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

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`.

Revision #14
Created 2026-10-31 03:42:14 UTC by vv
Updated 2026-10-31 07:09:11 UTC by vv