



CLAUDE CODE + INTERACTIVE BROKERS / PAPER TRADING

BUILD THE TREND JOIN LONG BOT

**AN AUTOMATED IBKR PAPER-
TRADING BOT, BUILT PROMPT BY
PROMPT WITH CLAUDE CODE.**

14 copy-paste prompts. Every bug we hit and how we fixed it. Simulated screenshots of every stage.

01

SCAN
S&P 500 gappers

02

FILTER
six rules, 5m bars

03

ENTER
limit + broker stop

04

MANAGE
1/3 at 0.75R, BE,
trail

05

CLOSE
by 15:51 ET

Edition 1 / October 2026

Paper trading only. Not financial advice.

[START HERE](#)

WHAT YOU WILL BUILD

A Python bot that trades one strategy, Trend Join Long, on an Interactive Brokers paper account, by itself, every weekday. You build it by pasting 14 prompts into Claude Code. This guide gives you every prompt, the reasons behind every decision, what you should see at each step, and every problem we hit along the way with its fix.

THE BOT WILL

- Scan the S&P 500 for 3%+ gappers every morning
- Check six filters on every completed 5-minute bar
- Buy with a marketable limit and a broker-held stop
- Sell a third at +0.75R, move to breakeven at +1R, trail the rest
- Close everything by 15:51 ET
- Text you on Telegram and keep a live dashboard

YOU WILL NEED

- An IBKR account with paper trading enabled
- Live market data shared to paper (about \$500 equity in the live account)
- Trader Workstation (TWS)
- Windows 11 (a VM on a Mac is fine), Python 3.12+
- Claude Code, and about a day
- Telegram for alerts (optional but recommended)

PAPER TRADING ONLY. NOT FINANCIAL ADVICE.

This guide documents a software build. The bot is designed to refuse live accounts, and its paper results are not evidence that the strategy makes money. Trading involves risk of loss. You are responsible for anything you run.

[READ THIS FIRST / CONTINUED](#)

● ● ● HOW TO USE THIS GUIDE



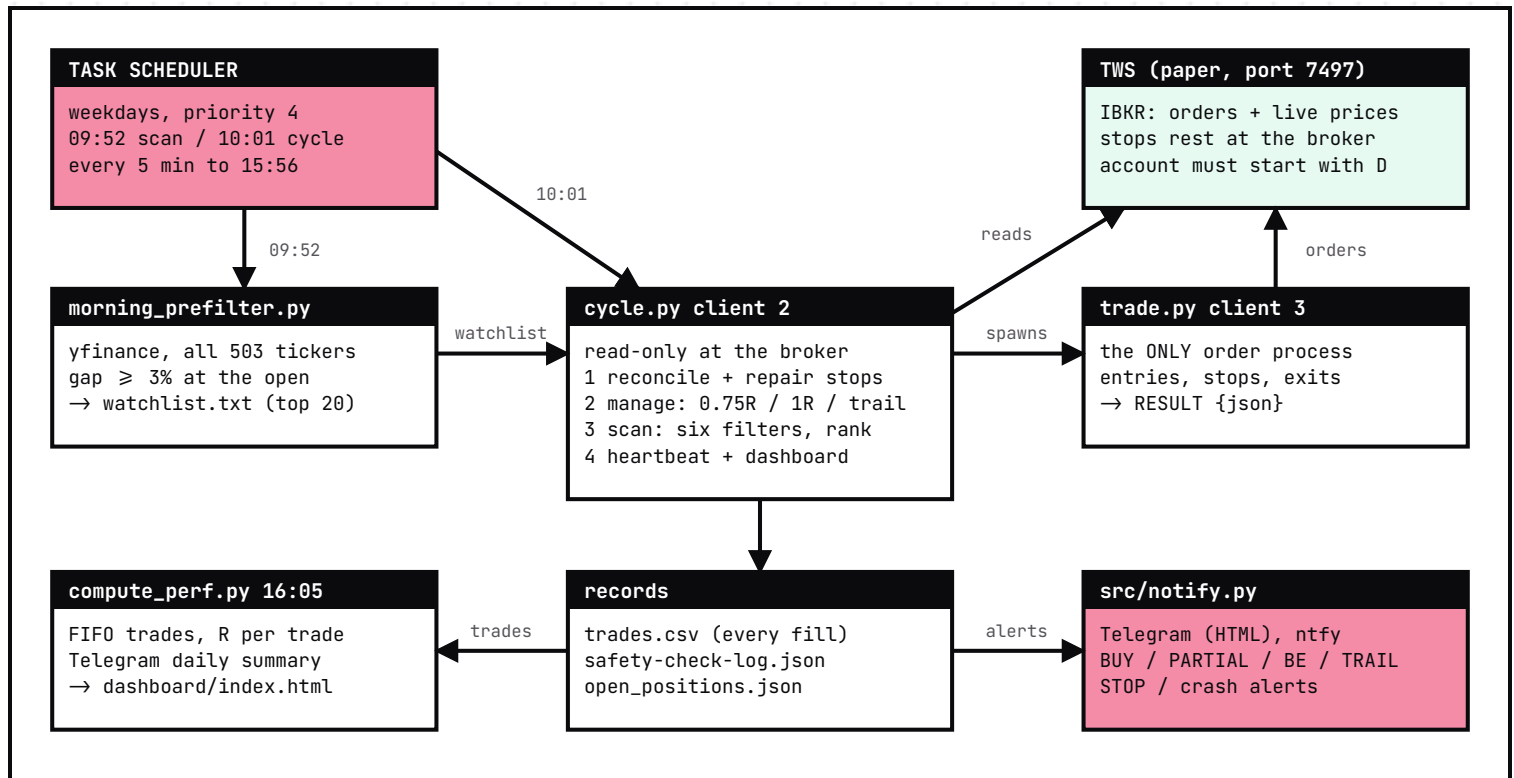
1. Do Step 00 first. Most problems people hit are account and market-data settings, not code.
2. Paste one prompt at a time into Claude Code, opened in your project folder. Read what Claude says before moving on.
3. Run the check at the end of each step. The screenshots show what success looks like.
4. Something wrong? The troubleshooting section lists every error we hit, by code and message.

Credit: the step structure follows the Humbled Trader tutorial "AI trading bot with Claude and IBKR", which this build started from. The prompts here are Workflow Factory's own and include the bug fixes and safety changes we made along the way.

THE BIG PICTURE

HOW THE BOT WORKS

Small scripts with one job each. Windows runs them on a schedule; Interactive Brokers holds the orders.



The decision-maker (cycle.py, client ID 2) never places orders. Every order goes through trade.py on client ID 3, because IBKR only lets the client that placed an order change or cancel it.



HOW IT WORKS / CONTINUED

● ● ● ONE 5-MINUTE CYCLE



1 Time gate	Weekend, too early, closed, manage-only, force-close or ok. Off-hours runs exit in under a second.
2 Reconcile	Compare with IBKR: detect stop-outs by order ID, adopt orphans, make sure every position has a stop for exactly its shares.
3 Manage	+0.75R: shrink the stop, then sell a third. +1R: stop to entry. Then trail under 5-minute swing lows.
4 Scan	Today's watchlist, six filters on completed bars, rank by relative volume, live safety gate, size, buy with a stop.
5 Record	Decision log, heartbeat, dashboard. Every step writes something you can read later.

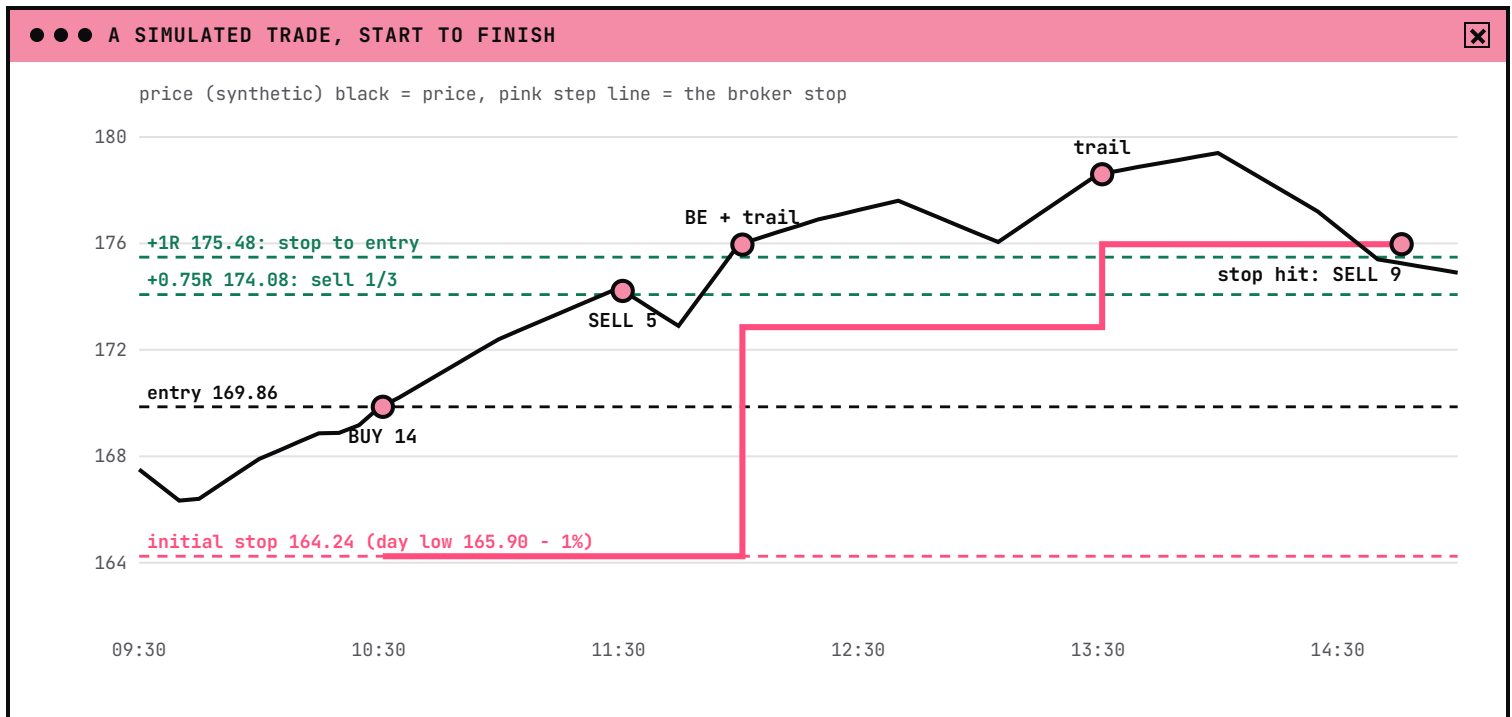
TREND JOIN LONG

ONE STRATEGY, WRITTEN DOWN

Long only, S&P 500, 5-minute bars. The bot buys strength that is already proven and joins the trend, rather than guessing a bottom.

ID	FILTER	EXACTLY AS THE BOT CHECKS IT
D1	Above yesterday's high	Last completed 5m close > prior day's high
D2	With the long-term trend	Prior close > 200-day simple moving average
D3	Gapped up	Today's official open at least 3% above the prior close
I1	Above the premarket high	Last close > highest premarket price (04:00 to 09:30)
I2	New high of the day	A completed 5m bar closes above the high made before it
I3	Heavy volume	Volume so far today at least 2x the 14-day average at the same time of day

THE STRATEGY / CONTINUED



Synthetic prices from this guide's simulator, run through the real bot code. Entry 169.86, stop 164.24, so $R = 5.62$. The bot sold 5 of 14 shares at $+0.75R$, moved the stop to entry at $+1R$, trailed it up twice, and was stopped out at 175.97 ($+0.98R$ on the whole trade).

● ● ● RISK

1% of \$25,000 at risk per trade, max 10% per position, max 5 open, max 5 new entries a day, 3% daily loss stop.

