Software by Design
Nextcloud apps

Log monitoring and instance health

HealthCheck

User manual

Log watching, health dashboard and notifications.

For app administrators and Nextcloud system administrators.

App version	Language	Date
1.4.3	**English**	**25 October 2026**

Commit 5701f69+ · Web AGPL

Contents

Introduction iii
1 Roles: who can open HealthCheck and do what 1
1.1 The three doors . . . 1
1.2 What each role may do . . . 1
1.3 Why there is no open mode . . . 2
1.4 Edge cases . . . 2
2 Getting started: from sign-in to a watched log 3
2.1 Step 1: sign in and open the app . . . 3
2.2 Step 2: read the health page once . . . 3
2.3 Step 3: prove one alert channel . . . 3
2.4 Step 4: turn watching on . . . 4
2.5 What happens next . . . 4
3 Health: the check cards and what they watch 4
3.1 The summary badge . . . 5
3.2 The seven cards . . . 5
3.3 The Watching card . . . 6
3.4 Warnings the page can raise . . . 6
3.5 Edge cases . . . 6
4 Logs: read, search, and start fresh 6
4.1 Which log? — the file picker . . . 6
4.2 Filtering and searching . . . 7
4.3 Working with the lines . . . 7
4.4 The danger zone . . . 7
4.5 Edge cases and limits . . . 8
5 Alerts: channels, tests and digests 8
5.1 The channels . . . 9
5.2 Testing and re-enabling a channel . . . 9
5.3 Guardrails worth knowing before you flip them . . . 9
5.4 How a digest is shaped . . . 10
5.5 Edge cases . . . 10
6 Rules: severity, pacing and filters 10
6.1 How serious? — the severity chips . . . 10
6.2 How often? — the pacing chips . . . 10
6.3 Advanced filters . . . 11
6.4 Checks on the Health page (Nextcloud admin) . . . 11
6.5 Edge cases . . . 12
7 People: the app-admin list 12
7.1 How the list works . . . 12
7.2 Adding an app admin (Nextcloud admin only) . . . 12
7.3 Removing an app admin . . . 13
7.4 What app admins see here . . . 13
7.5 Edge cases . . . 13
8 Access: the door, and the doors inside 13
8.1 The front door . . . 13
8.2 The inner doors . . . 14
8.3 Default-deny details worth knowing . . . 14
8.4 URL housekeeping . . . 14
8.5 When someone should get in . . . 14
9 Settings: how saving works 15
9.1 Finding the right page . . . 15
9.2 One form, one save . . . 15
9.3 Versioned saves: nothing is silently overwritten . . . 15
9.4 Validation and field-level errors . . . 15

9.5 **Support us** . . . 15
10 occ, backups and getting data out 16
10.1 The occ commands . . . 16
10.2 Upgrade backups: export and restore . . . 16
10.3 Getting log data out . . . 16
10.4 Uninstalling . . . 17
11 Troubleshooting: every message, decoded 17
11.1 On the Health page . . . 17
11.2 Inside the health cards . . . 17
11.3 On the Logs page . . . 18
11.4 While saving settings . . . 19
11.5 At the door . . . 19
11.6 When nothing here matches . . . 19
12 FAQ and “how do I…” 19
12.1 How do I…? — the quick index . . . 19
12.2 Frequently asked questions . . . 20
13 Accessibility 21
13.1 Keyboard paths . . . 21
13.2 What a screen reader announces . . . 21
13.3 Contrast, targets and zoom . . . 21
13.4 Motion and forced colours . . . 22
13.5 Known limits . . . 22
14 Glossary 22
15 Catalog: every surface and where it is documented 22
15.1 Pages and folds . . . 23
15.2 Addresses . . . 23
15.3 Commands . . . 24
15.4 Background machinery . . . 24
About Software by Design 25
Support & how to contribute 25
Index 27

Introduction

HealthCheck (app ID `logcheck`) watches the Nextcloud log file for you: it tails new lines in the background, groups them into digests, and delivers them through email, Slack, a webhook or in-app notifications. The **Health** page alongside summarises instance state in seven cards — one per health probe: log alerts, background jobs, PHP, database, data disk, HTTPS and updates. The app replaces neither Nextcloud administration nor a SIEM; it makes errors visible before they disappear, and it tells you when it cannot watch at all.

Requirements: file-based logging (`log_type = file`) — on several app servers that must be one shared log file, otherwise the app says **Can't watch** instead of silently covering one node; running Nextcloud cron for background watching and digests. The web app is free software (AGPL).

What you will find here

- Who may open the app at all and what each role may do (chapters 1 and 8).
- The ten-minute path from sign-in to a watched log with a proven channel (chapter 2).
- Daily work: the health cards, the log browser, alert channels, rules, the app-admin list and how saving works (chapters 3–9).
- Reference and rescue: occ commands and backups, every error message decoded, a "how do I…" index, accessibility notes, a glossary and a surface catalog (chapters 10–15).

Audience

- **App admin** — Health, Logs, Alerts, Rules and a read-only People page; the daily operator.
- **Nextcloud admin** — everything above plus people management, downloads, the danger zone, log excerpts and private webhooks.

Everyone else sees only **Not authorized** with **Back to Nextcloud** — chapters 1 and 8 explain why.

App version and screenshots

HealthCheck **1.4.3** (25 October 2026). All figures are real captures from the seeded lab instance at `http://127.0.0.1:8081` — so the **HTTPS** card may show **Not using HTTPS** and **Updates Not checked yet**: a lab, not a print error. Numbered callouts in the text (Marker 1, Marker 2, …) refer to the badges burned into the screenshots.

1 Roles: who can open HealthCheck and do what

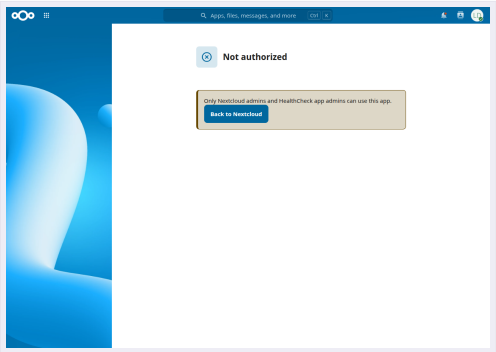
HealthCheck reads the server log, and a server log is one of the most sensitive files on the machine: it can contain user names, IP addresses, file names, tokens and stack traces. That is why the app has *no* open mode. Every visitor passes exactly one of three doors, and the door decides everything they can see and do afterwards.

1.1 The three doors

Signed out. Anyone who is not signed in is sent to the Nextcloud sign-in page. HealthCheck itself never renders for anonymous visitors.

Signed in but not entitled. A regular user — someone who is neither a Nextcloud admin nor listed under **People** — sees the **Not authorized** page with a single way back: **Back to Nextcloud**. Nothing else of the app is delivered, not even the navigation. On the demo instance this is what *Dieter Krüger* sees.

Entitled. Nextcloud admins always get in. Everyone else gets in only because a Nextcloud admin added them as an **App admin** under **People** (chapter 7). On the demo instance *Mona Appel*, *Lars Kranz* and *Karl Lorenz* hold this role.

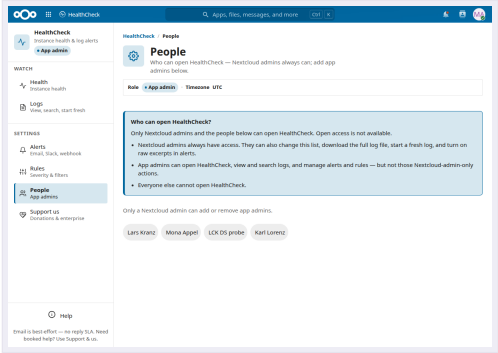


1.2 What each role may do

Being entitled opens the app — but the two entitled roles are not equal. An app admin runs the daily work; a Nextcloud admin additionally owns every action that can destroy data or widen exposure. The split is deliberate: the people who watch for errors should not need the keys that could erase the evidence.

