Instalação e Configuração Droppy Docker

- Instalação Droppy Github

Instalação Droppy Github

Link: https://github.com/silverwind/droppy

Maio/2025.

Development on droppy has ceased because I don't have enough time or motivation to properly support it and because of its outdated technology stack, it became exceedingly boring to work.

68747470733a2f2f63646e2e7261776769742e636f6d2f73696c76657277696e642f64726f7070792f6

68747470733a2f2f696d672e736869656c64732e696f2f6e706d2f762f64726f7070792e737667
68747470733a2f2f696d672e736869656c64732e696f2f62616467652f6c6963656e63652d6273642d
68747470733a2f2f696d672e736869656c64732e696f2f6e706d2f646d2f64726f7070792e737667

droppy is a self-hosted file storage server with a web interface and capabilities to edit files and view media directly in the browser. It is particularly well-suited to be run on low-end hardware like the Raspberry Pi.

Features

- Responsive, scalable HTML5 interface
- Realtime updates of file system changes
- Directory and Multi-File upload
- Drag-and-Drop support
- Clipboard support to create image/text files
- Side-by-Side mode
- Simple and fast Search
- Shareable public download links
- Zip download of directories
- Powerful text editor with themes and broad language support
- Image and video gallery with touch support
- Audio player with seeking support
- Fullscreen support for editor and gallery
- Supports installing to the homescreen
- Docker images available for x86-64, ARMv6, ARMv7 and ARMv8

General Information

Two directories will be used, one for configuration and one for the actual files:

- `config`: defaults to `~/.droppy/config`, override with `-c /some/dir`
- `files`: default `~/.droppy/files` override with `-f /some/dir`

droppy maintains an in-memory representation of the `files` directory. If you're on slow storage and/or serving 100k or more files, the initial indexing on startup will likely take some time.

Installation

Local Installation □□

With `Node.js` >= 12.10.0 installed, run:

```
$ npm install -g droppy
$ droppy start -c /srv/droppy/config -f /srv/droppy/files
```

To make droppy run in the background, you can use the `--daemon` option, thought it is adviced that you install it as a persistent service in your system. For Linux, see these guides:

- Systemd-based distributions
- Debian (Pre-Jessie)
- Nginx reverse proxy
- Apache reverse proxy

Docker installation □□

The `silverwind/droppy` multi-arch images supports `amd64`, `arm64`, `arm/v7` and `arm/v6` architectures. To pull and run, use:

```
$ docker run --name droppy -p 127.0.0.1:8989:8989 silverwind/droppy
```

This method uses automatic volumes for `/config` and `/files` which can be overridden through `-v /srv/droppy/config:/config` and `-v /srv/droppy/files:/files`. If you're using existing files, it's advisable to use `-e UID=1000 -e GID=1000` to get new files written with correct ownership.

To update a docker installation, run

```
$ docker pull silverwind/droppy
$ docker stop droppy && docker rm droppy
$ docker run --name droppy -p 127.0.0.1:8989:8989 silverwind/droppy
```

docker-compose

Alternatively, you can use the example `docker-compose.yml`:

```
$ curl -O https://raw.githubusercontent.com/silverwind/droppy/master/examples/docker-compose.yml
$ docker-compose up
```

This example `docker-compose.yml` uses the subdirectories `config` and `files` of the current working directory for storing data.

Caddy

See the example Caddyfile.

Configuration

By default, the server listens on all IPv4 and IPv6 interfaces on port 8989. On first startup, a prompt to create login data for the first account will appear. Once it's created, login credentials are enforced. Additional accounts can be created in the options interface or the command line. Configuration is done in `config/config.json`, which is created with these defaults:

```
{
  "listeners" : [
    {
      "host": ["0.0.0.0", "::"],
      "port": 8989,
      "protocol": "http"
    }
  ],
  "public": false,
  "timestamps": true,
  "linkLength": 5,
  "linkExtensions": false,
  "logLevel": 2,
  "maxFileSize": 0,
  "updateInterval": 1000,
```

```
  "pollingInterval": 0,
  "keepAlive": 20000,
  "allowFrame": false,
  "readOnly": false,
  "ignorePatterns": [],
  "watch": true,
  "headers": {}
}
```

