

System & maintenance commands



- [Admin delegation](#)
 - [admin-delegation:add](#)
 - [admin-delegation:remove](#)
 - [admin-delegation:show](#)
- [Background jobs](#)
 - [background:cron](#)
 - [background-job:delete](#)
 - [background-job:execute](#)
 - [background-job:history](#)
 - [background-job:list](#)
 - [background-job:running](#)
 - [background-job:worker](#)
- [Broadcast](#)
- [Info](#)
 - [info:file](#)
 - [info:file:space](#)
 - [info:storage](#)
 - [info:storages](#)
- [Logging commands](#)
 - [log:file](#)
 - [log:manage](#)
- [Maintenance commands](#)
 - [maintenance:data-fingerprint](#)
 - [maintenance:mimetype:update-db](#) and [maintenance:mimetype:update-js](#)
 - [maintenance:mode](#)
 - [maintenance:repair](#)

- [maintenance:repair-share-owner](#)
- [maintenance:theme:update](#)
- [maintenance:update:htaccess](#)
- [OAuth2](#)
- [Security](#)
 - [security:bruteforce:attempts](#)
 - [security:bruteforce:reset](#)
 - [security:certificates](#)
 - [security:certificates:export](#)
 - [security:certificates:import](#)
 - [security:certificates:remove](#)
- [Setup checks](#)
- [Share operations](#)
 - [share:list](#)
- [Snowflake IDs](#)
- [Status](#)
 - [Status return code](#)
- [Task processing](#)
 - [taskprocessing:task-type:set-enabled](#)
 - [taskprocessing:task:cleanup](#)
 - [taskprocessing:task:get](#)
 - [taskprocessing:task:list](#)
 - [taskprocessing:task:stats](#)
 - [taskprocessing:worker](#)
- [Theming](#)
 - [theming:config](#)
- [Webhook listeners](#)
- [Workflows](#)
 - [workflows:list](#)
- [Command line installation](#)
- [Command line upgrade](#)
- [Antivirus](#)
 - [files_antivirus:background-scan](#)
 - [files_antivirus:mark](#)

- [files_antivirus:scan](#)
- [files_antivirus:status](#)

The `occ` commands in this section cover server administration, background job management, and operational tasks. They let you configure logging and theming, manage background jobs and AI task processing, administer OAuth2 clients, delegation rules and workflows, check server status, and perform installation and upgrade procedures from the command line.

Admin delegation

The `admin-delegation` commands allow granting non-admin groups access to specific admin settings panels, without giving them full administrator privileges:

```
admin-delegation
  admin-delegation:add      add setting delegation to a group
  admin-delegation:remove  remove settings delegation from a group
  admin-delegation:show    show delegated settings
```

admin-delegation:add

Delegate an admin settings class to a group:

```
sudo -E -u www-data php occ admin-delegation:add \
  'OCA\Settings\Settings\Admin\Sharing' milliways
```

The `settingClass` argument must be the fully qualified PHP class name of a settings class that implements `IDelegatedSettings`. Common examples:

- `OCA\Settings\Settings\Admin\Sharing` — sharing administration
- `OCA\Settings\Settings\Admin\Users` — user management
- `OCA\Settings\Settings\Admin\Mail` — email configuration
- `OCA\Theming\Settings\Admin` — theming settings

admin-delegation:remove

Remove a settings delegation from a group:

```
sudo -E -u www-data php occ admin-delegation:remove \  
'OCA\Settings\Settings\Admin\Sharing' milliways
```

admin-delegation:show

Show all currently configured settings delegations:

```
sudo -E -u www-data php occ admin-delegation:show
```

Use `--output=json_pretty` for machine-readable output.

Background jobs

The `background-job` commands let you inspect and execute individual background jobs. The separate `background:cron`, `background:ajax`, and `background:webcron` commands configure how the background job scheduler is triggered:

```
background  
background:ajax      use ajax to run background jobs  
background:cron      use cron to run background jobs  
background:webcron   use webcron to run background jobs  
background-job  
background-job:delete Remove a background job from database  
background-job:execute Execute a single background job manually  
background-job:history Show past jobs  
background-job:list   List background jobs  
background-job:running Show currently running jobs  
background-job:worker Run a background job worker
```

background:cron

Set the background job execution mode to cron. Nextcloud recommends the system cron mode for production instances:

```
sudo -E -u www-data php occ background:cron  
Set mode for background jobs to 'cron'
```

Use `background:ajax` to switch to the in-browser AJAX scheduler or `background:webcron` for an external webcron service instead.

