

User & group commands



- [User commands](#)
 - [user:add](#)
 - [user:resetpassword](#)
 - [user:delete](#)
 - [user:disable and user:enable](#)
 - [user:lastseen](#)
 - [user:list](#)
 - [user:info](#)
 - [user:profile](#)
 - [user:setting](#)
 - [user:report](#)
 - [user:auth-tokens:list](#)
 - [user:auth-tokens:add](#)
 - [user:auth-tokens:delete](#)
 - [user:clear-avatar-cache](#)
 - [user:keys:verify](#)
 - [user:sync-account-data](#)
 - [user:welcome](#)
- [Group commands](#)
 - [group:add](#)
 - [group:adduser and group:removeuser](#)
 - [group:delete](#)
 - [group:list](#)
 - [group:info](#)
- [Two-factor authentication](#)
 - [twofactorauth:disable and twofactorauth:enable](#)
 - [twofactorauth:enforce](#)
 - [twofactorauth:state](#)
 - [twofactorauth:cleanup](#)

- [System Tags](#)
 - [tag:list](#)
 - [tag:add](#)
 - [tag:edit](#)
 - [tag:delete](#)
 - [Assigning tags to files](#)
-

User commands

The `user` commands create and manage user accounts, reset passwords, manage authentication tokens, and report on user activity:

<code>user</code>	
<code>user:add</code>	adds a user
<code>user:add-app-password</code>	(deprecated) alias for <code>user:auth-tokens:add</code>
<code>user:auth-tokens:add</code>	add an app password for an account
<code>user:auth-tokens:delete</code>	delete an authentication token
<code>user:auth-tokens:list</code>	list authentication tokens for an account
<code>user:clear-avatar-cache</code>	clear avatar cache
<code>user:delete</code>	deletes the specified user
<code>user:disable</code>	disables the specified user
<code>user:enable</code>	enables the specified user
<code>user:info</code>	show information about a user
<code>user:keys:verify</code>	verify the stored public key matches the stored private key
<code>user:lastseen</code>	show when a user was last logged in
<code>user:list</code>	list all registered users
<code>user:profile</code>	read and modify user profile data
<code>user:report</code>	show how many users have access
<code>user:resetpassword</code>	reset the password for a user
<code>user:setting</code>	read and modify user settings
<code>user:sync-account-data</code>	sync user backend data to the accounts table
<code>user:welcome</code>	send the welcome email to a user

user:add

Create a new user with a display name, login name, and optional group memberships:

```
user:add [--password-from-env] [--generate-password] [--display-name=["..."]] [-g|--group=["..."]] [--email EMAIL] uid
```

The `display-name` corresponds to the **Full Name** on the Users page in your Nextcloud web interface, and the `uid` is their **Username** (login name). Any groups that do not exist are created automatically:

```
sudo -E -u www-data php occ user:add --display-name="Layla Smith" \
  --group="users" --group="db-admins" layla
Enter password:
Confirm password:
The user "layla" was created successfully
Display name set to "Layla Smith"
User "layla" added to group "users"
User "layla" added to group "db-admins"
```

`--password-from-env` reads the password from the `OC_PASS` environment variable. This keeps the password out of the process list and allows scripting the creation of multiple users. Note that `sudo` strips environment variables by default; the `-E` flag preserves them, as shown in the example below:

```
export OC_PASS=newpassword
sudo -E -u www-data php occ user:add --password-from-env \
  --display-name="Layla Smith" --group="users" layla
The user "layla" was created successfully
Display name set to "Layla Smith"
User "layla" added to group "users"
```

`--generate-password` sets a securely generated password that is never shown in the output. Combined with `--email`, this creates a user with a temporary password and sends a welcome email:

```
sudo -E -u www-data php occ user:add layla --generate-password --email layla@example.tld
The account "layla" was created successfully
Welcome email sent to layla@example.tld
```

