Install NextCloud on Ubuntu 20.04 with Nginx (LEMP Stack)

This tutorial will be showing you how to install NextCloud on Ubuntu 20.04 LTS with Nginx web server.

What’s NextCloud?

NextCloud is a free open-source self-hosted cloud storage solution. It’s functionally similar to Dropbox. Proprietary cloud storage solutions (Dropbox, Google Drive, etc) are convenient, but at a price: they can be used to collect personal data because your files are stored on their computers. If you worried about privacy, you can switch to NextCloud, which you can install on your private home server or on a virtual private server (VPS). You can upload your files to your server via NextCloud and then sync those files to your desktop computer, laptop or smartphone. This way you have full control of your data.

NextCloud Features

- Free and open-source
- End-to-end encryption, meaning files can be encrypted on client devices before uploaded to the server, so even if someone steals your server, they can not read your files.
- Can be integrated with an online office suite (Collobora Online, OnlyOffice) so you can create and edit your doc, ppt, xls files directly from NextCloud.
- The app store contains hundreds of apps to extend functionality (like calendar app, contacts app, note-taking app, video conferencing app, etc).
- The sync client is available on Linux, macOS, Windows, iOS and android.

Prerequisites

NextCloud is written in PHP programing language. **To follow this tutorial, you first need to install LEMP stack on Ubuntu 20.04**. If you haven't already done so, please check out the following tutorial.

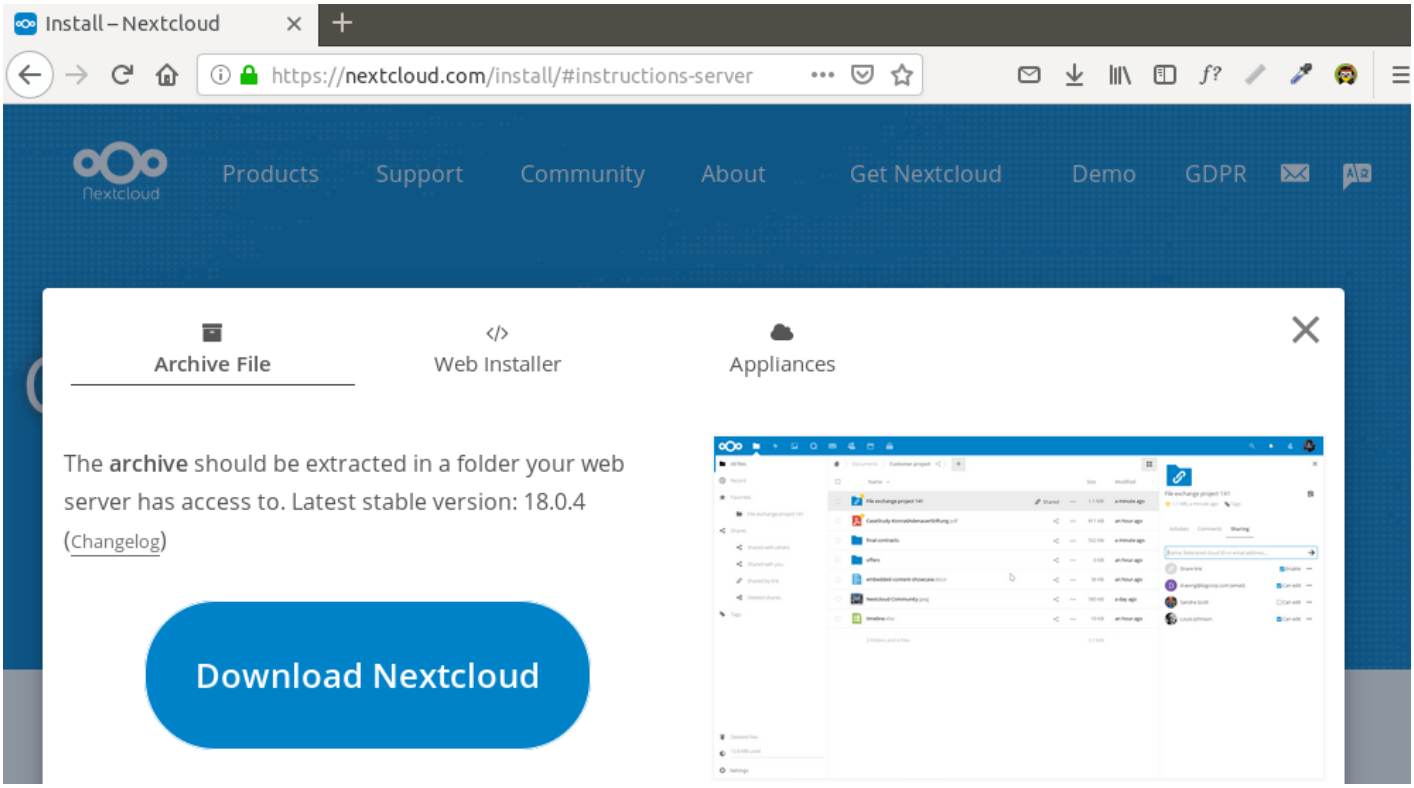
- How to Install LEMP Stack (Nginx, MariaDB, PHP7.4-FPM) on Ubuntu 20.04

You can install NextCloud on your home server or a VPS (virtual private server). You also need a domain name, so later on your will be able to enable HTTPS to encrypt the HTTP traffic. I registered my domain name from NameCheap because the price is low and they give whois privacy protection free for life. Nextcloud can be installed without a domain name, but it really doesn't make sense if you don't encrypt the HTTP connection to prevent snooping. I recommend buying a domain name, if you really want to tinker with server software and use them to the fullest potential.

Now let's install NextCloud.

Step 1: Download NextCloud on Ubuntu 20.04

Log into your Ubuntu 20.04 server. Then download the NextCloud zip archive onto your server. The latest stable version is 21.0.1 at time of this writing. You may need to change the version number. Go to https://nextcloud.com/install and click the `download for server` button to see the latest version.



You can run the following command to download it on your server.

```
wget https://download.nextcloud.com/server/releases/nextcloud-21.0.1.zip
```

You can always use the above URL format to download NextCloud. If a new version comes out, simply replace `21.0.1` with the new version number.