See [../configuration_server/background_jobs_configuration](#) for a full guide to configuring the background job scheduler.

background-job:delete

Remove a background job from the database. The command shows the job details and asks for confirmation:

```
sudo -E -u www-data php occ background-job:delete 42
Job class: OCA\Files\BackgroundJob\ScanFiles
Arguments: []

Do you really want to delete this background job? It could create some misbehaviours in
Nextcloud. (y/N)
```

“ ”

Deleting a background job can cause Nextcloud features to stop working correctly. Only delete jobs that you know are safe to remove, such as duplicate or orphaned entries.

background-job:execute

Execute a single background job by its database ID:

```
sudo -E -u www-data php occ background-job:execute 42
```

The command shows the job's class, type, last run time, and next scheduled execution before running it. If the job was recently run and is not yet due, it may be skipped; use `--force-execute` to run it regardless:

```
sudo -E -u www-data php occ background-job:execute --force-execute 42
```

background-job:history

Show job history with their duration and peak memory usage:

```
sudo -E -u www-data php occ background-job:history -l 5
```

Run ID	Status	Class	Started		
at	Server ID	PID	Duration	Memory usage	
98109593040965632	Succeeded	OC\Command\CommandJob			2026-06-22
09:15:19	30	3847229	284 ms	251.6 MB	
98109592927719424	Succeeded	OC\FilesMetadata\Job\UpdateSingleMetadata			2026-06-22
09:15:19	30	3847229	16 ms	222 MB	
98109592852221952	Succeeded	OC\FilesMetadata\Job\UpdateSingleMetadata			2026-06-22
09:15:19	30	3847229	8 ms	221.4 MB	
98109588267847680	Succeeded	OC\Command\CommandJob			2026-06-22
09:15:18	30	3847229	909 ms	293 MB	
97985889682313216	Crashed	OCA\Text\Cron\Cleanup			2026-06-22
01:15:17	30	3733554	594627 ms	0 B	

Use `-l/--limit` to control how many jobs are shown (default: 200), one or multiple `-c/--class` to filter by job class, and one or multiple `-s/--status` to filter by status (0: running, 1: succeeded, 2: failed, 3: crashed).

background-job:list

List all background jobs registered in the database:

```
sudo -E -u www-data php occ background-job:list
```

id	class	last_run
argument		
1	OCA\Files\BackgroundJob\ScanFiles	2025-06-23T10:00:00+00:00
[]		
2	OCA\Activity\BackgroundJob\DigestMail	2025-06-23T08:00:00+00:00

```

[]          |
| 3  | 0C\Share20\SharesReminderJob          | 2025-06-23T07:00:00+00:00 |
[]          |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+

```

Use `-c` / `--class` to filter by job class, `-l` / `--limit` to control how many jobs are shown (default: 500), and `-o` / `--offset` to page through results.

background-job:running

List all currently running jobs:

```

sudo -E -u www-data php occ background-job:running
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| Run ID          | Class                  | Started at          | Server ID | PID      |
Running since |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
| 98108257692012544 | 0C\Command\CommandJob | 2026-06-22 09:10:08 | 30         | 3845056 | 1
seconds      |
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+

```

Use `-l` / `--limit` to limit the number of job shown (default: 200).

background-job:worker

Run a continuous background job worker. This is an alternative to system cron for environments where you want to run background jobs from a persistent process rather than a periodic cron invocation:

```

sudo -E -u www-data php occ background-job:worker

```

By default the worker runs indefinitely, polling for new jobs every second.

