How to Easily Integrate OnlyOffice and NextCloud Using Docker

Previously I've written about integrating OnlyOffice and NextCloud the traditional way, which is a long process. Now you can easily integrate OnlyOffice and NextCloud using Docker.

There's a new Github repository created by OnlyOffice developer team to help with integration of OnlyOffice document server and NextCloud, which allows users to create and edit Office documents directly from NextCloud. As a matter of fact, it also supports integration of OnlyOffice and OwnCloud. Since most of us are NextCloud users, I will focus on NextCloud only.

Using this method, both OnlyOffice document server and NextCloud will be installed as Docker container application and Nginx will be used as web server. I will show you how to integrate these two and how to enable HTTPS with Let's Encrypt.

Prerequisites

To follow this tutorial, you need

- A server on which port 80 and 443 are available, and at least 1GB of RAM, preferrably 2GB of RAM. I recommend Vultr where you can get a high-performance 2GB RAM Linux VPS for only $10 per month.
- A domain name. I bought my domain name from NameCheap. Not only is their price lower than Godaddy, but also they give whois privacy protection for free.

Install Docker on Your Server

First we need to install Docker and Docker Compose, the latest version of which can be installed from Docker's official repository. The following steps are for Ubuntu 16.04. Users of other Linux distributions can check out the official installation instructions.

Create a source list file for Docker repository.

```
sudo nano /etc/apt/sources.list.d/docker.list
```

Copy the following line and paste it into the file.

```
deb [arch=amd64] https://download.docker.com/linux/ubuntu xenial stable
```

Save and close the file. Then import Docker's PGP key by running the command below.

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Since this repository uses HTTPS connection, we also need to install `apt-transport-https` and `ca-certificates` package.

```
sudo apt install apt-transport-https ca-certificates
```

Next, update package index and install the latest version of Docker CE (Community Edition).

```
sudo apt update


sudo apt install docker-ce
```

To install the latest version of Docker Compose, run the following commands.

```
sudo curl -L https://github.com/docker/compose/releases/download/1.17.1/docker-compose-`uname -s`-`uname -m` -o /usr/local/bin/docker-compose


sudo chmod +x /usr/local/bin/docker-compose
```

Check Docker version.

```
docker -v
```

Sample output:

```
Docker version 17.09.0-ce, build afdb6d4
```

Check Docker Compose version.

```
docker-compose --version
```

Sample output:

```
docker-compose version 1.17.1, build 6d101fb
```

Once installed, the Docker daemon should be automatically started. You can check it with:

```
systemctl status docker
```

Output:

```
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Sat 2017-11-11 12:40:23 UTC; 3min 32s ago
     Docs: https://docs.docker.com
 Main PID: 4090 (dockerd)
   CGroup: /system.slice/docker.service
           ├─4090 /usr/bin/dockerd -H fd://
           └─4159 docker-containerd -l unix:///var/run/docker/libcontainerd/docker-
containerd.sock --metrics-inter
```

If it’s not running, then start the daemon with this command:

```
sudo systemctl start docker
```

And enable auto-start with system boot:

```
sudo systemctl enable docker
```

Clone the Github Repo

We will use git to clone the Github repository.

```
git clone --recursive https://github.com/ONLYOFFICE/docker-onlyoffice-owncloud

cd docker-onlyoffice-owncloud
```

```
git submodule update --remote
```

Edit the `docker-compose.yml` file.

```
nano docker-compose.yml
```

Find the 5th line.

