The Problem

- M4 Mac (macOS Tahoe) + Pop!_OS PC (Linux) need to talk to each other
- From home and from anywhere
- Behind CGNAT (no public IP, can't port forward)
- $0 budget
- No middleman if possible

The Pieces

1. WireGuard (the tunnel)

NOTE: BUILT. Configs at infrastructure/wireguard/ + Headscale at infra/headscale/. Also integrated into transport priority stack (services/transport/transport.go).

What it does: Encrypted tunnel between your devices. Once connected, they see each other like they're on the same network. SSH, file sharing, everything works through it.

Cost: Free. Built into Linux kernel. Free app on Mac/iOS/Android.

Depends on: Nothing — it's just two devices talking to each other. But they need to FIND each other first (that's the signaling problem below).

2. The Signaling Problem (how devices find each other behind CGNAT)

NOTE: SOLVED DIFFERENTLY. FRP (infrastructure/frp/) handles CGNAT bypass. Yggdrasil (infrastructure/ops/config/yggdrasil.conf) provides global IPv6 mesh with automatic LAN discovery. Headscale provides DERP relay (STUN port 3478). The signaling options below are still valid for ADDITIONAL use cases (notifications, canary, Nostr publishing) but NOT needed for basic device connectivity.

Both devices are behind CGNAT. Neither has a public IP. They can't find each other directly. They need a tiny coordination point — something that says "hey, Mac is currently at this address, Linux box is currently at that address." Once they know, WireGuard connects direct.

Options (pick any one):

METHOD	HOW IT WORKS	ACCOUNT NEEDED	SELF-HOSTABLE	REAL-TIME
Cloudflare DNS TXT	Update a TXT record with IP:port. Other device does dig.	You already have CF	N/A (your domain)	No (DNS cache delay)
GitHub repo	Push IP:port to a file via API. Other device reads it.	You already have GH	No	No (polling)
GitLab/Codeberg	Same as GitHub	Free account	Codeberg is open source	No (polling)
ntfy.sh	Publish IP:port to a topic. Other device subscribes. Instant.	None	Yes	Yes
Nostr	Publish to decentralized relay. Keypair only. No account.	None	Yes	Yes
MQTT	Lightweight IoT messaging. Free brokers exist (HiveMQ).	Free account	Yes	Yes

Cloudflare Worker	Tiny script stores IP:port. Devices read/write via HTTP.	You already have CF	N/A	No (polling)
Cloudflare R2	Store a tiny file. S3-compatible.	You already have CF	N/A	No (polling)
Upstash Redis	Free tier REST API key-value store.	Free account	No	No (polling)

Recommendation: ntfy.sh for real-time + self-hostable. Cloudflare DNS TXT as backup (you already own it). GitHub as third backup. Three independent signaling methods means no single point of failure.

---

3. Syncthing (file sync)

NOTE: NOT INTEGRATED. Koofr sync is used instead. Syncthing could still add value for real-time bidirectional sync without cloud dependency.

What it does: Keeps folders in sync between devices automatically. Change a file on Mac, it appears on Linux box. Bidirectional.

Cost: Free, open source.

Depends on: Devices finding each other (works over WireGuard tunnel, or discovers on local network automatically).

Good for: Keeping your workspace, research, wiki content, and FOI documents synced across both machines without touching anyone's cloud.

---

4. SSH (remote terminal)

NOTE: BUILT. Works over WireGuard/Yggdrasil tunnels. Ansible playbooks provision eternal_terminal + tmux (tools/ansible-networking/client-setup.yml).

What it does: Terminal access from one machine to the other.

Cost: Free. Built into both operating systems.

Depends on: Network connection (works over WireGuard tunnel).

Good for: Running commands on the Linux box from the Mac at a cafe. Or vice versa.

---

5. Input Leap (shared keyboard/mouse)

What it does: One keyboard and mouse controls both screens. Move your cursor off the edge of the Mac screen, it appears on the Linux screen.

Cost: Free, open source (fork of Synergy).

Depends on: Both machines on same local network (or WireGuard tunnel).