Options

- `listeners` *Array* - Defines on which network interfaces, port and protocols the server will listen. See listener options below. `listeners` has no effect when droppy is used as a module. The default listens on HTTP port 8989 on all interfaces and protocols.
- `public` *boolean* - When enabled, no user authentication is performed. Default: `false`.
- `timestamps` *boolean* - When enabled, adds timestamps to log output. Default: `true`.
- `linkLength` *number* - The amount of characters in a shared link. Default: `5`.
- `linkExtensions` *boolean* - Whether shared links should include the file extension. This can be used to allow other software to make a guess on the content of the file without actually retrieving it. Default: `false`.
- `logLevel` *number* - Logging amount. `0` is no logging, `1` is errors, `2` is info (HTTP requests), `3` is debug (Websocket communication). Default: `2`.
- `maxFileSize` *number* - The maximum file size in bytes a user can upload in a single file. `0` means no limit. Default: `0`.
- `updateInterval` *number* - Interval in milliseconds in which a single client can receive update messages through changes in the file system. Default: `1000`.
- `pollingInterval` *number* - Interval in milliseconds in which the file system is polled for changes, which **may necessary for files on external or network-mapped drives**. Corresponds to chokidar's usePolling option. This is CPU-intensive. `0` disables polling. Default: `0`.
- `keepAlive` *number* - Interval in milliseconds in which the server sends websocket keepalive messages, which may be necessary when proxies are involved. `0` disables keepalive messages. Default: `20000`.
- `uploadTimeout` *number* - Request timeout for upload requests in milliseconds. Default: `604800000` which is 7 days.
- `allowFrame` *boolean* - Allow the page to be loaded into a `<frame>` or `<iframe>`. Default: `false`.
- `readOnly` *boolean* - Treat all files as being read-only. Default: `false`.
- `dev` *boolean* - Enable developer mode, skipping resource minification and enabling live reload. Default: `false`.
- `ignorePatterns` *Array* - Array of file path glob patterns to ignore when indexing files. See here for supported patterns. Default: `[]`.

- `watch` *boolean* - Whether to watch the local file system for changes. Disabling this may improve performance when dealing with a large number of files, but with the downside that changes not done via droppy won't be detected. Default: `true`.
- `headers` *Object*: A object with key-value pairs of custom HTTP headers to set on all responses, for example `{"Access-Control-Allow-Origin": "*"}`. Default: `{}`.

Listener Options

`listeners` defines on which network interfaces, ports and protocol(s) the server will listen. For example:

```
"listeners": [
  {
    "host": "::",
    "port": 80,
    "socket": "/tmp/droppy",
    "protocol": "http"
  },
  {
    "host": "0.0.0.0",
    "port": 443,
    "protocol": "https",
    "key": "~/certs/example.com.key",
    "cert": "~/certs/example.com.crt"
  }
]
```

The above configuration will result in:

- HTTP listening on all IPv4 and IPv6 interfaces, port 80 and on the unix domain socket `/tmp/droppy`.
- HTTPS listening on all IPv4 interfaces, port 443 using the provided TLS files.

A listener object accepts these options:

- `host` *string/Array* - Network interface(s) addresses to listen on. Required when `port` is given. Note that "::" will typically bind to both IPv4 and IPv6 on all addresses but a "0.0.0.0" address might be required if IPv6 is disabled.
- `port` *number/string/Array* - Network port(s) to listen on. Required when `host` is given.
- `socket` *string/Array* - Unix domain socket(s) to listen on.
- `protocol` *string* - Protocol to use, `http` or `https`. Required.

For TLS the following additional options are available. Paths can be given relative to the configuration directory and `~` is resolved as expected.

- `cert` *string* - Path to PEM-encoded TLS certificate file, which can include additional intermediate certificates concatenated after the main certificate. Required.
- `key` *string* - Path to PEM-encoded TLS private key file not encrypted with a passphrase. Required.

Downloading from the command line

To download shared links with `curl` and `wget` to the correct filename:

```
$ curl -OJ url
$ wget --content-disposition url
```

Development

To start a live-reloading dev server:

```
$ git clone https://github.com/silverwind/droppy && cd droppy
$ npm i
$ node droppy start --dev
```

The Makefile has a few tasks for updating dependencies, pushing docker images, see the comment above for dependencies of those tasks.

© silverwind, distributed under BSD licence.