Socialhome Documentation

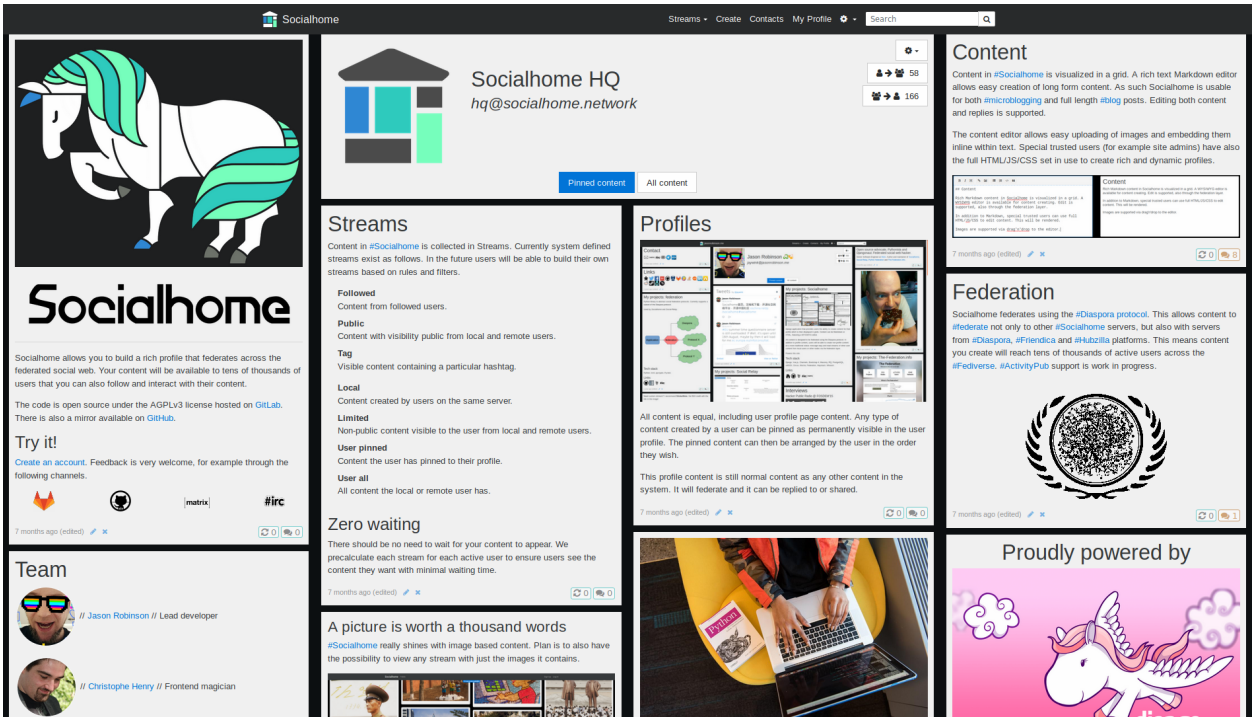
Release 0.12.0

Jason Robinson

Dec 11, 2020

Contents

Welcome to the documentation of Socialhome!



CHAPTER 1

Description

Socialhome is best described as a federated personal profile with social networking functionality. Users can create rich content using Markdown and even HTML/JS/CSS (if set as trusted user). All content can be pinned to the user profile and all content will federate to contacts in the federated social web. Currently federation happens using the ActivityPub and Diaspora protocols.

Please check the official site for more information about features. Naturally, the official site is a Socialhome profile itself.

Official site: https://socialhome.network

CHAPTER 2

Joining

Yes! The official server is open for registrations. Sign up and play around!

Please see the *Community* pages for how to interact with the community.

CHAPTER 3

Installation

Please see the *Installation* pages.

CHAPTER 4

Running an instance

Please see the *Running an instance* pages.

CHAPTER 5

Development

Please see the *Development* pages.

CHAPTER 6

Source code

Socialhome is fully open source, licenced under the AGPLv3 license.

Check the code on GitHub.

CHAPTER 7

Table of contents

7.1 Installation

7.1.1 System requirements

Socialhome requires a Linux server with root access. It is not possible to install Socialhome on a shared server.

Resources

Socialhome isn't particularly heavy, though obviously that depends on the amount of users and connections to remote nodes. In default set up, Socialhome will be running the following:

- uWSGI (~200mb)
- Daphne (~60mb)
- Circus (~25mb)
- 5x RQ workers (~75mb each == 375mb)
- RQ scheduler (~75mb)

Memory values are taken from a running production instance. This gives a total of 735mb of RAM used by the application. Additionally, you need to allocate for PostgreSQL and Redis. A few gigabytes of available memory should easily be enough to run a node with medium activity.

Disk space will mostly be required for content and image uploads. As an example of database size, 100K content objects takes approx 230mb in the database. Image upload disk requirements depends entirely on the sizes of the uploads.

7.1.2 Install guides

These instructions are for a production installation. For development installation instructions, see the *Development* pages.

If you have issues following these instructions, please contact us via *Community*.

The recommended installation method is using the official Docker images. Other guides will possibly be out of date and will likely be removed from the official documentation at some point.

Available guides:

- installation-docker
- *Ubuntu via Ansible*
- installation-ubuntu
- *Other Linuxes or newer Ubuntu using SystemD*

7.1.3 Docker

The Docker images are relatively new, but have had some real production testing.

Currently the main application of Socialhome is packaged as one image. Inside the image there is a single Circusd process which runs 1) gunicorn, 2) django daphne, 3) a channels worker and 4) X amount of RQ worker processes. It's possible we will extract various components into separate Docker images in the future.

In addition to the main image, the following are needed:

- Redis
- PostgreSQL
- Nginx or similar to serve media and/or provide SSL

There is an example docker-compose.yml that shows how things can be set up. Socialhome environment variables can be used to point to an external Redis or PostgreSQL if need be. See *Configuration*.

Find the Socialhome images in the Docker registry.

7.1.4 Ubuntu via Ansible

See this Ansible role.

7.1.5 Ubuntu (14.04)

This guide is very opinionated and experienced sysadmins will most likely want to do things differently. This guide will give you a Socialhome production install on uWSGI using an Apache2 web server.

Supported versions

This guide is written for **Ubuntu 14.04** (with Upstart). For a SystemD config file, see *Other Linuxes or newer Ubuntu using SystemD*.

Steps

Fix locales

Ubuntu 14.04 has a problem with locales which could bring problems when installing PostgreSQL. If you have already installed PostgreSQL, you can probably skip this step.

Check these two commands:

```
echo $LANGUAGE
echo $LC_ALL
```

if both of them come out empty, edit the file `/etc/default/locale` and add the two following lines:

```
LANGUAGE="en_US.UTF-8"
LC_ALL="en_US.UTF-8"
```

Save, logout and log back in.

See this post for example for a description of this problem.

Install system packages

```
# Generic packages needed
sudo apt-get install git python-virtualenv python3-setuptools python-dev python3-dev␣
↪build-essential

# PostgreSQL dependencies
sudo apt-get install libpq-dev

# federation dependencies
sudo apt-get install libxml2-dev libxslt-dev lib32z1-dev

# Redis
sudo apt-get install redis-server
```

Install Node.js

Node.js version 6-8 has been tested to work. Install one by following the Node.js install guides.

If you already have a system Node installed which is either too old or newer, consider using [NVM](https://github.com/creationix/nvm) to install a specific version for Socialhome only.

Install PostgreSQL

If not installed or not using a remote PostgreSQL DB, install the database engine.

```
sudo apt-get install postgresql
```

Create a database and user. Note down password for later.

```
sudo su - postgres
createuser -P socialhome
createdb -O socialhome socialhome
exit
```

Create a local user

It's better to run applications under their own user.

```
sudo adduser socialhome --disabled-login
sudo chmod 750 /home/socialhome

# Add user group to www-data groups so we can protect users home folder
sudo adduser www-data socialhome
```

Set up uWSGI

```
# Create logs path
sudo -u socialhome mkdir /home/socialhome/logs
```

Create the ini file with `/home/socialhome/uwsgi.ini` and add the following contents to it.

```
[uwsgi]
chdir=/home/socialhome/socialhome
module=config.wsgi:application
master=True
pidfile=/tmp/socialhome-master.pid
vacuum=True
max-requests=5000
logto=/home/socialhome/logs/uwsgi-master.log
virtualenv=/home/socialhome/.virtualenvs/socialhome
processes=2
threads=2
enable-threads=True
socket=127.0.0.1:31452/
uid=socialhome
gid=socialhome
harakiri=30
```

Set up Apache

if not already installed, install the Apache2 web server.

```
sudo apt-get install apache2 libapache2-mod-proxy-uwsgi
```

Enable some necessary modules.

```
sudo a2enmod proxy_uwsgi
sudo a2enmod proxy_wstunnel
sudo a2enmod proxy
sudo a2enmod ssl
```

Add an Apache virtualhost file `/etc/apache2/sites-available/socialhome.conf` with the following content, replacing instances of `yourdomain.tld` with your real domain for your Socialhome instance:

```
<VirtualHost *:80>
    ServerName yourdomain.tld
    ServerAlias www.yourdomain.tld
    RedirectPermanent / https://yourdomain.tld/
</VirtualHost>

<VirtualHost *:443>
```

(continues on next page)

(continued from previous page)

```
    ServerName yourdomain.tld
    ServerAlias www.yourdomain.tld
    ServerAdmin webmaster@yourdomain.tld

    Alias /robots.txt /home/socialhome/socialhome/staticfiles/robots.txt
    Alias /favicon.ico /home/socialhome/socialhome/staticfiles/favicon.ico
    Alias /media /home/socialhome/socialhome/socialhome/media

    <Directory /home/socialhome/socialhome/socialhome/media>
        Require all granted
        Options -MultiViews -Indexes
    </Directory>

    ProxyPass /media !
    ProxyPass /ch/ ws://127.0.0.1:23564/ch/
    ProxyPass / uwsgi://127.0.0.1:31452/

    SSLEngine on
    SSLCertificateFile /etc/letsencrypt/live/yourdomain.tld/cert.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/yourdomain.tld/privkey.pem
    SSLCertificateChainFile /etc/letsencrypt/live/yourdomain.tld/chain.pem
</VirtualHost>
```

Enable Apache virtualhost

```
sudo a2ensite socialhome
```

Get LetsEncrypt certificate

We wouldn't want to run our site without HTTPS. Install Certbot and get an LetsEncrypt certificate.

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:certbot/certbot
sudo apt-get update
sudo apt-get install python-certbot-apache
```

Launch Certbot and answer any questions to install the certificates.

```
certbot --apache certonly
```

Now you should be able to restart Apache.

```
sudo service apache2 restart
```

Change to Socialhome user

Change to user `socialhome` for the rest of the guide.

```
sudo su - socialhome
```

Install Virtualenvwrapper

This is the easiest way to manage Python virtualenvs. We also add production Django configuration reference at the same time.

```
pip install --user virtualenvwrapper
```

Add the following lines to your `.bashrc` and reload it via `source ~/.bashrc`.

```
export WORKON_HOME=$HOME/.virtualenvs
source ~/.local/bin/virtualenvwrapper.sh
export DJANGO_SETTINGS_MODULE=config.settings.production
```

Create Python virtualenv

Python 3.6+ is required! If your system Python 3 is not at least this version, please install Python 3.6 and replace `/usr/bin/python3` in the below command with the path to the Python 3.6 binary.

```
mkvirtualenv -p /usr/bin/python3 socialhome
```

The virtualenv is automatically activated. When you need it in the future, just type `workon socialhome`.

Update pip and setuptools

```
pip install -U pip setuptools
```

Install pip-tools

`pip-tools` is a handy tool to keep environments clean and all dependencies nicely pinned.

