

[EXT] ZoomISO Cloud Setup

ZoomISO Cloud

A headless, containerized Zoom meeting attendee that captures per-participant **raw audio and raw video** from a Zoom meeting and publishes them as shared-memory **MXL (Media eXchange Layer) flows** that other broadcast applications on the same host can consume in real time. Each output can instead be published as an **NDI** source, and an NDI source on the network can be sent **into** the meeting (virtual camera + virtual microphone). MXL input works the same way. NDI from this container only works after you set a discovery server and publish the NDI media ports one-to-one; see §5.4.

An operator-facing web dashboard is bundled with the image, so the only external dependency is Docker itself.

This document is for **clients receiving the product as a Docker image** (`.tar`).

1. What is in the delivery

File	Architecture	Host CPU
<code>zoomiso-cloud-amd64- <tag>.tar</code>	<code>linux/amd64</code> (x86_64)	Intel / AMD 64-bit

The image is based on `ubuntu:24.04` .

1.1 Supported features

Feature
Join / leave meeting (as signed-in user)

Per-participant raw audio capture → MXL
Per-participant raw video capture → MXL
Screen-share capture (subscribing to a participant's shared screen)
Virtual camera (stream MXL or NDI video into the meeting)
Virtual microphone (stream MXL or NDI audio into the meeting)
NDI output (publish a capture row as an NDI source)
NDI input (bring an NDI source into the meeting)

2. Host prerequisites

Requirement	Notes
Docker Engine	20.10 or newer. Docker Desktop is fine for evaluation; for production use the OS-native Docker Engine on a Linux host.
CPU architecture	<code>x86_64</code> (<code>linux/amd64</code>). <code>uname -m</code> must report <code>x86_64</code> .
Kernel / OS	Any Linux that Docker supports. Tested on Ubuntu 22.04/24.04.
RAM	4 GB minimum per container; 8 GB recommended when capturing multiple HD participants.
Network egress	Outbound HTTPS/TLS to <code>*.zoom.us</code> and <code>liminal.zoom.us</code> (OAuth + ZAK), plus

	standard Zoom media ports. No inbound ports required from the public internet.
Inbound (operator)	One TCP port on the host (default <code>3000</code>) reachable by whoever will operate the dashboard.
Zoom account	A Zoom user account that is authorized to join the target meetings and holds either a Zoom Enhanced Media license or the hub host or hub owner role of an active Zoom Events hub. The dashboard enforces this check on sign-in.
Meeting permissions	To subscribe to raw participant audio/video the instance must be host, co-host, or granted local-recording permission , for the target meeting.
Clock	Host clock must be reasonably accurate (NTP). MXL flows carry timestamps.

No GPU, no X server, and no sound card are required.

NDI is the exception to the “no inbound ports” note in the table above. Receivers on your broadcast LAN must reach a discovery server and the container’s published NDI media ports. Keep that traffic off the public internet. Details, including why the host port number has to match the container port number, are in **§5.4**.

3. Installing the image

1. Copy the `.tar` file to the target host.
2. Load it into Docker:

```
1 docker load -i zoomiso-cloud-amd64-<tag>.tar
```

3. Confirm the image is present and note its repository / tag (the tag is baked in at build time; `docker load` preserves it):

```
1 docker images | grep zoomiso
```

4. Throughout this document the image is referenced as `zoomiso-cloud-amd64-<tag>.tar` — substitute the tag shown by `docker images` .

3.1 Architecture-specific notes

Run this image on an x86_64 host. Cross-architecture execution via `qemu-user-static` is unsupported for this workload (the Zoom Meeting SDK and MXL shared-memory paths are latency-sensitive).

4. Running the container

Minimum viable command (publishes the dashboard on host port `3000`):

```
1 docker run -d \  
2   --name zoomiso \  
3   -p 3000:3000 \  
4   -p 5960-5990:5960-5990/tcp \  
5   -p 5960-5990:5960-5990/udp \  
6   --restart unless-stopped \  
7   zoomiso-cloud-amd64:<tag>
```

Then browse to `http://<host>:3000/` from a machine that can reach the host.

The backend launches the native `meetingSDKDemo` binary as a child process when it creates or restores an instance.

4.1 Recommended production command

```

1 docker run -d \
2   --name zoomiso \
3   --restart unless-stopped \
4   -p 3000:3000 \
5   -p 5960-5990:5960-5990/tcp \
6   -p 5960-5990:5960-5990/udp \
7   -e TZ=Etc/UTC \
8   -v zoomiso-mxl-domain:/app/mxl-main/mxl-domain \
9   -v zoomiso-state:/app/web_app/persistence \
10  --shm-size=1g \
11  zoomiso-cloud-amd64:<tag>

```

Notes:

- `--shm-size=1g` gives PulseAudio and MXL enough `/dev/shm` room for 1080p flows. Increase for higher resolutions / more participants. The Docker default (64 MB) is not enough.
- The named volume `zoomiso-mxl-domain` persists the MXL domain directory; see **§5.2 MXL output** for why this matters.
- The `zoomiso-state` volume preserves the instance settings and output definitions saved at `/app/web_app/persistence/state.json`. Without this volume, saved state is lost when the container is replaced, and **Restore Instance** after a native crash will have nothing to restore.
- `--restart unless-stopped` brings the container back after a host reboot. The native child does **not** auto-restart after a crash; use the dashboard recovery actions (see **§6**).
- The image declares `TZ=America/Los_Angeles`. Override with `-e TZ=...` if you want log timestamps in your local zone.
- That command publishes only the dashboard. It is the right command for MXL-only use. Before anyone selects the **NDI** protocol, also publish the NDI media ports and point the instance at a discovery server (**§5.4**). Publishing `5960-5990` onto a host that already has another NDI sender on those ports mixes the feeds.

4.2 Changing the listening port

The server reads `PORT` from the environment. You can either remap the host port only

(common) or change the container's listen port as well.

Remap host port, keep container default (3000):

```
1 docker run -d --name zoomiso -p 8080:3000 zoomiso-cloud-amd64:<tag>
```

Change the container's listen port too:

```
1 docker run -d --name zoomiso -e PORT=8080 -p 8080:8080 zoomiso-cloud-amd64:<tag>
```

Dashboard asset URLs are all relative, so either form works.

4.3 Mounting under a URL sub-path

If you reverse-proxy the dashboard at, say, `https://ops.example.com/zoomiso/` , set `MOUNT_POINT` so the server rewrites its asset URLs and `basename` :

```
1 -e MOUNT_POINT=/zoomiso
```

Your proxy should forward both HTTP and WebSocket upgrades under that prefix (the dashboard uses `/socketio` and `/ws` under the mount point).