Good for: When they're sitting side by side on your desk.

---

6. Cloudflare Tunnel (web services)

NOTE: BUILT. Sovereign Network Worker at services/sovereign-network/ proxies all traffic at $0. Multiple CF Workers deployed (anchor-worker, sovereign-network, cloudflare worker). Pages hosts wiki, search, and other sites.

What it does: Exposes local web services through Cloudflare. Wiki, dashboard, anything running on your home machine becomes accessible at a URL.

Cost: Free.

Depends on: Cloudflare (you already use it for everything).

Works behind CGNAT: Yes — connects outbound.

Not good for: SSH (requires clunky wrapper on client side). Use WireGuard for SSH instead.

---

7. ntfy.sh (notifications + signaling)

NOTE: NOT BUILT. Still valuable for: push notifications, deploy alerts, Seren's heartbeat publishing, FOI reminders, canary updates. Should be self-hosted on the Pop!_OS box.

What it does: Publish/subscribe messaging. Send a message to a topic, anyone listening gets it instantly.

Cost: Free. No account.

Self-hostable: Yes.

Good for:

- WireGuard signaling (device posts its IP:port every 30 sec)
- Push notifications to your phone (deploy done, wiki edited, canary updated)
- Seren's heartbeat could publish here
- FOI deadline reminders
- Anything you want to know about from anywhere

---

8. Warrant Canary (transparency)

NOTE: PARTIALLY BUILT. Template exists at content/dev/templates/canary_js.txt. Not wired up to wiki.omxus.com or ntfy.sh yet.

What it does: A signed statement published regularly saying "we have not received secret government orders." If it stops updating, that IS the message.

Cost: Free.

How it connects:

- Signed with your secp256k1 key (same as OMXUS auth)
- Published to wiki.omxus.com
- Update notification pushed via ntfy.sh
- Subscribers (anyone) get notified each update
- If notification stops → canary is dead → everyone knows

---

9. Nostr (decentralized messaging)

NOTE: NOT BUILT. Key insight: Nostr uses secp256k1 = same curve as VexID/Ethereum. Your OMXUS identity IS your Nostr identity. Zero new crypto needed. Should be prioritised for censorship-resistant publishing and identity bridging.

What it does: Publish messages to decentralized relays. No account — just a keypair. No one controls it.

Cost: Free.

Good for: Signaling, public statements, canary publishing, communication that no one can censor or shut down.

Aligns with: Everything OMXUS stands for. No middleman. Keypair identity. Censorship-resistant.

---

10. Headscale (self-hosted Tailscale)

NOTE: BUILT. Config at infra/headscale/docker-compose.yml. DERP relay, magic DNS (.omxus.net), SQLite, ready to deploy.