Once downloaded, extract the archive with `unzip`.

```
sudo apt install unzip


sudo unzip nextcloud-21.0.1.zip -d /usr/share/nginx/
```

The `-d` option specifies the target directory. NextCloud web files will be extracted to `/usr/share/nginx/nextcloud/`. Then we need to change the owner of this directory to `www-data` so that the web server (Nginx) can write to this directory.

```
sudo chown www-data:www-data /usr/share/nginx/nextcloud/ -R
```

Step 2: Create a Database and User for Nextcloud in MariaDB

Database Server

Log into MariaDB database server with the following command. Since MariaDB is now using `unix_socket` plugin to authentication user login, there's no need to enter MariaDB root password. We just need to prefix the `mysql` command with `sudo`.

```
sudo mysql
```

Then create a database for Nextcloud. This tutorial name the database nextcloud. You can use whatever name you like.

```
create database nextcloud;
```

Create the database user. Again, you can use your preferred name for this user. Replace `your-password` with your preferred password.

```
create user nextclouduser@localhost identified by 'your-password';
```

Grant this user all privileges on the `nextcloud` database.

```
grant all privileges on nextcloud.* to nextclouduser@localhost identified by 'your-password';
```

Flush privileges and exit.

```
flush privileges;


exit;
```

```
MariaDB [(none)]> create database nextcloud;
Query OK, 1 row affected (0.002 sec)

MariaDB [(none)]> create user nextclouduser@localhost identified by 'your-password';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> grant all privileges on nextcloud.* to nextclouduser@localhost identified by 'your-password';
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.002 sec)

MariaDB [(none)]> exit;
Bye
```

Step 3: Create a Nginx Config File

for Nextcloud

Create a `nextcloud.conf` file in `/etc/nginx/conf.d/` directory, with a command-line text editor like Nano.

```
sudo nano /etc/nginx/conf.d/nextcloud.conf
```

Copy and paste the following text into the file. Replace `nextcloud.example.com` with your own preferred sub-domain. Don't forget to create DNS A record for this sub-domain in your DNS zone editor. If you don't have a real domain name, I recommend going to NameCheap to buy one. The price is low and they give whois privacy protection free for life.

```
server {
    listen 80;
    listen [::]:80;
    server_name nextcloud.example.com;

    # Add headers to serve security related headers
    add_header X-Content-Type-Options nosniff;
    add_header X-XSS-Protection "1; mode=block";
    add_header X-Robots-Tag none;
    add_header X-Download-Options noopen;
    add_header X-Permitted-Cross-Domain-Policies none;
    add_header Referrer-Policy no-referrer;

    #I found this header is needed on Ubuntu, but not on Arch Linux.
    add_header X-Frame-Options "SAMEORIGIN";

    # Path to the root of your installation
    root /usr/share/nginx/nextcloud/;

    access_log /var/log/nginx/nextcloud.access;
    error_log /var/log/nginx/nextcloud.error;

    location = /robots.txt {
        allow all;
        log_not_found off;
        access_log off;
    }
```

```
# The following 2 rules are only needed for the user_webfinger app.
# Uncomment it if you're planning to use this app.
#rewrite ^/.well-known/host-meta /public.php?service=host-meta last;
#rewrite ^/.well-known/host-meta.json /public.php?service=host-meta-json
# last;

location = /.well-known/carddav {
    return 301 $scheme://$host/remote.php/dav;
}
location = /.well-known/caldav {
   return 301 $scheme://$host/remote.php/dav;
}

location ~ /.well-known/acme-challenge {
  allow all;
}

# set max upload size
client_max_body_size 512M;
fastcgi_buffers 64 4K;

# Disable gzip to avoid the removal of the ETag header
gzip off;

# Uncomment if your server is build with the ngx_pagespeed module
# This module is currently not supported.
#pagespeed off;

error_page 403 /core/templates/403.php;
error_page 404 /core/templates/404.php;

location / {
   rewrite ^ /index.php;
}

location ~ ^/(?:build|tests|config|lib|3rdparty|templates|data)/ {
   deny all;
}
location ~ ^/(?:\.|autotest|occ|issue|indie|db_|console) {
```

```
        deny all;
     }

    location ~ ^/(?:index|remote|public|cron|core/ajax/update|status|ocs/v[12]|updater/.+|ocs-provider/.+|core/templates/40[34])\.php(?:$|/) {
        include fastcgi_params;
        fastcgi_split_path_info ^(.+\.php)(/.*)$;
        try_files $fastcgi_script_name =404;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        fastcgi_param PATH_INFO $fastcgi_path_info;
        #Avoid sending the security headers twice
        fastcgi_param modHeadersAvailable true;
        fastcgi_param front_controller_active true;
        fastcgi_pass unix:/run/php/php7.4-fpm.sock;
        fastcgi_intercept_errors on;
        fastcgi_request_buffering off;
    }

    location ~ ^/(?:updater|ocs-provider)(?:$|/) {
        try_files $uri/ =404;
        index index.php;
    }

    # Adding the cache control header for js and css files
    # Make sure it is BELOW the PHP block
    location ~* \.(?:css|js)$ {
        try_files $uri /index.php$uri$is_args$args;
        add_header Cache-Control "public, max-age=7200";
        # Add headers to serve security related headers (It is intended to
        # have those duplicated to the ones above)
        add_header X-Content-Type-Options nosniff;
        add_header X-XSS-Protection "1; mode=block";
        add_header X-Robots-Tag none;
        add_header X-Download-Options noopen;
        add_header X-Permitted-Cross-Domain-Policies none;
        add_header Referrer-Policy no-referrer;
        # Optional: Don't log access to assets
        access_log off;
   }
```

```
    location ~* \.(?:svg|gif|png|html|ttf|woff|ico|jpg|jpeg)$ {
         try_files $uri /index.php$uri$is_args$args;
         # Optional: Don't log access to other assets
         access_log off;
    }
}
```

Save and close the file. (To save a file in Nano text editor, press `Ctrl+O`, then press `Enter` to confirm. To exit, press `Ctrl+X`.)

Then test Nginx configuration.

```
sudo nginx -t
```

If the test is successful, reload Nginx for the changes to take effect.

```
sudo systemctl reload nginx
```