5. Configuration reference

All configuration is via environment variables passed with `docker run -e ...` .

Nothing on the host filesystem needs to be edited.

5.1 Core runtime variables

Variable	Default	Purpose
----------	---------	---------

PORT	3000	HTTP + WebSocket listen port inside the container.
MOUNT_POINT	(unset)	URL sub-path when reverse-proxied (e.g. /zoomiso).
TZ	America/Los_Angeles	Container timezone (affects log timestamps).
MXL_DOMAIN	/app/mxl-main/mxl-domain	Directory where MXL creates per-flow shared-memory files.
AUTO_CREATE_INSTANCE	on	Automatically create a instance at startup when no saved instance is available.
PERSIST_STATE_WRITE	on	Write-through of the settings+outputs snapshot to disk.
PERSIST_STATE_PATH	<cwd>/persistence/state.js on	Snapshot file location. Mount its parent directory as a volume so it survives container recreation.
LANG	en_US.utf8	Locale. Change only if you know you need to.

At startup, a valid saved instance is offered in the dashboard instead of being launched automatically. Choose **Restore Instance** to recreate its settings and outputs with a fresh instance identity, or **Create Instance** to start with defaults and replace the saved configuration after confirmation. If no saved instance exists, the server creates the normal default instance automatically unless `AUTO_CREATE_INSTANCE=off` . With `PERSIST_STATE_WRITE=off` , a saved instance can still be restored, but it cannot be discarded through **Create Instance** because the replacement could not be committed safely. During recovery, the original snapshot is retained until the new native instance connects and receives its saved state.

5.2 MXL output (exposing flows to the host)

Output lifecycle (always-on)

Outputs are decoupled from meeting lifecycle. Each output row creates a pair of MXL flow writers (one video, one audio) as soon as the operator clicks **Add output**, regardless of whether a meeting has been joined yet. Until a participant is bound, the writer thread emits the configured video-loss pattern — **Black**, **Freeze**, or **Test Card**, controlled by the Video Loss Mode setting on the instance **Configure** tab — together with silent audio. Binding a participant attaches live capture to that already-running writer. Unbinding (selection clear, meeting exit, or the participant leaving) tears down only the capture side; the writer continues emitting loss frames so downstream consumers never see an MXL flow appear or disappear.

You can add up to **16** outputs per instance.

Captured per-participant flows are written into the MXL domain directory inside the container (`/app/mxl-main/mxl-domain` by default). Two deployment shapes are supported:

Option A — self-contained (flows consumed only inside the container).

Use this if no external tool needs to read the flows. No extra flags:

```
1 docker run -d -p 3000:3000 zoomiso-cloud-amd64:<tag>
```

Option B — share flows with host processes.

Mount the MXL domain from the host so other processes (your broadcast app, the MXL CLI tools, etc.) can read the same shared-memory files:

```
1 HOST_MXL_DIR=/var/lib/zoomiso/mxl-domain
2 mkdir -p "$HOST_MXL_DIR"
3 chmod 0777 "$HOST_MXL_DIR"
4
5 docker run -d \
6   --name zoomiso \
7   -p 3000:3000 \
8   --shm-size=2g \
9   --ipc=host \
10  -v "$HOST_MXL_DIR:/app/mxl-main/mxl-domain" \
11  zoomiso-cloud-amd64:<tag>
```

Guidance:

- `--ipc=host` is required if the consumer lives on the host and expects to share POSIX / System V IPC namespaces with the container.
- The bind-mounted host directory should live on a fast, local filesystem (ideally `tmpfs` or an NVMe-backed ext4/xfs). Do not point it at NFS, SMB, or a distributed filesystem.
- The container's web service runs as `root` in standalone mode, so permission mode `0777` (or matching UID ownership) on the host directory is the simplest way to allow downstream host processes to read the files.

5.3 Finding MXL flow IDs

Flow IDs are **not** a published constant table. Each output mints its own video and audio flow IDs when the row is created. Those IDs are what any MXL-capable receiving app should subscribe to.

Look them up in the dashboard:

- On the instance card, open **Settings** → **MXL** → **Current MXL flows** to see every output's video and audio flow IDs, labels, and group hint tags.
- Each output row also surfaces its runtime flow IDs once the writers are running.

Treat the dashboard values as authoritative. IDs change when writers are rebuilt, including:

- changing an output's video format

- changing the instance MXL domain or MXL group prefix
- **Restore Instance** (settings and output definitions come back; flow IDs are new)

Do not hard-code IDs from an earlier session into a downstream router.

Video flows are `video/v210` , BT.709, progressive. Audio flows are `audio/float32` , 48 kHz, 2 channels. NMOS `grouphint` tags use the instance **MXL group prefix** (defaults to the instance name), for example `ZoomISO Cloud Instance 1 Output 1:Video 1` / `ZoomISO Cloud Instance 1 Output 1:Audio 1` .

5.4 NDI output and input

An output row whose protocol is **NDI** is an NDI sender. The source name is `<instance name> - <output name>` (for example `Studio A - Output 1`). Until a participant is bound, the sender emits the configured video-loss pattern (Black, Freeze, or Test Card) and silent audio, the same always-on behavior as MXL. The footer can also take an NDI source from the network and send it into the meeting as the virtual camera and virtual microphone.

NDI inside this container does not behave like NDI on a bare-metal machine. Two setup steps are mandatory, and a third mistake (publishing the same host port for two senders) makes receivers show the wrong picture under the right name.

Why you must set a discovery server

NDI's default discovery is mDNS (multicast DNS on the local subnet). This image does not rely on mDNS, and it would not work from Docker anyway:

- The container sits on a Docker bridge (and, on Docker Desktop, inside a VM). Multicast from that network does not reach the LAN, and LAN multicast does not reach the container.
- Host networking is not a supported deployment for this product.

Without another directory, receivers never list the source, and this instance never lists sources that live outside the container. An **NDI Discovery Server** is that directory: a small unicast service, almost always on **TCP 5959**, that both sides register with over a normal TCP connection. That is why **Settings → NDI → Discovery Server** is required. Leaving it empty still lets you select the NDI protocol, but the dashboard warns you, and receivers typically never see the source.