● ● ● TIME

Entries 10:05 to 15:30 ET. Manage only 10:00 to 10:05 and after 15:30. Everything closed at 15:51 (earlier on half-days).

● ● ● ENTRY

Limit at ask + 0.2%, cancelled after 30 s. Ranked by relative volume. One entry per stock per day.

CONTENTS

THE BUILD, STEP BY STEP

00	Before you start	p. 09
01	Place one paper order	p. 11
02	Close the position safely	p. 13
03	Write the strategy down	p. 15
04	Configuration and secrets	p. 18
05	The core bot files	p. 20
06	The S&P 500 universe	p. 24
07	The morning gap scanner	p. 25
08	The 5-minute trading cycle	p. 28
09	Phone alerts	p. 34
10	Daily summary and log rotation	p. 37
11	Put it on autopilot	p. 39
12	The performance dashboard	p. 41
13	Test, then go live on paper	p. 43
>>	Troubleshooting: every error and its fix	p. 45
>>	Running it every day	p. 51
>>	FAQ	p. 53



BEFORE YOU START

Get every account, program and setting in place so the first prompt works first time.

● ● ● WHY IT'S BUILT THIS WAY



- The bot only ever talks to an IBKR PAPER account through Trader Workstation (TWS) on port 7497. No real money is involved, and two separate guards in the code refuse anything else.
- Market data is the step most people get stuck on. Paper accounts borrow the live account's data subscriptions, and IBKR only enables streaming API data once the live account holds about US\$500.
- Python lives in a project virtual environment (.venv) so the scheduled tasks always use the same packages you tested with.

● ● ● CHECKLIST



Interactive Brokers account	Open an IBKR account and enable its paper-trading account (IDs start with DU).
TWS installed	Install Trader Workstation. Log in with the PAPER username. Keep it running while the bot works.
TWS API settings	File > Global Configuration > API > Settings: tick Enable ActiveX and Socket Clients, port 7497, untick Read-Only API, add 127.0.0.1 to Trusted IPs.
TWS auto restart	Lock and Exit: choose Auto restart (e.g. 11:45 PM), not Auto log off. IBKR still asks for a manual login about once a week.
Market data	In the LIVE account's Client Portal: subscribe (Non-Professional) to US Securities Snapshot and Futures Value Bundle plus US Equity and Options Add-On Streaming Bundle, then share market data with the paper account. Needs roughly US\$500 equity in the live account.
Python	Python 3.12 or newer (the reference build used 3.14.7 on Windows 11).
Claude Code	Install Claude Code and open it in your project folder, e.g. C:\Projects\trading-bot.
Telegram (step 10)	A Telegram account for phone alerts. You will create a bot with @BotFather later.
A machine that stays awake	Windows 11 (a VM on a Mac works). Plugged in, sleep disabled during market hours.

STEP 00 / CONTINUED

● ● ● CLAUDE CODE PROMPT / STEP 00 / PASTE AS ONE MESSAGE



Set up a Python project for an Interactive Brokers PAPER-trading bot in the current directory.

PREREQ CHECK:

1. Print ``python --version`` (or ``py -3 --version`` on Windows). It must be 3.12 or newer. If not, STOP and tell me.
2. Confirm the current directory is empty apart from hidden files. If not, STOP and list what is here.

DO:

1. Create a virtual environment in `.venv` and install `ib_async` into it (`pip install ib_async`).
2. Create `test_connect.py` that connects to `127.0.0.1:7497` with `clientId=10` and a 10-second timeout, prints the managed accounts, prints "PAPER OK" if every account ID starts with "D" (paper accounts are DU...), otherwise prints "NOT A PAPER ACCOUNT" and exits 1, then disconnects in a finally block.
3. Run `test_connect.py` with the venv's Python and show me the output.

If the connection is refused, tell me to check that TWS is running, logged in to PAPER, and that File > Global Configuration > API > Settings has socket clients enabled on port 7497.

From now on, always run Python through `.venv` (on Windows: `.venv\Scripts\python.exe`).

CHECK IT

`.venv\Scripts\python.exe test_connect.py` prints your DU... account and PAPER OK.

● ● ● TEST_CONNECT.PY

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe test_connect.py
Accounts: ['DU1234567']
PAPER OK
```

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data.

● ● ● TWS CLOSED

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe bot.py --symbol AMD --check-only
[10:40:00 ET] ERROR: IBKR connection failed: ConnectionRefusedError: [Errno 111] Connect call failed ('127.0.0.1', 7497)
```

● ● ● IF IT GOES WRONG



Connection refused: TWS is closed, logged in to live, or the API is not enabled.

Running plain `python` instead of the venv: packages appear to be missing.

STEP 01 / BUY_ONE.PY

PLACE ONE PAPER ORDER

Prove that code can place an order on your paper account: buy 1 share.

● ● ● WHY IT'S BUILT THIS WAY



- Every order sets time-in-force DAY explicitly. Without it TWS applies its own preset, and the first real close attempt in our build was cancelled with error 10349.
- An account guard refuses to continue unless the account ID starts with D (paper).
- The script waits for a final status (Filled / Cancelled), not the transient PendingSubmit.

● ● ● CLAUDE CODE PROMPT / STEP 01 / PASTE AS ONE MESSAGE



Place a paper-trading BUY order for 1 share of MU on Interactive Brokers.

PREREQ CHECK:

1. Confirm `ib_async` imports in the venv. If not, STOP and tell me to install it.
2. Confirm TWS answers on 127.0.0.1:7497 (quick connect with `clientId=10`). If not, STOP.

CREATE `buy_one.py`:

- Connect to 127.0.0.1:7497 with `clientId=10`.
- Account guard: if any managed account does not start with "D", print the accounts and exit 1. Never trade a live account.
- Qualify Stock("MU", "SMART", "USD").
- MarketOrder BUY 1 share. Set `order.tif = "DAY"` explicitly (without it TWS applies its preset TIF and can cancel the order with error 10349). Set `order.account` to the paper account.
- Wait up to 10 seconds for a terminal status (Filled, Cancelled, ApiCancelled, Inactive). PendingSubmit and PreSubmitted are not final.
- Print order ID, fill price (or "pending"), final status. If the order died, print `trade.log` entries (status, message, errorCode).
- Disconnect in a finally block. On a connection error print it and exit nonzero.

RUN `buy_one.py` with the venv's Python. Print its output and: "ORDERED: 1 share MU. Check TWS positions."
Do NOT close the position. That is the next step.

CHECK IT

TWS shows a 1-share MU position on the DU account. Outside market hours the order may sit in PendingSubmit until the open; that is normal.



STEP 01 / CONTINUED

● ● ● IF IT GOES WRONG



Error 10349 (order cancelled, TIF preset): add `order.tif = "DAY"`.

Order stuck in PendingSubmit outside market hours: it fills at the next open, or cancel it in TWS.



STEP 02 / CLOSE_ONE.PY

CLOSE THE POSITION SAFELY

Flatten one symbol cleanly. This script becomes your manual emergency exit for any position.

● ● ● WHY IT'S BUILT THIS WAY

- Later the bot leaves a protective stop order resting at the broker. If you sell without cancelling that stop first, the stop can trigger later on a position you no longer own and open a short. So `close_one.py` cancels working SELL orders first.
- IBKR lets only the client ID that placed an order modify or cancel it, so each order is cancelled from a connection using its owner's client ID.
- `--symbol` is required so you can never close the wrong thing by accident, and short positions are refused.

● ● ● CLAUDE CODE PROMPT / STEP 02 / PASTE AS ONE MESSAGE

Create a safe manual close script for the paper account.

PREREQ CHECK:

1. Confirm `buy_one.py` exists. 2. Confirm TWS answers on 127.0.0.1:7497.

CREATE `close_one.py` (usage: `python close_one.py --symbol MU`; `--symbol` is REQUIRED):

- Connect with `clientId=11`. Same paper-account guard as `buy_one.py`.
- Find the position for `--symbol` via `ib.positions()`. If none, print "NO POSITION TO CLOSE" and exit 0. If the position is short (negative), refuse and exit 1. This script only closes longs.
- FIRST cancel every working SELL order for that symbol (stops included), from `ib.reqAllOpenOrders()`. IBKR only lets the placing client cancel an order, so cancel each one from a short-lived connection that uses that order's `clientId` (`order.clientId`), then confirm none remain working. If any remain, STOP without selling.
- Then MarketOrder SELL the full position with `tif="DAY"` and the account set. Wait up to 10 s for a terminal status.
- Print sold quantity, fill price, status. Append the SELL to `trades.csv` if that file exists (columns: `timestamp_iso` (UTC), `symbol`, `side`, `size`, `fill_price`, `order_id`, `status`, `stop_price`).
- Disconnect cleanly in `finally` blocks.

Also update `buy_one.py` if needed so both scripts set `tif="DAY"` on every order.

RUN: `python close_one.py --symbol MU`. Print the output and "CLOSED: position flat. Verify in TWS."

CHECK IT

TWS shows no MU position and no working orders.

**STEP 02 / CONTINUED**

● ● ● CLOSE_ONE.PY AFTER A TEST BUY

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe close_one.py --symbol AMD
Cancelling working STP SELL 14 (orderId 2, clientId 3)
Order ID:      3
Sold qty:      14 of 14
Fill price:    171.07
Status:        Filled
```

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data.

● ● ● IF IT GOES WRONG



"Error 202 ... Order Canceled" while closing is the confirmation that a stop was cancelled, not a failure.



WRITE THE STRATEGY DOWN

Turn Trend Join Long into one JSON file the bot reads every cycle.

It is the only place strategy and risk numbers live.

● ● ● WHY IT'S BUILT THIS WAY



- Trend Join Long, in plain words: buy S&P 500 stocks that gapped up at least 3%, are above yesterday's high, trending above their 200-day average, above the premarket high, breaking to a new high of the day on heavy volume.
- The tutorial's JSON left real ambiguities. We pinned each one down: relative volume compares volume so far with the same time of day (a full-day comparison almost never passes at 10:05); "above today's high" means a completed 5-minute bar CLOSES above the high made before it (a price can't be above its own high); the gap is measured from the official open.
- Safety additions: 3% daily loss stop, max 5 new entries a day, one entry per symbol per day, a stop order resting at the broker, a marketable-limit entry instead of a market order, and an account guard.

STEP 03 / CONTINUED

● ● ● CLAUDE CODE PROMPT / STEP 03 (1/2) / PASTE AS ONE MESSAGE



Create the strategy file for the paper-trading bot. It is the ONLY place strategy and risk parameters live.

PREREQ CHECK: Confirm rules.json does not exist yet. If it does, STOP.

CREATE rules.json with exactly this content, then validate it parses as JSON:

```
{
  "schema_version": 1,
  "strategy_name": "Trend Join Long",
  "direction": "long_only",
  "trade_timeframe": "5m",
  "timezone": "America/New_York",
  "account_guard": {"mode": "paper", "required_account_prefix": "D", "port": 7497},
  "universe_filters": {"index": "S&P 500", "min_price_usd": 3.0, "exclude_if_halted": true},
  "daily_filters": {
    "D1_above_prior_day_high": true, "D1_price_source": "last_trade_at_evaluation",
    "D2_prior_close_above_sma200": true, "D2_sma_length_days": 200,
    "D3_min_gap_pct_from_prior_close": 3.0, "D3_gap_direction": "up_only",
    "D3_gap_measured_at": "regular_session_open", "D3_max_gap_pct_from_prior_close": null
  },
  "intraday_filters": {
    "I1_above_premarket_high": true, "I1_premarket_window_et": ["04:00", "09:30"],
    "I2_above_today_hod": true, "I2_trigger": "5m_bar_close_above_prior_hod",
    "I3_rvol_min": 2.0, "I3_rvol_lookback_days": 14,
    "I3_rvol_method": "cumulative_volume_vs_same_time_of_day_avg"
  },
  "time_filter": {"earliest_entry_et": "10:05", "latest_entry_et": "15:30", "force_close_et": "15:51"},
  "entry": {
    "order_type": "marketable_limit", "limit_offset_pct_above_ask": 0.2, "unfilled_cancel_after_sec": 30,
    "tif": "DAY", "max_entries_per_symbol_per_day": 1, "candidate_ranking": "rvol_desc"
  },
  "exit": {
    "initial_stop_rule": "lod_minus_1pct", "lod_reference": "regular_session_low_at_entry",
    "R_definition": "entry_price_minus_initial_stop",
    "partial_profit_trigger_R": 0.75, "partial_profit_fraction": 0.3333, "breakeven_trigger_R": 1.0,
    "post_breakeven_trail": "swing_low_5m_2_2", "trail_pivot_leftBars": 2, "trail_pivot_rightBars": 2,
    "trail_only_ratchets_up": true, "stop_order_resides_at_broker": true, "tif": "DAY"
  },
  "risk": {
    "max_risk_per_trade_pct": 1.0, "max_position_size_pct_of_portfolio": 10, "max_concurrent_positions": 5,
    "max_daily_loss_pct": 3.0, "max_new_entries_per_day": 5,
    "sizing_rule": "min(risk_based_shares, position_cap_shares)", "skip_if_sized_shares_below": 1
  }
}
```

● ● ● CLAUDE CODE PROMPT / STEP 03 (2/2) / PASTE AS ONE MESSAGE



Print the file back. Do not create other files.



STEP 03 / CONTINUED

CHECK IT

```
.venv\Scripts\python.exe -c "import json; json.load(open('rules.json')); print('valid JSON')"
```

● ● ● IF IT GOES WRONG



Editing numbers in .env instead of rules.json. From here on, rules.json is the single source of truth.



STEP 04 / .ENV, .GITIGNORE, REQUIREMENTS

CONFIGURATION AND SECRETS

Keep connection settings and secrets apart from strategy settings.

●●● WHY IT'S BUILT THIS WAY

- The tutorial's .env repeated three risk limits that also live in rules.json, and they had already drifted (5 trades a day in one file, 8 in the other). We removed them from .env. One source of truth means no silent conflicts.
- PORTFOLIO_VALUE_USD sizes trades as if the account were \$25,000, even though IBKR paper accounts start near \$1M. That keeps the paper test realistic for a small account.
- Client ID 2 is the read-only brain; client ID 3 is the only process that places or changes orders.

●●● CLAUDE CODE PROMPT / STEP 04 / PASTE AS ONE MESSAGE

Create the configuration files.

PREREQ CHECK: rules.json exists; .env, .gitignore and requirements.txt do not. Otherwise STOP.

CREATE:

1. .env (connection, account size and secrets ONLY; strategy and risk limits live in rules.json)
Environment and secrets only. Strategy and risk parameters live in rules.json.
IBKR_HOST=127.0.0.1
IBKR_PORT=7497
IBKR_CLIENT_ID=2
IBKR_EXEC_CLIENT_ID=3
PAPER_TRADING=true
PORTFOLIO_VALUE_USD=25000
TELEGRAM_BOT_TOKEN=
TELEGRAM_CHAT_ID=
2. .env.example (same keys, empty secret values, safe to share)
3. .gitignore: .env .venv/ __pycache__/ *.pyc trades.csv logs/ open_positions.json
safety-check-log.json watchlist.txt dashboard/
4. requirements.txt (unpinned): ib_async python-dotenv pandas numpy yfinance requests

Install requirements.txt into .venv, then write requirements.lock with `pip freeze` so the exact tested versions can be reinstalled later. Print the file tree. Never print the contents of .env after secrets are added.



STEP 04 / CONTINUED

CHECK IT

```
.venv\Scripts\python.exe -c "import ib_async, dotenv, pandas, yfinance, requests; print('deps ok')"
```

● ● ● IF IT GOES WRONG



Never paste your Telegram token into a chat or commit .env anywhere.

STEP 05 / IBKR_CLIENT, STRATEGY, TRADE, BOT

THE CORE BOT FILES

Build the four files that talk to IBKR, check safety, size trades and place orders.

● ● ● WHY IT'S BUILT THIS WAY



- Two processes, two client IDs: the decision-maker (client 2) never places orders; trade.py (client 3) does every order action. IBKR only lets the client that placed an order change or cancel it, so one owner keeps every stop manageable.
- Entries use a marketable limit (ask + 0.2%, cancelled after 30 s) and attach a protective stop at the broker for whatever filled. If the stop is rejected, trade.py sells immediately. A position is never left unprotected.
- Size = the smaller of risk-based shares (1% of \$25k / distance to the stop) and a 10% position cap.
- The daily-loss gate fails closed: if today's P&L can't be read, no new trade. IBKR's P&L stream can stay silent after a fresh TWS login, so the bot falls back to its own ledger plus the portfolio's unrealized P&L.
- Dates are compared in Eastern time. trades.csv stores UTC, and a plain string compare counted trades after 8 PM ET toward the wrong day.

STEP 05 / CONTINUED

● ● ● CLAUDE CODE PROMPT / STEP 05 (1/2) / PASTE AS ONE MESSAGE



Create the core files of the IBKR paper-trading bot. Read rules.json and .env first; strategy and risk numbers come ONLY from rules.json. Use ib_async (not ib_insync), pathlib, zoneinfo, and anchor every path to the project folder (Task Scheduler starts scripts in C:\Windows\System32).

PREREQ CHECK: rules.json, .env, requirements.txt exist; Python $\geq$ 3.12; ib_async, dotenv, pandas import; src/ibkr_client.py, strategy.py, trade.py, bot.py do not exist.

CREATE:

1. src/__init__.py (empty)
2. src/ibkr_client.py: class IBKRClient(host, port, client_id, required_account_prefix="D", timeout=10)
 - connect; ACCOUNT GUARD: if any managed account doesn't start with the prefix, disconnect and raise.
 - place_order(symbol, side, qty, limit_price=None, settle_timeout=10): qualify Stock(symbol,"SMART","USD"); MarketOrder if no limit, else LimitOrder (outsideRth=True); ALWAYS order.tif="DAY" and order.account set; wait for a terminal state (Filled, Cancelled, ApiCancelled, Inactive), not PendingSubmit.
 - place_stop(symbol, qty, stop_price): SELL StopOrder, tif DAY, wait until TWS accepts or rejects it.
 - cancel_if_working(trade), disconnect().
3. strategy.py: evaluate(symbol, ib, portfolio_value_usd=None) $\rightarrow$ dict, safety gates only (no D1-I3 filters yet):
 - a) already holding the symbol, OR a BUY order for it working from ANY client (reqAllOpenOrders) $\rightarrow$ fail
 - b) weekday and inside earliest/latest entry time (ET) from rules.json
 - c) IBKR session check from contract details liquidHours (catches holidays and 1 PM half-days)
 - d) max concurrent positions from rules.json
 - e) daily loss gate: today's P&L from ib.reqPnL (poll up to ~6 s); if silent, fall back to realized P&L from trades.csv + unrealized from ib.portfolio(); if neither is trustworthy, FAIL CLOSED with "daily PnL unavailable; cannot verify loss limit"; fail if $P\&L \leq -\text{max_daily_loss_pct}$ of portfolio.
 - f) live price: reqMarketDataType(3) then reqMktData; wait up to 10 s for the first tick (first ticks after a TWS login can take ~5 s); fall back to today's 5-minute bars; return price, ask, day_low, price_source, data_delayed (True when marketDataType is 3/4 or the last bar is stale).

Cast numpy values to Python types; append every result to logs/safety_log.jsonl.
4. trade.py: the ONLY process that places or changes orders, on IBKR_EXEC_CLIENT_ID. Last stdout line is always `RESULT {json}`. Port guard: PAPER_TRADING=true with port 7496/4001 (or the reverse) aborts.

Modes:

 - order: --symbol --side BUY|SELL --size N [--limit P] [--stop P]. A BUY with --limit waits unfilled_cancel_after_sec, cancels any unfilled remainder, keeps a partial fill. A BUY with --stop places a broker stop for exactly the FILLED quantity; if that stop is rejected, market-sell the fill immediately.
 - --modify-stop PERMID [--stop P] [--stop-qty N]: modify in place (same orderId), REFUSE to lower the price.
 - --place-stop --size N --stop P.
 - --flatten: cancel this client's working SELLS for the symbol; REFUSE if another client's SELL is still working; then market-sell the whole position.

Append every fill to trades.csv: timestamp_iso (UTC), symbol, side, size, fill_price, order_id, status, stop_price (the initial stop on BUY rows). Retry once if the file is locked. Migrate an older header automatically.
5. bot.py: single-symbol tool. --symbol (required), --check-only, --allow-delayed.

Load .env, port guard, paper-mode check from rules.json, daily entry cap from rules.json (count today's BUY rows comparing ET dates, not UTC strings), connect on IBKR_CLIENT_ID, strategy.evaluate, disconnect.

Refuse to trade on delayed data unless --allow-delayed. Size = $\min(\text{floor}(\text{portfolio} \times \text{risk} \% / (\text{price} - \text{stop}))$,

STEP 05 / CONTINUED

● ● ● CLAUDE CODE PROMPT / STEP 05 (2/2) / PASTE AS ONE MESSAGE



```
floor(portfolio*cap%/price)) with stop = day_low * (1 - 1%). Limit = ask * 1.002.  
Spawn trade.py BUY with --limit and --stop; subprocess timeout = fill wait + 30 s (a 30 s timeout would kill a 30 s  
fill wait). Report from the ledger. Prefix prints with [HH:MM:SS ET].
```

AFTER: print the file tree and "READY: run .venv\Scripts\python.exe bot.py --symbol NVDA --check-only". Do not run it.

CHECK IT

During market hours: `bot.py --symbol NVDA --check-only` shows 'pass': True, price_source 'ticks', data_delayed False.

● ● ● CHECK-ONLY, MARKET OPEN

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe bot.py --symbol AMD --check-only  
[10:41:00 ET] evaluate: {'ts_et': '2026-10-05T10:41:00-04:00', 'symbol': 'AMD', 'pass': True, 'reasons': ['time gate ok', 'no  
existing position', 'session open', 'portfolio limits ok'], 'price': 170.73, 'ask': 170.74, 'day_low': 165.9, 'price_source':  
'ticks', 'data_delayed': False, 'daily_pnl': 0.0, 'pnl_source': 'ibkr_pnl'}  
[10:41:00 ET] check-only: done
```

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data.

