Developers Handbook

Developers contribute code to the software that makes the weague run! If anything feels out of place or missing, or if you just have a general question, the Developer Chat on Zulip is a good place to start.

- How to Develop for Website League
- How to Build Every League Project
- Docker Cookbook
- External Developer References
- Contacting Upstream Developers

How to Develop for Website League

The Website League runs on a lot of different software, mostly adopted from the existing Fediverse and ActivityPub ecosystems.

Where to Contribute

The primary repository for all of the projects we've adopted is hosted on gitlab:

https://gitlab.com/website-league

For every repository here, the default branch is the branch we recommend node operators use in production. for example, Akkoma's version of this branch is named **wl-stable**. If you want a change to be shared across the league, your goal should be to submit a gitlab merge request to one of these branches.

This also means you don't need to be a node operator, or even a steward, to contribute! Though, before recommending a patch be adopted everywhere, it may help to contact a node operator directly to test your changes in production.

For each feature, start by making a local feature branch. The best patches are small, self-contained, and easy for an outside observer to review, but a bad patch is better than no patch! A maintainer will look at your patch and provide feedback.

Up, or downstream?

- If you're making a change to specific node to provide a local-only feature, you're very welcome to do that in a private or node-specific fork.
- If you're making a change that would make the website league better generally, please start from **wl-stable** branch and submit a merge request. Then, to get that change on your local node, you can pull from upstream, the same as any other change.
- If you're fixing a technical issue or bug: also consider submitting it to the upstream project ! This way, everyone benefits, and we can minimize the work league contributors need to provide to keep your patch alive.

summarizing: **Start as far upstream as you can, and let the patch flow downstream**.

Thank you for contributing to the Website League.

How to Build Every League Project

Building the website league stack from source is a good first step to making changes. you can test things locally on a single instance, but having all software ready to go makes it easy to test changes that work across different nodes and frontends.

Platform

To keep this document small and linear, we'll assume an ubuntu-based linux distribution. upstream documents often cover alternate distributions, if you'd prefer.

If you're on windows, you can get an ubuntu container relatively easily using the Windows Subsystem for Linux. Open up a powershell instance and copy in:

```
wsl --install -d ubuntu
```

Now you can pull up your container by just typing "ubuntu" into the start menu.

Could we use Docker?

possibly!

Backends

Akkoma

The upstream akkoma docs are split into Installing, and Development Environment pages.

Operators also have their own guide for installing a production environment.

Step 1 is to get Elixir, and other system dependencies. Do this how you like, but on ubuntu asdf will work:

```
# deps
sudo apt install git build-essential cmake libmagic-dev postgresql postgresql-contrib \
  curl unzip build-essential autoconf m4 libncurses5-dev libssh-dev unixodbc \
  imagemagick ffmpeg libimage-exiftool-perl

# elixir

# for bash
echo 'export PATH="$HOME/go/bin:$HOME/.asdf/shims:$PATH"' >> ~/.bash_profile
# for fish
fish_add_paths -U ~/go/bin ~/.asdf/shims

go install github.com/asdf-vm/asdf/cmd/asdf@v0.18.0
asdf plugin add erlang https://github.com/asdf-vm/asdf-erlang.git
asdf plugin add elixir https://github.com/asdf-vm/asdf-elixir.git
asdf install erlang 27.2.4
asdf install elixir 1.18.2-otp-27
asdf set erlang 27.2.4
asdf set elixir 1.18.2-otp-27
```

Next, we need to prepare the elixir environment and generate a local configuration:

```
# download dependencies
mix deps.get
# generate config using an interactive wizard
mix pleroma.instance gen
# apply generated config
mv config/{generated_config.exs,dev.secret.exs}
```

If it asks you to install hex or rebar3, say yes. During the interactive config generator:

- For domain you can use localhost
- Use the defaults for anything database related (hostname, username, database name, password) - the configuration guide will effectively generate a script to create that database, user, and password on Postgres for you.

