

Nextcloud

- [Configure Redis UNIX socket](#)
- [Upgrade php8.3 to php8.4](#)

Configure Redis UNIX socket

1. 1

Set Redis to listen on a Unix socket Edit **/etc/redis/redis.conf** and add or uncomment:
unixsocket /var/run/redis/redis-server.sock unixsocketperm 770 You can leave the TCP
`port 6379` line as-is or set `port 0` to disable TCP entirely once you confirm the socket
works.

2. 2

Fix the socket directory and ownership The socket file is created by the redis user (usually
`redis`). Your web server user (usually `www-data` for PHP-FPM/nginx or apache) needs
read/write access to it. Add www-data to the redis group: `sudo usermod -aG redis www-`
`data` Then make sure the socket's group matches: `unixsocketperm 770` combined with
group ownership `redis:redis` means www-data (now in the redis group) can connect.

3. 3

Restart Redis and verify the socket exists `sudo systemctl restart redis-server` `ls -l`
`/var/run/redis/redis-server.sock` You should see something like `srwxrwx--- 1 redis redis ...`
`redis-server.sock`. If the file's missing, check `sudo systemctl status redis-server` and
`journalctl -u redis-server` for errors — often it's a directory permissions or AppArmor
issue.

4. 4

Test the socket connection `redis-cli -s /var/run/redis/redis-server.sock ping` This should
return PONG. If it hangs or errors, the socket permissions or path are wrong — recheck
step 2.

5. 5

Update config.php to use only the socket block Remove or comment out the TCP `redis`
=> array('host' => 'localhost', 'port' => 6379, ...)` block, keeping only: `'redis' => array (`
`'host' => '/var/run/redis/redis-server.sock', 'port' => 0, 'timeout' => 1.5, 'dbindex' => 0,`
`)`, Use dbindex 0 unless you're intentionally isolating Nextcloud from other apps sharing
this Redis instance — keep it consistent, don't leave a stray db index from testing.

6. 6

Clear cache and restart PHP-FPM `sudo systemctl restart php8.x-fpm` (match your actual
PHP version) Then as your Nextcloud admin user, run: `sudo -u www-data php occ status`
`sudo -u www-data php occ maintenance:repair` This confirms Nextcloud can reach Redis
and forces it to re-read config.php.

7. 7

Confirm Nextcloud is actually using it Check Settings > Administration > Overview for
warnings, or run: `sudo -u www-data php occ check` You can also watch live traffic on the
socket with: `redis-cli -s /var/run/redis/redis-server.sock monitor` while clicking around in
Nextcloud — you should see GET/SET/lock commands streaming in.

Upgrade php8.3 to php8.4

```
grep -E  
'memory_limit|upload_max_filesize|post_max_size|max_execution_time|opcache|date.timezone|output_buffering' /etc/php/8.3/fpm/php.ini /etc/php/8.3/fpm/conf.d/*.ini
```

for php.ini

```
clear_env = no
```

```
request_terminate_timeout = 300
```

```
; Match max_execution_time to handle long uploads gracefully.
```

```
process.priority = -19
```

```
; Optional: Boost priority (requires root and SYS_NICE capability).
```

```
pm = dynamic
```

```
; "dynamic" adjusts the number of child processes based on load; use "static" for dedicated servers with predictable traffic.
```

```
pm.max_children = 120
```

```
; Max number of child processes; calculate as (Total RAM - 1G for OS/other services) / 50-100 MB per process.
```

```
; Example: 4 GB RAM → ~60-80 children; 8 GB RAM → ~120-150 children.
```

```
pm.start_servers = 12
```

```
; Initial processes; set to ~10-20% of max_children.
```

```
pm.min_spare_servers = 6
```

```
; Minimum idle processes; set to ~5-10% of max_children.
```

```
pm.max_spare_servers = 18
```

```
; Maximum idle processes; set to ~15-25% of max_children.
```

```
pm.max_requests = 500
```

```
; Restart child processes after this many requests to prevent memory leaks.
```

for pool.d/www.conf

Check **conf.d/*** Copy **10-opcache**