqctl list_policies
rabbitmqctl list_operator_policies
rabbitmqctl list_queues name type messages
rabbitmqctl export_definitions /path/to/definitions.json
```

The definitions export contains users and permissions, so store it as a secret, and read the `environment` output before pasting it, because the effective configuration can include credentials. On large estates `list_queues` can be slow; run it with a limited column set, as above.

Identity: so a ticket can be matched to a cluster without asking

Record who owns the cluster and what breaks when it stops, because that decides how fast anyone has to move.

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
Environment name	prod-eu, uat, dr	Your own naming	
Business purpose	Order events for the payments platform	Ask the owning team	
Owning team and on-call rota	Platform Engineering, the pager schedule named 'messaging'	Your on-call tool	
Blast radius if it stops	Checkout halts and N orders per hour queue up	Ask the application owners	

Table • Identity fields. Scroll sideways on small screens.

Versions: the first thing anyone asks, and the field most often out of date

Record exact versions from the running nodes, not from the install ticket, and compare them with the support matrix.

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
RabbitMQ version	4.3.6 (or 3.13.7 on an estate that has not upgraded)	rabbitmq-diagnostics server_version	
Erlang/OTP version	28.x for 4.3.6 and 4.4.0; 26.2.x for a 3.13 estate, which is past Erlang 26 end of life	rabbitmq-diagnostics erlang_version, on every node	
Distribution	Open source, Tanzu RabbitMQ, or a vendor or Bitnami image	Package or image source	
Install method	RPM, Debian package, OCI image, Kubernetes Cluster Operator	Your deployment tooling	
Container image and tag (Docker or Kubernetes only)	rabbitmq:4.4-management, pinned to a version tag rather than latest	Your Dockerfile, Compose file or Helm values	
Last upgraded	March 2026, from 3.12.14	Change record	
Support status of this series	Community support ended, or ends on a stated date	The version support matrix	

Table • Versions fields. Scroll sideways on small screens.

Topology: what survives a failure and what does not

Record where the nodes sit and which metadata store the cluster uses, because partition behaviour depends on it.

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
Node count and names	3 nodes: rmq-prod-01, 02, 03	rabbitmq-diagnostics cluster_status	
Placement	One per availability zone, eu-west-1a, 1b, 1c	Your infrastructure inventory	
Platform	EC2 instances, or GKE with the Cluster Operator	Your deployment tooling	
Resources per node	8 vCPU, 32 GB RAM, 250 GB SSD volume	Cloud console or Kubernetes resource limits	
Metadata store	Khepri. On 4.3 and later it is the only store, so there is no partition handling setting to record. On 4.2 and earlier, state Khepri or Mnesia	khepri_db in rabbitmqctl list_feature_flags (4.0 to 4.2)	
Partition handling (Mnesia clusters on 4.2 and earlier only)	cluster_partition_handling = pause_minority	rabbitmq.conf	
Load balancer	A named appliance, TLS terminated at the balancer or passed through	Network team	
Ports exposed and to whom	5672 and 5671 for AMQP, 15672 for management, 15692 for Prometheus; 4369 and 25672 between nodes only	rabbitmq-diagnostics listeners, firewall rules	

Table • Topology fields. Scroll sideways on small screens.

Configuration that matters: the settings that turn up in nearly every incident

Record anything that differs from the defaults, and say which defaults you rely on.

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
Configuration files in use	rabbitmq.conf and advanced.config with their paths, one line per setting that differs from the defaults	rabbitmq-diagnostics status (Config files section), rabbitmq-diagnostics environment	
Environment variables set	RABBITMQ_NODENAME and RABBITMQ_CONFIG_FILE, set in rabbitmq-env.conf or the service unit	rabbitmq-env.conf, the service unit or container spec; a shell variable overrides the file	
Memory high watermark	0.6 relative (the default), on a 32 GB node	rabbitmq.conf (vm_memory_high_watermark)	
Disk free limit	10 GB absolute (the default is 50 MB)	rabbitmq.conf (disk_free_limit)	
Open file limit for the RabbitMQ user	At least 50,000, more on busy clusters	The service unit or container limits	
Default queue type	quorum, set for the node or per virtual host	rabbitmq.conf (default_queue_type), rabbitmqctl list_vhosts name default_queue_type	
Queue types in use	Quorum for orders, classic exclusive for RPC replies, one stream for audit	rabbitmqctl list_queues name type	
Virtual hosts and what uses them	/ for payments, /reporting for the warehouse feed	rabbitmqctl list_vhosts name description tags default_queue_type	

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
Plugins enabled	management, prometheus, shovel, federation	rabbitmq-plugins list	
Policies and operator policies	Dead letter exchange on the retry queues; no classic mirroring policies (mirroring was removed in 4.0)	rabbitmqctl list_policies, list_operator_policies	
Feature flags	All stable flags enabled, last checked after the March upgrade	rabbitmqctl list_feature_flags	
Deprecated features in use	None reported, or a named list	rabbitmq-diagnostics list_deprecated_features	

Table • Configuration that matters fields. Scroll sideways on small screens.