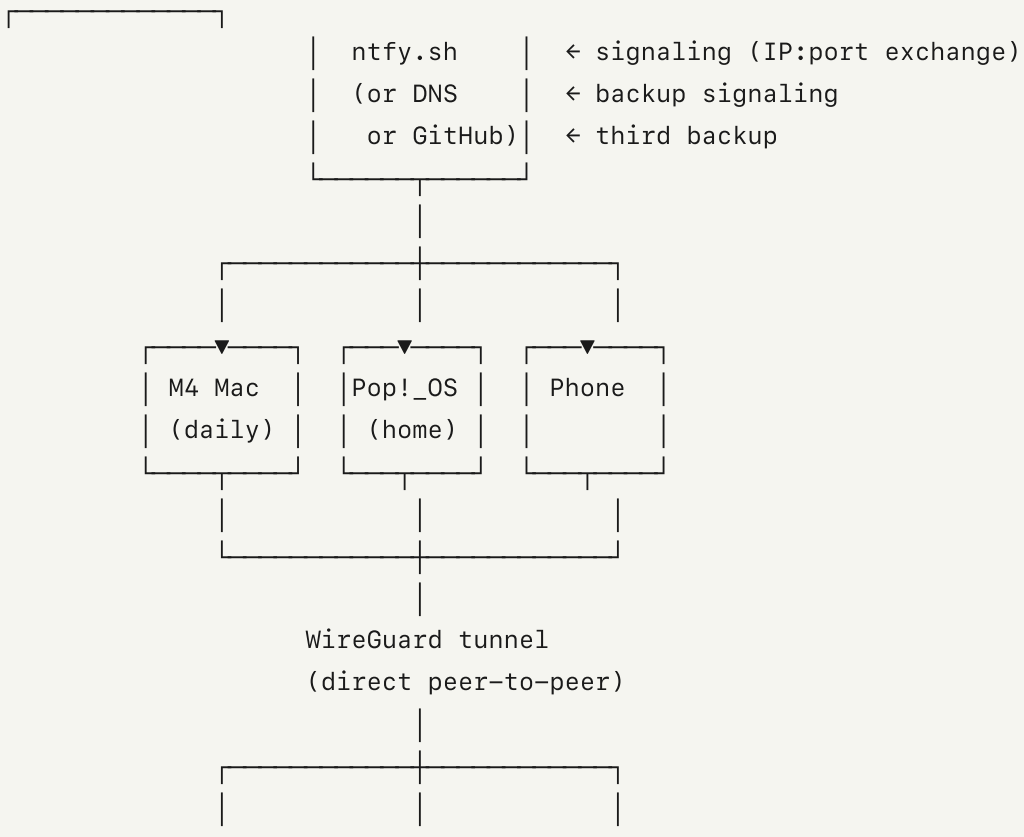
What it does: Same as Tailscale (mesh VPN, NAT traversal, devices find each other) but you run the coordination server yourself.

Cost: Free.

You already have it: Config exists in your `infra/` directory.

Depends on: A server to run the coordination (could be the Pop!_OS box + Cloudflare Tunnel to make it reachable, or any free VPS).

How It All Fits Together



```
          Syncthing    SSH access    Input Leap
          (file sync)  (terminal)   (shared kb/mouse)
```

The flow:

1. Each device runs a tiny script every 30 seconds
2. Script learns its own public IP:port (via STUN)
3. Script publishes IP:port to ntfy.sh topic (and/or DNS TXT, and/or GitHub)
4. Script reads other devices' IP:port from the same place
5. Script updates WireGuard with the new endpoint
6. WireGuard connects directly between devices — no relay, no middleman
7. Over that tunnel: Syncthing syncs files, SSH gives terminal access, everything works

What you depend on:

THING	CAN IT DISAPPEAR?	BACKUP PLAN
WireGuard	No — it's local software, open source, in the kernel	N/A
Syncthing	No — local software, open source	N/A
ntfy.sh (public)	Yes — one guy runs it	Self-host it on your Linux box
GitHub API	Unlikely but yes	Switch to GitLab, Codeberg, or DNS
Cloudflare DNS	Unlikely but yes	Switch to GitHub or ntfy
STUN servers	Google's could go	Many free alternatives exist

Nothing in this stack costs money. Nothing requires a specific provider. Everything has a backup. Everything is open source. You own all of it.

What To Build First

1. Install WireGuard on both machines (5 min)
2. Install ntfy on the Linux box — self-hosted, no dependency (10 min)
3. Write the signaling script — 20 lines of bash on each device (10 min)
4. Install Syncthing on both — point at workspace folder (5 min)
5. Test from a cafe — turn on WireGuard on Mac, SSH into Linux box

Total: about 30 minutes and you're connected indefinitely for free.

Later (when you want)

- Warrant canary on wiki.omxus.com, signed, pushed to ntfy
- Nostr for censorship-resistant publishing
- Headscale if you want more devices with zero config
- Input Leap when they're side by side
- Cloudflare Tunnel for exposing wiki/services to the public
- Two people open the same wiki page
- Hyperswarm connects them (same topic key = same document)
- WebRTC data channel opens — direct, browser to browser
- Yjs (CRDT) syncs keystrokes in real-time
- Two cursors, same doc, no server processing edits

- When done, one commits to GitHub

Libraries:

- Yjs — CRDT for real-time text, has WebRTC provider built in
- Loro — Rust-based CRDT, faster, newer
- Automerge — another CRDT, works offline too

The "Google Docs" experience except data goes browser-to-browser and saves to GitHub.

