Infrastructure

- Mastodon
  - Mastodon - Building the image
  - Mastodon - Deployment
- Traefik
- Bookwyrm

Mastodon

Mastodon is a federated microblogging platform.

Mastodon - Building the image

Our Mastodon deployment uses a custom-built image based on glitch-soc. Don't build it on the server; instead use your development computer to do so, and then push it to our container registry on Forgejo.

Local developer setup

You'll need a simple Mastodon dev setup to test our custom Solarpunk skin on a new release of glitch-soc Mastodon.

Make sure you have docker installed, with docker compose. If you're not in the docker group, you'll also have to prepend all the docker commands with `sudo`.

Pull the glitch-soc repository into a directory, like one named mastodon.

`git switch` into the latest release branch, like `stable-4.7`. Make sure you pull the latest revision. If you have an edit of the docker-compose.yml, you can do a `git stash`, `git pull`, then `git stash pop` and resolve any merge conflicts.

Follow the official instructions for setting up a development Mastodon setup in docker if you haven't already.

Then, clone our Solarpunk/mastodon repository to a parallel directory, such as solarpunk-mastodon.

Local testing of the skin (custom theme)

You can then copy the contents of the `app/` directory in our solarpunk-mastodon repo, into the developer glitch-soc repository to add the custom skin. That should be all that's necessary to register the skin with glitch-soc.

Finally, test the developer instance of mastodon, and ensure the theme hasn't crashed and burned. It might print errors, or look wrong if so. You'll have to fix up the CSS if that happens. When Mastodon 5.0 comes out it'll be a bit of a pain - we plan to do a big rework then. If the theme is too hard to manage, you can delete the skin folder and use stock glitch-soc. In that case, you wouldn't have to build a custom image if you didn't want to, but it wouldn't hurt. We only have the custom image for the skin, currently.

I recommend creating some posts, interacting a bit, checking menus and such. Don't forget the admin interface as well.

You can also create a new account and it'll open a fake email page to let you verify (you can use Firefox Container Tabs to have multiple accounts when testing, just make sure to copy the verify link into the correct container). This lets you test what notifications look like and other interactions with users.

Building the production image

> right now the forge is down so i am using codeberg for a package registry so these are not quite accurate but this should be fixed soon.

Once done testing, you can stop the dev containers. Time to build.

Open up `docker-compose.yml`, and uncomment the lines that start with `build:`. This will allow us to build the images.

There are two images: `mastodon`, and `mastodon-streaming`. If you use compose to build, it'll build both of them for you. Just run `docker compose build`. This likely will take a while. Assuming you tested and there were no CSS errors, this should go without hitches.

Once done, you need to tag the two images with the solarpunk registry tag. For convenience, you can export a variable `SLPK_VERSION` to the version of glitch you just built (replace the Xs):

```
export SLPK_VERSION=vX.X.X
```

Then run:

```
docker tag ghcr.io/glitch-soc/mastodon:$SLPK_VERSION
git.solarpunk.moe/solarpunk/mastodon:$SLPK_VERSION-glitch
```

```
docker tag ghcr.io/glitch-soc/mastodon-streaming:$SLPK_VERSION
git.solarpunk.moe/solarpunk/mastodon-streaming:$SLPK_VERSION-glitch
```

Now, push to the repository. You'll need to have previously logged in with `docker login git.solarpunk.moe` to be able to push.

```
docker push git.solarpunk.moe/solarpunk/mastodon:$SLPK_VERSION-glitch
docker push git.solarpunk.moe/solarpunk/mastodon-streaming:$SLPK_VERSION-glitch
```

You're now ready to update the deployment on the server side.

Mastodon - Deployment

Deployment of the Solarpunk Mastodon is through `docker compose` via a custom-built image (see Building the image). Currently, the stack is run on the main OVH box accessed at the main domain solarpunk.moe.

Quick Links

- Production site
- Configuration Repository

Anatomy

The service consists of a docker compose stack consisting of several containers. These containers connect to the Internet through our OVH Traefik reverse proxy deployment (to be documented). The configuration of the containers for Traefik is defined through the use of docker labels defined in the docker-compose.yml. The containers used are mostly standard for Mastodon: redis, postgres, the main mastodon service, and sidekiq. Persistent data is stored in the subfolder `public/system`.