Step 4: Install and Enable PHP Modules

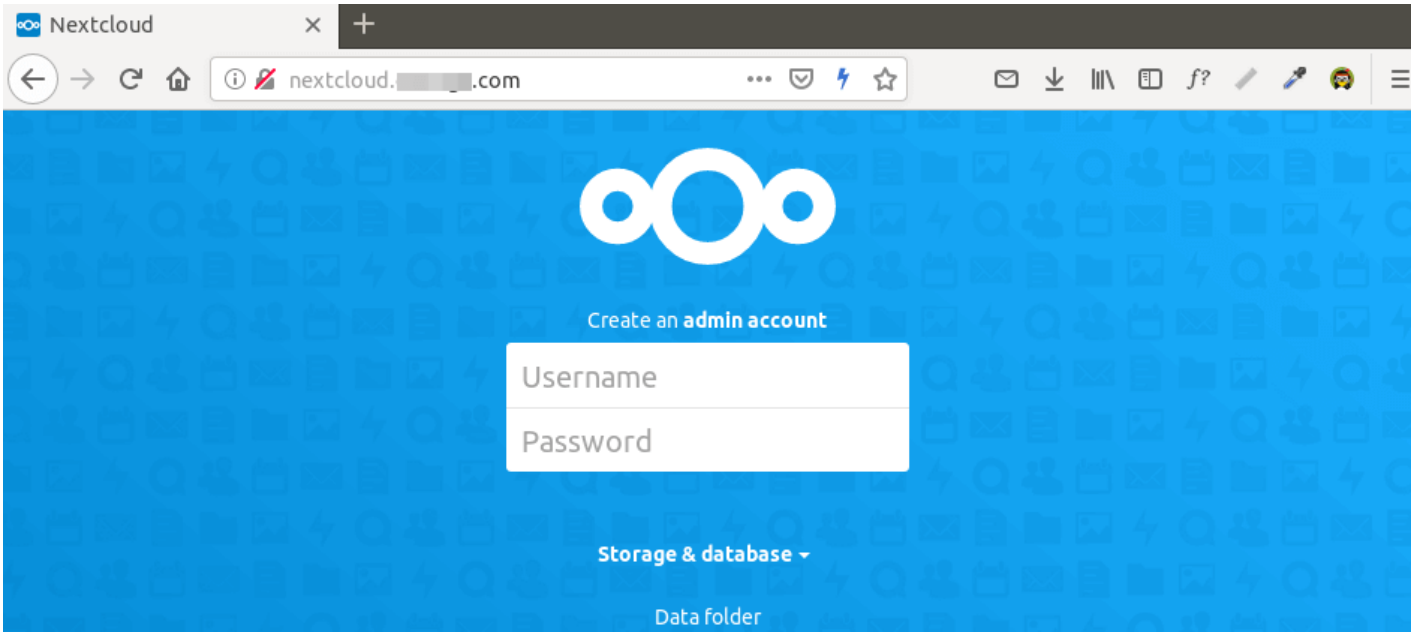
Run the following commands to install PHP modules required or recommended by NextCloud.

```
sudo apt install imagemagick php-imagick php7.4-common php7.4-mysql php7.4-fpm php7.4-gd php7.4-json php7.4-curl  php7.4-zip php7.4-xml php7.4-mbstring php7.4-bz2 php7.4-intl php7.4-bcmath php7.4-gmp
```

Step 5: Enable HTTPS

Now you can access the Nextcloud web install wizard in your web browser by entering the domain name for your Nextcloud installation.

```
nextcloud.example.com
```



If the web page can't load, you probably need to open port 80 in firewall.

```
sudo iptables -I INPUT -p tcp --dport 80 -j ACCEPT
```

And port 443 as well.

```
sudo iptables -I INPUT -p tcp --dport 443 -j ACCEPT
```

Before entering any sensitive information, we should enable secure HTTPS connection on Nextcloud. We can obtain a free TLS certificate from Let's Encrypt. Install Let's Encrypt client (certbot) from Ubuntu 20.04 repository.

```
sudo apt install certbot python3-certbot-nginx
```

`Python3-certbot-nginx` is the Nginx plugin. Next, run the following command to obtain a free TLS certificate using the Nginx plugin.

```
sudo certbot --nginx --agree-tos --redirect --hsts --staple-ocsp --email you@example.com -d nextcloud.example.com
```

Where:

- **-nginx**: Use the Nginx authenticator and installer
- **-agree-tos**: Agree to Let's Encrypt terms of service
- **-redirect**: Enforce HTTPS by adding 301 redirect.
- **-hsts**: Enable HTTP Strict Transport Security. This defends against SSL/TLS stripping attack.
- **-staple-ocsp**: Enable OCSP Stapling.
- **-email**: Email used for registration and recovery contact.
- **-d** flag is followed by a list of domain names, separated by comma. You can add up to 100

domain names.

You will be asked if you want to receive emails from EFF(Electronic Frontier Foundation). After choosing Y or N, your TLS certificate will be automatically obtained and configured for you, which is indicated by the message below.

```
IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/nextcloud.linuxbabe.com/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/nextcloud.linuxbabe.com/privkey.pem
   Your cert will expire on 2020-07-28. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot again
   with the "certonly" option. To non-interactively renew *all* of
   your certificates, run "certbot renew"
 - If you like Certbot, please consider supporting our work by:

   Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
   Donating to EFF:                    https://eff.org/donate-le
```

I found that Certbot may not be able to add HSTS header in the Nginx config file for Nextcloud. If you would like to enable HSTS (HTTP Strict Transport Security), then edit the file.

```
sudo nano /etc/nginx/conf.d/nextcloud.conf
```

We can then add the following line in the SSL server block to enable HSTS header. (If it's already there, then your configuration are fine.)

```
add_header Strict-Transport-Security "max-age=31536000" always;
```

Also, you can enable HTTP2 protocol by adding the option `http2`, which will speed up webpage loading.

```
listen 443 ssl http2; # managed by Certbot
```

Like below.

```
    listen 443 ssl http2; # managed by Certbot
    ssl_certificate /etc/letsencrypt/live/nextcloud.linuxbabe.com/fullchain.pem; # managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/nextcloud.linuxbabe.com/privkey.pem; # managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    add_header Strict-Transport-Security "max-age=31536000" always;


    ssl_trusted_certificate /etc/letsencrypt/live/nextcloud.linuxbabe.com/chain.pem; # managed by Certbot
    ssl_stapling on; # managed by Certbot
    ssl_stapling_verify on; # managed by Certbot

}
server {
    if ($host = nextcloud.linuxbabe.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name nextcloud.linuxbabe.com;
    return 404; # managed by Certbot


}
```