The address you type has to be reachable **from inside the container**, as `host` or `host:port` (port defaults to 5959 when omitted):

- **Discovery server on the Docker host, Docker Desktop:** `host.docker.internal:5959`
- **Discovery server on the Docker host, Linux Docker Engine:** the host's LAN IP, for example `192.168.1.10:5959` . `host.docker.internal` is a Docker Desktop name and is not present on a typical Linux engine.
- **Discovery server on another machine:** that machine's IP and port, and the container network must be able to route to it.

Every receiver (NDI Studio Monitor, AMPP, another ZoomISO) must be pointed at the **same** discovery server, using an address that works from **that** machine. `host.docker.internal` only resolves on the Docker host itself. A receiver on another computer uses the discovery server's LAN address.

Applying a discovery server, group list, adapter selection, or multicast switch restarts every active NDI output on the instance. NDI reads this configuration only when the sender is created, so the picture drops for a moment and receivers reconnect. That is expected.

Which ports to map

Three different port roles show up as soon as NDI is in use. Only one of them is remapped the way the dashboard port is.

Port	Role	How to publish it
3000/tcp	Dashboard (HTTP and WebSocket), inside the container	Remap the host side freely, for example <code>-p 8080:3000</code> . The app does not advertise this port to NDI receivers. See §4.2.
5959/tcp	NDI Discovery Server. It usually runs on the Docker host or another studio machine, not inside this container.	Do not <code>-p</code> 5959 from the ZoomISO container. Make sure the container can open TCP to the discovery server, and that every receiver can

		too. If a firewall sits in front of the discovery server, allow 5959/tcp from both.
5960–5990 tcp and udp	NDI media. The NDI SDK inside the container binds these ports, one or more per source, and tells receivers the port number it bound.	Publish both protocols, and keep the host number equal to the container number. Widen the range the same way on both sides if you run more than about 15 NDI outputs at once.

Example (dashboard plus NDI media, discovery server already running on the host):

```

1  docker run -d \
2    --name zoomiso \
3    --restart unless-stopped \
4    -p 3000:3000 \
5    -p 5960-5990:5960-5990/tcp \
6    -p 5960-5990:5960-5990/udp \
7    -e TZ=Etc/UTC \
8    -v zoomiso-mxl-domain:/app/mxl-main/mxl-domain \
9    -v zoomiso-state:/app/web_app/persistence \
10   --shm-size=1g \
11   zoomiso-cloud-amd64:<tag>

```

TCP alone is not enough. NDI uses both TCP and UDP on that range, so publish both or receivers connect and then show nothing. Also open the same range on the host firewall for the machines that will receive.

There is no setting in the dashboard for an NDI port base. Every container's senders start at **5960** inside their own network namespace.

Why the host port has to equal the container port

The sender advertises the port it bound **inside** the container. Receivers then dial that port on whatever IP you gave them (the Extra IP below). Docker's `-p HOST:CONTAINER` mapping is

invisible to NDI.

- `-p 5960-5990:5960-5990` works, because a receiver told “connect to port 5961” hits host port 5961, which Docker forwards to container port 5961.
- `-p 7000-7030:5960-5990` does **not** work. Discovery still says port 5960 (the container port). The receiver never opens 7000. It opens host port 5960, which is either closed or, worse, some other program.

The dashboard port does not have this constraint. `-p 8080:3000` is fine, because browsers use the URL you type. NDI receivers use the port baked into the discovery record.

How a shared host port overlaps two NDI feeds

Identity-mapping `5960` onto the host means that host IP and port are a single socket.

Whoever is actually listening there is the feed a receiver gets, even when the source list shows a different name.

Discovery records carry a name (`Studio A - Output 1`) and a URL (`192.168.1.10:5961`).

Receivers that connect by URL follow the URL. If that host port is already owned by something else, the name and the picture come from different senders. Typical collisions:

- **Another NDI sender on the Docker host** already bound 5960+, including NDI Test Patterns, Studio Monitor’s own outputs, or a different application. The ZoomISO source can still appear in the picker with the right name, while the pixels (and audio) belong to the other sender.
- **A second ZoomISO container** on the same host. Both processes bind 5960 inside their own namespace. You cannot publish `5960-5990` on the host twice: the second `docker run` fails to bind, or, if you shift it with `-p 6060-6090:5960-5990`, every receiver still dials 5960 and lands on the first container. The second instance’s name shows up in discovery; its media does not.
- **This instance’s own output selected as its meeting input** on a host that already has another sender on the same port. The footer can show this instance's source name while the virtual camera displays the other feed.

Rules that avoid the overlap:

1. Publish NDI ports one-to-one, TCP and UDP, and only on a host whose `5960-5990` range is otherwise unused. Check with `ss` or `netstat` before `docker run`, and stop host-side NDI senders that sit in that range.

2. Run **one** NDI-publishing ZoomISO container per host. A second meeting on the same machine can use a different dashboard port (`-p 3001:3000`) and stay on **MXL**. If the second meeting also needs NDI, put it on another host.
3. Do not “fix” a clash by choosing a different host port for the same container port. NDI will keep advertising the container port, and receivers will keep opening the original host port.
4. If you need more than about 15 NDI outputs, widen both sides together (`-p 5960-6010:5960-6010/tcp` and the same for UDP). The left-hand number still has to equal the right-hand number.

What to set on each receiving machine

The container’s own address is a Docker bridge IP. Receivers cannot route to it, so a source that appears in the picker still connects zero times until each receiver is told a real address.

1. In that machine’s NDI Access Manager, set the **Discovery Server** to the same server the container uses, via an address the receiver can reach.
2. Add the Docker host’s reachable IP as an **Extra IP** (remote source). Same machine as the Docker host: `127.0.0.1` . A different machine: the host’s LAN IP, not `127.0.0.1` and not the container bridge IP.
3. If you set an NDI **group** on the instance, put the receiver in that same group. An empty group list uses NDI’s default group.

Leave **Multicast** off unless you have a network that actually routes multicast out of the container. The default is unicast, which is what the published ports carry.

