Olika installationer (self-hosted)

detta är min agile i ett framgångsrikt installerande på min server

- Soapbox!
- Firefish!

Soapbox!

We recommend installing Soapbox on a **dedicated VPS (virtual private server) running Ubuntu 22.04 LTS**. You should get your VPS up and running before starting this guide.

make sure to be the root user

```
sudo su
```

System setup

Before installing Soapbox, we have to prepare the system.

Install updates

Usually a fresh VPS already has outdated software, so run the following commands to update it:

```
apt update
```

```
apt upgrade
```

When prompted ([Y/n]) type Y and hit Enter.

Install system dependencies

Soapbox relies on some additional system software in order to function. Install them with the following command:

```
apt install git curl build-essential postgresql postgresql-contrib cmake libmagic-dev imagemagick ffmpeg libimage-exiftool-perl nginx certbot unzip libssl-dev automake autoconf libncurses5-dev fasttext
```

Create the Pleroma user

For security reasons, it’s best to run Rebased as a separate user with limited access.

We’ll create this user and call it pleroma:

```
useradd -r -s /bin/false -m -d /var/lib/pleroma -U pleroma
```

Install Rebased

It’s time to install Rebased, the backend for Soapbox. Let’s get things up and running.

Downloading the source code

Download the Rebased source code with git:

```
git clone https://gitlab.com/soapbox-pub/rebased /opt/pleroma
```

```
chown -R pleroma:pleroma /opt/pleroma
```

Enter the source code directory, and become the pleroma user:

```
cd /opt/pleroma
```

```
sudo -Hu pleroma bash
```

(You should be the pleroma user in /opt/pleroma for the remainder of this section.)

Install Elixir

Rebased uses the Elixir programming language (based on Erlang). It’s important we use a specific version of Erlang (24), so we’ll use the asdf version manager to install it.

Install asdf:

```
git clone https://github.com/asdf-vm/asdf.git ~/.asdf --branch v0.10.0
```

```
echo ". $HOME/.asdf/asdf.sh" >> ~/.bashrc
```

```
echo ". $HOME/.asdf/completions/asdf.bash" >> ~/.bashrc
```

```
exec bash
```

```
asdf plugin-add erlang
```

```
asdf plugin-add elixir
```

Finally, install Erlang/Elixir:

```
asdf install
```

(This will take about 15 minutes. ☕)

Compiling Rebased

Install basic Elixir tools for compilation:

```
mix local.hex --force
```

```
mix local.rebar --force
```

Fetch Elixir dependencies:

```
mix deps.get
```

Finally, compile Soapbox:

```
MIX_ENV=prod mix compile
```

(This will take about 10 minutes. ☕)

Generate the configuration

It’s time to preconfigure our instance. The following command will set up some basics such as your domain name:

```
MIX_ENV=prod mix pleroma.instance gen
```

If you're happy with it, rename the generated file so it gets loaded at runtime:

```
mv config/generated_config.exs config/prod.secret.exs
```

Provision the database

The previous section also created a file called config/setup_db.psql, which you can use to create the database.

Exit back to the root user (for the remainder of this document):

```
exit
```

Execute the SQL file as the postgres user:

```
sudo -Hu postgres psql -f config/setup_db.psql
```

Now run the database migration as the pleroma user:

```
sudo -Hu pleroma bash -i -c 'MIX_ENV=prod mix ecto.migrate'
```

Start Rebased

Copy the systemd service and start Soapbox:

```
cp /opt/pleroma/installation/pleroma.service /etc/systemd/system/pleroma.service
```

```
systemctl enable --now pleroma.service
```

If you've made it this far, congrats! You're very close to being done. Your Rebased server is running, and you just need to make it accessible to the outside world.

Getting online

The last step is to make your server accessible to the outside world. We’ll achieve that by installing Nginx and enabling HTTPS support.

HTTPS

We’ll use certbot to get an SSL certificate.

First, shut off Nginx:

```
systemctl stop nginx
```

Now you can get the certificate:

```
mkdir -p /var/lib/letsencrypt/
```

```
certbot certonly --email your@emailaddress -d yourdomain --standalone
```

Replace your@emailaddress and yourdomain with real values.

Nginx

Copy the example nginx configuration and activate it:

```
cp /opt/pleroma/installation/pleroma.nginx /etc/nginx/sites-available/pleroma.nginx
```

```
ln -s /etc/nginx/sites-available/pleroma.nginx /etc/nginx/sites-enabled/pleroma.nginx
```

You must edit this file:

```
nano /etc/nginx/sites-enabled/pleroma.nginx
```

Change all occurrences of example.tld with your site’s domain name. Use Ctrl+X, Y, Enter to save.

