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
```

Revision #6
Created 2025-10-04 17:18:57 UTC by alloyed
Updated 2025-10-14 07:18:00 UTC by alloyed