Using NDI in the dashboard

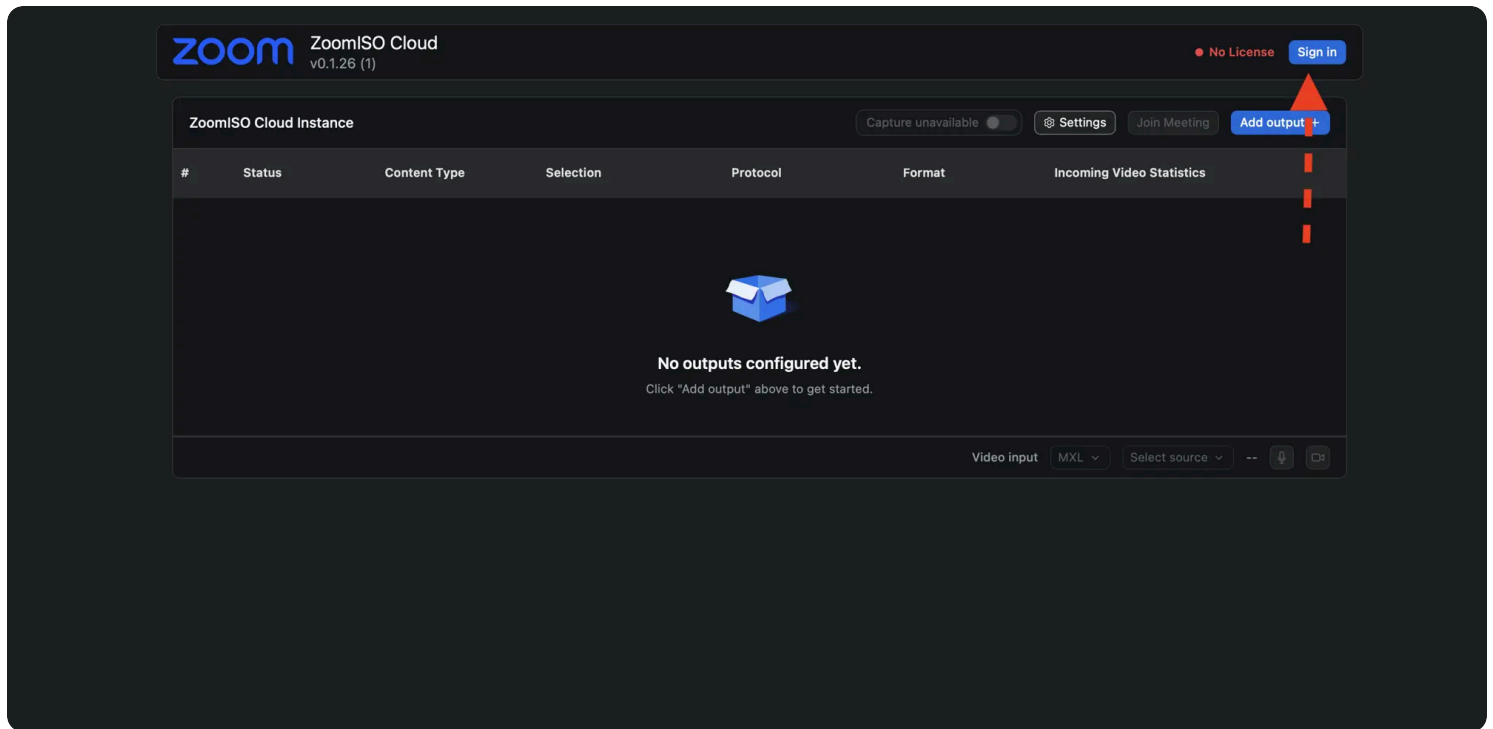
1. Start the discovery server, then start the container with the port publish above.
2. Open **Settings** → **NDI** and set **Discovery Server**. Optional: comma-separated **Groups**, and **Network Adapters** if you need to pin NDI to one interface. Apply. Active NDI outputs restart.
3. Rename the instance if more than one ZoomISO will register with that discovery server. The default name is shared, so every untouched instance publishes `ZoomISO Cloud Instance - Output N` .
4. On the output row, set the protocol to **NDI**. The source should appear to receivers as `<instance name> - <output name>` . Switching protocol stops and starts that row (a short

gap). SRT is still not available.

5. To send NDI **into** the meeting, set the footer protocol to **NDI** and pick a discovered source. One source feeds both the virtual camera and the virtual microphone. Changing protocol clears the current source. **Hide output flows from input** applies to MXL flows only; this instance's own NDI outputs still appear in the NDI picker.

6. Using the dashboard

1. Open `http://<host>:<PORT>/` in a recent Chrome, Edge, or Safari.
2. Click **Sign in**.



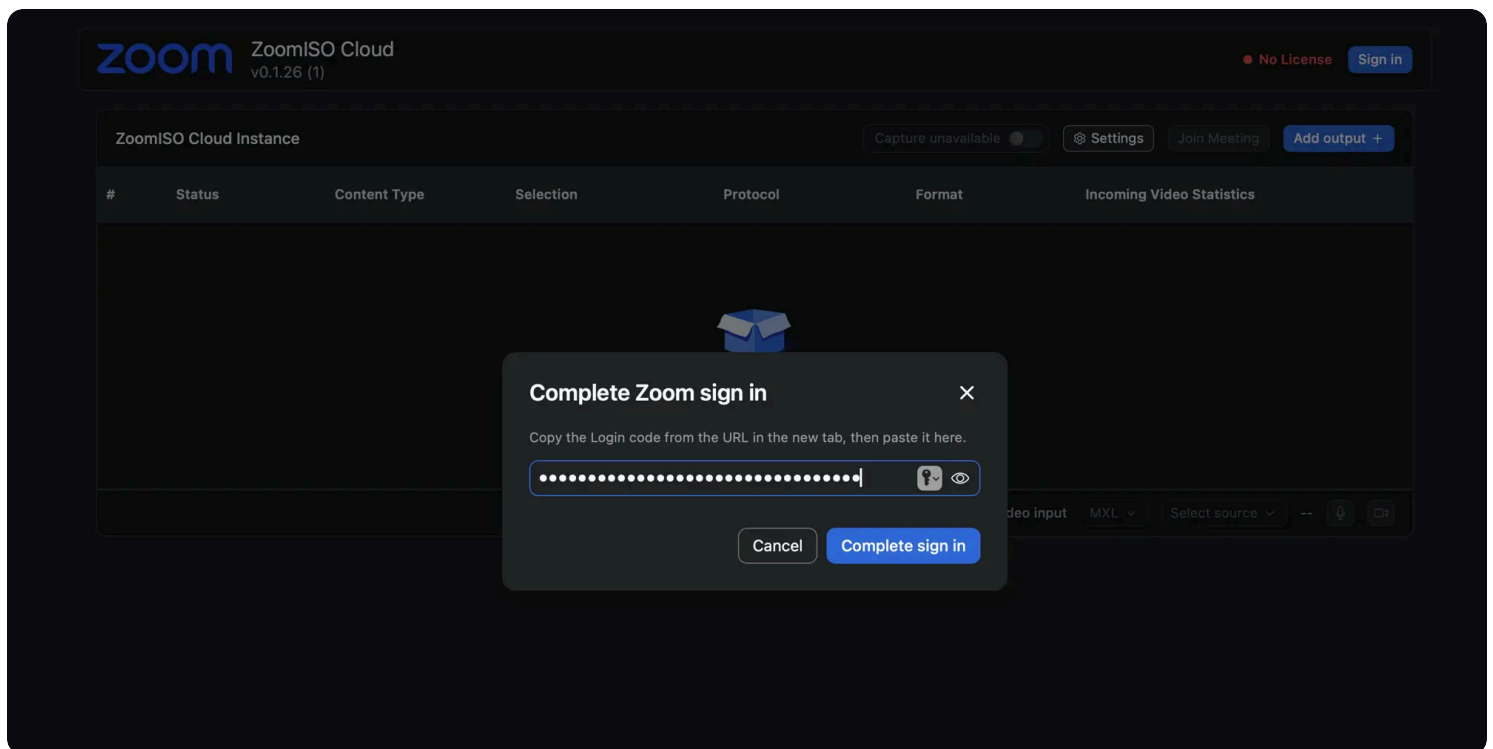
3. The dashboard opens a Zoom OAuth page (`https://zoom.us/oauth/authorize?...`) in a new tab. Complete Zoom sign-in.
4. Zoom redirects to `https://liminal.zoom.us/zoomiso-cloud-loginsuccess?code=...` .
5. Click the **Copy** icon to put the `code` query parameter on your clipboard.

