File Fortress

Jean-Pierre Appel Owen Halliday David Marrero

Advisor: Dr. Benjamin Coleman

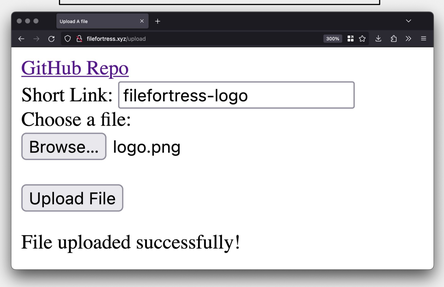
MORAVIAN UNIVERSITY

Abstract

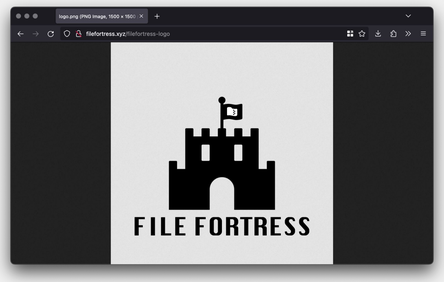
For websites with user-generated content, uploading images and other media may not always be supported, requiring the use of hyperlinks instead. As the final project for our Web Programming course, we created File Fortress, a file hosting service. Users are able to manage files via web interface, command line application, or a REST API. Our project consists of a Python backend, Flask web server, MariaDB database, with a nginx reverse proxy and file server. The tech stack can easily be launched via a single command by the use of Docker Compose. Overall, File Fortress simplifies the installation, uploading, and sharing process of file hosting.

Interfaces and Features

Web



Users can upload and view files on the web interface.



Command Line Application

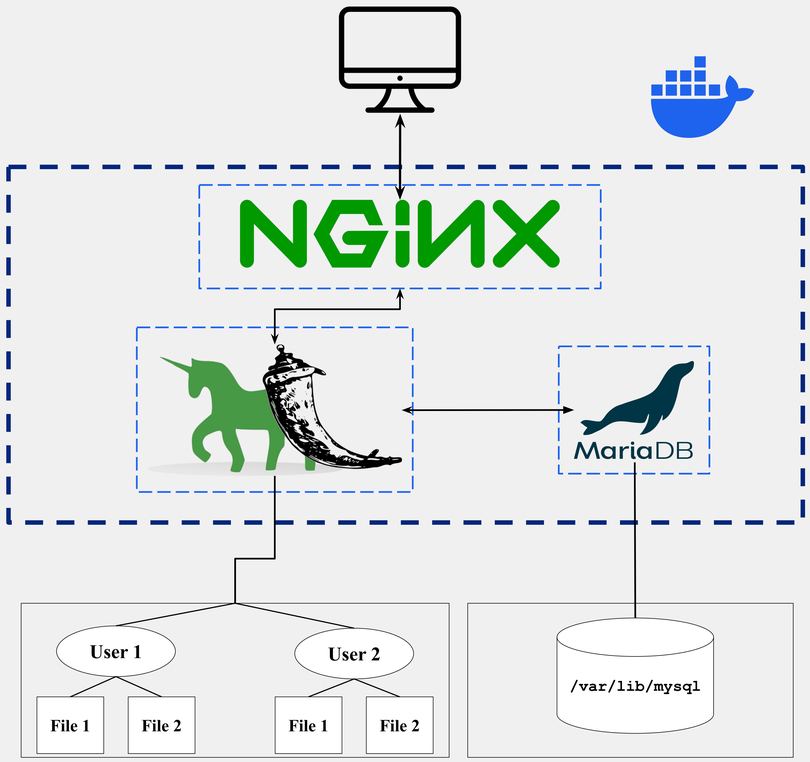
```
$ file-fortress remove filefortress-logo
File removed!
```

The bundled command line application supports file deletion.

```
$ file-fortress upload -shortlink demo demo.txt
File demo.txt uploaded successfully!
Link: http://filefortress.xyz/demo
```

Users can also upload files directly from the command line.

System Architecture



- File Fortress is run as a collection of Docker containers
- nginx is the reverse proxy and exposed container
- Gunicorn WSGI server connects to Flask
- Flask web server interacts with database and storage volume
- MariaDB database stores relations between files and URLs
- Docker volumes persist user files and the database between sessions

Public Demo

Demo Site

Login Details
User: kutztown
Password: pacise2024

Picture of this Poster

REST API

C	`POST /api/v1/file/<identifier>` `POST /api/v1/file`
R	`GET /api/v1/file/<identifier>` `OPTIONS /api/v1/file/<identifier>`
D	`DELETE /api/v1/file/<identifier>`

- API endpoints are versioned to allow live modification
- (C)reate (R)ead (U)pdate (D)elete is an interface that describes file storage operations
- Users can create a file at a specified path or have one generated for them
- Files are publicly accessible at a path after upload
- The availability of a path can be checked via an options request
- Privileged users are able to delete uploaded files

Deployment and Containerization

- Used Docker for containerization
- Used Docker Compose to manage multiple containers
- Separated into individual containers
  - Database
  - Backend
  - Web server
- Docker allowed more
  - Stability
  - Reproducibility

Future Work

The time constraints of the semester left us with some untackled features

- Implementation of user authentication through OAuth 2.0
- Support for file storage via S3-compatible object-based storage
- Endpoints for creating, updating, and deleting user file collections