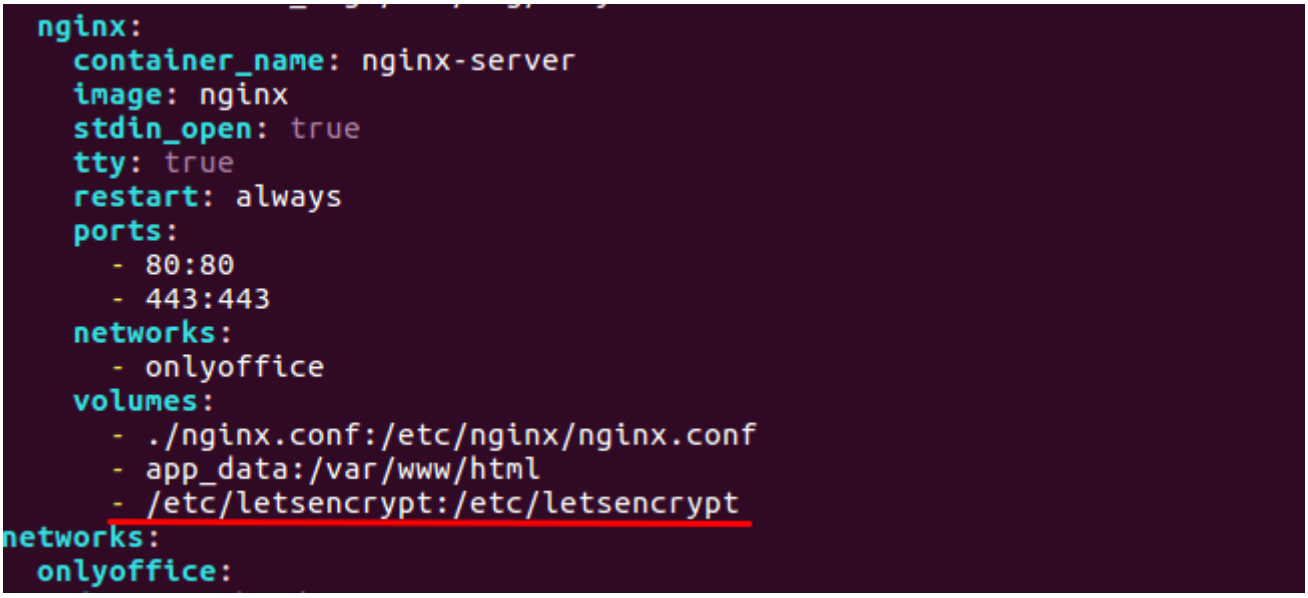
```
image: owncloud:fpm
```

Since we want to install NextCloud, change this line to :

```
image: nextcloud:fpm
```

If you want to enable HTTPS with Let's Encrypt, add the following line in `nginx` service. This line tells Docker to mount directory `/etc/letsencrypt` on the host into Nginx container.

```
- /etc/letsencrypt:/etc/letsencrypt
```



Save and close the file. Then edit `nginx.conf` file in `docker-onlyoffice-owncloud` directory. This file will be mounted as `/etc/nginx/nginx.conf` in the Nginx container.

```
nano nginx.conf
```

Add a `server_name` directive in the server block. Don't forget to set A record for your domain name.

```
server {
     listen 80;
```

```
    server_name cloud.example.com;
....
```

Also, add the following lines in the server block because later on we will use Certbot webroot plug-in to obtain SSL certificate.

```
location ~ /.well-known/acme-challenge {
      root /var/www/html/;
      allow all;
}
```

