📘 The Han YOLO Handbook

Market Making, Explained From Lemonade Stands to Avellaneda–Stoikov

> **How to read this book.** Every section is marked with a belt color:
>
> 🟢 **Rookie** — no finance knowledge needed. Start here.
>
> 🟡 **Pro** — you know what a bid and ask are; now learn the machinery.
>
> 🔴 **Quant** — the college-level math and the research it comes from.
>
> You can stop at any belt and still drive the bot. The config reference in Part II tells you, for every knob: **What it is → Why it exists → When to turn it → How it can hurt you.** Several "Why it exists" stories come from Han YOLO's own shakedown week — we paid real (simulated) money for those lessons, so read them twice.

Part I — The Big Ideas 🟢

1. What is a market maker? (The trading-card shop)

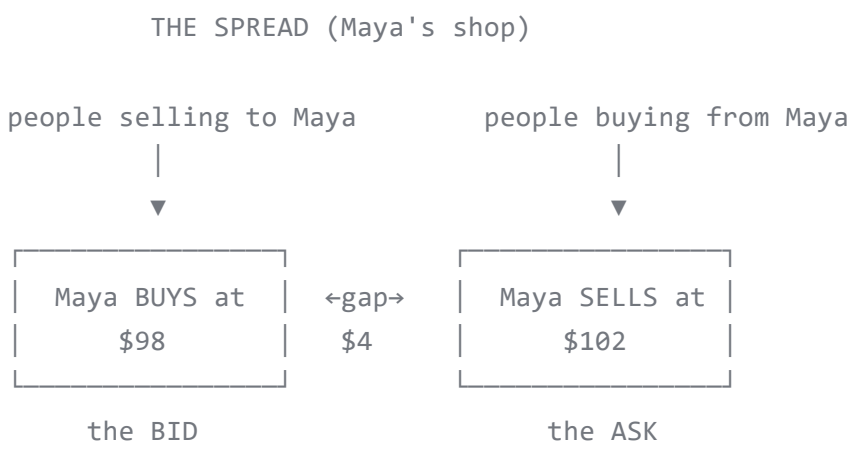
Imagine a kid named Maya who runs a trading-card shop.

A customer walks in wanting to **sell** a LeBron rookie card. Maya offers **$98**. An hour later, a different customer wants to **buy** that same card. Maya charges **$102**.

Maya just earned **$4** without ever guessing whether LeBron cards would go up or down. Her profit came from being *available on both sides* — always ready to buy a little below "fair value" and sell a little above it.

That's a market maker. Han YOLO is Maya, except:

- The "cards" are **volatility perpetual futures** (more on those soon),
- She quotes prices **every second, 24/7**, on three products at once,
- And her $4 is called the **spread**.



Words to know: the price a maker will buy at is the **bid**; the price they'll sell at is the **ask**; the gap is the **spread**; the middle ($100) is the **mid price**.

2. The order book (everyone's offers, stacked up)

A real exchange is like a bulletin board where *everyone's* bids and asks are pinned, best prices in the middle:

```
     ORDER BOOK: QQQ-VOL-PERP
ASKS (sellers)         price    size
                       0.1415   14,157
                       0.1396        2
best ask ──────────►   0.1395       64   ┐
                                         ├── the spread
best bid ──────────►   0.1394   28,119   ┘
                       0.1393        1
BIDS (buyers)          0.0930  47,901,355
```

Han YOLO's whole job is keeping one bid and one ask of its own pinned on this board for each product, all day and all night, priced cleverly enough to earn the spread more often than it gets hurt.

3. Why isn't this free money? The four monsters

Maya's shop sounds easy until you meet the monsters that eat card shops.

Monster #1: Inventory risk 📦

If ten people in a row *sell* LeBron cards to Maya, she now owns ten cards. If LeBron then announces retirement, her whole pile loses value at once. Holding stuff is risk. Market makers want to end up holding **nothing** — buy one, sell one, repeat. The fancy word for "how much stuff am I stuck holding" is **inventory**.

> **Han YOLO war story.** On June 12, 2026 our bot bought into a crashing market until it held **36,882 contracts** — five times more than intended — because its position limit was measured in *contracts* and the crash made each contract cheaper, so the limit silently grew. We got lucky and the market bounced. The fix (`max_position_notional` , measured in **dollars**) is in the config forever. Inventory limits must be in dollars, not pieces.

