LocalCosmos Private Server Documentation

Thomas Uher

Aug 13, 2021

Contents:

Contents:

CHAPTER 1

Installation using docker

Duration: 15-30 minutes.

This tutorial has been tested on a fresh minimal install of Ubuntu server 20.04.

1.1 1. Install and Configure nginx

This is only required for a production environment. If you test on your local machine, you can skip this step.

```
sudo apt-get install nginx
```

Create the nginx conf file for your app. Replace `<my-project.org>` with the domain of your project.

```
sudo touch /etc/nginx/sites-available/<my-project.org>.conf
```

For example, if you plan to host your server at `treesofbavaria.org`, use the following command:

```
sudo touch /etc/nginx/sites-available/treesofbavaria.org.conf
```

Put the following in your just created `.conf` file. Again, replace `<my-project.org>` with the domain of your project

```
upstream lcprivate_docker {
        server localhost:9202;
}

server {
        listen 80;
        server_name <my-project.org> www.<my-project.org>;
        return 301 https://<my-project.org>/$request_uri;
}

server {
```

(continues on next page)

(continued from previous page)

```
        listen 443;
        server_name <my-project.org> www.<my_project.org>;

        client_max_body_size 50M;

        location / {
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_set_header X-Real-IP $remote_addr;
                proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
                proxy_set_header X-Forwarded-Proto $scheme;
                proxy_cache_bypass $http_upgrade;
                proxy_pass http://lcprivate_docker;
        }

}
```

1.2 2. Install docker

Install docker and the docker-compose-utility

```
sudo apt-get install docker.io docker-compose
```

Start docker

```
sudo systemctl start docker
```

make docker start on boot

```
sudo systemctl enable docker
```

1.3 3. Get the Local Cosmos Private Server docker image

All docker commands have to be run as the superuser.

```
sudo docker pull docker.sisol-systems.com/localcosmos-private-server
```

1.4 4. Configuration with docker-compose.yml

On your server, create a folder for your project.

```
sudo mkdir /opt/<my-project-name>
```

Create the file docker-compose.yml

```
cd /opt/<my-project-name>
sudo touch docker-compose.yml
```

Put the following content into `docker-compose.yml`. Replace `<my-project-name>` with the name of your project. Also Replace `<db_username>` and `<db_password>`. This will **set** your database credentials, so do not share these values openly.

Also replace `<.myproject.org>` with the domain you run your Localcosmos Private Server on. Do not forget the leading `.`. Finally, replace `<APP_UID>` with app_uid of your App. You find your app_uid in the App Kit on localcosmos.org. If you just want to run the Demo App on localhost, use `treesofbavaria` as the app_uid. You cannot run the Demo App on something else than localhost.

```
version: '3.3'

services:
  lc-private:
        container_name: '<my-project-name>'
        image: 'docker.sisol-systems.com/localcosmos-private-server'
        restart: always
        build: .
        volumes:
          - type: volume
            source: www
            target: /var/www/localcosmos/
          - type: volume
                source: database_config
                target: /etc/postgresql/
          - type: volume
            source: database_log
            target: /var/log/postgresql/
          - type: volume
            source: database_data
            target: /var/lib/postgresql/
        ports:
          - 9202:8001
        environment:
          - DATABASE_NAME=localcosmos
          - DB_USER=<db_username>
          - DB_PASSWORD=<db_password>
          - ALLOWED_HOSTS=localhost|<.myproject.org>
          - APP_UID=<APP_UID>
          - SERVE_APP_URL=/

volumes:
  www:
  database_config:
  database_log:
  database_data:
```

Optionally, you can add email settings to the environment. This enables django to send email to you if an error occurs server-side.

```
- EMAIL_HOST=<email_host>
- EMAIL_PORT=<email_port>
- EMAIL_HOST_USER=<email_host_user>
- EMAIL_HOST_PASSWORD=<email_host_password>
- EMAIL_USE_TLS=1
```

Replace `<email_host>`, `<email_port>`, `<email_host_user>`, `<email_host_password>` with your parameters and set `EMAIL_USE_TLS` to 1 or 0.