Capability	App admin	Nextcloud admin
Open **Health**, **Logs**, **Alerts**, **Rules**	yes	yes
Turn **Watch log file** on and off, use **Check again**	yes	yes
Search, filter and copy log lines	yes	yes
Configure channels, test them, re-enable failed ones	yes	yes
Change severity, pacing, filters and mutes	yes	yes
See the **People** list	yes	yes
Add or remove app admins	—	yes
Download the full log file	—	yes
Start fresh log, **Delete log file**, **Remove this copy**	—	yes
See the log file path on disk	—	yes
Enable raw log excerpts in alerts	—	yes
Allow private-network webhook addresses	—	yes
Turn individual health checks on and off	—	yes

App admins see the **People** page read-only: the list is visible so they know who else is responsible, but the **Add app admin** field and the **Save** button are not rendered for them, and the API refuses the write on the server side — the read-only page is a courtesy, not the actual guard.



1.3 Why there is no open mode

The **People** page states it plainly: *open access is not available*. This is a product decision, not a missing feature. A log line that looks harmless (“login failed for alice”) already leaks behavioural data about a colleague; an excerpt can leak a password. Restricting the audience to two small, accountable groups keeps the blast radius of every new feature small — and it is the same reason raw excerpts and private-network webhooks are opt-in even for Nextcloud admins (chapter 5).

1.4 Edge cases

- **Removing yourself.** A Nextcloud admin can remove their own account from the app-admin list — harmless, because Nextcloud admins never lose access. The app admin list may also be empty; the app still works, it is then simply Nextcloud-admin-only.
- **A removed app admin keeps no residue.** Deleting a chip under **People** and saving revokes access at the next request; there are no sessions to flush and no per-user settings left behind.
- **LDAP or deleted users.** A uid that no longer exists in the directory can never sign in, and saving the list rejects unknown uids outright with **Unknown**. A stale chip is cosmetic, not a door — remove it at the next occasion.
- **List size.** The app-admin list is capped at 100 entries; far above anything a readable audit trail survives anyway.

2 Getting started: from sign-in to a watched log

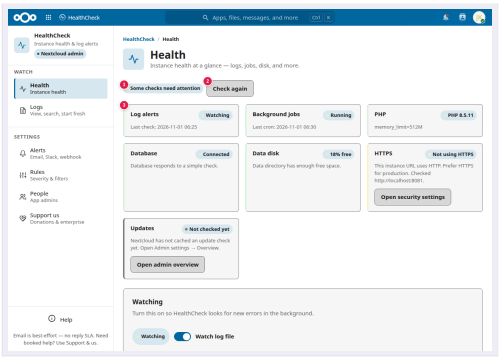
This chapter walks the shortest path from signing in to a HealthCheck that actually notifies you: health page reviewed, one alert channel proven, watching switched on. It takes about ten minutes and uses the live state of the demo instance — your own instance will show different names and times, but the same controls.

2.1 Step 1: sign in and open the app

1. Sign in to Nextcloud with the account your admin gave you. If you land on **Not authorized** instead of the app, you are missing an entitlement — chapter 8 explains who can fix that.
2. Open **HealthCheck** in the app menu. On a phone or a narrow window the sidebar folds away: tap **Menu** at the top of the content area to open it.
3. **You see:** the **Health** page — a summary badge on top, seven check cards below, and the **Watching** card that controls the background monitor.

2.2 Step 2: read the health page once

Before enabling anything, look at the summary badge (chapter 3 explains every card). On the demo instance it reads **Some checks need attention** — the lab URL is plain HTTP and the update check has not run, which is exactly the kind of noise a lab produces. A summary that already says **Something needs fixing** is worth a detour into chapter 11 first.

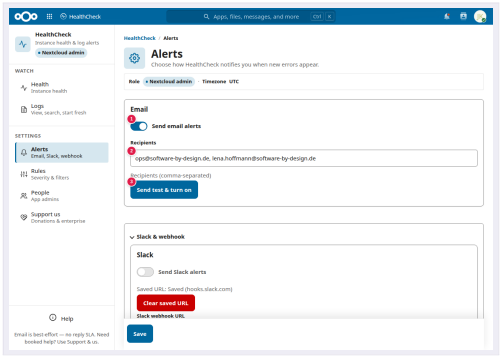


Marker 1: the summary badge rolls the seven cards into one state. Marker 2: **Check again** runs one probe pass immediately. Marker 3: a single card — title, state badge and a detail line.

2.3 Step 3: prove one alert channel

Watching without a channel stays silent, so set one up first.

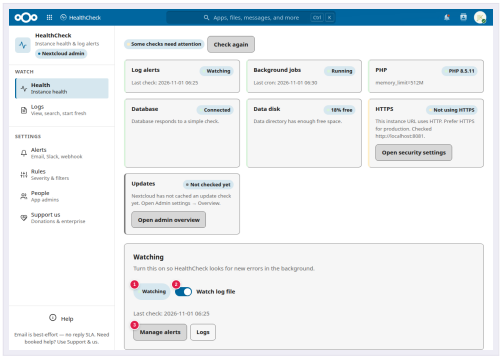
1. Open **Alerts** in the sidebar.
2. In the **Email** card, switch **Send email alerts** on and enter at least one address in **Recipients** — comma-separated for several. On the demo instance `ops@software-by-design.de` and `lena.hoffmann@software-by-design.de` are configured.
3. Click **Send test & turn on**. The app sends a real test message *before* it counts the channel as on — a channel that cannot reach you would otherwise fail silently for weeks.
4. **You see:** a short status line under the button confirming the test, and the channel now counts as active.



Marker 1: the channel switch **Send email alerts**. Marker 2: **Recipients**. Marker 3: **Send test & turn on** delivers a real test message first.

2.4 Step 4: turn watching on

1. Go back to **Health** and scroll to the **Watching** card.
2. Switch **Watch log file** on.
3. **You see:** the badge flips to **Watching**, and a **Last check** timestamp appears once the background job has run. If no channel is active yet, a checklist appears instead — **Set up at least one alert channel** plus **Watch log file is on** — with **Set up alerts** leading back to step 3.



Marker 1: the watching state badge. Marker 2: the **Watch log file** switch. Marker 3: the actions row — **Manage alerts** once a channel is ready, otherwise the setup checklist.

2.5 What happens next

From now on a Nextcloud background job tails the log file at the server's cron cadence (system cron is the reliable choice; AJAX cron only works while pages are open). New lines at or above the severity you set under **Rules** collect into a digest, and the digest leaves through every channel you turned on — the demo instance batches every fifteen minutes. You can always force one pass with **Check again**.

That is the complete daily loop: the cards warn you, the log explains why, the channels bring it to you. The next chapters cover each part in the order you will meet it — and chapter 11 names every error message the app can show.

3 Health: the check cards and what they watch

The **Health** page is the landing view and the fastest answer to "is something wrong right now?". It runs a set of probes — independent little checks — and rolls their results into one summary badge

plus one card per probe. The page is a snapshot, not a live monitor: cards reflect the moment the page (or **Check again**) ran.

3.1 The summary badge

Everything looks fine — every enabled probe returned OK.

Some checks need attention — at least one probe is degraded. The guilty card carries the matching badge colour.

Something needs fixing — at least one probe reports a critical state; treat this as a go-and-look-now signal.

Some checks are unknown — one or more probes could not decide (for example before the first run).

Check again next to the badge runs one full probe pass on demand — the same work the background job does — and redraws the cards. Use it after you fixed something to confirm the fix landed; you do not have to wait for the next cron tick.

3.2 The seven cards

Each card follows the same shape: title, state badge, a one-line detail, and optionally a button that takes you to the right place.

