Miniflux Documentation

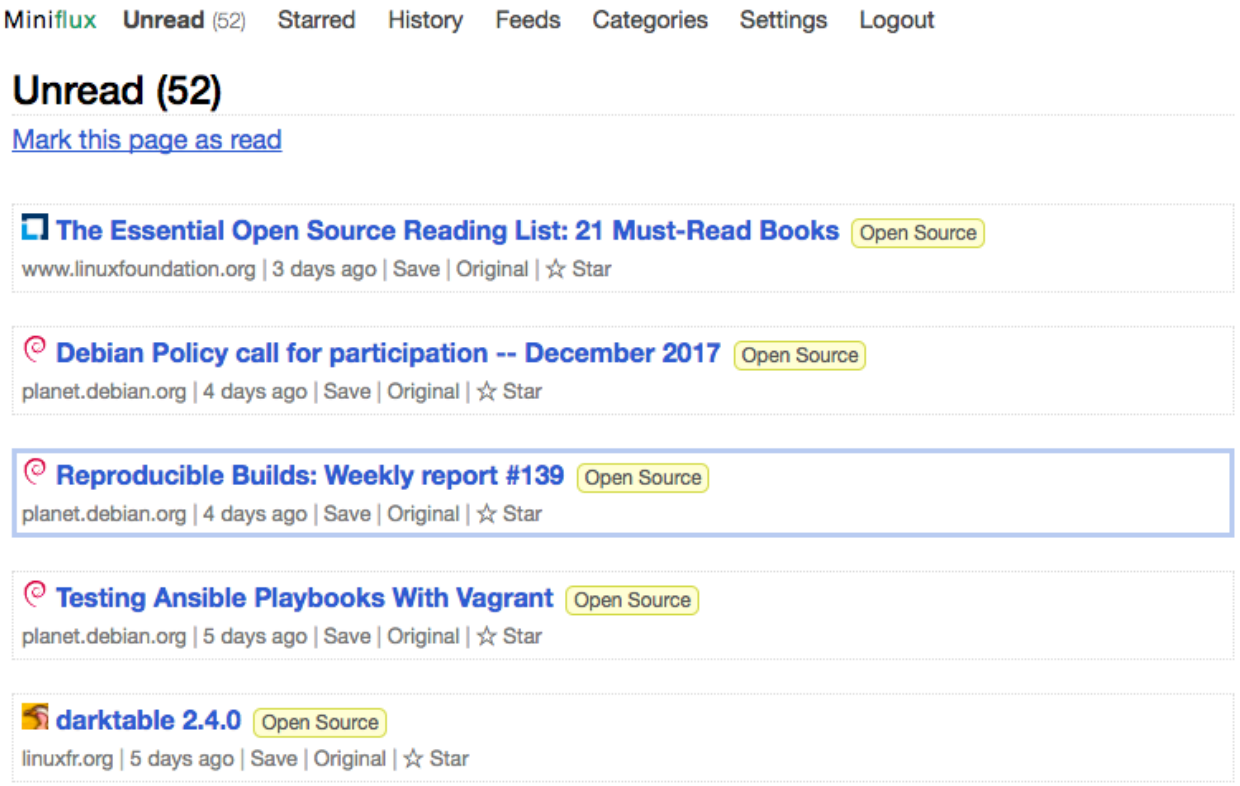
Release 2.0.0

The Miniflux Authors

Jun 05, 2018

Contents:

Miniflux is a minimalist and opinionated feed reader.



CHAPTER 1

Introduction

Miniflux is a self-hosted software to read RSS/Atom/JSON feeds.

- Miniflux 2 is a rewrite of Miniflux v1 in Golang/Postgres.
- **Miniflux 2 doesn't try to implement all features of Miniflux 1**.
- Miniflux 1 is not maintained anymore, the project is archived here: https://github.com/miniflux/miniflux-legacy.

1.1 License

Apache 2.0

CHAPTER 2

Opinionated?

Miniflux is a minimalist software. The purpose of this application is to read feeds. *Nothing else*.

2.1 Focus on Simplicity

- Having a gazillion of features makes the software hard to maintain, hard to troubleshoot and increase the number of bugs.
- This software doesn't try to satisfy the needs of everyone.

2.2 Why the user interface is ugly?

- The Miniflux layout is optimized to scan entries quickly.
- The design of Miniflux is inspired by Hacker News, Lobsters and Pinboard.
- To be honest, the main developer of Miniflux is not a UI/UX guy.

2.3 Why are you not developing my feature request?

- Developing a software takes a lot of time. Don't expect anyone to work for free.
- As mentioned above, the number of features is voluntarily limited. Nobody likes bloatware.
- Improving existing features is more important than adding new ones.

2.4 Why choose Golang as a programming language?

Go is probably the best choice for self-hosted software:

- Go is a simple programming language.
- Running code concurrently is part of the language.
- It's faster than a scripting language like PHP or Python.
- The final application is a binary compiled statically without any dependency.
- You just need to drop the executable on your server to deploy the application.
- You don't have to worry about what version of PHP/Python is installed on your machine.
- Packaging the software using RPM/Debian/Docker is straightforward.

2.5 Why Postgresql?

Miniflux is compatible only with Postgres.

- Supporting multiple databases increases the complexity of the software.
- Testing the software with all major versions of Mysql, MariaDB, Sqlite, Postgres is a lot of work.
- ORM abstracts some interesting features provided your database.
- Managing schema migrations with Sqlite is painful.
- Postgresql is powerful, rock solid and battle tested.
- Postgresql is a great independent open source software.
- Miniflux uses *hstore/jsonb/inet* data types and handles user timezones with Postgres.

2.6 Why no Javascript framework?

Miniflux uses Javascript only where it's necessary.

- Rendering templates server side is so simple and fast enough for that kind of application.
- Using Javascript frameworks increase complexity.
- The Javascript ecosystem is moving all the time, sticking to the standard is probably more sustainable.

2.7 Why only ECMAScript 6?

Miniflux uses ES6 and the Fetch API.

- All modern browsers support ES6 nowadays.
- Only Internet Explorer 11 doesn't support ES6, but who cares?
- Using a Javascript transpiler introduce another set of useless dependencies.

2.8 Why there is no mobile application?

Using the web UI on your smartphone is not so bad. The stylesheet is responsive and you can even swipe entries horizontally.

- Developing a native mobile application for each platform (iOS and Android) and different devices (smartphones and tablets) takes a lot of work.
- The main developer of Miniflux is not a mobile application developer.
- The development of mobile clients is left to the open source community.
- You have to pay a fee to publish your app on the store even if your app doesn't make any money.

CHAPTER 3

Features

3.1 Reader

- Feed formats supported: Atom, RSS 1.0/2.0 and JSON
- OPML import/export
- Support multiple enclosures/attachments (Podcasts, videos, music, and images)
- Play videos from YouTube channels directly inside Miniflux
- Categories
- Bookmarks
- Fetch website icons (favicons)
- Save articles to third-party services
- Available in Chinese, Dutch, English, French, German, and Polish

3.2 Privacy

- Remove pixel trackers
- Fetch original links when the feed is coming from FeedBurner
- Open external links with the attributes `rel="noopener noreferrer" referrerpolicy="no-referrer"`
- Image proxy to avoid mixed content warnings with HTTPS
- Play Youtube videos by using the domain *youtube-nocookie.com*
- Block any external Javascript code to avoid tracking

3.3 Content Manipulation

- Fetch original article and returns relevant contents (Readability)
- Custom scraper rules based on CSS selectors
- **Custom rewriting rules**
  - Append image title for comics
  - Add Youtube video

3.4 User Interface

- Stylesheet optimized for readability
- Responsive design (works on desktop, tablet, and mobile devices)
- No fancy user interface
- Doesn't require to download an application from the App/Play Store
- You could add Miniflux to the home screen
- Keyboard shortcuts
- Touch events on mobile devices
- Themes (black and white)

3.5 Integration

- Send articles to Pinboard, Instapaper, Pocket, Wallabag, or Nunux Keeper
- Bookmarklet to subscribe to a website directly from any browsers
- Use existing mobile applications to read your feeds by using the Fever API
- REST API with clients written in Go and Python

3.6 Authentication

- Username/password
- Google (OAuth2)

3.7 Technical stuff

- Self-hosted
- Written in Go (Golang)
- Designed to run only with Postgresql
- Single static binary (no more dependency hell)

- Automatic HTTPS configuration with Let's Encrypt
- Use your own SSL certificate
- Supports HTTP/2.0 if TLS is configured
- Feeds are updated in the background by an internal scheduler
- External content is sanitized before being displayed
- Use content security policy that allows only application Javascript and block inline code and styles
- Works only in modern browsers

CHAPTER 4

Requirements

4.1 Hardware

- Rasbperry Pi, small virtual machine, platform as a service…
- Probably anything that can run Linux
- x86_64 or ARM architecture

4.2 Operating Systems

- GNU/Linux
- Darwin

4.3 Databases

- Postgresql >= 9.4

4.4 Web Browsers

A browser compatible with **ECMAScript 6 is required**.

- Mozilla Firefox
- Chrome
- Safari
- Microsoft Edge

Warning: Internet Explorer 11 is not supported.

CHAPTER 5

Installation

You can download pre-compiled Miniflux binaries and packages on the releases page: https://github.com/miniflux/miniflux/releases.

5.1 Manual Installation

1. Copy the binary somewhere
2. Make the file executable: `chmod +x miniflux`
3. Define the environment variable `DATABASE_URL` if necessary
4. `CREATE EXTENSION hstore` in the database or specify a user with `SUPERUSER` privileges. (*Details*)
5. Run the SQL migrations: `miniflux -migrate`
6. Create an admin user: `miniflux -create-admin`
7. Start the application: `miniflux`

You should configure a process manager like systemd or supervisord to supervise the Miniflux daemon. The Debian or RPM packages are doing that for you.

5.2 Debian Package Installation

You must have Debian >= 8 or Ubuntu >= 16.04. When using the Debian package, the Miniflux daemon is supervised by systemd.

1. Install Debian package: `dpkg -i miniflux_2.0.8_amd64.deb`
2. Check process status: `systemctl status miniflux`
3. Define the environment variable `DATABASE_URL` if necessary
4. Run the SQL migrations: `miniflux -migrate`

5. Create an admin user: `miniflux -create-admin`

Systemd reads the environment variables from the file `/etc/miniflux.conf`. You must restart the service to take the new values into consideration.

The files to build the Debian packages are available here: https://github.com/miniflux/package-deb.

5.3 RPM Package Installation

You must have Fedora or Centos/Redhat >= 7. When you use the RPM package, the Miniflux daemon is supervised by systemd.