1.5 5. Run the docker container

```
cd /opt/<my-project-name>
sudo docker-compose up -d
```

1.6 6. Enable nginx conf and reload nginx conf

First, add your nginx conf to `sites-enabled`. Replace `<my-project.org>` with the name of you project.

```
sudo ln -s /etc/nginx/sites-available/<my-project.org>.conf /etc/nginx/sites-
↪enabled/
```

Now, reload your nginx conf with the following command.

```
sudo service nginx reload
```

After Installation, visit `localhost:9202/server/control-panel/` or `<myproject.org>/server/control-panel/` and follow the on-screen instructions.

You now have your Local Cosmos Private Server up and running.

If you are on a local machine and want to test the Demo App, proceed to **Installing the Demo App**.

CHAPTER 2

Installation without docker

Duration: 30-45 minutes.

This tutorial covers setting up a running LocalCosmos Private Server as a development server. This tutorial is intended for people not familiar with django. For more information about django visit https://www.djangoproject.com/ .

2.1 1. Prerequisites

Before you can install django and the localcosmos_server package, you have to install the requirements below. All code examples are for Debian/Ubuntu based systems.

2.2 1.1 Required server components

- nginx or apache2 web server
- python3 and python3-dev
- virtualenv (optional, recommended)
- PostgreSQL 10.x with PostGIS 2.x
- kmeans PostgreSQL extension: https://github.com/umitanuki/kmeans-postgresql

On Debian/ubuntu, you can install the required packages as follows:

```
sudo apt-get install nginx
sudo apt-get install python3 python3-dev
sudo apt-get install virtualenv
sudo apt-get install postgresql-10 postgresql-10-postgis-2.4 libpq-dev␣
↪postgresql-server-dev-10
```

2.3 1.2 Create Postgres database

If not yet done, create a postgres database. If your desired database name is `localcomsos`, you can create the database as follows:

```
sudo -u postgres -i
psql
# create new database by the name localcosmos
create database localcosmos;
```

You also have to create a database user which has the right to alter the just created `localcosmos` database. In this example the user is named `lcuser`, but you can use any other name. You have to replace `<lcpassword>` with your desired password.

```
create user lcuser;
alter role lcuser SUPERUSER;
alter user lcuser PASSWORD '<lcpassword>';
\q
```

2.4 1.3 Install the kmeans PostgreSQL extension

2.4.1 1.3.1 Switch back to your server user

This step only applies if you follow this tutorial step by step. Switch back from the postgres user to the user you use on your server. Replace `<serveruser>` with the username you use on your server.

```
su <serveruser>
```

2.4.2 1.3.2 Download kmeans

Download and unzip https://github.com/umitanuki/kmeans-postgresql on your server. In this example, kmeans is downloaded into `/opt/kmeans`, but you can use any other folder.

```
sudo mkdir /opt/kmeans
cd /opt/kmeans
# replace <serveruser> with your username
sudo chown <serveruser>:<serveruser> /opt/kmeans
wget https://github.com/umitanuki/kmeans-postgresql/archive/master.zip
# install unzip
sudo apt-get install unzip
unzip master.zip
```

2.4.3 1.3.3 Activate kmeans extension

In this example, the database which we want to install the kmeans extension for, is named `localcosmos`. Replace `localcosmos` if your database has a different name.

```
cd /opt/kmeans/kmeans-postgresql-master
# if not yet done, install build requirements
sudo apt-get install make gcc
```

(continues on next page)

(continued from previous page)

```
# make and make install kmeans
make
sudo make install
# switch to the postgres user
sudo -u postgres -i
# activate the kmeans extension for the database localcosmos, replace the db␣
↪name if necessary
psql -f /usr/share/postgresql/10/extension/kmeans.sql -d localcosmos
exit
```

2.5 2. Create a new django project

Create a project folder

Create a folder on your disk where your Local Cosmos Private Server can live. eg: `/opt/localcosmos`. Make sure you have permissions to write this folder. In the example the server user is `<serveruser>` - replace this with the username you use.

Create a python3 virtual environment

```
sudo mkdir /opt/localcosmos
sudo chown <serveruser>:<serveruser> /opt/localcosmos
cd /opt/localcosmos
virtualenv -p python3 venv
```

Activate the virtual environment

```
source venv/bin/activate
```

Install django and localcosmos_server

```
pip install django==2.2.*
pip install localcosmos_server
```

This will install django, localcosmos_server and its requirements in your created and activated virtualenv.

Create a new django project

In `/opt/localcosmos` execute the following:

```
django-admin startproject localcosmos_private
```

This will automatically create the folder `/opt/localcosmos/localcosmos_private`, which contains your newly created django project.