Log alerts — the state of the watcher itself: whether it is on, when it last ran, and whether a run errored. This card is the heartbeat of chapters 2 and 5.

Background jobs — whether Nextcloud cron ran recently. If this card degrades, digests and **Check again** fall behind too: the watcher lives inside the same job system.

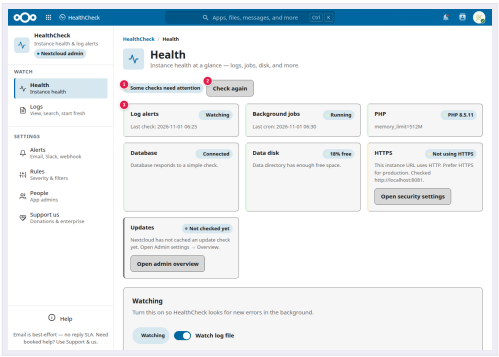
PHP — the PHP version and headline limits (for example `memory_limit`); degrades when a value risks breaking apps.

Database — a live query against the database; the first card to fail when the DB stalls.

Data disk — free space in the data directory (the demo instance reports *18% free*).

HTTPS — whether the configured instance URL is HTTPS. A lab instance on `http://localhost:8081` correctly shows **Not using HTTPS**; on production this is the card that must stay green. **Open security settings** jumps to the matching Nextcloud settings.

Updates — the cached update check. A lab may sit on **Not checked yet**; **Open admin overview** opens the Nextcloud overview that triggers it.



Marker 1: the summary badge. Marker 2: **Check again** reruns all probes immediately. Marker 3: one card — here **Log alerts**, currently **Watching**.

3.3 The Watching card

Below the grid, the **Watching** card is the on/off for the background monitor. Its badge mirrors the watcher state (**Watching** / off / error), the switch persists immediately, and the actions row adapts to what is missing: **Set up alerts** while no channel is ready, **Manage alerts** and **Logs** once everything works, **Try again** after a failed run. Chapter 2 walks this flow end to end.

3.4 Warnings the page can raise

The page is honest about situations a green grid would hide:

- **Can't watch** — the log backend is not file-based (`log_type` is not `file`; the longer form reads **Log watching is not available on this server.**), or the running server is not the one the watcher saw before: **HealthCheck noticed a different server. Multi-server setups need one shared log file.** Watching stays off; the health cards still run.
- **Background checks look stuck.** — cron has not ticked for a while; try **Check again**, then check the system cron entry on the host.
- **Stored channel secrets cannot be read. Re-enter webhook URLs.** — saved webhook URLs became undecryptable (for example after a secret rotation); re-enter them under **Alerts**.
- **All health checks are turned off.** — every probe was disabled under **Rules**; only Nextcloud admins see the link to turn them back on (chapter 6).

3.5 Edge cases

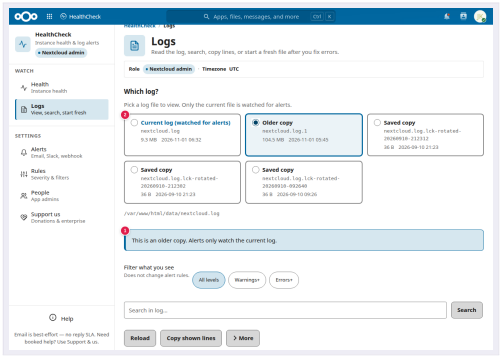
- A probe that cannot run on this server (say HTTPS behind a reverse proxy that terminates TLS elsewhere) can be switched off under **Rules** — it then disappears from the grid instead of crying wolf forever.
- The summary mirrors exactly the worst card — no averaging, no accumulation. **Some checks need attention** therefore always means "at least one card is degraded", never "all are borderline".
- Card states are per-run. A transient DB hiccup can paint one snapshot red and the next green; sustained patterns are what the log alerts channel is for.

4 Logs: read, search, and start fresh

The **Logs** page is where a red card becomes an explanation. It reads the same file the watcher tails, plus every rotated copy still on disk — so you can answer "what happened" and "what happened before" without shell access.

4.1 Which log? — the file picker

The top of the page lists every log file HealthCheck can see: the **Current log (watched for alerts)** first, then **Older copy** entries (server-rotated files such as `nextcloud.log.1`) and **Saved copy** entries (copies the app set aside itself). Only the current file feeds alerts — picking a copy shows the banner **This is an older copy. Alerts only watch the current log.** so nobody mistakes a quiet archive for a quiet server.



Marker 1: the archive banner — a copy never feeds alerts. Marker 2: the file picker with the current log on top and rotated copies below.

Nextcloud admins additionally see the absolute path under the list, which matters the moment someone asks “which directory was that again?” on a host with several mounts.

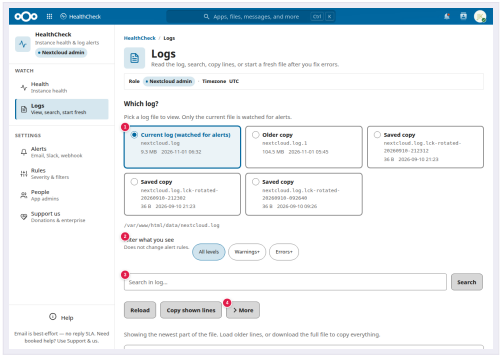
4.2 Filtering and searching

The chips **All levels** / **Warnings+** / **Errors+** thin the view to what you can act on. They only change what is displayed — the hint **Does not change alert rules.** is literal: muting a view never mutes an alert (chapter 6 is where alerting is shaped).

Search in log matches against the loaded part of the file and reports where it looked — the honest label is **Showing the newest part of the file. Load older lines to see more.** An empty result means “not in this window”, not “never happened”; **Load older lines** pulls the previous chunk in, repeatedly, until you reach the beginning of the file.

4.3 Working with the lines

- **Reload** refetches the tail — useful while an incident unfolds.
- **Copy shown lines** puts the visible lines on the clipboard for a ticket or a chat message. This works for app admins too: it copies what they may already see, nothing more.
- **More** hides the heavier tools: **Show raw lines** switches the view between formatted rows and the raw JSON lines for when a field looks odd; **Download full file** (Nextcloud admin) fetches the whole log; **Remove this copy** appears only while a copy is selected and deletes that file.



Marker 1: the file picker. Marker 2: level chips. Marker 3: the search box. Marker 4: the **More** menu with raw view, download and copy removal.

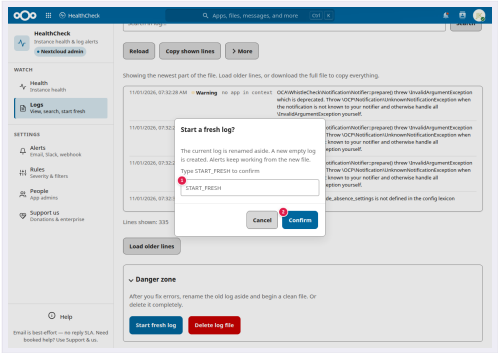
4.4 The danger zone

After the errors are understood, the log often deserves a clean start. Two irreversible tools live behind the **Danger zone** fold — both Nextcloud-admin-only, both behind a typed confirmation, because “I clicked the wrong button” must never be a way to lose evidence:

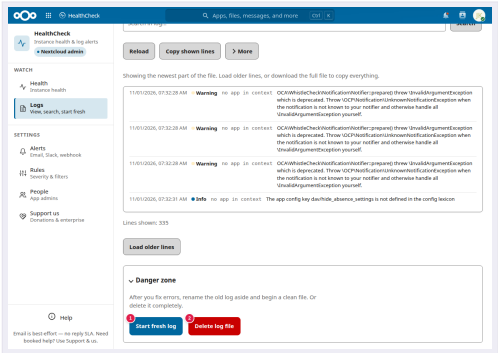
Start fresh log renames the current file aside as a `lck-rotated` copy and begins an empty log. Alerts keep working — the cursor follows onto the new file, and the copy stays readable (and removable) from the file picker.

