

Hardening and security guidance



- [Passwords](#)
 - [Storage of account passwords](#)
 - [Storage of authentication tokens](#)
 - [Token-associated storage of login passwords](#)
 - [Security consequences](#)
 - [Password Length Limits](#)
- [Operating system](#)
 - [Give PHP read access to /dev/urandom](#)
 - [Enable hardening modules such as SELinux](#)
- [Deployment](#)
 - [Place data directory outside of the web root](#)
 - [Place config directory outside of the web root](#)
 - [Disable preview image generation](#)
 - [Disable Debug Mode](#)
- [Use HTTPS](#)
 - [Redirect all unencrypted traffic to HTTPS](#)
 - [Enable HTTP Strict Transport Security](#)
 - [Proper SSL configuration](#)
- [Restrict admin actions to a specific range of IP addresses](#)
- [Use a dedicated domain for Nextcloud](#)
- [Ensure that your Nextcloud instance is installed in a DMZ](#)
- [Serve security related headers by the Web server](#)
- [Connections to remote servers](#)
- [Setup fail2ban](#)
 - [Nextcloud prerequisites](#)
 - [Fail2ban introduction](#)

- [Setup a filter and a jail for Nextcloud](#)
-

Nextcloud aims to ship with secure defaults that do not need to get modified by administrators. However, in some cases some additional security hardening can be applied in scenarios where the administrator has complete control over the Nextcloud instance. This page assumes that you run Nextcloud Server on Apache2 in a Linux environment.



Nextcloud will warn you in the administration interface if some critical security-relevant options are missing. However, it is still up to the server administrator to review and maintain system security.

Passwords

Storage of account passwords

Nextcloud's built-in database user backend stores a salted, one-way hash of each account password. It prefers Argon2id (when supported by the PHP installation), with Argon2i and bcrypt used as fallbacks. The algorithm, salt, and cost parameters are included in the stored hash. Existing hashes are automatically upgraded following successful password verification when they no longer match the preferred algorithm or parameters.

The hash is used to verify password-based login attempts and is not designed to be decrypted. When an external user backend (such as LDAP) is used, storage and verification of the account password are controlled by that backend.

This account-password hash is separate from any recoverable copy of the login password that Nextcloud stores in connection with authentication tokens, as described below.

Storage of authentication tokens

After successful authentication, Nextcloud issues an authentication token that the client presents with subsequent authenticated requests. A valid token can authenticate as the associated user, subject to the token's scope, expiration, type, and server-side validity checks. Depending on the token type and client, the token may be transmitted in a session cookie, used as an app password, or sent as a bearer token.

Nextcloud does not store the plaintext authentication token in the database. Instead, it stores a SHA-512 hash derived from the token and the instance-specific `secret`. The corresponding server-side token record contains the associated user identity, authentication metadata, and cryptographic key material. Authentication tokens must therefore be protected like passwords. They should not be logged, placed in URLs, or intentionally persisted outside the client that uses them.

Token-associated storage of login passwords

By default, `auth.storeCryptedPassword` is enabled. When this setting is enabled and the login password is available during token creation, Nextcloud stores a reversibly encrypted copy of it in the server-side authentication-token record.

For an account using the built-in database user backend, this encrypted copy is separate from the one-way account-password hash. The database contains the one-way account-password hash and also contains an encrypted password copy for each authentication-token record created with password storage enabled and a login password supplied.

The recoverable copy is used by features that need the original login credentials, such as connecting to external storage, autoconfiguring accounts in the Mail app, and periodically checking whether the login credentials remain valid. When token creation receives no login password, the resulting token record contains no recoverable password. Such a token record remains without a stored password until Nextcloud receives the password during a later login or password update.

The authentication token itself does not contain the login password. Each token record has a separate RSA key pair. Nextcloud encrypts the login password with the record's public key and encrypts the corresponding private key using the authentication token together with the instance-specific `secret` from `config.php`. Possession of the authentication token, instance secret, and corresponding database record is therefore sufficient to decrypt a password stored in that record.