The current deployment is run from `/home/vivi/mastodon` on solarpunk.moe. *There's also `/home/vivi/solarpunk.moe/mastodon`, but that's not where the production deployment lives!*

Differences from official docker-compose.yml

Ours is very similar to the official Mastodon docker-compose.yml, with some small changes.

We have tuned postgres slightly through the use of commandline arguments.

We've spread the sidekiq queues into a number of specialized separate containers. The Sidekiq modification is important for tuning the server's performance. The queues were too slow otherwise.

We also remove the exposed port mappings. These are unnecessary due to the use of the Traefik reverse proxy which runs inside the `httpsproxy` network.

Additionally, we use a custom-built image. Please see the Building page for information on how to build the custom Mastodon images on your local machine, and push the image to our forgejo docker registry.

We also version this image in the `.env.production` file for convenience.

Finally, instead of being hosted directly in the docker stack, Elasticsearch is provided through an external DigitalOcean droplet which is tunneled through to the OVH box via wireguard.

First-time setup

We assume you're not a time traveler setting up solarpunk.moe's keys. If you lost the keys, what the heck, why'd you do that? *If so you'll have to follow the Mastodon docs to generate new ones but it'll break the instance...*

Prerequisites

Our configuration assumes the following basic prerequisites on the server:

- Git
- External elasticsearch server tunneled with a known static IP address.
- Docker and docker compose installed.
- The user added to the `docker` group, so that it can manage docker without sudo. [this effectively grants root access to the user, so grant this with care]
- Traefik installation connected to an external docker network called `httpsproxy`.
- simple `sendmail` installation connected to an external docker network called `mail` (more of an afterthought, as it generally is considered spam by large email providers like gmail--something to improve).
- The Configuration Repository cloned to a clean directory.
- Creds for a S3-compatible bucket at digitaloceanspaces.com with the name solarpunk.
- Existing Mastodon-specific secret keys.

Setting up the .env.production file

First, copy `.env.production.template` from the git repo to `.env.production`.

Populate `ES_HOST` and `ALLOWED_PRIVATE_ADDRESSES` with the Elasticsearch IP.

Populate `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` with the secrets from the s3 bucket.

You will also want to populate the other mastodon-specific secret keys from the env file. They're not stored in the repo of course. We don't have a good secrets storage system at the moment so you'll have to track down someone who has them if you don't have access to the old env file.

Setup script

Now you should be able to run `make setup` to set up some local files and directories. Should be quick.

Database

There are two options for this:

Restoring from backup

We need to open up the db container, and run commands to import the SQL dump into the database. I haven't tested it, but in theory:

```
cat some_dump.sql | ./admin.sh exec db psql -U postgres
```

Fresh (danger zone!)

I hope we're not starting fresh! But if we were, you could run `make db_setup`, I guess. We'd need to evaluate whether we need a new domain/new keys or whatever, it'd be a whole crisis.

Startup

Finally, `make pull` (just to verify the image is fetched properly) followed by `make start` should start everything up. You can use `./admin.sh logs` and `./admin.sh ps` to see the logs and running containers, and verify things are good.

Updating to a new release

> right now the forge is down so i am using codeberg for a package registry so these are not quite accurate but this should be fixed soon. make sure it matches expectations. normally should be git.solarpunk.moe registry

When an update comes around, you'll need to build and push a new image on your local machine, first. Once that's done, follow these steps!

Open up the `.env.production` file, and set the `MASTO_VERSION` variable to the proper release that you tagged.

Based on observations when building, ensure that no changes have been made to the official `docker-compose.yml`'s structure, or the `.env.production` file. Sometimes major releases rework some things here and there, so it's good to diff to see that nothing's changed. If changes have been made, adjust accordingly.

Run `make pull` to download the new tag.

Run `make update` to apply any DB migrations and recreate images. After that, you should be done!

As usual, you can run `./admin.sh logs` to check the logs, `./admin.sh ps`, or use Portainer.