Delete log file removes the file entirely. Nothing is renamed aside; what is gone is gone.

Each action opens the same confirmation dialog: a sentence stating what happens, an input that wants the exact word (`START_FRESH`, `DELETE` or `DELETE_COPY`), and **Confirm**. A wrong word produces an inline error, not a mutation.



Marker 1: type the word exactly as shown. Marker 2: **Confirm** only mutates after the word matches.



Marker 1: **Start fresh log** — the non-destructive choice. Marker 2: **Delete log file** — gone is gone.

4.5 Edge cases and limits

- **No log file yet. Nextcloud creates it when something is logged.** — Nextcloud creates `nextcloud.log` with the first logged line; a brand-new instance simply has nothing to show.
- **Cannot read the log file. Check permissions.** — filesystem permissions; the web user cannot open it. Fix ownership/mode on the host.
- **The log file is not writable from here. Ask your host for file permissions.** — the danger zone is hidden when the file cannot be renamed or removed by the web process.
- **A rotated copy with that name already exists. Try again.** — two “start fresh” runs within the same name window; try again in a moment.
- Search and the tail view cover the *recent* part of the file for speed; **Download full file** is the escape hatch when the recent window is not enough.

5 Alerts: channels, tests and digests

Alerts decides *where* a digest goes. HealthCheck can deliver through four channels — email, Slack, a generic webhook, and in-app notifications — and every channel follows the same rule: it only counts as on after a real test message went through. That verify-then-enable order is deliberate: the worst alert channel is the one that looked configured and never worked.

5.1 The channels

Email — the zero-setup channel; it uses the instance's own mail configuration. Recipients are a comma-separated list of plain addresses, one digest to all of them. On the demo instance alerts go to `ops@software-by-design.de` and `lena.hoffmann@software-by-design.de`.

Slack — an incoming-webhook URL from your Slack workspace. Once saved, the URL is stored encrypted and shown only masked; **Clear saved URL** marks it for removal on the next save instead of asking you to blank a field you cannot read.

Webhook — any HTTPS endpoint; the digest goes out as a JSON payload for your own automation. Public addresses only by default — see the guardrails below.

In-app notifications — a Nextcloud notification for every entitled user, the same bell your other apps use — and therefore also inside the Nextcloud mobile apps, which makes this the closest thing to a phone alert the suite offers.

Slack and webhook live behind the **Slack & webhook** fold, in-app notifications and the two safety switches behind **More options** — progressive disclosure, not hidden features.

5.2 Testing and re-enabling a channel

Send test & turn on sends a real message first and only then flips the switch — a failed test leaves the channel off and says so inline. There is no "turn on first, find out later" path.

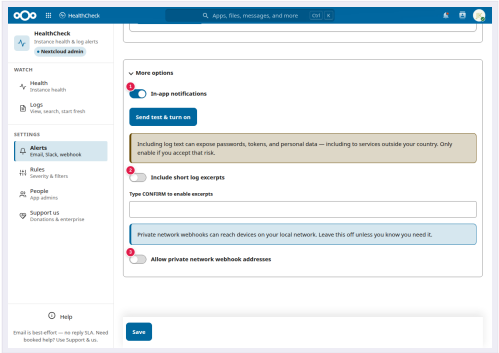
A channel that repeatedly fails at delivery time disables itself and shows **Channel disabled after repeated failures.** on the **Alerts** page — this is the state the demo instance's Slack channel is in right now. The switch is locked off (a persisted "on" plus a runtime failure would be a lie), and **Re-enable & test** is the way back: fix the cause (URL, credentials, mail transport), then prove it with a test. Both states are also visible on the **Health** page's **Log alerts** card.

5.3 Guardrails worth knowing before you flip them

Two switches under **More options** are Nextcloud-admin-only because they deliberately widen exposure:

Include short log excerpts — adds short message excerpts (capped at 200 characters) to digests. The warning next to it is real: log text can contain passwords, tokens and personal data, and a channel like Slack carries it out of your legal jurisdiction. Enabling requires typing `CONFIRM` into the field below — a consent gate, not a checkbox you can brush past.

Allow private network webhook addresses — lets a webhook URL point at private IPs or hostnames on your LAN. It is off by default because an URL field that can dial inside your network is a textbook SSRF foot-gun; turn it on only when the receiving service genuinely lives there.



Marker 1: **In-app notifications** — the Nextcloud bell, also on mobile. Marker 2: **Include short log excerpts** with its consent field. Marker 3: **Allow private network webhook addresses**.

5.4 How a digest is shaped

Rules decide *what* lands in a digest (chapter 6); the channel chapter is where it goes. In short: matching lines collect with a per-line fingerprint, similar lines group together, and a digest leaves at most once per pacing interval — never a mail per log line. The payload carries counts and fingerprints; the actual log text only travels when excerpts are on.

5.5 Edge cases

- Email depends on the instance mailer; when it fails, the channel error is sanitized to **Email could not be sent. Check mail settings.** rather than echoing transport detail — and a webhook failure reports **Webhook failed. Check the URL and try again.**, never the resolved address.
- Removing the last recipient while the switch is on is a validation error, not a silent black hole.
- Replacing a saved webhook URL means typing a new one — you never edit the masked value itself.
- **Stored channel secrets cannot be read. Re-enter webhook URLs.** on **Health** means the encrypted URLs became unreadable (secret rotation, moved instance): re-enter them once.

6 Rules: severity, pacing and filters

Rules shapes the stream between "everything that is logged" and "what reaches a human". Three questions, one page: how serious must a line be, how often may a digest leave, and which noise should never become an alert at all. Getting these right is the difference between an alert you read and an alert you train yourself to ignore.

6.1 How serious? — the severity chips

Two choices, stored as a minimum level:

Errors — only lines at error level and above (the classic "something broke"). The quietest setting and the right default for a calm production instance.

Warnings too — warnings and above; catches deprecations, failed background jobs and retry loops before they turn into errors. The demo instance runs this mode — expect a livelier mailbox.

6.2 How often? — the pacing chips

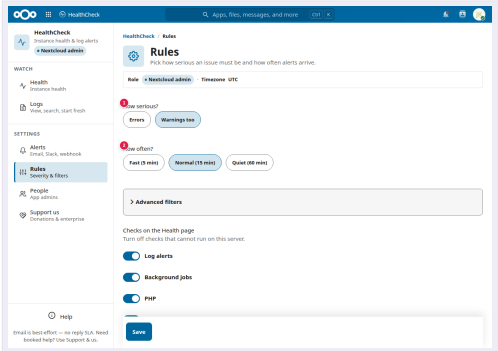
Matching lines do not email themselves. Every run collects what passed the filter into an accumulator, and a digest leaves at most once per interval:

Fast (5 min) — for active incident watching; at the price of more mail.

Normal (15 min) — the shipped default and the demo setting; a burst of identical lines still arrives once.

Quiet (60 min) — for stable instances where an hourly rollup is enough.

Grouping is by fingerprint — a signature of the message shape — so five hundred identical warnings become one counted line in the digest, not five hundred emails. The interval only limits how often a digest may leave; the collection itself never drops matching lines.



Marker 1: **How serious?** — **Errors** or **Warnings too**. Marker 2: **How often?** — the digest pacing.

6.3 Advanced filters

Behind **Advanced filters** live the scalpel options for when severity alone is too blunt:

Which apps to watch — **All apps** watches everything; **Only these apps** is an allow-list; **All except these apps** is a deny-list. The app names go into the comma-separated field below.

Ignore messages that match (one pattern per line) — one regular expression per line; any matching line is muted. Patterns are validated on save — a broken expression is rejected with a field-level error instead of silently never matching.