1. Install Miniflux RPM: `rpm -ivh miniflux-2.0.8-1.0.x86_64.rpm`
2. Define the environment variable `DATABASE_URL` if necessary
3. Run the SQL migrations: `miniflux -migrate`
4. Create an admin user: `miniflux -create-admin`
5. Enable the systemd service: `systemctl enable miniflux`
6. Start the process with systemd: `systemctl start miniflux`
7. Check process status: `systemctl status miniflux`

Systemd reads the environment variables from the file `/etc/miniflux.conf`. You must restart the service to take the new values into consideration.

The files to build the RPM packages are available here: https://github.com/miniflux/package-rpm.

5.4 Docker Usage

Pull the image and run the container: `docker run -d -p 80:8080 miniflux/miniflux:version` (Replace version). You will probably need to pass some environment variables like the `DATABASE_URL`.

You could also use Docker Compose. Here an example of `docker-compose.yml` file:

```
version: '3'
services:
  miniflux:
    image: miniflux/miniflux:2.0.8
    ports:
      - "80:8080"
    depends_on:
      - db
    environment:
      - DATABASE_URL=postgres://miniflux:secret@db/miniflux?sslmode=disable
  db:
    image: postgres:10.1
    environment:
      - POSTGRES_USER=miniflux
      - POSTGRES_PASSWORD=secret
    volumes:
      - miniflux-db:/var/lib/postgresql/data
volumes:
  miniflux-db:
```

Remember that you still need to run the database migrations and create the first user:

```
# Run database migrations
docker exec -ti <container-name> /usr/local/bin/miniflux -migrate

# Create the first user
docker exec -ti <container-name> /usr/local/bin/miniflux -create-admin
```

The Dockerfile is available here: https://github.com/miniflux/docker.

Another way of doing the same thing is to populate the variables `RUN_MIGRATIONS`, `CREATE_ADMIN`, `ADMIN_USERNAME` and `ADMIN_PASSWORD`.

CHAPTER 6

Upgrading to a New Version

6.1 Procedure

1. Disconnect all users by flushing all sessions: `miniflux -flush-sessions`
2. Stop the process
3. Backup your database
4. Check that your backup is really working
5. Run database migrations: `miniflux -migrate`
6. Start the process

6.2 Debian Package

Follow instructions mentioned above and run: `dpkg -i miniflux_2.x.x_amd64.deb`.

6.3 RPM Package

Follow instructions mentioned above and run: `rpm -Uvh miniflux-2.x.x-1.0.x86_64.rpm`.

6.4 Docker Image

- Pull the new image with the new tag: `docker pull miniflux/miniflux:2.x.x`
- Stop and remove the old container: `docker stop <container_name> && docker rm <container_name>`

- Start a new container with the latest tag: `docker run -d -p 80:8080 miniflux/miniflux:2.x.x`

If you use Docker Compose, define the new tag in the YAML file and restart the container.

CHAPTER 7

Installation Tutorials

Here are a couple of tutorials to help you to install Miniflux (They are only examples).

7.1 Installing Miniflux on your own server

7.1.1 Ubuntu 16.04

1. Install Postgresql: `apt install postgresql`
2. Prepare the database:

```
# Switch to the postgres user
$ su - postgres

# Create a database user for Miniflux
$ createuser -P miniflux
Enter password for new role: ******
Enter it again: ******

# Create a database for miniflux that belongs to our user
$ createdb -O miniflux miniflux

# Create the extension hstore as superuser
$ psql miniflux -c 'create extension hstore'
CREATE EXTENSION
```

3. Install Miniflux:

```
# Download the latest Debian package from the release page
# In this example, this is the version 2.0.8
$ wget https://github.com/miniflux/miniflux/releases/download/2.0.8/miniflux_2.0.8_
↪amd64.deb
```

(continues on next page)

(continued from previous page)

```
# Install the package
$ dpkg -i miniflux_2.0.8_amd64.deb

# Run the SQL migrations
$ export DATABASE_URL=postgres://miniflux:secret@localhost/miniflux?sslmode=disable
$ miniflux -migrate
Current schema version: 0
Latest schema version: 16
Migrating to version: 1
Migrating to version: 2
Migrating to version: 3
[...]

# Create the first user
$ miniflux -create-admin
Enter Username: superman
Enter Password: ******

# Update the config file /etc/miniflux.conf
# Add/Edit this lines:
# DATABASE_URL=postgres://miniflux:secret@localhost/miniflux?sslmode=disable
# LISTEN_ADDR=0.0.0.0:80
$ vim /etc/miniflux.conf

# Authorize Miniflux to listen on port 80
$ setcap cap_net_bind_service=+ep /usr/bin/miniflux

# Restart the process to take the new config values into consideration
systemctl restart miniflux

# Check the logs to make sure the process is running properly
$ journalctl -u miniflux
[INFO] Starting Miniflux...
[INFO] [Worker] #0 started
[INFO] [Worker] #1 started
[INFO] [Worker] #2 started
[INFO] [Worker] #3 started
[INFO] Listening on "0.0.0.0:80" without TLS
```

4. Now, you can access to your Miniflux instance via `http://your-server/`

7.1.2 Fedora 28

Database installation and configuration:

1. Install Postgresql: `sudo dnf install -y postgresql-server postgresql-contrib`
2. Enable Postgres service: `sudo systemctl enable postgresql`
3. Initialize the database: `sudo postgresql-setup --initdb --unit postgresql`
4. Start Postgres service: `sudo systemctl start postgresql`
5. Create Miniflux database user: `sudo su - postgres` and `createuser -P miniflux`
6. Create Miniflux database: `createdb -O miniflux miniflux`
7. Create HSTORE extension: `psql miniflux -c 'create extension hstore'`

Note: More information available on the Fedora Wiki.

Miniflux installation:

1. Install RPM package:

```
sudo dnf install https://github.com/miniflux/miniflux/releases/download/2.0.8/
↪miniflux-2.0.8-1.0.x86_64.rpm
```

2. Run SQL migrations and create first user:

```
export DATABASE_URL=postgres://miniflux:secret@127.0.0.1/miniflux?sslmode=disable

# Create database structure:
miniflux -migrate

# Create frist user:
miniflux -create-admin
```

3. Start the service:

```
systemctl enable miniflux
systemctl start miniflux

# To watch the logs:
journalctl -f -u miniflux
```

5. Access your Miniflux instance via `http://your-server:8080/`

7.2 Running Miniflux with Docker Compose

You could use Docker to try quickly Miniflux on your local machine:

Create a `docker-compose.yml` file into a folder called `miniflux` for example.

```
version: '3'
services:
  miniflux:
    image: miniflux/miniflux:2.0.8
    ports:
      - "80:8080"
    depends_on:
      - db
    environment:
      - DATABASE_URL=postgres://miniflux:secret@db/miniflux?sslmode=disable
      - RUN_MIGRATIONS=1
      - CREATE_ADMIN=1
      - ADMIN_USERNAME=admin
      - ADMIN_PASSWORD=test123
  db:
    image: postgres:10.1
    environment:
      - POSTGRES_USER=miniflux
      - POSTGRES_PASSWORD=secret
    volumes:
```

(continues on next page)

(continued from previous page)

```
      - miniflux-db:/var/lib/postgresql/data
volumes:
  miniflux-db:
```

Then run `docker-compose up` and go to `http://localhost/`.

After the first user has been created, you should remove the variables `CREATE_ADMIN`, `ADMIN_USERNAME` and `ADMIN_PASSWORD`.

7.3 Deploying Miniflux on Heroku

Since the version 2.0.6, you can deploy Miniflux on Heroku in few seconds.

1. Clone the repository on your machine: `git clone https://github.com/miniflux/miniflux.git`
2. Switch to a stable version, for example `git checkout 2.0.8` (master is the development branch)
3. Create a new Heroku application: `heroku apps:create`
4. Add the Postgresql addon: `heroku addons:create heroku-postgresql:hobby-dev`
5. Add environment variables to setup the application:

```
# This parameter will create all tables in the database.
heroku config:set RUN_MIGRATIONS=1

# The following parameters will create the first user.
heroku config:set CREATE_ADMIN=1
heroku config:set ADMIN_USERNAME=admin
heroku config:set ADMIN_PASSWORD=test123
```

6. Deploy the application on Heroku: `git push heroku master`
7. After the application is installed successfully, you don't need these variables anymore:

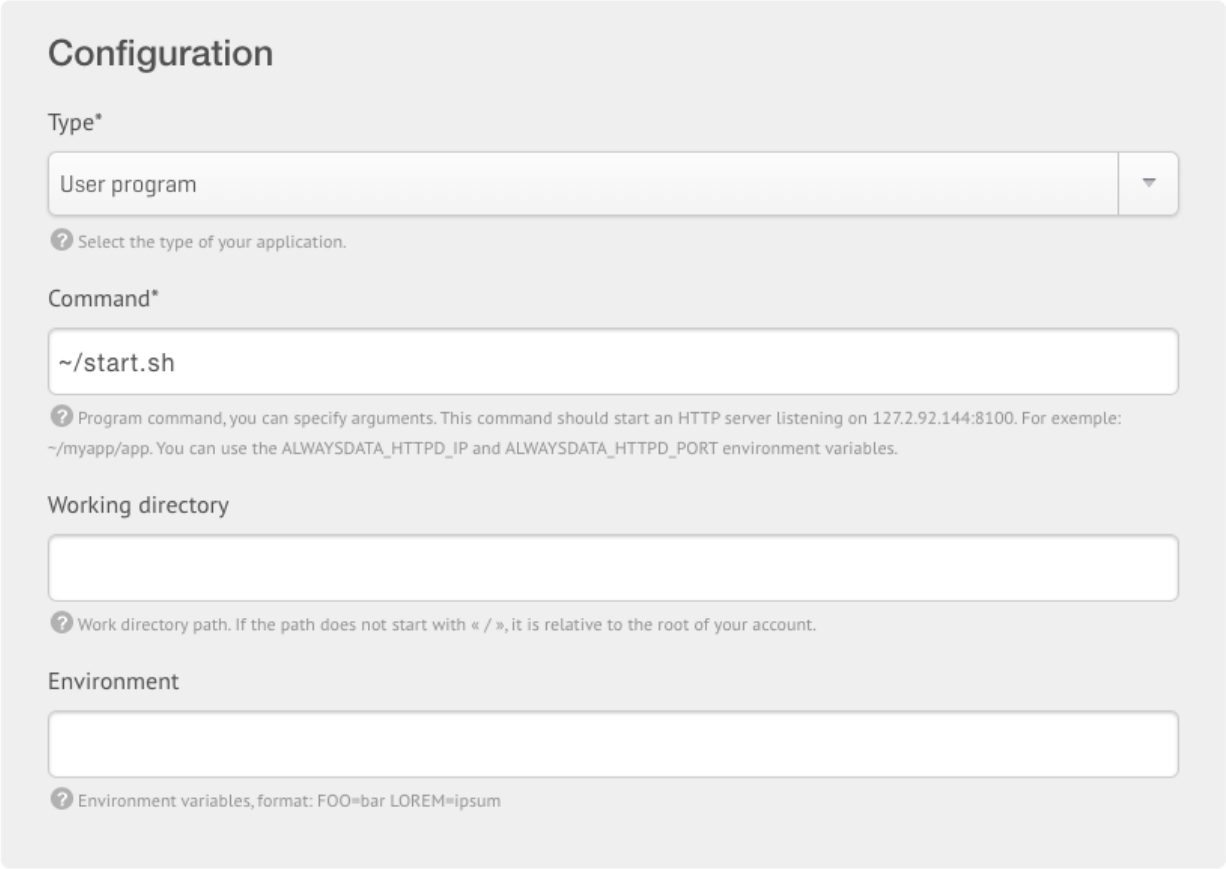
```
heroku config:unset CREATE_ADMIN
heroku config:unset ADMIN_USERNAME
heroku config:unset ADMIN_PASSWORD
```

- To watch the logs, use `heroku logs`.
- You can also run a one-off container to run the commands manually: `heroku run bash`. The Miniflux binary will be located into the folder `bin`.
- To update Miniflux, pull the new version from the repository and push to Heroku again.