● ● ● CHECK-ONLY, AFTER 15:30

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe bot.py --symbol AMD --check-only  
[16:20:00 ET] evaluate: {'ts_et': '2026-10-05T16:20:00-04:00', 'symbol': 'AMD', 'pass': False, 'reasons': ['outside entry  
window 10:05-15:30'], 'price': 0.0}  
[16:20:00 ET] check-only: done
```

● ● ● PORT GUARD: PAPER FLAG WITH LIVE PORT 7496

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe bot.py --symbol AMD --check-only  
ABORT: paper flag but live port
```

● ● ● ACCOUNT GUARD: TWS LOGGED IN TO A LIVE ACCOUNT

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe bot.py --symbol AMD --check-only  
[10:40:00 ET] ERROR: IBKR connection failed: RuntimeError: account guard: expected 'D*' accounts, got ['U7654321']
```



STEP 05 / CONTINUED

● ● ● IF IT GOES WRONG



'no usable market price': no live data subscription yet, or the live account is logged in elsewhere (Error 162).

'daily PnL unavailable': open TWS fully and retry; the fallback covers the usual cause.



STEP 06 / SRC/SP500_TICKERS.PY

THE S&P 500 UNIVERSE

A hard-coded list of the 503 current S&P 500 tickers in IBKR format.

● ● ● WHY IT'S BUILT THIS WAY



- IBKR writes class shares with a space (BRK B); Yahoo uses a hyphen (BRK-B). The list stays in IBKR format and the scanner converts. Fetch the list from Wikipedia rather than from the model's memory, which goes stale.
- Refresh it every few months when the index changes.

● ● ● CLAUDE CODE PROMPT / STEP 06 / PASTE AS ONE MESSAGE



Create src/sp500_tickers.py with one constant SP500_TICKERS: every current S&P 500 ticker (about 503), sorted, hard-coded (no fetching at import). Use IBKR format with a space for class shares ("BRK B", "BF B").

Get the list from the current Wikipedia "List of S&P 500 companies" page (fetch it; do not rely on memory). Add a comment with the source and date. Print the count and the first and last 5 entries. Create no other files.

CHECK IT

```
.venv\Scripts\python.exe -c "from src.sp500_tickers import SP500_TICKERS as t; print(len(t))"
```

prints about 503.

● ● ● IF IT GOES WRONG



Lists written from memory miss recent index changes.

STEP 07 / MORNING_PREFILTER.PY

THE MORNING GAP SCANNER

Every 30 minutes until 12:52 ET, find the S&P 500 stocks that opened 3%+ above yesterday's close and write the top 20 to watchlist.txt.

● ● ● WHY IT'S BUILT THIS WAY

- yfinance does the scanning (free, no 400-ticker limit); IBKR only does execution.
- Gap = today's OPEN versus yesterday's close, as rules.json defines it. The tutorial measured the live price instead: a stock that gapped +3.63% but was down by the afternoon would have been missed, and one that drifted up late would be wrongly included.
- Before the open, on weekends and holidays there is no bar for today, so the scanner refuses to write a stale list rather than comparing yesterday with the day before.
- Thresholds looser than rules.json (for testing) are only allowed with --dry-run, so a test can never feed 1% gappers to the live bot. threads=5 keeps Windows from running out of file handles.

● ● ● CLAUDE CODE PROMPT / STEP 07 / PASTE AS ONE MESSAGE

Create morning_prefilter.py, the gap scanner. Do not modify other files.

PREREQ CHECK: src/sp500_tickers.py exists; yfinance imports; morning_prefilter.py does not exist.

BEHAVIOUR:

- Args: --min-gap (default 3.0), --min-price (default 3.0), --dry-run (don't write watchlist.txt). If either threshold is LOOSER than the defaults and --dry-run is not set, refuse with exit 2: test thresholds must never overwrite the bot's real watchlist.
- Paths anchored to the script's folder. Convert "BRK B" → "BRK-B" for Yahoo.
- yf.download(tickers, period="2d", interval="1d", group_by="ticker", threads=5, progress=False, auto_adjust=True). threads=5, not more: higher concurrency exhausts file handles under Task Scheduler.
- Per ticker: only use it if the last bar is dated TODAY (ET). gap_pct = (today's OPEN - prior close) / prior close. Keep if gap ≥ min-gap and price ≥ min-price. Count below_gap, below_price, failed (bad data, continue), no_today_bar.
- If at least half the tickers have no bar for today (pre-open, weekend, holiday): do NOT write a watchlist; print an error JSON and exit 1.
- Sort by gap, keep top 20. Write watchlist.txt atomically (temp file + os.replace) with a header line "# Auto-generated by morning_prefilter.py at YYYY-MM-DD HH:MM ZONE" (the bot checks this date), then lines like "AAPL # gap +3.45% open \$185.20 prev \$180.00".
- Print a JSON summary: success, as_of_date_et, total_screened, survivors_count, below_gap, below_price, failed, failed_tickers, no_today_bar, elapsed_seconds, top_20_survivors, watchlist_path.
- Tripwires to stderr: failed ≥ 95% → "ALERT: Yahoo-wide failure suspected"; ≥ 30% → "ALERT: yfinance degradation".

AFTER: print "READY: run morning_prefilter.py --dry-run". Do not run it.

STEP 07 / CONTINUED

CHECK IT

After 09:30 ET on a weekday: `morning_prefilter.py --dry-run` prints success true and today's date.

SCANNER TEST RUN

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe morning_prefilter.py --dry-run
{
  "success": true,
  "as_of_date_et": "2026-10-05",
  "total_screened": 503,
  "survivors_count": 3,
  "survivors_before_cap": 3,
  "below_gap": 500,
  "below_price": 0,
  "above_max_gap": 0,
  "failed": 0,
  "failed_tickers": [],
  "no_today_bar": 0,
  "elapsed_seconds": 1.67,
  "top_20_survivors": [
    "HOOD (+5.20%)",
    "AMD (+4.69%)",
    "CCL (+3.40%)"
  ],
  "watchlist_path": null
}
```

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data.

TOO EARLY: NO BAR FOR TODAY YET

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe morning_prefilter.py --dry-run
{
  "success": false,
  "error": "no daily bar for 2026-10-05 yet (503/503); run after the 09:30 ET open on a trading day",
  "elapsed_seconds": 1.32
}
```

LOOSE TEST THRESHOLD WITHOUT --DRY-RUN

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe morning_prefilter.py --min-gap 1.0
{
  "success": false,
  "error": "--min-gap 1 / --min-price 3 are looser than rules.json (3% / $3); add --dry-run so the bot's watchlist.txt is not overwritten with test survivors"
}
```



STEP 07 / CONTINUED

● ● ● IF IT GOES WRONG



Run before the open and it correctly refuses: there is no daily bar for today yet.

On a quiet day zero stocks gap 3%; test with `--min-gap 1.0 --dry-run`.



THE 5-MINUTE TRADING CYCLE

The brain. Every 5 minutes: reconcile with the broker, protect and manage open trades, then look for new entries.

● ● ● WHY IT'S BUILT THIS WAY



- The broker is the source of truth. Each cycle first compares its own state with IBKR's positions and orders, so a crash, a manual trade or a stop fill can never leave the bot confused.
- Stop-outs are matched by the stop order's permanent ID, never by share count. Matching by quantity caused false stop-outs after partial sells in an earlier version.
- Partial profit first, then breakeven: at $+0.75R$ sell a third, at $+1R$ move the stop to entry, then trail under 5-minute swing lows. Before selling the third, the stop is SHRUNK to the shares left, so a full-size stop can never sell shares you no longer hold.
- Stops are modified in place (same order ID), never cancelled and re-placed, so the position is never without a stop.
- Only completed 5-minute bars count. The live, half-finished bar can't trigger a signal.
- Off-hours runs exit in well under a second because heavy libraries load only after the time check. A lock file stops two cycles overlapping, and yfinance gets a hard 60-second deadline so a stalled download can't eat the next cycles.

STEP 08 / CONTINUED

● ● ● CLAUDE CODE PROMPT / STEP 08 (1/2) / PASTE AS ONE MESSAGE



Create cycle.py, the autonomous 5-minute trading cycle. It reuses bot.py sizing and strategy.evaluate. cycle.py connects on IBKR_CLIENT_ID and is READ-ONLY at the broker: every order action (entries, stop changes, partial sells, force-closes) runs `trade.py` as a subprocess and parses its RESULT line.

PREREQ CHECK: bot.py, trade.py, strategy.py, rules.json, morning_prefilter.py, src/ibkr_client.py exist; cycle.py does not.

main() in order:

1. TIME GATE using only the standard library (import pandas/yfinance/ib_async lazily, after the gate, so weekend and off-hours runs exit in under 1 s): weekend | too_early (<10:00) | closed ($\geq 16:00$) | manage_only (10:00-10:05 and after latest entry) | force_close ($\geq$ force_close_et) | ok. After connecting, re-check against IBKR's real session close for today (SPY liquidHours): exit on holidays; on half-days shift force-close and last-entry by the same distance from the close.
 2. LOCK file logs/cycle.lock (stale after 600 s) so runs never overlap. Write logs/heartbeat.json every run and one line per stage to logs/cycle_runs.jsonl.
 3. CONNECT (retry once after 5 s; on failure log connect_failed, alert, exit 1).
 4. RECONCILE with the broker (source of truth), state in open_positions.json (atomic writes):
 - position gone at the broker: look for a fill whose permId = the position's stop_perm_id (reqExecutions). Found $\rightarrow$ stop-out: append a SELL row with status StopFilled, alert. Not found $\rightarrow$ closed externally, alert. NEVER match by quantity.
 - quantity differs $\rightarrow$ trust the broker.
 - adopt broker positions bought TODAY by the bot (BUY rows in trades.csv) that state doesn't know (crash recovery).
 - every position must have a broker stop covering exactly its quantity: fix the quantity in place, or place one at the current stop / day low - 1%; if impossible, log UNPROTECTED_POSITION and alert at high priority.
 5. MANAGE with live IBKR prices (skip a position whose data is delayed; its broker stop still protects it):
 - state "open" and price $\geq$ entry + 0.75R: $k = \text{ceil}(\text{qty} \times 0.3333)$. FIRST shrink the stop to qty-k (modify in place), THEN sell k. If the sale fails, restore the stop to the full quantity. If k would be the whole position, flatten.
 - state "partial_done" and price $\geq$ entry + 1R: modify the stop to the entry price (breakeven), then state "trailing". Partial and breakeven may both happen in the same cycle.
 - "trailing": newest confirmed swing low among COMPLETED regular-session 5-minute bars since entry (low below the 2 bars before and 2 after). New stop = swing low - 0.01, only if above the current stop and below price (ratchet up only).
 - stops are always MODIFIED IN PLACE (trade.py --modify-stop), never cancelled and re-placed.
 6. FORCE CLOSE window: trade.py --flatten each position; keep failures in state so the next cycle retries.
 7. MANAGE ONLY window: stop here.
 8. ENTRY SCAN: daily entry cap from rules.json; read watchlist.txt only if its header date is TODAY; split lines on "#" (class shares contain a space); skip anything held (ib.positions()) FIRST) or in state. Pull yfinance daily (1y) and 5-minute (30d, prepost=True) data in one batch each, through a helper with a hard 60-second wall-clock deadline (daemon thread); on timeout log yfinance_timeout, alert, skip the step, and hard-exit at the end. Evaluate the six filters on COMPLETED bars only:
 - D1 last completed 5m close > prior day high; D2 prior close > 200-day SMA; D3 (official daily open - prior close) / prior close $\geq 3\%$; I1 last close > premarket high (04:00-09:30); I2 last completed bar closes above the high of the earlier bars today; I3 cumulative RTH volume through that bar / average cumulative volume at the same time of day over the previous 14 sessions ≥ 2.0 .
- Rank passing symbols by RVOL. For each: strategy.evaluate (live gate; skip if delayed), size with bot.size_position,

STEP 08 / CONTINUED

● ● ● CLAUDE CODE PROMPT / STEP 08 (2/2) / PASTE AS ONE MESSAGE



```

limit = ask x 1.002, run trade.py BUY --limit --stop, save state IMMEDIATELY after a fill.
9. Save state, release the lock, disconnect, rebuild the dashboard if one exists (never let that break trading).

Log every decision as one JSON line in safety-check-log.json (cast numpy types). Unhandled exception → traceback to
logs/cycle_errors.log, exit 1. Prefix prints with [HH:MM:SS ET].

AFTER: print "READY: run cycle.py; outside market hours it exits in under a second". Do not run it.

```

CHECK IT

Outside market hours `cycle.py` prints e.g. `too_early: nothing to do` instantly. In market hours it prints `status=ok` and one decision per watchlist symbol.

● ● ● 08:00, BEFORE THE SESSION

SIMULATED

```

PS C:\Projects\trading-bot> .venv\Scripts\python.exe cycle.py
[08:00:00 ET] too_early: nothing to do

```

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data.