Use `--once` to process at most one job then exit (equivalent to a single cron run):

```

sudo -E -u www-data php occ background-job:worker --once

```

Use `--stop_after` to set a maximum runtime before the worker exits cleanly (the currently running job is allowed to finish). Accepts seconds, or values like `30s`, `10m`, `2h`:

```
sudo -E -u www-data php occ background-job:worker --stop_after 1h
```

Use `--interval` to set the poll interval in seconds (default: 1). Set to `0` to process jobs once and exit if none are found:

```
sudo -E -u www-data php occ background-job:worker --interval 5
```

Pass one or more job class names to process only jobs of those types:

```
sudo -E -u www-data php occ background-job:worker \  
'OCA\Files\BackgroundJob\ScanFiles'
```

Broadcast

Send a test Server-Sent Events (SSE) broadcast to verify that real-time push notifications are working:

```
sudo -E -u www-data php occ broadcast:test layla
```

Pass a custom event name as the second argument (default: `test`):

```
sudo -E -u www-data php occ broadcast:test layla my-event
```

Info

The `info` commands display detailed information about files and storages. They are useful for debugging file access problems, investigating storage layout, and identifying space usage:

```
info  
info:file          get information for a file  
info:file:space    summarize space usage of a folder  
info:storage       get information for a single storage  
info:storages      list storages ordered by file count
```

info:file

Show detailed information for a file or folder identified by file ID or path:

```
sudo -E -u www-data php occ info:file /layla/files/Documents/report.pdf

  fileId: 12345
  mimeType: application/pdf
  modified: 2025-06-01T14:30:00+00:00
  encrypted: false
  size: 204800
  eTag: abc123
  permissions: 27
  users with access: layla
```

Use `-c` / `--children` to list the immediate contents of a folder. Use `--storage-tree` to display the storage and cache wrapping tree.

info:file:space

Show how much space a folder uses, with a ranked breakdown of the largest items:

```
sudo -E -u www-data php occ info:file:space /layla/files/Videos

Total: 14.7 GB

+-----+-----+
| Path                | Size  |
+-----+-----+
| lecture-2025.mp4    | 8.2 GB |
| demo-video.mp4      | 4.1 GB |
+-----+-----+
```

Use `-c` / `--count` to change the number of items shown (default: 25), or `-a` / `--all` to show all items.

info:storage

Show information for a single storage by its numeric storage ID:

```
sudo -E -u www-data php occ info:storage 5
```

info:storages

List all storages ordered by the number of files they contain:

```
sudo -E -u www-data php occ info:storages
```

Use `-c` / `--count` to limit the number of storages shown (default: 25), or `-a` / `--all` to list all storages.

Logging commands

These commands view and configure your Nextcloud logging preferences:

```
log
log:file    manipulate Nextcloud logging backend
log:manage  manage logging configuration
log:tail    tail the Nextcloud logfile [requires app "Log Reader" to be enabled]
log:watch   watch the Nextcloud logfile live [requires app "Log Reader" to be enabled]
```

log:file

Show the current logging status:

```
sudo -E -u www-data php occ log:file
Log backend Nextcloud: enabled
Log file: /opt/nextcloud/data/nextcloud.log
Rotate at: disabled
```

Use `--enable` to turn on the Nextcloud log backend, `--file` to set a different log file path, and `--rotate-size` to rotate the log at a given file size in bytes (`0` disables rotation).

log:manage

Set the logging backend, log level, and timezone. The defaults are `file`, `warning`, and `UTC`:

```
sudo -E -u www-data php occ log:manage --backend file --level warning --timezone UTC
Log backend: file
```

Log level: Warning (2)

Log timezone: UTC

Options:

- `--backend` — one of: `file`, `syslog`, `errorlog`, `systemd`
- `--level` — one of: `debug`, `info`, `warning`, `error`, `fatal`
- `--timezone` — a PHP timezone name; see <https://www.php.net/manual/en/timezones.php>

Maintenance commands

Use these commands when you upgrade Nextcloud, perform backups, or carry out other tasks that require locking users out temporarily:

```
maintenance
maintenance:data-fingerprint  update the systems data-fingerprint after a backup is
restored
maintenance:mimetype:update-db update database mimetypes and update filecache
maintenance:mimetype:update-js update mimetypeelist.js
maintenance:mode              set maintenance mode
maintenance:repair            repair this installation
maintenance:repair-share-owner fix some shares owner if it fell out of sync
maintenance:theme:update      apply custom theme changes
maintenance:update:htaccess   update the .htaccess file
```

maintenance:data-fingerprint

After restoring a backup of your data directory or the database, run this command once. It updates the ETag for all files, allowing sync clients to detect that files were modified:

```
sudo -E -u www-data php occ maintenance:data-fingerprint
```

maintenance:mimetype:update-db and maintenance:mimetype:update-js

Update the Nextcloud database and file cache with changed mimetypes from `config/mimotypemapping.json`. Run after modifying that file. Pass `--repair-filecache` to apply the

change to existing files:

```
sudo -E -u www-data php occ maintenance:mimetype:update-db --repair-filecache
```