Login successful

Please return to the ZoomISO Cloud tab and enter the code below in the "Login Code" field

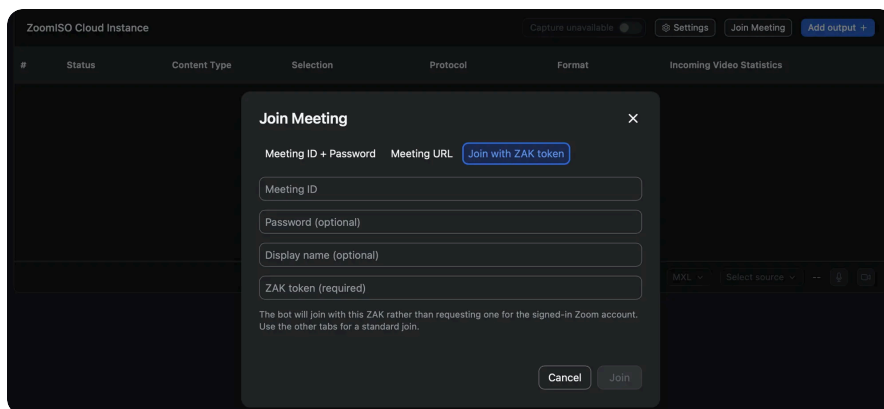
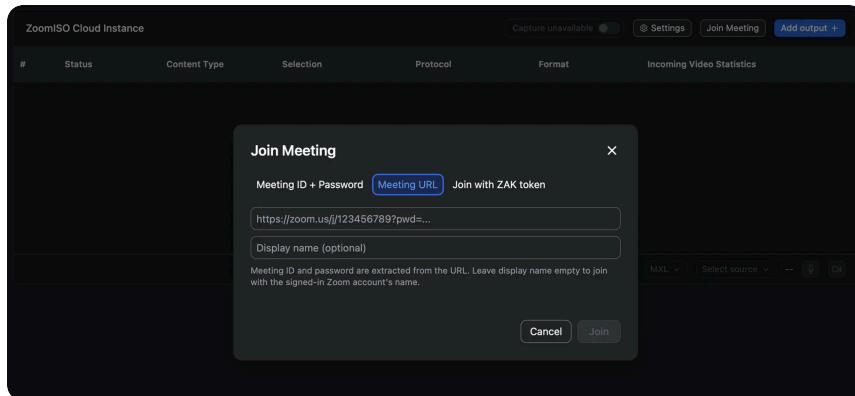
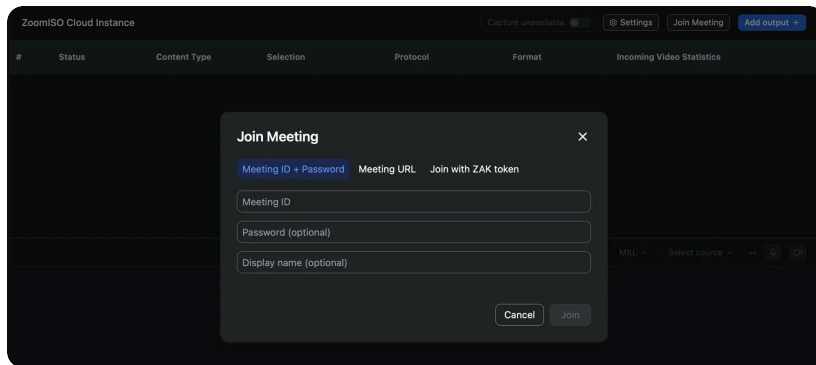
..... 

6. Paste the code into the **Complete Zoom sign in** dialog in the dashboard and confirm.



7. The dashboard exchanges the code for an access token and verifies that the account carries a qualifying license (Zoom Enhanced Media **or** active Zoom Events hub host). If the check fails the UI will show an unlicensed/error state.