Administrators can disable this behavior with `auth.storeCryptedPassword`. Disabling it does not affect the one-way account-password hash used by the built-in database user backend. Features that rely on recovering the login password from an authenticate-token record cannot retrieve it from records created without a stored password.

When an authentication token contains a stored password, Nextcloud periodically checks that password against the user backend. If the password is no longer valid, Nextcloud marks the token as having an invalid password and rejects authentication with that token. When the token contains no stored password, Nextcloud skips this password check. Consequently, changing a password directly in an external user backend does not cause a token without a stored password to be rejected through the periodic credential check. The password change alone does not invalidate the token; the token remains valid until it expires, is otherwise invalidated, or the user is disabled.

Security consequences

Leakage of authentication data has the following security consequences:

- An actor with a valid authentication token can authenticate as the associated user, subject to the token's scope, expiration, type, and server-side validity checks.
- An actor with the authentication token, the instance-specific `secret` from `config.php`, and the corresponding database record can decrypt the login password stored in that record.
- An account-password hash does not reveal the original password directly, but an actor who obtains it can perform offline password-guessing attacks.

Password Length Limits

Nextcloud accepts account passwords of up to 469 bytes through its standard account-creation, password-change, and password-reset interfaces. This is the maximum account-password length enforced by these interfaces. Because the limit is measured in bytes, a password containing multibyte characters (such as emojis or characters from non-Latin scripts) can reach the limit with fewer than 469 characters.

Administrators can use the [Password Policy app](#) to configure requirements such as a minimum password length and other complexity rules. External user backends can impose additional or different requirements.

The following implementation details do not change the 469-byte account-password maximum, but are relevant when selecting a password policy:

Token Encryption Performance

When `auth.storeCryptedPassword` is enabled and an account password is longer than 214 bytes, Nextcloud uses a larger RSA key when creating authentication-token records. This increases token-generation overhead, but does not prevent passwords between 215 and 469 bytes from being accepted. The 214-byte threshold is therefore a performance consideration, not a password-length limit.

Administrators who expect very long or one-time passwords to be used may consider disabling `auth.storeCryptedPassword` to avoid this overhead, subject to the functional consequences described above.

Algorithmic Truncation (bcrypt fallback)

Nextcloud prefers Argon2id for one-way password hashing (when supported by the PHP installation), with Argon2i and bcrypt as fallbacks. Bcrypt considers only the first 72 bytes of its input. Therefore, if bcrypt is selected, input after the first 72 bytes does not contribute to password verification. This is a bcrypt-specific behavior, not a general 72-byte limit imposed by Nextcloud.

Passwords protecting public link and mail shares use the same one-way password hasher and are subject to the applicable share-password policy. They are not stored in authentication-token records, so the encryption-related performance considerations above do not apply.

Operating system

Give PHP read access to `/dev/urandom`

Nextcloud uses a [RFC 4086 \("Randomness Requirements for Security"\)](#) compliant mixer to generate cryptographically secure pseudo-random numbers. This means that when generating a random number Nextcloud will request multiple random numbers from different sources and derive from these the final random number.

The random number generation also tries to request random numbers from `/dev/urandom`, thus it is highly recommended to configure your setup in such a way that PHP is able to read random data from it.

“ ”

When having an `open_basedir` configured within your `php.ini` file, make sure to include `/dev/urandom`.

Enable hardening modules such as SELinux

It is highly recommended to enable hardening modules such as SELinux where possible. See [SELinux configuration](#) to learn more about SELinux.

Deployment

Place data directory outside of the web root

It is highly recommended to place your data directory outside of the Web root (i.e. outside of `/var/www`). It is easiest to do this on a new installation.

Place config directory outside of the web root

You can move the `config/` directory outside the web root using the `NEXTCLOUD_CONFIG_DIR` environment variable. This ensures `config.php` — which contains database credentials, secret keys, and other sensitive values — is not accessible via HTTP even in the event of a web server misconfiguration.