`maintenance:mimetype:update-js` regenerates the client-side mimetype list:

```
sudo -E -u www-data php occ maintenance:mimetype:update-js
```

maintenance:mode

Lock the sessions of all logged-in users, including administrators, and display a status screen warning that the server is in maintenance mode. Users who are not already logged in cannot log in until maintenance mode is turned off. When the server comes out of maintenance mode, logged-in users must refresh their browser to continue working:

```
sudo -E -u www-data php occ maintenance:mode --on
```

```
Maintenance mode enabled
```

```
sudo -E -u www-data php occ maintenance:mode --off
```

```
Maintenance mode disabled
```

maintenance:repair

Runs automatically during upgrades to clean up the database. You can also run it manually if needed:

```
sudo -E -u www-data php occ maintenance:repair
```

maintenance:repair-share-owner

Fix share owner records that have fallen out of sync with the actual file ownership:

```
sudo -E -u www-data php occ maintenance:repair-share-owner
```

maintenance:theme:update

Run when icons in a custom theme are not updating correctly. This rebuilds the mimetype list and clears the image cache:

```
sudo -E -u www-data php occ maintenance:theme:update
```

maintenance:update:htaccess

Regenerate the `.htaccess` file from the current configuration. Run this after changing URL rewriting settings or after an upgrade if the file appears outdated:

```
sudo -E -u www-data php occ maintenance:update:htaccess
```

OAuth2

“ ”

This command is only available when `'oauth2.enable_oc_clients' => true` is set in `config/config.php`. It is intended for ownCloud-to-Nextcloud migrations only.

Import an OAuth2 client record from an ownCloud database during a migration. The `client-id` and `client-secret` arguments come directly from the `oc_oauth2_clients` table in the ownCloud database:

```
sudo -E -u www-data php occ oauth2:import-legacy-oc-client \  
<client-id> <client-secret>
```

Security

Use these commands to manage server-wide security parameters, including [Brute force protection](#) and trusted SSL certificates. Trusted certificates are useful when creating federation connections with servers that use self-signed certificates:

```
security  
security:bruteforce:attempts  show bruteforce attempts status for a given IP address  
security:bruteforce:reset     reset bruteforce attempts for a given IP address  
security:certificates         list trusted certificates  
security:certificates:export  export the certificate bundle
```

```
security:certificates:import  import trusted certificate
security:certificates:remove  remove trusted certificate
```

security:bruteforce:attempts

Show the current bruteforce throttle status for an IP address, including the number of recorded attempts and the current delay being applied. Optionally filter by action name:

```
sudo -E -u www-data php occ security:bruteforce:attempts 192.168.1.100
sudo -E -u www-data php occ security:bruteforce:attempts 192.168.1.100 login
```

Use `--output=json_pretty` for machine-readable output.

security:bruteforce:reset

Clear all recorded bruteforce attempts for an IP address, removing any login delay immediately:

```
sudo -E -u www-data php occ security:bruteforce:reset 192.168.1.100
```

security:certificates

List all trusted certificates installed on the server:

```
sudo -E -u www-data php occ security:certificates

+-----+-----+-----+-----+-----+
| File Name           | Common Name | Organization   | Valid Until | Issued By |
+-----+-----+-----+-----+-----+
| myserver.crt        | myserver   | My Org         | 2026-01-01  | My CA     |
+-----+-----+-----+-----+-----+
```

Use `--output=json_pretty` for machine-readable output.

security:certificates:export

Export the full certificate bundle to stdout. Useful for backup or inspection:

```
sudo -E -u www-data php occ security:certificates:export
```

security:certificates:import

Install a trusted certificate from a file. The certificate name is derived from the filename:

```
sudo -E -u www-data php occ security:certificates:import /path/to/certificate.crt
```

security:certificates:remove

Remove a trusted certificate by its file name (as shown in `security:certificates`):

```
sudo -E -u www-data php occ security:certificates:remove my-certificate
```