7.4 Deploying Miniflux on AlwaysData

AlwaysData is a French shared hosting provider. You can install Miniflux in few minutes on their platform.

1. Open an account
2. Via the admin panel, create a Postgresql database and define a user/password
3. Create a website, choose "User Program", use a custom shell-script, for example `~/start.sh`



4. Enable the SSH access and open a session *ssh account@ssh-account.alwaysdata.net*
5. Install Miniflux:

```
wget https://github.com/miniflux/miniflux/releases/download/2.0.8/miniflux-linux-amd64
mv miniflux-linux-amd64 miniflux
chmod +x miniflux
```

6. Create a shell script to start miniflux, let's call it `start.sh`:

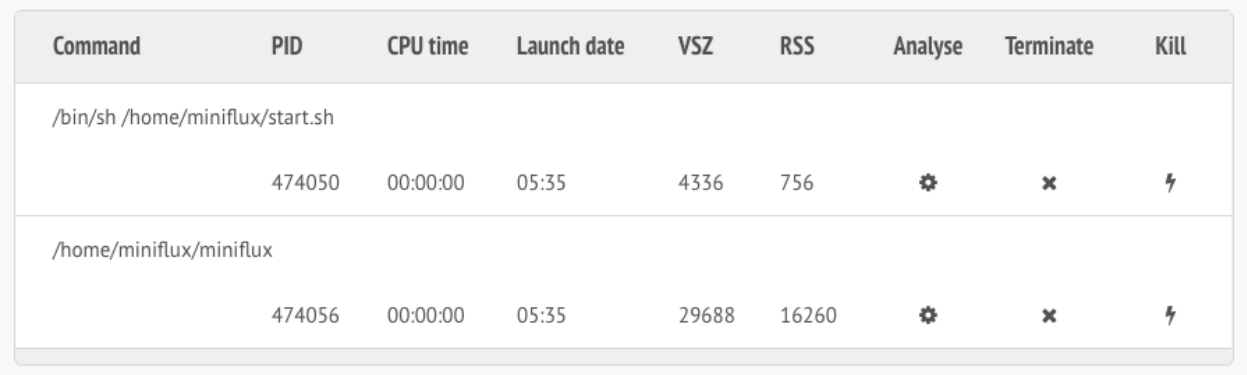
```
#!/bin/sh

export LISTEN_ADDR=$ALWAYSDATA_HTTPD_IP:$ALWAYSDATA_HTTPD_PORT
export DATABASE_URL="host=postgresql-xxxxx.alwaysdata.net dbname=xxxx user=xxxx␣
↪password=xxx sslmode=disable"

~/miniflux
```

7. Make the script executable: `chmod +x start.sh`
8. Run the db migrations and a create the first user:

```
export DATABASE_URL=".... replace me...."
./miniflux -migrate
./miniflux -create-admin
```

9. Go to `https://your-account.alwaysdata.net`

Via the admin panel, in Advanced > Processes, you can even see the Miniflux process running:



Command	PID	CPU time	Launch date	VSZ	RSS	Analyse	Terminate	Kill
/bin/sh /home/miniflux/start.sh								
	474050	00:00:00	05:35	4336	756			
/home/miniflux/miniflux								
	474056	00:00:00	05:35	29688	16260			

CHAPTER 8

Configuration

Miniflux doesn't use any config file, **only environment variables**.

- `DEBUG`: Toggle debug output (default is off)
- `WORKER_POOL_SIZE`: Number of background processes (default=5)
- `POLLING_FREQUENCY`: Refresh interval in minutes for feeds (default=60)
- `BATCH_SIZE`: Number of feeds to send to the queue for each interval (default=10)
- `DATABASE_URL`: Connection URL to connect to Postgresql (default=postgres://postgres:postgres@localhost/miniflux2?sslmode=
- `DATABASE_MAX_CONNS`: Number of concurrent database connections (default=20)
- `LISTEN_ADDR`: HTTP server address (default=127.0.0.1:8080)
- `BASE_URL`: Base URL (default=http://localhost/)
- `CLEANUP_FREQUENCY`: Cleanup job frequency, remove old sessions and archive read entries (Default is 24 hours)
- `HTTPS`: Force cookies to use secure flag (default is empty, try to detect automatically if HTTPS is used)
- `CERT_FILE`: SSL certificate (default="")
- `KEY_FILE`: SSL private key (default="")
- `CERT_DOMAIN`: Use Let's Encrypt to configure automatically a certificate for this domain (default="")
- `CERT_CACHE`: Let's Encrypt cache directory (default="/tmp/cert_cache")
- `OAUTH2_PROVIDER`: OAuth2 provider to use, at this time only "google" is supported (default="")
- `OAUTH2_CLIENT_ID`: OAuth2 client ID (default="")
- `OAUTH2_CLIENT_SECRET`: OAuth2 client secret (default="")
- `OAUTH2_REDIRECT_URL`: OAuth2 redirect URL (default="")
- `OAUTH2_USER_CREATION`: Set to 1 to authorize user creation (default=0)
- `DISABLE_HSTS`: Disable HTTP Strict Transport Security header (enabled by default if HTTPS)

- RUN_MIGRATIONS: Run database migrations, set to value to 1 (default="")
- CREATE_ADMIN: Create admin user automatically, set value to 1 (default="")
- ADMIN_USERNAME: Admin user login, used only if CREATE_ADMIN is enabled (default="")
- ADMIN_PASSWORD: Admin user password, used only if CREATE_ADMIN is enabled (default="")
- POCKET_CONSUMER_KEY: Pocket consumer API key for all users (default="")

8.1 Database Connection Parameters

Miniflux use the Golang library pq to communicate with Postgres. Connection parameters are available on this page.

The default value for DATABASE_URL is postgres://postgres:postgres@localhost/miniflux2?sslmode=disable. You don't necessary need to use the URL format.

If you would like to connect via a Unix socket, you could do:

```
export DATABASE_URL="user=postgres password=postgres dbname=miniflux2 sslmode=disable␣
↪host=/path/to/socket/folder"
./miniflux
```

Warning: Password that contains special characters like ^ might be rejected since Miniflux 2.0.3. Golang v1.10 is now validating the password and will return this error: net/url: invalid userinfo. To avoid this issue, do not use the URL format for DATABASE_URL or make sure the password is URL encoded.

8.2 Running Miniflux on port 443 or 80

Ports less than 1024 are reserved for privileged users. If you have installed Miniflux with the RPM or Debian package, systemd run the process as the *miniflux* user.

To give Miniflux the ability to bind to privileged ports as a non-root user, add the capability *CAP_NET_BIND_SERVICE* to the binary:

```
setcap cap_net_bind_service=+ep /usr/bin/miniflux
```

Check that the capability is added:

```
getcap /usr/bin/miniflux
/usr/bin/miniflux = cap_net_bind_service+ep
```

Here, we assume you installed the Miniflux binary into /usr/bin.

8.3 Let's Encrypt Integration

You could use Let's Encrypt to handle the SSL certificate automatically and activate HTTP/2.0.

```
export CERT_DOMAIN=my.domain.tld
miniflux
```

- Your server must be reachable publicly on port 443 and port 80 (http-01 challenge)

- In this mode, `LISTEN_ADDR` is automatically set to `:https`
- A cache directory is required, by default `/tmp/cert_cache` is used, it could be overrided by using the variable `CERT_CACHE`

Note: Miniflux supports http-01 challenge since the version 2.0.2

8.4 Manual HTTPS Configuration

Here an example to generate your self-signed certificate:

```
# Generate the private key:
openssl genrsa -out server.key 2048
openssl ecparam -genkey -name secp384r1 -out server.key

# Generate the certificate:
openssl req -new -x509 -sha256 -key server.key -out server.crt -days 3650
```

Start the server like this:

```
# Configure the environment variables:
export CERT_FILE=/path/to/server.crt
export KEY_FILE=/path/to/server.key
export LISTEN_ADDR=":https"

# Start the server:
miniflux
```

Then you can access to your server by using an encrypted connection with the HTTP/2.0 protocol.

8.5 OAuth2 Authentication

OAuth2 allows you to sign in with an external provider. At this time, only Google is supported.

8.5.1 Google

1. Create a new project in Google Console
2. Create a new OAuth2 client
3. Set an authorized redirect URL: `https://my.domain.tld/oauth2/google/callback`
4. Define the OAuth2 environment variables and start the process

```
export OAUTH2_PROVIDER=google
export OAUTH2_CLIENT_ID=replace_me
export OAUTH2_CLIENT_SECRET=replace_me
export OAUTH2_REDIRECT_URL=https://my.domain.tld/oauth2/google/callback

miniflux
```

Now from the settings page, you can link your existing user to your Google account.

If you would like to authorize anyone to create user account, you must set `OAUTH2_USER_CREATION=1`. Since Google do not have the concept of username, the email address is used as username.

8.6 Reverse-Proxy Configuration with Subfolder

Since the version 2.0.2, you can host your Miniflux instance under a subfolder.