```
pip install -U pip-tools
```

Get Socialhome code

```
git clone https://git.feneas.org/socialhome/socialhome
cd socialhome
```

Install Python dependencies

We use the `pip-tools` command to ensure dependencies are at the correct versions.

```
pip-sync
```

Create configuration

Create the file `.env` with the following contents, replacing values as needed.

You must change or add the following values:

- Replace `DATABASEPASSWORDHERE` with the database password typed in earlier.
- `DJANGO_SECRET_KEY` must be added. Generate one for example here.
- Place your domain in `DJANGO_ALLOWED_HOSTS` and `SOCIALHOME_DOMAIN`.

```
DATABASE_URL=postgres://socialhome:DATABASEPASSWORDHERE@127.0.0.1:5432/socialhome
DJANGO_SECRET_KEY=
DJANGO_ALLOWED_HOSTS=yourdomain.tld
DJANGO_SECURE_SSL_REDIRECT=True
DJANGO_ACCOUNT_ALLOW_REGISTRATION=True
SOCIALHOME_DOMAIN=yourdomain.tld
SOCIALHOME_HTTPS=True
SOCIALHOME_LOGFILE=/home/socialhome/logs/socialhome.log
```

For further configuration tips, see *Running an instance*.

Make the env file a bit less readable.

```
chmod 0600 .env
```

Configure email sending

Note, email *is* required for signing up. Users will **not** be able to sign up if the instance does not have working email sending. See *DJANGO_EMAIL_BACKEND* on how to configure email sending.

Run migrations

```
python manage.py migrate
```

Install statics

```
npm install
node_modules/.bin/bower install
npm run build
python manage.py collectstatic
```

Search index

The search indexes must be initialized, otherwise there will be an error when trying to use search. Run this command once:

```
python manage.py rebuild_index
```

Any further changes to indexes objects will be maintained automatically from this point onwards. If you ever need to rebuild the index from scratch, use the same command.

Set the correct domain name in Django

Load up the Django shell with `python manage.py shell_plus` and then execute the following, replacing "yourdomain.tld" with your domain and "Socialhome" as the name of your site, assuming you want the name changed:

```
Site.objects.filter(id=1).update(domain="yourdomain.tld", name="Socialhome")
exit
```

Set up Circus

Exit Socialhome user and create Upstart configuration for Circus process manager. Circus is used to control various processes that are needed in addition to the web server. This allows starting one process that will start and maintain a bunch of other processes we need. A configuration file for the processes is provided within the repository.

Create Upstart configuration `/etc/init/socialhome.conf` with the following content:

```
description "Socialhome"
start on runlevel [2345]
stop on runlevel [06]
setuid socialhome
setgid socialhome

respawn

env PYTHONPATH="/home/socialhome/socialhome"
env SOCIALHOME_HOME="/home/socialhome"
env RQWORKER_NUM=5
env VIRTUAL_ENV=/home/socialhome/.virtualenvs/socialhome
env LC_CTYPE=en_US.UTF-8
env LC_ALL=C.UTF-8
env LANG=C.UTF-8
env DJANGO_SETTINGS_MODULE=config.settings.production

chdir /home/socialhome/socialhome

exec /home/socialhome/.virtualenvs/socialhome/bin/circusd config/circus.ini
```

Start Circus. It will automatically start on system boot.

```
sudo service socialhome start
```

For a SystemD config file, see *Other Linuxes or newer Ubuntu using SystemD*.

Done!

That wasn't so hard was it? Navigate to the domain you chose to install Socialhome on and hopefully you will see a landing page. Signups will be open.

Final tweaks

Unless you want to keep signups open, after creating your own account, you should close the signups to avoid random people signing up to your instance. See configuration tips at *Running an instance*.

If you didn't configure emails, you cannot complete your user account registration without the email confirmation link. See *Confirming user emails via the shell*.

If you want to set your initially created user as admin, see *Admin user*.

Terms of Service and Privacy policy documents are good to have. These tell people visiting your site what rules you operate with. Socialhome provides default templates you can activate with a few clicks. See *Policy documents* for more info.

7.1.6 Other Linuxes or newer Ubuntu using SystemD

Follow the Ubuntu 14.04 guide, tweaking it to your system. For SystemD, try the following service config (for example saved to `/etc/systemd/system/socialhome.service`):

```
[Unit]
Description=Socialhome Django script
After=syslog.target network.target

[Service]
Environment=DJANGO_SETTINGS_MODULE="config.settings.production"
Environment=PYTHONPATH="/home/socialhome/socialhome"
Environment=SOCIALHOME_HOME="/home/socialhome"
Environment=RQWORKER_NUM=5
Environment=VIRTUAL_ENV=/home/socialhome/.virtualenvs/socialhome

User=socialhome
Group=socialhome

WorkingDirectory=/home/socialhome/socialhome
ExecStart=/home/socialhome/.virtualenvs/socialhome/bin/circusd /home/socialhome/
↪socialhome/config/circus.ini
Restart=always

[Install]
WantedBy=multi-user.target
```

7.1.7 Other platforms

PR's welcome for guides for more platforms!

7.2 Updating

If using the installation-docker, any actions required by an upgrade will be handled by just updating the image tag and recreating the container.

If using the *Ubuntu via Ansible* there is no need to do anything special. Just re-run the role!

For manual installs, see the following steps. Commands might vary depending on your OS.

7.2.1 Check the changelog

When updating the code, make sure you check the *Changelog* for any notes about the changes. Sometimes extra manual steps might be required or an update could take a long time due to database migrations.

7.2.2 Change to Socialhome user

Change to user `socialhome` for the rest of the guide.

```
sudo su - socialhome
```

7.2.3 Activate virtualenv

```
workon socialhome
```

7.2.4 Pull in latest code or release

To pull in a release:

```
# Replace release tag with the release, for example "v0.3.1"
git fetch && git checkout <release tag>
```

To pull in master branch head:

```
git pull
```

7.2.5 Install Python dependencies

We use the `pip-tools` command to ensure dependencies are at the correct versions.

```
pip-sync
```

7.2.6 Run migrations

```
python manage.py migrate
```

7.2.7 Install statics

```
npm install
node_modules/.bin/bower install
npm run build
python manage.py collectstatic
```

7.2.8 Restart the app

```
sudo service socialhome restart
```

7.2.9 Done!

Check the application and have fun!

7.3 Running an instance

Some notes on running a production instance.

7.3.1 Django admin

The normal Django admin can be found at `/admin`.

7.3.2 Executing the Django shell

Assuming included installation instructions were used, do the following:

```
sudo su - socialhome
workon socialhome
cd socialhome
python manage.py shell_plus
```

7.3.3 Confirming user emails via the shell

You can manually confirm user emails via the shell by running the following:

```
from allauth.account.models import EmailAddress
EmailAddress.objects.filter(email=<email>).update(verified=True)
```

This will allow the user to log in without clicking the confirmation email link.

7.3.4 Admin user

To make a user an admin, log in to the shell and execute the following to set the user as superuser:

```
from socialhome.users.models import User
User.objects.filter(username=<username>).update(is_staff=True, is_superuser=True)
```

7.3.5 Backups

Three places should be backed up from the Socialhome instance to ensure recovery in the event of a disaster.

- The database
- Local settings in `.env` (assuming you are using this way to configure the application)
- The path `socialhome/media/` which contains for example image uploads
- The Redis database (example instructions)

7.3.6 Give your instance some visibility

If you want some public visibility to your instance, consider registering it at some lists that track nodes in "The Federation". Here are a few:

- https://the-federation.info

- https://podupti.me

Why not also contribute to the numbers of the federated social web? Turn on *SOCIALHOME_SILKY* to expose some activity counts.

7.3.7 Log files

There are two main logs where Socialhome sends information during runtime.

- Circus process log

  Rotated log files in `/var/log/upstart/socialhome-circus.log`. The location will differ if not using an Upstart based system.

  This log contains the output of all the processes required to run Socialhome, if using the recommended way of running Socialhome using Circus. Any errors for example when starting uWSGI or the worker processes will be found here.

- Application log

  See *SOCIALHOME_LOG_TARGET* configuration value. This log contains logging entries from the application itself. Useful for debugging federation issues or other problems with the actual code.

7.3.8 Deleting users and locking remote profiles

To delete users and their content, a Django management command has been provided. This command can also be used to delete local content of remote profiles and optionally lock the profile so any new content is rejected. This makes it possible to lock out spam accounts for example. For locally created content, an automatic retraction will be sent to remotes.

NOTE! Any deletion is **permanent**. There is no possibility to get the data back, except by restoring database and uploaded file backups. Be sure before using the command and be extra sure about the UUID's passed in!

To delete a local user, run the command as follows:

```
python manage.py delete_users_and_profiles --users <uuid>
```

To delete a remote profile, run the command as follows:

```
python manage.py delete_users_and_profiles --profiles <uuid>
```

To only delete remote profile content and then lock the profile, run as follows:

```
python manage.py delete_users_and_profiles --profiles <uuid> --lock-remote-profiles
```

Multiple `uuid`'s can be passed in by separating them with commas. A confirmation dialog is produced for each user or profile to be deleted.

7.3.9 Policy documents

Terms of Service and Privacy policy documents are good to have. These tell people visiting your site what rules you operate with. Socialhome provides default templates you can activate with a few clicks.

To review and enable the policy documents, log in as admin and access the admin pages through the navigation bar cogs menu. Scroll down and locate "Policy documents". There are two types of documents, the Terms of Service and Privacy Policy. Each one can be edited in draft mode and then published. Further updates in draft mode will not overwrite the last published version, until published.

To publish the documents, open them, review the text and then change the status below the document to "published". Click Save - this version is now published. To edit in draft mode, switch the status back and the current edited revision will not show to users. You can also send email updates to users from the policy documents list. Select the policy documents you wish the send an email about, choose "Send email" from the actions list and confirm.

Published policy documents are shown to both authenticated and unauthenticated users via the navigation bar cogs menu.

7.3.10 Configuration

Configuration mainly happens through environment variables. Those are passed to Django via the file `.env` in the repository root. The following items of note can be changed.

After making changes to this file, don't forget to reload the app with `sudo service socialhome restart`.

DATABASE_URL

Default: `postgres:///socialhome`

This must be set to a proper database URL, for example `postgres://socialhome:DATABASEPASSWORDHERE@127.0.0.1:5432/socialhome`.

DJANGO_ACCOUNT_ALLOW_REGISTRATION

Default: `True`

Set this to `False` if you want to disable signups.

DJANGO_ADMIN_MAIL

Default: `info@socialhome.local`

Admin email for terms of service and privacy policy documents, outgoing emails and server metadata.

DJANGO_ADMIN_NAME

Default: `Socialhome Admin`

Admin display name or organization name for Terms of Service, outgoing emails and server metadata.

DJANGO_ALLOWED_HOSTS

Default: `socialhome.local`

Domain that is used for this instance. Must be set to the right domain. Note, it's not a good idea to use a sub-domain wildcard for www, ie `.` as per Django docs. Federated sites work better with only one absolute domain.

DJANGO_EMAIL_BACKEND

Default: `django.core.mail.backends.console.EmailBackend`

Must be set to some real email backend if you wish to send emails. See docs for backend options and additional configuration help.

The possible other email related additional settings are as follows. Please see Django documentation link above for details.

- `DJANGO_EMAIL_HOST` (default `localhost`)
- `DJANGO_EMAIL_PORT` (default `587`)
- `DJANGO_EMAIL_HOST_USER` (default '')
- `DJANGO_EMAIL_HOST_PASSWORD` (default '')
- `DJANGO_EMAIL_USE_TLS` (default `True`)
- `DJANGO_EMAIL_USE_SSL` (default `False`)
- `DJANGO_EMAIL_TIMEOUT` (default '')
- `DJANGO_EMAIL_SSL_KEYFILE` (default '')
- `DJANGO_EMAIL_SSL_CERTFILE` (default '')
- `DJANGO_EMAIL_SUBJECT_PREFIX` (default `[Socialhome]`)
- `DJANGO_DEFAULT_FROM_EMAIL` (default `noreply@socialhome.local`)
- `DJANGO_SERVER_EMAIL` (defaults to `DJANGO_DEFAULT_FROM_EMAIL` value)

Note, email *is* required for signing up. Users will **not** be able to sign up if the instance does not have working email sending.

DJANGO_SECRET_KEY

Default: ''

Must be set to a long secret string. Don't expose it to anyone. See docs

DJANGO_SECURE_CONTENT_TYPE_NOSNIFF

Default: `True`

See docs.

DJANGO_SECURE_FRAME_DENY

Default: `True`

See docs.

DJANGO_SECURE_HSTS_INCLUDE_SUBDOMAINS

Default: `True`

See docs.

DJANGO_SECURE_SSL_REDIRECT

Default: `True`

Redirect all requests to HTTPS. See docs.

DJANGO_TIMEZONE

Default: `UTC`

REDIS_DB

Default: `0`

REDIS_HOST

Default: `localhost`

REDIS_PASSWORD

Default: ''

REDIS_PORT

Default: `6379`

SENTRY_DSN

Default: `None`

Setting a Sentry project DSN will make all error level exceptions be raised to Sentry. To change the level, see below.

SENTRY_LEVEL

Default: `ERROR`

Logging level used for Sentry reporting (see above). Possible options: `DEBUG`, `INFO`, `WARNING`, `ERROR`.

SOCIALHOME_ADDITIONAL_APPS

Default: `None`

Allows to plug in additional third-party apps, string with comma-separated values, for example `django.contrib.gis,myapp`.

SOCIALHOME_ADDITIONAL_APPS_URLS

Default: `None`

Allows to use additional third-party app url-conf, string with two comma-separated values, url prefix and path to urlpatterns, for example `myapp/,myapp.urls`. If you need to include urls from more than one app, this could be done by creating intermediary app which aggregates urls.

SOCIALHOME_DOMAIN

Default: `socialhome.local`

Must be set to your Socialhome instance domain. Used for example to generate outbound links.

SOCIALHOME_HOME_VIEW

Default: `None`

Allows to use on main page custom view from third-party app, string with path to view, for example `myapp.views.AwesomeHomeView`.

SOCIALHOME_HTTPS

Default: `True`

Force HTTPS. There should be no reason to turn this off.

SOCIALHOME_LOG_DISABLE_ADMIN_EMAILS

Default: `False`

Set this to `True` to disable the Django admin error emails in production.

SOCIALHOME_LOG_TARGET

Default: `file`

Define target for Django and application logs. Possible options:

- `file`, logs will go to a file defined in `SOCIALHOME_LOGFILE`. Note, due to multiple processes logging to the same file, this file log is only really useful for tailing or if running different processes on separate containers or machines.
- `syslog`, logs to syslog, to the `local7` facility.

SOCIALHOME_LOGFILE

Default: `/tmp/socialhome.log`

Where to write the main application log.

SOCIALHOME_NODE_LIST_URL

Default: `https://the-federation.info/socialhome`

URL to make signup link go to in the case that signups are closed.

SOCIALHOME_RELAY_ID

Default: `relay@relay.iliketoast.net`

Which relay instance ID to send outgoing content to. For more info see relay system.

SOCIALHOME_RELAY_SCOPE

Default: `all`

Possible values `all` or `none`. Defines at which level the defined relay is subscribed with. Set to `none` to disable the relay incoming subscription. For more info see relay system.

SOCIALHOME_ROOT_PROFILE

Default: ''

If this is set to a local username, that users profile will be shown when navigating to `/` as not logged in user. Logged in users will still see their own profile. Good for single user instances.

SOCIALHOME_SHOW_ADMINS

Default: `False`

If set to `True`, allows showing the server admins to users and in server metadata. The settings used are `DJANGO_ADMIN_NAME` and `DJANGO_ADMIN_MAIL`.

SOCIALHOME_SILKY

Default: `False`

Set to `True` to enable the Django-Silk debugging tool. Information about requests will be available at `/_silk/`.

SOCIALHOME_STATISTICS

Default: `False`

Controls whether to expose some generic statistics about the node. This includes local user, content and reply counts. User counts include 30 day and 6 month active users.

SOCIALHOME_STREAMS_PRECACHE_SIZE

Default: `100`

Amount of items to keep in stream precaches, per user, per stream. Increasing this setting can radically increase Redis memory usage. If you have a lot of users, you might consider decreasing this setting.

Note the amount actually stored can temporarily go over the limit. Cache trimming is done as a daily job, not every time a new item needs to be added to the cache.

SOCIALHOME_STREAMS_PRECACHE_INACTIVE_DAYS

Default: `90`

Amount of days since user has logged in to be considered inactive for streams precaching. See notes about `SOCIALHOME_STREAMS_PRECACHE_INACTIVE_SIZE`.

SOCIALHOME_STREAMS_PRECACHE_INACTIVE_SIZE

Default: `0`

Amount of items to keep in stream precaches, per user, per stream, for inactive and anonymous users. By default maintenance will always clear the cache for inactive and anonymous users daily. See notes about `SOCIALHOME_STREAMS_PRECACHE_SIZE`.

SOCIALHOME_SYSLOG_FACILITY

Default: `local7`

Define the logging facility for syslog, if `SOCIALHOME_LOG_TARGET` is set to `syslog`.

SOCIALHOME_SYSLOG_LEVEL

Default: `INFO`

Define the logging level of syslog logging, if `SOCIALHOME_LOG_TARGET` is set to `syslog`. Possible options: `DEBUG`, `INFO`, `WARNING`, `ERROR`.

SOCIALHOME_TOS_JURISDICTION

Default: `None`

Define what jurisdiction (country) should be printed on the terms of service document. If not given, jurisdiction will not be included in the terms of service documents.

7.4 API

Socialhome has a REST API. This allows to build clients, bots and alternative frontends.

Note, some parts of the API are still work in progress and thus changes could still happen.

7.4.1 API routes

The API methods and data can be browsed using the ReDoc docs endpoint `/api/`. Shown endpoints and data depends on your user credentials (log-in using the menu if not already). If you prefer Swagger docs, you can find them at `/api/swagger/`.

7.4.2 Authenticating

The API supports two authentication methods:

Session authentication

This means when you are logged into Socialhome, you automatically have usage of the API from the browser. Note however that POST/PUT/PATCH methods will require CSRF tokens.

Token authentication

API authentication can happen by setting the required HTTP header as follows:

```
Authorization: Token 9944b09199c62bcf9418ad846dd0e4bbdfc6ee4b
```

Your client can obtain a token for the user by posting `username` and `password` as form data or JSON to the view `/api-token-auth/`. This will return a token and limited user profile informations as follows:

```
{
  "url": "http://127.0.0.1:8000/p/68f61327-1305-4354-b56c-6549d196a325/",
  "image_url_small": "http://127.0.0.1:8000/static/images/pony50.png",
  "uuid": "68f61327-1305-4354-b56c-6549d196a325",
  "is_local": true,
  "handle": "user-0@127.0.0.1:8000",
  "home_url": "http://127.0.0.1:8000/p/68f61327-1305-4354-b56c-6549d196a325/",
  "name": "",
  "token": "2a37e74f235b044b275141b2d4605ca900617d13",
  "id": 1
}
```

Users can also retrieve and regenerate tokens from the UI from their profile menu.

7.4.3 Development

The API is made with Django REST Framework. Help is welcome to expand the API!

7.4.4 Clients

See the *Clients* section.

7.5 Clients

This page will have a list of clients using the Socialhome *API*. Please send PR's if you have made one!

7.5.1 shcli

This is the official Python library and command line client.

See documentation and code on GitHub.

7.6 Community

7.6.1 Official project account

To keep up to date with the project you can follow the official project account from Socialhome or compatible software.

The official account can be found at `hq@socialhome.network` (https://socialhome.network/u/hq/).

7.6.2 Feedback and community chat

We have a few chat roomts you can join. All of these are bridged so you only need to join with which ever is your favourite to use.

- Matrix room `#socialhome:feneas.org`
- Freenode IRC, channel `#socialhome` (webchat here)

Join in if you have questions regarding installation, development or usage, or just to say hi!

7.7 Development

Socialhome is missing features and needs a lot of polish on the UI side. If you are familiar with Django or Vue (or want to learn!) and are interested in getting involved, please don't hesitate to get in touch!

For guidelines how to contribute, please first read the *Contributing* guide.

- Source code repo
- Issue tracker
- Kanban board

7.7.1 Environment setup

Instructions are for Ubuntu 16.04+ (+ simple Alpine 3.6 dependencies script). Please contribute via PR's if you notice anything missing or want to contribute instructions for another platform.

Python Virtualenv

Python 3.6 is the minimum supported version. Ensure the following are installed:

- Python system dependencies
- NodeJS (version 6-8)
- PostgreSQL server
- Redis

The file `requirements.apt` contains other various dependencies. You can use the `install_ubuntu_dependencies.sh` script to help installing these.

You can use the `install_alpine_dependencies.sh` script to install required dependencies (including Python, NodeJS, PostgreSQL and Redis) on Alpine.

To generate profiling SVG's with pytest, also install the `graphviz` package (for example from apt), which provides `dot`.

Install Python dependencies

We use `pip-tools` as the way to install Python dependencies. All the "base" dependencies, including production deployment dependencies are locked in `requirements.txt`. The file `dev-requirements.txt` includes both the base and the extra development/testing related dependencies.

To use `pip-tools`, first install it:

```
# Ensure pip and setuptools are up to date as well
pip install -U pip pip-tools
```

Then install dependencies:

```
# Production environment
pip-sync

# Development environment
pip-sync dev-requirements.txt
```

It is not mandatory to use `pip-tools` for running a production installation. For development it is mandatory. All dependencies should be placed (unlocked) in either `requirements/requirements.in` (base) or `requirements/requirements-dev.in` (development extras). Then execute `./compile-requirements.sh` to update the locked dependency files after each change to the `.in` files. See pip-tools for more information.

Do NPM, Bower

```
npm install
node_modules/.bin/bower install
npm run dev
```

To watch files and build bundles automatically, use this.

```
npm run watch
```

Configure

Configuration is done via environment variables. For the meaning of them, look them up under files in `config/settings`. Values in the file `.env` will be used automatically.