Monster #2: Adverse selection 🦈 (the card shark)

A man in sunglasses sells Maya twenty LeBron cards at her $98 bid. Why twenty? Because *he knows something she doesn't* — the retirement announcement drops in an hour. People who trade **a lot, suddenly, in one direction** usually know something. Getting picked off by informed traders is called **adverse selection**, and it is the #1 reason market makers lose money. (🔴 The classic academic treatment is Glosten & Milgrom, 1985 — see References.)

> **Han YOLO war story.** We once quoted spreads of just 8 bps (0.08%) in a market that was swinging 50% per hour. Our fill ratio hit **92%** — almost every quote got hit instantly. That sounds great until you realize it means *we were the easiest target in the building*. 562 fills later we were down thousands. Wide spreads are not cowardice; they are the price tag for dangerous conditions.

Monster #3: Volatility 🌪️

Volatility = how violently prices move. When prices are calm, a $4 spread is plenty of cushion. When prices can move $50 between Maya buying and selling, $4 is a joke. Good makers **widen their spread when volatility rises** — like a ferry charging more in a storm.

Real-world case: on **Black Monday (October 19, 1987)**, U.S. stocks fell ~22.6% in one day. The official Brady Commission report (1988) documented how market makers' quoted spreads blew out and some stopped quoting entirely — the storm-pricing instinct, at civilization scale.

Monster #4: Operational risk 🤖💥 (the Knight Capital story)

On **August 1, 2012**, Knight Capital — one of the biggest market makers in U.S. stocks — deployed new software with a bug. In about **45 minutes** the runaway code bought and sold billions of dollars of stock it shouldn't have, losing roughly **$440 million** and nearly destroying the firm (see the SEC's

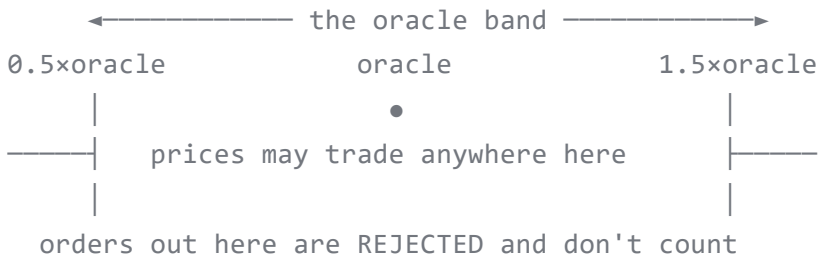
2013 enforcement order, Admin. Proc. File No. 3-15570). The lesson every trading firm tattoos on its arm: **your own software is a monster too.** That's why half of Han YOLO's config file is brakes, fuses, and kill switches.

And the flip side: **Virtu Financial**, a modern electronic maker, disclosed in its 2014 IPO filing (Form S-1) that it had exactly **one losing day out of 1,238 trading days**. Tiny edges, harvested millions of times, with brutal risk discipline — that's the ceiling this craft can reach.

4. Our particular game: vol perps in a padded room 🟢 → 🟡

Han YOLO trades three products: `QQQ-VOL-PERP`, `NVDA-VOL-PERP`, `TSLA-VOL-PERP`. Decoding the name:

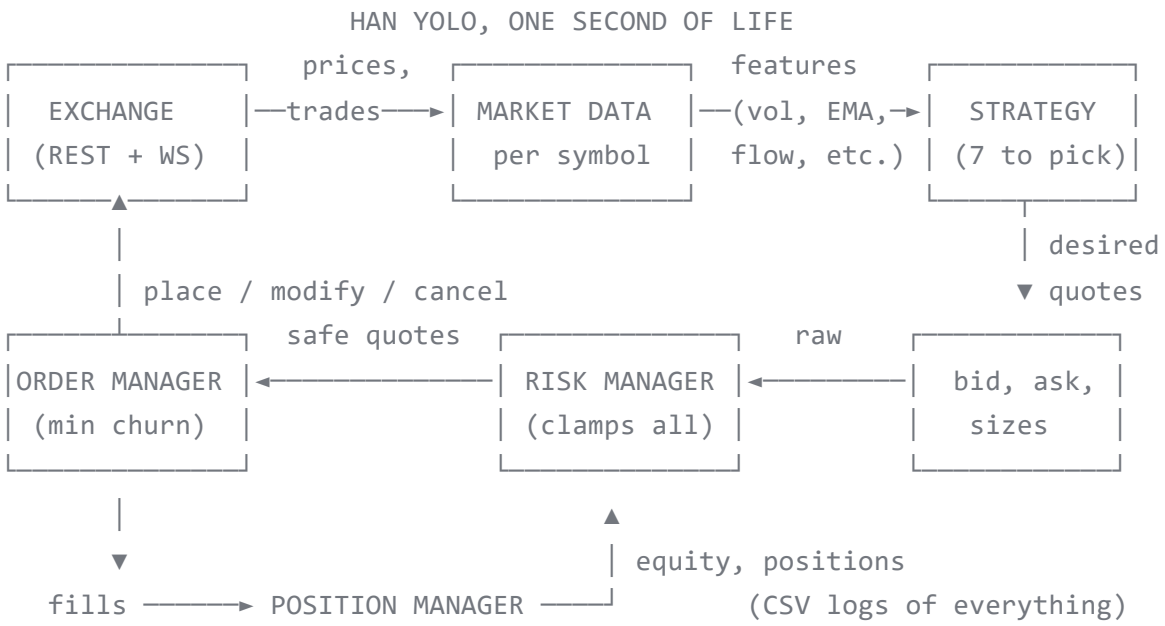
- **VOL**: the price *is* a volatility level. `0.14` means "14% annualized implied volatility of QQQ." We are making markets on *how shaky* a stock is, not on the stock itself. Volatility levels tend to **snap back toward normal** (mean-revert) and **spike upward** in scares.
- **PERP** (perpetual future): a contract with no expiry date. To keep its price tied to reality, the exchange uses **funding**: every hour, whichever side is "winning the tug-of-war" pays rent to the other side. If the perp trades *above* the official reference price, longs pay shorts; *below*, shorts pay longs. Think of funding as **rent paid by the crowded side of the boat**.
- **The oracle and its band** 🫐: the exchange publishes an official fair price (the **oracle**) and only allows orders within **±50%** of it. Like bowling with bumpers — prices can careen around, but only inside the lane:



- **The contest twist**: to qualify as a market maker, Han YOLO must show **two-sided in-band quotes on all three products ≥98% of the time**, sampled every 10 seconds, 24/7. Think of it as a perfect-attendance trophy — one week of slacking disqualifies all your good grades.
- **Fees (measured, not rumored)**: every fill where we *provided* the resting quote pays us a **rebate of 1.75 bps** of the trade's dollar value; if we *take* someone else's quote we pay **3.5 bps**. We verified this on 172 of our own fills: exact to the cent. The bot therefore only ever posts **ALO (post-only)** orders — it is structurally incapable of paying taker fees by accident.

bps? A basis point is 1/100th of a percent. 100 bps = 1%. Spread people speak in bps the way carpenters speak in millimeters.

5. The machine at a glance 🟡



Every ~1 second: read the market → compute features → strategy proposes quotes → risk clamps them to safe → order manager makes the book match, using as few API calls as possible. Fills come back, inventory and PnL update, repeat ~86,400 times a day.

Part II — Every Knob in `config.yaml` 🟡

> Format: **What** · **Why** · **When to change** · ⚠️ **Danger**. The file hot-reloads every `reload_secs` seconds — most changes apply to a *running* bot within ~5s, no restart. (Exceptions: `mode` and `symbols` require a restart; the bot tells you if you try.)

Top level

`bot_name` — What: the name shown in logs and the plain dashboard line. Why: morale. When: never, obviously — it's HAN YOLO. (The big ASCII art lives in `banner.txt`, hot-swappable by just editing that file.)

`mode` — What: `paper` (built-in simulator), `live` (real exchange), or `replay` (re-run recorded data via `replay.py`). Why: you must be able to test without risking anything. When: `paper` for experiments, `live` for the contest. ⚠️ Requires restart; always glance at the dashboard header — `[paper|…]` vs `[live|…]` — before trusting any number. We once stared at paper prices thinking they were real.

`role` — What: a label (`market_maker` / `trader`) describing which contest seed this account has. Why: documentation for humans; the exchange decides your real category by how you authenticated. When: leave it.

`symbols` — What: the exact market names to quote. Why: must match the exchange's `/info/meta` precisely (`QQQ-VOL-PERP`, not `QQQ-VOL`). When: only if the exchange adds markets. ⚠️ Restart required; a typo means the bot quotes nothing and your uptime dies quietly — startup prints a warning and the real list if you get one wrong.

`strategy` — What: which of the seven brains drives quoting (see Part III for each). Why: different regimes reward different brains. When: freely, live — it hot-swaps. Current calibrated default: `hybrid`.

`reload_secs` — What: how often the bot re-reads this file. Why: lets you tune a *running* bot. When: leave at 5. ⚠️ Setting it huge means your edits "don't work" (they're just waiting).

`exchange:` — talking to the venue

`base_url` / `ws_url` — What: the REST and WebSocket addresses. Why: where the exchange lives. When: never, unless they migrate. The WebSocket is opportunistic; REST polling is the reliable backbone and backs off automatically while WS is streaming.