`--email` sets the user's email address and sends a welcome email if `newUser.sendEmail` is set to `yes` in the `core` app config, or if it is not set at all (`yes` is the default):

```
sudo -E -u www-data php occ user:add layla --email layla@example.tld
Enter password:
Confirm password:
The account "layla" was created successfully
Welcome email sent to layla@example.tld
```

user:resetpassword

Reset any user's password, including administrators (see [../configuration_user/reset_admin_password](#)):

```
sudo -E -u www-data php occ user:resetpassword layla
Enter a new password:
Confirm the new password:
Successfully reset password for layla
```

Clear a user's password with `--no-password`:

```
sudo -E -u www-data php occ user:resetpassword --no-password layla
Are you sure you want to clear the password for layla?
Successfully reset password for layla
```

You may also use `--password-from-env` to reset passwords non-interactively:

```
export OC_PASS=newpassword
sudo -E -u www-data php occ user:resetpassword --password-from-env layla
Successfully reset password for layla
```

user:delete

Delete a user:

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

user:disable and user:enable

Disable a user. Their active sessions will be invalidated within 5 minutes. To invalidate sessions immediately, use `user:auth-tokens:delete` before or after disabling the account:

```
sudo -E -u www-data php occ user:disable <username>
```

Re-enable a disabled user:

```
sudo -E -u www-data php occ user:enable <username>
```

user:lastseen

Show a specific user's most recent login:

```
sudo -E -u www-data php occ user:lastseen layla  
layla's last login: 2024-03-20 17:18
```

Show all users' most recent logins:

```
sudo -E -u www-data php occ user:lastseen --all  
albert's last login: 2024-03-18 10:30  
bob has never logged in.  
layla's last login: 2024-03-20 17:18  
stephanie's last login: 2024-01-11 13:26
```

user:list

List all registered users. By default, output is limited to 500 users; use `--limit` and `--offset` to page through larger sets:

```
sudo -E -u www-data php occ user:list  
- admin: admin  
- layla: Layla Smith  
- fred: Fred Jones
```

Use `--disabled` to list only disabled users, and `--info` to include additional backend details.

user:info

Show account details for a user, including display name, email, groups, quota, and storage usage:

```
sudo -E -u www-data php occ user:info layla
- user_id: layla
- display_name: Layla Smith
- email: layla@example.tld
- cloud_id: layla@cloud.example.tld
- enabled: true
- groups:
  - users
  - db-admins
- quota: none
- storage:
  - free: 162409623552
  - used: 1110
  - total: 162409624662
  - relative: 0
  - quota: -3
- first_seen: 2024-03-01T08:44:46+00:00
- last_seen: 2024-03-20T17:18:00+00:00
- user_directory: /var/www/nextcloud/data/layla
- backend: Database
```

user:profile

Read user profile properties:

```
sudo -E -u www-data php occ user:profile layla
- displayname: Layla Smith
- address: Berlin
- email: layla@example.tld
- profile_enabled: 1
- pronouns: they/them
```

Get a single profile property:

```
sudo -E -u www-data php occ user:profile layla address
Berlin
```

Set a profile property:

```
sudo -E -u www-data php occ user:profile layla address Stuttgart
```

Delete a profile property:

```
sudo -E -u www-data php occ user:profile layla address --delete
```

user:setting

Read user settings:

```
sudo -E -u www-data php occ user:setting layla
- core:
  - lang: en
- login:
  - lastLogin: 1465910968
- settings:
  - email: layla@example.tld
```

Filter by app:

```
sudo -E -u www-data php occ user:setting layla core
- core:
  - lang: en
```

Get a single setting:

```
sudo -E -u www-data php occ user:setting layla core lang
en
```

Set a setting:

```
sudo -E -u www-data php occ user:setting layla settings email "new-layla@example.tld"
```

Delete a setting:

```
sudo -E -u www-data php occ user:setting layla settings email --delete
```

user:report

Show a count of all users, including users on external authentication backends such as LDAP:

```
sudo -E -u www-data php occ user:report
```

```
+-----+-----+
| User Report      |   |
+-----+-----+
| Database         | 12 |
| LDAP             | 86 |
|                  |    |
| total users      | 98 |
|                  |    |
| user directories | 2  |
| active users     | 15 |
| disabled users   | 0  |
+-----+-----+
```