```
cp .env.example .env
```

Edit any values necessary. By default the `SECRET_KEY` is empty. You MUST set something to it. We don't supply a default to force you to make it unique in your production app.

Create a database

Docker

The easiest way to run a database is using Docker. Install Docker and then use the following command:

```
docker run -e 'POSTGRES_PASSWORD=socialhome POSTGRES_USER=socialhome POSTGRES_
↪DB=socialhome' \
    --name socialhomedb -d -p '5432:5432' postgres
# Migrate the database
python manage.py migrate
```

Later to start the same container just do:

```
docker start socialhomedb
```

System

For example on Ubuntu:

```
sudo su - postgres
createuser -s -P socialhome  # give password 'socialhome'
createdb -O socialhome socialhome
exit
python manage.py migrate
```

Install Redis

Either use system Redis (for example on Ubuntu *sudo apt install redis-server*) or use Docker:

```
docker run --name socialhomeredis -d -p '6379:6379' redis
```

To later start it:

```
docker start socialhomeredis
```

Running the development server

Just use the standard command:

```
python manage.py runserver
```

Unfortunately `runserver_plus` cannot be used as it does not integrate with Django Channels.

Running the Django shell

It is recommended to use the enhanced "shell plus" provided by Django Extensions package which is automatically installed via the development dependencies. One of the benefits is that it will automatically import all project models.

To launch the shell:

```
python manage.py shell_plus
```

Creating a user

To create an *superuser account*, use this command:

```
python manage.py createsuperuser
```

After this you need to log in once with the user via the user interface (which creates an email confirmation) and then run the following in the Django shell to confirm the email:

```
from allauth.account.models import EmailAddress
EmailAddress.objects.all().update(verified=True)
```

You should now be able to log in as the user `admin`.

Search index

The search indexes must be initialized, otherwise there will be an error when trying to use search. Run this command once:

```
python manage.py rebuild_index
```

Any further changes to indexes objects will be maintained automatically from this point onwards. If you ever need to rebuild the index from scratch, use the same command.

7.7.2 Running tests

The user needs right to create databases:

```
# On some distributions, regular users are not sudoers; you may need to type:
#    su -
su - postgres
psql -c "ALTER USER socialhome CREATEDB;"
```

Python tests

```
py.test
```

To also generate profiling information, add `--profile --profile-svg` to the command.

JavaScript tests

Execute the following to run the frontend JavaScript tests.

```
npm run test
```

7.7.3 API Routes

There is a dependency in the API route URL configurations with the new Vue based frontend tests. If you change or add new API routes during development, you must also do the following:

```
python manage.py collectstatic_js_reverse
mv staticfiles/django_js_reverse/js/reverse.js socialhome/frontend/tests/unit/
↪fixtures/Urls.js
```

This updates the JavaScript fixtures with the new URL configuration.

7.7.4 Linters

ESLint

There is an `.eslintrc` provided. We follow the Airbnb and Vue guidelines with some tweaks. It's recommended to add this configuration to your editor directly. To run ESLint directly, use the following command. NOTE! This is only valid for the new Vue based frontend, not JS in `socialhome/static`.

```
npm run lint
```

7.7.5 Building local documentation

```
cd docs
make html
```

7.7.6 Doing a release

Bump version number in three places:

- `socialhome/__init__.py`
- `docs/conf.py`
- `docs/changelog.rst`

To generate a markdown version of the release changelog, first install `Pandoc`:

```
sudo apt install pandoc
```

Then execute the following and copy the markdown version for pasting to GitHub releases or a Socialhome post:

```
pandoc --from rst --to markdown_github docs/changelog.rst | less
```

Docker images

Publish a new Docker image by running *./docker_release v<version>*.

Start new development cycle

Set a development version in the same above files. This is basically the next minor release postfixed by `-dev`.

Commit and author stats

Some commands to get nice stats for release posts.

Authors

```
git shortlog -s -n -e <first release commit>..HEAD --no-merges
```

Changes

```
git diff --stat <first release commit>..HEAD
```

7.7.7 Developing with Docker

If you choose, you may develop Socialhome using Docker, rather than installing Postgres and Redis manually on your computer.

Supported versions

This guide assumes you are running Docker on a GNU/Linux based system such as Ubuntu, Debian or Fedora Linux. It may be possible to run this on other platforms where Docker is supported, but those are untested.

The docker development installation was tested on Docker version 17.09 and docker-compose 1.16.1.

Steps

The first step is to copy the example docker-compose file `docker/dev/docker-compose.yml.example` file to the root of the project. eg

`cp docker/dev/docker-compose.yml.example ./docker-compose.yml`

You also need to set an .env file as per the above instructions. Use the `.env.example` as a starting point.

From there, you can build the images:

`docker-compose build`

And then the steps you would normally do, but throught he django image, ala:

`docker-compose run django manage migrate` and `docker-compose run django manage createsuperuser`

And then just

`docker-compose up`

Defaults

The defaults are that that the Docker image will be running on port 8000 and then exposed to the host OS on the same port (ie you can browse to http;//localhost:8000 to see the Django instance running). Redis and Postgres will be running but not exposed to the host OS by default. These can be changed on the `docker-compose.yml` file.

7.7.8 Generating dummy content

There is a management command to generate a bunch of dummy `Content` objects. Please feel free to expand it with more configuration options and different types of content. To use it, run the following:

```
python manage.py create_dummy_content
```

`--help` will give you available options.

7.7.9 Contact for help

See our communication channels in the *Community* page.

You can also ask questions or give feedback via issues.

7.8 Contributing

Want to contribute to Socialhome? Great! <3 Please read on for some guidelines.

7.8.1 First things first

Please make sure you have some knowledge of what the software is for before jumping in to write code. You don't even have to install a development environment. Just head to https://socialhome.network, create an account and play around.

If you already are a user or run your own instance, you probably have some ideas on how to contribute already. The best contributions come from real personal need.

7.8.2 Finding things to do

We have an issue tracker on GitHub. If you don't already have an idea on what to do, check out the issues listed there. Some issues are labeled as newcomer. These are easy picking tasks for those either new to Socialhome or with less knowledge of Django.

7.8.3 Logging issues

Contributions are not just code. Please feel free to log not only bugs but also enhancement ideas in the issue tracker. For issues that have not been confirmed (= they don't have the label "ready"), triaging is important contribution also. Reproducing and supplying more information on these issues is valuable contributon to the project.

Welp I found a security issue, shall I just file an issue?

7.8.4 Writing code

So you're ready to write code! Great! Please remember though that the project already has a vision, the software has architecture and the project maintainer will have strong opinions on how things should be implemented. Before you write a lot of code that even remotely feels like it would need a design decision, please *always* discuss your plan first with the project maintainer. Otherwise you might spend a lot of time writing code only to be told the code will not be merged because it doesn't fit into the grand plan.

Please don't be afraid to get in touch, see channels in the *Community* pages.

7.8.5 Creating pull requests

Before submitting a pull request, please ensure you've read and understood the following checklist.

Checklist

Please review the guidelines for contributing (https://socialhome.readthedocs.io/en/latest/contributing.html) to this repository and then go through the following checklist.

- Make sure you are requesting to merge a non-master branch from your fork. Do not create PR's from your *master* branch!
- Does your code have unit tests? All major code paths should be tested sufficiently. See existing tests for examples.
- Did you run the whole test suite through and ensure no existing tests break?
- Please check the documentation whether your PR will require changes or additions to any documentation pages. Use proper English!
- If changing or adding UI pages or elements add screenshots to the PR.
- If the PR is not ready to be merged:
  - Prefix the PR title with "[WIP] ".
  - Add a list of TODO's that are missing from the PR to the description.
- Consider adding a changelog entry to the current unreleased version. Use proper English, reference the right issue (if any, not your PR) and make sure to add the entry to the correct section ("Added", "Changed", "Fixed" or "Removed"). If you are not sure whether you need a changelog entry - don't add one!
- Ensure commit messages are descriptive and sufficiently detailed. If possible, split the feature into smaller easier to follow commits. As a rule of thumb, small PR's and small commits are always preferred. See https://chris.beams.io/posts/git-commit/ for notes on what is a good commit.

Thank you!

7.8.6 Reviewing code

Code review is a valuable way to contribute, and also to learn about the code base! Don't be afraid to give some comments to open pull requests! You don't have to be a veteran or know everything to be able to give opinions. Pull request reviews are not just for reviewing, they're a valuable opportunity for learning too.

7.8.7 Tests

As a general rule all code must have unit tests. For bug fixes provide a test that ensures the bug will not be back and for features always add a good enough coverage. PR's without sufficient test coverage will not be merged.

Testing tools

Django

We use `py.test` as test runner but the tests themselves are Django based test classes. We have our own base classes which should be used as a base for all Django tests. Some old tests are pure py.test function based tests, feel free to convert these to Django test classes.

Focus is placed in pure unit tests instead of complex integration or browser tests. In terms of coverage, 100% is not the key, meaningful tests and coverage of critical lines is. Don't worry if a PR drops coverage a bit if the coverage diff clearly shows all critical code paths are covered by meaningful tests.

Vue

The JS tests are using the Avoriaz Vue test utils and Mocha test runner.

7.8.8 Code style

Python

As a general rule, for Python code follow PEP8, except with a 120 character line length. We provide an `.editorconfig` in the repository root.

JavaScript

There is an `.eslintrc` configuration provided.

Alphabetical ordering

When possible, try to always make all list items, dict keys, class methods, classes / functions in file, etc, everything alphabetically organized. This helps finding things when files grow and classes get a lot of methods. Sometimes this is not possible, for example when classes subclass another class in the same file. In this case for example, alphabetical ordering can be forgotten for logical placement.

7.8.9 Python dependencies

We use `pip-tools` as the way to install Python dependencies. All the "base" dependencies, including production deployment dependencies are locked in `requirements.txt`. The file `dev-requirements.txt` includes both the base and the extra development/testing related dependencies.

To use `pip-tools`, first install it:

```
# Ensure pip and setuptools are up to date as well
pip install -U pip pip-tools
```

Then install dependencies:

```
# Production environment
pip-sync

# Development environment
pip-sync dev-requirements.txt
```

It is not mandatory to use `pip-tools` for running a production installation. For development it is mandatory. All dependencies should be placed (unlocked) in either `requirements/requirements.in` (base) or `requirements/requirements-dev.in` (development extras). Then execute `./compile-requirements.sh` to update the locked dependency files after each change to the `.in` files. See pip-tools for more information.

7.9 Brand

Documentation relating to branding of Socialhome as a product. Each server in the network is free to apply whatever branding for their particular server.

When talking about Socialhome as software or a platform, the following graphics and colour schemes should be used.

7.9.1 Logo

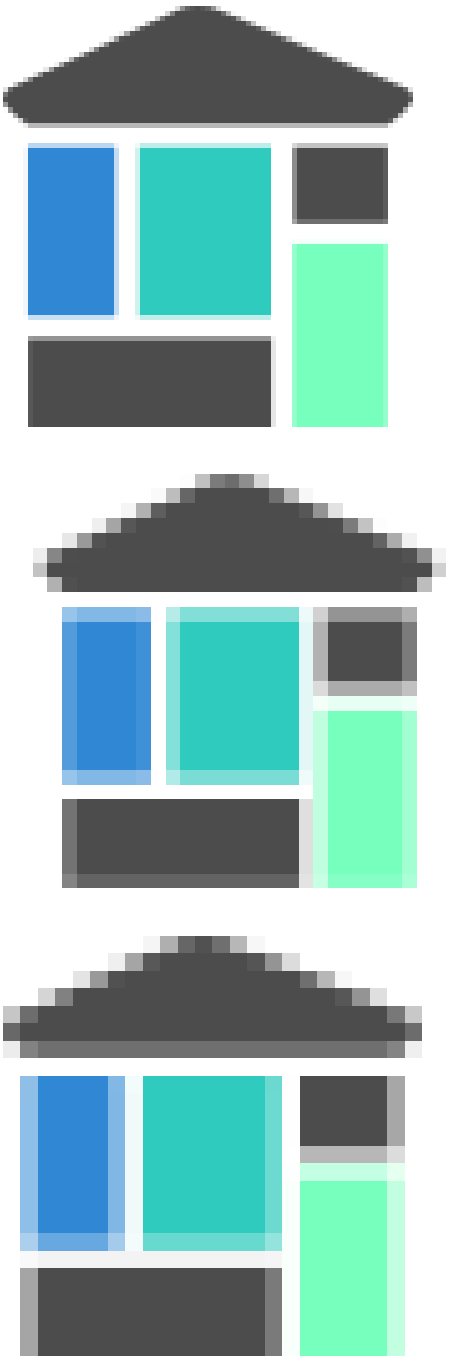
Our logo is available as an SVG and PNG's of various sizes. The logo comes in a dark and light variant, for different backgrounds.

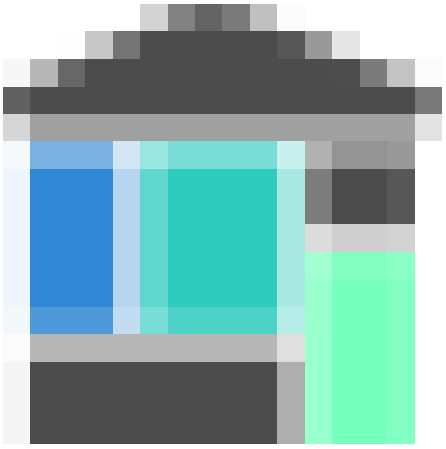
The logo is contributed by lostinlight, licensed under WTFPL.

Dark

SVG

PNG

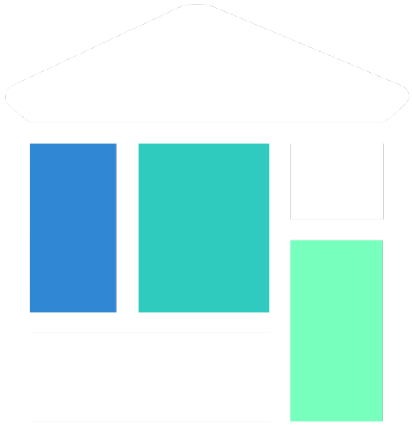


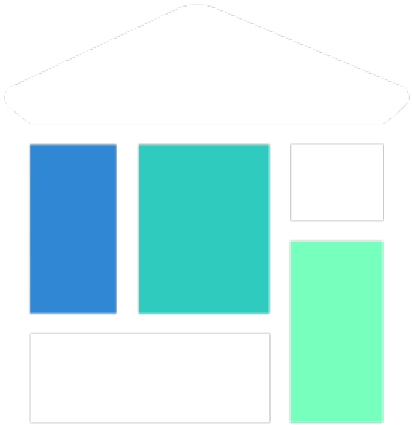


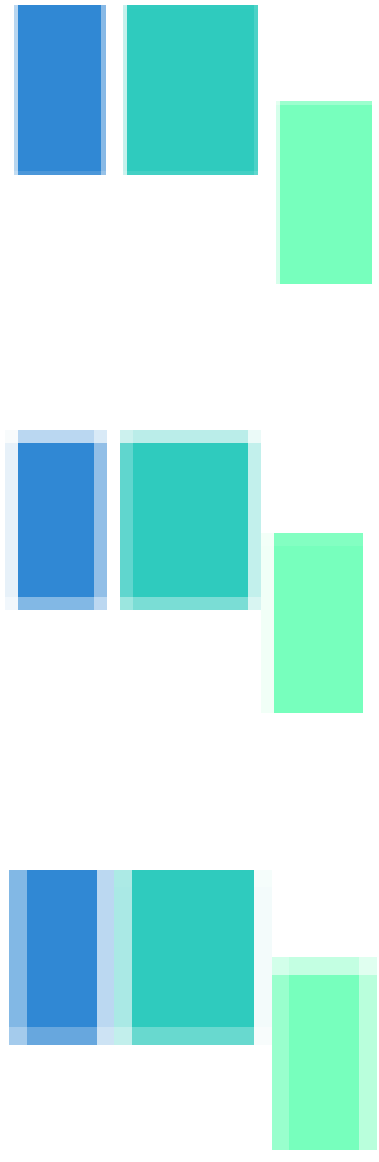
Light

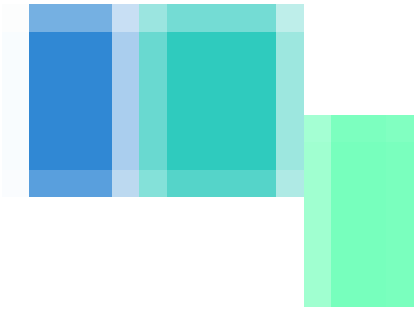
SVG

PNG









7.9.2 Stickers

Feel free to print these out and spread the love!

The stickers are contributed by lostinlight, licensed under CC-BY-4.0.

PDF's available here.

7.10 FAQ

Generic questions regarding Socialhome.

7.10.1 If I run an instance, will it automatically connect to the network?

No, Socialhome doesn't work in a peer2peer sense. Incoming content does not happen automatically, outbound does in a limited sense. Federation happens on two main principles:

- Social relationships

  This means content you create will be sent to the servers that your followers are on. Content that users you follow will send content they create to your server. The more inter-server relationships users on the server you are on, the more content will reach your server. This is the standard federation model in use by the Diaspora, OStatus and ActivityPub powered networks.

- The public content relay system

  The relay system is a network of servers that exist for one single purpose, to receive public content and to distribute it to places it would not otherwise reach (with social relationships based federation). Socialhome by default integrates with the relay system by sending all public content to it and subscribing to all public content on the relays. To receive content, this requires registering the Socialhome server to the relays so that they know about the server and can start sending content.

  Registration happens at The-Federation.info. For more technical details about the relay system, check the reference server documentation.

As a recap:

- To receive content from around the network, a new server can do the following:
  - Following other users on other servers to create social relationships. This happens by using the full handle (for example `hq@socialhome.network`) of the user.
  - Registering to the relay system.

- To send out content to the network, a new server can do the following:
  - Following other users on other servers to create social relationships. Public content will also be sent out to the servers whose users *you* follow, not just who follow you.
  - No need to register with the relay. All public content created on a Socialhome server is by default sent out to the relay system.

7.10.2 Welp I found a security issue, shall I just file an issue?

No, please report security issues directly to the maintainers in private messages. Reporting security issues publicly in the issue tracker would allow other people to use the security issue to their benefit before a fix is released. Reporting security issues to the maintainers in private allows us to fix the issue and roll out a fix hopefully before many users are affected.

Please report security issues directly to the maintainer by email at `mail@jasonrobinson.me`.

7.10.3 As admin, how do I delete a local user or block a remote spambot?

Please see *Deleting users and locking remote profiles* section for instructions.

7.10.4 I accidentally shared someones content and it's stuck on my profile stream!

This is a known issue with stream caching. Until it is fixed, ask your instance admin to manually clear your profile cache. It can be done with the following command, on the machine where Redis is running.

