// WHITEPAPER ARCHITECTURE · MODELS · SECURITY · RELEASE 1.8.2

SELF-LEARNING HASHRATE EXECUTION, ENGINEERED IN RUST.

PUBLISHER Asgard Hashworks
RUNTIME native Rust · Windows 10/11 · Linux x86_64
SCOPE ODIN 1.8.2 · SLEIPNIR 2.6.1 · HUGINN
SOURCE tag v1.8.2

01 EXECUTIVE SUMMARY

Hashrate marketplaces move on a seconds-to-hours timescale, not nanoseconds. Speed is not the edge: knowing what a terahash is worth now, what it will be worth after the next retarget, and how the market will fill your bid, and committing capital only when that number is positive, is. ODIN computes this on a 2-second loop from live chain data and corrects itself against what your pool actually pays. The same model runs live on five venues (NiceHash, Braiins Hashpower, MiningRigRentals, Clore.ai, Vast.ai) and scores the sell side (OUTBOUND) in shadow. All of it runs inside the operator's network.

> **Engineering thesis.** Every parameter that can be measured is measured on the operator's own account, not assumed: the payout delay, the price step NiceHash accepts, the fill rate at each price, the hour-of-day dip pattern and the market regime. Every parameter that has to be estimated carries its uncertainty into the decision.

02 SYSTEM TOPOLOGY

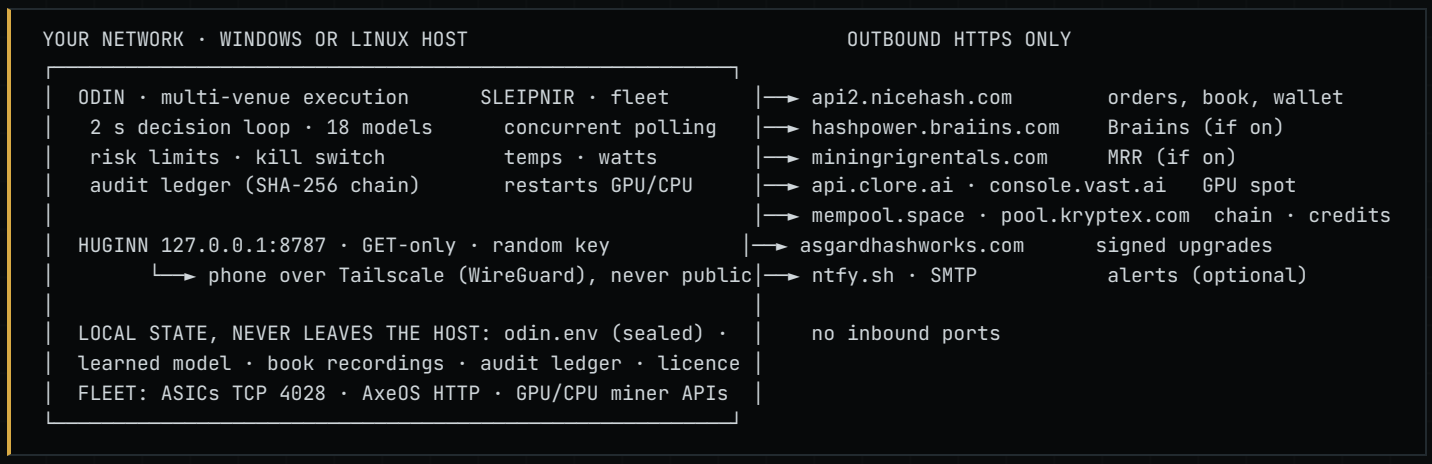


Figure 1. Execution boundary. Everything inside runs on the operator's hardware; every arrow is an outbound connection.

03 DECISION LOOP

1 · READ

Order book, difficulty, block rewards, own orders, pool credits, every 2 s. Stale (>20 s) or malformed data holds the desk.

2 · VALUE

Income per EH/s now and after the retarget, calibrated to pool credits, with its error band.