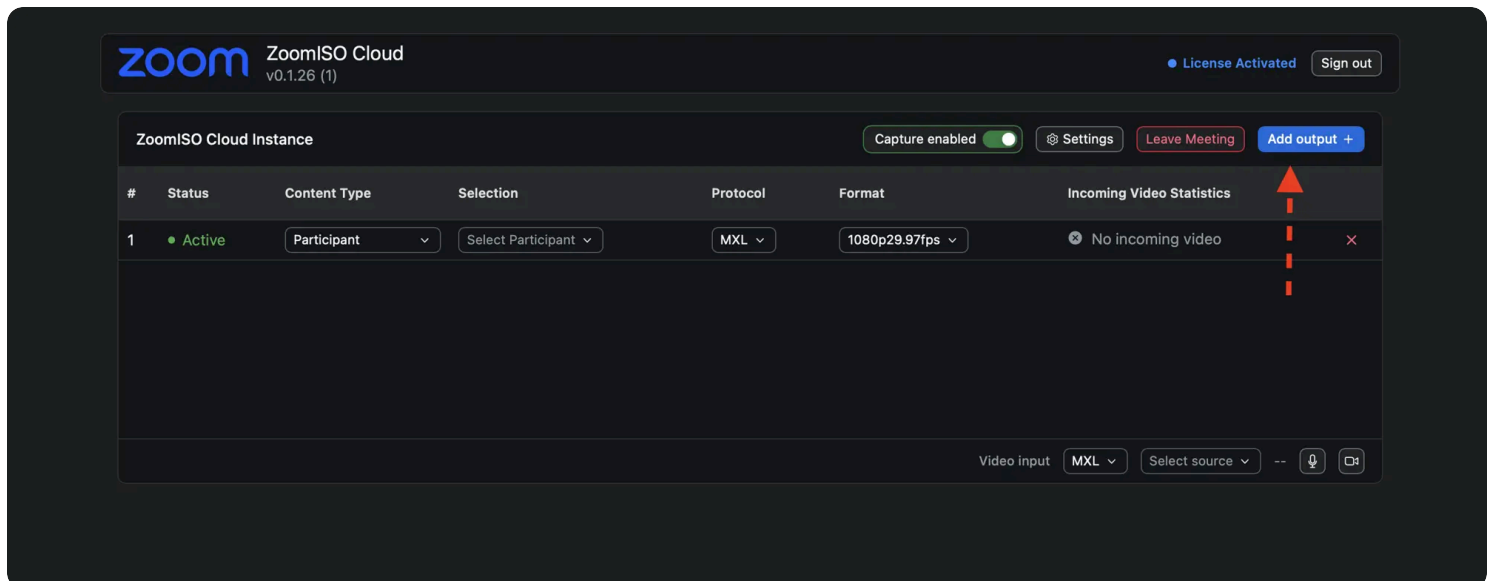
8. To join a meeting, click **Join** on the instance card. You can either enter a meeting ID (and password, if the meeting requires one), use the **Meeting URL** tab, or select **Join with ZAK Token** to input your own ZAK token. After **Leave**, Join stays disabled until the previous session has fully ended; wait for the Join button to come back rather than retrying immediately. While **Joining...** is shown, hover the button to reveal **Cancel**.



9. After joining, the instance requests recording permission from the host so it can capture audio and video. Once that is approved, capture state switches to enabled. You can also turn **auto-start capture on join** on or off under **Configure**.



10. Click **Add output** for each capture slot you need (up to 16). Each row immediately creates a pair of MXL flows and starts emitting the configured video-loss pattern plus silent audio. Removing the row tears the flows down.



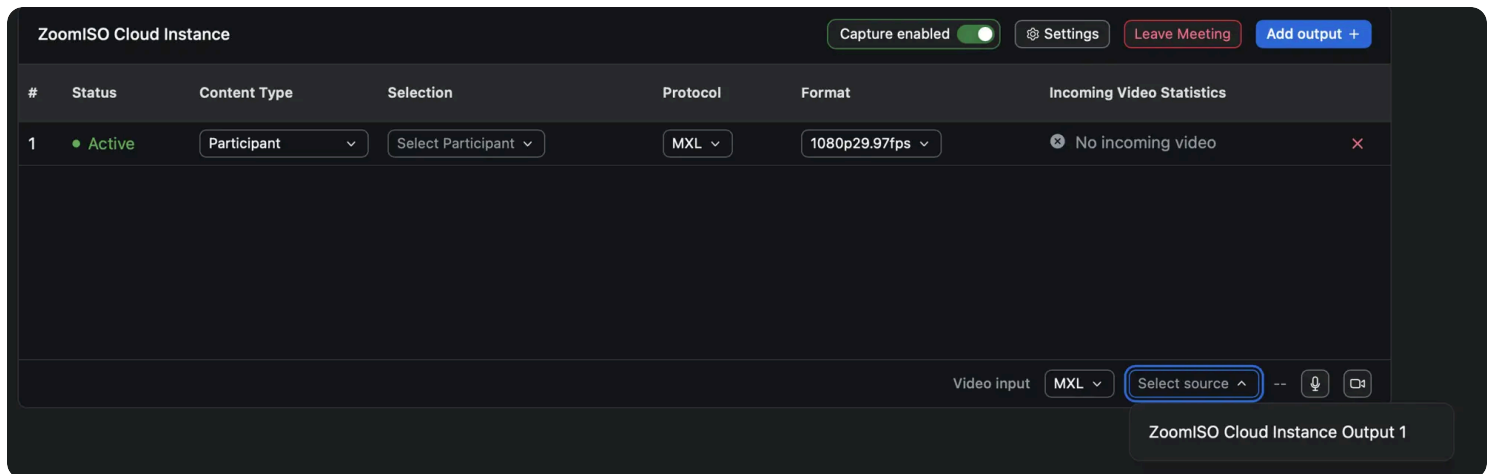
11. For each output row, pick a content type and a selection:

- **Participant** — that person's camera (and their audio).

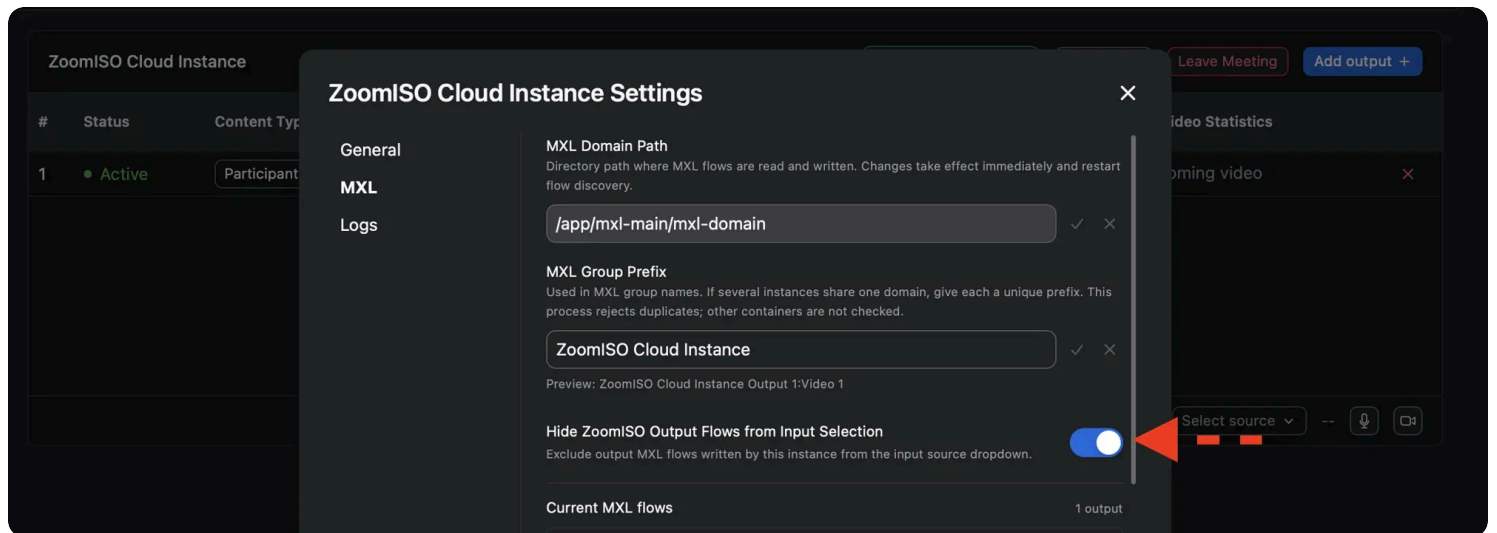
- **Participant Share** — that person's shared screen (x86_64 only).
- **Active Speaker** — whoever Zoom currently marks as the active speaker.
- **Active Screenshare** — the meeting's current shared screen (x86_64 only).

