stoatworks-unraid user guide

Container packaging for the Stoatworks web fleet — Dockerfiles and compose files for each app, and **Unraid Community Applications templates that install them.**

> **Before you rely on this: every image builds** — the workflow in each packaged repo is green, and that is the only evidence any of this works. **The machine these files were written on has no container runtime and no nginx**, so nothing here has ever been built, started or requested locally.
>
> **A green CI run proves an image *builds*. It does not prove the app inside it serves correctly**, and **no container has yet been installed on a real Unraid server from these templates.**
>
> This codebase was created with AI assistance, directed and reviewed by a human author.

What is packaged, and what deliberately is not

The things worth running on your own server: the always-on services — ATEM Overseer, RFutils, caspar-AV, srt-router, flock — and **the browser tools**, which are static pages you would self-host so they still work in a venue with no internet.

Not the plugin browser demos or the public websites. A demo shows what an effect looks like and a marketing site is already deployed, so neither is something anyone would install.

One browser tool is missing on purpose: **birddog-play-patcher** needs a small server beside the page, to fetch the Tailscale tarball on the browser's behalf, and a plain static image would lose that. It will arrive when that server exists.

Installing on Unraid

Public apps are in `templates/`, and this repository is in the Community Applications feed, so they are installable by search: open **Apps**, search for the tool's name, and install.

If an app is not there yet — the feed rebuilds on its own irregular schedule, and a template added today can take a day or two to appear — use **Docker → Add Container → Template** and paste the raw URL of its XML file from `templates/`. That path always works.

Community Applications does not scan GitHub for template repos — a repository has to be submitted to its feed once, which was done for this one on 2026-08-06. After that, every template

committed to `templates/` is picked up by the next feed build; nothing needs resubmitting.

If any app ever ships from a private repo, its package is private too, and the server needs `docker login ghcr.io` once before installing it.

Three things that are easy to get wrong

Host networking is a correctness flag, not a preference

ATEM Overseer, RFutils, srt-router and flock discover devices over mDNS and broadcast. Docker's bridge network does not reliably forward multicast — **and the failure mode is not an error. Discovery simply returns nothing, which reads as "there are no devices on this network".**

Those templates ship with host networking. You can switch them to bridge if you would rather add devices by IP, but do it knowingly.

`_headers` is the source of truth for headers, not the nginx config

Every static app carries a `_headers` file that the hosted build uses. nginx cannot read it, so the generator translates it. **Edit `_headers` and regenerate; a change made directly to the nginx config is overwritten.**

The translation is not a copy, and the reason matters: nginx's `add_header` **does not inherit into a location that sets any header of its own** — the child list *replaces* the parent list rather than merging. So an assets block setting only `Cache-Control` would **silently drop every security header for exactly the files that carry the application's JavaScript.**

The base headers therefore live in a snippet every generated location includes explicitly, and everything is emitted so the headers survive 304s.

Package visibility follows the repo, and is decided on first publish

A public template pointing at a private package produces a pull error for every user who clicks Install — and that reads as a broken template rather than a permissions one.

If an app ever starts life private, make its repo public *before* its first workflow run. Flipping the repo afterwards does not retrospectively publish a package that was created private.

Nothing here is written by hand

Two data files describe the fleet, and two scripts turn them into the 74 files spread across the app repos:

```
fleet.json    what each app is and how it builds
unraid.json   how each app is packaged, and whether its repo is public
```

The generators are deterministic and their output is committed, so every generated file is reviewable in the app repos' own history rather than produced at build time.

> **If you change a Dockerfile, a compose file or an nginx config directly, the next regeneration discards it.** Change the data files or the generators.

If something is wrong

Symptom	Cause
An app finds no devices	Bridge networking. Multicast does not cross it, and the failure is silent.
A security header is missing on the JavaScript	An nginx location that sets its own header replaced the inherited list. That is what the include snippet exists to prevent.
A header change did nothing	It was made in the nginx config rather than in `_headers`, and regenerated away.
Install fails with a pull error	The GHCR package is private. Log in on the server, or fix the package's visibility.
A hand edit disappeared	Everything under the generators is regenerated from the data files.
