Dreadful Work - Lemmy

Today we take a small break from OIDC matters and move to the usual "username + password (+OTP)", turning our gaze towards the RAT (actually I think they are lemmings, but as far as I remember they were things that committed suicide both in video games and in real life, so I feel more comfortable thinking of them as rats): Lemmy, a link aggregator like Reddit, but decentralized, federatable and so on, the usual song and dance.

The attentive reader will wonder why I intend to tackle the installation of a gadget so backward it doesn't have OIDC: the answer is simple: it doesn't have it yet. We are at version 0.19.17, and SSO will be possible (apparently) only from version 1.x, which is in development but hasn't even been released in alpha. And there are no alternatives.

So, cut the chatter.

Step 1: Installation

Like the other times, we will use good old Docker. Let's create the directory:

```
# sudo mkdir /opt/lemmy && cd /opt/lemmy
```

Then let's go with `docker-compose.yml`:

```
version: "3.7"

x-logging: &default-logging
  driver: "json-file"
  options:
    max-size: "50m"
    max-file: "4"

services:
  proxy:
    image: dessalines/lemmy:0.19.17
    hostname: lemmy
    restart: unless-stopped
    environment:
      -
RUST_LOG="warn,lemmy_server=debug,lemmy_api=debug,lemmy_api_common=debug,lemmy_api_crud=debug,lemmy_apub=debug,lemmy_db_sche
      - RUST_BACKTRACE=full
    ports:
      - "8536:8536"
      - "6669:6669"
    volumes:
      - ./lemmy.hjson:/config/config.hjson:Z
    depends_on:
      - postgres
      - pictrs
    logging: *default-logging

  lemmy-ui:

    image: dessalines/lemmy-ui:0.19.17
    ports:
      - "1235:1234"
    environment:
      - LEMMY_UI_LEMMY_INTERNAL_HOST=lemmy:8536
      - LEMMY_UI_LEMMY_EXTERNAL_HOST=lemmy.example.org
      - LEMMY_UI_HTTPS=false
      - LEMMY_UI_DEBUG=true
    depends_on:
      - lemmy
    restart: unless-stopped
    logging: *default-logging
    init: true
```

```
  pictrs:
    image: asonix/pictrs:0.5
    hostname: pictrs

    environment:
      - PICTRS_OPENTELEMETRY_URL=http://otel:4137
      - PICTRS__API_KEY="e851f7ee8ca705c7a04af1374694e4c0de188687b71f11cef8750a4260197ff1"
      - RUST_LOG=debug
      - RUST_BACKTRACE=full
      - PICTRS__MEDIA__VIDEO_CODEC=vp9
      - PICTRS__MEDIA__GIF__MAX_WIDTH=256
      - PICTRS__MEDIA__GIF__MAX_HEIGHT=256
      - PICTRS__MEDIA__GIF__MAX_AREA=65536
      - PICTRS__MEDIA__GIF__MAX_FRAME_COUNT=400
    user: 991:991
    ports:
      - "127.0.0.1:8537:8080"
      - "127.0.0.1:6670:6669"
    volumes:
      - ./volumes/pictrs:/mnt:Z
    restart: unless-stopped
    logging: *default-logging

  postgres:
    image: postgres:16-alpine
    hostname: postgres
    command:
      [
        "postgres",
        "-c",
        "session_preload_libraries=auto_explain",
        "-c",
        "auto_explain.log_min_duration=5ms",
        "-c",
        "auto_explain.log_analyze=true",
        "-c",
        "track_activity_query_size=1048576",
      ]
    ports:
      - "5433:5432"
    environment:
      - POSTGRES_USER=lemmy
      - POSTGRES_PASSWORD=ab1c3da5295ff9f3f5e2e6da0309cf3af3cf88e8581c3b4f4ea465b6bf0835c3
      - POSTGRES_DB=lemmy
    volumes:
      - ./volumes/postgres:/var/lib/postgresql/data:Z
    restart: unless-stopped
    logging: *default-logging
```

Keep in mind that `PICTRS__API_KEY="e851f7ee8ca705c7a04af1374694e4c0de188687b71f11cef8750a4260197ff1"` and `POSTGRES_PASSWORD=ab1c3da5295ff9f3f5e2e6da0309cf3af3cf88e8581c3b4f4ea465b6bf0835c3` were created with `openssl rand -hex 32`.

