Sidekiq Health Audit — Sample Report

> **SAMPLE — SYNTHETIC DATA.** “Ledgerly” is a fictional company. Every number below was invented to show what a real report looks like. Nothing here comes from a real customer.

Client: Ledgerly (fictional B2B invoicing SaaS) · **Tier:** Quick Scan
Snapshot captured: 2026-10-14 14:02 UTC · **Sidekiq** 7.2.4 · **Ruby** 3.3.1 · **Redis** 7.2
Collector: read-only, aggregate-only. Job arguments were never collected, and exception messages were redacted.

---

1. Executive summary

Ledgerly’s queues are not failing randomly. Three specific causes explain about 90% of the symptoms the team reported (“emails arrive hours late”, “the dead set keeps growing”):

#	Finding	Severity	Confidence	Est. fix effort
F1	The `mailers` queue has **no process listening to it**. 4,812 jobs are waiting forever	● Critical	High	15 min
F2	`ReportExportJob` **occupies most threads** on `default`, starving every other job	● High	Medium	2–4 h
F3	One webhook job causes **71% of all retries** (timeouts to a slow third party)	● High	High	2–3 h
F4	**88% of dead jobs** are `ImportRowJob` hitting `RecordNotFound`, a race with the DB transaction	● Medium	High	1 h
F5	31% of scheduled jobs fire within the **same minute** each hour	● Low	Medium	1 h

Top 3 actions, in order:

1. Add `mailers` to the Sidekiq process config (F1).
2. Move `ReportExportJob` to its own queue and process (F2).
3. Cap the webhook timeouts and back off retries (F3).

Fixing F1 alone is expected to clear the late-email complaints within minutes of deploying.

---

2. Snapshot at a glance

Metric	Value
Processes / total threads	3 / 30 (concurrency 10 each)
Busy threads at capture	30 / 30 (100%)
Enqueued (all queues)	7,391
Retry set	2,407 (sampled 1,000)
Dead set	9,940 (sampled 1,000)
Scheduled set	32,118

Queue	Size	Latency	Listened by
`critical`	12	4 s	3 processes
`default`	2,561	**14 min 20 s**	3 processes
`mailers`	4,812	**6 h 41 min**	**none**
`low`	6	38 s	3 processes

---

3. Findings

F1 — `mailers` queue is orphaned · ● Critical · Confidence: High

Evidence

- `mailers` holds 4,812 jobs, and the oldest was enqueued 6 h 41 min ago.
- None of the 3 running processes lists `mailers` in its queues (`critical`, `default`, `low`).
- Every `ActionMailer::MailDeliveryJob` in the sample targets `mailers`. That's the Rails default for `deliver_later`.

Likely cause Rails' default mailer queue is `mailers`. The Sidekiq config (`config/sidekiq.yml` or the `-q` flags in the Procfile) was probably rewritten at some point and dropped it.

Recommended checks

1. `grep -rn "queues\|-q " config/sidekiq.yml Procfile*` to confirm `mailers` is missing.
2. Check `config.action_mailer.deliver_later_queue_name`, in case it was set on purpose.

Fix Add `mailers` to the queue list, weighted, for example `[critical, 6], [mailers, 4], [default, 3], [low, 1]`. Deploy, then watch the queue drain.

Limitation A single snapshot can't tell whether the queue was *ever* listened to. Git history on the config will.

F2 — Long-running export job starves `default` · ● High · Confidence: Medium

Evidence

- 30 of 30 threads were busy at capture.
- 22 of those 30 were running `ReportExportJob`.
- `default` latency is 14 min, while `critical` (which is weighted first) is 4 s. Normal jobs are waiting behind exports.

Likely cause Exports are slow (likely minutes each) and share `default` with short jobs. When several customers export at once, they take over the thread pool.

Recommended checks

1. Look at `ReportExportJob` runtime in your APM or logs. Is p50 over 60 s?
2. Does the “exports” UI let users trigger several at once?

Fix

- Move it to its own queue: `sidekiq_options queue: :exports`.
- Run a separate process with low concurrency (for example `-q exports -c 3`).
- Optionally, add a per-account concurrency limit.

Limitation This is a point-in-time sample. A second snapshot at a different hour (Full Audit) would confirm whether the pattern repeats or was a spike.

