weblinked-docker user guide

A URL in. NDI out. No display, no card, no desktop.

WebLinked renders a web page offscreen at a broadcast raster and exact frame rate and sends it out as video. This is the container for the part of it that makes sense on a server: point it at a dashboard, a scoreboard or a clock, and it becomes an NDI source your vision mixer, decoder or recorder can take.

⚠️ Read this before you deploy it

This is designed to run purely inside a Tailscale tailnet, or another private network you trust end to end. Do not expose it to the public internet.

The control API has no authentication until you set a token, and its script endpoint **runs arbitrary JavaScript in the rendered page — an unauthenticated port here is remote code execution in a browser on your network**, not merely a feed someone can retune. Its security has **not** been independently reviewed by a human.

This image binds every interface by default and runs with host networking. Set the bind address to your tailnet address, set a token, and keep it off the open internet. No reverse proxy, no port forward, no "it's fine, it has a password".

Before you rely on this: the image has been **built and run**. Measured inside it at 1080p50: **50.1 ticks/sec, 0 dropped ticks**, 289 µs clock lateness, and 97% of ticks carrying a fresh paint on a *pathological* full-screen animation — a dashboard is far lighter. WebLinked's own **89 tests, 25,225 checks, 0 failures** pass inside it, the first time that suite has ever run on Linux.

No NDI receiver has ever seen output from this image, because the NDI runtime is not in it and is not ours to ship. **Everything below the NDI line is verified; the NDI line itself is not.**

This packaging was created with AI assistance, directed and reviewed by a human author.

You have to supply the NDI runtime

The NDI library is loaded at run time by design — WebLinked never links it. **Its licence permits redistribution only under terms forbidding modification and reverse engineering, which MIT cannot impose, so baking it into a public image would be a licence violation rather than a packaging shortcut.**

Without it **the container still starts**, still serves the control page, and simply reports the NDI backend unavailable.

To get NDI:

1. Download the **NDI SDK for Linux** and accept their licence yourself. It is free, behind an email.
2. Take the library and the symlinks beside it out of the SDK.
3. Put them in a directory on the host and **mount it at** `/opt/ndi/lib`.

The image's library path already includes that directory. Confirm it took by reading the compiled backends out of the state endpoint — **do that before a show rather than after**, because an unavailable backend is a running container that produces nothing.

Host networking is not optional for NDI

NDI discovery is mDNS. On Docker's bridge network **the sender is invisible to every receiver on the LAN — it does not error, it just never appears.** Use host networking, or run an NDI Discovery Server and point both ends at it.

The same applies to WebLinked's own advertisement. From 0.8.0 it publishes its control API so that rookery and the Companion module can find it — and that is mDNS too.

It needs three things here, and **missing any of them costs you discovery and nothing else** — the container runs, and the control API works perfectly for anyone who types the address:

- **host networking**;
- **avahi reachable** — this image does not run one, so mount the host's socket and have avahi running there;
- the advertisement enabled.

What is in the image, and what cannot be

Output	In the image	Why
NDI	Yes, with a mounted runtime	The point of the exercise
OMT	Yes	Compiled in, but **has never been tested against any receiver**
Preview + HTTP/OSC control	Yes	The control page is the whole UI
DeckLink, AJA	**No**	SDI wants a card and a kernel driver on the host
Syphon / Spout	**No**	macOS and Windows respectively; Linux has no equivalent
Fullscreen screen output	**No**	Needs a GPU-attached display, which a server has not got

Chromium renders on the CPU, deliberately — no GPU passthrough, nothing to arrange on the host. A dashboard is well within that; a full-screen animation is where the 3% of repeated frames above came from.

One number not to trust yet

The overwritten-frames counter climbs roughly in step with published frames, which under external pacing should stay near zero. **It may be a counter that means something different from what its name suggests; it has not been chased down.** Do not read it as dropped output until it has been.

If something is wrong

Symptom	Cause
No NDI source appears anywhere	Either the runtime is not mounted — check the compiled backends — or bridge networking is hiding the mDNS.
The container runs but nothing is discoverable	Bridge networking, or no avahi socket. The control API still works by address.
NDI reports unavailable	The library is not mounted where the image expects it.
Frames are repeated on a heavy page	CPU rendering. That is the trade for needing nothing on the host.
The overwritten counter is climbing	Known and unexplained. See above.