You must define the environment variable `BASE_URL` for Miniflux, for example:

```
export BASE_URL=http://example.org/rss/
```

You can use the reverse-proxy software of your choice, here an example with Nginx:

```
location /rss/ {
    proxy_pass http://127.0.0.1:8080/rss/;
    proxy_set_header Host $host;
    proxy_redirect off;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
}
```

This example assumes that you are running the Miniflux daemon on *127.0.0.1:8080*.

Now you can access your Miniflux instance at *http://example.org/rss/*. In this configuration, cookies are using the path */rss*.

CHAPTER 9

Command Line Usage

9.1 Show Version

```
miniflux -version
2.0.0
```

9.2 Show Build Information

```
miniflux -info
Version: 2.0.0
Build Date: 2017-11-20T22:45:00
Go Version: go1.9
```

9.3 Enable Debug Mode

```
miniflux -debug
[2018-02-05T02:38:32] [INFO] Debug mode enabled
[2018-02-05T02:38:32] [INFO] Starting Miniflux...
```

9.4 Run Database Migrations

```
export DATABASE_URL=replace_me

miniflux -migrate
Current schema version: 0
```

(continues on next page)

(continued from previous page)

```
Latest schema version: 12
Migrating to version: 1
Migrating to version: 2
Migrating to version: 3
Migrating to version: 4
Migrating to version: 5
Migrating to version: 6
Migrating to version: 7
Migrating to version: 8
Migrating to version: 9
Migrating to version: 10
Migrating to version: 11
Migrating to version: 12
```

When you run the migrations, make sure that all Miniflux processes are stopped.

Creating extensions requires `SUPERUSER` privileges. Several solutions are available:

1. Give `SUPERUSER` privileges to miniflux user only during the schema migration:

```
ALTER USER miniflux WITH SUPERUSER;
-- Run the migrations (miniflux -migrate)
ALTER USER miniflux WITH NOSUPERUSER;
```

2. You could create the hstore extension as a postgres user with `SUPERUSER` privileges before to run the migrations.

```
sudo -u postgres psql $MINIFLUX_DATABASE
> CREATE EXTENSION hstore;
```

Warning: Password that contains special characters like ^ might be rejected since Miniflux 2.0.3. Golang v1.10 is now validating the password and will return this error: `net/url: invalid userinfo`. To avoid this issue, do not use the URL format for `DATABASE_URL` or make sure the password is URL encoded.

9.5 Create Admin User

```
miniflux -create-admin
Enter Username: root
Enter Password:
```

9.6 Reset User Password

```
miniflux -reset-password
Enter Username: myusername
Enter Password: ****
```

9.7 Flush all Sessions

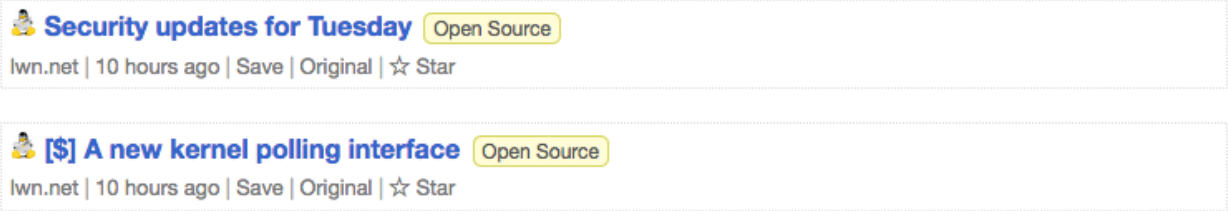
Flushing all sessions disconnect all users.

```
miniflux -flush-sessions
Flushing all sessions (disconnect users)
```

CHAPTER 10

User Interface Usage

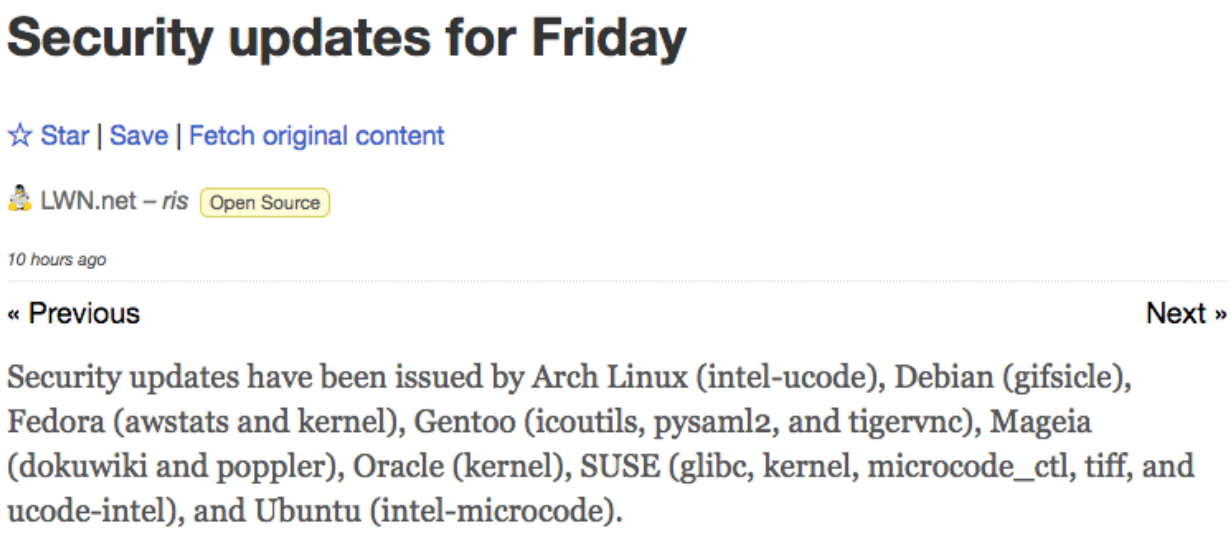
10.1 List View



- **Save**: Send an article to third-party services if enabled (Pinboard, Instapaper, etc…)
- **Star/Unstar**: Add/Remove entry to/from bookmarks
- **Original**: Open original entry link in a new tab

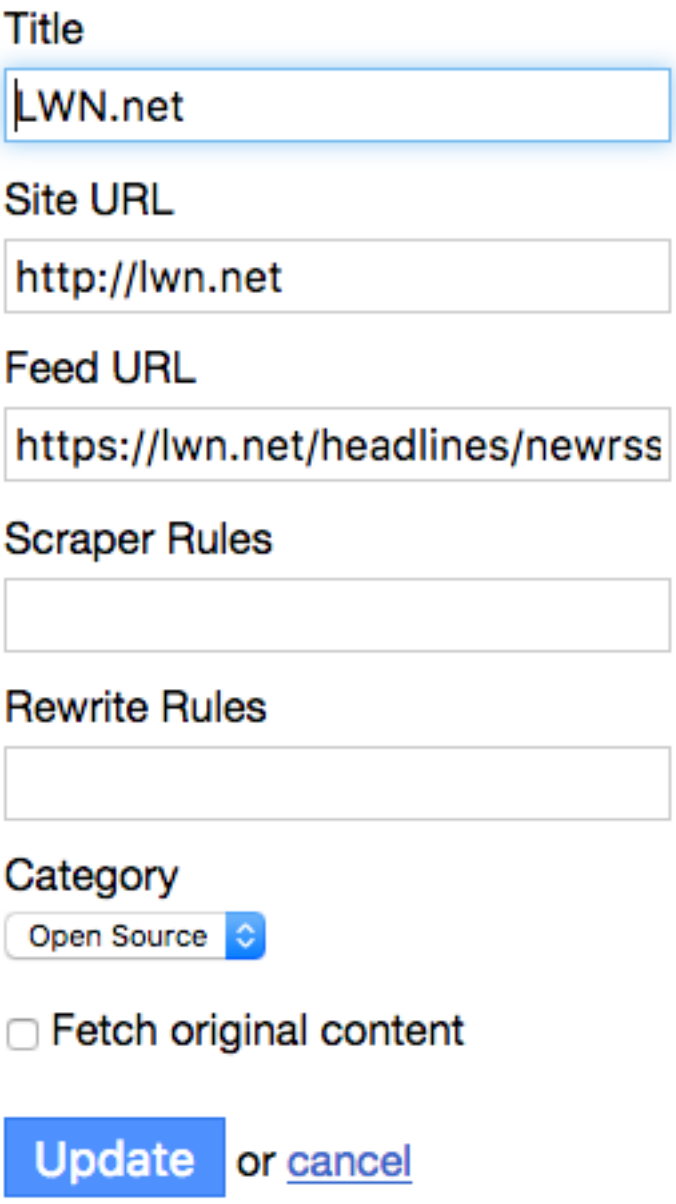
Note: Since the version 2.0.7, the link "Save" will be shown only if at least one integration is configured.

10.2 Article View



- **Fetch original content**: Download the original web page and try to find relevant contents

10.3 Edit Feed



- **Scraper Rules**: CSS selectors to use when fetching the original web page (Use Readability if nothing provided)
- **Rewrite Rules**: Name of the rewrite rules to use to alter item content
- **Fetch original content**: Always download original articles (consume more resources)

CHAPTER 11

Keyboard Shortcuts

Sections navigation:

- `g u`: Go to unread page
- `g b`: Go to bookmark page
- `g h`: Go to the history page
- `g f`: Go to feed page
- `g c`: Go to categories page
- `g s`: Go to settings page
- `?`: Show keyboard shortcuts help

Items navigation:

- `Left Arrow`: Go to previous item
- `j`: Go to previous item
- `p`: Go to previous item
- `Right Arrow`: Go to next item
- `k`: Go to next item
- `n`: Go to next item

Pages navigation:

- `h`: Go to previous page
- `l`: Go to next page

Actions:

- `o`: Open selected item
- `v`: Open original link to a new tab
- `m`: Mark selected item as read/unread

- `A`: Mark current page as read
- `d`: Fetch original web page and apply scraper rules or readability
- `f`: Star or unstar the current entry
- `Escape`: Close modal dialogs

Note: Mozilla Firefox will block the opening of a new tab if you use the shortcut `v`. You must authorize Miniflux to open new tabs in your Firefox settings.