Finally, enable and start nginx:

```
systemctl enable --now nginx.service
```

□□ Congrats, you’re done! Check your site in a browser and it should be online.

```
Install Soapbox
```

It’s finally time to install Soapbox itself! First, get the latest build.

```
curl -O https://dl.soapbox.pub/main/soapbox.zip
```

Next, unzip it.

```
busybox unzip soapbox.zip -o -d /opt/pleroma/instance/static
```

Refresh your website. That’s it!

Post-istallation

Below are some additional steps you can take after you’ve finished installation.

Create your first user

If your instance is up and running, you can create your first user with administrative rights with the following task:

```
cd /opt/pleroma
```

```
sudo -Hu pleroma bash -i -c 'MIX_ENV=prod mix pleroma.user new username your@emailaddress --admin'
```

Renewing SSL

If you need to renew the certificate in the future, uncomment the relevant location block in the nginx config and run:

```
certbot certonly --email your@emailaddress -d yourdomain --webroot -w /var/lib/letsencrypt/
```

Upgrading

To upgrade Rebased (the backend), shell into your server and issue the following commands.

```
sudo -Hu pleroma bash
```

```
cd /opt/pleroma
```

```
git pull origin main
```

```
asdf install
```

```
mix deps.get
```

```
MIX_ENV=prod mix ecto.migrate
```

```
exit
```

```
systemctl restart pleroma
```

To upgrade Soapbox (frontend), shell into your server and re-run the install commands.

```
curl -O https://dl.soapbox.pub/main/soapbox.zip
```

```
busybox unzip soapbox.zip -o -d /opt/pleroma/instance/static
```

Firefish!

1. Install dependencies

Make sure that you can use the `sudo` command before proceeding.

Utilities

```
sudo apt update
sudo apt install build-essential python3 curl wget git lsb-release
```

Node.js and pnpm

Instructions can be found at this repository.

```
NODE_MAJOR=20
curl -fsSL "https://deb.nodesource.com/setup_${NODE_MAJOR}.x" | sudo -E bash -
sudo apt install nodejs

# check version
node --version
```

You also need to enable `pnpm`.

```
sudo corepack enable
corepack prepare pnpm@latest --activate

# check version
pnpm --version
```

PostgreSQL and PGroonga

PostgreSQL install instructions can be found at this page.

```
sudo sh -c 'echo "deb https://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" >
/etc/apt/sources.list.d/pgdg.list'
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
sudo apt update
sudo apt install postgresql-16

sudo systemctl enable --now postgresql

# check version
psql --version
```

PGroonga install instructions can be found at this page.

```
wget "https://apache.jfrog.io/artifactory/arrow/$(lsb_release --id --short | tr 'A-Z' 'a-z')/apache-arrow-apt-source-
latest-$(lsb_release --codename --short).deb"
sudo apt install "./apache-arrow-apt-source-latest-$(lsb_release --codename --short).deb"
wget "https://packages.groonga.org/debian/groonga-apt-source-latest-$(lsb_release --codename --short).deb"
sudo apt install "./groonga-apt-source-latest-$(lsb_release --codename --short).deb"
sudo apt update
sudo apt install postgresql-16-pgdg-pgroonga

rm "apache-arrow-apt-source-latest-$(lsb_release --codename --short).deb" "groonga-apt-source-latest-
$(lsb_release --codename --short).deb"
```

Redis

Instructions can be found at this page.

```
curl -fsSL https://packages.redis.io/gpg | sudo gpg --dearmor -o /usr/share/keyrings/redis-archive-keyring.gpg
echo "deb [signed-by=/usr/share/keyrings/redis-archive-keyring.gpg] https://packages.redis.io/deb $(lsb_release
-cs) main" | sudo tee /etc/apt/sources.list.d/redis.list
sudo apt update
sudo apt install redis

sudo systemctl enable --now redis-server
```

```
# check version
redis-cli --version
```

FFmpeg

```
sudo apt install ffmpeg
```

2. Set up a database

1. If you forgot the password you typed, you can reset it by executing `sudo -u postgres psql -c "ALTER USER firefish PASSWORD 'password';"`.

