There is a new version of this tutorial available for Ubuntu 22.04 (Jammy Jellyfish).

How to Install Nextcloud with Nginx and Let's Encrypt SSL on Ubuntu 20.04 LTS

Nextcloud is a free (Open Source) Dropbox-like software, a fork of the ownCloud project. Nextcloud is written in PHP and JavaScript, it supports many database systems such as MySQL/MariaDB, PostgreSQL, Oracle Database, and SQLite.

This tutorial exists for these OS versions

- Ubuntu 22.04 (Jammy Jellyfish)
- **Ubuntu 20.04 (Focal Fossa)**
- Ubuntu 18.04 (Bionic Beaver)

On this page

In order to keep your files synchronized between Desktop and your own server, Nextcloud provides applications for Windows, Linux, and Mac desktops and a mobile app for Android and iOS.

Nextcloud is not just a Dropbox clone, it provides additional features like Calendar, Contacts, Schedule tasks, and streaming media with Ampache etc.

In this tutorial, we will show you how to install and configure the latest Nextcloud release (at the time of writing this, the latest release is 18) on an Ubuntu 20.04 server. We will run Nextcloud with an Nginx web server and PHP7.4-FPM and use MariaDB server as the database system.

Prerequisites

- Ubuntu 20.04
- Root privileges

What we will do

- Install Nginx Webserver
- Install and Configure PHP7.4-FPM
- Install and Configure MySQL Server
- Generate SSL Letsencrypt
- Download Nextcloud 18
- Configure Nginx Virtual Host for Nextcloud
- UFW Firewall Configuration
- Nextcloud Post-Installation

Step 1 - Install Nginx Webserver

The first step we will do in this nextcloud guide is to install the Nginx web server. We will be using the Nginx web server instead of Apache webserver.

Log in to the server and update the repository, then install the Nginx web server using the apt command as shown below.

```
sudo apt update
sudo apt install nginx -y
```

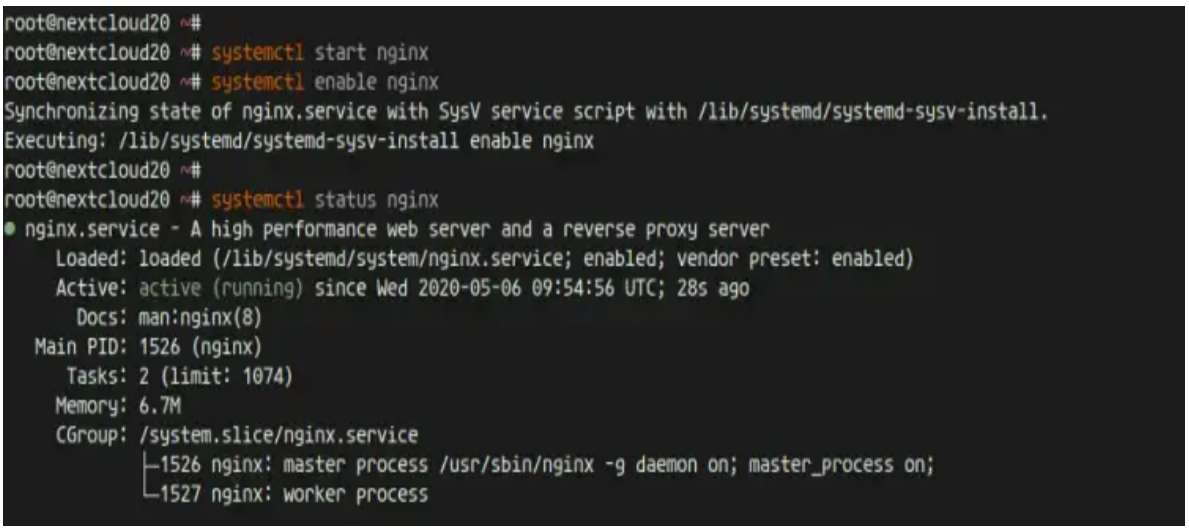
After the installation is complete, start the Nginx service and enable the service to launch every time at system boot using systemctl.

```
systemctl start nginx
systemctl enable nginx
```

The Nginx service is up and running, check it using the following command.

```
systemctl status nginx
```

And you will get the result as below.



As a result, the Nginx web server has been installed on Ubuntu 20.04.