Active users are those who have logged in at least once. Users who have never logged in are not counted as active or disabled. Some backends do not support a user count and may show as zero.

user:auth-tokens:list

List all active authentication tokens (sessions and app passwords) for a user:

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

```
+---+-----+-----+-----+-----+
| id | name           | lastActivity           | type    | scope    |
+---+-----+-----+-----+-----+
| 42 | Firefox on Linux | 2024-03-20T17:18:00+00:00 | temporary | filesystem |
| 47 | Backup script    | 2024-03-19T08:00:00+00:00 | permanent | filesystem |
+---+-----+-----+-----+-----+
```

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

user:auth-tokens:add

Create an app password for a user. If no login password is provided, the generated token will have limited capabilities (operations that require the login password will fail):

```
sudo -E -u www-data php occ user:auth-tokens:add --name="Backup script" layla
```

Enter password:

app password: kFrH9-TXk4s-gUo0Q-K0VH8

Use `--password-from-env` to read the login password from `NC_PASS` non-interactively:

```
export NC_PASS=userpassword
sudo -E -u www-data php occ user:auth-tokens:add --name="CI runner" \
  --password-from-env layla
```

user:auth-tokens:delete

Delete a specific token by its ID (from `user:auth-tokens:list`):

```
sudo -E -u www-data php occ user:auth-tokens:delete layla 47
```

Delete all tokens for a user that have not been used since a given date:

```
sudo -E -u www-data php occ user:auth-tokens:delete layla \
  --last-used-before="2024-01-01"
```

user:clear-avatar-cache

Clear the cached avatar images for all users. Useful after changing avatar storage settings or after migrating user data:

```
sudo -E -u www-data php occ user:clear-avatar-cache
```

user:keys:verify

Verify that the stored public key for a user matches their stored private key. Returns a confirmation or mismatch notice. Useful for diagnosing end-to-end encryption issues:

```
sudo -E -u www-data php occ user:keys:verify layla
Stored public key matches stored private key
```

user:sync-account-data

Sync user data from the configured user backends (LDAP, SAML, etc.) to the Nextcloud accounts table. Useful after backend changes to ensure profile data, email addresses, and display names are up to date:

```
sudo -E -u www-data php occ user:sync-account-data  
layla - updated
```

Use `--limit` and `--offset` to process users in batches.

user:welcome

Send the welcome email to a user. The instance must have a working email configuration:

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

Add `--reset-password` to include a password reset link in the email:

```
sudo -E -u www-data php occ user:welcome --reset-password layla
```

Group commands

The `group` commands create and manage groups and their memberships:

group	
group:add	add a group
group:adduser	add a user to a group
group:delete	remove a group
group:info	show information about a group
group:list	list configured groups
group:removeuser	remove a user from a group

group:add

Create a new group:

```
sudo -E -u www-data php occ group:add milliways
```

group:adduser and group:removeuser

Add one or more existing users to the specified group with the `group:adduser` command. The syntax is:

```
group:adduser <gid> <uid1> [uid2 ... uidN]
```

This example adds the users "denis", "dora" and "daisy" to the existing group "milliways":

```
sudo -E -u www-data php occ group:adduser milliways denis dora daisy
```

You can remove one or more users from the group with the `group:removeuser` command. This example removes the existing users "denis", "dora" and "daisy" from the existing group "milliways":

```
sudo -E -u www-data php occ group:removeuser milliways denis dora daisy
```

group:delete

Remove a group. This does not delete the users in the group. The `admin` group cannot be removed:

```
sudo -E -u www-data php occ group:delete milliways
```

group:list

List configured groups. Optionally filter by a search string:

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

- admin:
  - admin
- milliways:
  - layla
- users:
  - layla

sudo -E -u www-data php occ group:list milli

- milliways:
  - layla
```

Use `--limit` and `--offset` to page through large numbers of groups. Use `--info` to include the backend for each group. Use `--output=json` or `--output=json_pretty` for machine-readable output.

group:info

Show details about a group, including its members and backend:

```
sudo -E -u www-data php occ group:info admin
- groupID: admin
- displayName: admin
- backends:
  - Database
```

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

Two-factor authentication

The `twofactorauth` commands manage two-factor authentication (2FA) enforcement and provider state:

<code>twofactorauth</code>	
<code>twofactorauth:cleanup</code>	clean up provider associations for a removed provider
<code>twofactorauth:disable</code>	disable 2FA for a user (provider-specific)
<code>twofactorauth:enable</code>	enable 2FA for a user (provider-specific)
<code>twofactorauth:enforce</code>	enforce or disable mandatory 2FA globally or per group
<code>twofactorauth:state</code>	show the 2FA state for a user

twofactorauth:disable and twofactorauth:enable

If a user loses access to their second factor (for example, a lost phone), an admin can disable 2FA for that user for a specific provider:

```
sudo -E -u www-data php occ twofactorauth:disable <uid> <provider_id>
```

Re-enable 2FA for the user:

```
sudo -E -u www-data php occ twofactorauth:enable <uid> <provider_id>
```

“ ”

Not all 2FA providers support per-user enable/disable via occ.

twofactorauth:enforce

Enforce 2FA for all users:

```
sudo -E -u www-data php occ twofactorauth:enforce --on  
Two-factor authentication is enforced for all users
```

Enforce 2FA only for specific groups, optionally excluding others:

```
sudo -E -u www-data php occ twofactorauth:enforce --on \  
--group=admin --group=finance --exclude=service-accounts  
Two-factor authentication is enforced for members of the group(s) admin, finance
```

Disable enforcement:

```
sudo -E -u www-data php occ twofactorauth:enforce --off  
Two-factor authentication is not enforced
```

twofactorauth:state

Show whether 2FA is enabled, enforced, and which providers are active for a user:

```
sudo -E -u www-data php occ twofactorauth:state layla  
Two-factor authentication is not enabled for user layla  
  
Disabled providers:  
- backup_codes  
- totp
```

twofactorauth:cleanup

Remove stored 2FA provider associations for a provider that has been uninstalled. This cleans up stale data after removing a 2FA app:

```
sudo -E -u www-data php occ twofactorauth:cleanup <provider_id>
```

System Tags

System tags are admin-managed labels that can be assigned to files for use in workflows and automated actions.

Tags have three access levels:

Level	Visible ¹	Assignable ²
public	Yes	Yes
restricted	Yes	No
invisible	No	No

¹ User can see the tag

² User can assign the tag to a file

See [../file_workflows/automated_tagging](#) for typical use cases for restricted and invisible tags, such as retention and access control.

tag:list

List all system tags:

```
sudo -E -u www-data php occ tag:list
- 1:
  - name: confidential
  - access: restricted
- 2:
  - name: needs-review
  - access: public
```

tag:add

Create a new system tag:

```
sudo -E -u www-data php occ tag:add confidential restricted
- id: 1
- name: confidential
- access: restricted
```

tag:edit

Rename a tag or change its access level. Use the tag ID from `tag:list`:

```
sudo -E -u www-data php occ tag:edit --name "reviewed" --color="" 2
Tag updated ("reviewed", true, true, "")
```

`--name` and `--access` are optional. `--color=""` must be passed explicitly due to a known issue with the command.

tag:delete

Delete a tag by its ID:

```
sudo -E -u www-data php occ tag:delete 1
The specified tag was deleted
```

Assigning tags to files

Add one or more tags to a file or directory (specified by file ID or path). The `access` argument identifies which tag to apply — because two tags can share the same name but have different access levels, both name and access level are required to uniquely identify a tag. If no matching tag exists, it is created automatically:

```
sudo -E -u www-data php occ tag:files:add /layla/files/report.pdf confidential restricted
restricted tag named confidential added.
```

Multiple tags can be specified as a comma-separated list. All tags in a single call must share the same access level.

Remove specific tags from a file:

```
sudo -E -u www-data php occ tag:files:delete /layla/files/report.pdf confidential restricted
```

Remove all tags from a file:

```
sudo -E -u www-data php occ tag:files:delete-all /layla/files/report.pdf
```

🔒 #2

🔒 cocobook-sync 🔒 2026-10-28 16:27:29 UTC

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