Reference

Makefile

An installation `Makefile` is included in the config repository, for easier setup and maintenance. The following make targets are defined:

- `make setup` - Initial server setup.
- `make pull` - Runs `docker compose pull` with appropriate environment, to pull fresh images.
- `make update` - Stops the containers, runs DB pre-migration, recreates docker containers, then runs the rest of the migrations. Generally this is all that's required for a new Mastodon release, but always check the release notes for any special Docker install instructions (they typicallly put them at the bottom of the notes under **When using docker**).
- `make start` - Runs `docker compose up -d` with appropriate environment, to start up the containers.
- `make stop` - Runs `docker compose stop` to stop all containers.
- `make deploy_search` - Runs the tootctl command for deploying Elasticsearch.

For more fine-grained Docker control, see the scripts, below.

Scripts

Two helper scripts are included in the config repository, for easier administration of the containers inside docker compose.

- `./admin.sh` - Runs a docker-compose command with the appropriate environment settings. Include trailing arguments to specify the docker compose command.
- `./toot.sh` - Runs `tootctl`, the mastodon admin interface, inside a temporary docker container using docker compose with appropriate environment settings, to run mastodon admin commands. See the Official Mastodon Documentation for details--simply replace `tootctl` in the docs with `./toot.sh`.

Traefik

Traefik is the reverse proxy we use for some of our services, particularly on the solarpunk.moe OVH box.

Bookwyrm

Bookwyrm is a federated book sharing webapp.

Quick links

- Production site
- Configuration git repository

Anatomy

Our Bookwyrrm instance is hosted on the solarpunk.moe OVH box, and uses docker compose. The directory it is deployed from is `/home/vivi/bookwyrm/`. It uses Traefik.

The docker compose setup is straightforward, but no images exist so we need to build them manually. However, the bookwyrm image is not CPU intensive like Mastodon is, so it's safe to build it directly on the box.

As a result, we have the bookwyrm repo cloned into that directory.

Most of the administration is done with the handy ./bw-dev script.

First-time setup

Prerequisites

This config expects the following on the server:

- Git
- docker and docker compose installed
- The user added to the `docker` group, so that it can manage docker without sudo. [this effectively grants root access to the user, so grant this with care]
- Traefik installed, running on the `httpsproxy` docker network.

- Sendmail installed in the `mail` docker network (though generally that goes to spam with email hosts like Gmail).
- Creds for a S3-compatible bucket at digitaloceanspaces.com with the name solarpunk.
- The keys for the instance

Setup

Our setup will be very close to the official Docker instructions, but using our .env file and customized docker-compose.yml.

Clone our repository somewhere separate from where you want to deploy this docker compose stack.

Clone the official bookwyrm repository at the location you want to deploy to.

Copy over the docker-compose.yml and .env file from the solarpunk repository to the bookwyrm repository.

Fill in the secrets in the .env file. Time to build the image.

```
./bw-dev build
```

Then run setup:

```
./bw-dev setup
```

Now you probably want to restore the database from a backup.

```
cat some_backup.sql | ./bw-dev exec db psql -U fedireads
```

And then finally, start it up:

```
./bw-dev up -d
```

As mentioned in the updating guide, the main container takes a while to start up the first time due to collecting static files and building the themes.

Updating

To update, I follow a simple strategy. First, I stash the local changes (mostly to the docker-compose.yml), and pull the updates on the `production` branch:

```
git stash
git pull
git stash pop
```

Then generally there are conflicts in the docker-compose.yml. Open it in a text editor and resolve them (don't let the anubis or certbot configurations come back -- those are all handled externally at the Traefik layer). Make sure you understand how git conflict files are structured so you don't mess up the docker-compose.yml with a bad merge.

`git add docker-compose.yml` and `git status` to make sure there are no leftover conflicted files.

Now, build:

```
./bw-dev build
```

It will take a couple minutes to install everything.

Now migrate the DB:

```
./bw-dev migrate
```

And, start it up:

```
./bw-dev up -d
```

Note that the newly created containers will take a while to start up, as some compiling of themes happens as well as collecting of static files.