`auth.session_token` — What: a ready-made login token (from a browser session). Why: quick manual override. When: rarely — tokens expire in about an hour (we measured this the hard way, overnight). Prefer the key file.

`auth.bls_secret_key_file` — What: path to your bot's BLS private key (made by `make_bot_key.py`). Why: with the key, the bot can log itself back in forever — sessions expire hourly and the bot silently re-authenticates (`~ session refreshed` in the log). **This file IS your contest account**; the exchange permanently pins your identity to its public key. When: set once. ⚠️ Lose the file → lose the account, balance, and leaderboard spot. Back it up. (If left blank, the bot also tries the default location `~/.permuto/bot_key.hex` as a safety net.)

`auth.bls_secret_key_hex` — What: the same key pasted inline. Why: convenience. When: avoid — secrets don't belong in config files that get shared. `report.py` redacts it, but habits beat redaction.

`auth.signing_mode` — What: `raw` (sign the 32 challenge bytes directly) or `chip0002` (Chia's standard message-signing wrapper). Why: the exchange accepts both from bots. When: leave `raw` — it's what we've battle-tested.

`auth.signing_role` — What: `master` for standalone bot keys, `wallet_outer` for keys derived inside the Sage wallet. Why: tells the exchange how to derive your trading identity from your public key. When: `master` unless you deliberately re-link a Sage-derived key — changing it changes *which account you are*.

`auth.challenge_route` — What: which login endpoint to use. Why: the exchange deprecated the old two-step flow. When: leave on `/exchange/wallet_link_challenge` .

`leverage` — What: borrowed-money multiplier, set per market at startup. Why: **the single most expensive lesson of shakedown week.** At 10x, the exchange backs each position with a sliver of margin, putting the forced- liquidation price only ~7.5% away — in a market that moves 50% per hour, that meant our positions were repeatedly force-closed at terrible prices plus a 0.5% liquidation fee, bleeding ~$5,900 before we understood why. At **1x**, your entire balance backs each position and liquidation is effectively unreachable. When: **never raise this.** Spread capture needs almost no leverage; leverage here only buys you a faster funeral.

`rate_limit_per_sec` — What: a self-imposed cap on API calls. Why: the exchange rate-limits each user (we've seen `rate_limited: true` and 429s); exceeding limits delays *your own* orders. When: lower it if you see rate-limit errors in `console.log` ; raising it only helps if the venue publishes a higher allowance.

`user_agent` / `waf_cookie` — What: browser-disguise headers. Why: a Cloudflare firewall sits in front of the API and blocks "robot-looking" software outright (we got 403s on *everything* until we sent a browser User-Agent). When: touch only if every request suddenly 403s; then copy a fresh cookie/UA from your browser per the comments.

`poll_l2_secs` / `poll_trades_secs` / `poll_fills_secs` — What: how often to fetch the order book, the public trade tape, and *your own* fills. Why: polling is the dependable data path; fills polling is also how the bot learns it got filled (typical detection delay = this number). When: tighten `poll_fills_secs` (e.g., 1.0) if inventory reactions feel slow; loosen all three if you're brushing rate limits. ⚠️ Three symbols share the budget: `poll_l2_secs: 0.5` alone costs ~6 requests/second when the WebSocket is down.

`reconcile_secs` — What: how often the bot asks the exchange "what do *you* think my positions, balance, and open orders are?" and adopts the answer. Why: trust but verify — this is how the bot catches hourly funding charges, fills it missed, and the exchange's **stale-order sweeper** (a "dead-man's switch" that cancels resting orders it considers too old; without reconciliation we'd believe phantom quotes existed and bleed uptime invisibly). When: leave at 20s. It also self-heals the books after any crash.

`recv_timeout_secs` / `reconnect_backoff_secs` — What: how long to wait on a stuck request, and the retry schedule after disconnects (1s, 2s, 5s, 10s, 30s, then repeat 30s). Why: 98% uptime is won by reconnecting fast but not hammering a struggling server. When: leave them.

`quoting:` — the shape of our presence

`starting_cash` — What: the bankroll the bot's internal accounting starts from (500k MM seed). Why: PnL math needs a baseline. When: only if the contest seed changes. (The exchange's number always wins at reconcile.)

`quote_notional` — What: **dollars** per quote per side per symbol; the bot converts to contracts at the current price. Why: prices here range 0.03–0.36, so "5 contracts" is meaningless — dollars are the honest unit. This is your main **throttle**. When: this is the knob you'll turn most. Small (200–500) while learning a regime; larger (1000–3000) when fills are profitable. ⚠️ Bigger quotes fill more *and* lose more when conditions turn — we tripled it once to reduce inventory faster and tripled the bleed instead.