To prepare the database

```
# run once to set up the database
sudo -Hu postgres psql -f config/setup_db.psql
# primary migration script. re-run if the db schema changes
```

```
mix ecto.migrate
```

To recompile the backend on its own:

```
mix compile
```

To run:

```
mix phx.server
```

GoToSocial

Upstream, GoToSocial stores it's local build and development information in the contributing.md file. Operators also have their own guide for installing a production environment.

Step 1 is to get Go, nodejs and other dependencies. On Ubuntu:

```
sudo apt install git build-essential nodejs npm golang
sudo npm install -g yarn
```

Time to build:

```
# backend
DEBUG=1 ./scripts/build.sh
# frontend
yarn --cwd ./web/source install && yarn --cwd ./web/source ts-patch install
```

Now, to run locally:

```
# run backend in "testrig" mode
DEBUG=1 GTS_PORT=8080 ./gotosocial testrig start
# automatically recompile GoToSocial if changed:
DEBUG=1 GTS_PORT=8080 nodemon -e go --signal SIGTERM --exec "go run ./cmd/gotosocial --host localhost
testrig start || exit 1"
# automatically recompile js frontend
NODE_ENV=development yarn --cwd ./web/source dev
```

GoToSocial doesn't require a database setup pass; it uses sqlite by default.
At the login screen, enter the email address zork@example.org and password password. You will get a confirmation prompt. Accept, and you are logged in as Zork.

Frontends

For most every frontend, you will need at least node.js, and possibly yarn too. On Ubuntu:

```
sudo apt install git build-essential nodejs npm golang
sudo npm install -g yarn
```

Akkoma-fe

Upstream akkoma-fe docs are in the readme

```
# serve with hot reload at localhost:8080
npm run dev

# build for production with minification
npm run build

# run unit tests
npm run unit
```

Pillbug

Upstream pillbug docs are in the readme

```
# serve with live reload at localhost:3000
npm run dev

# build for production with minification
npm run build
```

Docker Cookbook

Because so many services are involved in running the weague, self-contained docker images can be handy in replicating larger pieces of it at a time.

Postgres

Postgres is used by Akkoma for storage. Adminer (formerly phpMyAdmin) is a handy web interface for introspecting postgres.

```
services:
  postgres:
    image: postgres:17
    restart: unless-stopped
    environment:
      POSTGRES_DB: akkoma,
      POSTGRES_USER: akkoma,
      POSTGRES_PASSWORD: akkoma,
    volumes:
      - postgres:/var/lib/postgresql/data
    ports:
      - "5432:5432"
  adminer:
    image: adminer
    restart: unless-stopped
    ports:
      - "8080:8080"
volumes:
  postgres:
```

External Developer References

API references

The website league communicates through **ActivityPub**, which is the underlying specification that powers Mastodon and the fediverse. The specifics of this standard are open, published, and come with a test suite for conformance.

Clients to either type of node can be written using the Mastodon API, which both (partially) support. Pillbug, akkoma-fe and friends are all written to the Mastodon API. Akkoma and GoToSocial also have instance specific extensions.

Additionally, all Website League nodes require **Allowlist Support** . This is a non-standardized feature that allows nodes to reject messages from nodes that are not part of the allowlist. this is how we keep the League separate from the wider fediverse, and enforce our standards of conduct (nodes not enforcing them are removed from the allowlist).

Akkoma achieves this through their Message Rewrite Facility, an intentionally flexible mechanism for handling the routing of messages.

GoToSocial achieves this through Allowlist Federation Mode, a user configuration that does exactly what it says on the tin.

Contacting Upstream Developers

GoToSocial

(copied from the readme)

> For questions and comments, you can join our Matrix space at `#gotosocial-space:superseriousbusiness.org`. This is the quickest way to reach the devs. You can also mail admin@gotosocial.org.
>
> For bugs and feature requests, please check to see if there's already an issue, and if not, open one or use one of the above channels to make a request (if you don't have a Codeberg account).

Akkoma

(from docs page)

> For support or general questions, pop over to #akkoma and #akkoma-dev at irc.akkoma.dev (port 6697, SSL)
>
> For more general meta-discussion, for example discussion of potential future features, head on over to meta.akkoma.dev

issues and pull requests can be filed on the upstream repository.

Pillbug

Issues can be filed on github.