Setup checks

Run the setup checks from the command line to verify your installation configuration:

```
sudo -E -u www-data php occ setupchecks
```

Example output:

```
dav:
  ✓ DAV system address book: No outstanding DAV system address book sync.
network:
  ✓ WebDAV endpoint: Your web server is properly set up to allow file synchronization over WebDAV.
  ✓ Data directory protected
  ✓ Internet connectivity
...
```

Use `--output=json_pretty` for machine-readable output suitable for automated monitoring.

Use `--class=CLASS_NAME` to run only the check for the setup check with the given class name.

Use `--category=CATEGORY` to run only the checks for the given category, e.g. `network` or `system`.

Share operations

Available `occ` commands for the `share` namespace:

```
share
share:list  list available shares
```

share:list

List all shares on the system:

```
sudo -E -u www-data php occ share:list
```

Filter by owner, recipient, or the user who created the share:

```
sudo -E -u www-data php occ share:list --owner layla
sudo -E -u www-data php occ share:list --recipient fred
sudo -E -u www-data php occ share:list --by layla
```

Filter by a specific file path:

```
sudo -E -u www-data php occ share:list --file "/layla/files/Documents/report.pdf"
```

Filter by a folder; use `--recursive` to include shares nested anywhere inside it:

```
sudo -E -u www-data php occ share:list --parent "/layla/files/Projects" --recursive
```

Filter by share type (one of `user`, `group`, `link`, `email`, `remote`, `room`, `deck`) or by share status:

```
sudo -E -u www-data php occ share:list --type link
sudo -E -u www-data php occ share:list --status 0
```

Snowflake IDs

Nextcloud uses Snowflake IDs for unique identifiers in several subsystems. Decode a Snowflake ID to inspect its embedded timestamp and metadata:

```
sudo -E -u www-data php occ snowflake:decode 6768789079123765868
```

Example output:

```
+-----+-----+
| Snowflake ID      | 6768789079123765868 |
| Seconds           | 1575981518           |
| Milliseconds      | 50                    |
| Created from CLI  | no                    |
| Server ID         | 441                   |
| Sequence ID       | 2668                  |
| Creation timestamp | 1575981518.050        |
| Creation date     | 2019-12-10 11:18:38.050 |
+-----+-----+
```

Status

Use the `status` command to retrieve information about the current installation:

```
sudo -E -u www-data php occ status
- installed: true
- version: 30.0.0.0
- versionstring: 30.0.0
- edition:
- maintenance: false
- needsDbUpgrade: false
- productname: Nextcloud
- extendedSupport: false
```

Use `--output=json_pretty` for machine-readable output:

```
sudo -E -u www-data php occ status --output=json_pretty
{
  "installed": true,
  "version": "30.0.0.0",
  "versionstring": "30.0.0",
  "edition": "",
  "maintenance": false,
  "needsDbUpgrade": false,
  "productname": "Nextcloud",
  "extendedSupport": false
}
```

Status return code

Use the `-e` flag to get a machine-readable exit code reflecting the installation state. There is no output by default, making it suitable for scripts, monitoring checks, and systemd units:

```
sudo -E -u www-data php occ status -e
echo $?
0

sudo -E -u www-data php occ maintenance:mode --on
Maintenance mode enabled

sudo -E -u www-data php occ status -e
echo $?
1

sudo -E -u www-data php occ maintenance:mode --off
Maintenance mode disabled

sudo -E -u www-data php occ status -e
echo $?
0
```

Return code	Description
0	normal operation
1	maintenance mode is enabled; the instance is currently unavailable to users
2	<code>occ upgrade</code> is required