Step 2 - Install and Configure PHP7.4-FPM

By default, the Ubuntu 20.04 comes with default version PHP 7.4.

Install PHP and PHP-FPM packages needed by Nextcloud using the apt command below.

```
sudo apt install php-fpm php-curl php-cli php-mysql php-gd ph
p-common php-xml php-json php-intl php-pear php-imagick php-d
ev php-common php-mbstring php-zip php-soap php-bz2 -y
```

After the installation is complete, we will configure the php.ini files for php-fpm and php-cli.

Go to the '/etc/php/7.4' directory.

```
cd /etc/php/7.4/
```

Edit the php.ini files for php-fpm and php-cli using vim.

```
vim fpm/php.ini
vim cli/php.ini
```

Uncomment the 'date.timezone' line and change the value with your own timezone.

```
date.timezone = Asia/Jakarta
```

Uncomment the 'cgi.fix_pathinfo' line and change the value to '0'.

```
cgi.fix_pathinfo=0
```

Save and exit.

Next, edit the php-fpm pool configuration 'www.conf'.

```
vim fpm/pool.d/www.conf
```

Uncomment those lines below.

```
env[HOSTNAME] = $HOSTNAME
env[PATH] = /usr/local/bin:/usr/bin:/bin
env[TMP] = /tmp
env[TMPDIR] = /tmp
env[TEMP] = /tmp
```

Save and exit.

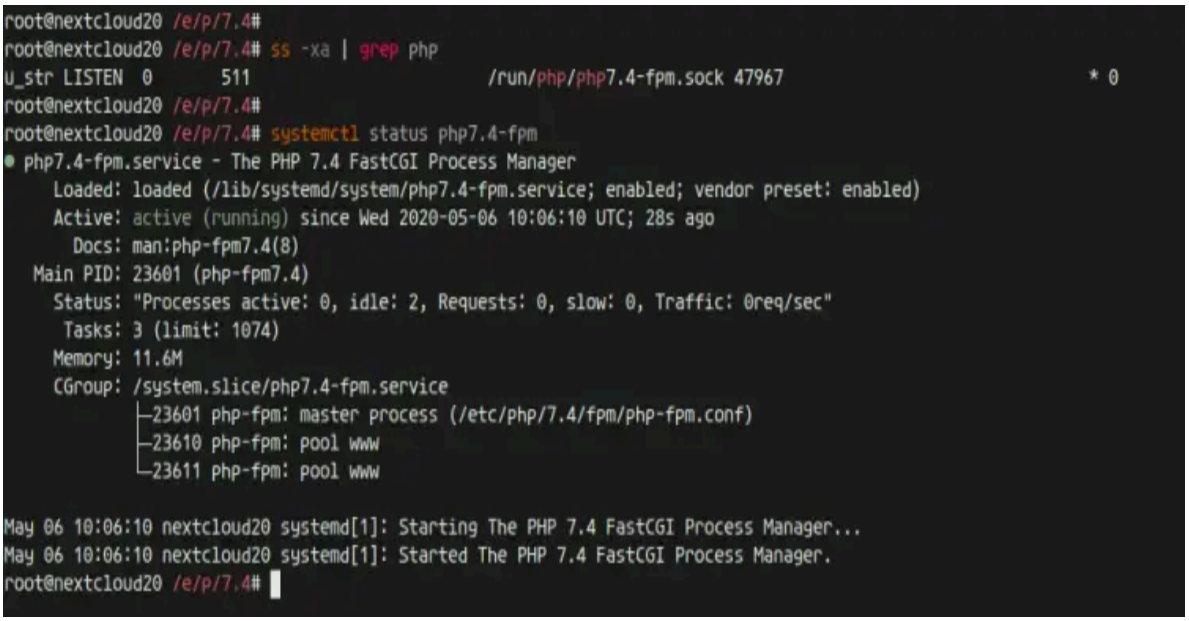
Restart the PHP7.4-FPM service and enable it to launch every time on system boot.

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

Now check the PHP-FPM service using the following command.

```
ss -xa | grep php
systemctl status php7.4-fpm
```

And you will get the php-fpm is up and running under the sock file '/run/php/php7.4-fpm.sock'.



Step 3 - Install and Configure MariaDB Server

In this step, we will install the latest MariaDB version and create a new database for the nextcloud installation. The latest version MariaDB packages are available on the repository by default.