● ● ● 10:06, THREE CANDIDATES, NONE READY

SIMULATED

```

PS C:\Projects\trading-bot> .venv\Scripts\python.exe cycle.py
[10:06:00 ET] status=ok (close 16:00)
[10:06:00 ET] entry_skipped: {"symbol": "H00D", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": false, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 103.73, "prior_high": 103.5, "sma200": 72.99, "gap_pct": 5.2, "premarket_high": 105.62, "prior_hod": 105.22, "rvol": 2.62, "trigger_bar": "10:00"}
[10:06:00 ET] entry_skipped: {"symbol": "AMD", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": false, "I2_close_above_prior_hod": true, "I3_rvol": true}, "last_close": 168.22, "prior_high": 162.1, "sma200": 129.23, "gap_pct": 4.69, "premarket_high": 168.4, "prior_hod": 167.95, "rvol": 3.47, "trigger_bar": "10:00"}
[10:06:00 ET] entry_skipped: {"symbol": "CCL", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": false, "D3_gap": true, "I1_above_premarket_high": true, "I2_close_above_prior_hod": true, "I3_rvol": true}, "last_close": 20.78, "prior_high": 20.3, "sma200": 22.33, "gap_pct": 3.4, "premarket_high": 20.69, "prior_hod": 20.72, "rvol": 2.9, "trigger_bar": "10:00"}

```

STEP 08 / CONTINUED

● ● ● 10:31, AMD PASSES ALL SIX FILTERS: BUY + BROKER STOP

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe cycle.py
[10:31:00 ET] status=ok (close 16:00)
[10:31:00 ET] entry_skipped: {"symbol": "HODD", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": false, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": false, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 102.9, "prior_high": 103.5, "sma200": 72.99, "gap_pct": 5.2, "premarket_high": 105.62, "prior_hod": 105.22, "rvol": 2.62, "trigger_bar": "10:25"}
[10:31:00 ET] entry_skipped: {"symbol": "CCL", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": false, "D3_gap": true, "I1_above_premarket_high": true, "I2_close_above_prior_hod": true, "I3_rvol": true}, "last_close": 21.01, "prior_high": 20.3, "sma200": 22.33, "gap_pct": 3.4, "premarket_high": 20.69, "prior_hod": 20.99, "rvol": 2.9, "trigger_bar": "10:25"}
[10:31:00 ET] entry_attempt: {"symbol": "AMD", "qty": 14, "limit": 170.2, "stop": 164.24, "sizing": "cap=14 risk-based=44 (stop 164.24, $5.61/sh at risk) → 14", "filters": {"pass": true, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": true, "I2_close_above_prior_hod": true, "I3_rvol": true}, "last_close": 169.82, "prior_high": 162.1, "sma200": 129.23, "gap_pct": 4.69, "premarket_high": 168.4, "prior_hod": 169.22, "rvol": 3.45, "trigger_bar": "10:25"}}
[10:31:00 ET] order 1: BUY 14/14 AMD @ 169.86 status=Filled
[10:31:00 ET] protective stop 2: SELL 14 @ 164.24 status=Submitted
RESULT {"mode": "order", "symbol": "AMD", "side": "BUY", "requested": 14, "filled": 14, "fill_price": 169.86, "status": "Filled", "order_id": 1, "perm_id": 1840211008, "stop_order_id": 2, "stop_perm_id": 1840211015, "stop_price": 164.24}
[10:31:00 ET] entered: {"symbol": "AMD", "qty": 14, "fill": 169.86, "stop": 164.24, "R": 5.62}
```

● ● ● 11:31, +0.75R: SHRINK THE STOP FIRST, THEN SELL A THIRD

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe cycle.py
[11:31:00 ET] status=ok (close 16:00)
[11:31:00 ET] stop 2 modified: 14@164.24 → 9@164.24 status=Submitted
RESULT {"mode": "modify_stop", "symbol": "AMD", "ok": true, "stop_order_id": 2, "stop_perm_id": 1840211015, "stop_price": 164.24, "stop_qty": 9, "status": "Submitted"}
[11:31:00 ET] order 3: SELL 5/5 AMD @ 174.21 status=Filled
RESULT {"mode": "order", "symbol": "AMD", "side": "SELL", "requested": 5, "filled": 5, "fill_price": 174.21, "status": "Filled", "order_id": 3, "perm_id": 1840211022}
[11:31:00 ET] partial_taken: {"symbol": "AMD", "sold": 5, "price": 174.21, "remaining": 9, "trigger_R": 0.75}
[11:31:01 ET] entry_skipped: {"symbol": "AMD", "reason": "already held"}
[11:31:01 ET] entry_skipped: {"symbol": "HODD", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": false, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": false, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 102.17, "prior_high": 103.5, "sma200": 72.99, "gap_pct": 5.2, "premarket_high": 105.62, "prior_hod": 105.22, "rvol": 2.61, "trigger_bar": "11:25"}
[11:31:01 ET] entry_skipped: {"symbol": "CCL", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": false, "D3_gap": true, "I1_above_premarket_high": true, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 21.18, "prior_high": 20.3, "sma200": 22.33, "gap_pct": 3.4, "premarket_high": 20.69, "prior_hod": 21.21, "rvol": 2.87, "trigger_bar": "11:25"}
```

STEP 08 / CONTINUED

● ● ● 12:01, +1R: BREAKEVEN, THEN TRAIL UNDER A SWING LOW

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe cycle.py
[12:01:00 ET] status=ok (close 16:00)
[12:01:00 ET] stop 2 modified: 9@164.24 → 9@169.86 status=Submitted
RESULT {"mode": "modify_stop", "symbol": "AMD", "ok": true, "stop_order_id": 2, "stop_perm_id": 1840211015, "stop_price": 169.86, "stop_qty": 9, "status": "Submitted"}
[12:01:00 ET] stop_to_breakeven: {"symbol": "AMD", "old": 164.24, "new": 169.86}
[12:01:00 ET] stop 2 modified: 9@169.86 → 9@172.85 status=Submitted
RESULT {"mode": "modify_stop", "symbol": "AMD", "ok": true, "stop_order_id": 2, "stop_perm_id": 1840211015, "stop_price": 172.85, "stop_qty": 9, "status": "Submitted"}
[12:01:01 ET] stop_trailed: {"symbol": "AMD", "old": 169.86, "new": 172.85, "swing_low": 172.86}
[12:01:01 ET] entry_skipped: {"symbol": "AMD", "reason": "already held"}
[12:01:01 ET] entry_skipped: {"symbol": "HODD", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": false, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": false, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 101.8, "prior_high": 103.5, "sma200": 72.99, "gap_pct": 5.2, "premarket_high": 105.62, "prior_hod": 105.22, "rvol": 2.6, "trigger_bar": "11:55"}
[12:01:01 ET] entry_skipped: {"symbol": "CCL", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": false, "D3_gap": true, "I1_above_premarket_high": true, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 21.17, "prior_high": 20.3, "sma200": 22.33, "gap_pct": 3.4, "premarket_high": 20.69, "prior_hod": 21.21, "rvol": 2.88, "trigger_bar": "11:55"}
```

● ● ● 13:31, STOP RATCHETS UP

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe cycle.py
[13:31:00 ET] status=ok (close 16:00)
[13:31:00 ET] stop 2 modified: 9@172.85 → 9@175.97 status=Submitted
RESULT {"mode": "modify_stop", "symbol": "AMD", "ok": true, "stop_order_id": 2, "stop_perm_id": 1840211015, "stop_price": 175.97, "stop_qty": 9, "status": "Submitted"}
[13:31:00 ET] stop_trailed: {"symbol": "AMD", "old": 172.85, "new": 175.97, "swing_low": 175.98}
[13:31:00 ET] entry_skipped: {"symbol": "AMD", "reason": "already held"}
[13:31:00 ET] entry_skipped: {"symbol": "HODD", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": false, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": false, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 101.58, "prior_high": 103.5, "sma200": 72.99, "gap_pct": 5.2, "premarket_high": 105.62, "prior_hod": 105.22, "rvol": 2.61, "trigger_bar": "13:25"}
[13:31:00 ET] entry_skipped: {"symbol": "CCL", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": false, "D3_gap": true, "I1_above_premarket_high": true, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 21.12, "prior_high": 20.3, "sma200": 22.33, "gap_pct": 3.4, "premarket_high": 20.69, "prior_hod": 21.21, "rvol": 2.88, "trigger_bar": "13:25"}
```



STEP 08 / CONTINUED

● ● ● 14:46, THE BROKER STOP FILLED; THE CYCLE RECONCILES IT

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe cycle.py
[14:46:00 ET] status=ok (close 16:00)
[14:46:00 ET] stopped_out: {"symbol": "AMD", "qty": 9, "price": 175.97, "r_multiple": 1.09}
[14:46:00 ET] entry_skipped: {"symbol": "H00D", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": false, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": false, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 101.39, "prior_high": 103.5, "sma200": 72.99, "gap_pct": 5.2, "premarket_high": 105.62, "prior_hod": 105.22, "rvol": 2.61, "trigger_bar": "14:40"}
[14:46:00 ET] entry_skipped: {"symbol": "AMD", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": true, "D3_gap": true, "I1_above_premarket_high": true, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 175.28, "prior_high": 162.1, "sma200": 129.23, "gap_pct": 4.69, "premarket_high": 168.4, "prior_hod": 179.44, "rvol": 3.41, "trigger_bar": "14:40"}
[14:46:00 ET] entry_skipped: {"symbol": "CCL", "reason": "filters", "pass": false, "checks": {"D1_above_prior_day_high": true, "D2_prior_close_above_sma200": false, "D3_gap": true, "I1_above_premarket_high": true, "I2_close_above_prior_hod": false, "I3_rvol": true}, "last_close": 21.09, "prior_high": 20.3, "sma200": 22.33, "gap_pct": 3.4, "premarket_high": 20.69, "prior_hod": 21.21, "rvol": 2.88, "trigger_bar": "14:40"}
```

● ● ● 15:51, ANOTHER RUN THAT HELD ON: FORCE CLOSE

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe cycle.py
[15:51:00 ET] status=force_close (close 16:00)
[15:51:00 ET] cancelling STP SELL 14 (orderId 2)
[15:51:00 ET] flatten AMD: SELL 14/14 @ 175.49 status=Filled
RESULT {"mode": "flatten", "symbol": "AMD", "ok": true, "filled": 14, "fill_price": 175.49, "status": "Filled", "order_id": 3}
[15:51:00 ET] force_closed: {"symbol": "AMD", "qty": 14, "price": 175.49}
```

● ● ● IF IT GOES WRONG



'watchlist is stale': the scanner hasn't run today yet (first scheduled run 09:52 ET).

Nothing trades for days: normal. All six filters must pass on the same completed bar.



STEP 09 / SRC/NOTIFY.PY (TELEGRAM)

PHONE ALERTS

Telegram messages for entries, partials, breakeven, trailing, stops, force closes, crashes and the daily summary.

● ● ● WHY IT'S BUILT THIS WAY



- HTML formatting, not Markdown: Telegram silently rejects Markdown messages that contain an unbalanced `_` `*` or `[` , which is common in error text, so you'd lose exactly the alerts that matter.
- If sending fails, the library's error contains the URL, token included. The token is redacted before anything is logged.
- The scanner only messages you when its list is new or changed (not 13 identical messages a day). Critical alerts are de-duplicated hourly. A notification can never break trading logic.

● ● ● BEFORE YOU PASTE THE PROMPT



1. On Telegram, message @BotFather, send `/newbot` and follow the prompts. Save the token.
2. Message @userinfobot with `/start` to get your chat ID.
3. Paste both into `.env` (`TELEGRAM_BOT_TOKEN=`, `TELEGRAM_CHAT_ID=`). Never paste them into a chat window.
4. Open a chat with your new bot and press Start, or it can't message you.



STEP 09 / CONTINUED

● ● ● CLAUDE CODE PROMPT / STEP 09 / PASTE AS ONE MESSAGE



Add Telegram push notifications. Do NOT change any trading logic; only add notification calls.

PREREQ CHECK: cycle.py, morning_prefilter.py, src/__init__.py exist; requests imports; src/notify.py does not exist.

CREATE src/notify.py with notify(title, body, priority="default") → None:

- Load TELEGRAM_BOT_TOKEN, TELEGRAM_CHAT_ID and optional NOTIFY_URL (ntfy) from .env (explicit path).
- Telegram: POST sendMessage with parse_mode="HTML" (NOT Markdown) and html-escaped title/body, "⚠️" prefix for high priority, 5 s timeout, max 4096 chars. ntfy: POST body with Title and Priority headers.
- Never raise. Log failures to logs/notify_errors.log with the bot token REDACTED (also redact any bot<digits>:<key>).
- Never print the token or chat ID anywhere.