Set the variable in your web server virtual host configuration:

```
# Apache
SetEnv NEXTCLOUD_CONFIG_DIR /etc/nextcloud
```

```
# nginx – set via fastcgi_param or the PHP-FPM pool's env[] setting
fastcgi_param NEXTCLOUD_CONFIG_DIR /etc/nextcloud;
```

Also set it for CLI work (`occ`, cron):

```
export NEXTCLOUD_CONFIG_DIR=/etc/nextcloud
```

“ ”

The variable must be set for **both** the web server process and CLI invocations. Verify with `occ config:list system` after changing it.

“ ”

[Configuration Parameters](#) for full details on `NEXTCLOUD_CONFIG_DIR` and other configuration loading behaviour.

Disable preview image generation

Nextcloud is able to generate preview images of common filetypes such as images or text files. By default the preview generation for some file types that we consider secure enough for deployment is enabled. However, administrators should be aware that these previews are generated using PHP libraries written in C which might be vulnerable to attack vectors.

For high security deployments we recommend disabling the preview generation by setting the `enable_previews` switch to `false` in `config.php`. As an administrator you are also able to manage which preview providers are enabled by modifying the `enabledPreviewProviders` option switch.

Disable Debug Mode

Verify that `debug` is `false` in your `config.php`. The default is `false` in new installations (or when not specified). It should not be enabled in production environments or outside of targeted troubleshooting situations. When enabled, things like server-wide WebDAV collection listings are permitted. It is intended for local development and usage in controlled environments only.

Use HTTPS

Using Nextcloud without using an encrypted HTTPS connection opens up your server to a man-in-the-middle (MITM) attack, and risks the interception of user data and passwords. It is a best practice, and highly recommended, to always use HTTPS on production servers, and to never allow unencrypted HTTP.

How to setup HTTPS on your Web server depends on your setup; please consult the documentation for your HTTP server. The following examples are for Apache.

Redirect all unencrypted traffic to HTTPS

To redirect all HTTP traffic to HTTPS administrators are encouraged to issue a permanent redirect using the 301 status code. When using Apache this can be achieved by a setting such as the following in the Apache VirtualHosts configuration:

```
<VirtualHost *:80>
    ServerName cloud.nextcloud.com
    Redirect permanent / https://cloud.nextcloud.com/
</VirtualHost>
```

Enable HTTP Strict Transport Security

While redirecting all traffic to HTTPS is good, it may not completely prevent man-in-the-middle attacks. Thus administrators are encouraged to set the HTTP Strict Transport Security header, which instructs browsers to not allow any connection to the Nextcloud instance using HTTP, and it attempts to prevent site visitors from bypassing invalid certificate warnings.

This can be achieved by setting the following settings within the Apache VirtualHost file:

```
<VirtualHost *:443>
    ServerName cloud.nextcloud.com
    <IfModule mod_headers.c>
        Header always set Strict-Transport-Security "max-age=15552000; includeSubDomains"
    </IfModule>
</VirtualHost>
```

“ ”

We recommend the additional setting `; preload` to be added to that header. Then the domain will be added to a hardcoded list that is shipped with all major browsers and enforce HTTPS upon those domains. See the [HSTS preload website for more information](#). Due to the policy of this list you need to add it to the above example for yourself once you are sure that this is what you want. [Removing the domain from this list](#) could take some months until it reaches all installed browsers.

This example configuration will make all subdomains only accessible via HTTPS. If you have subdomains not accessible via HTTPS, remove `includeSubDomains`.

This requires the `mod_headers` extension in Apache.

Proper SSL configuration

Default SSL configurations by Web servers are often not state-of-the-art, and require fine-tuning for an optimal performance and security experience. The available SSL ciphers and options depend completely on your environment and thus giving a generic recommendation is not really possible.

We recommend using the [Mozilla SSL Configuration Generator](#) to generate a suitable configuration suited for your environment. To verify your configuration you can use the free [Web TLS Profiler](#) service. This service gives detailed error messages, if your server's TLS settings deviate from the Mozilla Configuration. Another useful tool to check your server's TLS configuration is the free [Qualys SSL Labs Test](#) which provides general information about the TLS settings.