3 · DECIDE

Lock, park or pace: price and speed per order, inside every risk limit and the venue hold.

4 · ACT & LOG

Re-price or re-size within the venue's rules, then append the decision to the audit ledger.

Each cycle is six isolated steps (read, decide, orders, venues, records, present). Only the order desk may create, fund or re-price a NiceHash order; a test fails the build if any other module calls those functions.

04 CORE QUANTITATIVE MODELS

Notation: D difficulty, R block reward incl. fees, f_p pool fee, f_b NiceHash buyer fee (3%), p bid in BTC/EH/day, δ delivery ratio. Every model is unit-tested in the shipped build.

4.1 · HASHVALUE

$$I = 86400 \cdot 10^{18} \cdot R \,/\, (2^{32} \cdot D) \cdot (1 - f_p) \cdot k$$

Income per EH/s/day from live difficulty and the measured block reward, net of pool fee, times calibration k. Margin at bid p: $m = \delta I / (p(1+f_b)) - 1$.

4.2 · DELAY-AWARE CALIBRATION

$$k = \text{clamp}(\ \Delta C[t-W,t] \,/\, \Delta M[t-W-L,\ t-L],\ 0.85,\ 1.02\ )$$

Pool credits C vs. modelled earnings M, shifted by payout lag L, chosen by least squares on hourly increments. Live: credits at **100.6%** of model.

4.3 · RETARGET UNCERTAINTY

$$\sigma_c = (1+c) \cdot (B/2016) \cdot \sqrt{(1/n + 1/B)}$$

Rate-estimate error over n found blocks plus the luck of the B still to come; post-retarget income carries the Jensen correction E[1/(1+X)].

4.4 · DIFFICULTY GLIDE

$$p_{max}(t) = \min(\ p_{now},\ p_{post} + s \cdot \lceil (T-t)/600 \rceil\ )$$

NiceHash lowers prices one step s per 10 min (learned on your account), so the ceiling starts gliding early enough to reach p_{post} by the retarget.

4.5 · BUDGET PACING

$$\max \Sigma_h\ f_h L(\delta I - p(1+f_b)) \ \text{s.t.}\ \text{spend} \le B$$

With multiplier μ the problem separates by hour; μ found by bisection, μ = 0 when money isn't the limit.

4.6 · SPEND INTO THE DIP

$$L = L_0 + (L_{max} - L_0) \cdot \text{clamp}((m - m_{req})/2\sigma,\ 0,\ 1)$$

Speed scales with how far margin clears the requirement in model standard deviations; L_{max} capped at ¼ of profitable depth.

4.7 · CHANGE-POINTS

$$S^+ = \max(0,\ S^+ + z - k) \cdot \text{alarm if } S > h$$

Two-sided CUSUM (k = 0.5, h = 6) plus Bayesian online change-points; after a shift, older fill history counts at 25%.

4.8 · PARTIAL-FILL CURVE

$$q(g) = \text{PAV}(\ (\Sigma\ \text{share} + n_0 q_0)/(n + n_0)\ ),\ g = p - p_{clear}$$

Served share of each order by its gap to the clearing price; Bayesian bin means made monotone by isotonic regression.

05 QUANT DESK: MEASURED ON 224 HOURS OF REAL ORDER BOOKS

MODEL	METHOD · REFERENCE	MEASURED, 17–26 NOV 2026 (25,352 SNAPSHOTS)
Regime	Bayesian online change-points · Adams & MacKay 2007	33 confirmed shifts (posterior > 90%, held 20+ min); CUSUM alarms 45
Forecast, 30 min	Quantile regression + CQR + adaptive conformal · Koenker–Bassett 1978, Gibbs–Candès 2021	**79.0%** coverage of the 80% band over 500 forecasts; **+12.1%** pinball skill vs. random walk
Mean reversion	Two-scale Ornstein–Uhlenbeck, exact AR(1) MLE, Dickey–Fuller	Half-life 4 min (fast), 1.8 h (slow); DF t = −12.9 / −5.6, random walk rejected
Delivery	Kalman local-level, innovation-adaptive · Kalman 1960	Needs ≥ 6 hourly credit observations per window; consistency checked by NIS
Rivals & war game	Kaplan–Meier ceilings, discounted Thompson sampling	Learned live on every Clore and Vast bid war
Overfitting gate	CSCV, probability of backtest overfitting · Bailey et al. 2017	Every `--optimize` candidate; refused if PBO > 0.30

Reproduce on your own data: `odin --quant-report odin_book_*.jsonl`.