Task processing

The `taskprocessing` commands manage AI and background task processing jobs. Task processing provides infrastructure for AI features such as text generation, image classification, and speech-to-text:

```
taskprocessing
taskprocessing:task-type:set-enabled enable or disable a task type
taskprocessing:task:cleanup      cleanup old tasks
taskprocessing:task:get          display all information for a specific task
taskprocessing:task:list        list tasks
taskprocessing:task:stats        get statistics for tasks
```

taskprocessing:worker
providers

run a dedicated worker for synchronous TaskProcessing

taskprocessing:task-type:set-enabled

Enable or disable a task type:

```
sudo -E -u www-data php occ taskprocessing:task-type:set-enabled core:text2text 1  
sudo -E -u www-data php occ taskprocessing:task-type:set-enabled core:text2text 0
```

taskprocessing:task:cleanup

Delete old task records and their associated output files. By default, tasks older than four months are removed:

```
sudo -E -u www-data php occ taskprocessing:task:cleanup
```

Pass a maximum age in seconds to override the default:

```
sudo -E -u www-data php occ taskprocessing:task:cleanup 2592000
```

taskprocessing:task:get

Display all information for a specific task by its numeric ID:

```
sudo -E -u www-data php occ taskprocessing:task:get 42
```

taskprocessing:task:list

List task processing tasks:

```
sudo -E -u www-data php occ taskprocessing:task:list
```

Filter by user, type, app, or custom ID:

```
sudo -E -u www-data php occ taskprocessing:task:list --userIdFilter layla  
sudo -E -u www-data php occ taskprocessing:task:list --type core:text2text
```

Filter by status (0=UNKNOWN, 1=SCHEDULED, 2=RUNNING, 3=SUCCESSFUL, 4=FAILED, 5=CANCELLED):

```
sudo -E -u www-data php occ taskprocessing:task:list --status 4
```

Other available filters: `--appId`, `--customId`, `--scheduledAfter`, `--endedBefore`.

taskprocessing:task:stats

Show statistics for tasks (max/average running time, queuing time, and input/output sizes).

Accepts the same filter options as `taskprocessing:task:list`:

```
sudo -E -u www-data php occ taskprocessing:task:stats
```

taskprocessing:worker

Run a dedicated worker that processes tasks from synchronous TaskProcessing providers. Use this when you want to offload AI task execution to a separate process:

```
sudo -E -u www-data php occ taskprocessing:worker
```

Use `--once` to process at most one task then exit. Use `--timeout` to set a maximum runtime in seconds (default: `0` = run indefinitely). Use `--taskTypes` (repeatable) to restrict the worker to specific task type IDs:

```
sudo -E -u www-data php occ taskprocessing:worker \  
--taskTypes core:text2text --once
```

Theming

The Theming app (`theming`) is always enabled. The `theming:config` command lets you view and update theming settings without logging into the web interface:

```
theming  
theming:config set theming app config values
```

theming:config

With no arguments, show all current theming values:

```
sudo -E -u www-data php occ theming:config  
Current theming config:  
name: Nextcloud  
url: https://nextcloud.com  
slogan: a safe home for all your data  
...
```

Show the current value of a single key:

```
sudo -E -u www-data php occ theming:config name
```

Set a value:

```
sudo -E -u www-data php occ theming:config name "Acme Cloud"  
sudo -E -u www-data php occ theming:config url "https://acme.example.com"  
sudo -E -u www-data php occ theming:config slogan "Secure file sync for Acme"  
sudo -E -u www-data php occ theming:config primary_color "#0082c9"  
sudo -E -u www-data php occ theming:config background_color "#00679e"  
sudo -E -u www-data php occ theming:config disable-user-theming true
```

To set an image, pass the path to the image file as the value:

```
sudo -E -u www-data php occ theming:config logo /path/to/logo.png  
sudo -E -u www-data php occ theming:config favicon /path/to/favicon.ico  
sudo -E -u www-data php occ theming:config background /path/to/background.jpg
```

Reset a key to its default:

```
sudo -E -u www-data php occ theming:config --reset name
```

Supported text keys: `name`, `url`, `imprintUrl`, `privacyUrl`, `slogan`, `primary_color`, `background_color`, `disable-user-theming`.

Supported image keys: `background`, `logo`, `logoheader`, `favicon`.

Webhook listeners





The Webhook listeners app (`webhook_listeners`) is shipped with Nextcloud but not enabled by default. Enable it in the Apps menu before using these commands.