Install MariaDB server's latest version using the apt command below.

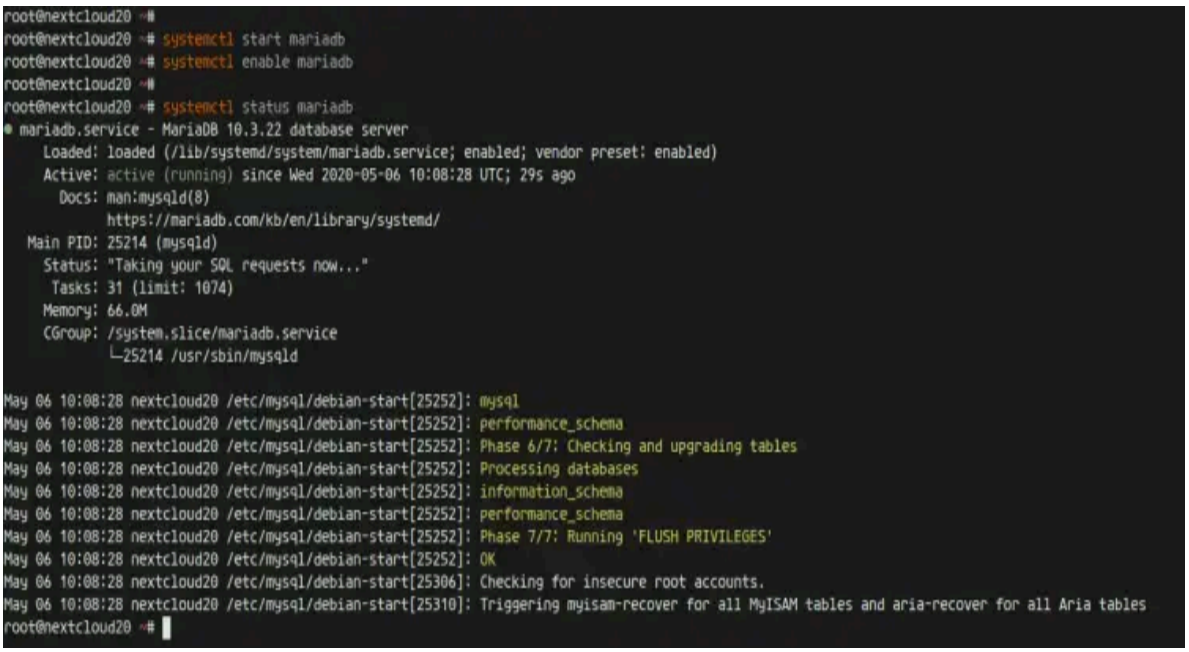
```
sudo apt install mariadb-server -y
```

After the installation is complete, start the MariaDB service and enable it to launch everytime at system boot.

```
systemctl start mariadb
systemctl enable mariadb
```

Now check the MySQL service using the following command.

```
systemctl status mariadb
```



The MariaDB server is up and running on Ubuntu 20.04.

Next, we will configure the MariaDB root password using the

'mysql_secure_installation' command.

Run the following command.

```
mysql_secure_installation
```

And you will be asked for some configuraiton of MariaDB Server. Also, type the new root password for MariaDB Server.

```
Enter current password for root (enter for none): Press Enter
Set root password? [Y/n] Y
Remove anonymous users? [Y/n] Y
Disallow root login remotely? [Y/n] Y
Remove test database and access to it? [Y/n] Y
Reload privilege tables now? [Y/n] Y
```

And the MariaDB root password has been set up.

Next, we will create a new database for nextcloud installation. We will create a new database named 'nextcloud_db' with the user 'nextclouduser' and password 'Nextclouduser421@'.

