



Tactical Media Broadcast Hub

Independent Self-Hosting: Technical Brief

This document describes what an independent collective needs to operate its own Tactical Media Broadcast Hub.

It is intentionally separate from the public presentation brief. The public brief explains what people can do with the Hub; this document explains the systems, storage, security, and operational responsibilities behind it.

1. Operating Principle

The Hub should keep user file and media payloads in durable object storage, not on the application host.

The practical boundary is:

- R2/S3: user files, media, document payloads, pad exports, and large objects.
- Local service state: small metadata databases, indexes, room state, configuration, queues, and temporary processing data.

A literally zero-local-state deployment is not supported by every component. The goal is zero local storage for user payloads, with the remaining metadata protected by backups and volume snapshots.

2. Recommended System Layout

Control plane

- AnarchoPoster web application.
- API and authentication.
- PostgreSQL.
- Redis.
- Temporal or an equivalent workflow service.
- Scheduler and media-worker control services.

Conversation plane

- Tuvunel Matrix homeserver.
- mautrix-telegram bridge.
- Element or another Matrix client.
- A small archive bot for approved rooms and attachments.

The current deployment uses Conduwuit, but Conduwuit is archived and does not provide the native S3 media storage needed for the long-term design. Plan a controlled Conduwuit-to-Tuvunel migration before moving Matrix media to R2.

Media plane

- PeerTube.
- RTMP/RTMPS ingest.
- Transcoding and broadcast workers.
- Preview and archive processing.

The media plane should be separate from the files/docs plane when the collective runs sustained livestreams or video processing.

The broadcast system supports persistent virtual channels rather than only one-off uploads. A channel can combine live sources, playlists, and prepared programs into a continuous output.

Core production capabilities include:

- Source ingest from live feeds, PeerTube, files, and playlists.
- Scene composition with grids, picture-in-picture, shared sources, and custom layouts.
- Preview and program buses for preparing and taking a source live.
- A/B switching and frame-continuous source replacement.
- Lower thirds, chyrons, logos, clocks, and other typed overlays.
- Per-source audio selection, gain, mixing, and master output control.
- Scheduled playlists with slate fallback when an asset is unavailable.
- RTMP/RTMPS output to PeerTube and other supported destinations.
- JPEG and low-latency HLS preview taps for operators and control-room clients.
- Recording, replay, archive, and later clip/export workflows.

The media worker is the production engine. The control plane owns channel state, permissions, schedules, and signed commands; it should not carry raw video bytes.

Files and documents plane

- ownCloud Infinite Scale (oCIS) for folders, group files, search, versions, and share links.
- Collabora CODE or OnlyOffice for collaborative ODT and office-document editing.
- Tika or equivalent content extraction if full-text search is required.

oCIS is the recommended primary files layer because its S3NG storage driver keeps file blobs in R2 while retaining only necessary metadata on a local POSIX volume.

Pad plane

- Etherpad or HedgeDoc for fast shared pads.
- Pads are for live notes, run sheets, planning, and event coordination.
- Pad text and revisions should remain in the pad service, with exports stored in R2 when retained as formal project records.

Matrix is the conversation layer, not the office editor or files library.

3. R2 Bucket Layout

Use separate buckets. Do not share an exclusive or opaque application bucket between services.

The initial dedicated buckets are:

Bucket	Purpose
hub-matrix-media	Matrix/Tuwunel media payloads

Bucket	Purpose
hub-files-blobs	oCIS file and document blobs
hub-media-library	AnarchoPoster working media and library assets
hub-pads	Retained pad exports and attachments

Each service needs its own scoped R2 S3 credential. Cloudflare account API credentials are not the same thing as R2 S3 access keys.

Do not place credentials in Git. Store them in an encrypted secret manager or protected deployment environment.

4. Group File Contract

Every archived group file should retain enough context to answer: what is this, who contributed it, where did it come from, and who may use it?

Minimum metadata:

- workspaceId
- groupId
- roomId
- eventId
- senderId
- origin such as Matrix, Telegram bridge, upload, or application
- filename
- mimeType
- sizeBytes
- sha256
- createdAt
- storageKey
- shareUrl when one exists
- retention
- status such as pending, archived, published, revoked, or deleted

The Hub database stores metadata and indexing information. The file bytes live in the appropriate R2 bucket.

Archive writes must be idempotent on the combination of workspace, room, and event ID. A retried sync event must not create duplicate files.

5. Matrix and Telegram Media Flow

1. Telegram content enters through mautrix-telegram.
2. The Matrix room remains the conversation and attribution source.
3. An archive bot watches designated rooms.
4. For eligible file/image/video/audio events, the bot streams the payload into the files platform or R2-backed archive.
5. The bot records room, sender, event, origin, checksum, and retention metadata.
6. The bot can post a share link back to the room instead of duplicating the file.