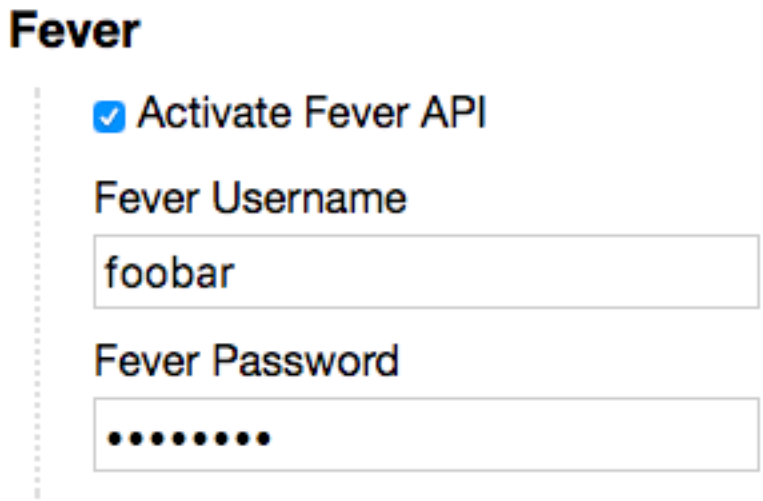
CHAPTER 12

Integration with External Services

12.1 Fever API

Miniflux implements the Fever API in addition to its own REST API. The Fever API allows you to use existing mobile applications to read your feeds instead of the web user interface.

To activate the Fever API, go into the integration section and choose a username/password.



The API endpoint is `https://example.org/fever/`

12.1.1 Compatible Apps

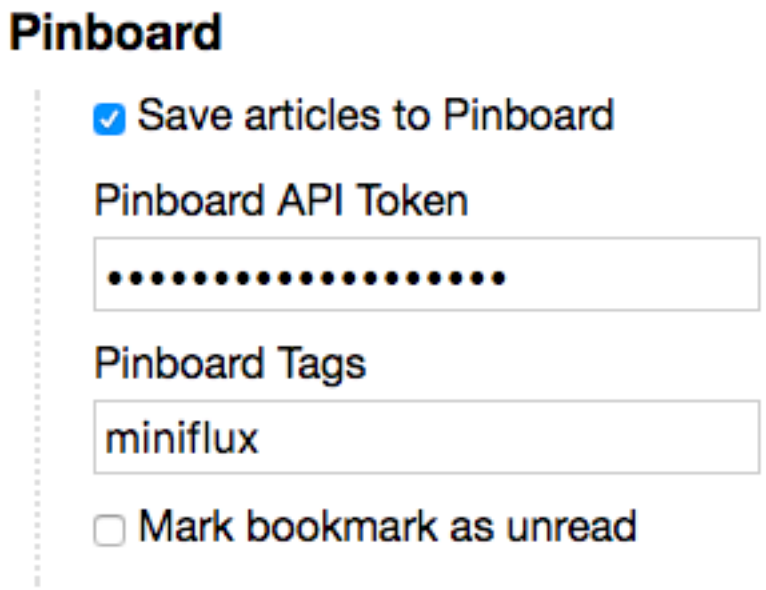
- Reeder (iOS/Mac OS)
- Unread (iOS)

Note:

- Saving an entry will add a new bookmark and save the article
- Only the JSON format is supported
- Refreshing feeds is not possible with Reeder because no user information is sent
- Links, sparks, and kindlings are not supported

12.2 Pinboard

You could save articles to Pinboard.



To activate this service, go to the integration section and enter your Pinboard API credentials. You must use the API token, not your password.

12.3 Instapaper

You could save articles to Instapaper.

To activate this service, go to the integration section and enter your Instapaper credentials.

12.4 Pocket

12.4.1 Configuration

To configure the Pocket integration on your own Miniflux instance, you need to create an application on Pocket's website.

Go to https://getpocket.com/developer/apps/new and create a new application.

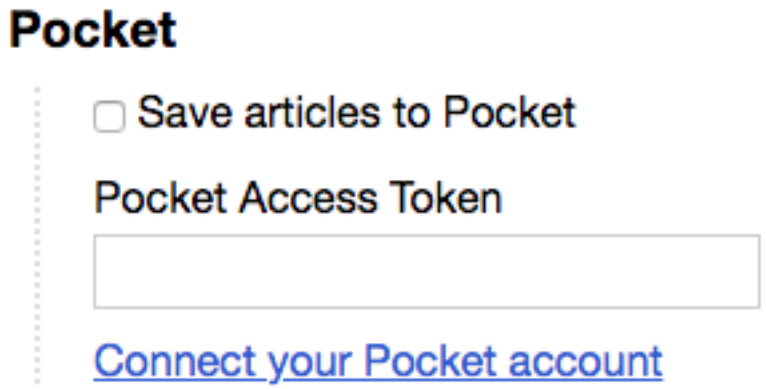
- You can define the *Pocket Consumer Key* for all users by using the environment variable `POCKET_CONSUMER_KEY`.
- Or, you can set the *Pocket Consumer Key* only for your user by using the form.

If the environment variable is defined, the text field for the *Pocket Consumer API key* will be hidden.

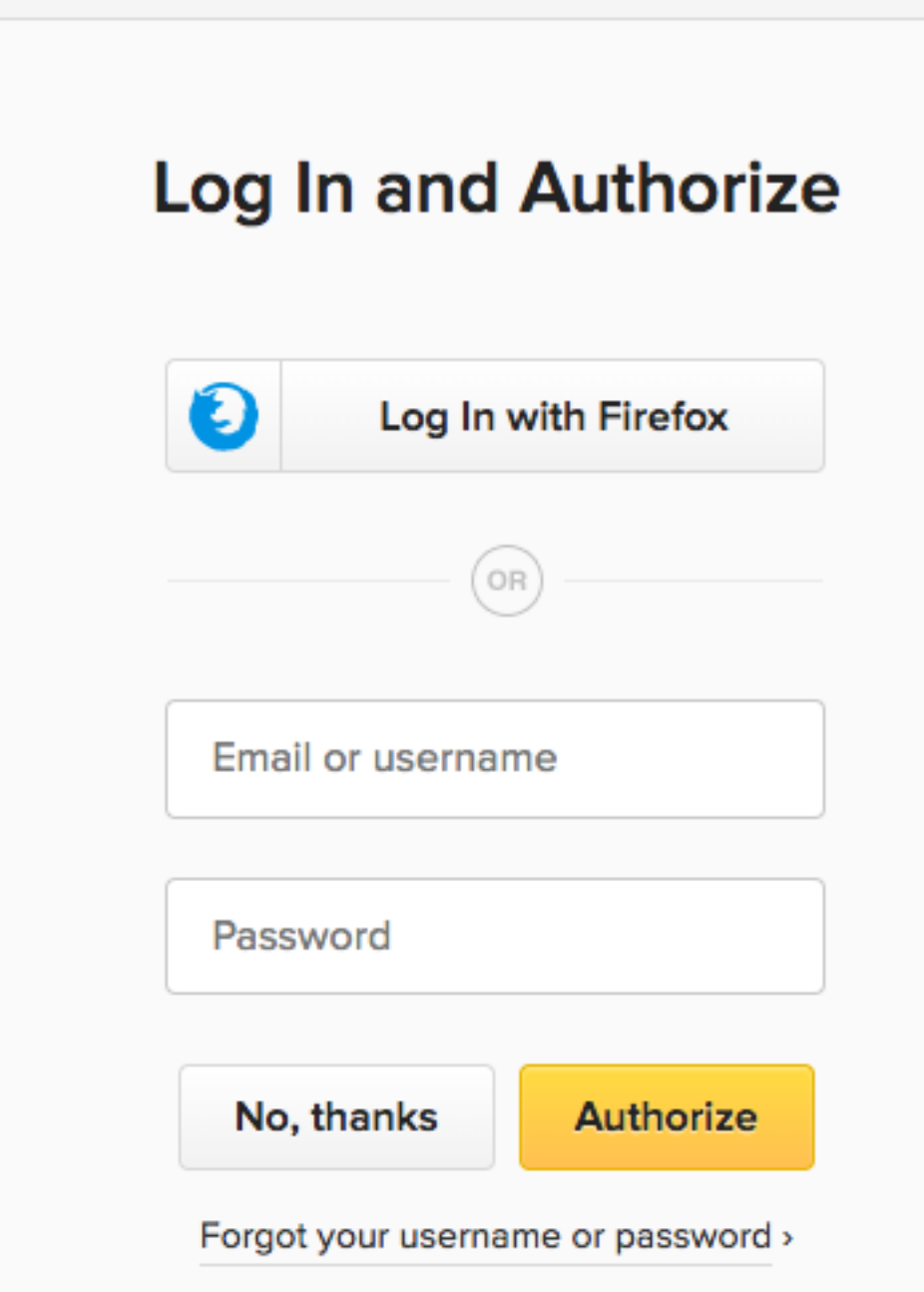
Make sure the environment variable `BASE_URL` is defined properly to allow the authorization flow to work afterward.

12.4.2 Usage

Once the consumer key is configured, you need to get a *Pocket Access Token*. This token could be fetched automatically by using the authorization flow or manually by making some HTTP calls.



The simplest solution is to use the link, **"Connect your Pocket account"**. This method redirects the end user to Pocket's website and ask for authorization.

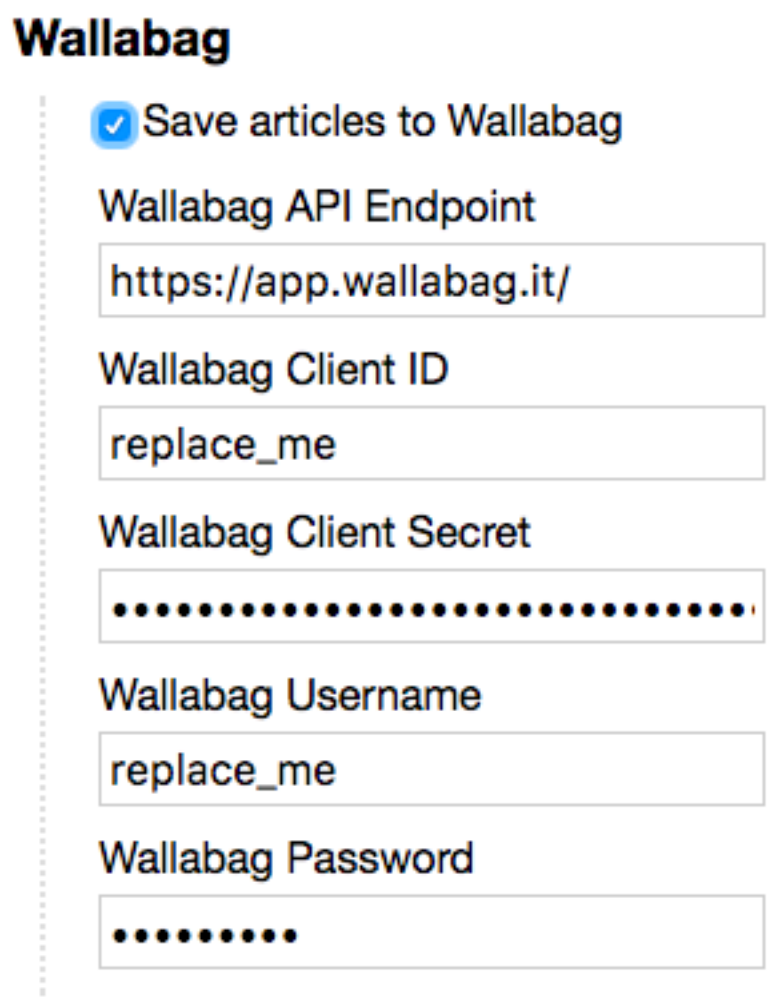


After the authorization is given, Miniflux will fetch the *Pocket Access Token* for you.