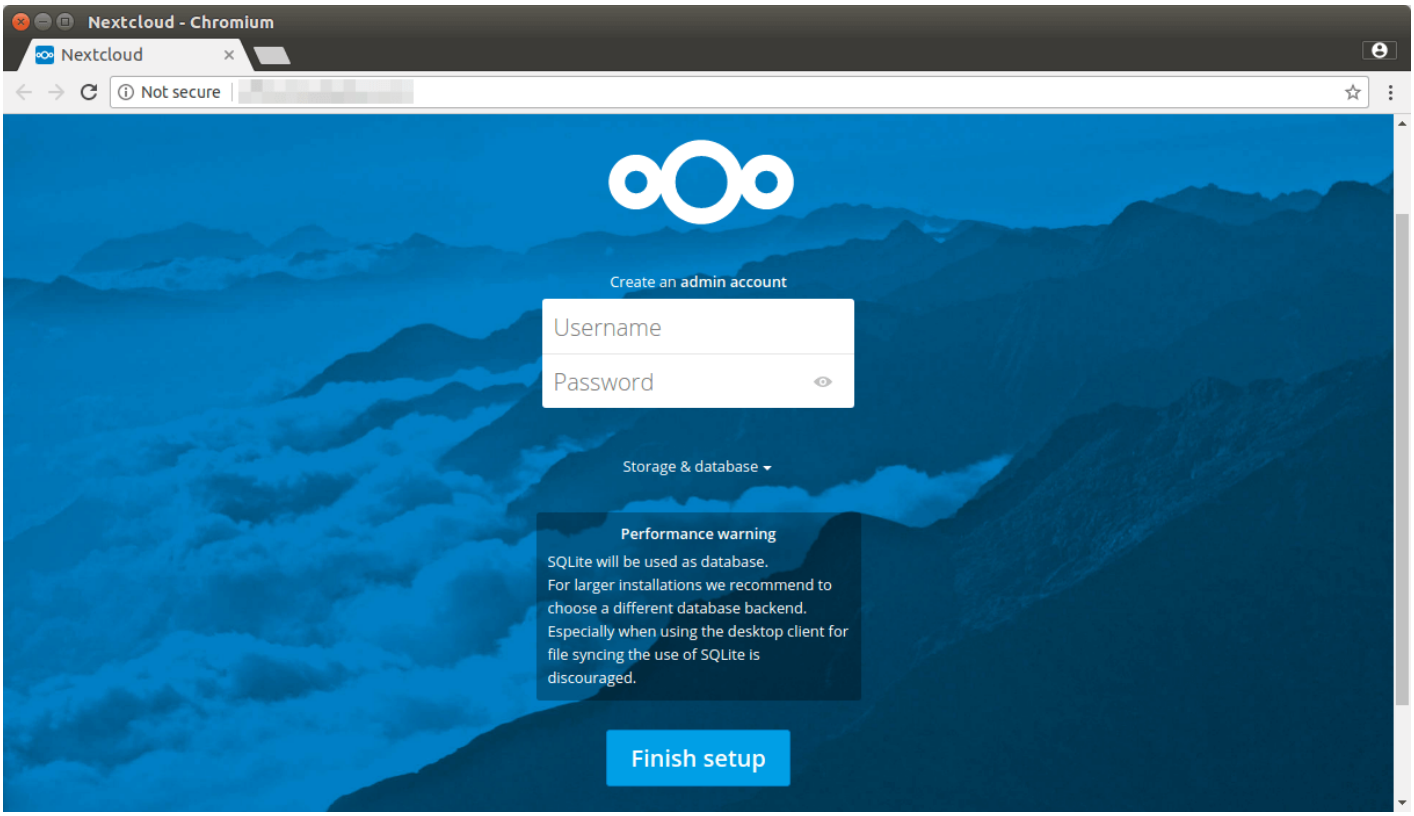
Save and close the file. Now start containers defined in `docker-compose.yml` file.

```
sudo docker-compose up -d
```

The above command will create the **onlyoffice network** and start three containers: **NextCloud**, **OnlyOffice document server** and **Nginx**, as can be seen by issuing the following commands:

```
sudo docker network ls
sudo docker ps
```

Now point your web browser to cloud.example.com and you will be greeted by NextCloud install wizard. Before we enter anything in the wizard, let's enable HTTPS with Let's Encrypt.



Enabling HTTPS With Let’s Encrypt

Now install Let’s Encrypt client (certbot) on your server. The following instructions is for Ubuntu.

```
sudo apt install software-properties-common

sudo add-apt-repository ppa:certbot/certbot

sudo apt update

sudo apt install certbot
```

Since Nginx is running in a Docker container, we won’t be able to use the Nginx plugin to obtain and install SSL/TLS certificate. Instead, we can use the webroot plugin to obtain a certificate and then manually configure SSL/TLS. Run the following command to obtain a certificate.

```
sudo certbot certonly --webroot --agree-tos --email your-email-address -d cloud.example.com -w /var/lib/docker/volumes/dockeronlyofficeowncloud_app_data/_data
```

Explanation:

- certonly: Obtain a certificate. Don’t install it.
- –webroot: Use webroot plugin
- –agree-tos: accept Let’s Encrypt terms of service
- –email: Your email address used for account registration and recovery.
- -d: your domain name.