Security: who can reach the cluster and how it proves who they are

Record the shape of authentication and certificates, never the secrets themselves.

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
Authentication backend	Internal users, LDAP, or OAuth 2	rabbitmq.conf	
TLS listeners and certificate expiry	5671 with a certificate expiring on a stated date	rabbitmq-diagnostics check_certificate_expiration	
Management UI and API exposure	Internal network only, behind the load balancer	Firewall rules and listeners	
Erlang cookie handling	Provisioned by automation, stored in the secrets manager (record where, not the value)	Your secrets tooling	

Table • Security fields. Scroll sideways on small screens.

Clients: where most throughput problems live

Record library versions, because they decide behaviour on reconnect, heartbeat and prefetch.

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
Languages and libraries	Java (amqp-client), Python (pika), Go (amqp091-go), each with its version	Application dependency manifests	
Producer count and pattern	12 services, publisher confirms on	Application owners	
Consumer count and prefetch	20 consumers, prefetch 10, manual acknowledgement	Application config, management UI channels	
Heartbeat	30 seconds (the value RabbitMQ suggests is 60)	Client config	
Connection pattern	One connection per service, one channel per thread	Application owners	

Table • Clients fields. Scroll sideways on small screens.

Message flows: lets an engineer reason about a backlog without reverse-engineering it live

Record the shape of normal traffic, so an abnormal queue is obvious.

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
Peak and average throughput	1,200 msg/s average, 9,000 at peak	Prometheus or the management UI	
Typical payload size	4 KB, an occasional 900 KB batch	Application owners	
Busiest queues	orders.inbound, payments.retry	rabbitmqctl list_queues name messages	
Dead lettering	orders.dlq, drained manually, weekly	rabbitmqctl list_policies, queue arguments	
Federation or shovel links	Upstream from prod-us to prod-eu on the audit exchange	Runtime parameters in the management UI	

Table • Message flows fields. Scroll sideways on small screens.

Operations: what you can see, and what happens when something goes wrong

Record monitoring, backups and the change window, because these decide how an incident is handled.

FIELD	EXAMPLE OF THE DETAIL WANTED	HOW TO GET IT	YOURS
Monitoring	Prometheus plugin scraped every 30 seconds into a dashboard tool	Your monitoring config	
Alerts that page	Queue growth rate, node down, memory alarm approaching	Your alerting config	
Definitions backup	Nightly export to object storage with a stated retention	rabbitmqctl export_definitions <file>	
DR position	Warm standby in a second region, federation-fed, with a stated RPO	Your DR runbook	
Change window	Sundays 02:00 to 05:00 UTC, change-board approval required	Change process	
Support contact and contract	Provider, plan and escalation route	Your support agreement	
Known issues	Node 03 has logged busy_dist_port warnings since May	Your incident log	

Table • Operations fields. Scroll sideways on small screens.

How to fill in the version, Erlang and metadata-store fields

Record versions per node, because mixed versions mean an upgrade is in progress or has stalled. Compare the result with the current compatibility matrix: RabbitMQ 4.4.0 runs on Erlang 27.0 to 29.x, 4.3.6 runs on 27.0 to 28.x, and Erlang 26 has reached end of life.

For the metadata store, the answer depends on the series. Khepri became the default store for new clusters in RabbitMQ 4.2.0, an existing cluster upgraded to 4.2 keeps Mnesia until Khepri is explicitly enabled, and from 4.3.0 Mnesia and the partition handling strategies were removed. If the sheet says 4.3 or later, leave the partition handling row empty. If it says 4.2 or earlier on Mnesia, record the `cluster_partition_handling` value, because it decides what happens to data during a split.

