Atho Node Quickstart

Status: Alpha documentation snapshot (2026-03-12).

This walks you through running a full node, miner node, and wallet/API node, then using the CL
All auth is secured via API key + user/pass, and ports are auto-assigned to avoid collisions.

This is the fastest setup path. For the full docs map, start at the documentation index. For developer-oriented setup, use ONBOARDING.md.
If anything fails during setup, check Troubleshooting.md first.

Prereqs (why you need them)
- Python 3.11+ (or 3.9+) to run nodes and CLI.
- git and curl for fetching code and checking APIs.
- macOS/Linux/WSL recommended (Windows native works but WSL is smoother for toolchains).
- Prefer containers? See the Docker guide for building/running via docker compose (full/miner/wallet) as an alternative to local toolchains.
- Hardware guidance: OpenGL 2.0 compatible system, i3-class CPU or better, 4-8 GB RAM minimum, 120 GB SSD recommended.
- Get the code + venv + deps
macOS / Linux / WSL:

Windows (PowerShell):

Why: venv keeps deps isolated; requirements pulls FastAPI/LMDB/requests, etc.
If install fails (especially lmdb/kivy/cryptography), go to Troubleshooting.md -> Dependency install issues (venv + pip).
- Create an API key (required)
- Pick a username/password (default user is atho). Password set = full permissions; blank password = read-only.
- Writes Src/Config/Api_Keys.json (keep private; chmod 600).
Why: All API calls (including CLI) require X-API-Key/X-User/X-Pass to prevent misuse.

2.5) Falcon + Kyber binary/runtime check (important)
Falcon:
- Runtime expects falcon_cli for your host platform.
- Install/register current host binary:
- If Falcon is missing, build with platform compiler first:
- macOS: clang (Xcode Command Line Tools)
- Linux: gcc/clang (build-essential)
- Windows: MSYS2 MinGW-w64 (gcc) or Visual Studio Build Tools (cl)

Falcon compile commands (full flags):

macOS (Apple Silicon/Intel):

Linux:

Windows (MSYS2 MinGW-w64):

Kyber:
- Wallet encryption/decryption paths use vendored Kyber and can auto-build shared libs.
- Ensure a C compiler is installed (cc/clang/gcc) so Kyber build can succeed when first needed

Kyber compile commands (full flags, Kyber-1024 / KYBER_K=4):

macOS:

Linux:

Windows (MSYS2 MinGW-w64):
- Join the network (start nodes)
- Ports auto-assign and are saved to Src/Config/NodePorts.json (P2P base is 56000).
- To point at an existing seed:
- Check sync: tail -f logs/<network>/network/network.log and look for handshake_ok / sync_progress with growing tip_height.
- Verify via API (optional but recommended)
- Full API port is in NodePorts.json (default ~10100). Success shows network, tip height/hash.
- Optional network hashrate telemetry check:
- Use the CLI
macOS / Linux / WSL:

Windows (PowerShell):
- CLI loads Api_Keys.json and NodePorts.json; prompts for password if not set in env.
- Common commands: wallet addresses|balance|new [12|24|48]|recover|import|export|send, blocks recent|height <n>|hash <hash>|latest|tx <txid>|export500, mining start|stop|status, system peers.
- wallet new defaults to 24-word mnemonic generation when word count is not provided.
- Launch desktop GUI + Web Explorer (optional)
macOS / Linux / WSL:

Windows (PowerShell):
- From the GUI, click the explorer icon to open the browser-based explorer.
- Explorer supports search by block/tx/address and accepts Base56 or HPK address input.
- Explorer runs through a local GUI bridge (127.0.0.1, random free port) and proxies to your authenticated node API.

6.5) GUI first-run checklist
- Open GUI.
- Go to Settings.

- Set API key, username, password.
- Open Node tab and start nodes.
- If miner is not needed, run full + wallet only.

6.6) Request inbox workflow (GUI)
- Request inbox UI is currently disabled by default in this alpha build for stability.
- If enabled in code/runtime, the flow is:
- Open inbox tab.
- Select a request and click Send.
- Copy requester address.
- Paste same address into verify field (must match exactly).
- Continue to Send modal (recipient + amount prefilled; amount can be edited).
- The verify step prevents accidental/misdirected sends.

Common env vars (set before runnode/CLI)
- ATHO_API_URL ? override API base (CLI target).
- ATHO_API_KEY, ATHO_API_USER, ATHO_API_PASS ? avoid prompts.
- ATHO_P2P_PORT ? set specific P2P port; otherwise auto.
- ATHO_BOOTSTRAP ? seed list (ip:port[,...]) to join existing network.
- ATHO_API_URL + wallet recent pagination can be used to pull bounded history pages (limit/cursor) efficiently from the CLI/API.

Security essentials
- Every API call needs X-API-Key, X-User, X-Pass; keys live in Src/Config/Api_Keys.json.
- Local dev: HTTPS off by default; if exposing, run behind TLS/reverse proxy and consider enabling HMAC.
- Protect Api_Keys.json and backups; rotate keys/passwords; use read-only keys where possible.
- For release artifact integrity, generate and sign release file checksums:
- ./.venv/bin/python Src/Main/checksum.py build --root <release_payload_dir> --out <release_payload_dir>/checksums.sha256
- ./.venv/bin/python Src/Main/checksum.py sign --checksums <...>/checksums.sha256 --key-json <falcon_key.json> --release <version>
- ./.venv/bin/python Src/Main/checksum.py verify --checksums <...>/checksums.sha256 --signatur <...>/checksums.sha256.sig.json --root <release_payload_dir>

Quick troubleshooting
- Port in use: rerun runnode.py (auto-assign) or set a free ATHO_P2P_PORT/API port.
- 401/403: wrong/missing headers or missing permissions (send_tx, mining).
- Stuck tip / sync_no_headers: check ATHO_NETWORK and ATHO_BOOTSTRAP; ensure seed is correct a reachable.
- CLI can?t connect: verify API port in NodePorts.json, set ATHO_API_URL, ensure API is runnir
- unsupported hash type during mnemonic keygen: ensure dependencies are installed from requirements.txt (cryptography provides the SHA3-512 PBKDF2 path on Python builds lacking nati support).
- falcon_cli/toolchain pain: use Docker as an easy path:
- GUI node start issues / stale PIDs:
- list: ./.venv/bin/python Src/Main/stop.py --active
- stop all: ./.venv/bin/python Src/Main/stop.py --all
- restart from GUI Node tab or via Src/Main/runnode.py
- Check live P2P behavior in:
- logs/mainnet/network/network.log
- look for peer_connected, header sync, mempool activity, and block acceptance

- On low-power systems, skip miner and run only full + wallet.

Support: labs@atho.io

What to read next
- ONBOARDING.md for repo layout and developer workflow.
- ApiAuth.md for the full auth model.
- Troubleshooting.md if ports, sync, auth, or builds fail.