The `-w` flag is followed by the path to web root directory, which is `/var/www/html/` in Nginx container. Its mount point on the host is `/var/lib/docker/volumes/dockeronlyofficeowncloud_app_data/_data`. Certbot can’t access the web root in Nginx container and must use its mount point. As you can see, I have successfully obtained an SSL certificate.

```
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Plugins selected: Authenticator webroot, Installer None
Obtaining a new certificate
Performing the following challenges:
http-01 challenge for office.linuxdashen.com
Using the webroot path /var/lib/docker/volumes/dockeronlyofficeowncloud_app_data/_data for all unmatched domains.
Waiting for verification...
Cleaning up challenges

IMPORTANT NOTES:
 - Congratulations! Your certificate and chain have been saved at:
   /etc/letsencrypt/live/[redacted]/fullchain.pem
   Your key file has been saved at:
   /etc/letsencrypt/live/[redacted]/privkey.pem
   Your cert will expire on 2018-02-13. To obtain a new or tweaked
   version of this certificate in the future, simply run certbot
   again. 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
```

After obtaining the certificate, edit `nginx.conf` file in `docker-onlyoffice-owncloud` directory to configure SSL.

```
nano nginx.conf
```

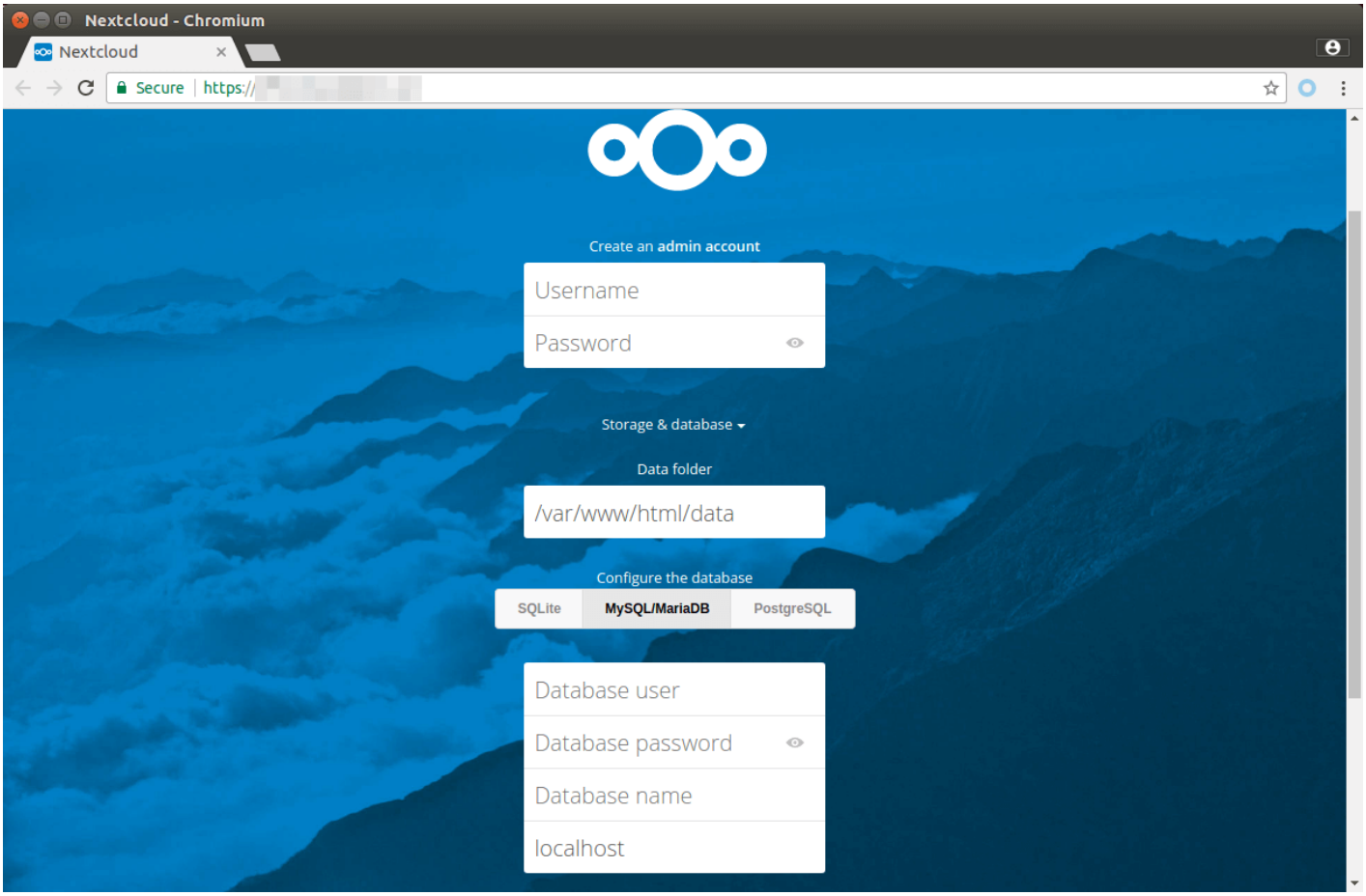
Add the following lines in server block. Remember to replace red text with your actual domain name.

```
    listen 443 ssl http2;

    if ($scheme != "https") {
        return 301 https://$host$request_uri;
    }

    ssl_certificate /etc/letsencrypt/live/cloud.example.com/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/cloud.example.com/privkey.pem;
    ssl_session_cache shared:le_nginx_SSL:1m;
    ssl_session_timeout 1440m;

    ssl_protocols TLSv1.2;
    ssl_prefer_server_ciphers on;

   ssl_ciphers 'ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256';
```