`quote_size` — What: contract-count fallback used only if `quote_notional` is unset. When: ignore it.

`min_spread_bps` — What: the floor — the bot will never quote tighter than this around the mid, no matter what a strategy asks. Why: a seatbelt against a strategy gone over-confident. When: raise toward 40–60 in scary regimes as a hard backstop.

`max_spread_bps` — What: the ceiling on quote width. Why: two jobs — keep quotes inside the oracle band (out-of-band quotes don't count for uptime), and bound how silly "defensive" quoting gets. When: in this venue's chaos regime, the *calibrated* recommendation is **2000** so volatility- widening has room to work; the band itself (±50% ≈ 5000 bps each side) is the true outer wall and the risk layer separately clamps quotes inside ~90% of it using the live oracle. ⚠️ Too small = you stand close to the bonfire in a storm; too large does nothing harmful as long as it's under the band.

`requote_tolerance_bps` — What: how far the *desired* price must drift from a resting order before the bot bothers to move it (using the exchange's native `modify`, which keeps the same order ID). Why: every move costs an API call against rate limits, and resting orders that don't move keep their place in line (**queue priority** — earlier orders at a price fill first). When: raise (30–40) if you see rate-limit errors; lower (10–15) only in calm markets where precision matters more than churn. ⚠️ At 5 bps in a violent market we fired thousands of modifies and got throttled by the venue.

`refresh_secs` — What: the heartbeat — how often the whole think-and-quote loop runs. Why: 1s is plenty for this venue's speed. When: leave it; faster mostly burns rate limit.

`stale_book_secs` — What: if a symbol's order book hasn't updated in this long, treat it as unreliable. Why: quoting from old prices is how you donate. When: leave at 5. Note: when a book goes *fully*

empty (it happens here at night), the bot doesn't go dark — it anchors wide quotes on the **oracle** instead, keeping uptime alive (v0.9.6 feature, born from watching TSLA's book literally vanish).

`risk:` — the brakes, fuses, and kill switches

`max_position_notional` — What: the most **dollars** of one symbol the bot may hold, long or short. Crucially it binds on the position's *current value*, not just at entry — learned from the flash-crash story in Part I, where a contract-count limit silently quintupled. Why: the inventory monster. When: scale with confidence; 2% of bankroll per symbol (10k) is the contest opening stance. The bot enforces it gracefully: over the cap, it keeps quoting **only the side that shrinks the position**.

`max_total_notional` — What: the same idea across all symbols combined. Why: three "small" positions are one big bet on the same chaos. When: ~5% of bankroll (25k) to start.

`max_order_notional` / `max_position` / `max_order_size` — What: per- order dollar cap, plus contract-count fallbacks used only when the notional keys are absent. When: keep `max_order_notional` ≈ 2–3× `quote_notional` ; ignore the fallbacks.

`max_drawdown_pct` — What: if equity falls this % below its best-ever point (the *high-water mark*), trip the circuit breaker. Why: Knight Capital. When: 10–12% is sane.

`daily_loss_limit` — What: dollars lost since the day began that trip the breaker; resets at midnight UTC. Why: bounds any single bad day — exactly the fuse that finally stopped the June-12 bleed. When: contest recommendation 10,000 (2% of seed). ⚠️ Once tripped, it re-trips all day by design; the bot stays parked in defense till midnight. That's a feature.

`circuit_breaker_secs` — What: how long one breaker trip lasts. Why: cooling-off period. What "tripped" looks like: the bot does **not** go dark (that would murder the uptime trophy) — it quotes **1-lot at maximum width**: present, compliant, and nearly unhittable. Like a shop that stays open but puts everything behind glass. Also trips on an error burst (≥10 errors in 60s).

`flatten_on_breaker` — What: if true, a breaker trip also market-sells all inventory immediately. Why: nuclear option. When: **leave false.** Market-dumping into this venue's thin books is how you turn a paper loss into a crater — we measured books with *nine contracts* of total depth.

`cancel_orders_on_exit` — What: on shutdown, cancel resting quotes (true) or leave them on the book (false). Why: the exchange holds GTC orders server-side, so quotes left resting **keep earning uptime while the bot is down** — restarts become free. When: `true` normally (clean state); `false` **during the judged uptime window** so upgrades cost nothing.

`cancel_all_on_error` — What: after an order error, sweep that symbol's orders to reach a known state. When: leave true.