Canary Tokens (documents that connect)

A document with an embedded invisible resource (1x1 pixel image, DNS lookup, or external template reference). When someone opens the document, it reaches out to a URL. The server logs who, when, where.

How it works:

- PDF/Word contains a hidden 1px image:
- Document opens → reader loads the image → server logs the request
- IP, time, device, location — all captured
- Unique token per recipient = you know exactly who opened it

Harder-to-block version: DNS canary. Document references a unique hostname. Even if image loading is blocked, the DNS lookup still happens. Almost impossible to prevent without breaking everything.

The connection potential:

A document that creates a room just by being opened:

1. Someone opens the document
2. The embedded resource triggers a notification (ntfy.sh / WebSocket)

3. The author knows someone is reading right now
4. WebRTC connection auto-established via the canary callback
5. Reader and author are now live-connected
6. Multiple readers of the same document find each other

A document is not a file. It's a doorbell. Reading becomes connecting.

The author feels the document being read — not analytics, not "542 views," but a living presence. Someone is with your words right now.

The OMXUS Identity Thread

Nostr uses secp256k1. Same curve as Ethereum. Same curve as VexID. Your OMXUS identity IS your Nostr identity IS your mesh address IS your document signature IS your canary key.

One key, everywhere:

- Signs into wiki.omxus.com
- Signs WireGuard peer config
- Signs Nostr messages
- Signs canary statements
- Signs WebRTC offers
- Verifies document authorship

Concept #1 (one identity everywhere) connects to concept #14 (vouched chat) connects to concept #21 (peer-to-peer without infrastructure). It's all the same key.

CRDTs + Offline-First

Automerge/Loro/Yjs solve the vouched chat and collaboration problem:

- Two vouched humans edit or chat
- Works offline — on a plane, in a dead zone, wherever
- Syncs when they reconnect — no conflicts, no "which version is right"
- No server storing messages
- The mesh IS the chat infrastructure

The Full Picture

```
Person opens OMXUS app (or document)
    → Hyperswarm joins the mesh (automatic, no config)
    → Finds other OMXUS users (DHT, no server)
    → WebRTC connects them directly
    → Yjs/Automerge syncs state in real-time
    → WireGuard available for persistent encrypted tunnel
    → Canary tokens make documents into live connections
    → ntfy.sh for notifications (self-hosted)
    → Nostr for censorship-resistant publishing
    → All signed with one secp256k1 key
    → Everything encrypted, everything peer-to-peer
    → Nothing costs money
    → Nothing depends on anyone
```

A person doesn't "join a network." They ARE the network the moment they open the app. The document doesn't just contain words. It contains a doorway. Reading is connecting. The act of opening is the act of arriving.

The 23 Concepts (Updated)

NOTE: This list has grown from 21 to 23. Concepts 1-13 map to existing OMXUS goals. Concepts 14-23 are design principles from this research. Several are implemented (1, 2, 3, 7, 10), most are partially built or not started.

One identity, everywhere

$29 ring — hardware second factor

Zero-trust without the complexity

Policy as code as prose as law

Single binary, every platform

The mesh IS the distribution

Device attestation without surveillance

Brutal honesty built in

Selfishness as infrastructure

Every decision auditable

Call of Duty — protagonist game mechanics

Step up — challenges, never pleads

Voice of god — system has personality

Vouched chat — verified humans, real talk

WebRTC live collab

Proof of work, not proof of opinion

The system handles the awkward part

Building codes — the door is already open

Don't punish the helper

The app's job is to get you off the app

Peer-to-peer without infrastructure — Hyperswarm DHT, WireGuard, STUN, no server

Documents as doorways — canary tokens, reading as connecting, presence not analytics

The Slave gets fed — fitness has purpose, bouldering because you're a responder, fight club without the violence

14 nodes armed. You're covered. Go climb something.