2.6 3. Configure your django project

2.6.1 3.1 settings.py

You now have to adjust the contents of the file `settings.py` located in `/opt/localcosmos/localcosmos_private/localcosmos_private/` to set up your LocalCosmos Private Server.

Replace `INSTALLED_APPS` with the following:

```
INSTALLED_APPS = [

        # django defaults
        'django.contrib.admin',
        'django.contrib.auth',
        'django.contrib.contenttypes',
        'django.contrib.sessions',
        'django.contrib.messages',
        'django.contrib.staticfiles',

        # localcosmos
        'django.contrib.sites',

        'localcosmos_server',
        'localcosmos_server.app_admin',
        'localcosmos_server.server_control_panel',
        'localcosmos_server.datasets',
        'localcosmos_server.online_content',

        'django_road',
        'anycluster',
        'content_licencing',

        'rules',
        'el_pagination',
        'django_countries',
        'corsheaders',
        'rest_framework',
        'rest_framework.authtoken',

        'octicons',
        'imagekit',

        'django.forms',
]
```

Replace the `MIDDLEWARE` setting with the following

```
MIDDLEWARE = [
        'localcosmos_server.middleware.LocalCosmosServerSetupMiddleware', #␣
↪has to be on top
        'django.middleware.security.SecurityMiddleware',
        'django.contrib.sessions.middleware.SessionMiddleware',
        'django.middleware.locale.LocaleMiddleware',
        'corsheaders.middleware.CorsMiddleware',
        'django.middleware.common.CommonMiddleware',
        'django.middleware.csrf.CsrfViewMiddleware',
        'django.contrib.auth.middleware.AuthenticationMiddleware',
        'django.contrib.messages.middleware.MessageMiddleware',
        'django.middleware.clickjacking.XFrameOptionsMiddleware',
        'localcosmos_server.app_admin.middleware.AppAdminMiddleware',
        'localcosmos_server.server_control_panel.middleware.
↪ServerControlPanelMiddleware',
]
```

Replace the `TEMPLATES` setting with the following

```
TEMPLATES = [
        {
                'BACKEND': 'django.template.backends.django.DjangoTemplates',
                'DIRS': [],
                'APP_DIRS': False,
                'OPTIONS': {
                    'context_processors': [
                        'django.template.context_processors.debug',
                        'django.template.context_processors.request',
                        'django.contrib.auth.context_processors.auth',
                        'django.contrib.messages.context_processors.messages
↪',
                        'localcosmos_server.context_processors.localcosmos_
↪server',
                    ],
                    'loaders' : [
                        'django.template.loaders.filesystem.Loader',
                        'django.template.loaders.app_directories.Loader',
                    ]
                }
        },
]
```

Set up the database. Replace the `DATABASE``setting with the setting below. Make sure you replace ``<lcpassword>` with the correct password. If you did not follow the **Preparing your webserver** tutorial, you will also have to adjust the `NAME` and `USER` paramters according to your postgresql database name and your postgresql username.

```
DATABASES = {
        'default': {
                'ENGINE': 'django.contrib.gis.db.backends.postgis',
                'NAME': 'localcosmos',
                'USER' : 'lcuser',
                'PASSWORD' : '<lcpassword>',
                'HOST' : 'localhost',
        }
}
```

Replace `ALLOWED_HOSTS` with the following.

```
ALLOWED_HOSTS = ['localhost']
```

Replace or add `STATIC` and `MEDIA` paths

```
STATIC_URL = '/static/'
STATIC_ROOT = '/var/www/localcosmos/static/'

MEDIA_ROOT = '/var/www/localcosmos/media/'
MEDIA_URL = '/media/'
```

Inlude localcosmos_server settings in your `settings.py` file. This automatically covers anycluster, django_road and cors settings. Insert these lines at the bottom of settings.py

```
from localcosmos_server.settings import *

# location where apps are installed
```

(continues on next page)

(continued from previous page)

```
# your apps index.html will be in LOCALCOSMOS_APPS_ROOT/{APP_UID}/www/index.
↪html
LOCALCOSMOS_APPS_ROOT = '/var/www/localcosmos/apps/'
```

2.6.2 3.2 urls.py

The file `urls.py` located in `/opt/localcosmos/localcosmos_private/localcosmos_private/` also needs configuration. You `urls.py` should look like this:

```
from django.conf import settings
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
        path('admin/', admin.site.urls),
        path('', include('localcosmos_server.urls')),
]
```

As long as you run the django development server, add the following at the bottom of `urls.py`.

```
# remove these lines after development
if settings.DEBUG:
        from django.conf.urls.static import static
        urlpatterns += static(settings.MEDIA_URL, document_root=settings.
↪MEDIA_ROOT)
```

Make sure your remove these lines before deploying django. For better security, static and media files should be served directly by nginx in a production environment.

