Flickr to Immich Migration

Complete guide using Docker/Podman

Cross-platform: Linux, Mac, Windows

Platform	Browser	Support
Linux	X11 forwarding (automatic)	Fully supported
Linux	Domain Socket (`USE_DSOCKET`)	Fully supported
Linux	D-Bus (`USE_DBUS`)	Fully supported (Podman recommended)
Mac	Open URL manually	Supported
Windows (WSL2)	Open URL manually	Supported

Part 1 - Basics

1. Overview

This guide describes how to migrate a Flickr photo library to Immich using the Flickr API. The process runs inside a Docker/Podman container and works on Linux, Mac and Windows.

Workflow:

- **flickr_download** - Download photos and videos via API (metadata is embedded automatically)
- **Immich CLI** - Upload to Immich (included in the container)

  Note: The script automatically detects the operating system and container runtime (Docker/Podman) and adapts accordingly. A manual metadata embedding script is **no longer needed** - metadata is automatically processed during download via the `--metadata_store` flag.

2. Prerequisites

Component	Linux	Mac	Windows
Container Runtime	Docker or Podman	Docker Desktop	Docker Desktop / WSL2
X11	Yes (`xauth`)	Not needed	Not needed
Shell	Bash (native)	Bash (native)	WSL2 or Git Bash

3. Create a Flickr API Key

1. Go to: https://www.flickr.com/services/apps/create/
2. Click “Request an API Key” -> “Non-Commercial”
3. Fill out the form (Name: e.g. “Flickr Backup”)
4. Note down the **API Key** and **API Secret**

4. Setup

Option A: Use Docker Hub image (recommended) The pre-built image can be pulled directly from Docker Hub:

```
docker pull xomoxcc/flickr-download:latest
```

Alternatively with an explicit tag:

```
docker pull xomoxcc/flickr-download:python-3.14-slim-trixie
```

Option B: Build locally

```
# Clone the repository
git clone https://github.com/<user>/flickrtoimmich.git
cd flickrtoimmich

# Build the image locally
make build
```

Create API configuration

```
mkdir -p flickr-config
cat > flickr-config/.flickr_download << EOF
api_key: YOUR_API_KEY
api_secret: YOUR_API_SECRET
EOF
```

Show system info

```
./flickr-docker.sh info
```

5. Authentication

Linux (X11 - default): On Linux, a browser window opens automatically inside the container for OAuth authentication.

```
# Default (Chrome)
./flickr-docker.sh auth

# With Firefox
BROWSER=firefox ./flickr-docker.sh auth
```

Mac / Windows: On Mac and Windows, the OAuth URL is displayed in the terminal. This URL must be opened manually in a browser.

```
./flickr-docker.sh auth

# Output:
# NOTE for Mac/Windows:
#   - A URL will be displayed in the terminal
```

```
#   - Open this URL manually in your browser
#   - After login, the callback will be processed automatically
```

Important: After successful login, the token is saved in `flickr-config/.flickr_token`.

Network: On Linux, `--network=host` is used. On Mac/Windows, ports 8080-8100 are automatically published for the OAuth callback server.

6. List Albums

```
./flickr-docker.sh list your_flickr_username
```

The output shows album ID, title, photo count and video count per album.

7. Download Photos

All albums of a user:

```
# With username
./flickr-docker.sh download your_flickr_username

# Or with full URL
./flickr-docker.sh download https://www.flickr.com/photos/username/
```

Single album:

```
./flickr-docker.sh album 72157622764287329
```

Photos are saved in `./flickr-backup/`, along with JSON metadata.

Metadata: Flickr metadata (date, title, tags) is **automatically** embedded into the image files during download. A separate metadata script is no longer required.

Resumable Downloads: API responses are cached in `./flickr-cache/`. If interrupted, the download can simply be restarted - already downloaded files will be skipped.

Date fix: Photos with invalid dates ("0000-00-00 00:00:00") are handled automatically and no longer cause crashes.

8. Upload to Immich

The Immich CLI (`@immich/cli`) is already pre-installed in the container. There are two ways to upload:

Option A: Separate upload after download

```
# Set environment variables
export IMMICH_INSTANCE_URL=https://immich.example.com
export IMMICH_API_KEY=your_api_key

# Start upload
./upload-to-immich.sh
```

The script automatically detects whether it is running inside a container or on the host: - **Inside the container:** Runs the upload directly - **On the host:** Automatically starts a suitable container for the upload

Each subdirectory in `flickr-backup/` is created as a separate album in Immich.