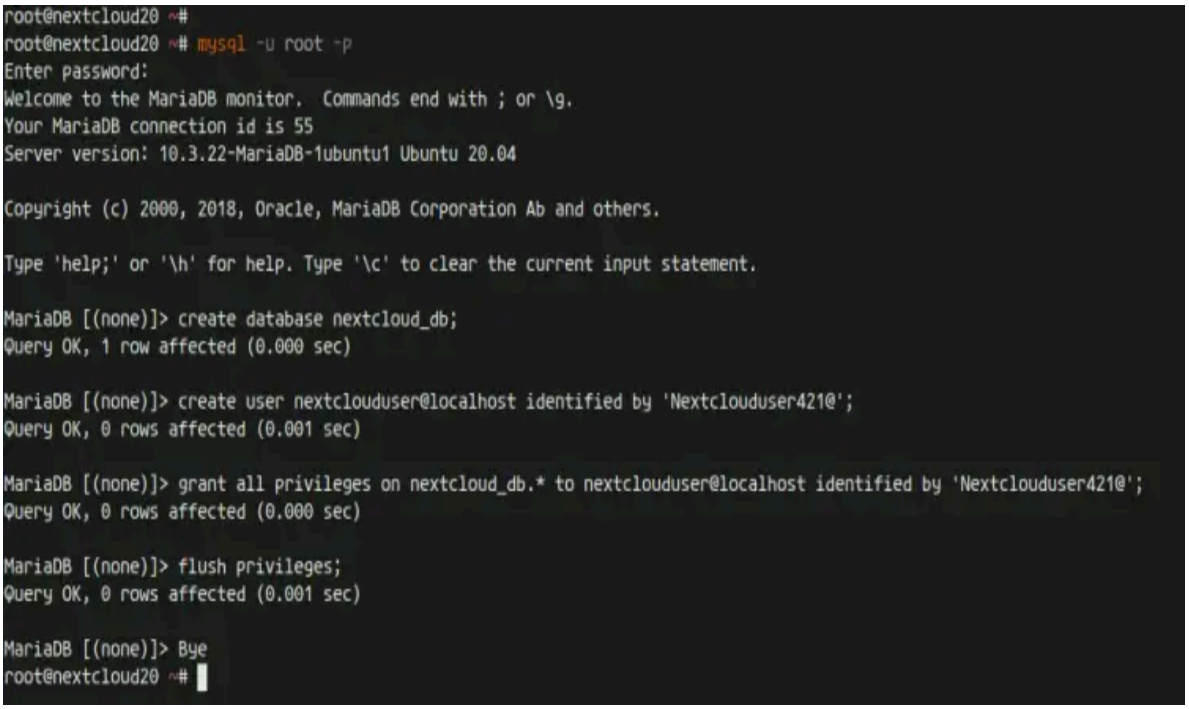
Login to the MySQL shell as a root user with mysql command.

```
mysql -u root -p
TYPE THE MYSQL ROOT PASSWORD
```

Now create the database and user with the password by running following MySQL queries.

```
create database nextcloud_db;
create user nextclouduser@localhost identified by 'Nextcloudu
ser421@';
grant all privileges on nextcloud_db.* to nextclouduser@local
host identified by 'Nextclouduser421@';
flush privileges;
```

And the new database and user for the nextcloud installation has been created.



The MariaDB installation and configuration for nextcloud has been completed.

Step 4 - Generate SSL Letsencrypt

In this tutorial, we will secure nextcloud using free SSL from Letsencrypt, and we will generate certificates files using the letsencrypt tool.

If you do not have a domain name or install nextcloud on the local computer, you can generate the Self-Signed certificate using OpenSSL.

Install the 'letsencrypt' tool using the apt command below.

```
sudo apt install certbot -y
```