That's it for the django configuration.

2.7 4. Migrate database

In your django project directory, `/opt/localcosmos/localcosmos_private/`, run

```
python manage.py migrate
```

to migrate the database.

2.7.1 5. Create localcosmos www folder

We need the folder `/var/www/localcosmos` and django has to be able to write into it. Replace `<server_user>` with the user you use on your computer.

```
# if the folder does not exist yet
sudo mkdir /var/www/localcosmos
# run this command in any case
sudo chown <server_user>:www-data /var/www/localcosmos
```

2.8 6. Run the development server

In your django project directory, `/opt/localcosmos/localcosmos_private/`, run the following command to start the development server.

```
python manage.py runserver 0.0.0.0:8080
```

Now open a browser and navigate to `http://localhost:8080` . Follow the instructions to complete the setup.

Also check if the API works. Browse to `http://localhost:8080/api/` .

After you completed the setup, the Server Control Panel ist available at `http://localhost:8080/server/control-panel/`.

2.9 7. Re-running the development server

If you want to start the development server after rebooting, you have to activate the virtual environment first.

```
cd /opt/localcosmos
source venv/bin/activate
cd localcosmos_private
python manage.py runserver 0.0.0.0:8080
```

CHAPTER 3

Deploy your Server (without docker)

Once you have successfully tested your LocalCosmos Private Server in development mode, you can deploy your server and make it accessible for the public. This tutorial covers deployment of Local Cosmos Private Server using `nginx`, `uwsgi` and Ubuntu 18.04.

3.1 1. Make your django application ready

3.1.1 1.1 Update settings.py

Add your Domain name, in this example `localcosmos-private.org`, to `ALLOWED_HOSTS` in your `settings.py` file. Also set `DEBUG` to `False`.

```
ALLOWED_HOSTS = ['localcosmos-private.org']

DEBUG = False
```

3.1.2 1.2 Check the application

Run the development server once to check if there are errors

```
cd /opt/localcosmos
source venv/bin/activate
python manage.py runserver 0.0.0.0:8080
```

If you installed the Demo App (Trees Of Bavaria), you have to remove it before continuing. The Demo App is not configured to run with a deployed Local Cosmos Private Server. Go to `http://localhost:8080/server/control-panel/` and remove the App.

If there are no errors, stop the development server and continue.

3.1.3 1.3 Clean urls.py

Remove the development lines from `urls.py`

```
# remove these lines after development
#if settings.DEBUG:
#       from django.conf.urls.static import static
#       urlpatterns += static(settings.MEDIA_URL, document_root=settings.
↪MEDIA_ROOT)
```

3.2 2. uwsgi

3.2.1 2.1 Install uwsgi

If still active, deactivate your virtual environment. We have to install uwsgi system-wide and not inside the virtual environment.

```
deactivate
```

Install `uwsgi` using `pip`:

```
sudo apt-get install python3-pip
sudo -H pip3 install uwsgi
```

3.2.2 2.2 Create uwsgi.ini

First, create a `uwsgi` folder where all the uwsgi stuff will go

```
cd /opt/localcosmos
mkdir uwsgi
```

Create the file `localcosmos_private_uwsgi.ini` in `/opt/localcosmos/uwsgi/`

```
cd /opt/localcosmos/uwsgi/
touch localcosmos_private_uwsgi.ini
```

and put the following in it:

```
# localcosmos_private_uwsgi.ini file
[uwsgi]

# Django-related settings

# the base directory (full path)
chdir           = /opt/localcosmos/localcosmos_private

# Django's wsgi file
module          = localcosmos_private.wsgi:application

# the virtualenv (full path)
home            = /opt/localcosmos/venv

# process-related settings
```

(continues on next page)

(continued from previous page)

```
# master
master          = true
# maximum number of worker processes
processes       = 10

# the socket (use the full path to be safe)
socket          = /opt/localcosmos/uwsgi/socket/localcosmos-private.sock

# ... with appropriate permissions - may be needed
chmod-socket    = 666

# clear environment on exit
vacuum          = true

daemonize               = /var/log/uwsgi/localcosmos-private.log
```

