

Alerting and Scheduling

Pattern: getting signals to a human, on a cadence. Analysis produces signals, this stage runs the analysis on a schedule and delivers whatever warrants attention. There are two ways to do it, a scheduler and a script, or a dedicated automation platform, and the right choice depends on how much orchestration you want and how much attack surface you are willing to add.

The two approaches

Cron and a mail command. The simplest possible alerting layer, cron runs each analysis script on a schedule, and the script (or a wrapper) mails its output when there is something to report. Nothing to install, nothing new to secure, universally understood. This is what most people should reach for first, and for many environments it is entirely sufficient.

An automation platform. A dedicated workflow tool runs the scripts, collects their output, and handles delivery, formatting, and routing, with a visual interface for building and maintaining the workflows. It offers more than cron, richer notification formatting, conditional routing, easy fan-out to multiple destinations, retries, and a place to see run history, at the cost of being another piece of software to run, secure, and keep patched.

This environment uses an automation platform (n8n) for the alerting workflows, so this book describes that approach more fully, but the transferable idea is the *pattern*. An automation layer runs the checks and routes their output, not the specific product. If you prefer cron and a mail command the whole system works exactly the same. Only this final delivery stage changes.

A necessary caution about the automation layer

Recommending any automation platform to a security-minded reader requires honesty about a real consideration. **An internet-adjacent automation platform is attack surface, and it has to be secured and kept patched like everything else.** These platforms are powerful precisely because they can reach across your environment and take actions, which is exactly what makes them a high-value target. N8n specifically has had security vulnerabilities disclosed, as such platforms periodically do, and running one means committing to keeping it current and to put reasonable constraints on what it can reach.

If you adopt an automation platform for alerting:

- Keep it patched, and follow its security advisories, because a tool that can reach your whole environment is one you cannot afford to run stale.
- Constrain what it can reach and what credentials it holds to the minimum its workflows actually need.
- Treat it as sensitive infrastructure, not just a convenience.

There is a broader principle underneath this, returned to on the operating page: the monitoring system is part of the environment it monitors. The automation layer that sends your alerts can reach your hosts and send mail on your behalf. That makes it one of the more sensitive things you run, and it deserves scrutiny accordingly.

Frequency: matching the cadence to the category

How often a check runs should match how quickly its category matters. The reasoning by class:

- **Security-relevant categories run frequently**, on the order of every hour or so. Authentication attacks, boundary probes, and critical errors are things you want to know about while they are happening, not the next day. A shorter interval means faster awareness, at the cost of running the checks more often, which at this scale is negligible.
- **Health and capacity categories run less frequently**, on a daily rhythm for the slower-moving ones. Disk filling over time is a trend, not an emergency measured in minutes; a daily check catches it with plenty of margin. The per-host health snapshots underneath still happen every few minutes, but the *alerting* check that reads them can run hourly for acute spikes and daily for slow trends.
- **Reports and summaries run on a long cadence**, weekly or monthly, because their value is in the trend over time rather than immediate notification.

The point is to run each check as often as its category's trouble develops, and no more often than that. Running everything every minute would add load and noise for no benefit. Running everything daily would miss a fast-moving attack. Match the cadence to how fast the thing you are watching for actually moves.

Sample cron job

In order to get email alerts, you need to have a working MTA like `postfix` or `sendmail`. They are well documented and easy to set up. <https://ubuntu.com/server/docs/how-to/mail-services/>.

```
# /etc/cron.d/fleet-health-check
# Run the fleet health check hourly, at 5 past the hour.
# Runs on the COLLECTOR (the files host), as root, since it reads
# under /var/log/remote. Output (WARN lines, or nothing) is mailed
# to the address in MAILTO when the script produces any output.
```

```
MAILTO=alerts@example.org
```

```
5 * * * * root /usr/local/bin/fleet-health-check.sh
```

You would need to set up a job like this to execute each script on the cadence that aligns to the requirements of the category.

Adapt this for...

Any environment turning signals into notifications. Start with cron and a mail command; it is the simplest thing that works and adds no attack surface. Move to an automation platform only if you genuinely want its orchestration, and if you do, run it as the sensitive, patched, minimally-privileged infrastructure it is. Either way, match each check's frequency to how fast its category moves. The delivery mechanism is the most swappable part of the whole system; the analysis and the reasoning behind it are what matter.

Revision #2

Created 2026-07-31 22:37:25 UTC by Chris Landis

Updated 2026-07-31 23:04:31 UTC by Chris Landis