Archive policy

Encrypted rooms cannot be read by a server-side archive bot without a deliberate decryption path. The default safe policy is:

- Archive only designated unencrypted archive rooms.
- Do not silently decrypt or mirror private E2E rooms.
- Make the archive policy visible to room members.
- Preserve attribution and the original event ID.

The first implementation should be a plain sync-loop bot using matrix-js-sdk, not a full appservice. An appservice can be considered later if room volume requires it.

6. Collaborative ODT Editing

ODT and other office documents should be stored in oCIS and opened through a WOPI-compatible editor such as Collabora or OnlyOffice.

The Hub should store:

- The document identity.
- The owning group and workspace.
- The current share or launch reference.
- Review and publication status.
- A rendered export or snapshot when a document is approved.

The Hub should not copy the full document revision history into its own database. The files platform and office editor remain responsible for collaborative editing and document versions.

7. Shared Pads

Pads should have a small integration boundary:

- Group and room ownership.
- Pad URL and service identity.
- Membership and access policy.
- Export-to-file action.
- Optional retention policy.

Pad content stays with the pad service during editing. When a pad becomes an official project artifact, export it to ODT, Markdown, HTML, or PDF and store the export in R2-backed group files.

8. Access and Security

- Use group roles rather than shared account passwords.
- Prefer OAuth/OIDC/delegated authorization for external platforms.
- Scope each platform connection to the smallest useful permission set.
- Keep publishing, reviewing, editing, and administration as separate capabilities.
- Record access changes and publication actions in the audit trail.
- Rotate credentials when a member leaves or a platform connection changes.
- Use separate R2 credentials per bucket and service.
- Do not put R2 secrets, Matrix registration tokens, or platform refresh tokens in the repository.

9. Compute and Capacity

Small pilot

- 4 vCPUs and 8 GB RAM is a comfortable starting point for control, chat, and files without heavy video processing.
- 2 vCPUs and 4 GB RAM can support a reduced pilot, but not the full stack with Collabora, search, PeerTube, and sustained livestreaming.
- Keep large media processing on a separate host whenever possible.

Full deployment

The honest target for the complete Hub is approximately 4 vCPUs and 8 GB RAM or more. The main elastic components are:

- Collabora/OnlyOffice document sessions.
- Tika content extraction.
- PeerTube transcoding.
- Live broadcast workers.
- Initial search/indexing jobs.

Disk capacity is less important than keeping payloads in R2. A self-hosting operator should still maintain enough local space for metadata volumes, temporary processing, logs, and recovery snapshots.

10. Domains and Network

At minimum, an independent deployment needs:

- One main web domain.
- Separate hostnames for API, Matrix, files, collaboration, pads, and video.
- HTTPS for every public service.
- A tunnel or reverse proxy.
- Long upload and WebDAV/WOPI timeouts.

Large-upload hostnames should not use a proxy path with a low request-size limit. Use a tunnel or grey-cloud DNS path, or use chunked/TUS uploads.

11. Backups and Recovery

Back up both payloads and the metadata that makes them meaningful:

Artifact	Backup
R2 buckets	Replication, lifecycle protection, or a second-account backup
Tuwunel room state	Volume snapshot and config backup
oCIS metadata volume	Volume snapshot; use a filesystem suited to the service
Hub PostgreSQL	Regular pg_dump plus tested restore
Redis/Temporal	Rebuildable state documented; preserve durable workflow data where required
Deployment config	Versioned config with secrets stored separately

Test restoring a group, its files, document links, and publication history onto a clean host. R2 durability alone does not protect against a bad lifecycle rule, deleted metadata, or leaked credentials.

12. Deployment Phases

Phase 1: Foundation

- Create separate R2 buckets and scoped S3 credentials.
- Deploy the files platform without migrating existing Matrix or PeerTube data.
- Establish group/file metadata contracts.
- Add backup and restore procedures.
- Add a basic pad service.

Phase 2: Collaboration

- Add Collabora or OnlyOffice.
- Connect ODT launch and export flows.
- Add group library views and share links.
- Add Matrix room-to-group mappings.

Phase 3: Archive integration

- Upgrade Conduwuit to Tuvunel with the permanent server name preserved.
- Enable Tuvunel media storage providers for R2.
- Run the archive bot for designated unencrypted rooms.
- Add event-to-file indexing and share-link replies.

Phase 4: Production hardening

- Add OIDC or another shared identity layer.
- Add retention and deletion workflows.
- Add search extraction and indexing.
- Run restore drills and permission audits.
- Separate heavy media processing onto its own host.

13. Explicit Non-Goals

- Do not run Nextcloud AIO as the default; it is local-datadir-oriented and wants to own the Docker host.
- Do not use Seafile CE if R2 storage and content search are requirements.
- Do not make Matrix the files library.
- Do not run a second chat platform beside Matrix.
- Do not migrate live Matrix data to R2 until the Tuvunel migration is planned and backed up.
- Do not claim that user payloads are R2-backed until the service has been configured with real scoped R2 credentials and verified with an upload/restore test.