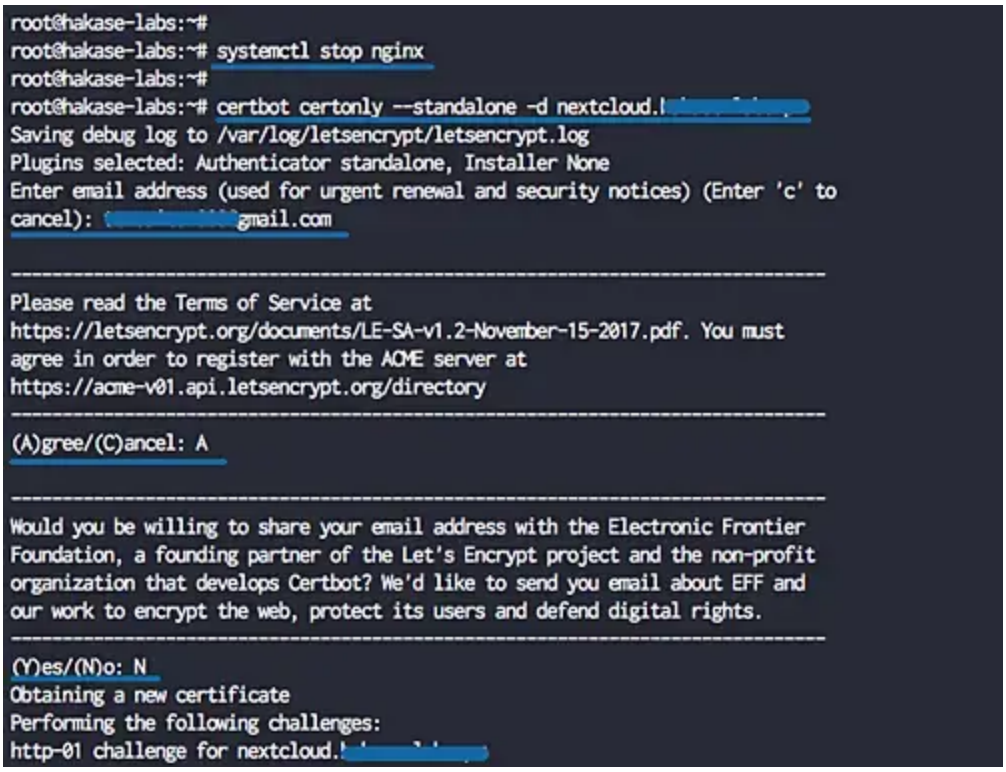
After the installation is complete, stop the nginx service.

```
systemctl stop nginx
```

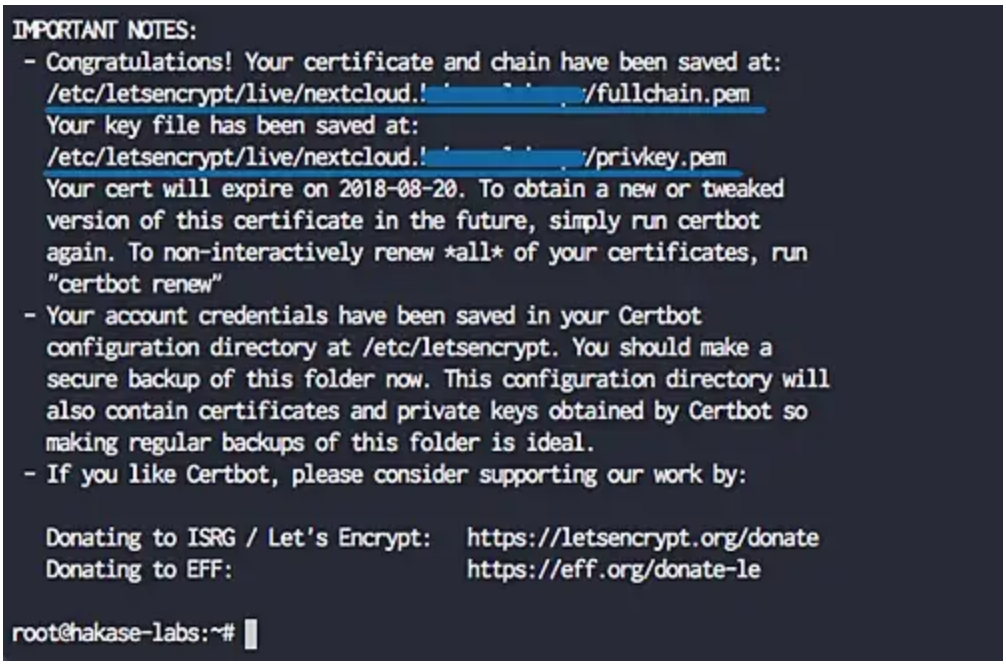
Next, we will generate the SSL certificates for our domain name 'nextcloud.hakase-labs.io' using the cerbot command line. Run the command below.

```
certbot certonly --standalone -d cloud.hakase-labs.io
```

You will be asked for the email address, and it's used for the renew notification. For the Letsencrypt TOS agreement, type 'A' to agree and for the share email address, you can type 'N' for No.



When it's complete, you will get the result as shown below.



The SSL certificates Letsencrypt for the netxcloud domain name has been generated, all located at the '/etc/letsencrypt/live/your-domain' directory.