Save and close the file. Then restart Nginx container.

```
sudo docker restart nginx-server
```

Refresh NextCloud install wizard and you shall see a green padlock in browers's address bar.



If there's an error, you can check out nginx-server container's log to find out the error.

```
sudo docker logs nginx-server
```

Using MariaDB Database with NextCloud

If you want to use MariaDB with NextCloud, then you will need to run a MariaDB Docker container. The following command will run the official MariaDB docker container and add it to the **onlyoffice** network. Replace your-pass with your preferred MariaDB root password.

```
sudo docker run --restart=always --net dockeronlyofficeowncloud_onlyoffice --name mariadb-server -e MYSQL_ROOT_PASSWORD=your-pass -d mariadb --log-bin --binlog-format=MIXED
```

Check status.

```
sudo docker ps
```

Once MariaDB container is running, we can access it by issuing the following command.

```
sudo docker exec -it mariadb-server bash
```

Then log into MariaDB server as root.

```
mysql -u root -p
```

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. Notice that we want to create an user which would be able to login from NextCloud container, which has the IP address 172.18.0.3.

```
create user nextclouduser@172.18.0.3 identified by 'your-password';
```

Note: Your NextCloud container may have a different IP address. Run the following command on the host to find it.

```
sudo docker inspect app-server |  grep IPAddress
```

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

```
grant all privileges on nextcloud.* to nextclouduser@172.18.0.3 identified by 'your-password';
```

Flush privileges and exit.