Create a database

```
sudo -u postgres createdb --encoding='UTF8' --owner=firefish firefish_db
```

Create a database user

```
sudo -u postgres createuser --no-createdb --no-createrole --no-superuser --encrypted --pwprompt firefish
```

Enable PGronnga extension

```
sudo -u postgres psql --command='CREATE EXTENSION pgroonga;' --dbname=firefish_db
```

3. Configure Firefish

Create an user for Firefish and switch user

```
sudo useradd --create-home --user-group --shell /bin/bash firefish
sudo su --login firefish

# check the current working directory
# the result should be /home/firefish
pwd
```

Install Rust toolchain

Instructions can be found at this page.

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
. "${HOME}/.cargo/env"

# check version
cargo --version
```

Clone the Firefish repository

```
git clone --branch=main https://firefish.dev/firefish/firefish.git
```

Copy and edit the config file

```
cd firefish
cp .config/example.yml .config/default.yml
nano .config/default.yml
```

```
url: https://your-server-domain.example.com  # change here
port: 3000

db:
  host: localhost
  port: 5432
  db: firefish_db
  user: firefish
  pass: your-database-password  # and here
```

4. Build Firefish

Build

```
pnpm install --frozen-lockfile
NODE_ENV=production NODE_OPTIONS='--max-old-space-size=3072' pnpm run build
```

Execute database migrations

```
pnpm run migrate
```

Logout from `firefish` user

```
exit
```

5. Preparation for publishing a server

1. Set up a firewall

To expose your server securely, you may want to set up a firewall. We use ufw in this instruction.

```
sudo apt install ufw
# if you use SSH
# SSH_PORT=22
# sudo ufw limit "${SSH_PORT}/tcp"
sudo ufw default deny
sudo ufw allow 80
sudo ufw allow 443
sudo ufw --force enable

# check status
sudo ufw status
```

2. Set up a reverse proxy

In this instruction, we use Caddy to make the Firefish server accesible from internet. However, you can also use Nginx if you want (example Nginx config file).

Install Caddy

```
sudo apt install debian-keyring debian-archive-keyring apt-transport-https
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/gpg.key' | sudo gpg --dearmor -o
/usr/share/keyrings/caddy-stable-archive-keyring.gpg
curl -1sLf 'https://dl.cloudsmith.io/public/caddy/stable/debian.deb.txt' | sudo tee /etc/apt/sources.list.d/caddy-
stable.list
```

```
sudo apt update
sudo apt install caddy

# check version
caddy version
```

Replace the config file

```
sudo mv /etc/caddy/Caddyfile /etc/caddy/Caddyfile.bak
sudo nano /etc/caddy/Caddyfile
```

```
your-server-domain.example.com {
	reverse_proxy http://127.0.0.1:3000

	log {
		output file /var/log/caddy/firefish.log
	}
}
```

Restart Caddy

```
sudo systemctl restart caddy
```

6. Publish your Firefish server

Create a service file

```
sudo nano /etc/systemd/system/firefish.service
```

```
[Unit]
Description=Firefish daemon
Requires=redis.service caddy.service postgresql.service
After=redis.service caddy.service postgresql.service network-online.target

[Service]
Type=simple
User=firefish
Group=firefish
UMask=0027
```

```
ExecStart=/usr/bin/pnpm run start
WorkingDirectory=/home/firefish/firefish
Environment="NODE_ENV=production"
Environment="npm_config_cache=/tmp"
Environment="NODE_OPTIONS=--max-old-space-size=3072"
# uncomment the following line if you use jemalloc (note that the path varies on different environments)
# Environment="LD_PRELOAD=/usr/lib/x86_64-linux-gnu/libjemalloc.so.2"
StandardOutput=journal
StandardError=journal
SyslogIdentifier=firefish
TimeoutSec=60
Restart=always

CapabilityBoundingSet=
DevicePolicy=closed
NoNewPrivileges=true
LockPersonality=true
PrivateDevices=true
PrivateIPC=true
PrivateMounts=true
PrivateUsers=true
ProtectClock=true
ProtectControlGroups=true
ProtectHostname=true
ProtectKernelTunables=true
ProtectKernelModules=true
ProtectKernelLogs=true
ProtectProc=invisible
RestrictNamespaces=true
RestrictRealtime=true
RestrictSUIDSGID=true
SecureBits=noroot-locked
SystemCallArchitectures=native
SystemCallFilter=~@chown @clock @cpu-emulation @debug @ipc @keyring @memlock @module @mount
@obsolete @privileged @raw-io @reboot @resources @setuid @swap
SystemCallFilter=capset pipe pipe2 setpriority

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

Start Firefish

```
sudo systemctl enable --now firefish
```