In **Selection**, pick a participant (or their screen share on x86_64) to attach live capture; reselect “Select Participant” to return that row to video-loss. Set the row’s **protocol** to **MXL** or **NDI**. NDI is enabled only when this image includes the NDI SDK. Choosing NDI with no discovery server applied shows a warning and still switches the row; receivers will not find the source until **Settings** → **NDI** is filled in (§5.4). SRT is listed for a later build and is not usable yet.

12. To **stream into** the meeting, choose an input source in the footer and unmute the virtual camera / virtual microphone. The footer protocol is **MXL** (an existing MXL video flow, or the built-in test card / tone) or **NDI** (a source returned by the discovery server). Input controls are available only after Join. Changing protocol clears the selected source. The instance can hide its own **MXL** output flows from this list so you do not loop an MXL output back into the meeting. Its own NDI outputs stay listed; looping one of those back is subject to the port-overlap problem in §5.4.



If you would like to stream the output feeds back into the meeting, please make sure to disable "Hide ZoomISO Output Flows from Input Selection" under **Settings** → **MXL**



6.1 Instance configuration

On the instance card, **Configure** exposes settings that apply to every output on that instance:

Setting	What it does
MXL domain	Directory where this instance reads and writes MXL flows. Changing it rebuilds all writers. Must be an absolute, writable path.
MXL group prefix	Stem used in generated NMOS grouphint tags and default output names. Changing it restarts active outputs.
Video loss mode	What an unbound or stalled output emits: Black , Freeze , or Test Card .
Video loss delay	How long a stalled bound stream waits before falling back to the loss pattern (low 0.5s, medium 1.0s, high 2.0s).
Default output format	Format used for newly added rows (for example <code>1080p29.97fps</code>). Each row can then be changed independently.

Auto-start capture on join	When on, the instance requests recording permission as soon as it joins.
Hide output flows from input	Exclude this instance's own MXL outputs from the stream-into-meeting source list.

Settings → **NDI** is separate from the MXL settings above. **Discovery Server** is required for any container deployment (see §5.4). **Groups** is an optional comma-separated list (empty uses NDI's default group). **Network Adapters** limits NDI to selected interfaces. **Multicast** defaults off; leave it off on Docker. Applying any of these restarts every active NDI output, because NDI cannot change them on a sender that is already running.

If the native process crashes, the card shows a recovery overlay. The process does not auto-restart. **Restore Instance** rebuilds the last saved settings and outputs (needs the `zoomiso-state` volume). **Delete and restart** creates a new default instance and replaces the snapshot. Export logs from the overlay before replacing the instance if you need them.

Only one instance is permitted per container is supported in this build. Deploy multiple containers on different **dashboard** ports to capture multiple meetings. Do not publish the same host NDI ports (`5960-5990`) for more than one container: the second container either fails to bind, or its receivers see the first container's media under the second container's source name. See §5.4. A second container on the same host should stay on MXL, or move to another host if it must send NDI.

7. Observability

Container logs

```
1 docker logs -f zoomiso
```

The web server and the native meeting SDK instance both log to stdout/stderr.

Log lines follow a structured JSON-ish format with timestamps and a source tag (e.g. `server.vite`).

Stopping / restarting

```
1 docker stop zoomiso          # graceful stop; the native instance is given 5s before SI
2 docker start zoomiso
3 docker restart zoomiso
```

8. Troubleshooting

Symptom	Likely cause / fix
Dashboard loads but Sign in never completes and the license banner says unlicensed.	The signed-in Zoom account is missing the Zoom Enhanced Media license and is not an active Zoom Events hub host. Use a qualifying account or contact your Zoom account manager.
Sign in dialog errors out.	Container has no outbound connectivity to <code>zoom.us</code> / <code>liminal.zoom.us</code> . Check the host firewall and any outbound HTTPS proxy.
Instance joins but no audio is captured.	PulseAudio did not start. Check <code>docker logs zoomiso</code> for PulseAudio errors. Ensure <code>--shm-size</code> is at least <code>512m</code> . Running the container with a UID that has no write access to <code>/tmp</code> or <code>/var/run</code> can also cause this.
Instance joins but cannot subscribe to a participant.	The Zoom account used to sign in lacks host/co-host/local-recording permission for that meeting. Host the meeting with that account or grant the permission from the meeting host.
Join stays disabled after Leave .	Expected. The Zoom Meeting SDK rejects a new Join until the previous meeting has fully

	ended. Wait until the Join button re-enables.
Native process dies; card shows a recovery overlay.	The native child does not auto-restart. Restore Instance (persistence volume mounted) or Delete and restart . Export logs from the overlay before replacing the Instance if you need them.
External tooling cannot see MXL flows on the host.	Make sure the container was started with <code>-v <host-dir>:/app/mxl-main/mxl-domain</code> (§5.2 Option B) and <code>--ipc=host</code> , and that the host directory is on a local filesystem with adequate permissions.
Downstream app still uses yesterday's flow IDs.	Flow IDs are minted per output and change when writers are rebuilt or the Instance is restored. Copy the current IDs from Current MXL flows (§5.3).
Bound output goes to black/test card after the host opens breakout rooms.	Expected if that participant is not in the same breakout room as the instance. Re-bind from the refreshed participant list. Set <code>AUTO_JOIN_BREAKOUT_ROOMS=false</code> if you need the instance to stay in the main session unless force-moved.
WebSocket disconnects every few minutes when running behind a reverse proxy.	The proxy has a low idle timeout. Set it to at least 120 s, and ensure it forwards <code>Upgrade</code> / <code>Connection</code> headers on <code>/socketio</code> and <code>/ws</code> .
"exec format error" on <code>docker run</code> .	The host is not x86_64. This image is <code>linux/amd64</code> ; <code>uname -m</code> must report <code>x86_64</code> .
MXL shared-memory errors (<code>shm_open</code> , <code>mmap</code>).	Increase <code>--shm-size</code> ; on small VMs the default 64 MB is insufficient.