3.2.3 2.3 Prepare the socket

The socket `localcosmos-private.sock` will automatically be created. Therefore, we need a folder `www-data` can write into.

```
cd /opt/localcosmos/uwsgi
mkdir socket
# set permissions
sudo chgrp www-data /opt/localcosmos/uwsgi/socket
```

3.2.4 2.4 Get uwsgi_params

```
cd /opt/localcosmos/uwsgi
wget https://raw.githubusercontent.com/nginx/nginx/master/conf/uwsgi_params
```

3.2.5 2.5 Logging

```
sudo mkdir /var/log/uwsgi
sudo chown <server-user>:www-data /var/log/uwsgi
```

3.3 3. Configure nginx

3.3.1 3.1 Create nginx conf file

First you have to create an nginx configuration file. Best practice is to name the file after the domain. For this tutorial we assume the domain is `localcosmos-private.org`, so we create the file `localcosmos-private.org.conf`. Adjust the filename according to the domain name you will use for your Local Cosmos Private Server.

```
cd /etc/nginx/sites-available
sudo touch localcosmos-private.org.conf
```

Now put the following code into this file.

```
# localcosmos-private.org.conf

# the upstream component nginx needs to connect to
upstream django {
        # according to recommendations, we use a file socket
        server unix:///opt/localcosmos/uwsgi/socket/localcosmos-private.sock;
}

# configuration of the server
server {

        # the port your site will be served on
        listen      80;

        # the domain name it will serve for
        server_name localcosmos-private.org;

        charset     utf-8;

        # max upload size
        client_max_body_size 75M;   # adjust to taste

        # serve django media files according to settings.py
        location /media  {
                alias /var/www/localcosmos/media;
        }

        # serve django static files according to settings.py
        location /static {
                alias /var/www/localcosmos/static;
        }

        # pass /server to django
        location /server {
                uwsgi_pass  django;
                include     /opt/localcosmos/uwsgi/uwsgi_params;
        }

        # pass /app-admin to django
        location /app-admin {
                uwsgi_pass  django;
                include     /opt/localcosmos/uwsgi/uwsgi_params;
        }

        # pass /api to django
        location /api {
                uwsgi_pass  django;
                include     /opt/localcosmos/uwsgi/uwsgi_params;
        }

        # (optional) the app you are going to install at a later point

        location / {
                alias /var/www/localcosmos/apps/<APP_UID>/www/;
                try_files $uri $uri/index.html;
        }
}
```

3.3.2 3.2 Make your site available

Create the symlink to `localcosmos-private.org.conf` in `/etc/nginx/sites-enabled/`

```
sudo ln -s /etc/nginx/sites-available/localcosmos-private.org.conf /etc/
↪nginx/sites-enabled/
```

3.3.3 3.3 Collect static files

Create the folder `localcosmos` in `/var/www` with the correct permissions, if it does not exist yet. Replace `<server_user>` with your username on your server.

```
cd /var/www
sudo mkdir localcosmos
sudo chown <serveruser>:www-data localcosmos

# if not yet active, activate the virtual environment
cd /opt/localcosmos
source venv/bin/activate

# collect static files
cd localcosmos_private
python manage.py collectstatic

# deactivate virtualenv
deactivate
```

3.3.4 3.4 Reload nginx

```
sudo service nginx reload
```

Test your uwsgi setup using this command.

```
/usr/local/bin/uwsgi --ini /opt/localcosmos/uwsgi/localcosmos_private_uwsgi.
↪ini --uid www-data --gid www-data
```

Now open `http://YOUR_DOMAIN.org/server/control-panel/` in a browser and check if it works.

On some installations you have to remove `default` from `sites-enabled` (NOT `sites-available` !!)

```
cd /etc/nginx/sites-enabled
sudo rm default
sudo service nginx reload
```

3.3.5 3.5 Make uwsgi startup when the system boots

Create the file `/etc/rc.local` if it does not exist yet.

```
sudo touch /etc/rc.local
sudo chmod +x /etc/rc.local
```

Put the following in it:

```
#!/bin/sh -e
# rc.local

/usr/local/bin/uwsgi --ini /opt/localcosmos/uwsgi/localcosmos_private_uwsgi.
↪ini --uid www-data --gid www-data

exit 0
```

That's it! You now have a fully working Local Cosmos Private Server!

