

ZeusTrading

An autonomous Bitcoin trading robot for Indonesian rupiah (BTC/IDR), designed so that a person who is not a trader can trust it, supervise it from a phone, and stop it in one tap. The strategy is deliberately plain. The product is the safety, the explanations and the operations around it.

MY ROLE Founder who leads product and design: the rules, the tests and the operations.	STATUS Running 24/7. Small, capped real-money allocation with a hard loss limit. A simulated twin runs alongside.	BUILT WITH Python (standard library only), Cloudflare Workers + Durable Objects, WebAuthn passkeys, Steam Deck home server	DATE October 2026
--	---	--	-----------------------------

Internal tool, not for sale. ZeusTrading trades only the founder's own account. It is not offered as a product, signal or fund-management service, and nothing here is an invitation to invest.

Alat internal, tidak dijual. ZeusTrading hanya memperdagangkan akun pribadi pendiri. Bukan produk, jasa sinyal, atau jasa pengelolaan dana, dan halaman ini bukan ajakan berinvestasi.

SNAPSHOT

+357% vs +140% Backtest return of the rule vs buying and holding from the first day the rule could act, ~5.2 years, all trading costs and tax included*	-36% vs -74% Worst peak-to-trough fall of the rule vs buy-and-hold. Protecting capital is the stated goal	93 variants From 29 open-source strategies tested. None beat the rule in every period, so the rule stayed	13 trades Completed trades plus the current open position: 5 of 13 profitable, average win +51%, average loss -6.7%, ~76 days held, ~52% of time invested, costs about 5% of profit
---	---	---	---

Past results are not a guarantee of future results. Backtest figures use 0.2% exchange fee per side, 0.05% slippage per side and 0.21% sales tax.

* Buy-and-hold from the very first data day (before the rule could act) would be +80%; we show the stricter comparison.

1 · PROBLEM AND GOAL

Automated crypto trading usually fails for human reasons: too much risk, overfitted strategies, no one watching, and reports only the builder can read. I wanted the opposite. The owner (me) is not a trader, will not stare at charts, and must still be able to answer three questions at any time: *is it running, is my money safe, and can I stop it?*

- ▶ **Goal:** protect capital first; profit second. Avoid large losses rather than chase large gains.
- ▶ **Autonomous:** decides once a day, with no human in the loop, and survives the failure of any single machine.
- ▶ **Safe by construction:** every failure mode defaults to "do nothing" and tells the owner, in plain language.
- ▶ **Explainable:** one rule, in one sentence: buy when the price is more than 1.5% above its 100-day average, sell when it is more than 1.5% below, and do nothing in between.

Honest framing. The strategy is a simple, classic trend-following rule chosen for robustness, not novelty. What I am showing is product judgement: scoping, safety design, operations, and research discipline for an AI-assisted system that touches real money.

2 • PRODUCT DECISIONS

Twelve decisions shaped the product. Each one answers a specific way the system could hurt the owner or mislead them.

01

Plain-language reports

Daily and weekly reports read like a note from a colleague. Jargon such as moving averages, long, flat and drawdown is banned from owner-facing text, and this is written into the rules every change must follow.

02

Owner-only passkey login

The dashboard accepts only the owner's device passkey (WebAuthn). There are no passwords. It adds a one-time recovery code, a lockout after repeated failures, and an alert on every new login.

03

One-tap emergency STOP

The owner can only *reduce* risk from the phone. STOP halts the robot; resuming requires physical access to the server. If the cloud layer dies, the robot keeps its own loss limit.

04

Staged loss alerts

Escalating push alerts when the real-money value falls 10%, 15% and 20%, before a hard stop ends trading at the loss limit. Also alerts for held trades, stale data, a silent robot and real-vs-simulation mismatch.

05

Pre-trade sanity checks

Before any real order: daily data must be fresh, the live price must be near the last close, and the exchange balance must cover the robot's own ledger. A failed check means no trade and a retry next hour. Orders are never auto-repeated, so a double buy cannot happen.