Step 5 - Download Nextcloud

Before downloading the nextcloud source code, make sure the unzip package is installed on the system. If you don't have the package, install it using the apt command below.

```
sudo apt install wget unzip zip -y
```

Now go to the '/var/www' directory and download the latest version of Nextcloud using the following command.

```
cd /var/www/
wget -q https://download.nextcloud.com/server/releases/latest.zip
```

Extract the Nextcloud source code and you will get a new directory 'netxcloud', change the ownership of the nextcloud directory to user 'www-data'.

```
unzip -qq latest.zip
sudo chown -R www-data:www-data /var/www/nextcloud
```

As a result, the Nextcloud has been downloaded under the '/var/www/nextcloud' directory, and it will be the web root directory.

Step 6 - Configure Nginx Virtual Host for Nextcloud

In this step, we will configure the nginx virtual host for nextcloud. We will configure nextcloud to run under the HTTPS connection and will force the HTTP connection automatically to the secure HTTPS connection.

Now go to the '/etc/nginx/sites-available' directory and create a new virtual host file 'nextcloud'.

```
cd /etc/nginx/sites-available/
vim nextcloud
```

There, paste the following nextcloud virtual host configuration.

```
upstream php-handler {
    #server 127.0.0.1:9000;
    server unix:/var/run/php/php7.4-fpm.sock;
}

server {
    listen 80;
    listen [::]:80;
    server_name cloud.hakase-labs.io;
    # enforce https
    return 301 https://$server_name:443$request_uri;
}
```

```
server {
    listen 443 ssl http2;
    listen [::]:443 ssl http2;
    server_name cloud.hakase-labs.io;

    # Use Mozilla's guidelines for SSL/TLS settings
    # https://mozilla.github.io/server-side-tls/ssl-config-gen
erator/
    # NOTE: some settings below might be redundant
    ssl_certificate /etc/letsencrypt/live/cloud.hakase-labs.i
o/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/cloud.hakase-lab
s.io/privkey.pem;

    # Add headers to serve security related headers
    # Before enabling Strict-Transport-Security headers please
read into this
    # topic first.
    #add_header Strict-Transport-Security "max-age=15768000; i
ncludeSubDomains; preload;" always;
    #
    # WARNING: Only add the preload option once you read about
    # the consequences in https://hstspreload.org/. This optio
n
    # will add the domain to a hardcoded list that is shipped
    # in all major browsers and getting removed from this list
    # could take several months.
    add_header Referrer-Policy "no-referrer" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header X-Download-Options "noopen" always;
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-Permitted-Cross-Domain-Policies "none" alway
s;
    add_header X-Robots-Tag "none" always;
    add_header X-XSS-Protection "1; mode=block" always;

    # Remove X-Powered-By, which is an information leak
    fastcgi_hide_header X-Powered-By;

    # Path to the root of your installation
    root /var/www/nextcloud;

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

    # The following 2 rules are only needed for the user_webfi
nger 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;

    # The following rule is only needed for the Social app.
    # Uncomment it if you're planning to use this app.
    #rewrite ^/.well-known/webfinger /public.php?service=webfi
nger last;

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

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

    # Enable gzip but do not remove ETag headers
    gzip on;
    gzip_vary on;
    gzip_comp_level 4;
    gzip_min_length 256;
    gzip_proxied expired no-cache no-store private no_last_mod
ified no_etag auth;
    gzip_types application/atom+xml application/javascript app
lication/json application/ld+json application/manifest+json ap
plication/rss+xml application/vnd.geo+json application/vnd.ms-
fontobject application/x-font-ttf application/x-web-app-manife
st+json application/xhtml+xml application/xml font/opentype im
age/bmp image/svg+xml image/x-icon text/cache-manifest text/cs
s text/plain text/vcard text/vnd.rim.location.xloc text/vtt te
xt/x-component text/x-cross-domain-policy;

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

    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\/upda
te|status|ocs\/v[12]|updater\/.+|oc[ms]-provider\/.+)\.php
(?:$|\/) {
        fastcgi_split_path_info ^(.+?\.php)(\/.*|)$;
        set $path_info $fastcgi_path_info;
        try_files $fastcgi_script_name =404;
        include fastcgi_params;
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_s
cript_name;
        fastcgi_param PATH_INFO $path_info;
        fastcgi_param HTTPS on;
        # Avoid sending the security headers twice
        fastcgi_param modHeadersAvailable true;
        # Enable pretty urls
        fastcgi_param front_controller_active true;
        fastcgi_pass php-handler;
        fastcgi_intercept_errors on;
        fastcgi_request_buffering off;
    }

    location ~ ^\/(?:updater|oc[ms]-provider)(?:$|\/) {
        try_files $uri/ =404;
        index index.php;
    }

    # Adding the cache control header for js, css and map file
s
    # Make sure it is BELOW the PHP block
    location ~ \.(?:css|js|woff2?|svg|gif|map)$ {
        try_files $uri /index.php$request_uri;
        add_header Cache-Control "public, max-age=15778463";
        # Add headers to serve security related headers (It is
intended to
        # have those duplicated to the ones above)
        # Before enabling Strict-Transport-Security headers pl
ease read into
        # this topic first.
        #add_header Strict-Transport-Security "max-age=1576800
0; includeSubDomains; preload;" always;
        #
        # WARNING: Only add the preload option once you read a
bout
        # the consequences in https://hstspreload.org/. This o
ption
        # will add the domain to a hardcoded list that is ship
ped
        # in all major browsers and getting removed from this
list
        # could take several months.
```

```
        add_header Referrer-Policy "no-referrer" always;
        add_header X-Content-Type-Options "nosniff" always;
        add_header X-Download-Options "noopen" always;
        add_header X-Frame-Options "SAMEORIGIN" always;
        add_header X-Permitted-Cross-Domain-Policies "none" al
ways;
        add_header X-Robots-Tag "none" always;
        add_header X-XSS-Protection "1; mode=block" always;

        # Optional: Don't log access to assets
        access_log off;
    }

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

Save and exit.

Enable the virtual host and test the configuration, and make sure there is no error.