3.4 4. Troubleshooting

1. Check `/var/log/nginx/error.log`
2. Check `/var/log/uwsgi/localcosmos-private.log`
3. Read https://uwsgi-docs.readthedocs.io/en/latest/tutorials/Django_and_nginx.html

CHAPTER 4

Installing the Demo App

The Demo App should **only be installed on a local development server for testing**.

4.1 1. Installation on a DOCKER LC Private Server

4.2 1.1 Download the Demo App

The Demo App is a .zip file named `TreesOfBavaria.zip`. You can dowload it here .

Note: The Demo App covers the Webapp version of Trees Of Bavaria. If you build your own App on localcosmos.org, you will receive Android and iOS versions alongside the Webapp version.

4.3 1.2 Install the Demo App

Open `http://localhost:9202/server/control-panel/` and click on `Install App`.

1. Select the zipfile `TreesOfBavaria.zip` which you just downloaded.
2. Enter `http://localhost:9202/` (or the URL according to your webserver configuration) as the URL of this App.
3. Click the Install button

Once the installation is complete, visit `http://localhost:9202/` to open the Webapp. To test the API, try logging in with the Superuser Account credentials you created in the server setup tutorial, or try to report an observation.

4.4 2. Installation on a non-docker LC Private Server

This tutorial covers nginx examples. If you plan to use apache2, you have to translate the examples into apache2 syntax.

We will configure the following setup:

URL	Function	Served by
http://localhost:8080/server/control-panel/	Server Control Panel	django development server
http://localhost:8080/api/	API	django development server
http://localhost:8080/app-admin/treesofbavaria/	App Admin (Trees Of Bavaria)	django development server
http://localhost/	Trees Of Bavaria Webapp	nginx

The App Admin for the Demo App, Trees Of Bavaria, will be available after the App has been installed. This Demo App expects the Local Cosmos Private Server api running at `http://localhost:8080/api/`. Otherwise the Demo App will not work.

Start your Local Cosmos Development Server:

```
cd /opt/localcosmos
# activate virtual environment if not yet activated
source venv/bin/activate
# start the server
cd /opt/localcosmos/localcosmos_private
python manage.py runserver 0.0.0.0:8080
```

4.5 2.1 Download the Demo App

The Demo App is a .zip file named `TreesOfBavaria.zip`. You can dowload it here .

Note: The Demo App covers the Webapp version of Trees Of Bavaria. If you build your own App on localcosmos.org, you will receive Android and iOS versions alongside the Webapp version.

4.6 2.2 Configure nginx to serve your Webapp

Local Cosmos Webapps are served by nginx or apache2, not by django.

Later, you will install your webapp using the **Server Control Panel** of your Local Cosmos Private Server. Your webapps will automatically be stored in a subfolder of the folder defined by `LOCALCOSMOS_APPS_ROOT` in your `settings.py` file. The `UID` of your app will be the name of this subfolder. The `UID` of the Demo App is `treesofbavaria` and `LOCALCOSMOS_APPS_ROOT` is set to `/var/www/localcosmos/apps/`. So this app will be installed in `/var/www/localcosmos/apps/treesofbavaria/`.

For this test we serve the App at the root directory `/`. Open the configuration file named `default` living in `/etc/nginx/sites-available/` and modify `location /` as follows:

```
location / {
        # /LOCALCOSMOS_APPS_ROOT/<APP_UID>/www/
        alias /var/www/localcosmos/apps/treesofbavaria/www/;
        try_files $uri $uri/index.html;
}
```

Do not forget to reload the nginx conf

```
sudo service nginx reload
```

With this configuration, the Demo App will later be available the URL `http://localhost/` (after it has been installed).

It is very important to **remember the url** which your webapp will be served at **by nginx** because you will have to enter this url in the **Server Control Panel** when installing an app.

4.7 2.3 Install the Demo App

Open `http://localhost:8080/server/control-panel/` and click on `Install App`.

1. Select the zipfile `TreesOfBavaria.zip` which you just downloaded.
2. Enter `http://localhost/` (or the URL according to your webserver configuration) as the URL of this App.
3. Click the Install button

Once the installation is complete, visit `http://localhost/` to open the Webapp. To test the API, try logging in with the Superuser Account credentials you created in the server tutorial, or try to report an observation.