`fat_finger_pct` — What: reject any quote more than this % from the last mid before it leaves the building. Why: the final tripwire against a strategy bug emitting nonsense. When: leave at 10.

Strategy sections (the seven brains) 🟡

Pick one with `strategy:` ; each has its own block. All of them *always quote both sides* — opinions are expressed by shifting, widening, and resizing, never by going dark (the uptime trophy again).

`symmetric` — equal spread both sides of the mid, no opinion. `half_spread_bps` : distance from mid to each quote. Use: calm markets, baselines, and calibration runs. ⚠️ It's the brain that got farmed at 8 bps — it has no defenses; its safety *is* its width.

`inventory_skew` — symmetric, plus: the more you hold, the more you shift both quotes against your pile (long → both quotes lower: harder to buy more, easier to sell out). `max_skew_bps` : shift size at full inventory — the knob is *normalized*, i.e., it's the shift **at the position cap**, so it scales correctly whether the cap is 50 contracts or 50,000 (a normalization we added after the un-normalized version produced absurd numbers at scale). `size_taper` : also shrink the dangerous side's size as inventory grows.

`vol_adaptive` — measures recent realized volatility and widens with it: `half = base_half_spread_bps + vol_mult × σ` . `vol_window_secs` : lookback. `ewma_lambda` : memory of the fast volatility estimate (0.97 ≈ remembers a few minutes; closer to 1 = calmer, slower). The storm-ferry brain. Use: whenever you can't predict the regime — it prices the regime for you.

`trend` — leans inventory *with* the trend when fast EMA > slow EMA, momentum exceeds `momentum_threshold_bps` , and RSI isn't already extreme. `ema_fast` / `ema_slow` : averaging lengths in ticks; `rsi_period` : overbought/ oversold check; `target_inventory` : how hard to lean. Use: cautiously — vol products mean-revert, which is trend's natural predator.

`flow_imbalance` — watches the public tape: if 80% of recent volume is buyers, shift quotes up before the price moves. `window_secs` : how much tape; `imbalance_shift_bps` : max shift at total one-sidedness; `microprice_weight` : how much to trust the **microprice** (a mid that leans toward the heavier side of the book — see Part III). Use: active two-way markets with real flow.

`avellaneda` — the famous optimal-market-making model (Avellaneda & Stoikov, 2008). Computes a personal "fair price" that drifts away from your inventory (the **reservation price**) and a mathematically optimal spread. `gamma` : risk aversion (bigger = more scared = wider and more skewed); `k` : how fast fill probability dies off with distance (calibrate from your own `fills.csv` !); `tau` : planning horizon in seconds; `vol_window_secs` : where σ comes from. Full math in Part III. Use: once `k` is fitted from contest-regime data — garbage `k` in, garbage spread out.

`hybrid` ⭐ (calibrated default) — a weighted committee:

- `w_trend` × trend lean + `w_flow` × tape lean + `w_carry` × **funding/ premium lean** (if the perp trades under its oracle, shorts are paying longs rent *and* the price has gravity upward — lean long; flips symmetrically), all capped by `max_lean_bps` ;
- minus an inventory pull of up to `max_inv_skew_bps` at full position;
- spread = `base_half_spread_bps` (90, the measured calm-regime floor) + `w_vol_widen` × volatility widening;
- bigger size on whichever side reduces inventory. Use: the all-weather default. The carry term mattered here: we've watched funding swing from −0.16%/hr to +0.53%/hr across two days — rent worth collecting in both directions.

`logging:` / `dashboard:`

`logging.dir` , **`tick_csv`** , **`fills_csv`** , **`orders_csv`** , **`pnl_csv`** , **`flush_secs`** — What: the folder for the four CSV logs, each file's name (`ticks.csv` = market snapshots, `fills.csv` = every trade we got, `orders.csv` = every order action, `pnl.csv` = equity over time), and how many seconds between flushes to disk. Why: these files are the raw material for every calibration; **treat `logs/` as treasure** (we lost a day of it once to a careless upgrade — the README's upgrade ritual exists because of that). The bot also writes `logs/console.log` (timestamped events) by itself. When: leave everything.

`dashboard.enabled` / **`refresh_secs`** — What: the live terminal display and its frame rate. The dashboard auto-detects pipes and drops to one-line mode when not on a real terminal. When: leave on; 2s refresh.

`paper:` — the flight simulator

Used only in `mode: paper` . The simulated market follows an **Ornstein–Uhlenbeck process** (a price on a rubber band — it wanders but gets pulled back toward an anchor; the standard model for mean-reverting things like volatility — 🔴 see Part III), plus rare violent jumps, because that's what we measured on the real venue.