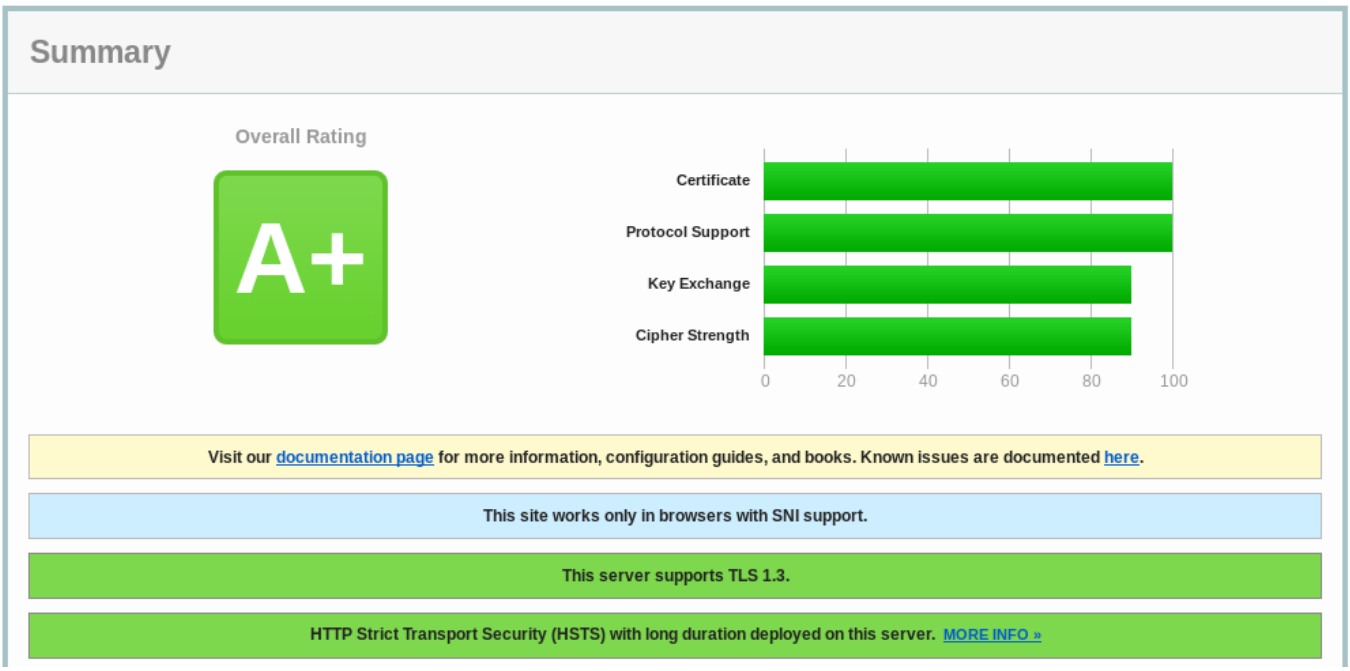
Save and close the file. Then text Nginx configurations.

```
sudo nginx -t
```

If the test is successful, reload Nginx for the change to take effect.

```
sudo systemctl reload nginx
```

The above configuration will get A+ score on SSL test.



Step 6: Finish the Installation in your Web Browser

Now you can access the Nextcloud web install wizard using HTTPS connection.

```
https://nextcloud.example.com
```

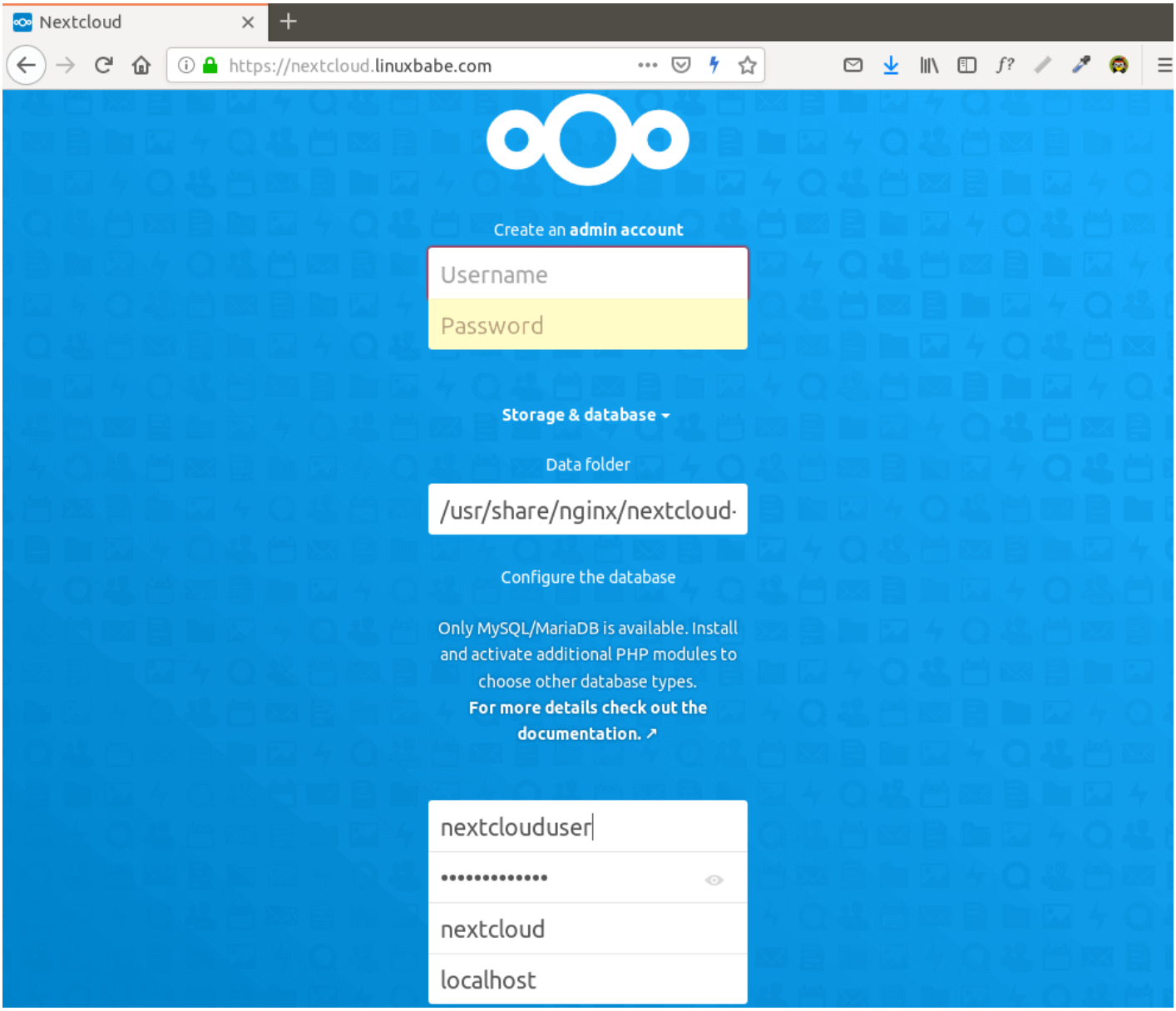
To complete the installation, you need to create an admin account, enter the path of Nextcloud data folder, enter database details you created in step 2. You can use the default `localhost` as host address, or you can enter `localhost:3306`, as MariaDB listens on port 3306.