Also ensure that HTTP compression is disabled to mitigate the BREACH attack.

Restrict admin actions to a specific range of IP addresses

Configure `allowed_admin_ranges` in `config.php` to restrict the admin actions to trusted IP ranges.

This can be achieved with this kind of setting, usually using private IP ranges:

```
'allowed_admin_ranges' => [  
    '127.0.0.1/8',  
    '192.168.0.0/16',  
    'fd00::/8',  
],
```

All requests originating from IP addresses outside of these ranges will not be able to execute admin actions.

Administrators connected from untrusted IP addresses will be able to use Nextcloud, but all admin specific actions will be hidden.

Use a dedicated domain for Nextcloud

Administrators are encouraged to install Nextcloud on a dedicated domain such as `cloud.domain.tld` instead of `domain.tld` to gain all the benefits offered by the Same-Origin-Policy.

Ensure that your Nextcloud instance is installed in a DMZ

As Nextcloud supports features such as Federated File Sharing we do not consider Server Side Request Forgery (SSRF) part of our threat model. In fact, given all our external storage adapters this can be considered a feature and not a vulnerability.

This means that a user on your Nextcloud instance could probe whether other hosts are accessible from the Nextcloud network. If you do not want this you need to ensure that your Nextcloud is properly installed in a segregated network and proper firewall rules are in place.

Serve security related headers by the Web server

Basic security headers are served by Nextcloud already in a default environment. These include:

- `X-Content-Type-Options: nosniff`
 - Instructs some browsers to not sniff the mimetype of files. This is used for example to prevent browsers from interpreting text files as JavaScript.
- `X-Robots-Tag: noindex, nofollow`
 - Instructs search machines to not index these pages and not follow any links there.
- `X-Frame-Options: SAMEORIGIN`
 - Prevents embedding of the Nextcloud instance within an iframe from other domains to prevent Clickjacking and other similar attacks.
- `Referrer-Policy: no-referrer`
 - The default no-referrer policy instructs the browser not to send referrer information along with requests to any origin.

These headers are hard-coded into the Nextcloud server, and need no intervention by the server administrator.

For optimal security, administrators are encouraged to serve these basic HTTP headers by the Web server to enforce them on response. To do this Apache has to be configured to use the `.htaccess` file and the following Apache modules need to be enabled:

- `mod_headers`
- `mod_env`

Administrators can verify whether this security change is active by accessing a static resource served by the Web server and verify that the above mentioned security headers are shipped.

Connections to remote servers

Some functionalities require the Nextcloud server to be able to connect remote systems via https/443. This paragraph also includes the data which is being transmitted to the Nextcloud GmbH. Depending on your server setup, these are the possible connections:

- connectivity.nextcloud.com, www.eff.org, edri.org
 - [optional \(config\)](#)
 - for checking the internet connection
- cloud.nextcloud.com
 - used for enterprise license monitoring
 - submitted data: subscription key, user count
- updates.nextcloud.com
 - to check for available Nextcloud server updates
 - submitted data: server version, subscription key, install time, instance id, instance size
- apps.nextcloud.com, ltd[1-3].nextcloud.com, garm[1-5].nextcloud.com
 - to check for available apps and their updates
 - source is apps.nextcloud.com the ltd and garm servers are just mirroring the apps.json file
 - submitted data: subscription key
- github.com, objects.githubusercontent.com, release-assets.githubusercontent.com
 - to download Nextcloud standard apps
 - to download Nextcloud server releases
- push-notifications.nextcloud.com
 - sending push notifications to mobile clients
 - submitted data: unique device identifier, public key, push token
- pushfeed.nextcloud.com
 - optional
 - checking for updates to be shown in the Nextcloud Announcements app
- lookup.nextcloud.com
 - optional
 - for updating and lookups to the federated sharing addressbook
 - submitted data: *pending*
- surveyserver.nextcloud.com
 - optional
 - if the admin has agreed to share anonymized server data
 - submitted data: statistical data. see here for the [detailed field list](#)
- nominatim.openstreetmap.org
 - optional
 - if the weather status app is enabled and used
 - submitted data: address manually entered by the user to resolve to longitude and latitude
- api.opentopodata.org
 - optional
 - if the weather status app is enabled and used
 - submitted data: address manually entered by the user to resolve the altitude of the location
- api.met.no
 - optional