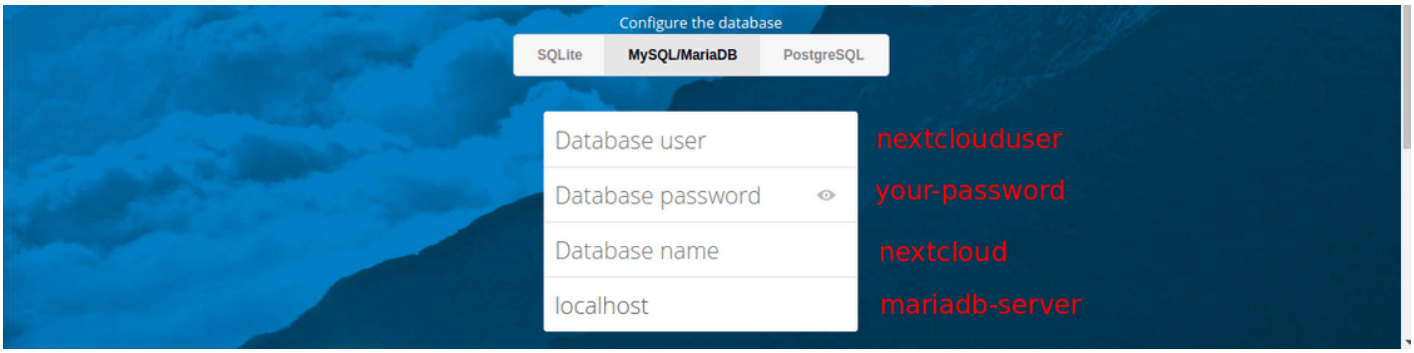
```
flush privileges;


exit;
```

Exit out of MariaDB container.