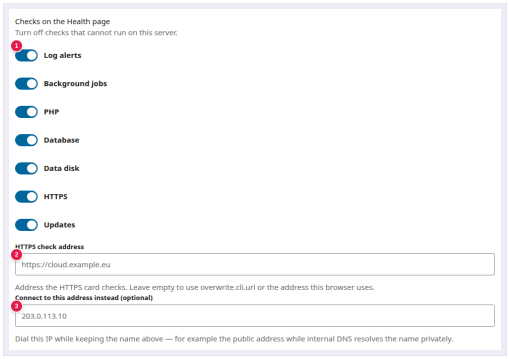
Always mute these apps (comma-separated) — a blunt "never alert me about this app" list. HealthCheck always mutes itself (`logcheck`) on top of whatever you enter — the watcher must never alert on its own noise.

All filtering happens *before* accumulation, so a muted line never reaches a digest — but it is still written to the log file and visible on the **Logs** page. Muting hides noise from humans; it never erases evidence.

6.4 Checks on the Health page (Nextcloud admin)

The bottom fieldset maps one-to-one to the health grid: each of the seven probes can be switched off when it cannot run meaningfully — the HTTPS card behind a TLS-terminating proxy, the updates card on an air-gapped host. A disabled probe disappears from chapter 3's grid entirely rather than sitting there red forever.

The **HTTPS check address** overrides which URL the HTTPS probe dials (empty = `overwrite.cli.url` or the browser URL), and **Connect to this address instead (optional)** pins the connection to a specific IP while keeping the checked name — the classic "public name, private DNS" split-horizon case.



Marker 1: one switch per probe. Marker 2: **HTTPS check address**. Marker 3: **Connect to this address instead (optional)** — dial this IP while checking the name above.

6.5 Edge cases

- Muting everything until digests are always empty is legitimate — the watcher still runs, cards still render; only the channels go quiet.
- An allow-list that names no app watches nothing; combine **Only these apps** with an empty list only deliberately.
- Severity and pacing apply to *alerts*; the **Logs** page filter chips are a view and do not interact with them.
- Regex mutes run against the message text, not the JSON — anchor them to stable phrasing, not to timestamps or request ids.

7 People: the app-admin list

People is the shortest page in the app and the one with the longest reach: it holds the list of app admins — the only people besides Nextcloud admins who can open HealthCheck at all (chapter 1 explains what the role may do).

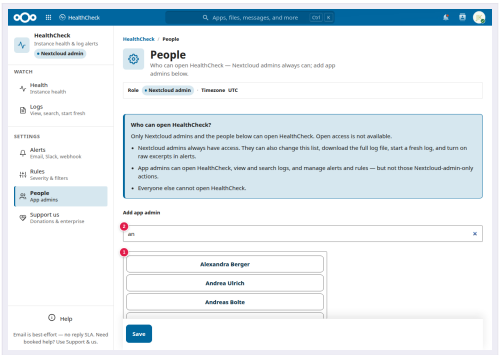
7.1 How the list works

The blue explainer at the top is the contract: Nextcloud admins are always in, app admins are in because they are listed, everyone else is out — *open access is not available*. The chips below it are the current list: on the demo instance *Lars Kranz*, *Mona Appel*, *LCK DS probe* and *Karl Lorenz*.

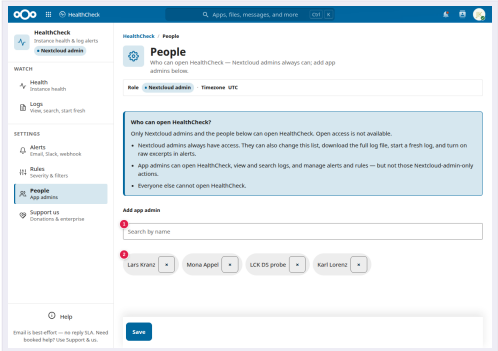
Editing is deliberately unexciting: there is no per-person settings page, no invite flow, no expiry. A person is either a chip on this page or they do not get in.

7.2 Adding an app admin (Nextcloud admin only)

1. Open **People** as a Nextcloud admin.
2. Type at least two characters of a name into **Search by name** — the field is a combobox over the user directory.
3. Pick the person from the result list; they appear as a chip right away.
4. Click **Save**. The choice only counts after saving — leaving the page discards a half-made pick.
5. **You see:** the person in the chip row after the reload; they can open HealthCheck from their next request on.



Marker 1: directory matches — *Alexandra Berger* would become a chip here. Marker 2: the search field; two characters minimum, arrows and Enter navigate the list.



Marker 1: the combobox. Marker 2: the current chips with their *remove* buttons.

7.3 Removing an app admin

Click the × on the chip (its label reads *Remove* name) and save. Access ends with the next request — no sessions to flush, no leftover per-user state. A Nextcloud admin removing their own uid keeps access anyway, because that role never depends on this list.

7.4 What app admins see here

App admins open the same page read-only: the explainer and the chips are visible (knowing who shares the duty is part of the job), but there is no search field, no remove button and no **Save** — **Only a Nextcloud admin can add or remove app admins.** The API enforces the same split server-side.

7.5 Edge cases

- **No people found.** — the directory genuinely has no match; check the spelling or the user's display name.
- **Search failed. Try again.** — the directory lookup errored; retry, and check whether the user backend (LDAP et al.) is reachable.
- **Unknown** on save — a chip whose uid vanished from the directory between picking and saving; remove the chip and save again.
- The list is capped at 100 entries.
- Two admins editing this page simultaneously are protected by the version check described in chapter 9 — the second save conflicts instead of silently overwriting.

8 Access: the door, and the doors inside

Access in HealthCheck is layered like a building with a locked front door and a few locked rooms inside. This chapter collects the whole model in one place — the boundary checks, the privacy defaults, and the deliberate dead ends — so a "why can't I…" question has one home instead of five.

8.1 The front door

Signing in is Nextcloud's job; the app never shows a public surface. Past the login, every request runs the entitlement check: Nextcloud-admin group membership, or a uid on the app-admin list (chapter 7). Failing it renders the **Not authorized** page with **Back to Nextcloud** — one message, one exit, no partial chrome to poke at.

Because the log can contain personal data and secrets, the restriction is intentionally unspectacular: no group mapping, no guest mode, no time-boxed invites. An uid that is not on the list fails closed; the list itself refuses the literal *open* mode server-side.

8.2 The inner doors

Inside, the Nextcloud-admin-only actions form a second ring — everything that can destroy evidence or widen exposure:

- destructive file operations: **Start fresh log**, **Delete log file**, **Remove this copy**, **Download full file**, and even the path display;
- exposure switches: raw log *excerpts* in digests and *private-network* webhook targets;
- people management and the probe toggles under **Rules**.

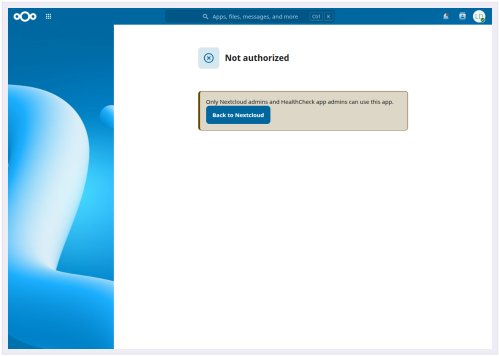
App admins see none of the controls for these — and the API answers **Not authorized.** if they are called anyway, so the hidden buttons are convenience, not the guard.