The data folder is where users' files are stored. For security, it's best to place the data directory outside of Nextcloud webroot directory. So instead of storing users' files under `/usr/share/nginx/nextcloud/data/`, we can change it to **/usr/share/nginx/nextcloud-data**. which can be created with the following command:

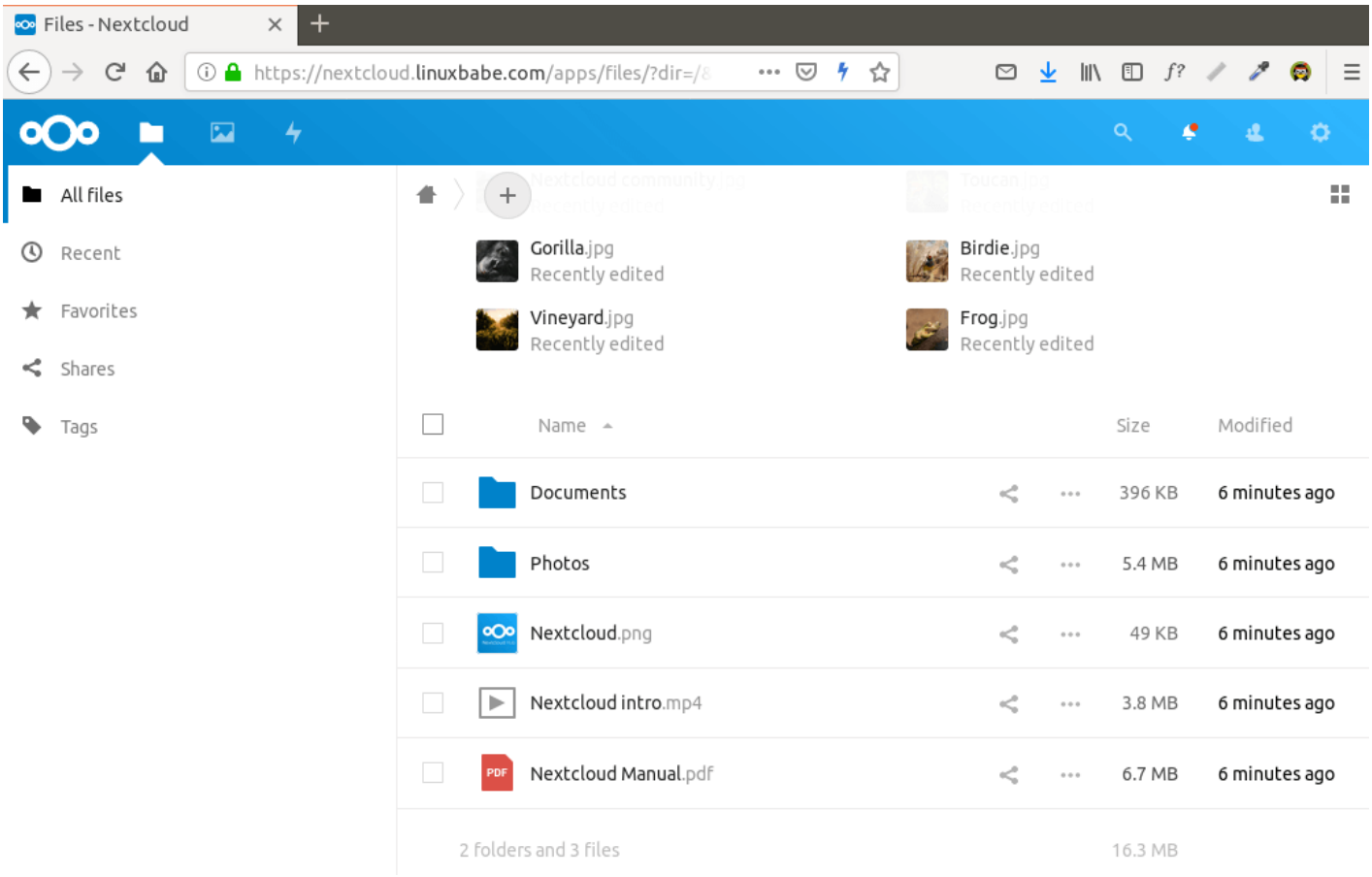
```
sudo mkdir /usr/share/nginx/nextcloud-data
```

Then make sure Nginx user (`www-data`) has write permission to the data directory.

```
sudo chown www-data:www-data /usr/share/nginx/nextcloud-data -R
```

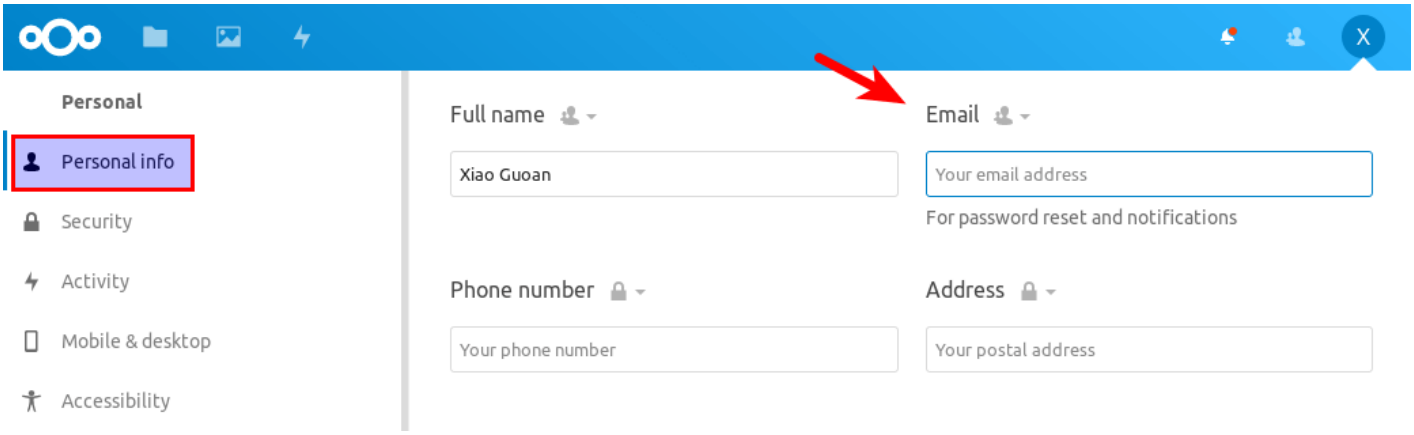


Click the `Finish Setup` button, you will see the Web interface of Nextcloud. Congrats! You can start using it as your private cloud storage.