```
ln -s /etc/nginx/sites-available/nextcloud /etc/nginx/sites-e
nabled/
nginx -t
```

Now restart PHP7.4-FPM service and nginx service using the systemctl command below.

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

The Nginx virtual host configuration for nextcloud has been created.

Step 7 - Configure UFW Firewall

In this tutorial, we will turn on the firewall, and we will be using the UFW firewall for Ubuntu.

Add the SSH, HTTP and HTTPS to the UFW firewall list using the command below.

```
for svc in ssh http https
do
ufw allow $svc
done
```

After that, enable the UFW firewall and check the allowed service and port.

```
ufw enable
ufw status numbered
```

And you will get the HTTP port 80 and HTTPS port 443 is on the list.

Step 8 - Nextcloud Post-Installation

Open your web browser and type the nextcloud URL address.

http://cloud.hakase-labs.io/

And you will be redirected to the secure HTTPS connection.

On the Top page, we need to create the admin user for nextcloud, type the admin user password. On the 'Data folder' configuration, type the full path of the 'data' directory '/var/www/nextcloud/data'.

Scroll the page to the bottom, and you will get the database configuration. Type the database info that we've created in step 3 and then click the 'Finish Setup' button.

If you check the option 'Install recommended apps', you will get the following page.

Nextcloud is installing additional recommended applications for you.

And after the installation is complete, you will get the Nextcloud Dashboard as below.

The Nextcloud 18 installation with Nginx web server and MySQL database on Ubuntu 20.04 has been completed successfully.

Reference

- https://docs.nextcloud.com/

About Muhammad Arul

Muhammad Arul is a freelance system administrator and technical writer. He is working with Linux Environments for more than 5 years, an Open Source enthusiast and highly motivated on Linux installation and troubleshooting. Mostly working with RedHat/CentOS Linux and Ubuntu/Debian, Nginx and Apache web server, Proxmox, Zimbra Administration, and Website Optimization. Currently learning about OpenStack and Container Technology.

Share this page:

21 Comment(s)

< Read 21 Comment(s) >

Tutorial Info

Author:	Muhammad Arul
Tags:	linux, ubuntu, web server
Comments:	Read or add comments

Share This Page

Popular Tutorials

ISPConfig Perfect Multiserver setup on Ubuntu 20.04 and Debian 10

Managing Xen With Xen-Tools, Xen-Shell, And Argo

Merging Multiple Apache Access Logs Into One Overall Access Log

Linux sha1sum Command Tutorial for Beginners (with Examples)

How to Install Tiki Wiki on Ubuntu 22.04

How to Install Consul Server on Ubuntu 22.04

How to Install Cachet Status Page on Ubuntu 24.04 Server