If you prefer to fetch *Pocket Access Token* manually, the process is described in Pocket's developer documentation.

12.5 Wallabag

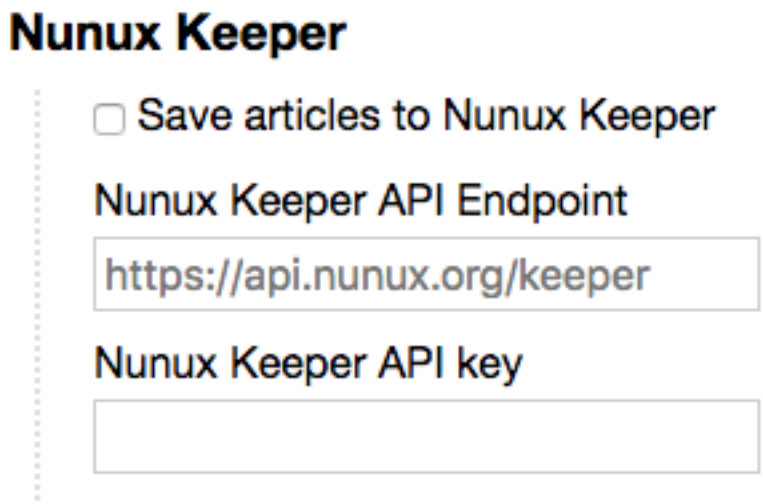
Wallabag is a self-hosted application for saving web pages.



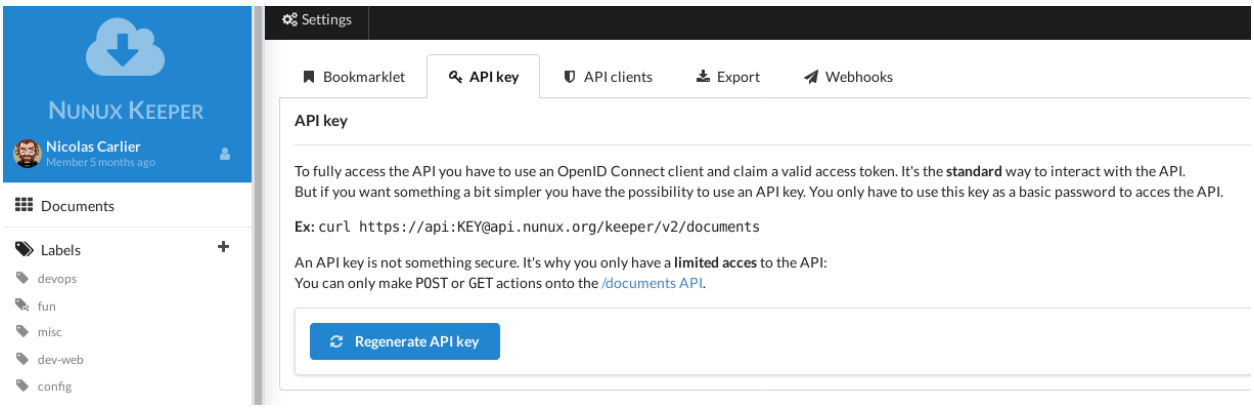
- The API URL is the root URL of your instance, for example, if you have the hosted version use: `https://app.wallabag.it/`.
- To create a new API client, go to the section "API clients management" and choose "Create a new client".

12.6 Nunux Keeper

Nunux Keeper is a *"personal content curation service"*. It's an alternative to Pocket or Wallabag.



- The API URL is the root URL of your instance, for example, if you are using the hosted version: `https://api.nunux.org/keeper`.
- To create a new API key, go to the settings tab: "API key" and choose "Regenerate API key".



CHAPTER 13

Scraper Rules

When an article contains only an extract of the content, you could fetch the original web page and apply a set of rules to get relevant contents.

Miniflux uses CSS selectors for custom rules. These custom rules can be saved in the feed properties (Select a feed and click on edit).

13.1 Examples

- `div#articleBody`: Fetch a *div* element with the ID *articleBody*
- `div.content`: Fetch all *div* elements with the class *content*
- `article, div.article`: Use a comma to define multiple rules

Miniflux includes a list of predefined rules for popular websites. You could contribute to the project to keep them up to date.

Under the hood, Miniflux uses the library Goquery.

CHAPTER 14

Rewrite Rules

To improve the reading experience, it's possible to alter the content of feed items.

For example, if you are reading a popular comic website like XKCD, it's nice to have to image title (the *alt* attribute) added under the image. Especially on mobile devices when there is no hover event.

14.1 List of Rules

- `add_image_title`: Get the first image and add the title under the image.
- `add_youtube_video`: Insert Youtube video inside the article (automatic for Youtube.com)

14.2 Adding Rewrite Rules

Miniflux includes a set of default rules for some websites, but you could define your own rules.

On the feed edit page, enter your custom rules in the field "Rewrite Rules" like this:

```
rule1,rule2
```

Separate each rule by a comma. As of now, only `add_image_title` is available. Of course, other rules could be added later.

CHAPTER 15

REST API

15.1 Authentication

The Miniflux API uses HTTP Basic authentication. The credentials are the username/password of your account.

15.2 Clients

There are 2 official API clients, one written in Go and another one written in Python.

15.2.1 Golang Client

- Repository: https://github.com/miniflux/miniflux-go
- Reference: https://godoc.org/github.com/miniflux/miniflux-go

Installation:

```
go get -u github.com/miniflux/miniflux-go
```

Usage Example:

```
package main

import (
    "fmt"

    "github.com/miniflux/miniflux-go"
)

func main() {
    client := miniflux.NewClient("https://miniflux.example.org", "admin", "secret")
```

(continues on next page)

(continued from previous page)

```
    // Fetch all feeds.
    feeds, err := client.Feeds()
    if err != nil {
        fmt.Println(err)
        return
    }
    fmt.Println(feeds)
}
```

15.2.2 Python Client

- Repository: https://github.com/miniflux/miniflux-python
- PyPi: https://pypi.org/project/miniflux/

Installation:

```
pip install miniflux
```

Usage example:

```
import miniflux

client = miniflux.Client("https://miniflux.example.org", "my_username", "my_secret_
↪password")

# Get all feeds
feeds = client.get_feeds()

# Refresh a feed
client.refresh_feed(123)

# Discover subscriptions from a website
subscriptions = client.discover("https://example.org")
```

List of `Client` methods:

- `me()`
- `export_feeds()`
- `import_feeds(opml)`
- `discover(website_url)`
- `get_feeds()`
- `get_feed(feed_id)`
- `get_feed_icon(feed_id)`
- `create_feed(feed_url, category_id)`
- `update_feed(feed_id, title=None, feed_url=None, site_url=None, scraper_rules=None, rewrite_rules=None, crawler=None, category_id=None)`
- `refresh_feed(feed_id)`
- `delete_feed(feed_id)`

- `get_feed_entry(feed_id, entry_id)`
- `get_feed_entries(feed_id, status=None, offset=None, limit=None, order=None, direction=None)`
- `get_entry(entry_id)`
- `get_entries(status=None, offset=None, limit=None, order=None, direction=None)`
- `update_entries(entry_ids, status)`
- `toggle_bookmark(entry_id)`
- `get_categories()`
- `create_category(title)`
- `update_category(category_id, title)`
- `delete_category(category_id)`
- `get_users()`
- `get_user_by_id(user_id)`
- `get_user_by_username(username)`
- `create_user(username, password, is_admin)`
- `update_user(user_id, username=None, password=None, theme=None, language=None, timezone=None, entry_direction=None)`
- `delete_user(user_id)`

15.3 API Reference

15.3.1 Status Codes

- `200`: Everything is OK
- `201`: Resource created/modified
- `204`: Resource removed/modified
- `400`: Bad request
- `401`: Unauthorized (bad username/password)
- `403`: Forbidden (access not allowed)
- `500`: Internal server error

15.3.2 Error Response

```
{
    "error_message": "Some error"
}
```

15.3.3 Discover Subscriptions

Request:

```
POST /v1/discover
Content-Type: application/json

{
    "url": "http://example.org"
}
```

Response:

```
[
    {
        "url": "http://example.org/feed.atom",
        "title": "Atom Feed",
        "type": "atom"
    },
    {
        "url": "http://example.org/feed.rss",
        "title": "RSS Feed",
        "type": "rss"
    }
]
```

15.3.4 Get Feeds

Request:

```
GET /v1/feeds
```

Response:

```
[
    {
        "id": 42,
        "user_id": 123,
        "title": "Example Feed",
        "site_url": "http://example.org",
        "feed_url": "http://example.org/feed.atom",
        "rewrite_rules": "",
        "scraper_rules": "",
        "crawler": false,
        "checked_at": "2017-12-22T21:06:03.133839-05:00",
        "etag_header": "KyLxEflwnTGF5ecaiqZ2G0TxBCc",
        "last_modified_header": "Sat, 23 Dec 2017 01:04:21 GMT",
        "parsing_error_count": 0,
        "parsing_error_message": "",
        "category": {
            "id": 793,
            "user_id": 123,
            "title": "Some category"
        },
        "icon": {
            "feed_id": 42,
```

(continues on next page)

(continued from previous page)

```
            "icon_id": 84
        }
    }
]
```

Notes:

- `icon` is `null` when the feed doesn't have any favicon.

15.3.5 Get Feed

Request:

```
GET /v1/feeds/42
```

Response:

```
{
    "id": 42,
    "user_id": 123,
    "title": "Example Feed",
    "site_url": "http://example.org",
    "feed_url": "http://example.org/feed.atom",
    "rewrite_rules": "",
    "scraper_rules": "",
    "crawler": false,
    "checked_at": "2017-12-22T21:06:03.133839-05:00",
    "etag_header": "KyLxEflwnTGF5ecaiqZ2G0TxBCc",
    "last_modified_header": "Sat, 23 Dec 2017 01:04:21 GMT",
    "parsing_error_count": 0,
    "parsing_error_message": "",
    "category": {
        "id": 793,
        "user_id": 123,
        "title": "Some category"
    },
    "icon": {
        "feed_id": 42,
        "icon_id": 84
    }
}
```

Notes:

- `icon` is `null` when the feed doesn't have any favicon.