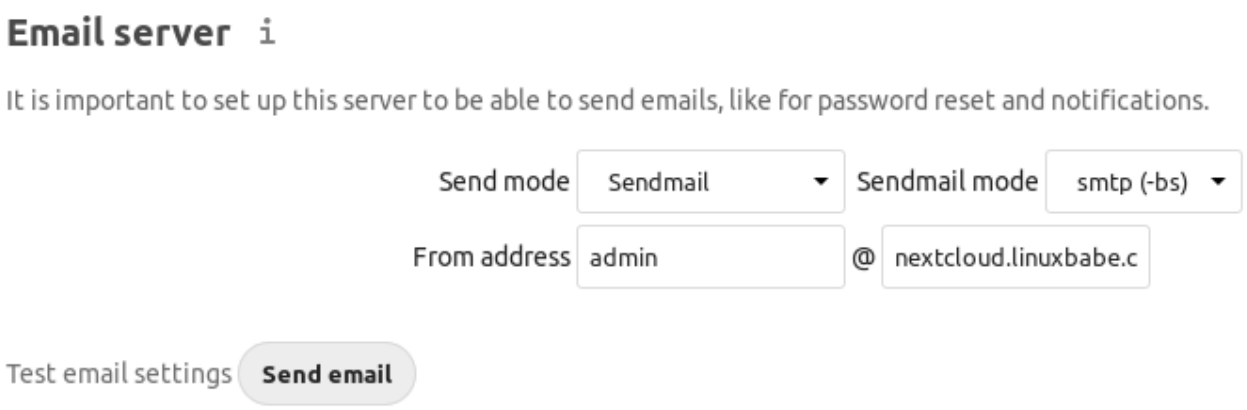


How to Set up NextCloud Email Notification

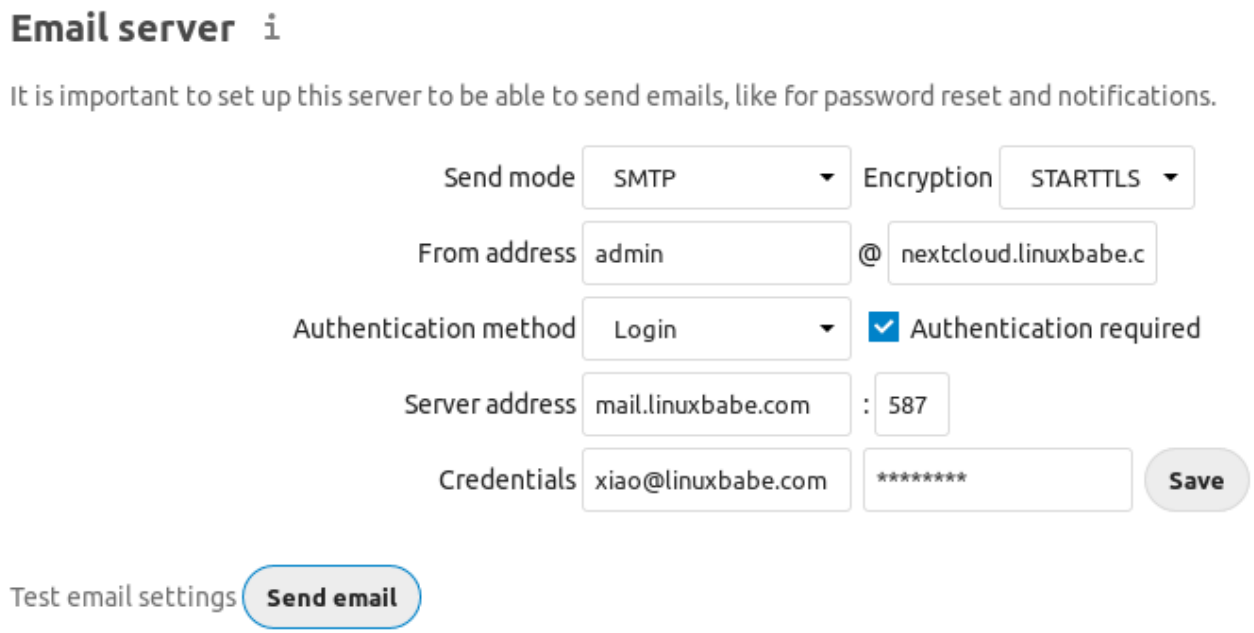
If your NextCloud instance will be used by more than one person, it's important that your NextCloud server can send transactional emails, such as password-resetting email. First, you should set an email address for your own account. Go to `Settings` -> `Personal Info` and set an email address for your account.



Then go to **Settings** -> **Basic settings**. You will find the email server settings. There are two send modes: `sendmail` and `smtp`. You can choose the `sendmail` mode if your NextCloud host has an SMTP server running.



If you would like to use an SMTP server running on another host, then choose `smtp` mode and enter the SMTP server address and login credentials like below. Choose STARTTLS for encryption.



For how to set up an email server, please check out the following tutorial. **Note** that I highly recommend running iRedMail mail server on a fresh clean OS. Installing iRedMail on an OS that has other web applications can fail, and likely break existing applications.

- How to easily set up a full-featured mail server on Ubuntu 20.04 with iRedMail

How to Reset Nextcloud User Password From Command Line

If you lost your admin account password, and you didn't set up email delivery in Nextcloud, then you need to reset the password by running the following command on your server. Replace `nextcloud_username` with your real username.

```
sudo -u www-data php /usr/share/nginx/nextcloud/occ user:resetpassword nextcloud_username
```

There are also other commands you might find useful. List available commands with:

```
sudo -u www-data php /usr/share/nginx/nextcloud/occ
```

or

```
sudo -u www-data php /usr/share/nginx/nextcloud/console.php
```

How to Move the Data Directory

In case you need to move the NextCloud data directory, there are 4 steps to accomplish this. First, you need to use the `cp` command to copy the data directory to the new directory. For example, the mount point of my external hard drive is `/media/linuxbabe/b43e4eea-9796-4ac6-9c48-2bcaa46353731`. I create the new data directory on the external hard drive.

```
sudo mkdir /media/linuxbabe/b43e4eea-9796-4ac6-9c48-2bcaa46353731/nextcloud-data/
```

Then I copy the original data directory to the new data directory. `-R` flag means the copy operation is recursive.

```
sudo cp /usr/share/nginx/nextcloud-data/* /media/linuxbabe/b43e4eea-9796-4ac6-9c48-2bcaa46353731/nextcloud-data/ -R
```

You also need to copy the `.ocdata` file.

```
sudo cp /usr/share/nginx/nextcloud-data/.ocdata /media/linuxbabe/b43e4eea-9796-4ac6-9c48-2bcaa46353731/nextcloud-data/
```

Next, you need to set `www-data` (Nginx user) as the owner.

```
sudo chown www-data:www-data /media/linuxbabe/b43e4eea-9796-4ac6-9c48-2bcaa46353731/nextcloud-data/ -R
```

Lastly, you need to edit the `config.php` file.

```
sudo nano /usr/share/nginx/nextcloud/config/config.php
```

Find the following line and change the value of `datadirectory`.

```
'datadirectory' => '/usr/share/nginx/nextcloud-data',
```

Save and close the file. Reload NextCloud web page and you are done.