F3 — Webhook retry storm · ● High · Confidence: High

Evidence

- In the 1,000 sampled retries, 712 (71%) are `WebhookDeliveryJob`.
- Errors in those retries: `Net::ReadTimeout` (64%), `Faraday::ConnectionFailed` (29%), other (7%).
- Messages are redacted. The host was replaced with `[HOST]`.

Likely cause One or more customer webhook endpoints are slow or down. With default retries (25, exponential backoff over about 20 days), each failing endpoint builds a long tail of retries. Each attempt holds a thread until the HTTP timeout fires.

Recommended checks

1. What is the configured HTTP read timeout? A default of 60 s means each attempt can hold a thread for a full minute.
2. Is most of the volume coming from a few endpoints? Check in your own DB. The collector doesn’t collect args, so it can’t tell.

Fix

- Set read and open timeouts to around 5 s.
- Set `sidekiq_options retry: 8`, then mark the endpoint as failing and stop.

- Add a simple per-endpoint circuit breaker.
- Put the job on a `webhooks` queue so it can't starve others.

Limitation Without args, the report can't name the failing endpoints. That analysis stays inside your systems, on purpose.

F4 — Dead set dominated by enqueue-before-commit race · ● Medium · Confidence: High

Evidence

- In the 1,000 sampled dead jobs, 881 (88%) are `ImportRowJob` with `ActiveRecord::RecordNotFound`.
- They failed on the first attempt and kept failing through all retries until they died.

Likely cause The job is enqueued inside a DB transaction, for example in an `after_save` callback. Sidekiq picks it up before the transaction commits, so the record doesn't exist yet. The error is consistent across attempts because the triggering import later rolled back, or the ID never got committed.

Recommended checks

1. Find where `ImportRowJob.perform_later` / `perform_async` is called. Is it inside `transaction do` or an `after_save` callback?
2. Do the dead jobs cluster around the times of failed imports?

Fix

- Enqueue in `after_commit`.
- On Rails 7.2+, use `enqueue_after_transaction_commit`.
- Then bulk-delete the existing dead `ImportRowJob` entries *after* confirming they aren't needed. The tool doesn't do this for you.

Limitation "Race" is the most common cause of this signature, not a certainty. A missing-record bug would look the same.

F5 — Scheduled-job thundering herd · ● Low · Confidence: Medium

Evidence

- 31% of the 32,118 scheduled jobs (sampled) fire at minute :00 of an hour.

Likely cause Reminders or cron-like jobs use `perform_at(time.beginning_of_hour)` or round times.

Fix Add jitter, for example `perform_in(delay + rand(0..300))`.

Limitation Low impact today. It will matter if traffic grows, because it stacks onto F2.

---

4. Remediation plan

Order	Action	Finding	Effort	Expected effect
1	Add `mailers` to the Sidekiq queue config	F1	15 min	Late emails stop
2	Move `ReportExportJob` to a dedicated `exports` process	F2	2–4 h	`default` latency back to seconds
3	Webhook timeouts + retry cap + own queue	F3	2–3 h	Retry set shrinks; threads freed
4	Move `ImportRowJob` enqueue to after commit	F4	1 h	New dead jobs stop
5	Add jitter to scheduled jobs	F5	1 h	Smoother hourly load

Re-check: run the collector again 48 h after deploying items 1–3 and compare. The Full Audit includes this comparison.

---

5. Scope and limitations

- **Point-in-time.** One snapshot shows state, not trends. The trend-based findings (retry growth, recurring starvation) need two snapshots.
- **No job arguments** were collected, so the report identifies *which job classes* are failing, not *which records or customers*.
- **Exception messages were redacted** for emails, tokens, hosts and long numbers. Some context is lost on purpose.
- **Sampling.** The retry, dead and scheduled sets were sampled at 1,000 entries each. Percentages are estimates.
- **No changes were made** to any system. All fixes are recommendations for your team to apply.

What the collector read

- Queue names, sizes and latency
- Process concurrency, busy counts and listened queues
- Retry, dead and scheduled counts
- Job class names
- Error class names
- Redacted error messages

What it never put in the snapshot

- Job arguments
- Redis keys outside Sidekiq's
- Credentials
- Application data