8.3 Default-deny details worth knowing

- **Webhooks cannot dial your LAN** unless explicitly allowed (SSRF guard, chapter 5) — **Webhook URL** covers unresolvable, non-HTTP(S), loopback and private-range targets alike.
- **Excerpts require typed consent** (`CONFIRM`) and are capped at 200 characters — a deliberate "think twice" built into the UI.
- **Confirmation words** guard the destructive actions; they are matched exactly.
- **Errors never leak internals**: transport failures surface as sanitized sentences (**Webhook failed. Check the URL and try again.**), never as stack traces, resolved IPs or credentials.

8.4 URL housekeeping

Bookmarks from older releases keep working: legacy settings addresses (`/settings/status`, `/channels`, `/watch`, `/access`, `/privacy`) redirect to the matching current page — status moves to **Health**, the rest to their sections — instead of 404-ing into confusion.



8.5 When someone should get in

1. Confirm they really need log access — the bar is "must act on instance errors", not curiosity.
2. A Nextcloud admin adds them under **People** (chapter 7).
3. They sign out and back in once if the app was open in another tab — then **HealthCheck** appears in their app menu.

If the intended audience is wider than "people who fix the server", the right answer is almost always a digest to a shared mailbox — not a longer admin list.

9 Settings: how saving works

All four settings pages — **Alerts**, **Rules**, **People**, **Support us** — share one behaviour model, and understanding it once saves a class of surprises later.

9.1 Finding the right page

The sidebar groups them under **Settings**; the address `/settings` alone lands on **Alerts**, and the legacy aliases from older releases redirect to their modern sections (chapter 8). On narrow screens the sidebar folds away — the in-page chip bar at the top of each settings page jumps between sections without opening the drawer.

9.2 One form, one save

Each settings page is a single form: every switch, chip, field and fold on it belongs to one document, and **Save** commits the whole page at once. Three consequences:

- Folding away a section (**Slack & webhook**, **More options**, **Advanced filters**) never discards it — the values inside still save with the rest.
- A switch that looks off but carries a saved URL still "counts" its stored values; **Clear saved URL** is the explicit removal path (chapter 5).
- Leaving the page discards unsaved changes; there is no autosave and no draft.

9.3 Versioned saves: nothing is silently overwritten

Every settings form loads with a hidden version number. On save the server compares it to the stored version: equal means commit and bump; different means somebody else saved in between — and the save *fails closed* with a conflict instead of merging.

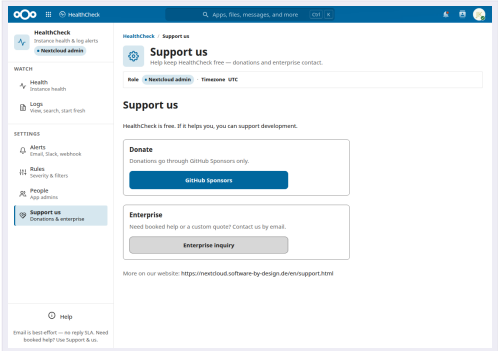
Picture the realistic case: *Mona Appel* edits the recipient list while *Karl Lorenz* tightens a mute pattern on another tab. Whoever saves second sees **Settings changed elsewhere — reload and try again.** instead of a silent last-writer-wins. The honest response is to reload (the page then shows the other admin's change) and re-apply your edit. This is optimistic locking, and it exists precisely because settings here steer alerting — a lost *off* or a lost recipient is a monitoring gap, not a cosmetic glitch.

9.4 Validation and field-level errors

Saves are validated server-side, and errors land on the offending control: the field turns red, gets `aria-invalid`, and an inline message names the problem (*Invalid mute pattern.*, **Unknown**, missing recipients and friends). Nothing half-saves: a rejected form changes nothing on the server.

9.5 Support us

The fourth settings page needs no saving at all: it links to **GitHub Sponsors** (the only donation channel), to an **Enterprise inquiry** mail address for booked help, and to the project website. It exists so that a free AGPL app can say so honestly in-product.



10 occ, backups and getting data out

Everything in this chapter happens outside the web page: command line, snapshots, downloads. It exists for the moment you suspect the UI — or need the data somewhere else.

10.1 The occ commands

HealthCheck ships five `occ` commands, all Nextcloud-admin territory run on the host:

`logcheck:status` — the same status block the **Health** page uses: watch state, backend, last check, per-channel health, settings version. The first thing to paste into a bug report.

`logcheck:run` — one watch iteration right now; the CLI twin of **Check again**.

`logcheck:test-channel <email|slack|webhook|notification>` — sends a real test on one channel and prints `OK` or a sanitized error. Faster than clicking when a channel is misbehaving.

`logcheck:reset-cursor` — moves the read position to the end of the current log. Use after restoring a huge file or when the watcher should skip a backlog of ancient history — it answers "why does the first digest contain last month's noise?" before it is asked.

`logcheck:upgrade-backup` — snapshot management (below).

10.2 Upgrade backups: export and restore

Before every app update a repair step snapshots HealthCheck's tables into app data — the safety net behind "update first, ask questions later". `occ logcheck:upgrade-backup list` shows what is on disk: snapshot id, timestamp, app version, reason (`pre-migration`, `integration-test`) and table count; the demo instance keeps seven-table snapshots per release, newest five retained.

- `create` takes a manual snapshot before a risky change.
- `restore` is the import direction: it puts a chosen snapshot's rows back — settings, cursor, channel state and pending digests as they were. Restore rewrites live state, so it asks for confirmation on the console; run it only when you mean it, and expect **Settings changed elsewhere — reload and try again.** on any web page that had the old version open.

10.3 Getting log data out

- **Download full file** (Logs →**More**, Nextcloud admin) — the raw file, the honest export for a ticket or an archive.
- **Copy shown lines** — the clipboard variant for a few lines; works for app admins.
- There is deliberately no "send me the log" button — exports are pulled by people who are already allowed to see everything, never pushed to a wider audience.

10.4 Uninstalling

Removing the app runs an uninstall repair step that drops its tables, deletes its app-config values and removes the stored upgrade-backup snapshots. What stays is what never belonged to the app: the log files themselves. *Disabling* the app, by contrast, keeps everything — data returns untouched on re-enable.

11 Troubleshooting: every message, decoded

HealthCheck fails in sentences, not codes. This chapter lists every message the app can show, what actually causes it, and the shortest honest fix — grouped by where you meet it.

11.1 On the Health page

Can't watch

The instance writes logs elsewhere (`log_type` is not `file`) — the longer form on the page reads **Log watching is not available on this server.** There is no adapter to configure away; the health cards still run, alerting does not.

HealthCheck noticed a different server. Multi-server setups need one shared log file.

Multi-server setup where each node writes its own log — the hint under it, *Turn watching off, or ask your host to use one shared log file for all servers.*, is the whole fix: point every node at one shared log file, or turn watching off.

Background checks look stuck.

Cron has not ticked. Try **Check again** to prove the probes work, then repair the system cron entry on the host — AJAX cron only runs while pages are open.

Stored channel secrets cannot be read. Re-enter webhook URLs.

The saved webhook URLs can no longer be decrypted (instance secret rotated, data moved). Re-enter them once under **Alerts**.

HealthCheck is busy checking the log. Try again in a moment.

Another run holds the watch lease — usually a cron pass or a second **Check again**. Wait and retry; it is a mutex, not an error.

Could not check the log right now. Try again in a moment.

/ **Check failed. Try again.** / **The last background check did not finish cleanly.** — a run died mid-pass (locked file, DB hiccup). The next run usually clears it; if it repeats, `occ logcheck:run` on the host shows the raw result.

HealthCheck paused n *alerts after repeated failures. Re-enable & test in Alerts.*

— a Nextcloud notification, not a page: a channel died enough times to be switched off. Fix the cause, then **Re-enable & test** under **Alerts** (chapter 5).

11.2 Inside the health cards

The cards report platform state in plain sentences — **Could not reach the database.**, **Data directory is not configured.**, **Could not read disk space for the data directory.**, **Update cache could not be read.**, **Nextcloud reported errors while running background jobs.** — each is the probe's own finding and points at the layer to check first.