Step 7: Increase PHP Memory Limit

The default PHP memory limit is 128MB. NextCloud recommends 512MB for better performance. To change PHP memory limit, edit the **php.ini** file.

```
sudo nano /etc/php/7.4/fpm/php.ini
```

Find the following line. (line 409)

```
memory_limit = 128M
```

Change the value.

```
memory_limit = 512M
```

Save and close the file. Alternatively, you can run the following command to change the value without manually opening the file.

```
sudo sed -i 's/memory_limit = 128M/memory_limit = 512M/g'  /etc/php/7.4/fpm/php.ini
```

Then reload **PHP-FPM** service for the changes to take effect.

```
sudo systemctl reload php7.4-fpm
```

Step 8: Set Up PHP to Properly Query System Environment Variables

Edit the **www.conf** file.

```
sudo nano /etc/php/7.4/fpm/pool.d/www.conf
```

Find the following line (line 396).

```
;clear_env = no
```

Remove the semicolon to uncomment this line.

```
clear_env = no
```

Save and close the file. Alternatively, you can run the following command to uncomment this line without manually opening the file.

```
sudo sed -i 's/;clear_env = no/clear_env = no/g'  /etc/php/7.4/fpm/pool.d/www.conf
```

Then reload **PHP-FPM** service for the changes to take effect.

```
sudo systemctl reload php7.4-fpm
```

Step 9: Increase Upload File Size Limit

The default maximum upload file size limit set by Nginx is 1MB. To allow uploading large files to your NextCloud server, edit the Nginx configuration file for NextCloud.

```
sudo nano /etc/nginx/conf.d/nextcloud.conf
```

We have already set the maximum file size in this file, as indicated by

```
client_max_body_size 512M;
```

You can change it if you prefer, like 1G.

```
client_max_body_size 1024M;
```

Save and close the file. Then reload Nginx for the changes to take effect.

```
sudo systemctl reload nginx
```

PHP also sets a limit of upload file size. The default maximum file size for uploading is 2MB. To increase the upload size limit, edit the PHP configuration file.

```
sudo nano /etc/php/7.4/fpm/php.ini
```

Find the following line (line 846).

```
upload_max_filesize = 2M
```

Change the value like below:

```
upload_max_filesize = 1024M
```

Save and close the file. Alternatively, you can run the following command to change the value without manually opening the file.

```
sudo sed -i 's/upload_max_filesize = 2M/upload_max_filesize = 1024M/g'
/etc/php/7.4/fpm/php.ini
```

Then restart PHP-FPM.

```
sudo systemctl restart php7.4-fpm
```

Step 10: Configure Redis Cache for NextCloud

If you go to your NextCloud **settings** -> **overview** page, you might see the following warning:

```
No memory cache has been configured. To enhance your performance please configure a memcache
if available.
```

We will enable memory caching for nextCloud by using Redis. Run the following command to install Redis server from Ubuntu repository.

```
sudo apt install redis-server
```

You can check the version with:

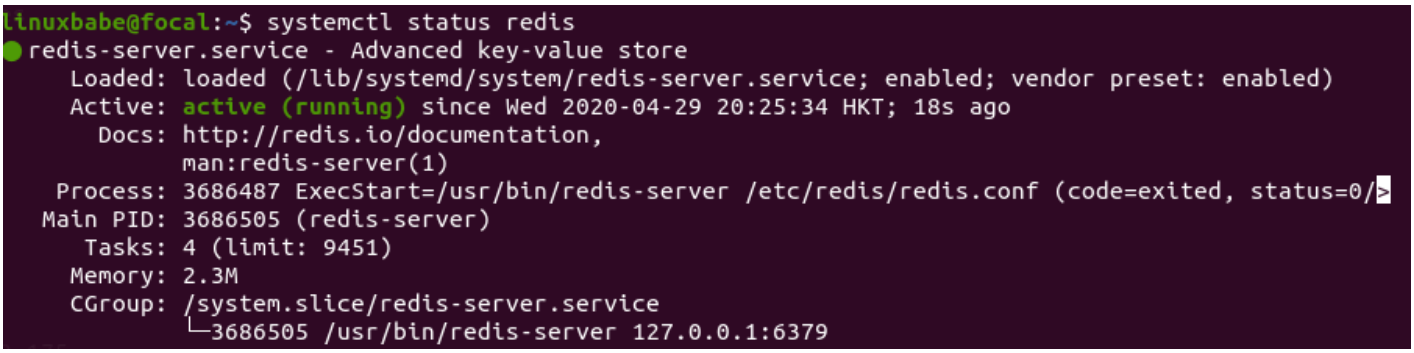
```
redis-server -v
```

Sample output:

```
Redis server v=5.0.7 sha=00000000:0 malloc=jemalloc-5.2.1 bits=64 build=636cde3b5c7a3923
```

Now we can check if redis server is running.

```
systemctl status redis
```



Hint: If the above command didn't quit immediately, you can press the Q key to gain back control of the terminal.

From the above screenshot, we can see that it's running and auto-start is enabled. If for any reason it's not running, execute the following command:

```
sudo systemctl start redis-server
```

And if auto-start at boot time is not enabled, you can use the following command to enable it:

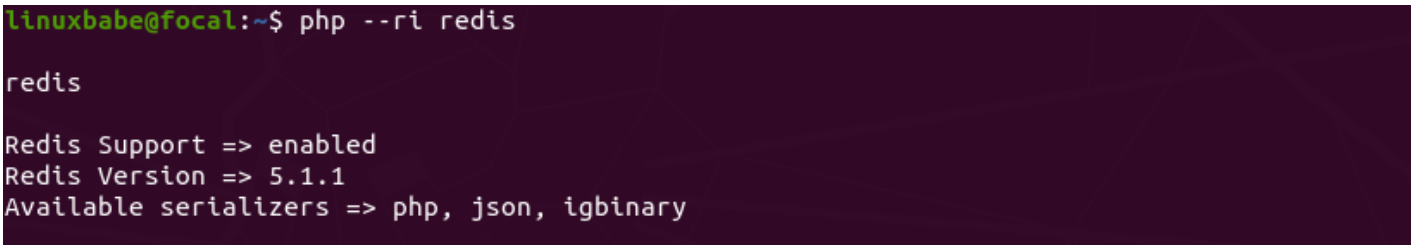
```
sudo systemctl enable redis-server
```