15.3.6 Get Feed Icon

Request:

```
GET /v1/feeds/42/icon
```

Response:

```
{
    "id": 262,
    "data": "image/png;base64,iVBORw0KGgoAAA....",
    "mime_type": "image/png"
}
```

Notes:

- If the feed doesn't have any favicon, a 404 is returned.

15.3.7 Create Feed

Request:

```
POST /v1/feeds
Content-Type: application/json

{
    "feed_url": "http://example.org/feed.atom",
    "category_id": 22
}
```

Response:

```
{
    "feed_id": 262,
}
```

15.3.8 Update Feed

Request:

```
PUT /v1/feeds/42
Content-Type: application/json

{
    "title": "New Feed Title",
    "category": {
        "id": 22
    }
}
```

Response:

```
{
    "id": 42,
    "user_id": 123,
    "title": "New Feed Title",
    "site_url": "http://example.org",
    "feed_url": "http://example.org/feed.atom",
    "rewrite_rules": "",
    "scraper_rules": "",
    "crawler": false,
    "checked_at": "2017-12-22T21:06:03.133839-05:00",
    "etag_header": "KyLxEflwnTGF5ecaiqZ2G0TxBCc",
```

(continues on next page)

(continued from previous page)

```
    "last_modified_header": "Sat, 23 Dec 2017 01:04:21 GMT",
    "parsing_error_count": 0,
    "parsing_error_message": "",
    "category": {
        "id": 22,
        "user_id": 123,
        "title": "Another category"
    },
    "icon": {
        "feed_id": 42,
        "icon_id": 84
    }
}
```

15.3.9 Refresh Feed

Request:

```
PUT /v1/feeds/42/refresh
```

Note:

- Returns `204` status code for success.
- This API call is synchronous and can takes hundred of milliseconds.

15.3.10 Remove Feed

Request:

```
DELETE /v1/feeds/42
```

15.3.11 Get Feed Entry

Request:

```
GET /v1/feeds/42/entries/888
```

Response:

```
{
    "id": 888,
    "user_id": 123,
    "feed_id": 42,
    "title": "Entry Title",
    "url": "http://example.org/article.html",
    "comments_url": "",
    "author": "Foobar",
    "content": "<p>HTML contents</p>",
    "hash": "29f99e4074cdacca1766f47697d03c66070ef6a14770a1fd5a867483c207a1bb",
```

(continues on next page)

(continued from previous page)

```
    "published_at": "2016-12-12T16:15:19Z",
    "status": "read",
    "starred": false,
    "feed": {
        "id": 42,
        "user_id": 123,
        "title": "New Feed Title",
        "site_url": "http://example.org",
        "feed_url": "http://example.org/feed.atom",
        "rewrite_rules": "",
        "scraper_rules": "",
        "crawler": false,
        "checked_at": "2017-12-22T21:06:03.133839-05:00",
        "etag_header": "KyLxEflwnTGF5ecaiqZ2G0TxBCc",
        "last_modified_header": "Sat, 23 Dec 2017 01:04:21 GMT",
        "parsing_error_count": 0,
        "parsing_error_message": "",
        "category": {
            "id": 22,
            "user_id": 123,
            "title": "Another category"
        },
        "icon": {
            "feed_id": 42,
            "icon_id": 84
        }
    }
}
```

Note:

- The field `comments_url` is available since Miniflux v2.0.5.

15.3.12 Get Entry

Request:

```
GET /v1/entries/888
```

Response:

```
{
    "id": 888,
    "user_id": 123,
    "feed_id": 42,
    "title": "Entry Title",
    "url": "http://example.org/article.html",
    "comments_url": "",
    "author": "Foobar",
    "content": "<p>HTML contents</p>",
    "hash": "29f99e4074cdacca1766f47697d03c66070ef6a14770a1fd5a867483c207a1bb",
    "published_at": "2016-12-12T16:15:19Z",
    "status": "read",
```

(continues on next page)

(continued from previous page)


```
    "starred": false,
    "feed": {
        "id": 42,
        "user_id": 123,
        "title": "New Feed Title",
        "site_url": "http://example.org",
        "feed_url": "http://example.org/feed.atom",
        "rewrite_rules": "",
        "scraper_rules": "",
        "crawler": false,
        "checked_at": "2017-12-22T21:06:03.133839-05:00",
        "etag_header": "KyLxEflwnTGF5ecaiqZ2G0TxBCc",
        "last_modified_header": "Sat, 23 Dec 2017 01:04:21 GMT",
        "parsing_error_count": 0,
        "parsing_error_message": "",
        "category": {
            "id": 22,
            "user_id": 123,
            "title": "Another category"
        },
        "icon": {
            "feed_id": 42,
            "icon_id": 84
        }
    }
}
```


15.3.13 Get Feed Entries

Request:

```
GET /v1/feeds/42/entries?limit=1&order=id&direction=asc
```

Available filters:

- `status`: Entry status (read, unread or removed)
- `offset`
- `limit`
- `order`: “id”, “status”, “published_at”, “category_title”, “category_id”
- `direction`: “asc” or “desc”

Response:


```
{
    "total": 10,
    "entries": [
        {
            "id": 888,
            "user_id": 123,
            "feed_id": 42,
            "title": "Entry Title",
            "url": "http://example.org/article.html",
            "comments_url": "",
```


(continues on next page)

(continued from previous page)

```
            "author": "Foobar",
            "content": "<p>HTML contents</p>",
            "hash": "29f99e4074cdacca1766f47697d03c66070ef6a14770a1fd5a867483c207a1bb
↪",
            "published_at": "2016-12-12T16:15:19Z",
            "status": "read",
            "starred": false,
            "feed": {
                "id": 42,
                "user_id": 123,
                "title": "New Feed Title",
                "site_url": "http://example.org",
                "feed_url": "http://example.org/feed.atom",
                "rewrite_rules": "",
                "scraper_rules": "",
                "crawler": false,
                "checked_at": "2017-12-22T21:06:03.133839-05:00",
                "etag_header": "KyLxEflwnTGF5ecaiqZ2G0TxBCc",
                "last_modified_header": "Sat, 23 Dec 2017 01:04:21 GMT",
                "parsing_error_count": 0,
                "parsing_error_message": "",
                "category": {
                    "id": 22,
                    "user_id": 123,
                    "title": "Another category"
                },
                "icon": {
                    "feed_id": 42,
                    "icon_id": 84
                }
            }
        }
    ]
```

15.3.14 Get Entries

Request:

```
GET /v1/entries?status=unread&direction=desc
```

Available filters:

- `status`: Entry status (read, unread or removed)
- `offset`
- `limit`
- `order`: “id”, “status”, “published_at”, “category_title”, “category_id”
- `direction`: “asc” or “desc”

Response:

```
{
    "total": 10,
    "entries": [
```

(continues on next page)

(continued from previous page)

```
        {
            "id": 888,
            "user_id": 123,
            "feed_id": 42,
            "title": "Entry Title",
            "url": "http://example.org/article.html",
            "comments_url": "",
            "author": "Foobar",
            "content": "<p>HTML contents</p>",
            "hash": "29f99e4074cdacca1766f47697d03c66070ef6a14770a1fd5a867483c207a1bb
↪",
            "published_at": "2016-12-12T16:15:19Z",
            "status": "unread",
            "starred": false,
            "feed": {
                "id": 42,
                "user_id": 123,
                "title": "New Feed Title",
                "site_url": "http://example.org",
                "feed_url": "http://example.org/feed.atom",
                "rewrite_rules": "",
                "scraper_rules": "",
                "crawler": false,
                "checked_at": "2017-12-22T21:06:03.133839-05:00",
                "etag_header": "KyLxEflwnTGF5ecaiqZ2G0TxBCc",
                "last_modified_header": "Sat, 23 Dec 2017 01:04:21 GMT",
                "parsing_error_count": 0,
                "parsing_error_message": "",
                "category": {
                    "id": 22,
                    "user_id": 123,
                    "title": "Another category"
                },
                "icon": {
                    "feed_id": 42,
                    "icon_id": 84
                }
            }
        }
    ]
```

15.3.15 Update Entries

Request:

```
PUT /v1/entries
Content-Type: application/json

{
    "entry_ids": [1234, 4567],
    "status": "read"
}
```

Note:

- Returns `204` status code for success.

15.3.16 Toggle Entry Bookmark

Request:

```
PUT /v1/entries/1234/bookmark
```

Note:

- Returns `204` status code for success.

15.3.17 Get Categories

Request:

```
GET /v1/categories
```

Response:

```
[
    {"title": "All", "user_id": 267, "id": 792},
    {"title": "Engineering Blogs", "user_id": 267, "id": 793}
]
```

15.3.18 Create Category

Request:

```
POST /v1/categories
Content-Type: application/json

{
    "title": "My category"
}
```

Response:

```
{
    "id": 802,
    "user_id": 267,
    "title": "My category"
}
```

15.3.19 Update Category

Request:

```
PUT /v1/categories/802
Content-Type: application/json

{
    "title": "My new title"
}
```

Response:

```
{
    "id": 802,
    "user_id": 267,
    "title": "My new title"
}
```

15.3.20 Delete Category

Request:

```
DELETE /v1/categories/802
```

15.3.21 OPML Export

Request:

```
GET /v1/export
```

The response is a XML document (OPML file).

Note: This API call is available since Miniflux v2.0.1.

15.3.22 OPML Import

Request:

```
POST /v1/import

XML data
```

- The body is your OPML file (XML).
- Returns `201 Created` if imported successfully.

Response:

```
{
  "message": "Feeds imported successfully"
}
```

Note: This API call is available since Miniflux v2.0.7.

15.3.23 Create User

Request:

```
POST /v1/users
Content-Type: application/json

{
    "username": "bob",
    "password": "test123",
    "is_admin": false
}
```

Response:

```
{
    "id": 270,
    "username": "bob",
    "language": "en_US",
    "timezone": "UTC",
    "theme": "default",
    "entry_sorting_direction": "asc"
}
```

Note:

- You must be an administrator to create users.

15.3.24 Update User

Request:

```
PUT /v1/users/270
Content-Type: application/json

{
    "username": "joe"
}
```

Available fields:

- `username`
- `password`
- `is_admin` (boolean)
- `theme`
- `language`
- `timezone`

Response:

```
{
    "id": 270,
    "username": "joe",
    "language": "en_US",
    "timezone": "UTC",
    "theme": "default",
    "entry_sorting_direction": "asc"
}
```

Note:

- You must be an administrator to update users.