```
redis-cli --scan --pattern 'sh:streams:profile_all:<profile.id>:*' | xargs redis-cli␣
↪del
```

Where `<profile.id>` should be replaced by the profile ID found via the Socialhome admin view.

7.11 Changelog

7.11.1 0.12.0 (2019-12-12)

Added

- The Content API now has full support for all visibility levels, including specifying recipients for limited content.
- The new publisher written in Vue is now feature complete with the current one. If you would like to try it out, go to your account settings and enable the new publisher. After a short trial period, it will become default and the old Django template code will be removed.
- Admins can now disable the Django admin error emails by setting the environment value `SOCIALHOME_LOG_DISABLE_ADMIN_EMAILS=True`.
- Outbound payloads can now also be logged by toggling the relevant preference in the admin. When enabled, all outbound payloads will be saved for debugging purposes in the "Payloads" model and available via the admin.
- Fetching of unknown remote content using ActivityPub URL id's now works. Just paste the remote ActivityPub content URL to the search bar to fetch it from compatible platforms.
- Add a scheduled task to clean up old scheduled jobs in Redis.

- Added Django-Redisboard. This gives admins some visibility into the way Socialhome uses Redis via the admin pages.

Changed

- Content interaction actions and icons have been fully redesigned to improve readability and to make it easier to interact with other peoples content. (#574)
  - Root level content now has "reply" and "share" actions more clearly on the left hand side of the content interactions area below the content text.
  - The new shares action now immediately shares (or unshares, if shared) instead of requiring a second action click.
  - The shares counter no longer acts as a share action.
  - Both the reply action and the replies counter expand the replies container.
  - Each reply now has a reply action as well. This is located on the right hand side of the content interactions area below the reply. This allows users to target their reply to a particular reply.
  - Replies now automatically insert mentions into the reply editor. For replies on root content, the root content author is mentioned. For replies on replies, both the root content author and the replied reply author are mentioned. The mentions can of course be removed.

  Hopefully these changes will make not only interacting easier, but also make interactions from Socialhome more compatible with other platforms like the microblogging side of the Fediverse. Feedback on these is most welcome!
- Improve rendering of outgoing mentions towards the ActivityPub network. (#572)

  Mentions from Socialhome now get rendered as links in outbound HTML format payloads.
- Render URL's in outgoing payloads towards the ActivityPub network as proper links. (#572)
- Make link preview images larger and ensure images have a title attribute.
- Stop rendering link previews for HTML style mentions that come from ActivityPub networks.
- Truncate link preview description to max 500 characters.
- Add a truncated link preview url to the link preview card.
- Don't render link preview image if the same image is already in the content text.
- OEmbed for Twitter profile streams is now skipped. Only single tweets will be embedded.
- Whitelist some more HTML tags for use in formatting content. All the tags that are purely for visual formatting or structuring (like div, table, dd) are now whitelisted.
- Totally rewritten publisher! The new publisher is now a part of the VueJS based frontend code and is based on the EasyMDE editor. There are some additional features like full-screen mode added, for those long blog post type content pieces. Thanks to Christophe Henry for work on this.

Fixed

- Fix regression in Diaspora follows introduced in 0.11.0. Due to stricter validation that was added to outbound payload sending, follows to the Diaspora protocol side broke in 0.11.0 (from 11th of October in the development branch). All follows/unfollows during this period towards remote Diaspora protocol profiles have failed and should be retriggered.

- Don't crash loudly incompatible content is fetched via the Diaspora protocol fetch view and a document fails to validate.
- Fix an issue in the OEmbed library that caused unnecessary fetches to Spotify OEmbed endpoint. Thanks to Alain for reporting this issue. (upstream issue)
- Fix rendering of quotes. Now rendered in italic and clearly marked as quote.
- Stop creating duplicate content items from remote content delivered by both ActivityPub and Diaspora protocols. When these refer to each other, they will be merged to avoid duplicate content items.

Internal changes

- Store an Activity on follow/unfollow. This allows retriggering follows/unfollows in the event of a regression.
- Move `socialhome.content.utils.process_text_links` to the `federation` library.
- Move the tags extraction logic from `Content` model to the `federation` library.

7.11.2 0.11.1 (2019-12-30)

Security

- Upgrade Django to fix CVE-2019-19844.

Fixed

- Support tag streams for non-ascii tags. (#517)

  Tags which fail to generate a slugified name (ie for example Russian alphabet tags) are now supported as streams. This also fixes the search internal server error when searching for a tag which fails to generate a stream URL.

7.11.3 0.11.0 (2019-12-15)

Added

- Searching of users on the ActivityPub protocol now works using a handle (ie `user@domain.tld`).
- Shared content in the streams now contain the name and link to the profile of the sharer.
- Django-Silk is now available for developers to turn on for their instance.

  To turn on, set the environment variable `SOCIALHOME_SILKY=True`.
- Users API now has an admin endpoint to return recently active users.

Changed

- UI changes:
  - The stamped element (the first stream element with information about the stream or profile information) is now 100% wide in all situations. The profile picture has also been made larger.
  - The author bar has been moved from the bottom of the content to the top of the content.

  - Clicking a profile name in the author bar now pops up the author federation ID and reaction buttons instead of expanding them. This saves having to re-render the whole stream grid.

Fixed

- All streams now properly push out websocket notifications on new content.

  Previously only certain streams knew how to push notifications on new content to the client. Now all streams will know how to do this. Additionally they will respect user preferences in the future when hiding of content from users is added. Shared content also gets a notification pushed out as other content.
- Don't process received shares if they point to a non-public content.
- Don't show a share icon for own content, unless there is a counter to show.
- Fix follow/unfollow of profiles from the search page.
- Don't raise a 500 error when Diaspora remotes try to fetch a content whose author has no `handle`.
- Don't raise a 500 error when an attempt is made to view a profile with an invalid profile identifier
- Fix a major race issue with the `through` value calculation for shared content in streams (#558)

  When calculating `through` values (ie what share caused a content to appear in the stream), there was a race condition between processing the saved share and a remote fetched shared content. Values are now correctly calculates irregardless of saving order to provide correct "shared by" information for streams.
- Don't raise a 500 error on fetch of content using a malformed identifier
- Fix inbound federation timing issue with ActivityPub platforms (#563)

  Signature verification time delta check if a background worker didn't process the inbound payload fast enough, which led to rejected payloads. Time delta check has now been relaxed to allow at most 24 hour old signatures.
- Improve performance of profile streams and fetching of replies by splitting the database queries into multiple queries instead of one larger one. (#562)

API changes

- **Backwards incompatible**: Removed duplicated `user_following_author` from the Content API since it is included in the serialized `author` as `user_following`.
- Stream API results now contain a `through_author` object in the case that the content is in the stream via share.
- The Stream API endpoints now accept an *accept_ids* query parameter, which should be a list of content ID's to fetch from this particular stream. This allows filling the stream with new items in the stream context without making multiple fetches.
- Replaced deprecated `django-rest-swagger` API docs module with `drf-yasg`. The new module provides not only Swagger but also ReDoc API docs. We've chosen ReDoc for the default docs mounted at `/api/` on each instance. Swagger is still available at `/api/swagger/`. (#537)

Internal changes

- Django Channels upgraded from 1.x to 2.x version. This freed up various pinned dependencies like Redis and RQ to be upgraded to their latest versions. This also makes it unnecessary to run a Channels worker process as before. That has already been removed from the `circus.ini` file. If you run the processes manually, the process does not need executing any more.

7.11.4 0.10.0 (2019-10-06)

Added

- Initial ActivityPub support has landed!

  Basic federation support with the ActivityPub protocol. There are likely to be many bugs and incompatibilities with this first release which will be ironed out in the next few releases.

  Importantly, Socialhome defaults to ActivityPub should a remote profile support both ActivityPub and Diaspora protocols. This means federation across Socialhome instances will use ActivityPub.

  Considerable effort was put into refactoring Socialhome internals to work with multiple protocols. This refactoring affects mostly the internals of Socialhome with only minor visual changes. Some of those include changes in URL's and fallback display names for non-local profiles.
- Added Tags API. In addition to listing Tag objects, it allows authenticated users to follow and unfollow tags.
- Profile API now includes a list of tags followed for logged in users.
- It is now possible to follow and unfollow tags from a tag stream (#465)

  Content from followed tags is available under a new "Tags" stream.
- It's now possible to disable incoming relay system integration by setting the environment variable `SOCIALHOME_RELAY_SCOPE` to `none`. (#94)
- User profile now has a link to a new followers contacts page. This is limited to the logged in user only.
- Added a model for federation payloads for debugging purposes. If the "log all federation payloads" admin setting is on, incoming payloads will now also be available in the database via the admin pages, in addition to the log file.

Changed

- **Backwards incompatible**. Python 3.6 is now the lowest supported Python version. Please do not try to upgrade Socialhome to this release before updating your Python virtualenv, if running an older Python!
- Code repository moved to the Feneas GitLab which offers a richer set of features compared to GitHub. What is Feneas? Check this post.

  Code will still be mirrored to GitHub so participation through there is still very much welcome. So you can still fork the repository on GitHub and submit a pull request. Issues however will be available only on the GitLab server to avoid these getting out of sync between the servers.
- The behaviour of the `delete_users_and_profiles` management command has been changed to lock remote profiles by default instead of deleting them. This is more efficient for spam control as deleted profiles just appear back. The lock option can still be set as false to delete the profile which is a good option for example when cleaning data of remote profiles on request.
- **Breaking change**. API changes. (#451)
  - Profile API has been migrated to use UUID's instead of ID's.

    All API endpoints will be receiving this change which is done while the API has a limited number of consumers and will be one of the last planned breaking changes planned before a 1.0 API can be announced.
  - Profile API following and unfollowing endpoints have changed.

    The action `add_follower` has been renamed to `follow` and `remove_follower` to `unfollow`. The change reflects the change of the endpoints themselves. Now to add a follower one does a POST to

the `follow` of the profile that one wants to follow, instead of doing a POST to the `add_follower` endpoint of ones own profile. Same change has been done for the follower removal endpoint.

  - Streams profile API's have moved to using UUID's instead of ID's.

- Make profile default visibility public (#515)

  This fixes an issue where new profiles can follow others but the others cannot follow back, since the default was SELF. For now, make all new profiles public by default. Later the visibility setting should be moved from the profile to profile fields themselves. Some core identity will always need to be public but what the profile shares could be controlled.

  Also make remote profiles always locally public to avoid situations where a user can see the post but can't see the local profile. Any profile that federates to us is public to some extent since it left the server.

- Global search now works also with ActivityPub ID's to fetch a remote profile
- Added a few additional HTML elements to content cleanup whitelist: tags *span, p, br* and attribute *class* on *span*.

Fixed

- API docs regression fixed (#509)
- Fix internal server error for anonymous user for certain internal user pages (#518)
- Timeout of the pre-calculated stream cache cleanup job has been extended so that it doesn't timeout on larger servers.
- Nested replies are now correctly shown as replies to the root level content instead of being hidden from view.
- Removed unnecessary federation of replies to local root authors.
- Removed quick reply possibility for non-public content. This fixes an issue of non-public replies created with the quick reply editor not federating.

  While the API support is being added needed by the quick reply editor for non-public content, only the full editor can be used for non-public replies.

- Fix retraction of limited visibility content sent out to the federation layer.

  There was a bug where limited visibility content (added in 0.9.0) retractions were not sent out correctly. This was caused by the usage of the Django `post_delete` signal to handle the retraction. This works for public content since all the information is present immediately after the delete for the background jobs, even if the database entry has been deleted. Unfortunately for limited content this did not work since they store visibilities to the limited content in a separate table. Due to the (awesome!) way Django relations work, the entries for the visibilities got deleted from the database before the `post_delete` signal got fired.

  Content retraction is now fired off into a background task in the Django `pre_delete` hook, which means the limited visibilities data is still available in the database.

Internal changes

- Removed `User` relationship fields. These were migrated to `Profile` a long time ago.
- Heavy refactoring of Vue frontend store.
- Django bumped to 2.2.

7.11.5 0.9.3 (2018-08-29)

Fixed

- Update `pycryptodome` due to CVE-2018-15560 security issue.

7.11.6 0.9.2 (2018-08-11)

Fixed

- Update to `federation` which switches crypto libraries to fix CVE-2018-6594.

  Note! If you don't use `pip-sync` to deploy, then you **must** do `pip uninstall pycrypto` before deploying, or things will break badly.

7.11.7 0.9.1 (2018-08-11)

Fixed

- Django bumped to 2.0.8 to fix a security issue. This issue did not affect Socialhome, but we're upgrading just to be sure.

7.11.8 0.9.0 (2018-07-21)

Added

- Add possibility to configure Sentry for error reporting.

  Adding the Sentry project DSN as `SENTRY_DSN=foo` to environment variables will make all error level exceptions be raised to Sentry. To change the level, define `SENTRY_LEVEL` with a valid Python logging module level.

- Add NodeInfo2 support. For organization details, admin name and email will be published if the new setting `SOCIALHOME_SHOW_ADMINS` is set to `True` (default `False`).

- Add possibility to delete user account (#131)

  Deletion is permanent and will delete all created content including uploaded images. Delete request for profile and related content will be sent to remote servers.

- Add user export API (#478)

  New API endpoints `/api/profiles/create_export/` will create an export and `/api/profiles/retrieve_export/` will retrieve the export zip file. Export will contain a JSON file of the user, profile, followers and content. A zip file of uploaded images will also be included.

- Add user data export to user account page (#478)

  The account page now has a button to request an export of user data. In addition to user and profile data, this export contains a list of profiles followed, content (including shares and replies) and a zip file of image uploads. An email notification will be sent to the user once the export is ready for download from the account page.

- New environment variable `DJANGO_TIMEZONE` allows easily customizing the time zone that the Socialhome instance runs on. It defaults to `UTC`.

- Staff users can now access the admin and task queue (background jobs) pages via the new "gears" menu in the navbar. See <documentation on how to make a user admin.
- Add an easily customizable `robots.txt` with default rules

  The rules by default disallow all except direct links to content, the root profile and the public stream. Server admins can customize the rules easily via the admin interface.
- Admins can now add Terms of Service and Privacy Policy documents to the site (#477)

  Terms of Service and Privacy policy documents are good to have. These tell people visiting your site what rules you operate with. Socialhome provides default templates you can activate with a few clicks.

  To review and enable the policy documents, log in as admin and access the admin pages through the navigation bar cogs menu. Scroll down and locate "Policy documents". There are two types of documents, the Terms of Service and Privacy Policy. Each one can be edited in draft mode and then published. Further updates in draft mode will not overwrite the last published version, until published.

