

Filux File Sharing

- [Install Filux for Easy, Secure File Sharing](#)

Install Filux for Easy, Secure File Sharing

<https://www.youtube.com/embed/Z8G9mJoB9bA>

Setup a Server

Install Updates

Update and upgrade your server's packages using the following commands:

- For Ubuntu/Debian: `sudo apt update && sudo apt upgrade -y`
- For RedHat/CentOS/Fedora/Alma/Rocky: `sudo dnf update -y`

Create a Non-Root User

Create a non-root user with superuser (sudo) privileges:

1. Add a new user using `adduser <username>`
2. Set the password for this user.
3. Enter the relevant information (optional)
4. Enter 'Y', then press Enter.
5. Add the user to the "sudo" group:
 - For Ubuntu/Debian: `usermod -aG sudo <username>`
 - For RedHat/CentOS/Fedora/Alma/Rocky: `usermod -aG wheel <username>`

Ubuntu / Debian

```
usermod -aG sudo <username>
```

RedHat / CentOS / Fedora / Alma / Rocky

```
usermod -aG wheel <username>
```

Now, you can log out of the system, and log back in as your new non-root super user.

Install Docker and Docker Compose

Install Docker and Docker Compose on your server:

1. Install the curl utility:

- For Ubuntu/Debian: `sudo apt install curl -y`
- For RedHat/CentOS/Fedora/Alma/Rocky: `sudo dnf install curl -y`

2. Run the command to install Docker and Docker Compose:

```
curl https://get.docker.com | sh
```

Add Your User to the Docker Group

Add your non-root user to the docker group so you can use Docker commands without sudo:

```
sudo usermod -aG docker <username>
```

Now log out and back into your system, and you are ready to setup your application.

Install Filux with Docker / Docker Compose

Let's create our folder structure for the Filux application:

```
mkdir -p docker/filux
```

This command will create the full path `docker/filux` from the directory you are in when you run the command. If the 'docker' folder already exists in that path, it will just create the 'filux' folder inside the existing 'docker' folder.

Now we'll move into the folder with

```
cd docker/filux
```

Here, we'll create two files: 'compose.yaml' and '.env'. Let's create the 'compose.yaml' file first.

```
nano compose.yaml
```

once inside the nano text editor, just copy the code block below, and paste it into the editor. You can paste in Linux with CTRL + Shift + V.

compose.yaml File

services:

postgres:

image: postgres:17-alpine

container_name: filux-postgres

environment:

POSTGRES_USER: \${POSTGRES_USER}

POSTGRES_PASSWORD: \${POSTGRES_PASSWORD}

POSTGRES_DB: \${POSTGRES_DB}

volumes:

- ./pgdata:/var/lib/postgresql/data

healthcheck:

test: ["CMD-SHELL", "pg_isready -U \${POSTGRES_USER}"]

interval: 10s

timeout: 5s

retries: 5

restart: unless-stopped

redis:

image: redis:7-alpine

container_name: filux-redis

command: redis-server --requirepass \${REDIS_PASSWORD}

volumes:

- ./redisdata:/data

healthcheck:

test: ["CMD", "redis-cli", "-a", "\${REDIS_PASSWORD}", "ping"]

interval: 10s

timeout: 5s

retries: 5

restart: unless-stopped

filux:

image: bmcgonag/filux:0.2.3-beta

container_name: filux

environment:

DATABASE_URL:

postgresql://\${POSTGRES_USER}:\${POSTGRES_PASSWORD}@postgres:5432/\${POSTGRES_DB}

REDIS_URL: redis://default:\${REDIS_PASSWORD}@redis:6379

APP_URL: \${APP_URL}

ports:

```

- "80:80"
volumes:
- ./uploads:/app/uploads #<- if you have a special folder or mounted volume you want
files uploaded into, put it here on the left side.
depends_on:
  postgres:
    condition: service_healthy
  redis:
    condition: service_healthy
healthcheck:
  test: ["CMD", "wget", "--no-verbose", "--tries=1", "--spider",
"http://127.0.0.1:80/api/v1/health"]
  interval: 30s
  timeout: 10s
  retries: 3
  start_period: 40s
restart: unless-stopped

```

In the above file, you'll want to setup the proper folder under the `filux` section for you ruploaded files. By default it will create a folder called 'uploads' in your working directory. If you have a special location, such as a special folder or mounted directory where you want uploads to go, you'd put the path to that directory in place of `./uploads` on the left side of the colon.

When done, user CTRL + O to save, then press Enter to confirm, and CTRL + X to exit the nano editor.

Next, we need to create our environment variables file. This is where we put things like JWT_Secretes, and passwords for the database. it's just there to help make those single entries defining the variable easier to use in multiple places in the compose file without a chance of a copy / paste or type mistake causing problems.

`nano .env` ← the '.' just means the file is hidden by default is using the command `ls` on the directory.

Once you are in the editor, copy the block of text below, and paste it into the nano editor.

.env File

```

# ---- Database ----
POSTGRES_USER=filux
POSTGRES_PASSWORD=< a-l0ng-str0n6-Pa5sw0rd-5h0uld-b3-here >
POSTGRES_DB=filux

```

```
# ---- Redis ----
REDIS_PASSWORD=< a-different-long-strong-password-should-be-h3r3-t00 >

# ---- Application Secrets ----
JWT_SECRET=< use-the-command--< openssl rand -base64 32 > >
ENCRYPTION_KEY=< use-the-command--< openssl rand -base64 32 >-again >

# ---- Uploads ----
UPLOAD_DIR=/app/uploads
MAX_UPLOAD_SIZE_MB=10240 # <- 10 Gigabytes
SESSION_TTL_HOURS=24

APP_URL=< "https://filux.yourgreatdomain.com" >
```

You need to update the values marked with less than ‘<’ and greater than ‘>’ signs around them. Remove the ‘<’ and ‘>’ to put the proper value.

For the JWT_SECRET and the ENCRYPTION_KEY use the command in your terminal:

```
openssl rand -base64 32
```

This will spit out along strong string to use. Do it once for the JWT_SECRET, and once for the ENCRYPTION_KEY.

Finally, replace the APP_URL with your intended URL for accessing your Filux app.

Save with CTRL + O, then press Enter to confirm, and exit the nano editor with CTRL + X.

Pull the Images and Run the App

Now, we need to pull the images that the application runs from:

```
docker compose pull
```

Once those have pulled down, you’ll run the command to start the application.

```
docker compose up -d
```

If you’d like to see the log output, you can use the command

```
docker compose logs -f
```

This will ‘follow’ the logs, and allow you to see what’s happening while the application starts. You can stop following the logs with CTRL + C.

Once you've given the application a little time to start, you can navigate to the IP address and port of your host where the application is running.

To fully make use of the app, however, I highly recommend setting up a domain name you own, and a reverse proxy from outside of your network, to the inside of your network.

For instance,

<https://filux.yourgreatdomain.com> would point from outside your network to the server inside your network. You can do this very securely using a Reverse Proxy like Pangolin. Here's a video on setting up and using Pangolin. [Install and Configure Pangolin](#)

This make it much easier, and much more secure to share data from inside your network with others who are not on your internal network.

Support this Channel and Content

Become a Patron at Patreon

[Check me out on Patreon](#)

Be me a Coffee or Beer at Paypal

[Paypal Support for Awesome Open Source](#)

Get the Awesome Open Source Merch

[Great Merch for Open Source Advocates!](#)