06

Real-vs-simulation parity

A simulated twin takes the same decisions on practice money. If the real robot ever decides differently from the simulation, the owner is alerted. The twin is the system's conscience.

07

Primary / standby baton

A home server is primary and a cloud server is standby. Only the holder of a single "baton" may trade; the standby takes over only after the primary has been silent for three hours. If the baton service is unreachable, nobody trades: better a day late than running twice.

08

Off-site record backup

After each daily decision the baton holder uploads the robot's ledger and journal off-site. A server taking over restores them first. If the records lag the real balance, the real-money robot holds back and warns.

09

Monthly PDF with tax

On the 1st of each month the owner receives a PDF: real and simulated profit or loss, every trade, the 0.21% tax per sale, and the audit result. Built with a small PDF writer that needs no third-party library.

10

Quarterly rule review

Every quarter the current rule is re-tested against its five nearest neighbours, with full costs, over all data, each half, and the last 12 months. The review only advises. Nothing changes automatically.

11

AI analyst, audited

An AI analyst writes market context and risk warnings but never decides or trades, because language models can be confidently wrong with numbers. Thirty days later its warnings are graded against what the price actually did, versus the chance baseline.

12

Capital changes need Face ID

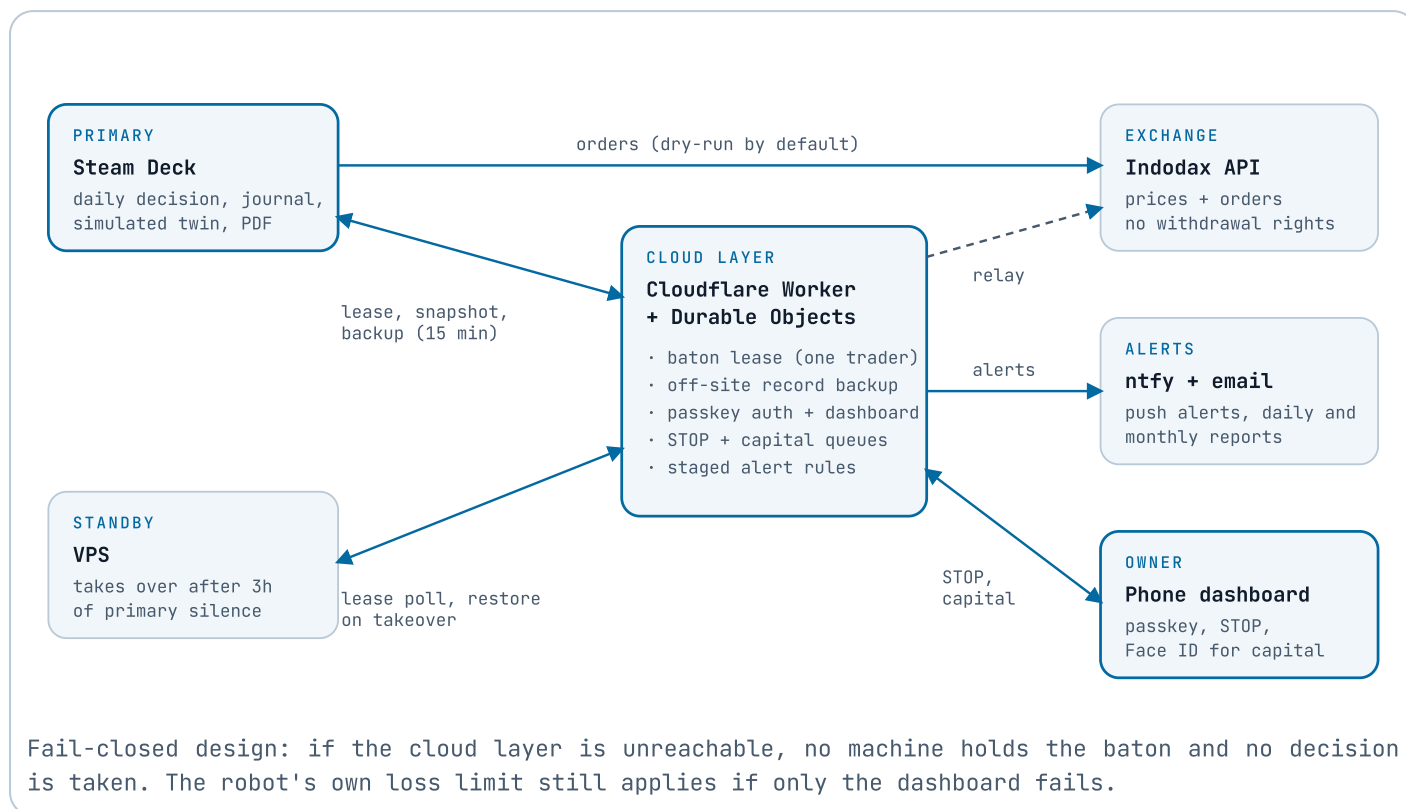
Adding or removing capital from the dashboard needs a fresh passkey confirmation, not just an open session. Requests have per-request limits, a unique id so none is applied twice, and are refused while the robot is stopped.