For feature flags, record that the flags were checked and when. Enable all stable flags after every completed upgrade, and treat a flag that cannot be enabled as a finding to record under known issues.

Configuration files, environment variables and containers

Record where each setting lives, because the same value can be set in three places and the one that wins decides what the node does. The RabbitMQ configuration documentation describes four files and one variable layer:

- **rabbitmq.conf** is the main configuration file, in a simple key-value format. Record its path and every line that differs from the defaults.
- **advanced.config** uses Erlang terms for the settings `rabbitmq.conf` cannot express, and is merged with it. Record whether the file exists and why.
- **rabbitmq-env.conf** sets environment variables for the node, such as the node name, without the `RABBITMQ_` prefix. An environment variable set in the shell overrides the same variable in this file.
- **rabbitmq.config** is the legacy format from before 3.7.0. It is deprecated but still supported, so an old estate may still carry one; record it and plan its replacement.

Changes to `rabbitmq.conf` and `advanced.config` take effect after a node restart, so also record when each node was last restarted, and verify the running values with `rabbitmq-diagnostics environment` afterwards. For a Docker container, the official

image documentation recommends supplying custom configuration as a `rabbitmq.conf` file mounted at `/etc/rabbitmq/rabbitmq.conf`, and plugins through a Dockerfile step or a mounted `enabled_plugins` file. Record the image tag, the path of the mounted configuration file and where the Erlang cookie is supplied, not its value.

Keeping the sheet true

A one-sheet that is eighteen months old is worse than none, because people trust it. We recommend a date at the top, a review after every upgrade and every change to topology, and keeping the sheet next to the runbook rather than in a wiki nobody opens.

Create the sheet from this template and keep it where it has a change history, such as version control or a document with tracked edits, so you can see who modified a field and when. Name one owner, limit who can overwrite it, and publish it to everyone on call so they can access it during an incident. After each change, verify the fields against the live nodes, using the RabbitMQ management UI or the commands above to connect to a node and read what it reports, and specify the date you verified them.

Do not write passwords, tokens, the Erlang cookie value or the definitions file into the sheet itself. Record where they are held. If you would rather have an engineer produce the first version with you, ask for it as part of a health check or an onboarding; the scope and terms are agreed in the engagement, not here.

Frequently asked questions

What should a RabbitMQ environment document contain?

Exact RabbitMQ and Erlang/OTP versions, node count and placement, the metadata store, queue types and virtual hosts, enabled plugins, policies and feature flags, client libraries with prefetch and heartbeat settings, normal message rates, monitoring and alerts, definitions backup, and who is on call. Record locations of secrets, never the secrets.

Does RabbitMQ 4.x still need a partition handling setting?

Not from RabbitMQ 4.3.0. Mnesia and the partition handling strategies were removed, and Khepri is the only metadata store. On 4.2 and earlier, a cluster still on Mnesia has a `cluster_partition_handling` value that should be recorded.

How do I find out which RabbitMQ and Erlang versions a cluster is running?

Run `rabbitmq-diagnostics server_version` and `rabbitmq-diagnostics erlang_version` against each node, and `rabbitmq-diagnostics cluster_status` for the node list. Check them against the Erlang compatibility matrix on rabbitmq.com.

How often should the environment one-sheet be updated?

After every upgrade, every topology change and every change to queue types, plugins or policies. Put the date at the top and review it at least when you review the runbook.

Does a definitions export replace the one-sheet?

No. A definitions export captures users, virtual hosts, queues, exchanges, bindings, policies and parameters. It does not capture node placement, client libraries, message rates, alerting, the change window or ownership, which are the parts an on-call engineer needs first.

Where do I find which RabbitMQ configuration file a node is using?

Run `rabbitmq-diagnostics status` and read the Config files section, or check the top of the node's boot log. `rabbitmq-diagnostics environment` shows the effective merged configuration. Record the paths of `rabbitmq.conf`, `advanced.config` and `rabbitmq-env.conf` on the sheet.

How do I document a RabbitMQ Docker container deployment?

Record the image and its version tag, the `rabbitmq.conf` file you mount at `/etc/rabbitmq/rabbitmq.conf`, the plugins enabled, the environment variables set on the container, and where the Erlang cookie comes from. Keep the Dockerfile or Compose file next to the sheet so the two stay in step.

Related