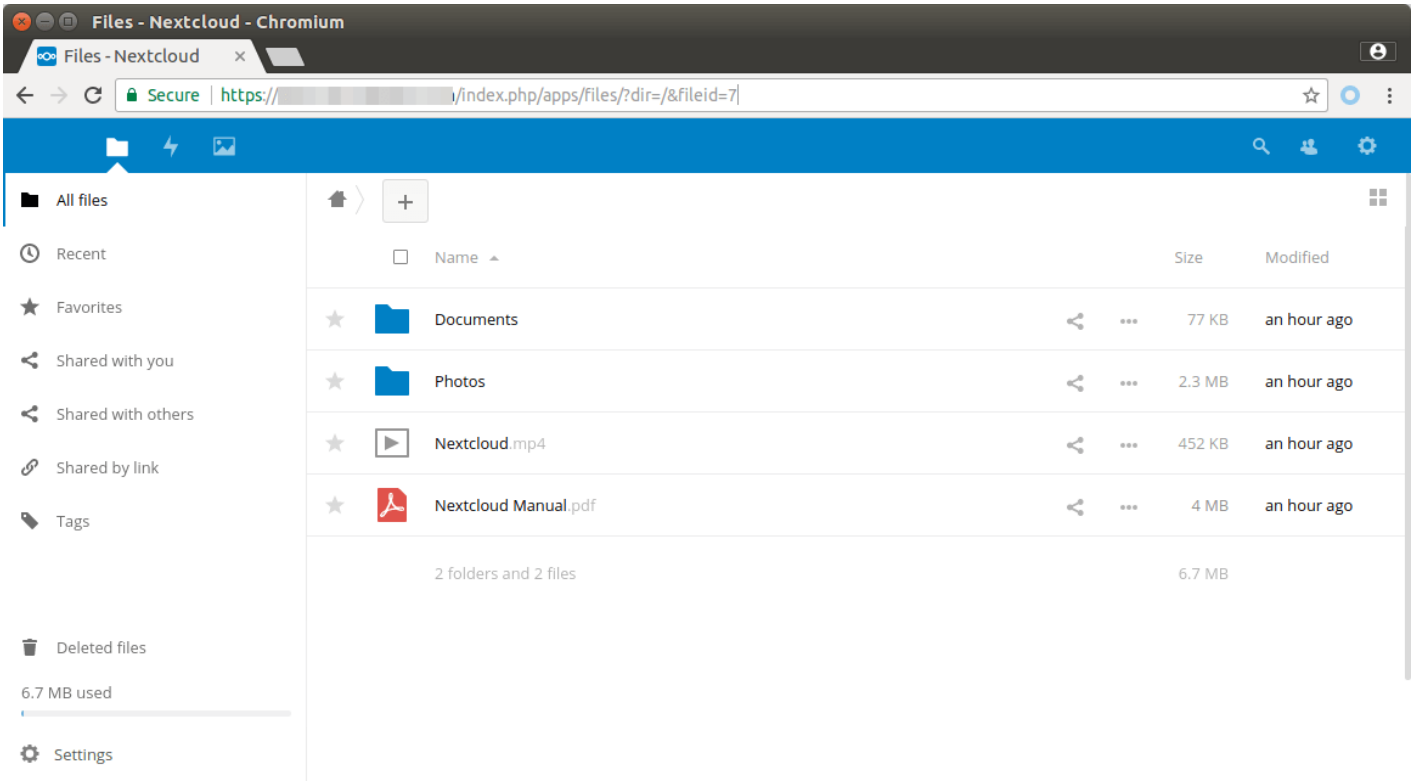
```
exit
```

Now in the NextCloud install wizard, you can create an admin account and enter the details of

MariaDB database server. Note that you need to replace `localhost` with `mariadb-server`, which is the name of MariaDB container. `localhost` here points to NextCloud container. Because NextCloud container and MariaDB container are both in the **onlyoffice** network, NextCloud can resolve `mariadb-server` using an embedded DNS server.



And now NextCloud is successfully installed.