List all webhook listener configurations registered on the server. Each entry shows the listener ID, the user or app that registered it, the HTTP method and target URI, the Nextcloud event it listens for, any event filter, and the configured authentication method (`none` or `header`):

```
sudo -E -u www-data php occ webhook_listeners:list
```

Use `--output=json_pretty` for machine-readable output, which also includes the full header and authentication data fields:

```
sudo -E -u www-data php occ webhook_listeners:list --output=json_pretty
```

Webhook listeners are configured through the Admin settings UI or the REST API. This command is read-only — it provides an audit view of what listeners are active without needing to access the web interface.

Workflows

The Workflow engine (`workflowengine`) is always enabled:

```
workflows
workflows:list  list configured workflows
```

workflows:list

List configured workflow rules. Use the optional `scope` argument to filter by scope; `admin` (default) or `user`:

```
sudo -E -u www-data php occ workflows:list
sudo -E -u www-data php occ workflows:list user
sudo -E -u www-data php occ workflows:list user layla
```

The output is a JSON representation of the configured workflow operations.

Command line installation

These commands are only available before Nextcloud has been installed, after you have unpacked the archive and copied Nextcloud into the appropriate directories.

Display the available installation options:

```
$ sudo -E -u www-data php /var/www/nextcloud/occ maintenance:install --help
Nextcloud is not installed - only a limited number of commands are available

Usage:
  maintenance:install [options]

Options:
  --database[=DATABASE]                Supported database type [default: "sqlite"]
  --database-name[=DATABASE-NAME]      Name of the database
  --database-host[=DATABASE-HOST]      Hostname of the database [default: "localhost"]
  --database-port[=DATABASE-PORT]      Port the database is listening on
  --database-user[=DATABASE-USER]      Login for database connection
  --database-pass[=DATABASE-PASS]      Password of the database login
  --database-table-space[=DATABASE-TABLE-SPACE]
                                         Table space of the database (`oci` only)
  --database-ssl-mode[=DATABASE-SSL-MODE]
                                         Encryption mode for the database connection,
                                         e.g. "require" or "verify-full" (`pgsql`
only)
  --database-ssl-ca[=DATABASE-SSL-CA]  Path to the CA certificate the database server
                                         is verified against (`mysql` and `pgsql`
only)
  --database-ssl-cert[=DATABASE-SSL-CERT]
                                         Path to the client certificate used to
                                         authenticate against the database
                                         (`mysql` and `pgsql` only)
  --database-ssl-key[=DATABASE-SSL-KEY] Path to the private key of the client certificate
                                         (`mysql` and `pgsql` only)
  --database-ssl-crl[=DATABASE-SSL-CRL] Path to the certificate revocation list
                                         (`pgsql` only)
  --database-ssl-no-verify              Do not verify that the database server
certificate
```

	matches the hostname used to connect (<code>`mysql`</code> only)
<code>--disable-admin-user</code>	Disable the creation of an administrator login
<code>--admin-user[=ADMIN-USER]</code>	Login for initial administrator account [default: "admin"]
<code>--admin-pass[=ADMIN-PASS]</code>	Password for initial administrator login
<code>--admin-email[=ADMIN-EMAIL]</code>	E-Mail to associate with the initial administrator login
<code>--data-dir[=DATA-DIR]</code>	Path to data directory
<code>--password-salt[=PASSWORD-SALT]</code>	Password salt; generated if not provided
(ADVANCED)	
<code>--server-secret[=SERVER-SECRET]</code>	Server secret; generated if not provided
(ADVANCED)	

This example installs Nextcloud with a MySQL database. `occ` prompts for the database and administrator passwords:

```
$ sudo -E -u www-data php /var/www/nextcloud/occ maintenance:install \
--database mysql \
--database-name nextcloud \
--database-host 127.0.0.1 \
--database-user nextcloud \
--admin-user admin
What is the password to access the database with user <nextcloud>?
What is the password you like to use for the admin account <admin>?
Nextcloud was successfully installed
```

Supported databases:

- `sqlite` — SQLite (not recommended for production usage)
- `mysql` — MySQL or MariaDB
- `pgsql` — PostgreSQL
- `oci` — Oracle; contact [Nextcloud GmbH](#) for enterprise support

🔒 *New in version 35:* The `--database-ssl-*` options set up an SSL/TLS encrypted connection to the database, see [Encrypted database connection](#).