Least privilege everywhere. The exchange key can view and trade but has no withdrawal permission. The robot trades only from its own ledger and sells only what it bought. Real-money execution sits behind

three separate locks, with dry-run as the default.

3 • ARCHITECTURE

Two machines can run the robot, but a single lease held in the cloud decides which one may act. The cloud layer also hosts the owner dashboard, authentication, the STOP and capital queues, alerting, and the off-site backup.



Daily flow

Check STOP flag → confirm baton → fetch data → decide once after the morning cut-off → sanity checks → trade or hold → write plain-language journal line → email report → push snapshot → back up records off-site. A failed data fetch leaves no "done" marker, so the job retries the next hour.

Separation of powers

Code changes cannot enable real money, add leverage, short, or raise trading frequency without written owner approval. Every strategy change must pass the automated tests and a full-cost backtest.

4 • RESEARCH PROCESS

I treated the strategy as a hypothesis to be falsified. The aim was to find the simplest rule that survives scrutiny, and to resist every tweak that merely looked better on past data.

- ▶ **Full-cost backtests.** Exchange fee on both sides, slippage and sales tax from the start. Decisions at the bar close, trades at the next open, with an automated test for look-ahead bias.
- ▶ **Walk-forward testing.** Settings were chosen on two years of history, then judged only on the following six unseen months, repeatedly. A change had to win here *and* in both halves of the data to qualify.
- ▶ **Randomised stress.** The order of trades was shuffled 1,000 times to estimate a plausible bad-luck worst fall.
- ▶ **Open-source bake-off.** 29 popular open-source strategies (moving-average crosses, MACD, Supertrend, Donchian breakouts, momentum, "buy the dip") across 93 variants, each with full costs. None beat the rule in every period. Some official sample strategies only looked good because they hardly ever sold, and fell as much as buy-and-hold.
- ▶ **Refusing to over-fit.** Six more tuning ideas were tested (blended averages, staged buying, skipping spikes, volatility sizing, faster exits, four-hourly decisions). A blend of three averages looked

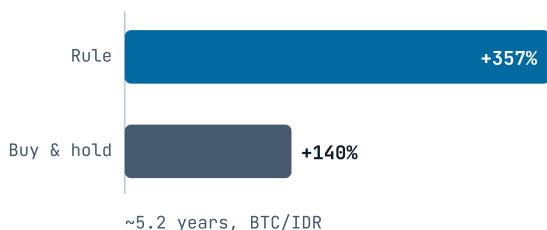
better on old data (+395% vs +358%) but differed from the rule on only 18 of 1,999 days and lost the walk-forward test. The rule stayed.

- **Accuracy is not profit.** The rule is right on only about 43% of trades, yet wins because losses are small and winning trends run long. A more "accurate" variant earned less.