Next, it's the turn of `lemmy.hjson`:

```
{
  setup: {
    admin_username: "errno0x0d"
    admin_password: "74a330c89f8a3f901e0a224eaedceb9c0766649cf016d9f039ef9f0c03e013ed"
    site_name: "example.org"
  }
  database: {
    host: "postgres"
    user: "lemmy"
    database: "lemmy"
    password: "b1c3da5295ff9f3f5e2e6da0309cf3af3cf88e8581c3b4f4ea465b6bf0835c3"
    port: 5432
    pool_size: 5

  }
```

```
slur_filter:
    '''
    (fag(g|got|tard)?\b|cock\s?sucker(s|ing)?|ni((g{2,}|q)+|[gq]{2,})[e3r]+(s|z)?|mudslime?s?|kikes?|\bspi(c|k)s?
\b|\bchinks?|gooks?|bitch(es|ing|y)?|whor(es?|ing)|\btr(a|@)nn?(y|ies?)|\b(b|re|r)tard(ed)?s?)
    '''
  hostname: "lemmy.example.org"
  bind: "0.0.0.0"
  port: 8536

  pictrs: {
    url: "http://pictrs:8080/"
    api_key: "e851f7ee8ca705c7a04af1374694e4c0de188687b71f11cef8750a4260197ff1"
    cache_external_link_previews: true
  }

email: {
    # hostname (smtp.example.com) and port (25, 465, 587, etc) of the smtp server
    smtp_server:  "xxxxxxxxxx"
    # login name for smtp server
    smtp_login: "xxxxxxxxx"
    # password to login to the smtp server
    smtp_password: "xxxxxxxxx"
    smtp_from_address: "lemmy@example.org"
    tls_type: "starttls"
  }
}
```

The `admin_password: "74a330c89f8a3f901e0a224eaedceb9c0766649cf016d9f039ef9f0c03e013ed"` was set, as before, via openssl; the values of api_key and postgres password were replaced. The email section must also be configured according to your mail provider (such as Brevo).

at this point:

```
docker compose up -d; docker compose down; sudo chown 991:991 volumes/pictrs; docker compose up -d
```

Step 2: the non-direct proxy (revproxy)

Lemmy consists of NO LESS than two main services: frontend and backend, so we will use nginx for the reverse proxy:

```
# /etc/nginx/sites-enabled/lemmy.example.org.conf

 upstream lemmy {
        server "127.0.0.1:8536";
    }
    upstream lemmy-ui {
        server "127.0.0.1:1235";
    }

    server {

        server_name lemmy.example.org;
        listen 80;
        server_tokens off;
        access_log /var/log/nginx/lemmy.example.org.access.log;
        error_log /var/log/nginx/lemmy.example.org.error.log;

        gzip on;
        gzip_types text/css application/javascript image/svg+xml;
        gzip_vary on;

        # Upload limit, relevant for pictrs
        client_max_body_size 20M;

        add_header X-Frame-Options SAMEORIGIN;
        add_header X-Content-Type-Options nosniff;
        add_header X-XSS-Protection "1; mode=block";
```

```
        location = /tos {
            root /var/www/html/example.org/lemmy;
            try_files /tos.html =404;
        }
        location = /legale {
            root /var/www/html/example.org/lemmy;
        try_files /legale.html =404;
        add_header Content-Type text/html;
        }
        location / {
            set $proxpass "http://lemmy-ui";

            if ($http_accept = "application/activity+json") {
              set $proxpass "http://lemmy";
            }
            if ($http_accept = "application/ld+json; profile=\"https://www.w3.org/ns/activitystreams\"") {
              set $proxpass "http://lemmy";
            }
            if ($request_method = POST) {
              set $proxpass "http://lemmy";
            }
            proxy_pass $proxpass;
            proxy_http_version 1.1;
            rewrite ^(.+)/+$ $1 permanent;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;

    }

        location ~ ^/(api|pictrs|feeds|nodeinfo|version|.well-known) {
            proxy_pass "http://lemmy";
            # proxy common stuff
            proxy_http_version 1.1;
            proxy_set_header Upgrade $http_upgrade;
            proxy_set_header Connection "upgrade";

            # Send actual client IP upstream
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header Host $host;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;

    }

}
```

This creates a service reachable at `http://lemmy.example.org` and it's in plain text (port 80); it is assumed that the reverse proxy is on the same machine running Docker with the services; otherwise, the `upstream` blocks must be changed to point to something other than `127.0.0.1`, namely the address of the machine hosting the containers.

If you then want to expose everything in TLS, you can use Cloudflare with its cloudflared tunnels (formerly Argo), or - if the machine with nginx is reachable from the outside - use certbot: `certbot nginx -d lemmy.example.org`.

Startup

Point your browser to `https://lemmy.example.org` and use the credentials entered in the `admin` block of `lemmy.hjson`. Those credentials are only for the setup: if you change them, they will remain changed.

[May the big mole be with you]

[EOF]