Command line upgrade

These commands are available after downloading an upgraded package or tarball, and before completing the upgrade.

“ ”

`occ upgrade` only runs the **migration phase**: database schema updates and app upgrades. It does **not** download or replace Nextcloud's code files.

Before running `occ upgrade` you must first replace the code using one of:

- The built-in updater: `sudo -E -u www-data php /var/www/nextcloud/updater/updater.phar`
- A manually extracted tarball (see [../maintenance/manual_upgrade](https://nextcloud.com/maintenance/manual_upgrade))

If `occ upgrade` reports "*Nextcloud is up to date*" when you expect an upgrade to run, the code files have not been replaced yet — run the updater or extract the new tarball first.

See [../maintenance/upgrade](https://nextcloud.com/maintenance/upgrade) for the full upgrade process.

When performing an upgrade, use `occ upgrade` instead of the web-based upgrader to avoid PHP timeout limits (the web interface enforces a 3600-second limit). On large installations this may not be sufficient, leaving the system in an inconsistent state.

After completing all preliminary steps (see [../maintenance/upgrade](https://nextcloud.com/maintenance/upgrade)), run the upgrade:

```
sudo -E -u www-data php occ upgrade

Nextcloud or one of the apps require upgrade - only a limited number of
commands are available
Turned on maintenance mode
Checked database schema update
Checked database schema update for apps
Updated database
Updating <gallery> ...
Updated <gallery> to 0.6.1
Updating <activity> ...
Updated <activity> to 2.1.0
Update successful
```

Turned off maintenance mode

Use `-v` to display timestamps on each step:

```
sudo -E -u www-data php occ upgrade -v
2025-06-23T09:06:15+0000 Turned on maintenance mode
2025-06-23T09:06:15+0000 Checked database schema update
2025-06-23T09:06:15+0000 Checked database schema update for apps
2025-06-23T09:06:15+0000 Updated database
2025-06-23T09:06:15+0000 Updated <files_sharing> to 0.6.6
2025-06-23T09:06:15+0000 Update successful
2025-06-23T09:06:15+0000 Turned off maintenance mode
```

If an error occurs, the exception is logged in the Nextcloud log file:

```
Turned on maintenance mode
Checked database schema update
Checked database schema update for apps
Updated database
Updating <files_sharing> ...
Exception
ServerNotAvailableException: LDAP server is not available
Update failed
Turned off maintenance mode
```

Antivirus



These commands require the [files_antivirus](#) app to be installed and enabled.

```
files_antivirus
files_antivirus:background-scan  run the background scan
files_antivirus:mark             mark a file as scanned or unscanned
files_antivirus:scan             scan a file
files_antivirus:status           show antivirus scanner status
```

files_antivirus:background-scan

Trigger the background antivirus scan manually. By default all pending files are processed. Use `--max` to limit the number of files scanned in a single run:

```
sudo -E -u www-data php occ files_antivirus:background-scan
sudo -E -u www-data php occ files_antivirus:background-scan --max 500
```

files_antivirus:mark

Mark a file as scanned or unscanned. The `file` argument is the path to the file and `mode` is either `scanned` or `unscanned`:

```
sudo -E -u www-data php occ files_antivirus:mark /layla/files/report.pdf scanned
sudo -E -u www-data php occ files_antivirus:mark /layla/files/report.pdf unscanned
```

Use `--forever` (`-f`) when marking a file as scanned to permanently exclude it from future rescans:

```
sudo -E -u www-data php occ files_antivirus:mark --forever /layla/files/report.pdf scanned
```

files_antivirus:scan

Scan a single file immediately and report the result:

```
sudo -E -u www-data php occ files_antivirus:scan /layla/files/report.pdf
```

Use `--debug` to enable verbose backend output, useful when diagnosing scanner connectivity issues:

```
sudo -E -u www-data php occ files_antivirus:scan --debug /layla/files/report.pdf
```

files_antivirus:status

Show the current antivirus scanner status, including the configured backend and whether the scanner is reachable:

```
sudo -E -u www-data php occ files_antivirus:status
```

Use `-v` for additional backend detail.

📁 #3

📁 cocobook-sync 📁 2026-10-28 16:27:31 UTC

📁 cocobook-sync 📁 2026-10-29 01:09:18 UTC