Option B: Download and upload in one step Using the `download_then_upload` entrypoint, download and upload can be combined:

```
docker run --rm \
  -e DATA_DIR=/home/poduser/flickr-backup \
  -e IMMICH_API_KEY=your_api_key \
  -e IMMICH_INSTANCE_URL=https://immich.example.com \
  -v ./flickr-config:/root \
  -v ./flickr-backup:/root/flickr-backup \
  -v ./flickr-cache:/root/flickr-cache \
  xomoxcc/flickr-download:latest \
  download_then_upload your_flickr_username
```

9. Command Reference

Command	Description
`auth`	Start OAuth authentication
`download <user>`	Download all albums of a user
`album <id>`	Download a single album
`list <user>`	List albums of a user
`shell`	Interactive shell inside the container
`test-browser [url]`	Test X11/browser connection
`info`	Show system information
`clean`	Remove image and temp files
`help`	Show help

10. Directory Structure

Directory	Contents
`./flickr-config/`	API keys (`.flickr_download`) and token (`.flickr_token`)
`./flickr-backup/`	Downloaded photos/videos and JSON metadata
`./flickr-cache/`	API cache for resume functionality

The directories are mounted as volumes into the container:

Host Path	Docker Container	Podman Container
`./flickr-config/`	`/root`	`/home/poduser`
`./flickr-backup/`	`/root/flickr-backup`	`/home/poduser/flickr-backup`
`./flickr-cache/`	`/root/flickr-cache`	`/home/poduser/flickr-cache`

Host Path	Docker Container	Podman Container

11. Troubleshooting

Browser does not open (Linux):

```
# Check DISPLAY
echo $DISPLAY

# Check xauth
xauth list

# Browser test inside the container
./flickr-docker.sh test-browser
```

Invalid token:

```
# Delete token and re-authenticate
rm flickr-config/.flickr_token
./flickr-docker.sh auth
```

Podman issues: The script detects Podman automatically and uses: - `--userns=keep-id` (for X11 access and correct file permissions) - `--security-opt label=disable` (for SELinux compatibility)

Mac/Windows OAuth callback: If the OAuth callback does not work, check whether ports 8080-8100 are free. The script automatically publishes these ports for the callback server.

Rate Limiting (HTTP 429): When API rate limits are hit, the script automatically pauses the download process and waits with exponential backoff. See the Advanced section (Chapter 13) for details.

Part 2 - Advanced

12. Alternative Authentication Modes

In addition to the default X11 mode, there are two more options for browser authentication in container environments.

Domain Socket Mode (`USE_DSOCKET`) Uses a Unix socket to forward URLs from the container to the host. A Python listener is started on the host that opens the URL with `xdg-open`.

```
USE_DSOCKET=true ./flickr-docker.sh auth
```

Configurable variables:

Variable	Default	Description
`USE_DSOCKET`	`false`	Enable domain socket mode

Variable	Default	Description
DSOCKET_PATH	/tmp/.flickr-open-url.sock	Socket path on the host
DSOCKET_CONTAINER_PATH	/tmp/open-url.sock	Socket path inside the container

D-Bus Mode (USE_DBUS) Uses the XDG Desktop Portal via D-Bus to open URLs through the system browser.

```
USE_DBUS=true ./flickr-docker.sh auth
```

> **Note:** D-Bus mode works most reliably with **Podman**, as Docker may have issues with D-Bus authentication due to UID mismatches.

Variable	Default	Description
USE_DBUS	false	Enable D-Bus mode
DBUS_SESSION_BUS_ADDRESS	(from system)	D-Bus session address

Mode Overview

Mode	Prerequisite	Recommended for
X11 (default)	DISPLAY set, xauth	Desktop Linux with X11
Domain Socket	Python 3 on the host	Headless servers, SSH
D-Bus	D-Bus session, Podman recommended	Wayland, modern desktops

> **Important:** The three modes are **mutually exclusive**. Only one may be active at a time.

13. Rate Limiting and Backoff Configuration

The Flickr API has rate limits. On HTTP 429 responses, the download process is automatically paused using SIGSTOP/SIGCONT signals and resumed with exponential backoff.

Variable	Default	Description
BACKOFF_BASE	60	Base wait time in seconds
BACKOFF_MAX	600	Maximum wait time in seconds
BACKOFF_EXIT_ON_429	false	Exit immediately on rate limit (exit code 42) instead of waiting

Calculation: Wait time = BACKOFF_BASE * number_of_consecutive_429s (capped at BACKOFF_MAX)

```
# Example: Shorter wait times
BACKOFF_BASE=30 BACKOFF_MAX=300 ./flickr-docker.sh download username

# For CI/Kubernetes: Exit immediately instead of waiting
BACKOFF_EXIT_ON_429=true ./flickr-docker.sh download username
```

14. Kubernetes Deployment

An Ansible-based Kubernetes deployment is available for automated operation.