- **`seed`** — random-number seed; same seed = identical run (great for A/B-testing one parameter; change the seed to test robustness instead).
- **`init_mid`** — starting prices; set to real June-12 levels.
- **`ou_theta`** — rubber-band strength (pull toward the anchor per second).
- **`ou_sigma_bps`** — calm-regime jiggle per second (calibrated: 3 bps).
- **`jump_prob`** / **`jump_bps`** — chance per tick of a violent jump, and its size (calibrated to a venue where ~45% of *hours* contained >1000 bps ranges).

- `fill_A` / `fill_k` — the fill model: resting orders fill at Poisson rate `λ(δ) = A·e^(−k·δ)` — fast near the mid, exponentially rarer farther away (🔴 Part III). These are the same shapes Avellaneda's `k` describes; refit both from real `fills.csv` during the contest.
- `taker_fee_bps: 3.5` / `maker_fee_bps: -1.75` — the **verified** real fee schedule (negative = rebate).
- `latency_ms` — simulated round-trip delay range; ours measures ~105 ms to the real venue.

Part III — Leveling Up 🔴

This is the collegiate floor. Nothing here is needed to operate the bot; all of it is needed to outgrow it.

3.1 The indicators, precisely

EMA (exponential moving average). With smoothing α = 2/(N+1): `EMA_t = α·price_t + (1−α)·EMA_{t−1}`. Recent prices matter more; `N` sets the memory. Trend strategy compares EMA(10) vs EMA(50).

RSI (relative strength index), period N. Average the last N upward moves (G) and downward moves (L): `RSI = 100 − 100/(1 + G/L)`. Near 100 = everything's been going up (stretched); near 0 = the opposite. Trend refuses to chase when RSI is already past 75/25.

EWMA volatility. Per tick return r (in bps): `σ²_t = λ·σ²_{t−1} + (1−λ)·r²`, volatility = $\sqrt{\sigma^2}$. λ = `ewma_lambda`. This is the RiskMetrics estimator (J.P. Morgan, 1996); λ=0.97 at ~0.5s ticks gives a few minutes of memory.

Microprice. Instead of the plain mid, weight by *opposite* depth: `micro = (bid·askSize + ask·bidSize)/(bidSize+askSize)`. If the bid is piled high and the ask is thin, the microprice leans up — the book itself is telling you where price wants to go next. (Stoikov, 2018 formalizes this as the best simple short-horizon price predictor.)

Flow imbalance. Over a window: `(buyVolume − sellVolume)/(total)` ∈ [−1, 1], where sides are the *aggressor's*. A reading of +0.8 means buyers are steamrolling — shift up or get run over.

3.2 The Avellaneda–Stoikov model, gently then exactly

Gently: Imagine your fair price isn't the market's mid — it's the mid *minus a nudge for every card already in your pile*. Long inventory? Your personal fair price is lower, so your quotes sit lower: you sell more eagerly, buy more reluctantly, and your pile drains. Then quote a spread around that personal price that's wider when (a) you're more risk-averse, (b) the market's more volatile, (c) fills don't decay too fast with distance.

Exactly (as implemented in `strategy.py`): with mid `s`, inventory `q`, risk aversion `γ`, per-second volatility `σ` (price units), horizon `τ` seconds, and fill-decay `k`:

```
reservation price   r = s − q·γ·σ²·τ
half-spread         δ = γ·σ²·τ/2 + (1/γ)·ln(1 + γ/k)
quotes              bid = r − δ,   ask = r + δ
```

The model assumes fills arrive as a Poisson process whose intensity decays exponentially with distance from the mid, `λ(δ) = A·e^(−kδ)` — exactly the paper simulator's fill engine, which is why fitting `A, k` from real fills calibrates both at once. (Fit recipe: bucket your maker fills by distance- from-mid at fill time, divide by time-at-risk per bucket, fit a line to log-intensity vs distance: slope = −k, intercept = ln A.)

Source: M. Avellaneda & S. Stoikov, *High-frequency trading in a limit order book*, **Quantitative Finance 8(3)**, 2008 — itself a modern treatment of Ho & Stoll's 1981 inventory model.

3.3 The processes

Ornstein–Uhlenbeck: `dX = θ(μ − X)dt + σ dW` — drift back toward μ at speed θ, plus noise. The canonical model for mean-reverting series (volatility, interest rates). Our paper sim adds jump events on top, matching the measured venue: median calm-hour movement near zero, but ~45% of hours containing >10× the calm range.