WIRE IT IN (each call wrapped so a failure can't affect trading):

- morning_prefilter.py (not on --dry-run): message only when the list is the first of the day or changed: "Prefilter HH:MM ET" with "N/503 survivors" and a bullet per ticker. Failures and tripwires at high priority.
- cycle.py: BUY, PARTIAL, BE, TRAIL, STOP, CLOSED (position gone, not a bot stop), EXIT, "EOD Force Close" (high), "Cycle CRASHED" (high), yfinance timeout; critical events (UNPROTECTED_POSITION, FORCE_CLOSE_FAILED, connect_failed, trade_timeout) at high priority, de-duplicated per symbol for an hour.

AFTER: print how to test: .venv\Scripts\python.exe -c "from src.notify import notify; notify('Test', 'Hello')"

CHECK IT

The test command prints nothing and a 'Test / Hello' message arrives on your phone.

STEP 09 / CONTINUED

Trend Join Bot (Telegram)

Prefilter 09:52 ET
3/503 survivors in 2s

- HOOD +5.20%
- AMD +4.69%
- CCL +3.40%

09:53

BUY AMD
@ \$169.86, stop \$164.24, qty 14

10:31

PARTIAL AMD
sold 5/14 @ \$174.21

11:31

BE AMD
stop → \$169.86

12:01

TRAIL AMD
stop \$169.86 → \$172.85

12:01

TRAIL AMD
stop \$172.85 → \$175.97

13:31

STOP AMD
exit \$175.97, P&L \$+54.99

14:46

Daily Summary 2026-10-05
Trades: 1 (1W / 0L, 100%)
P&L: \$+76.74 (gross)
Best: AMD \$+76.74
Worst: AMD \$+76.74
PF: inf

16:05

● ● ● WHAT YOU GET



Every message the simulated day produced, in order: the 09:52 scan, the entry, the partial, breakeven, two trailing moves, the stop-out, and the 16:05 summary.

Alerts are informational. The protective stop lives at IBKR, so a missed message never leaves a position unprotected.

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data.

● ● ● IF IT GOES WRONG



No message: you never pressed Start in the bot's chat, or the chat ID is wrong. Check logs/notify_errors.log.

STEP 10 / COMPUTE_PERF.PY, ROTATE_LOGS.PY

DAILY SUMMARY AND LOG ROTATION

One summary after the close, and housekeeping so the bot can run for months.

● ● ● WHY IT'S BUILT THIS WAY

- Trades are matched by SHARES, first-in-first-out. The bot exits in pieces (a third at +0.75R, the rest later), so pairing rows one-to-one would count one trade as two.
- R per trade is measured against the dollars actually risked (entry minus the initial stop recorded at entry), not a fixed 1% of the account.
- Only one summary a day, at 16:05, after the 15:56 force-close retry. rotate_logs never rewrites trades.csv during market hours, when trade.py might be appending a fill.

● ● ● CLAUDE CODE PROMPT / STEP 10 / PASTE AS ONE MESSAGE

Add the daily performance summary and log rotation.

CREATE compute_perf.py (read-only on trades.csv; safe any time):

- Load trades.csv. Match BUY and SELL by SHARES, FIFO per symbol: one BUY lot is one trade, closed when all its shares are sold (partials roll into the same trade). Use ET dates.
- Per trade: entry, average exit, P&L \$, P&L %, hold minutes, initial stop (BUY row's stop_price; else the cycle's decision log; else a 1% proxy flagged as estimated), $R = (\text{avg exit} - \text{entry}) / (\text{entry} - \text{initial stop})$, exit kind.
- Aggregates: total_trades, wins, losses, win_rate_pct, gross_pnl_usd, largest winner/loser, avg winner/loser, profit_factor ("inf" if no losses, "n/a" if no trades). Print JSON.
- Unless --no-notify: one Telegram message "Daily Summary YYYY-MM-DD". Options --date, --no-notify, --no-dashboard.

CREATE rotate_logs.py (exit 0 even with nothing to do):

- logs/*.log and logs/*.jsonl last modified before today (ET) → logs/archive/<date>/ with os.replace.
- trades.csv rows older than 90 days → logs/archive/trades_<today>.csv, but NEVER between 09:25 and 16:15 ET.
- safety-check-log.json over 5 MB → archived with today's date.
- Skip files another process has open (Windows) and retry next run. Staged, atomic swaps only.

AFTER: print how to run both. Do not run them.

CHECK IT

After a trading day: `compute_perf.py --no-notify` prints the JSON summary.

STEP 10 / CONTINUED

● ● ● AFTER THE CLOSE

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe compute_perf.py --no-notify
{
  "date": "2026-10-05",
  "total_trades": 1,
  "wins": 1,
  "losses": 0,
  "win_rate_pct": 100.0,
  "gross_pnl_usd": 76.74,
  "largest_winner": {
    "symbol": "AMD",
    "pnl_usd": 76.74
  },
  "largest_loser": {
    "symbol": "AMD",
    "pnl_usd": 76.74
  },
  "avg_winner": 76.74,
  "avg_loser": null,
  "profit_factor": "inf",
  "trades": [
    {
      "symbol": "AMD",
      "qty": 14,
      "entry": 169.86,
      "avg_exit": 175.3414,
      "pnl_usd": 76.74,
      "pnl_pct": 3.23,
      "hold_min": 255.0,
      "initial_stop": 164.24,
      "r": 0.98,
      "r_estimated": false,
      "exit_kind": "stop",
      "entry_ts": "2026-10-05T10:31:00-04:00",
      "exit_ts": "2026-10-05T14:46:00-04:00"
    }
  ],
  "open_at_run": [],
  "unmatched_sells": []
}
dashboard: C:\Projects\trading-bot\dashboard\index.html
```

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data.

● ● ● IF IT GOES WRONG



A 15:51 summary misses the 15:56 force-close retry. Keep the single 16:05 summary.



PUT IT ON AUTOPILOT

Register five Windows scheduled tasks that run everything on weekdays, plus a one-command emergency stop.

● ● ● WHY IT'S BUILT THIS WAY



- Priority 4, not Task Scheduler's default 7. At priority 7 Windows also lowers disk priority: a cycle that took 4 seconds by hand took about 60 seconds (3.5 minutes cold) when scheduled. At 4 it takes about 2 seconds.
- Tasks are registered from XML, which plain schtasks flags can't express: run on battery, wake the computer, catch up after sleep, never overlap, a 10-minute time limit, and repeat only inside a weekday window. XML dates are ISO, so regional date formats can't break it.
- pythonw.exe runs without a console window popping up every 5 minutes. Five tasks with repetition replace the tutorial's eleven.
- The scanner runs at :52 and :22, not :55 and :25, so its 1 to 2 minute download finishes before the next cycle asks Yahoo for data. Overlapping them stalled three cycles on day two.

TASK	WHEN (ET, MON-FRI)	RUNS
HT_LogRotate	09:15	rotate_logs.py
HT_KeepAwake	09:30	powercfg: no sleep on AC power
HT_Prefilter	09:52, every 30 min to 12:52	morning_prefilter.py (7 runs)
HT_Cycle	10:01, every 5 min to 15:56	cycle.py (72 runs)
HT_Dashboard	16:05	compute_perf.py



STEP 11 / CONTINUED

● ● ● CLAUDE CODE PROMPT / STEP 11 / PASTE AS ONE MESSAGE



Create `setup_schedule.py` and `cleanup_schedule.py` for Windows Task Scheduler (current user, no admin).

PREREQ CHECK: `cycle.py`, `morning_prefilter.py`, `compute_perf.py`, `rotate_logs.py` and `.venv\Scripts\pythonw.exe` exist; we are on Windows (``where schtasks``). Otherwise STOP.

`setup_schedule.py` [--dry-run]:

- Register these 5 tasks, Monday-Friday, times in ET converted to this PC's local time with zoneinfo:
HT_LogRotate 09:15 rotate_logs.py; HT_KeepAwake 09:30 `powercfg /change standby-timeout-ac 0`;
HT_Prefilter 09:52 repeating every 30 min until 12:52; HT_Cycle 10:01 repeating every 5 min until 15:56;
HT_Dashboard 16:05 compute_perf.py.
- Build each as Task Scheduler XML and register with ``schtasks /create /tn NAME /xml FILE /f`` (XML saved as UTF-16 in `logs/schedule/`). Settings: Priority 4 (NOT the default 7, which also lowers I/O priority and made cycles ~15x slower), `DisallowStartIfOnBatteries false`, `StopIfGoingOnBatteries false`, `WakeToRun true`, `StartWhenAvailable true`, `MultipleInstancesPolicy IgnoreNew`, `ExecutionTimeLimit PT10M`, working directory = project folder, `command = .venv\Scripts\pythonw.exe` (no console windows). Use ISO dates in the XML (no locale parsing).
- Before changing power settings, save the current sleep timeout to `logs/power_backup.json`.
- Warn if this PC's time zone changes daylight saving time on different dates than New York.
- --dry-run: write the XML only. Print a table of HT_* tasks with next run times.

`cleanup_schedule.py`: delete every HT_* task and restore the saved sleep setting (or 30 min if none). Idempotent; "Nothing to clean." when there are none. This is the emergency stop.

AFTER: print how to run both. Do NOT register anything yourself.

CHECK IT

`setup_schedule.py` prints SCHEDULED: 5 tasks created and the next HT_Cycle time.

● ● ● IF IT GOES WRONG



Task Scheduler result 0x41303 on a task that hasn't run yet just means 'not yet run'.

If cycles take a minute instead of seconds, the task is at priority 7.

STEP 12 / DASHBOARD.PY

THE PERFORMANCE DASHBOARD

A self-contained HTML page with P&L, R-multiples, open positions with live R, bot status and recent activity.

● ● ● WHY IT'S BUILT THIS WAY

- R comes from the actual stop recorded at entry, so the dashboard measures the edge, not the position-size cap.
- Status comes from the cycle's heartbeat (ACTIVE, STALE, IBKR DOWN, IDLE, NO DATA), not a hard-coded ACTIVE label.
- It is rebuilt every cycle (a once-a-day file that refreshes every 30 s would show stale data all day). No JavaScript and no CDNs, so it opens straight from disk and can be hosted anywhere.

● ● ● CLAUDE CODE PROMPT / STEP 12 / PASTE AS ONE MESSAGE

Create dashboard.py and call it from compute_perf.py and at the end of each working cycle (wrapped so it can never affect trading).

dashboard.py builds dashboard/index.html, self-contained: inline CSS and inline SVG charts, NO JavaScript, NO CDN links, <meta http-equiv="refresh" content="30">, <meta name="robots" content="noindex">. Dark theme.

Content, from trades.csv (all history), open_positions.json, logs/heartbeat.json and safety-check-log.json:

- KPI strip: today P&L, total P&L (% of PORTFOLIO_VALUE_USD), win rate, expectancy (R and \$), profit factor, max drawdown and loss streak.
- Bot status from the heartbeat: ACTIVE (recent cycle), STALE (cycles stopped during market hours), IBKR DOWN (connect_failed), IDLE (outside hours), NO DATA. Show heartbeat time, entries today vs cap, risk settings.
- Equity curve (cumulative P&L per closed trade), R-multiple histogram ($\leq -2R$... $>+3R$), daily P&L bars.
- Open positions: qty, entry, current stop, last price, unrealized \$ and R, and a small track from stop to +2R.
- Recent closed trades (last 20) with R colour-coded, a by-symbol table, and the latest bot decisions.

Use the FIFO/R logic from compute_perf.py. Escape all text. Axis ticks must enclose the data range.

AFTER: print "READY: run dashboard.py and open dashboard/index.html". Do not run it.

CHECK IT

Open dashboard/index.html in a browser.

STEP 12 / CONTINUED

● ● ● DASHBOARD/INDEX.HTML AFTER THE SIMULATED DAY



WF WORKFLOW FACTORY

● IDLE Paper account · IBKR

TRADING SYSTEMS / PAPER BOT

PERFORMANCE WORKSPACE / 01

TREND JOIN LONG

Long-only S&P 500 gap continuation on 5-minute bars. Every entry, exit and stop move comes from the bot's own records. Updated Mon Oct 05, 16:06:00 ET · refreshes every 30s.

+\$76.74

TODAY P&L
1 closed · 1W

+\$76.74

TOTAL P&L
+0.31% of \$25,000

100%

WIN RATE
1W / 0L

+0.98R

EXPECTANCY
+\$76.74 / trade

∞

PROFIT FACTOR
avg W +\$76.74

\$0.00

MAX DRAWDOWN
loss streak 0 · now
1W

01 EQUITY CURVE

cumulative gross P&L per closed trade

- The equity curve appears after two closed trades.

02 BOT STATUS

IDLE

Outside trading window · Last cycle 1.3 h ago ·
Mon 14:46 ET · ok

Heartbeat	Mon Oct 05 · 14:46:00
Cycle state	ok

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data. Shown with the Workflow Factory site styling, as on our live dashboard; the dashboard.py you build has the same panels with plainer styling.

● ● ● IF IT GOES WRONG



Dashboard never changes: cycles aren't running, or it is only rebuilt once a day.

STEP 13 / CHECK-ONLY, END-TO-END, SCHEDULE

TEST, THEN GO LIVE ON PAPER

Prove the whole chain on the paper account during market hours, then switch on the schedule.

● ● ● WHY IT'S BUILT THIS WAY



- Code that has only run off-hours hasn't met real quotes, real P&L, or a real order book. One supervised round trip catches what simulations can't.

● ● ● TEST-DAY RUNBOOK



09:45 ET	TWS open and logged in to PAPER. Live account logged OUT everywhere (Client Portal, IBKR Mobile).
After 10:05	<code>bot.py --symbol NVDA --check-only</code> → 'pass': True, price_source 'ticks', data_delayed False.
Then	<code>bot.py --symbol NVDA</code> → buys a small position and places the stop. Check both in TWS.
A few minutes later	<code>close_one.py --symbol NVDA</code> → cancels the stop (Error 202 = cancel confirmation), sells, flat.
Check	<code>compute_perf.py --no-notify</code> shows 1 trade with a real R; open the dashboard.
Only if all passed	<code>setup_schedule.py --dry-run</code> , then <code>setup_schedule.py</code> . Watch Telegram for the 09:52 scan.

● ● ● CLAUDE CODE PROMPT / STEP 13 / PASTE AS ONE MESSAGE



It's market hours. Walk me through the end-to-end paper test one command at a time and check each output before the next: (1) `bot.py --symbol NVDA --check-only`, (2) `bot.py --symbol NVDA`, (3) verify in IBKR that the position and its stop exist, (4) `close_one.py --symbol NVDA`, (5) verify flat with no working orders, (6) `compute_perf.py --no-notify`. If anything fails, stop and diagnose; do not schedule anything. If all pass, show me `setup_schedule.py --dry-run` and wait for my go-ahead before registering the tasks.

CHECK IT

Telegram shows the 09:52 prefilter message the next trading morning and the heartbeat updates every 5 minutes.



STEP 13 / CONTINUED

● ● ● END-TO-END TEST: BUY

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe bot.py --symbol AMD
[10:41:00 ET] evaluate: {'ts_et': '2026-10-05T10:41:00-04:00', 'symbol': 'AMD', 'pass': True, 'reasons': ['time gate ok', 'no
existing position', 'session open', 'portfolio limits ok'], 'price': 170.73, 'ask': 170.74, 'day_low': 165.9, 'price_source':
'ticks', 'data_delayed': False, 'daily_pnl': 0.0, 'pnl_source': 'ibkr_pnl'}
[10:41:00 ET] sizing: cap=14 risk-based=38 (stop 164.24, $6.49/sh at risk) → 14
[10:41:00 ET] placing: BUY 14 AMD limit 171.08, stop 164.24
[10:41:00 ET] order 1: BUY 14/14 AMD @ 170.74 status=Filled
[10:41:00 ET] protective stop 2: SELL 14 @ 164.24 status=Submitted
RESULT {"mode": "order", "symbol": "AMD", "side": "BUY", "requested": 14, "filled": 14, "fill_price": 170.74, "status":
"Filled", "order_id": 1, "perm_id": 1840211008, "stop_order_id": 2, "stop_perm_id": 1840211015, "stop_price": 164.24}
[10:41:00 ET] SUCCESS: BUY 14 AMD @ 170.74 (order 1)
```