The HTTPS card has its own vocabulary: **The instance did not answer over HTTPS.** / **Could not reach the instance over HTTPS.** / **The instance did not answer in time over HTTPS.**, **The HTTPS certificate could not be verified. If this server uses a private certificate authority, trust it under Administration settings → Security.** (private CA: trust it under Administration settings → Security), **The instance hostname could not be resolved on this server.**, **Instance URL could not be parsed.**, **Instance URL is not available.**, **Internet connection is disabled for this instance.**. On a lab the expected answer is **Not using HTTPS** — decide per instance whether that is a finding or a shrug, and use the probe address fields (chapter 6) for split-horizon DNS.

11.3 On the Logs page

No log file yet. Nextcloud creates it when something is logged.
— nothing has been logged yet; `nextcloud.log` appears with the first line.

Cannot read the log file. Check permissions.
/ **The log file is not writable from here. Ask your host for file permissions.** — the web process cannot read (or rename) the file; fix ownership on the host.

Cannot change the log file. Check permissions. / **Could not delete the log file.** / **Cannot delete the log file. Check permissions.** / **Could not create a new empty log file. Check permissions.** / **Could not start a fresh log.** / **Could not delete this log copy.** / **Could not download the log file.** / **Could not rename the log file. Check permissions.** / **Cannot delete this log copy. Check permissions.**
— all permission or filesystem races on the data directory; the suffix *Check permissions* is the fix.

This log file is too large to download here. Use SSH or your host panel, or start a fresh log.
— the download cap protects the web process; fetch the file off-band or rotate it first.

A rotated copy with that name already exists. Try again.
— two “start fresh” runs inside one second’s name window; simply retry.

Type the confirmation word exactly.
— the typed word did not match `START_FRESH` / `DELETE` / `DELETE_COPY`; copy the word character-for-character.

This log action is not available yet. Reload the page. If it still fails, disable and re-enable HealthCheck, then try again.
— the app was updated underneath an open page; reload, and if it persists disable and re-enable the app once.

Something went wrong while checking the log.
/ **Search failed. Try again.** / **Log is empty.** — transient read failures or a genuinely empty file; retry, then check file readability.

Enter something to search for. / **No matches for this search.**
— the search box talking: an empty pattern is refused, and an empty result means “not in the loaded window” — **Load older lines** widens it.

Could not copy. Select the text and copy manually.
— the clipboard API was unavailable (older browser, denied permission); the message itself tells you the fallback.

This log copy is no longer available. / **Switch to the current log first.**
— the copy you opened was deleted underneath you, or a destructive action was aimed at a non-current file; reload the picker.

This check could not run.
— a probe aborted on this server (missing platform capability); if it persists, turn that probe off under **Rules** (chapter 6).

11.4 While saving settings

Settings changed elsewhere — reload and try again.
— the optimistic-lock conflict of chapter 9: another save landed first. Reload, re-apply.

Save failed.
/ **Something went wrong. Try again.** / **The server could not be reached. Check your connection and try again.** — transport or a rejected save; the inline field errors (*Invalid mute pattern.*, **Unknown**, *Too many app admins.*, *Invalid access mode.*, *Open access is not allowed.*) name the precise problem where there is one.

Webhook URL
— the SSRF guard: the URL is not reachable, not HTTP(S), or points at a private address (chapter 5).

Unknown channel
— an API call named a channel outside `email|slack|webhook|notification`.

Could not turn on alerts. Try again.
/ **Test failed.** / **Email could not be sent. Check mail settings.** / **Webhook failed. Check the URL and try again.** — the test-and-enable pipeline telling you the proof failed; fix the address or the mailer, never the symptom.

11.5 At the door

Not authorized is a state, not a bug — the user is simply not entitled (chapter 8). **Only a Nextcloud admin can add or remove app admins.** on **People** is the same boundary seen from the inside.

11.6 When nothing here matches

1. Run `occ logcheck:status` — it prints the same status block the UI reads.
2. Run `occ logcheck:run` for one live iteration and `occ logcheck:test-channel <name>` for the failing channel.
3. Search **Logs** itself for `logcheck` — the watcher writes its own audit trail to the very file it watches.
4. Report via the **Help** menu in the navigation footer: **Report a problem** opens a pre-filled mail with diagnostics and no personal data, **Open GitHub Issues** the public tracker.

12 FAQ and “how do I…”

12.1 How do I…? — the quick index

…get notified when errors appear?
Prove a channel under **Alerts**, then switch **Watch log file** on — the full walkthrough is chapter 2.

…make alerts quieter or louder?
Severity and pacing chips under **Rules**; mutes and app filters behind **Advanced filters** — chapter 6.

…let a colleague watch too?
A Nextcloud admin adds them under **People** — chapter 7. For “they only need the mails”, a shared recipient address is the better answer — chapter 8.

…find out why a card is red?
Read its detail line, then open **Logs** — chapter 4; chapter 11 decodes every message.

…look at yesterday’s log?
Pick the rotated copy in the file picker — chapter 4.

…wipe the log after fixing things?
Start fresh log keeps a copy; **Delete log file** does not — chapter 4.

…route alerts to Slack or my own endpoint?
Slack & webhook fold on **Alerts** — chapter 5.

…get alerts on my phone?
In-app notifications reach the Nextcloud mobile apps’ bell — chapter 5.

…change who sees this app?
The app-admin list — chapter 7; there is no open mode by design (chapter 8).

…check HealthCheck from a shell?
`occ logcheck:status` — chapter 10.

…restore settings after a bad change?
`occ logcheck:upgrade-backup` snapshots — chapter 10.

…report a bug?
Help → **Report a problem** in the navigation footer — chapter 11.

12.2 Frequently asked questions

Watching is on — why is nothing arriving? The two most common causes: no channel is active (the checklist on **Health** says so), or every matching line was muted (chapter 6). A third classic: the digest pacing — **Quiet (60 min)** means an hour can pass between error and mail by design.

Why does the first digest contain old errors? The cursor starts at the end of the file the moment watching is turned on; a fresh install never replays history. If it *does* — for example after restoring a huge log — `occ logcheck:reset-cursor` fast-forwards to now (chapter 10).

Do muted lines still get logged? Yes — muting only shields the digest. The **Logs** page sees everything; evidence is never destroyed.

Can HealthCheck run on several app servers? Only when all of them share one log file. Per-server logs trigger the *different server* warning — chapter 11.

Does it work without cron? The pages and **Check again** do; background watching and digests need Nextcloud cron. The *stuck* callout tells you when it stopped ticking (chapter 11).

Why are the old settings URLs bookmarked somewhere still working? Legacy section addresses redirect instead of dying — chapter 8.

Is there a mobile app? No dedicated companion — the web UI is responsive, and **In-app notifications** ride the Nextcloud mobile apps (chapter 5).

What exactly is sent in a digest? Counts and fingerprints per error shape, up to a capped number of fingerprints per digest; log text only when excerpts are explicitly on (chapter 5).

13 Accessibility

HealthCheck follows the Check-suite accessibility floor — WCAG 2.1 AA — and this chapter is the map for everyone who does not use a mouse or does not see the screen the usual way.

13.1 Keyboard paths

Every workflow in this manual is keyboard-complete:

- **Tab order** follows the visual order: skip links first (*Skip to navigation*, *Skip to content*), then the sidebar, then the page.
- **Switches** are real checkboxes with `role=switch` — Space toggles them, no mouse required.
- **Chips** on **Rules** and the **Logs** level filter are radio groups: arrow keys move between options, not just Tab.
- **The people picker** is a WAI-ARIA combobox: type to search, *ArrowDown*/*ArrowUp* move the highlight, *Enter* picks, *Escape* closes the list.
- **Dialogs** (the typed confirmations in the **Danger zone**) are native `<dialog>` elements: focus lands inside, *Escape* cancels, focus returns to the button that opened them.
- ***More* folds** are `<details>` blocks — they open and close with *Enter*/*Space* on the summary.
- On narrow screens the navigation opens via **Menu**, traps focus while open and closes with *Escape*.