- if the weather status app is enabled and used
- submitted data: longitude and latitude configured in the weather status app by the individual user
- Any remote Nextcloud server that is connected with federated sharing
- When downloading apps from the App store other domains might be accessed, based on the choice of the app developers where they host the releases. For all official Nextcloud apps this is not the case though, because they are hosted on Github.

Setup fail2ban

Exposing your server to the internet will inevitably lead to the exposure of the services running on the internet-exposed ports to brute force login attempts.

This guide will enable blocking of the originating IP addresses at an operating system level, so the webserver, PHP and the database do not need to handle this unnecessary traffic at all.

Nextcloud prerequisites

Nextcloud logs failed login attempts in `nextcloud.log` with log level `2`, so you need to define a `loglevel` of `2` or less in `config.php`.

Make sure your `nextcloud.log` is writeable by your webserver user, possibly by defining a correct `logfilemode` in `config.php`.

Perform a bad login attempt and check whether it does get logged to `nextcloud.log`.

Note that `audit.log` (if enabled) currently only logs successful logins and cannot be used.

Fail2ban introduction

Fail2ban is a service that uses iptables to automatically drop connections for a pre-defined amount of time from IPs that continuously failed to authenticate to the configured services.

In order to setup fail2ban, you first need to download and install it on your server. Downloads for several distributions can be found on [fail2ban download page](#). It is often available from most distributions' package managers (e.g. `apt-get`).

The standard path for fail2ban's configuration is `/etc/fail2ban`.

Setup a filter and a jail for Nextcloud

A filter defines regex rules to identify when users fail to authenticate on Nextcloud's user interface, WebDAV, or use an untrusted domain to access the server.

Create a file in `/etc/fail2ban/filter.d` named `nextcloud.conf` with the following contents:

```
[Definition]
_groupsre = (?:(?:,?\s*"w+":(?:"[^"]+"|\w+))*)
failregex = ^\{%( _groupsre)s,?\s*"remoteAddr": "<HOST>"%( _groupsre)s,?\s*"message": "Login
failed:
            ^\{%( _groupsre)s,?\s*"remoteAddr": "<HOST>"%( _groupsre)s,?\s*"message": "Two-factor
challenge failed:
            ^\{%( _groupsre)s,?\s*"remoteAddr": "<HOST>"%( _groupsre)s,?\s*"message": "Trusted
domain error.
datepattern = ,?\s*"time"\s*:\s*"%%Y-%%m-%%d[T ]%%H:%%M:%%S(%%z)?"
```

The jail file defines how to handle the failed authentication attempts found by the Nextcloud filter.

Create a file in `/etc/fail2ban/jail.d` named `nextcloud.local` with the following contents:

```
[nextcloud]
backend = auto
enabled = true
port = 80,443
protocol = tcp
filter = nextcloud
maxretry = 3
bantime = 86400
findtime = 43200
logpath = /path/to/data/directory/nextcloud.log
```

Ensure to replace `logpath` with your installation's `nextcloud.log` location. If you are using ports other than `80` and `443` for your Web server you should replace those too. The `bantime` and `findtime` are defined in seconds.

Restart the fail2ban service. You can check the status of your Nextcloud jail by running:

```
fail2ban-client status nextcloud
```

If you need to unban certain IP addresses (`1.2.3.4` in this example), you may do so by issuing:

```
fail2ban-client unban 1.2.3.4
```

There may be scenarios where you want to more permanently ban certain IP addresses that repeatedly generate bad login attempts (or other attacks) by using fail2ban's `recidive` feature.

```
### #3
[] cocobook-sync [] 2026-10-28 16:26:22 UTC
[] cocobook-sync [] 2026-10-29 01:09:14 UTC
```