06 CAPITAL PROTECTION

CONTROL	BEHAVIOUR	SETTING
Watch-only default	New installs read every market and change nothing until switched live, per venue.	`DRY_RUN=1` · `--watch-only` · `odin --market`
One hold, five venues	Watch-only or a tripped kill switch stops every new order, rental, raised bid and re-bid on all venues; safety stops keep running.	`venue_hold()`
Profit guard	Above break-even the order parks at the profitable maximum and pays nothing until filled.	`ODIN_MIN_MARGIN_PCT`
Hard bounds & raises	Absolute ceiling no model can cross; step-limited raises; anti-chase freeze.	`ODIN_HARD_CEILING`
Exposure caps	Order count, speed and size per order; per-venue spend caps.	`ODIN_MAX_ORDERS` · `_MAX_LIMIT_THS`
Daily kill switch	Trips on the cross-market daily loss cap or a flag file; persists across restarts; stops running Clore and Vast rentals.	`ODIN_MAX_DAILY_LOSS_SATS`
Settings sanity	Any number setting that is not a number starts ODIN watch-only on every market; every limit is clamped where it is read.	`odin --doctor`

07 SLEIPNIR: FLEET SUPERVISION

HARDWARE	PROTOCOL	ACTIONS
Antminer, Whatsminer, Avalon, Braiins OS, LuxOS	cgminer-style API, TCP 4028	monitor · alert (never power-cycled)
Bitaxe, NerdQAxe, AxeOS	AxeOS HTTP API	monitor · alert · restart
NVIDIA, AMD, Intel Arc GPUs	SRBMiner / SpinelMiner API + LibreHardwareMonitor / HWiNFO	monitor · alert · restart
CPUs	XMRig API + sensors	monitor · alert · restart

All units are polled concurrently on an async runtime. Alerts cover core, hotspot, VRAM and chip temperatures, hashrate floors, frozen miners and the facility watt budget.

08 OUTBOUND: THE SELL SIDE, SCORED HONESTLY

$$r_{sell} = u \cdot m_{MRR,net} + (1 - u) \cdot r_{pool} \;,\; u = \text{rented} / (\text{rented} + \text{listed})$$

Every minute: what the pool pays after its fee vs. what buyers pay on NiceHash (clearing price, 2% seller fee) and MRR (median rented price, 3% fee), less a 1% switching cost. MRR earns only for the share of time a rig is actually rented, so it is weighted by live utilisation and excluded below 5%. The headline is the share of minutes selling beat the pool, the average edge in sats/PH/s/day and a confidence badge. **OUTBOUND reports; it does not re-point miners.** On 21 Nov our pool beat the best buyer by 1.6%, and OUTBOUND said so.

09 OUTBOUND CONNECTIONS

HOST	PURPOSE	WHEN
api2.nicehash.com	Order book, own orders, wallet; signed order changes	always
mempool.space	Difficulty, retarget progress, block rewards, BTC price	always
pool.kryptex.com	Pool-side hashrate and credits (verified P/L)	if used
hashpower.braiins.com · miningrigrentals.com	Venue books, rentals and bids (keys without withdraw rights)	if that market is on
api.clore.ai · console.vast.ai	GPU spot markets and rentals	if that market is on
api.exchange.coinbase.com · coingecko · safetrade	Coin prices for display and GPU valuation	public data
asgardhashworks.com	Signed releases for `odin --upgrade` and the installer	on demand
ntfy.sh · smtp.gmail.com · api.keygen.sh	Alerts; online licence checks (never with an offline licence)	if configured

10 SECURITY MODEL

KEY CUSTODY

Encrypted at rest. Windows: DPAPI-sealed to the operator's account. Linux: systemd encrypted credentials bound to the TPM2 chip (host key where no TPM exists), decrypted only into the service's private in-memory credential directory. Keys go only to the venue that issued them. Withdrawal permission is never needed and must stay off.