In order to configure Redis as a cache for nextCloud, we need to install the PHP extension for interfacing with Redis.

```
sudo apt install php-redis
```

Check if the extension is enabled.

```
php --ri redis
```



We can see that Redis extension is enabled. If it's not enabled, run the following command:

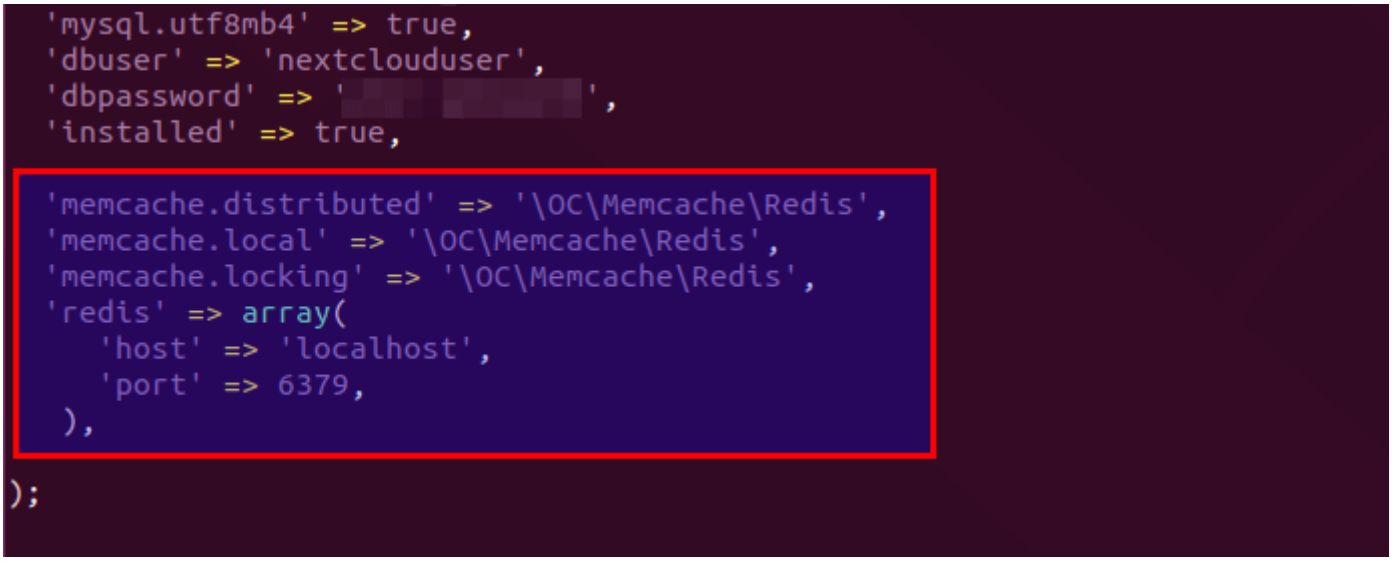
```
sudo phpenmod redis
```

Next, edit nextCloud configuration file.

```
sudo nano /usr/share/nginx/nextcloud/config/config.php
```

Add the following lines above the ending `);` line.

```
'memcache.distributed' => '\OC\Memcache\Redis',
'memcache.local' => '\OC\Memcache\Redis',
'memcache.locking' => '\OC\Memcache\Redis',
'redis' => array(
     'host' => 'localhost',
     'port' => 6379,
     ),
```



Save and close the file. Then restart Nginx and PHP-FPM.

```
sudo systemctl restart nginx php7.4-fpm
```

Now go to NextCloud **settings** -> **overview** page again and refresh the web page, the warning about memory caching should be gone.

Adding Missing Indexes

If you see the following message in the NextCloud **Settings** -> **Overview** page,

```
The database is missing some indexes. Due to the fact that adding indexes on big tables could take some time they were not added automatically.
```

Then you need to manually add those indexes. Change to the Nextcloud webroot directory.

```
cd /usr/share/nginx/nextcloud/
```

Run the following command to add indexes to the Nextcloud database.

```
sudo -u www-data php occ db:add-missing-indices
```

```
linuxbabe@focal:/usr/share/nginx/nextcloud$ sudo -u www-data php occ db:add-missing-indices
Check indices of the share table.
Check indices of the filecache table.
Check indices of the twofactor_providers table.
Check indices of the login_flow_v2 table.
Check indices of the whats_new table.
Check indices of the cards table.
Check indices of the cards_properties table.
Check indices of the calendarobjects_props table.
Adding calendarobject_calid_index index to the calendarobjects_props table, this can take some time
...
calendarobjects_props table updated successfully.
Check indices of the schedulingobjects table.
Adding schedulobj_principuri_index index to the schedulingobjects table, this can take some time...
schedulingobjects table updated successfully.
```

Now if you refresh the NextCloud **Settings** -> **Overview** page, the warning about missing indexes should be gone.

Conversion to Big Int

If you see the following message in the NextCloud **Settings** -> **Overview** page,

```
Some columns in the database are missing a conversion to big int. Due to the fact that
changing column types on big tables could take some time they were not changed automatically.
```

Then you need to manually change the column type. Change to the Nextcloud webroot directory.

```
cd /usr/share/nginx/nextcloud/
```

Change your Nextcloud into maintenance mode to prevent users from logging in and making changes.

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

Then run the following command to change the column type.

```
sudo -u www-data php occ db:convert-filecache-bigint
```

Once it’s done, switch off the maintenance mode.

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

```
linuxbabe@focal:/usr/share/nginx/nextcloud$ sudo -u www-data php occ maintenance:mode --on
Maintenance mode enabled
linuxbabe@focal:/usr/share/nginx/nextcloud$ sudo -u www-data php occ db:convert-filecache-bigint
Nextcloud is in maintenance mode - no apps have been loaded

Following columns will be updated:

* mounts.storage_id
* mounts.root_id
* mounts.mount_id

This can take up to hours, depending on the number of files in your instance!
Continue with the conversion (y/n)? [n] y
linuxbabe@focal:/usr/share/nginx/nextcloud$ sudo -u www-data php occ maintenance:mode --off
Maintenance mode disabled
linuxbabe@focal:/usr/share/nginx/nextcloud$
```

Now if you refresh the NextCloud **Settings** -> **Overview** page, the warning about big int should be gone.

Revision #1
Created 11 July 2021 18:47:16 by Admin
Updated 11 July 2021 18:47:50 by Admin