13.2 What a screen reader announces

- State badges pair their colour dot with real text — **Watching** is a word, not a hue; the state names are also announced as *OK*, *Needs attention*, *Critical*, *Unknown*.
- The log viewer is a `role=log` live region — appended lines are announced without stealing focus, and the status line under it (*Lines shown:* n) is a `role=status`.
- Callouts carry `role=status` or `role=alert` depending on urgency, so warnings surface without a visual scan.
- Field errors are inline, referenced with `aria-describedby` and marked `aria-invalid` — the message is announced where the mistake is, not in a banner far away.
- Icons are decorative (`aria-hidden`); every icon button has an accessible name — the × on a person chip announces *Remove* name.

13.3 Contrast, targets and zoom

- Text and controls meet AA contrast in light and dark themes; the app uses Nextcloud's colour variables, so a user theme or high-contrast mode propagates instead of being overridden.
- Interactive targets are at least 44 × 44 px (48 for primary and danger actions) — chips and switches are comfortably tappable.
- At 320 px width nothing scrolls horizontally; the sidebar folds into **Menu** and the pages stack.
- 200 % text zoom keeps every control reachable; nothing is sized in a way that traps a zoomed layout.

13.4 Motion and forced colours

Animations are decorative-only and honour the reduced-motion preference; nothing that matters moves. In forced-colours (Windows high contrast) the badge dots and switches keep visible state through system colours and text — colour is never the only signal anywhere in the app.

13.5 Known limits

Screen-reader announcements of the live log tail can be chatty during a busy incident — the region batches additions politely, but **Show raw lines** plus browser find is often the calmer path for long sessions. Report friction via **Help** → **Report a problem**.

14 Glossary

Terms a new admin meets here that the Nextcloud documentation never defined — UI labels are translated in the app itself and are not repeated.

Accumulator — the holding area where matching log lines collect between digests; emptied whenever a digest leaves.

App admin — the mid-tier role: full daily HealthCheck use, none of the Nextcloud-admin-only powers (chapter 1).

Channel — a delivery path for digests: email, Slack, webhook or in-app notification.

Coalescing — the pacing rule that lets at most one digest leave per interval; lines keep collecting meanwhile.

Cursor — the remembered read position in the log file; survives restarts, follows “start fresh” onto the new file.

Digest — one batched alert: counts and fingerprints for everything that matched since the last send.

Fingerprint — a signature of a line’s shape (level, app, message pattern) so identical repeats group into one counted entry.

Health check / probe — one of the seven diagnostics that paint the cards on **Health**; individually switchable under **Rules**.

Log backend — where Nextcloud writes its log; HealthCheck only supports `file`.

Older copy / rotated copy — a log file moved aside, by the server’s own rotation (`.1`) or by **Start fresh log** (`.lck-rotated-*`); never watched for alerts.

occ — Nextcloud’s command line on the host (`php occ …`).

SSRF — “server-side request forgery”: an URL field the server dials for you; why private webhook targets are opt-in.

Upgrade backup — a pre-update snapshot of the app’s tables, restorable via `occ` (chapter 10).

Watching — the background loop that tails the log between cron ticks; distinct from the *view* filters, which never touch alerting.

15 Catalog: every surface and where it is documented

The lookup table for “I am staring at X — where is it explained?”. Every page, fold and command in the app, mapped to its chapter.

15.1 Pages and folds

Surface	Chapter
Health — summary badge, seven cards, **Check again**	3
Watching card — switch, checklist, CTAs	2, 3
Logs — file picker, chips, search, raw view	4
More menu on Logs — download, raw, remove copy	4
Danger zone + confirmation dialogs	4
Alerts — email card	5
Slack & webhook fold	5
More options — in-app notifications, excerpts, private webhooks	5
Rules — severity, pacing chips	6
Advanced filters — app allow/deny, regex mutes, app mutes	6
Checks on the Health page + HTTPS probe fields (NC admin)	6
People — explainer, combobox, chips	7
Support us — sponsors, enterprise, website	9
Not authorized page	8, 1
Menu drawer on narrow screens	2, 13
Help menu — report, suggest, GitHub	11

15.2 Addresses

URL	Lands on
`/apps/logcheck` and `/home`	**Health**
`/apps/logcheck/logs`	**Logs**
`/apps/logcheck/settings`	**Alerts** (default section)
`/settings/alerts`, `/rules`, `/people`, `/support`	the matching section
`/settings/status`	**Health** (legacy redirect)
`/settings/channels`, `/privacy`	**Alerts** (legacy)
`/settings/watch`	**Rules** (legacy)
`/settings/access`	**People** (legacy)

15.3 Commands

occ command	Chapter
`logcheck:status`, `logcheck:run`	10
`logcheck:test-channel <channel>`	10, 5
`logcheck:reset-cursor`	10
`logcheck:upgrade-backup list|create|restore`	10

15.4 Background machinery

Log-watch background job, the notification sender, the pre-update backup repair step and the uninstall cleanup are covered in chapters 2, 5 and 10 — they have no screens of their own by design.

About Software by Design

Software by Design builds Nextcloud apps for organisations that want to host their own data — from time tracking and projects to inventory, fleet management and customer relations. Every app runs entirely on your Nextcloud instance; the web interfaces are free (AGPL).

Company

- **Website:** https://nextcloud.software-by-design.de/
- **Partner & quotes:** mailto:info@software-by-design.de
- **Report a bug / suggest an improvement:** mailto:dev@software-by-design.de

Support & how to contribute

HealthCheck remains free on your Nextcloud (AGPL). Bug reports and sensible ideas via GitHub or email are part of open-source maintenance — without a guaranteed response time.

In the app

Under **Settings** → **Help & support** (navigation chip may still read **Support us**) you find the same options as on our website: partner quote, onboarding, commissioned feature and voluntary support.

Check Partner (invoiceable)

If your organisation needs predictable help with an invoice — setup, hour packages, prioritised email — choose **Check Partner**. That is a professional service with quote and invoice, not a donation.

- Packages and terms on the support page: https://nextcloud.software-by-design.de/en/support/
- Request a partner quote: mailto:info@software-by-design.de?subject=Partner%20inquiry

Other invoiceable options

- **Onboarding & training** — remote setup or workshop for your team.
- **Commissioned feature** — scoped change with acceptance criteria and delivery date.
- **Official mobile & terminal licences** — where offered for HealthCheck; named seats with invoice.

GitHub Sponsors (voluntary)

Voluntary support without an invoice SLA — appreciated, but optional:
https://github.com/sponsors/aSoftwareByDesignRepository

Legal note

This manual does not replace legal, tax or compliance advice. Third-party product names and trademarks belong to their respective owners.

Index

Access, 13
Accessibility, 21
Alerts, 8
App admin, 1
App admins, 12
Backup, 16
Can’t watch, 17
Catalog, 22
Channels, 8
Check again, 4
Check cards, 4
Checklist, 3
Conflict, 15
Danger zone, 6
Digest, 10
Directory search, 12
Email, 8
Errors, 17
Export, 16
FAQ, 19
Filters, 10
Getting started, 3
Glossary, 22
Health, 4
Keyboard, 21
Logs, 6
Mute, 10
Nextcloud admin, 1
Not authorized, 13
occ, 16
People, 12
Restore, 16
Roles, 1
Rules, 10
Save, 15
Screen reader, 21
Search, 6
Security, 13
Settings, 15
Severity, 10
Slack, 8
Start fresh, 6
Troubleshooting, 17
Watch log file, 3
Webhook, 8