Funding (perpetuals): hourly payment ∝ premium = (perp − oracle)/oracle, paid by the side holding the perp away from the oracle. Mechanism introduced by BitMEX's XBTUSD (2016); here sampled continuously and settled hourly. Carry strategies earn it; the hybrid's `w_carry` leans toward the receiving side, which is *also* the mean-reversion direction — two edges, one lean.

3.4 The scoreboard math

Sharpe ratio = mean(period returns)/std(period returns) — reward per unit of wobble (Sharpe, 1966). The dashboard computes it on 1-minute equity buckets, unannualized: direction matters, the absolute number doesn't.

Max drawdown = the worst peak-to-valley fall of equity, as % of the peak. The risk manager's breaker watches the live version of this.

Adverse selection (advsel) = average move *against* you in the 5 seconds after each fill, in bps. Positive = you're being picked off (widen!). Negative = your fills tend to be at extremes that snap back (the good kind). This single number is the fastest health check of quote placement.

Fill ratio = fills/orders submitted. Near 100% = you're everyone's favorite snack (too tight). Near 0% = you're invisible (too wide). There is no universally right number, but in this venue, 92% preceded our worst day and ~30–50% accompanied our best behavior.

Part IV — Recipes

You observe…	Turn this	Direction
Fill ratio >80%, advsel positive	`base_half_spread_bps` / strategy width	wider
No fills for hours in a calm market	width	tighter (gently)
`rate_limited` / 429 in console.log	`requote_tolerance_bps` ↑, `rate_limit_per_sec` ↓	calmer
Inventory pinned at the cap	`max_skew_bps` / `max_inv_skew_bps` ↑, `quote_notional` ↓	drain it
Circuit breaker tripping on noise	`daily_loss_limit` ↑ *only if* the losses were tolerable	carefully
Violent regime ignites	nothing — `vol_adaptive` / `hybrid` widen automatically; verify `max_spread_bps` ≥ 2000	trust it
Upgrading during judged uptime	`cancel_orders_on_exit`	`false`
Equity ≠ exchange balance	nothing — wait one `reconcile_secs`; persistent gaps are a bug report	observe

The one-sentence philosophy: *get paid for being there, charge properly for danger, never hold a pile, and let the fuses blow before you do.*

Glossary

ALO / post-only — an order that may only *rest* on the book (maker); rejected if it would instantly trade. **bps** — 1/100 of 1%. **Funding** — hourly rent between perp longs and shorts keyed to the premium. **Inventory** — what you're stuck holding. **Liquidation** — the exchange force-closing an under-margined position (0.5% fee here). **Maker/Taker** — provided the resting quote / hit someone else's. **Mid** — (bid+ask)/2. **Notional** — price × size, in dollars. **Oracle** — the exchange's official reference price; quotes must sit within ±50% of it. **Queue priority** — first order at a price fills first. **Reservation price** — your inventory-adjusted private fair value. **Slippage** — the price you moved by trading. **Spread** — ask − bid. **Uptime** — fraction of 10-second checks with live two-sided in-band quotes on all three symbols (≥98% required).

References

1. Avellaneda, M. & Stoikov, S. (2008). *High-frequency trading in a limit order book.* Quantitative Finance, 8(3), 217–224.
2. Ho, T. & Stoll, H. (1981). *Optimal dealer pricing under transactions and return uncertainty.* Journal of Financial Economics, 9(1), 47–73.
3. Glosten, L. & Milgrom, P. (1985). *Bid, ask and transaction prices in a specialist market with heterogeneously informed traders.* Journal of Financial Economics, 14(1), 71–100.
4. U.S. SEC (2013). *In the Matter of Knight Capital Americas LLC*, Admin. Proc. File No. 3-15570 — the $440M software failure of Aug 1, 2012.
5. Virtu Financial, Form S-1 (2014) — one losing trading day in 1,238.
6. Brady Commission (1988). *Report of the Presidential Task Force on Market Mechanisms* — market-maker behavior in the October 1987 crash.
7. J.P. Morgan/Reuters (1996). *RiskMetrics — Technical Document* (EWMA volatility).
8. Stoikov, S. (2018). *The micro-price: a high-frequency estimator of future prices.* Quantitative Finance, 18(12).
9. O'Hara, M. (1995). *Market Microstructure Theory.* Blackwell — the textbook for Monsters #1–#3.

10. Han YOLO shakedown logs, June 11–13, 2026 — `CALIBRATION.md`, fee verification (172 fills), regime measurements, and all war stories above.

Built during the Permuto Volatility Cup, June 2026. Trade carefully; the monsters are real, even when the dollars aren't. 🟢🟡🔴