Deployment

This documentation has the following structure:

0. Prerequisites
1. Technical details of the deployment
2. How to use
3. Release

0. Prerequisites

In order to execute the deployment the following technologies need to be installed:

1. Docker - Installation Guide
2. Docker Compose - Installation Guide
3. NodeJS and npm - for windows and linux use *nvm*, for mac use *n*

After installing docker, you need to checkout this repository and open the deployment folder. You can do this using the following commands:

```
git clone git@github.com:amosproj/amos2021ws03-teams-to-nextcloud.git
cd amos2021ws03-teams-to-nextcloud/deployment
```

Important: The deployment is created for deploying on a live instance with a public IP and a registered top level domain.

1. Technical Details

For the deployment we use docker compose with the following services:

db: This service is an instance of MariaDB. It uses a volume for persisting the data and creates a database and an user by using the *db.env* file.

redis: This service runs a redis, which is used by Nextcloud for caching.

proxy and **letsencrypt-companion** are the two services responsible for exposing the applications to the world. The nginx proxy adds a proxy layer between Nextcloud and the internet. The proxy service uses the nginx-proxy image, which runs an nginx reverse proxy setup and automatically registers containers running on the same docker daemon and routes traffic to them. It is designed to serve multiple sites on the same host machine. The advantage in adding this layer is the ability to add a container for Let's Encrypt certificate handling. You can find more information also here.

app: this services uses the Nextcloud image, which is in turn based on the apache web server. It is automatically registered to the reverse proxy using the environment variables mentioned in the next section.

frontend: this service runs our frontend. It is also registered to the reverse proxy.

2. How to use

As already mentioned the deployment setup is created using a docker compose file. Before starting the deployment you need to configure the properties for the containers.

2.1. Configuration

In the *docker-compose.yml* you need to set the following properites:

The **db** container is where the data base is running.

- *MYSQL_ROOT_PASSWORD* - set this to the password for the root user of the database

The **app** container represents the Nextcloud instance.

- *VIRTUAL_HOST* - set this to the domain name you are going to use
- *LETSENCRYPT_HOST* - set this to the domain name you are going to use
- *LETSENCRYPT_EMAIL* - set this to the email, to which the SSL certificate will be registed

The **frontend** container requires the same properties as the **app** container.

In the *db.env* file you need to set the following properties:

- *MYSQL_DATABASE* - the name of the database for the nextcloud app
- *MYSQL_USER* - the name of the database user for connecting to the database
- *MYSQL_PASSWORD* - the password for the user

In order for the containers to persistantly store the data on the host machine, you need to setup the *volumes* bindings in the docker-compose file.

2.2. Building the frontend

In order to build the frontend you need NodeJS version 14ˆ.

Open a terminal in the main directory of the repository.

To install the required pacakges run:

```
npm install
```

To build the frontend run:

```
npm run build
```

When the build is finished the frontend can be found in the *dist* folder. In the *docker-compose.yml* edit the volume in the frontend service, so that it points to the *dist* folder.

2.3. Starting the deployment

Once everything is set up, you can start the deployment using the following command:

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

You should now be able to reach the Nextcloud instance and the frontend using the domains you have set up for them.

3. Release

In order to release a new version of the software enter the live instance where it is running and open the folder of the project in the terminal. Then run the following commands:

Checkout new version:

```
git fetch
git checkout tags/<new version>
```

Build frontend:

```
npm install
npm run build
```

Restart the frontend container:

```
docker ps # to get id of frontend container
docker restart <frontend container id>
```

Enabling CORS on NextCloud

This documentation has the following structure:

0. Prerequisites
1. A brief explanation
2. Dealing with simple requests
3. Dealing with "preflight" requests

0. Prerequisites

A good undersanding of the following guide is required to know exactly what we are doing here: Cross-Origin Resource Sharing (CORS).

1. A brief explanation

Basically the problem lies in loading resources from a different origin than the one we are currently at. For example let's say that our frontend webapp is running on `https://tms2nc.de/` and our NextCloud server on `https://nextcloud.tms2nc.de/`. When our webapp wants to make a request to the NextCloud server, the browser sees that it has a different origin that the one currently loaded (our frontend). When this happens CORS is the mechanism, which is supposed to control the process of resource loading. There are two types of requests: simple and "preflighted" requests:

1. The simple requests are triggered when we want to read a resource from the different origin. If the webserver doesn't explicitly specify that it allows cross-origin requests from our current origin (`https://tms2nc.de/`) the browser will automatically block all of these requests.

2. "Preflighted" requests are done when we want to do an operation to the resource that can eventually change it's state. In that case the browser first sends a "preflighted" request, basically asking the webserver if it will allow the operation to be executed. And if the server allows it, only then the browser sends the real request.

By default the NextCloud webserver is not configured to allow cross-origin requests, so we have to configure it ourselves.

2. Dealing with simple requests

Dealing with simple requests boils down to setting up the right HTTP request/response headers, which the browser expects to see. More details are provided in the guide, listed in the prerequisites. All we need to do to make this work, is configure our apache webserver to set up the right headers. For this add the following lines to the `.htaccess` file in the apache root directory (`/var/www/html`):

```
Header always add Access-Control-Allow-Origin "https://tms2nc.de"
Header always add Access-Control-Allow-Headers "Authorization, Origin, X-Requested-With, Co
Header always add Access-Control-Allow-Methods "GET, HEAD, POST, PUT, OPTIONS, MOVE, DELETE
Header always add Access-Control-Allow-Credentials "true"
```

where `https://tms2nc.de` is the address of our frontend. After that make sure to activate the apache module headers:

```
a2enmod headers
```

Source: Header set Access-Control-Allow-Origin in .htaccess doesn't work

3. Dealing with "preflight" requests

For the "preflight" requests, we just have to configure our webserver to respond with a 204 to the HTTP OPTIONS request, which checks if the webserver will allow the action. We can achieve this by using the rewrite in the same `.htaccess` file and adding the following lines:

```
RewriteEngine On
RewriteCond %{REQUEST_METHOD} OPTIONS
```

```
RewriteRule ^(.*)$ $1 [L,R=204]
```

Source: Return empty response from Apache

SSO Config

In order to facilitate the authentication to our Nextcloud instance we have setup a Single Sign On authentication using the Microsoft Azure Active Directory. Nextcloud itself provides the option to integrate Active Directory as Single Sign On.

Configuration Guide

In order to properly configure the SSO we followed this guide.

In the guide there is at the time of writing one mistake. In **Step 2** when setting the *Identifier (Entity ID)* you should set it to the following URL:

```
https://nextcloud.yourdomain.com/apps/user_saml/saml/metadata
```

Important: The SSO configuration requires a top level domain for your Nextcloud instance.

If you plan on using SSL with your Nextcloud instance then you should also set the following property in your *config/config.php* which is located in the Nextcloud container:

```
'overwriteprotocol' => 'https'
```