HUGINN: READ-ONLY BY CONSTRUCTION

GET only; no code path reaches the order functions. Binds loopback; published to the tailnet with `tailscale serve`. Random key (≥ 122 bits) checked in constant time; wrong keys answered after 400 ms; ≤ 32 connections, ≤ 12 per address. Every answer: CSP, no-referrer, no framing, nosniff.

AUDIT LEDGER

Each line holds time, kind, message, order state, prev and hash = SHA-256(canonical JSON without hash). `--verify-audit` finds the first edited, deleted or reordered line. Tamper-evident; ship to WORM storage for tamper-proof retention.

LICENSING

Offline Ed25519 `.lic` files, machine- or site-bound, verified locally. Online keys: every answer Ed25519-signed and fingerprint-bound; 7-day grace from the signed date. Lapsed or forged → watch-only.

SIGNED RELEASES & UPGRADE

`SHA256SUMS.txt` is Ed25519-signed by the Asgard release key (kept offline). `odin --upgrade` takes only a version number from `latest.json`, refuses an unsigned list, checks the program's hash, test-runs it, keeps `.previous` and rolls back if the new build doesn't report in.

LINUX SANDBOX

Service account with no login shell; `ProtectSystem=strict`, `ProtectHome`, `NoNewPrivileges`, empty capability set, `MemoryDenyWriteExecute`, AF_INET/AF_UNIX only, writes only to its state folder.

11 HARDENING RECORD

RELEASE	CHANGE	ENFORCED BY
1.8.2	One-click signed Windows setup; licensed plan shown in the console; setup wizard version check.	release pipeline
1.8.1	Summary e-mails capped per hour; profit windows, kill switch and failed stops always sent at once.	`notify::act_now` · tests
1.8.0	Self-hosted releases with Ed25519-signed checksums; upgrader requires the signature. Money as whole satoshis; settings validated and clamped; cycle split into six isolated steps; simulated sessions run the real binary against a stand-in exchange.	`lic::verify_release` · `money.rs` · `settings.rs` · `sim_session`
1.6.4 – 1.6.5	One hold for all venues; cross-market loss cap; kill switch stops running GPU rentals; HUGINN connection limits and headers.	`venue_hold()` · `kill_stops_rentals()`
1.6.0	Hostile-input replay suite; signed licence answers; six RustSec advisories removed (rustls 0.23).	`replay_tests` · `cargo audit`

12 VERIFY IT YOURSELF

```
# 1 · the release is what we signed
sha256sum -c SHA256SUMS.txt      # odin --upgrade also checks SHA256SUMS.txt.sig itself
# 2 · nothing listens except HUGINN on loopback
sudo ss -ltnp | grep -E 'odin|sleipnir'          # expect 127.0.0.1:8787 only
# 3 · keys are root-only / sealed, and the sandbox is applied
sudo stat -c '%U:%G %a %n' /etc/odin-suite/odin.env; systemd-analyze security odin.service
# 4 · the decision history has not been edited
sudo -u odin odin --verify-audit /var/lib/odin-suite/odin/odin_audit.jsonl
# 5 · rebuild the evidence from source
git checkout v1.8.2 && (cd odin && cargo audit && cargo test --release replay_tests)
jq -r '.components[] | "\(.name) \(.version)"' odin-sbom.json | head
```

ODIN Suite is infrastructure software, not financial advice. Trading hashrate can lose money. Not affiliated with NiceHash, Braiins, MiningRigRentals, Clore.ai, Vast.ai, Kryptex, Bitmain, MicroBT, Canaan or the Bitaxe project.