  To publish the documents, open them, review the text and then change the status below the document to "published". Click Save - this version is now published. To edit in draft mode, switch the status back and the current edited revision will not show to users. You can also send email updates to users from the policy documents list. Select the policy documents you wish the send an email about, choose "Send email" from the actions list and confirm.

  Published policy documents are shown to both authenticated and unauthenticated users via the navigation bar cogs menu.
- Searching for hashtags is now possible using the global search

  The global search now in addition to profile results returns also results of matching hashtags. If the search term includes the hash ('#') and matches exactly to a tag, an instant redirect will be made to the tag stream.
- Mentions are now parsed out of incoming remote content and locally created content.

  Currently the only syntax supported is the Diaspora mentions syntax, ie `@{Name; user@domain.tld}`. Currently Socialhome users can create mentions by using the syntax manually. UI layer will be added later to choose people using the standard @ syntax to trigger search.

  When mentioned, local users will be sent an email notification with a link to the content.

  Note to admins: A script is provided if you want to parse old content for mentions. Run `./manage.py runscript link_old_mentions` if you wish to parse the content from the last year and create the links. This will also send out email notifications.
- Admin now has a section for Content items and Profiles, for debugging purposes. The User admin was also improved.
- Limited content is now supported (#302)

  Limited content can now be created using the web create form. Note, API does not currently allow creating limited content (except replies to limited content). Once create form is ported to the API, things should be refactored there, right now had no bandwidth to ensure both work.

  Limited content is shown in the stream with a lock symbol. The create shows some extra fields for limited content. These include "recipients" and "include following". Recipients is a comma separated list of target profile handles the limited content will be sent to. Include following will populate recipients (on save) with all the profiles that one follows. Later on we will add contact lists for better targeting.

  Limited content visibilities can be edited. If someone is removed from the target recipients, a retraction will be sent to try and delete the content remotely from the target recipient.

Currently recipients must already be known to the server, in the future a remote search will be done if the profile is not known. Any known remote profile can be targeted - it is up to the receiving server to decide whether to accept it or not. For local profiles, those of visibility SELF (ie hidden) cannot be targeted.

There is also a new stream "Limited" available. It shows all limited content visible to you.

- Add "Local" stream which contains only content from users registered on the same server. (#491)

Changed

- Bump Django to 2.0 (#460)
- Only precache for users who have been active (#436)

  Don't precache items into streams for users who have not been active. Controlled by the same settings as the maintenance of precached streams. Will reduce unnecessary background jobs and make Redis memory usage even more stable.
- Provided Circus configuration now ensures RQ worker processes are not allowed to endlessly hog server memory. In some rare cases it has happened that normally very stable RQ worker processes have hogged several gigabytes of memory due to reasons which are still being investigated. Now Circus will end those processes automatically.
- Moved user account, logout, email management and API token pages links under the new "gears" menu in the navbar. These links used to be in the profile page menu.

Fixed

- Allow search with Diaspora handle that contains port (#457)
- **Important for server admins**. There was a mistake in the production Redis connection settings. The setting was not following the given configuration in the documentation. Now the possibility to set `REDIS_URL` (undocumented) directly has been removed and will raise an error. Use the `REDIS_HOST`, `REDIS_DB`, `REDIS_PORT` and `REDIS_PASSWORD` settings instead when needed.
- Ensure all streams Redis keys have a default expiry of 30 days.
- Fix parsing of remote profile names by also using `last_name` attribute, where given (#414)
- Show possible validation errors on create form instead of just not allowing a save.
- Fix failure of processing remote retractions of replies or shares in some situations.

Removed

- Legacy streams routes */public/*, */followed/* and */tags/<name>/* have been removed. They already partially broke in the Vue.js streams rewrite.

7.11.9 0.8.0 (2018-03-06)

Added

- RFC7033 webfinger support for Diaspora protocol (#405)

  This allows better profile discovery by remote non-Socialhome servers.

- Added better streams precache maintenance in regards to inactive users (#436)

  Two new settings have been added:

  - `SOCIALHOME_STREAMS_PRECACHE_INACTIVE_DAYS` (default 90)
  - `SOCIALHOME_STREAMS_PRECACHE_INACTIVE_SIZE` (default 0)

  If a user has been more than the set days without logging in, when trimming the precaches for that user, the inactive setting will be used instead. By default this means that precaches for users that haven't logged in for 90 days are removed. This is done to ensure Redis memory usage is predictable and stable in relation to active users.

  Users who have been inactive for longer than the X days will still get their stream content normally but instead of getting a fast stream render from the cache, the items will be calculated using databse queries, which produces a slower stream load experience.

- Added management command to delete local users and remote profiles

  This allows removing users who want their account to be deleted (coming to UI soon, sorry) and also deleting content and locking remote spam accounts. See documentation for details.

Changed

- Setting `SOCIALHOME_RELAY_DOMAIN` is now called `SOCIALHOME_RELAY_ID`. We're slowly replacing all direct Diaspora handle references with Diaspora URI format profile ID's in preparation for ActivityPub protocol addition.

  No action needed from server admins unless you have changed this setting, in which case it should be updated accordingly.

- Start sending profile changes to remote nodes as public messages for better efficiency
- Start sending federation payloads in new format (federation #59)

  This could drop federation compatibility with some really old servers in the fediverse, but adds compatibility to for example GangGo which is now able to receive Socialhome content.

- Stop requesting Twitter widget script for each tweet OEmbed (#202)

  Since Vue streams all tweets are initialized programmatically as they are rendered in the stream so we don't need to have the script tag on each oembed separately.

- `/api-token-auth/` endpoint now returns limited profile information in addition to token

Fixed

- Fix precached streams maintenance job. (#436)

  Due to mistake in regexp not all old precached stream items were pruned in maintenance. Now fixed which should ensure Redis memory usage does not suffer from unreasonable increase over time.

- Fix profile discovery from current stable Diaspora (#413)

  A bug in Diaspora caused Socialhome profile discovery to fail. Introduce some patches to our webfinger to work around the bug and make profiles available to latest Diaspora versions.

- Fix receiving public content from GangGo (federation #115)
- Fix various errors in search for remote profiles

  For example GNU Social implements webfinger but the necessary attributes we need are not present and were causing errors.

- Add missing titles and OG tags back to streams (#428)

  These disappeared in the rewrite of streams in 0.7.0. Also added a few new head tags improving author information in single content view and telling Twitter to not track users so much.

7.11.10 0.7.0 (2018-02-04)

New Vue.js frontend

The work that started at a small hackathon in Helsinki in July 2017 is finally finished! The old buggy and hard to maintain Django template + jQuery based frontend has been completely rewritten in Vue.js. This provides a modern frontend code base, making it possible to add new features faster and to spend less time fixing bugs in the spaghetti code.

A huge thanks goes out to @christophehenry doing most of the work in pushing this rewrite through!

Added

- Possibility to skip adding an OEmbed or OpenGraph preview to content. (#364)

  There is a new checkbox on content create that allows skipping adding a link preview to the content.

- Add maintenance job to groom precache information from Redis. This ensures Redis memory usage stays stable.

  Important for server admins. There is a new process to run that is responsible for scheduling these maintenance jobs. The

  - If you already use the provided Circus configuration to run Socialhome, you **don't need to do anything**. When you restart Socialhome, the updated Circus configuration will automatically be used and the scheduler process started by Circus.
  - If you have a custom setup, preferring to run all processes manually, ensure one `rqscheduler` process is running at all times to ensure maintenance jobs and other future scheduled jobs are executed.

  A new configuration item `SOCIALHOME_STREAMS_PRECACHE_SIZE` is available to set the maximum size of precached stream items per user, per stream. This defaults to 100 items. Increasing this setting can radically increase Redis memory usage. If you have a lot of users, you might consider decreasing this setting if Redis memory usage climbs up too high.

- It is now possible to use email for log-in. (#377)
- Added a Code of Conduct document. All contributors to Socialhome are expected to honour these simple rules to ensure our project is a safe place to contribute to.

  Read the Code of Conduct here.

- Profile API has 4 new read only fields:
  - `followers_count` - Count of followers the given Profile has. For remote profiles this will contain only the count of followers on this server, not all the followers the profile has.
  - `following_count` - Count of local and remote profiles this Profile is following. For remote profiles this will contain only the count of profiles following this profile on this particular server.
  - `has_pinned_content` - Boolean indication whether the local profile has pinned any Content to their profile stream. Always false for remote profiles.
  - `user_following` - Boolean whether logged in user is following the profile.

- There is now a management command to generate dummy content for development environment purposes. See *Development* pages.
- Installation docs now have an example SystemD service configuration, see *Other Linuxes or newer Ubuntu using SystemD*. (#397)
- Content API has a new read only field `has_twitter_oembed`. This is `true` if the content text had a Tweet URL *and* a fetch for the OEmbed code has been successfully made.
- Content create page now has an option to disable federating to remote servers when saving the content. (#296)

  The content will still update to local streams normally. Federating the content can be enabled on further saves.
- If signups are closed, the signup link will now stay active but will point to a list of Socialhome nodes. (#354)

  By default this URL is https://the-federation.info/socialhome, but can be configured by the server admin.

Changed

- When processing a remote share of local content, deliver it also to all participants in the original shared content and also to all personal followers. (#206)
- Allow creating replies via the Content API.

  Replies are created by simply passing in a `parent` with the ID value of the target Content. It is not possible to change the `parent` value for an existing reply or root level Content object once created. When creating a reply, you can omit `visibility` from the sent data. Visibility will be used from the parent Content item automatically.
- Removed Opbeat integration related configuration. The service is being ramped down. (#393)

  If as a server administrator you have enabled Opbeat monitoring, it will stop working on this update.
- New VueJS stream is now default o/ (#202)

  Old stream can still be accessed using the user preferences or by passing a *vue=0* parameter in the URL. All existing users have been migrated to use the new VueJS streams by default.

Fixed

- Redirect back to profile instead of home view after organize pinned content save action. (#313)
- Fix searching of an unknown remote profile by handle using uppercase letters resulting in an invalid local profile creation.
- Fix Content querysets not correctly including the 'through' information which tells what content caused a share to be added to a stream. (#412)

  This information was already correctly added in the streams precalculation phase, but if the cache started cold or a viewing user cycled through all cached content ID's and wanted some more, the database queries did not return the right results.
- Attempt to fetch OEmbed and OpenGraph previews of URL's in content in the order of the links found. (#365)

  Previous behaviour lead to fetching previews of urls in random order, leading to a different url preview on different Socialhome servers.
- Fix remote profile retrieval from remote servers which don't support legacy Diaspora protocol webfinger. (#405)

  New version of federation library defaults to trying the new style webfinger with a fall back to legacy.

7.11.11 0.6.0 (2017-11-13)

Added

- Profile "All content" streams now include the shares the profile has done. (#206)
- Streams API now has endpoints for profile streams to match the profile streams in the UI. (#194)
  - `/api/streams/profile-all/{id}/` - fetches all content by the given profile (including shares), ordered by created date in reverse order (= new stuff first).
  - `/api/streams/profile-pinned/{id}/` - fetches pinned content by the given profile, ordered as set by the profile owner.
- New fields added to Content API:
  - `is_nsfw`, boolean value, `true` if the content text has the tag `#nsfw` in it.
  - `share_of`, if the `content_type` is `share`, this will contain the ID of the shared Content.
- If an incoming share references a remote target that doesn't yet exist locally, it and the author profile will be fetched and imported over the network. (#206)
- There are now Docker files for doing development work for Socialhome. See the docs here.
- Third-party applications can now be added to enhance Socialhome or replace some of the core functionality, using configuration. The following new settings are available:
  - `SOCIALHOME_ADDITIONAL_APPS` - List of additional applications to use in Django settings.
  - `SOCIALHOME_ADDITIONAL_APPS_URLS` - Additional third-party URL's to add to core url configuration.
  - `SOCIALHOME_HOME_VIEW` - Override the home view with another view defined with this setting.
- Content API now has a new `shares` endpoint. (#206)

  This allows retrieving all the shares done on a Content.
- We now have a logo

  The logo also comes in a light version, for dark backgrounds. See *Brand* for details.

Changed

- Logging configuration changes:
  - Removed separate logfile for the federation loggers. Now all logs go to one place. Setting `SOCIALHOME_LOGFILE_FEDERATION` has been removed.
  - Added possibility to direct Django and application logs using a defined level to syslog. Adds three settings, `SOCIALHOME_LOG_TARGET` to define whether to log to file or syslog, `SOCIALHOME_SYSLOG_LEVEL` to define the level of syslog logging and `SOCIALHOME_SYSLOG_FACILITY` to define the syslog logging facility. See configuration documentation.
- **Important!** The file to place configuration environment variables has changed to `.env`.

  This is a more standard file name for environment variables than the previous `env.local`. For now we'll still load from the old file too, but a warning will be displayed to rename the file.
- **Breaking change**. API `Content` serialization now returns list of tags as *name of tag*, not ID as before. The names do not contain the character "#".
- Content API `replies` endpoint now includes all the replies on the shares of the Content too.
- Use modified timestamp for created timestamp when federating out to remote nodes. (#314)

  This makes edits federate more reliably to some remote platforms that support edits.
- Stream grid item reply icon changed from "envelope" to "comments". (#339)

Fixed

- Fix various issues with OpenGraph tags parsing by switching to self-maintained fork of `python-opengraph`.
- Share button is no longer visible if not signed in (#325)
- Remote profile image urls that are relative are now fixed to be absolute when importing the profile from remote (#327)
- Fix poor performance of fetching replies.

  When adding replies of shares to the collection of replies fetched when clicking the reply icon in the UI, a serious performance regression was also added. Database queries have now been optimized to fetch replies faster again.
- When editing a reply, the user is now redirected back to the parent content detail view instead of going to the reply detail view. (#315)
- Fix regression on visibility of remote replies on shares.

  Replies inherit the parent object visibility and share visibility defaults to non-public in the federation library. Diaspora protocol removed the `public` property from shares in a recent release, which meant that we started getting all shares as non-public from the federation layer. This meant that all comments on the shares were processed as non-public too.

  With a change in the federation layer, Diaspora protocol shares are now public by default.
- Fixed Streams API content `user_is_author` value always having `false` value.

7.11.12 0.5.0 (2017-10-01)

Python dependencies

Switched to `pip-tools` as the recommended way to install Python dependencies and cleaned the requirements files a bit. Now all the "base" dependencies, including production deployment dependencies are locked in `requirements.txt`. The new file `dev-requirements.txt` includes both the base and the extra development/testing related dependencies.

To use `pip-tools`, first install it:

```
pip install -U pip-tools
```

Then install dependencies:

```
# Production environment
pip-sync

# Development environment
pip-sync dev-requirements.txt
```

It is not mandatory to use `pip-tools` for running a production installation. For development it is mandatory. All dependencies should be placed (unlocked) in either `requirements/requirements.in` (base) or `requirements/requirements-dev.in` (development extras). Then execute `./compile-requirements.sh` to update the locked dependency files after each change to the `.in` files. See pip-tools for more information.

Added

- GIF uploads are now possible when creating content or replies. (#125)
- Content API has a new endpoint `/api/content/<id>/replies/`. This returns all the replies for the given content.
- Shares made by followed contacts are now pulled up to the "Followed" stream.

  This happens only if the user has not already seen this content in their "Followed" stream. Each content should only appear once, either directly by following the author or a followed contact sharing the content. Multiple shares do not raise the content in the stream again.

Changed

- Rendered link processing has been rewritten. This fixes issues with some links not being linkified when rendering. Additionally now all external links are made to open in a new tab or window. (#197)
- Previously previews and oEmbed's for content used to only pick up "orphan" links from the content text. This meant that if there was a Markdown or HTML link, there would be no link preview or oEmbed fetched. This has now been changed. All links found in the content will be considered for preview and oEmbed. The first link to return a preview or oEmbed will be used.
- Streams URL changes:
  - All streams will now be under `/streams/` for a cleaner URL layout. So for example `/public/` is now `/streams/public/`.
  - Tag stream URL has been changed from `/streams/tags/<tag>/` to `/streams/tag/<tag>/`. This small change allows us to later map `/stream/tags/` to the tags the user is following.

  Since lots of old content will point to the old URL's, there will be support for the legacy URL's until they are needed for something else in the future.

- **Breaking change**. Profile API field changes:
  - Added:
    * `url` (Full URL of local profile)
    * `home_url` (Full URL of remote profile, if remote user)
    * `is_local` (Boolean, is user local)
    * `visibility` (Profile visibility setting, either `public`, `limited`, `site` or `self`. Editable to self)
  - Removed (internal attributes unnecessary for frontend rendering):
    * `user`
    * `rsa_public_key`
- **Breaking change**. Content API field changes:
  - Added:
    * `timestamp` (ISO 8601 formatted timestamp of last save)
    * `humanized_timestamp` (For example "2 hours ago")
    * `url` (Full URL to content detail)
    * `edited` (Boolean whether content has been edited since creation)
    * `user_following_author` (Boolean whether current user is following content author)
    * `user_is_author` (Boolean whether current user is the author of the content)
    * `user_has_shared` (Boolean whether current user has shared the content)
  - Changed:
    * `author` is now a limited serialization of the author profile, containing the following keys: "guid", "handle", "home_url", "id", "image_url_small", "is_local", "name", "url".

      The reason for serializing the author information to content is related to privacy controls. A user who maintains a limited profile can still create public content, for example. A user who is able to view the content created by the user should also see some limited information about the creating profile. To get the full profile, the user needs to fetch the profile object by ID, which is subject to the visibility set by the profile owner.
  - Removed (internal attributes unnecessary for frontend rendering):
    * `created`
    * `modified`
    * `oembed`
    * `opengraph`
- Refactoring for streams views to use new Stream classes which support pre-caching of content ID's. No visible changes to user experience except a faster "Followed users" stream.

  A stream class that is set as cached will store into Redis a list of content ID's for each user who would normally see that content in the stream. This allows pulling content out of the database very fast. If the stream is not cached or does not have cached content ID's, normal database lookups will be used.

  This refactoring enables creating more complex streams which require heavier calculations to decide whether a content item should be in a stream or not.

Fixed

- Cycling browser tabs with CTRL-TAB when focused on the editor no longer inserts a TAB character in the editor.
- Don't federate shares to shared content local author. This caused unnecessary deliveries between the same host.

7.11.13 0.4.0 (2017-08-31)

Update notes

This release contains long running migrations. Please allow up to 10 minutes for the migrations to run, depending on your database size.

Added

- Allow user to change profile picture. (#151)

  Profile menu now has an extra option "Change picture". This allows uploading a new picture and optionally setting focus point for cropping a picture that is not square shape.
- Federate local profiles to remote followers on save. (#168)
- Process remote profiles entities on receive.

  Remote profiles were so far only created on first encounter. Now we also process incoming `Profile` entities from the federation layer.
- When following a remote profile, federate profile to them at the same time.
- It is now possible to expose statistics from a Socialhome node. This includes counts for users (total, 30 day, 6 month), local content and local replies. These will be exposed via the `NodeInfo` documents that for example the-federation.info node list consumes.

  By default statistics is off. Admins can switch the counts on by setting environment variable `SOCIALHOME_STATISTICS=True` and restarting Socialhome.
- Add user API token view. Allows retrieving an API token for usage in clients and tools. Allows also regenerating the token if it has been lost or exposed.
- Added bookmarklet to easily share external pages. The bookmarklet can be bookmarked from the 'Create' page. (#138)

  Sharing with the bookmarklet will copy the page url, title and optionally selected text into the create content text area. The bookmarklet is compatible with Diaspora, so for example the Firefox sharing service will work.
- Support receiving 'Share' entities. Show amount of shares on content. (#206)
- Show replies to shares on the original shared content. (#206)
- Add `share` endpoint to Content API. This enables creating and removing shares via the API. (#206)
- Allow sharing content. Clicking the share counter icon exposes a 'Share' button which when clicked will create a share. (#206)
- Allow unsharing content. Clicking the share counter icon exposes an 'Unshare' button (assuming the user has shared the content) which when clicked will remove the share. (#206)
- Federate local shares to remote nodes. (#206)

- There is now a 'My content' stream link in the navbar 'Streams' dropdown. This goes to your own profile all content stream.
- Add user preference for the new stream refactoring. If enabled, all streams that have a new version in progress will be rendered with the new frontend code based on Vue.js. (#202)

  Warning! The new frontent code doesn't have all the features of the current on yet.
- Content API has three new read only fields available:
  - `local`, boolean whether the content is local or remote.
  - `reply_count`, count of replies (including replies on shares)
  - `shares_count`, count of shares
- Make email notifications nicer by using HTML templates in addition to the plain text version. (#206)

  In addition to reply and follow notifications, send also when own content is shared.

Changed

- **Breaking change**. Content API results now return `visibility` as a string ('public', 'limited', 'site' or 'self'), not an integer.

Fixed

- There was no notification sent out when a local user followed a local user. This has now been fixed.

Removed

- **Breaking change**. Removed Content, Profile and Users API LIST routes. For now these are seen as not required for building a client and allow unnecessarily easy data mining.
- Removed content modal. Clicking timestamp in grid now directly loads the content detail view. (#162)

  Loading the content in a modal was an early experiment and didn't end out very usable.
- Removed reply button from replies. Technically, threaded replies are possible but the UI implementation is not done. Replying to a reply will be back once UI and federation layer will handle threaded replies properly.

7.11.14 0.3.1 (2017-08-06)

Fixed

- Bump `federation` library again to fix a regression in reply relaying due to security fixes in the library 0.14.0 release.

7.11.15 0.3.0 (2017-08-06)

Security

- Reject remote content updates via the federation layer which reference an already existing remote content object but have a different author.

Note that locally created content was previously safe from this kind of takeover. This, even though serious, affects only remote created content stored locally.

- Reject remote reply updates via the federation layer which try to change the parent content reference.
- Bump federation to ensure remote entity authorship is verified correctly.

Added

- API has two new endpoints, the "Content" and "Image Upload" routes. (#120)
  - Content API allows browsing content objects that are visible to self, or public for anonymous users. Content objects owned by self can be updated or deleted. Creating content is also possible.
  - Image Upload API allows uploading images via the same mechanism that is used in the content create UI form. The uploaded image will be stored and a markdown string is passed back which can be added to content created in for example mobile clients. Note, uploading an image doesn't create any content itself, it just allows embedding images into content, just like in the UI.
- New API docs exposed by Django REST Swagger. These are in the same place as the old ones, at `/api/`. Adding to the documentation is still a work in progress.
- Add image upload button to the create/reply editor. This makes it possible to upload images from mobile browsers. (#120)
- Make profile "following" button link to "following contacts" page, if user is logged in and own profile.

Changed

- Create and update content will now redirect to the content created or updated. Previous behaviour was user preferred landing page.
- Delete content will now redirect back to the page where the delete was triggered from. Previous behaviour was user preferred landing page. If the content delete is triggered from the content detail page, redirect will happen to user preferred landing page as before. (#204)

Fixed

- Fix internal server error when replying to content that contained only characters outside the western Latin character sets.
- Visual fixes for content rendering in content delete page.
- Make direct profile handle search survive extra spaces before or after the searched handle.

7.11.16 0.2.1 (2017-07-30)

Fixed

- Fix reply form regression introduced in v0.2.0. (#217)

7.11.17 0.2.0 (2017-07-30)

Security

- Fix XSS vulnerability in profile edit. Unsanitized profile field input was allowed and one place showed a field without escaping it. The fields are now sanitized and escaping has been ensured.

  The problem concerned only local users and not remote profile fields which were correctly sanitized already.

Added

- Added search for profiles (#163)

  There is now a global search in the right side of the header. The search returns matches for local and remote profiles based on their name and username part of the handle. Profiles marked with visibility `Self` or `Limited` are excluded from the search results. Profiles marked with visibility `Site` will be excluded if not logged in, leaving only public profile results. If a direct match happens with a full handle, a redirect is done directly to the searched profile.

  IMPORTANT for node maintainers. After pulling in this change, you MUST run the command `python manage.py rebuild_index` to create the search index. Not doing this will cause an error to be raised when trying to search. The indexes are kept up to date automatically after running this command once.

- When searching for profiles based on handle, fetch profile from remote if it isn't found locally (#163)

Changed

- Improved content/reply create/edit form. Replies don't contain visibility or pinned form elements any more. Added also some help texts regarding drag'n'drop image embed, visibility and content pinning.

Fixed

- Make reply notifications to local users not send one single email with all local participants, but one email per participant. Previous implementation would have leaked emails of participants to other participants.
- Correctly send replies to remotes (#210)

  If parent content is local, send via the relayable forwarding mechanism. This ensures parent author signs the content. If parent author is remote, send just to the remote author. The remote author should then relay it.

- Ensure calling `Profile.private_key` or `Profile.key` don't crash if the profile doesn't have keys. Now the properties just return `None`.
- Fix regression in profile all content stream load more functionality. (#190)
- Filter out "limited" visibility profiles from API list results. These profiles are not available in the search so they shouldn't be available to list through the API either.

7.11.18 0.1.0 (2017-07-27)

Initial versioned release. Main implemented features:

- Working streams (followed, public, profiles)
- Content creation
- Content OEmbed / OpenGraph previews

- Replies
- Follow/unfollow of profiles
- Contacts list
- Pinning content to profile