Architecture

- **Namespace:** `flickr-downloader`
- **Jobs:** One separate Kubernetes job per Flickr user
- **Operator:** Automatic restart controller (`flickr-immich-k8s-sync-operator`)
- **Entrypoint:** `download_then_upload` (download + Immich upload in one step)

Job Configuration Each job is created with the following settings:

```
command: ["/entrypoint.sh", "download_then_upload", "<flickr-user>"]
env:
  - DATA_DIR: /home/poduser/flickr-backup
  - IMMICH_API_KEY: <api-key>
  - IMMICH_INSTANCE_URL: <immich-url>
  - BACKOFF_EXIT_ON_429: "true"    # Exit immediately on rate limit
  - HOME: /home/poduser
  - TZ: Europe/Berlin
resources:
  requests: { cpu: 250m, memory: 1Gi }
  limits:   { cpu: 2000m, memory: 2Gi }
```

Operator Configuration The operator monitors jobs and restarts them as needed:

Variable	Default	Description
`NAMESPACE`	`flickr-downloader`	Kubernetes namespace
`JOB_NAMES`	-	Comma-separated list of job names
`CHECK_INTERVAL`	`60`	Check interval in seconds
`RESTART_DELAY`	`3600`	Delay before restart in seconds
`SKIP_DELAY_ON_OOM`	`true`	Restart immediately on OOM

Volume Mounts (hostPath)

```
/home/poduser                <- <prefix>/<flickr-user>/flickr-config
/home/poduser/flickr-backup <- <prefix>/<flickr-user>/flickr-backup
/home/poduser/flickr-cache  <- <prefix>/<flickr-user>/flickr-cache
```

15. Podman Specifics

The script detects Podman automatically and applies the following adjustments:

Setting	Value	Reason
`--userns=keep-id`	(Linux)	Preserves user namespace for X11 and D-Bus

Setting	Value	Reason
`--security-opt label=disable`	(Linux)	SELinux compatibility
Home directory	`/home/poduser`	Instead of `/root` (rootless container)

Multi-Arch Build with Podman

```
# Build locally (without push)
./repo_scripts/build-container-multiarch.sh onlylocal

# Build and push to Docker Hub
./repo_scripts/build-container-multiarch.sh
```

Podman creates a manifest with separate builds per platform (amd64 + arm64) and uses VM emulation for cross-compilation.

16. All Environment Variables

Container and Paths

Variable	Default	Description
`IMAGE_NAME`	`flickr-download`	Docker image name
`IMAGE_TAG`	`latest`	Docker image tag
`WORK_DIR`	`$(pwd)/flickr-backup`	Working directory for downloads
`CONFIG_DIR`	`$(pwd)/flickr-config`	Directory for API keys and token
`CACHE_DIR`	`$(pwd)/flickr-cache`	Directory for API cache
`FLICKR_HOME`	-	Home directory override (container detection)
`DATA_DIR`	`$(pwd)/flickr-backup`	Data directory for Immich upload

Browser and Authentication

Variable	Default	Description
`BROWSER`	`chrome` (Linux) / empty (Mac/Win)	Browser for OAuth
`DISPLAY`	(from system)	X11 display
`XAUTHORITY`	-	X11 auth file
`XAUTH_FILE`	`/tmp/.flickr-docker.xauth`	Temporary xauth file
`USE_DSOCKET`	`false`	Domain socket mode
`DSOCKET_PATH`	`/tmp/.flickr-open-url.sock`	Socket path (host)
`DSOCKET_CONTAINER_PATH`	`/tmp/open-url.sock`	Socket path (container)
`USE_DBUS`	`false`	D-Bus mode

Variable	Default	Description
DBUS_SESSION_BUS_ADDRESS	(from system)	D-Bus session address

Rate Limiting

Variable	Default	Description
BACKOFF_BASE	60	Base wait time (seconds)
BACKOFF_MAX	600	Maximum wait time (seconds)
BACKOFF_EXIT_ON_429	false	Exit immediately on rate limit (exit code 42)

Immich Upload

Variable	Default	Description
IMMICH_INSTANCE_URL	-	Immich instance URL (required)
IMMICH_API_KEY	-	Immich API key (required)

Kubernetes / CI

Variable	Default	Description
KUBERNETES_SERVICE_HOST	(from system)	Kubernetes pod detection
TZ	-	Timezone (e.g. Europe/Berlin)

Build

Variable	Default	Description
BUILDTIME	-	Build timestamp
PYTHONUNBUFFERED	1	Disable Python output buffering