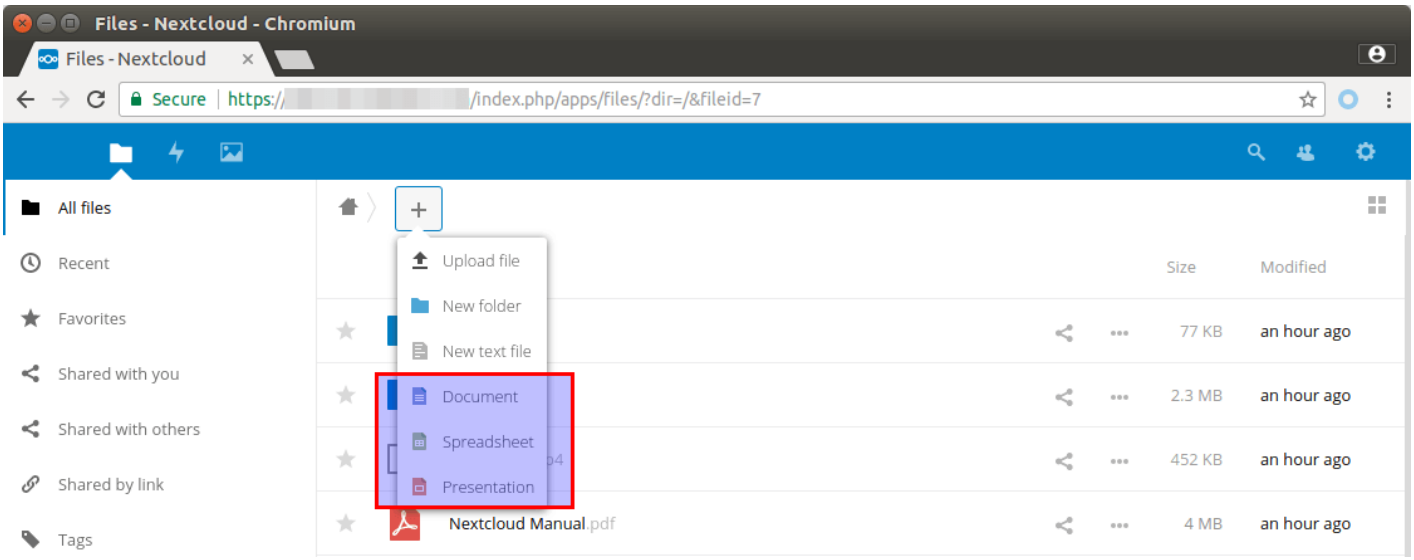


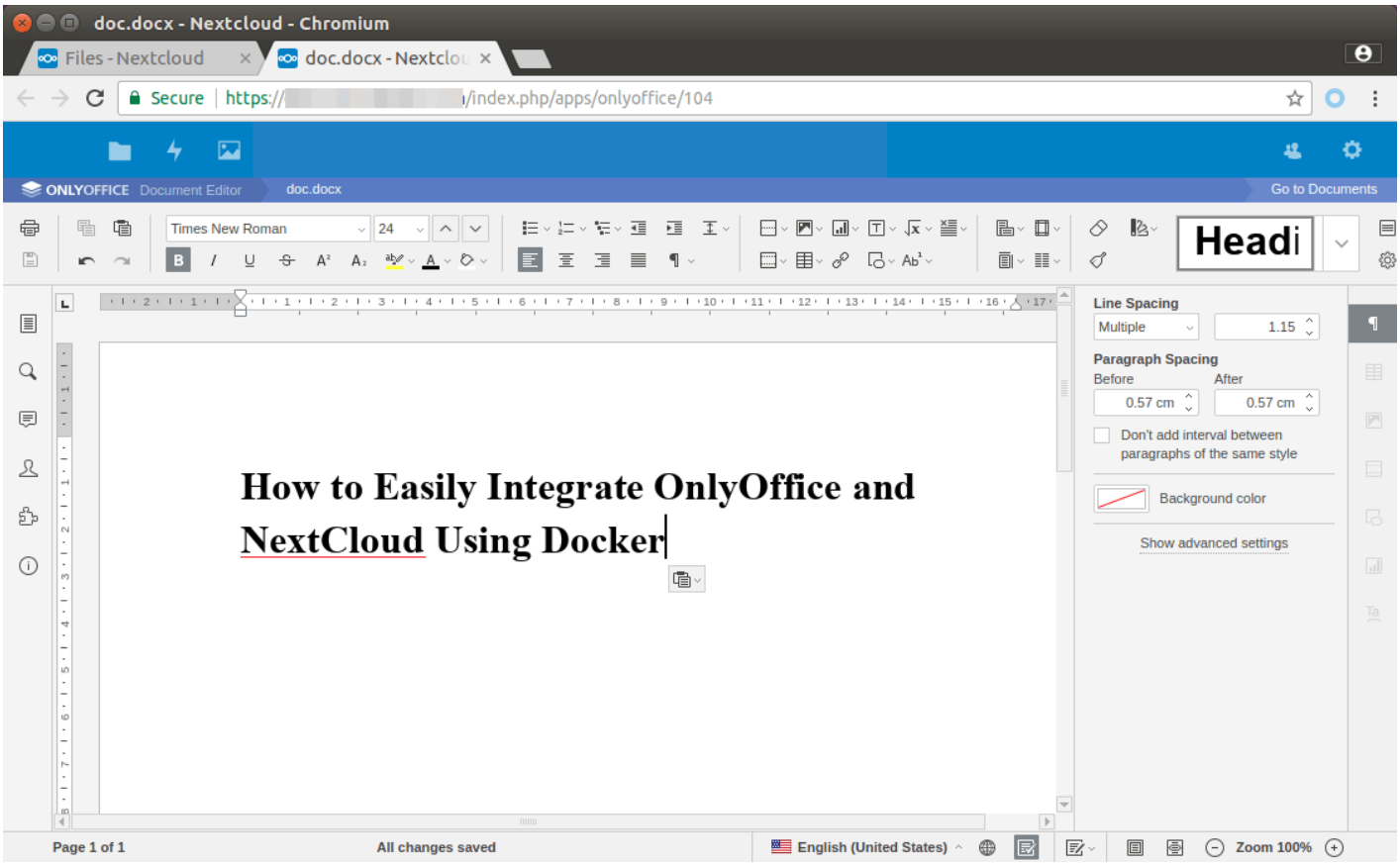
Using a Script to Integrate OnlyOffice and NextCloud

In the `docker-onlyoffice-owncloud` directory, there's a script named `set_configuration.sh`. Run this script to integrate OnlyOffice and NextCloud.

```
sudo bash set_configuration.sh
```

Now you can create and edit Office documents from NextCloud.





Revision #1
Created 11 July 2021 18:54:10 by Admin
Updated 11 July 2021 18:54:47 by Admin