Captured from the real bot code running against a simulated broker and synthetic prices (Mon 5 Oct 2026, a made-up day). Not real trades or real market data.

● ● ● END-TO-END TEST: CLOSE

SIMULATED

```
PS C:\Projects\trading-bot> .venv\Scripts\python.exe close_one.py --symbol AMD
Cancelling working STP SELL 14 (orderId 2, clientId 3)
Order ID: 3
Sold qty: 14 of 14
Fill price: 171.07
Status: Filled
```

● ● ● IF IT GOES WRONG



Error 162 'different IP address': the live account is logged in somewhere else. Log it out.

EVERY PROBLEM WE HIT

TROUBLESHOOTING

Real errors from the build, in the order you are likely to meet them. Search this PDF for the error code or the exact message.

Order cancelled right away: error 10349 **Step 01**

YOU SEE	The close order was cancelled immediately; TWS reported error 10349 (order TIF was set to DAY based on order preset / preset conflict).
CAUSE	The order had no time-in-force, so TWS applied its own preset and cancelled it.
FIX	Set order.tif = "DAY" explicitly on every order the bot sends: entries, stops, exits. All the prompts in this guide already do.

Order stuck in PendingSubmit **Step 01**

YOU SEE	buy_one.py printed status PendingSubmit and no fill.
CAUSE	The market was closed (or TWS was waiting on a confirmation). PendingSubmit is not a final state.
FIX	Wait for a terminal state (Filled, Cancelled, ApiCancelled, Inactive). Outside market hours the order fills at the next open; cancel it in TWS if you don't want that.

'no usable market price' **Step 05**

YOU SEE	bot.py --check-only returned pass False with reason 'no usable market price (no ticks, no bars)'.
CAUSE	No live market-data subscription reached the API (error 354 / warning 2186), and the delayed stream was silent.
FIX	Subscribe to live data in the LIVE account (see the next entries), share it with the paper account, restart TWS. The bot falls back to 5-minute bars and flags data_delayed so it never trades on stale prices.

Error 162: 'Trading TWS session is connected from a different IP address' **Step 00**

YOU SEE	Error 162, Historical Market Data Service error message: Trading TWS session is connected from a different IP address.
CAUSE	The LIVE account was logged in somewhere else (Client Portal, IBKR Mobile, another TWS). Market data can only stream to one session, and the paper account borrows the live account's data.
FIX	Log the live account out everywhere, then restart TWS on the paper login. Keep it logged out during trading hours.

TROUBLESHOOTING / CONTINUED

Warning 2186: real-time data 'requires additional subscription for API'

Step 00

YOU SEE	Warning 2186: Requested real-time market data requires additional subscription for API. You elected to receive delayed market data instead.
CAUSE	The free non-consolidated quotes in the portal are not enabled for the API.
FIX	In the live account's Client Portal subscribe (Non-Professional) to 'US Securities Snapshot and Futures Value Bundle' and 'US Equity and Options Add-On Streaming Bundle', then enable market data sharing with the paper account.

Can't subscribe to the data bundle

Step 00

YOU SEE	The Client Portal refused the market-data bundle subscription.
CAUSE	IBKR requires the live account to hold equity (about US\$500) before it sells market-data subscriptions.
FIX	Fund the live account (about US\$500 was enough), then subscribe and share data with paper. You still trade only on paper.

Professional or Non-Professional?

Step 00

YOU SEE	The subscription form asks whether you are a professional subscriber.
CAUSE	Exchanges charge professionals much more.
FIX	Individuals trading their own money are normally Non-Professional. Answer the questions honestly; the build used Non-Professional.

Error 10185 when cancelling P&L

Step 05

YOU SEE	Error 10185 (failed to cancel PnL) while checking the daily loss limit.
CAUSE	A fixed 1.5-second wait raced the first P&L update from TWS.
FIX	Poll until the value arrives (up to about 6 seconds) and fail closed if it never does.

'daily PnL unavailable; cannot verify loss limit'

Step 05

YOU SEE	bot.py refused the test trade: 'daily PnL unavailable; cannot verify loss limit'.
CAUSE	IBKR's reqPnL stream can stay silent with no error right after a fresh TWS login. The gate correctly refused to trade blind.
FIX	The bot now falls back to realized P&L from its own trades.csv plus unrealized P&L from the portfolio, and reports pnl_source. If both are unavailable it still refuses.

TROUBLESHOOTING / CONTINUED

First live price takes ~5 seconds **Step 05**

YOU SEE	Right after logging in to TWS, the first price tick took about 5 seconds; the code only waited 3.
CAUSE	A cold TWS session is slow to start streaming.
FIX	Wait up to 10 seconds for the first tick (both strategy.py and cycle.py).

'Error 202 Order Canceled' when closing **Step 02**

YOU SEE	close_one.py printed Error 202, reqId N: Order Canceled - reason:
CAUSE	That's IBKR confirming the cancellation of the resting stop before the sell. Not a failure.
FIX	Nothing to fix. The script cancels working SELL stops first on purpose, so the stop can't trigger later on a flat position.

Error 300: "Can't find EId with tickerId" **Step 05**

YOU SEE	Error 300, reqId N: Can't find EId with tickerId:N during a market-data test.
CAUSE	Cancelling a market-data request that TWS had already dropped (it happens when switching data types in quick succession).
FIX	Harmless. It doesn't affect the bot's own price requests.

'watchlist is stale (yesterday), not today' **Step 08**

YOU SEE	Decision log: no_entries, reason 'watchlist is stale (2026-09-28), not 2026-09-29'.
CAUSE	The cycle ran before the day's first scan. The bot refuses to trade yesterday's gappers.
FIX	Normal before 09:52 ET. If it persists after 10:00, check the HT_Prefilter task and the Yahoo connection.

Task Scheduler result 0x41303 **Step 11**

YOU SEE	Get-ScheduledTaskInfo shows LastTaskResult 0x41303 and a last run date of 1999.
CAUSE	0x41303 means 'the task has not yet run'.
FIX	Nothing to fix. It changes after the first scheduled run.

TROUBLESHOOTING / CONTINUED

Scheduled cycle takes a minute instead of seconds **Step 11**

YOU SEE By hand a cycle took about 4 seconds; scheduled it took about 58 seconds (3.5 minutes on a cold start).

CAUSE Task Scheduler's default priority 7 also lowers the process's disk (I/O) priority.

FIX Register the tasks from XML with `<Priority>4</Priority>`. Cycles dropped to about 2 seconds.

Some 5-minute cycles left no trace **Step 08**

YOU SEE Cycles at 10:26, 10:31 and 10:36 left no heartbeat or log line.

CAUSE Most likely the cycle stalled inside a yfinance download while the 10:25 scanner was pulling all 503 tickers from Yahoo at the same time (not proven: Task Scheduler history was off).

FIX Scanner moved to :52/:22 so it finishes before the next cycle; a hard 60-second deadline on the cycle's downloads (then a hard exit); a per-stage trace in logs/cycle_runs.jsonl to show where any future stall happens.

Bot stops when the computer sleeps **Step 13**

YOU SEE Scheduled runs are missed overnight or mid-day on a laptop or VM.

CAUSE The host went to sleep. On a Mac running a Windows VM, the Mac itself must stay awake.