15.3.25 Get Current User

Request:

```
GET /v1/me
```

Response:

```
{
    "id": 1,
    "username": "admin",
    "is_admin": true,
    "theme": "default",
    "language": "en_US",
    "timezone": "America/Vancouver",
    "entry_sorting_direction": "desc",
    "last_login_at": "2018-06-01T19:54:30.723051-07:00",
    "extra": {}
}
```

Note: This API endpoint is available since Miniflux v2.0.8.

15.3.26 Get User

Request:

```
# Get user by user ID
GET /v1/users/270

# Get user by username
GET /v1/users/foobar
```

Response:

```
{
    "id": 270,
    "username": "bob",
    "is_admin": false,
```

(continues on next page)

(continued from previous page)

```
    "language": "en_US",
    "timezone": "UTC",
    "theme": "default",
    "entry_sorting_direction": "asc",
    "last_login_at": "2017-12-27T16:40:58.841841-05:00",
    "extra": {
        "google_id": "42424242424242"
    }
}
```

Note:

- You must be an administrator to fetch users.
- The extra field is a dictionary of optional values.

15.3.27 Get Users

Request:

```
GET /v1/users
```

Response:

```
[
    {
        "id": 270,
        "username": "bob",
        "is_admin": false,
        "language": "en_US",
        "timezone": "UTC",
        "theme": "default",
        "entry_sorting_direction": "asc",
        "last_login_at": "2017-12-27T16:40:58.841841-05:00",
        "extra": {}
    }
]
```

Note:

- You must be an administrator to fetch users.
- The extra field is a dictionary of optional values.

15.3.28 Delete User

Request:

```
DELETE /v1/users/270
```

Note:

- You must be an administrator to delete users.

15.3.29 Healthcheck

The healthcheck endpoint is useful for monitoring and load-balancer configuration.

Request:

```
GET /healthcheck
```

Response:

```
OK
```

Return a status code 200 when the service is up.

CHAPTER 16

Development

16.1 Requirements

- Git
- Go >= 1.9
- Godep `go get -u github.com/golang/dep/cmd/dep` (https://github.com/golang/dep)

16.2 Checkout the source code

Create a fork of the project and clone the repository:

```
cd $GOPATH/src
mkdir -p github.com/miniflux
cd github.com/miniflux
git clone https://github.com/<your_username>/miniflux.git
```

16.3 Build a binary of the application

```
# All binaries
make build

# Only Linux
make linux

# Build for ARM architectures (32 and 64 bits)
make linux-arm
```

(continues on next page)

(continued from previous page)

```
# Only Mac OS
make darwin
```

16.4 Run the software locally

```
make run
```

This command execute `go generate` and `go run main.go`.

16.5 Regenerate embedded files

To avoid any dependencies, all assets (Javascript, CSS, images, translations) are automatically included in the source code.

```
go generate
```

16.6 Linter

```
make lint
```

16.7 Unit tests

```
make test
```

16.8 Integration tests

Integration tests are testing API endpoints with a real database.

You need to have Postgresql installed locally preconfigured with the user "postgres" and the password "postgres".

To run integration tests, execute the following command:

```
make integration-test ; make clean-integration-test
```

If the test suite fail, you will see the logs of Miniflux.

CHAPTER 17

Internationalization

17.1 Translation files

Translations are just a simple JSON file. This is a mapping table between English and the language, for example:

```
{
    "Username": "Nom d'utilisateur",
    "Password": "Mot de passe",
    "Unable to parse Atom feed: %v.": "Impossible de lire ce flux Atom: %v."
}
```

Placeholders are the ones used by the package fmt.

17.2 Plural forms

Some languages have different rules regarding plurals. These rules are defined in the file `locale/plurals.go`. You could add more rules if yours is not available.

In the JSON file, a plural translation is defined like that:

```
{
    "plural.feed.error_count": [
        "%d error",
        "%d errors"
    ]
}
```

This example is for the English language, the plural form is `plurals=2; plural=(n != 1);`. For one error, we will have `1 error`, for 2 or more errors: `3 errors`.

You could find the different plural forms here:

- https://localization-guide.readthedocs.io/en/latest/l10n/pluralforms.html

- http://www.unicode.org/cldr/charts/29/supplemental/language_plural_rules.html

17.3 How to add a new language?

17.3.1 1) Checkout the source code from the repository

Create a fork of the project and clone the repository:

```
cd $GOPATH/src
mkdir -p github.com/miniflux
git clone https://github.com/<your_username>/miniflux.git
```

17.3.2 2) Create a new translation file

- In the folder `locale/translations`, create a new JSON file, for example, `de_DE.json` for German.
- You could copy the translations from `fr_FR.json` and replace the strings.

17.3.3 3) Add the language to the list

Open the file `locale/locale.go`, and edit the function `AvailableLanguages()`.

```
func AvailableLanguages() map[string]string {
    return map[string]string{
        "en_US": "English",
        "fr_FR": "Français",
    }
}
```

This function returns a mapping table of available languages. On the left, you have the language code and on the right the language name written in native language.

17.3.4 4) Test the translations

Translation files are embedded into the application executable. You must compile the software to see the changes.

```
make run
```

You must have a local development environment configured properly.

17.3.5 5) Create a branch and send a pull-request

Your pull-request should contains only 3 files:

- `locale/translations/xx_XX.json`
- `locale/locale.go`
- `locale/translations.go`

If you don't know how to send a pull-request, here is the documentation of GitHub: https://help.github.com/articles/creating-a-pull-request/#creating-the-pull-request

CHAPTER 18

Migration from Miniflux 1.2.x

Miniflux 2.x is not backward compatible with Miniflux 1.x.

18.1 Differences between Miniflux 1.2 and Miniflux 2.0

- Miniflux 2 supports multiple attachments.
- Miniflux 2 uses categories instead of groups, only one category can be assigned to a feed.
- Miniflux 2 have fewer settings.
- Miniflux 2 doesn't have public RSS feed or cronjobs.
- Miniflux 2 doesn't use API tokens anymore, for the Fever API, choose your own password and for the REST API use your account password.
- Miniflux 2 stores favicons into the database instead of using the local filesystem.
- Miniflux 2 themes are embedded into the application.
- Miniflux 2 doesn't support RTL languages.
- Miniflux 2 supports only Postgresql.
- Miniflux 2 is written in Go (Golang) instead of PHP.

18.2 OPML Import

If you don't care about your previous data, export your feeds from Miniflux 1.x in OPML and import them into Miniflux 2.

18.3 Migration Script

There is a migration script in Miniflux Legacy repository: `scripts/migrate-v2.php`.

- This script requires direct access to the old and the new database.
- The first group linked to a feed will become the category associated with the imported feed.
- Only bookmarked items are migrated.
- Since entries are not identified in the same way in Miniflux 2, you may have duplicated entries when refreshing your imported feeds.

18.3.1 Step 1

Make sure you are using the latest version of Miniflux 1.2.x.

18.3.2 Step 2

Install Miniflux 2 without creating any users. Create only the database schema (just run the migrations).

18.3.3 Step 3

Go into the Miniflux 1.2.x directory and run the script:

```
php scripts/migrate-v2.php --dsn="pgsql:host=localhost;dbname=miniflux2;user=postgres;
↪password=postgres"

Destination is "pgsql:host=localhost;dbname=miniflux2;user=postgres;password=postgres"
* 2 user(s) to migrate
* Migrating user: #254 => #284
* Migrating integrations
* Migrating categories
* Migrating feeds
* Migrating entries
[...]
```

The script takes the PDO DSN of Miniflux 2 database as argument. Adjust the parameters to your own environment.

CHAPTER 19

Frequently Asked Questions

19.1 Feature X was available in Miniflux 1?

Miniflux 2 doesn't try to reimplement all features of Miniflux 1. The minimalist approach is pushed a little bit further.

If you really miss something, **you must contribute to the project**, but remember, **you have to keep the minimalist philosophy of Miniflux**.

19.2 Why are you not developing my feature request?

- Developing a software takes a lot of time.
- This is a free and open source project, no one owes you anything.
- If you miss something, contribute to the project.
- Don't expect anyone to work for free.
- As mentioned above, the number of features is voluntarily limited. Nobody likes bloatware.
- Improving existing features is more important than adding new ones.

19.3 Why Miniflux stores favicons into the database?

Miniflux follows the the Twelve Factors principle. Nothing is stored on the local file system. The application is designed to run on ephemeral containers without persistent storage.

19.4 How to create themes for Miniflux 2?

As of now, Miniflux 2 doesn't have any mechanism to load external stylesheets to avoid dependencies. Themes are embedded into the binary.

If you would like to submit a new official theme, you must send a pull-request. But do not forget that **you will have to maintain your theme over the time**, otherwise, your theme will be removed from the code base.

19.5 Why there is no plugin system?

- Because this software has a minimalist approach.
- Because implementing a plugin system increase the complexity of the software.
- Because people do not maintain their plugins after a while.

19.6 What is "Save this article"?

"Save" sends the feed entry to third-party services like Pinboard or Instapaper if configured.

19.7 How are items removed from the database?

When a subscription is refreshed, entries marked as "removed" and not visible anymore in the XML feed are removed from the database.

19.8 What "Flush History" does?

"Flush History" changes the status of entries from "read" to "removed" (except for bookmarks). Entries with the status "removed" are not visible in the user interface.

19.9 Is there any browser extensions for Miniflux?

- Miniflux Notifications: Chrome Web Store - Source Code

19.10 Which binary do I need to use on my Raspberry Pi?

- Raspberry Pi A, A+, B, B+, Zero: `miniflux-linux-armv6` compiled with `GOARM=6`
- Raspberry Pi 2, 3: `miniflux-linux-armv7` compiled with `GOARM=7`

19.11 Why there is no latest tag on the Docker image?

- There is no latest tag to avoid people updating the software blindly to a new version without reading the ChangeLog and without looking to the possible breaking changes.
- This is also to avoid people reporting bugs on the "latest" version, the latest tag is a moving target, this 2.0.4 today but it will be 2.0.5 another day. Having a bug reports that say "That doesn't work with the latest version" doesn't really help.
- Latest tag or not, you still need to destroy the current container and start a new one with the desired image.
- Using the latest tag is error-prone, it doesn't mean you are on the latest version, especially if you forget to pull the image. By pinning the exact version you avoid surprises.

19.12 Why SQL migrations are not executed automatically?

- Because it's a source of problems.
- Only one process should manipulate the database schema at once.
- If you run multiple containers with an orchestrator that may cause issues.
- You can still run the migrations by defining the variable `RUN_MIGRATIONS=1`.