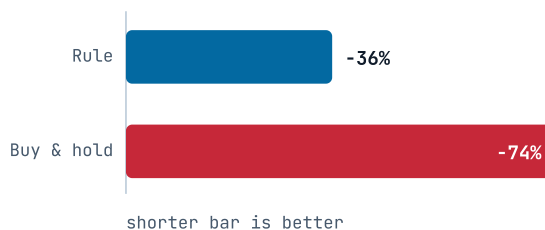
Second asset, same discipline. Applied to Ethereum, the rule beat buy-and-hold (+123% vs +6%) but with a deeper worst fall (-57%) and strong sensitivity to its settings. A 50/50 Bitcoin and Ethereum mix was weaker than Bitcoin alone because the two move together. Ethereum therefore runs on practice money only; real money requires at least six months of simulation, a hard cap, and my written approval.

5 • RESULTS

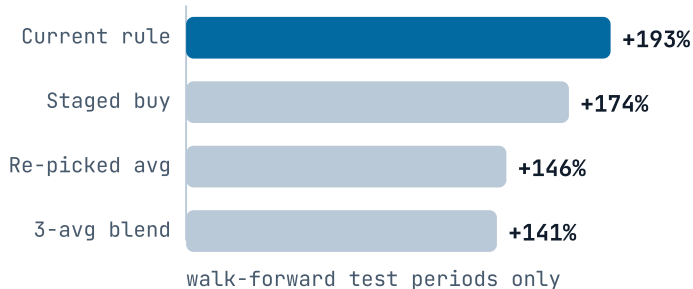
TOTAL RETURN, FULL COSTS



WORST FALL FROM A PEAK



TUNING IDEAS, UNSEEN MONTHS



Source: internal research notes of 19 October 2026. Buy-and-hold is measured from the first day the rule could act; from the very first data day it would be +80%. Charts are drawn from the stated figures; bar lengths are proportional within each chart. The walk-forward figure (+193%) is an upper bound, not a promise: the setting was chosen with knowledge of all data, so it is slightly flattering.

What the results do and do not say. Thirteen completed trades in about 5.2 years is a small sample. The market is one asset on one exchange. There is no stop-loss, so a sharp crash exits at a much lower price, and a fall of this kind could stay below the old peak for years. Past results are not a guarantee. The real evidence is forward: the live record, reviewed every quarter, on a deliberately small and capped allocation. This is a product case study, not financial advice.

6 • WHAT I LEARNED: PRODUCT PRINCIPLES

A

The product is the safety envelope

The signal is a commodity. Trust comes from what happens when data is stale, a server dies, or a number looks wrong.

B

Give the owner asymmetric powers

Stopping is one tap; adding risk needs Face ID and limits. Make the safe action easy and the risky one deliberate.

C

Fail closed, fail loudly

When unsure, do nothing and say so. A day late is acceptable; running twice is not.

D

Distrust your best number

I report the flattering figure as an upper bound and make every change survive unseen data. Most "improvements" were noise.

E

Audit the AI like a colleague

The AI analyst informs but never decides, and is graded against outcomes. Usefulness is measured, not assumed.

F

Decide the rules before the emotions

A pre-commitment document lists what may and may not justify changing the strategy, written when nothing is at stake.

G

Plain language is a spec

If the owner cannot read the report, the system is not done. It is a requirement, tested like any other.

H

Directing the build is a product job

My leverage is precise constraints, acceptance tests, written approval gates and review of evidence.

Next: keep the twelve-month forward test running, review the rule each quarter, and consider Ethereum with real money only after six months of simulation and written approval, at a small cap.

Ilyas A. Almadani · PT Jiji Labs Digital

Founder who leads product and design. Strategy ideas benchmarked from public open-source examples such as Freqtrade; no code or data from them is shared here.

Past performance does not guarantee future results. Cryptocurrency can lose most of its value. This document is a portfolio case study of a personal product, not investment advice. No credentials, balances, addresses or personal data appear here.

© 2026 PT Jiji Labs Digital