NDI

Symptom	Likely cause / fix
Output protocol is NDI, but no receiver lists the source.	Discovery server is empty, unreachable from inside the container, or the receiver is pointed at a different server. mDNS will not cross Docker. Set Settings → NDI → Discovery Server to an address the container can open (see §5.4), and point the receiver at that same server.
The source appears in the picker, but the receiver never shows video (no connection).	Media ports are not published, only TCP was published, or the receiver has no routable Extra IP . Publish <code>5960-5990</code> for both tcp and udp, one-to-one, and add the Docker host's reachable address as an Extra IP. Open that range on the host firewall.
The source name is correct, but the picture or audio is a different NDI feed (Test Patterns, another app, or the other container).	Host-port overlap. Something else on this host already owns the advertised <code>ip:port</code> . Stop the other sender, or move this container to a host whose <code>5960-5990</code> range is free. Shifting the publish (<code>-p 7000:5960</code>) does not help: NDI still tells receivers to dial the container port.
A second container fails to start, or its NDI output shows the first container's media.	Both containers want host ports <code>5960-5990</code> . Only one NDI-publishing ZoomISO container per host. Give the second container a different dashboard port and leave it on MXL, or run it on another host.
Receivers drop for a moment after you apply NDI settings.	Expected. Groups, discovery server, adapters, and multicast are applied by recreating the senders.

9. Security notes

- **Exposure.** The dashboard has no built-in authentication beyond the Zoom OAuth sign-in it requires before any meeting-control action. Do **not** expose port `3000` directly to untrusted networks. Front it with your own reverse proxy that adds TLS and, ideally, network-level ACLs or SSO.
- **NDI media ports.** `5960-5990` TCP and UDP carry live meeting audio and video with no ZoomISO login on the stream. Publish them only on the broadcast network, and do not put that range on a public interface. See §5.4. The discovery server (TCP 5959) should be reachable only by machines that are supposed to see these sources.
- **OAuth tokens.** Zoom access tokens and ZAKs are kept in memory only; they are never written to disk and never logged. They are discarded on **Sign out** and on container stop.
- **Meeting content.** Captured participant media is written to the MXL domain directory as shared-memory files. Anything that can read that directory can read meeting content. When using §5.2 Option B, `--ipc=host` plus a world-writable host directory means **any local process** can read live meeting media. Restrict directory permissions and who can run on the host.
- **Container user.** Standalone images run as `root`. That is required for PulseAudio / dbus in this build; do not assume a dropped-privilege user.
- **Updates.** The image is immutable; to deploy a patched release, load the new `.tar`, `docker stop && docker rm zoomiso`, then `docker run ...` again with the new tag.

10. Quick reference

```
1 # Load
2 docker load -i zoomiso-cloud-amd64-<tag>.tar
3
4 # Run (minimal, MXL only)
5 docker run -d --name zoomiso -p 3000:3000 zoomiso-cloud-amd64:<tag>
6
7 # Run (shared MXL domain + persistence + tuned shm)
8 docker run -d --name zoomiso \
9     -p 3000:3000 --shm-size=2g --ipc=host \
10     -v /var/lib/zoomiso/mxl-domain:/app/mxl-main/mxl-domain \
11     -v zoomiso-state:/app/web_app/persistence \
12     zoomiso-cloud-amd64:<tag>
13
14 # Run with NDI. Ports must be 1:1. Do not reuse 5960-5990 on this host.
15 # Discovery server is NOT published here; point Settings → NDI at it (port 5959).
16 docker run -d --name zoomiso \
17     -p 3000:3000 \
18     -p 5960-5990:5960-5990/tcp \
19     -p 5960-5990:5960-5990/udp \
20     --shm-size=1g \
21     -v zoomiso-state:/app/web_app/persistence \
22     zoomiso-cloud-amd64:<tag>
23
24 # Inspect
25 docker logs -f zoomiso
26 docker exec -it zoomiso bash
27
28 # Upgrade
29 docker stop zoomiso && docker rm zoomiso
30 docker load -i zoomiso-cloud-amd64-<new-version>.tar
31 docker run -d --name zoomiso -p 3000:3000 zoomiso-cloud-amd64:<newtag>
```

Installation and Accounts

Sign in with Zoom - Adding the App to your Account

ZoomISO Cloud has an online sign-in button which allows you to use your Zoom account to log into the app. Signing in allows ZoomISO Cloud to start and join meetings on behalf of the signed in account, access the display name and other account details, and join Zoom Meetings, Zoom Webinars, and Zoom Events.

To sign into ZoomISO Cloud:

1. Launch ZoomISO Cloud
2. Click the "Sign In" button
3. You will now be presented with a Zoom login webpage to sign into.

If this is the first time you have logged into ZoomISO Cloud, you will be prompted to add ZoomISO Cloud to your Zoom account after signing in. Click Allow.

1. This page should redirect you to the `liminal.zoom.us` login success page where you should proceed to connect to ZoomISO Cloud by pasting the Login Code into the app.
2. If the above steps are successful, you will be logged into ZoomISO Cloud. You will see your profile image in the top right corner of ZoomISO Cloud where you can view your account in the Zoom Web Portal or sign out.

Enterprise Provisioning

Are you a business/enterprise end-user? You may need to contact your IT department to allow ZoomISO Cloud to be added to your Zoom account, depending on how your organization is configured. IT can add ZoomISO Cloud to your organization at:

<https://marketplace.zoom.us/apps/cehXlnVcTxOBQ2vDpPDczA>

Removing ZoomISO Cloud from your Zoom Account

Login to your Zoom Account and navigate to the Zoom App Marketplace.

- Click Manage > Added Apps or search for the "ZoomISO Cloud" app.
- Click the "ZoomISO Cloud" app.
- Click "Remove".