FIX Windows: HT_KeepAwake disables sleep on AC power during the session. Mac host: keep it plugged in, lid open, and run ``caffeinate -s`` (check ``pmset -g assertions`` shows `PreventSystemSleep 1`). Keep the Windows user signed in (a locked screen is fine).

Does TWS need a manual login every morning? **Step 13**

YOU SEE TWS closes at night or asks for a login.

CAUSE TWS has a daily restart, and IBKR requires a full re-login about once a week.

FIX Set Lock and Exit to Auto restart (e.g. 11:45 PM), not Auto log off; then it reconnects by itself daily. Log in manually when IBKR asks (about weekly). If TWS is down the bot fails safe: `connect_failed` alert, no trades.

Connection refused / connect_failed **Step 00**

YOU SEE ERROR: IBKR connection failed: ConnectionRefusedError ... ('127.0.0.1', 7497)

CAUSE TWS isn't running, is logged in to live (port 7496), or the API socket isn't enabled.

FIX Start TWS on the PAPER login; File > Global Configuration > API > Settings: enable socket clients, port 7497, trusted IP 127.0.0.1.

TROUBLESHOOTING / CONTINUED

'ABORT: paper flag but live port' or 'account guard' **Step 05**

YOU SEE	ABORT: paper flag but live port / account guard: expected 'D*' accounts, got ['U...']
CAUSE	A safety guard fired: .env says paper but the port is a live one, or TWS is logged in to a live account.
FIX	Working as intended. Log TWS in to the paper account (DU...) on port 7497. Never weaken these guards.

The bot runs all day but never trades **Step 08**

YOU SEE	Every cycle logs entry_skipped; zero trades for days.
CAUSE	All six filters must pass on the same completed bar. Common blockers in the real logs: a stock below its 200-day average (D2), or a gap that never reclaims its premarket high or makes a new high (I1/I2). Relative volume also fades through gap days, so afternoon entries are rare.
FIX	That is the strategy being selective, not a bug. Read the decision log (safety-check-log.json): each skip lists which checks failed. Don't loosen rules without a backtest.

Telegram alerts silently missing **Step 09**

YOU SEE	Some alerts never arrive, especially ones with error text.
CAUSE	Telegram's Markdown mode rejects messages containing an unbalanced _ * or [.
FIX	Send with parse_mode HTML and escape the text (the notify.py prompt does this). Check logs/notify_errors.log (the token is redacted there).

Risk: a stale stop opening a short **Step 02**

YOU SEE	(Near miss, prevented) A position sold manually while its broker stop was still working.
CAUSE	If the stop triggers later, it sells shares you no longer own, which opens a short.
FIX	Every exit path cancels or shrinks the resting stop BEFORE selling: close_one.py, trade.py --flatten, and the partial-profit logic.

Bot can't modify its own stop **Step 08**

YOU SEE	(Spec bug, fixed before it ran) The tutorial's cycle.py tried to change stops placed by trade.py.
CAUSE	IBKR only lets the client ID that placed an order modify or cancel it.
FIX	All order actions go through trade.py on client ID 3; cycle.py (client 2) is read-only.

TROUBLESHOOTING / CONTINUED

BRK B / BF B fail in the scanner **Step 06**

YOU SEE	Class-share tickers return no data.
CAUSE	IBKR uses a space (BRK B), Yahoo uses a hyphen (BRK-B), and splitting watchlist lines on spaces breaks the symbol.
FIX	Keep IBKR format in the list, convert for Yahoo, and split watchlist lines on '#'.

ModuleNotFoundError: No module named 'ib_async' (or dotenv, pandas) **Step 04**

YOU SEE	ModuleNotFoundError: No module named 'ib_async' / No module named 'dotenv'
CAUSE	The script was run with the system Python instead of the project's virtual environment, or requirements weren't installed into .venv.
FIX	Always run <code>`.venv\Scripts\python.exe script.py`</code> . If it still fails, run <code>`.venv\Scripts\python.exe -m pip install -r requirements.lock`</code> .



DAY TO DAY

RUNNING IT EVERY DAY

● ● ● EVERY TRADING MORNING (BY 09:45 ET)



TWS	Running and logged in to PAPER (Auto restart keeps it up; IBKR asks for a manual login about weekly).
Live account	Logged out everywhere: Client Portal, IBKR Mobile. Otherwise market data stops (error 162).
Computer	Plugged in, awake, Windows user signed in (locked screen is fine). Mac host: <code>caffeinate -s</code> .
Telegram	The 09:52 prefilter message arrives. If not, check the HT_Prefilter task.

● ● ● USEFUL COMMANDS



Health check	<code>bot.py --symbol NVDA --check-only</code>
Close one position	<code>close_one.py --symbol XYZ</code> (cancels its stop first)
Scanner test	<code>morning_prefilter.py --dry-run</code>
Today's summary	<code>compute_perf.py --no-notify</code>
Rebuild dashboard	<code>dashboard.py</code>
Preview schedule	<code>setup_schedule.py --dry-run</code>
EMERGENCY STOP	<code>cleanup_schedule.py</code> : removes all HT_* tasks and restores your sleep setting



RUNNING IT / CONTINUED

● ● ● WHERE THINGS ARE WRITTEN



trades.csv	Every fill: time (UTC), symbol, side, size, price, order ID, status, initial stop
safety-check-log.json	Every decision, one JSON line each, including why a stock was skipped
open_positions.json	The cycle's view of open positions (the broker always wins)
logs/heartbeat.json	Last cycle time and status: what the dashboard's status badge reads
logs/cycle_runs.jsonl	Per-stage timing trace of each cycle
logs/cycle_errors.log	Tracebacks if a cycle crashes (you also get a Telegram alert)

CHANGE ONE THING AT A TIME

Edit `rules.json` outside market hours, run a check-only, and watch a full day before the next change. Never edit the bot's files between 09:15 and 16:15 ET: scheduled tasks import them every 5 minutes.



QUESTIONS PEOPLE ASK

FAQ

What is the Trend Join Long strategy?

A long-only, intraday strategy on S&P 500 stocks using 5-minute bars. It buys stocks that gapped up at least 3% and keep showing strength: above yesterday's high, above the 200-day average, above the premarket high, breaking to a new high of the day on at least twice the usual volume for that time of day. Entries run 10:05 to 15:30 ET and everything is closed by 15:51 ET.

What are the six filters?

Daily: D1 last completed 5-minute close above the prior day's high; D2 prior close above the 200-day SMA; D3 open at least 3% above the prior close. Intraday: I1 above today's premarket high (04:00 to 09:30); I2 a completed 5-minute bar closes above the high of the day made before it; I3 relative volume at least 2.0, comparing volume so far with the same time of day over the last 14 sessions. All six must pass on the same bar.

How does the bot exit a trade? Is there a take-profit?

No fixed take-profit. The initial stop is the day's low minus 1%, placed at the broker as soon as the entry fills. At $+0.75R$ it sells a third; at $+1R$ it moves the stop to the entry price; after that it trails the stop under each new confirmed 5-minute swing low. Anything left is sold at 15:51 ET. R is the distance from entry to the initial stop.

How big is each position?

The smaller of two numbers: risk-based shares (1% of `PORTFOLIO_VALUE_USD` divided by the distance to the stop) and a 10% position cap. With the default \$25,000 that's at most \$250 at risk and \$2,500 per position, with up to 5 positions and 5 new entries a day, and a 3% daily loss stop.

Does it trade real money?

No. It is built for an IBKR paper account only, and two independent guards enforce it: a port guard (paper flag with a live port aborts) and an account guard (every account ID must start with D). Paper results are not evidence that a strategy is profitable, and nothing here is financial advice.

[FAQ / CONTINUED](#)

Why is IBKR asking for a deposit when I only want paper trading?

Paper accounts get live market data by sharing the live account's subscriptions, and IBKR only sells those once the live account holds some equity (about US\$500 in our build). The money stays in your account; the bot never trades it.

What does it cost to run?

The libraries are free and open source (ib_async, pandas, yfinance). IBKR's market-data bundles have monthly fees set by IBKR; check the current prices on your account's market-data page. Telegram is free. Claude Code needs a Claude plan.

Why two client IDs?

IBKR only lets the client that placed an order modify or cancel it. So one process (trade.py, client 3) places and changes every order, and the decision-maker (cycle.py, client 2) only reads. That keeps every stop manageable.

Why yfinance for scanning and IBKR for trading?

Scanning 503 stocks through IBKR hits its concurrent market-data limits. Yahoo data via yfinance is free and fine for screening. Every order and every live price used for sizing and exits comes from IBKR.

What does R mean?

R is the risk per share at entry: entry price minus the initial stop. A trade closed at +1R made as much as it risked. The bot and dashboard measure R against the stop actually recorded at entry.

How do I stop the bot immediately?

Run `.venv\Scripts\python.exe cleanup_schedule.py`. It deletes all HT_* scheduled tasks and restores your sleep setting. To exit an open position by hand use `close_one.py --symbol XYZ`, which cancels the resting stop first.

FAQ / CONTINUED

Do I need to leave my computer on?

Yes, during market hours, with TWS running and logged in to paper and the Windows user signed in (a locked screen is fine). The schedule wakes the PC, and HT_KeepAwake blocks sleep on AC power. On a Mac running a Windows VM, keep the Mac plugged in and run `caffeinate -s`.

Can I run it on a Mac or Linux?

The trading code is cross-platform Python; only the scheduler scripts use Windows Task Scheduler. The reference build runs in a Windows 11 VM on a MacBook. On macOS or Linux you would ask Claude Code to schedule the same five jobs with launchd or cron instead.

Which Python version?

Python 3.12 or newer. The reference build ran 3.14.7 with `ib_async 2.1.0`, `pandas 3.0.6` and `yfinance 1.7.0` (pinned in `requirements.lock`).

Why does the bot skip a stock that looks like it's breaking out?

Open `safety-check-log.json`: every skip lists the six checks with true/false and the numbers behind them (prior high, 200-day SMA, gap, premarket high, prior high of day, RVOL). In the first live week, breakouts were blocked by D2 (below the 200-day average) and gappers that never reclaimed their premarket high.

How often does it trade?

Rarely, by design. Days with no gapper passing all six filters are common: the first two unattended days produced 13 candidates and zero trades. Don't judge it on a few days.

What changed compared with the Humbled Trader tutorial?

The structure is the same 13 steps. We fixed the bugs we found and added safety: stop changes routed through one client, partial-then-breakeven order, stop shrunk before partial sells, in-place stop modification, stop-outs matched by permanent ID with broker reconciliation, half-day awareness, lazy imports, gap measured at the open, explicit DAY time-in-force, account and port guards, a P&L fallback, priority-4 XML tasks, HTML Telegram with token redaction, FIFO share matching, and a heartbeat-driven dashboard.



FAQ / CONTINUED

Where are the logs?

trades.csv (every fill), safety-check-log.json (every decision), open_positions.json (cycle state), watchlist.txt, logs/heartbeat.json (last cycle), logs/cycle_runs.jsonl (per-stage trace), logs/cycle_errors.log (crashes), logs/notify_errors.log (alert failures), and logs/archive/ for rotated files.

Can I change the strategy settings?

Edit rules.json only; it is read at the start of every cycle. Change one thing at a time, outside market hours, and keep the account guard and daily loss stop. Backtest before loosening filters.

What's in the \$29 package?

The finished, tested source code from this build plus the full instruction guide, ready to configure and run. The build guide and every prompt are free if you prefer to build it yourself with Claude Code.

Why are stops modified in place instead of cancelled and re-placed?

Cancel-then-replace leaves a gap, however brief, where the position has no stop at the broker. Modifying the existing order (same order ID) changes its price or size without ever removing it, so the position is protected the whole time. trade.py also refuses to move a stop down.

What happens if the bot or the computer crashes while a trade is open?

The protective stop lives at IBKR, not in the bot, so it keeps working while your computer is off. When the next cycle runs it reconciles with the broker: it adopts any position the bot bought today that its state file missed, checks every position has a stop covering exactly its shares, and records any stop that filled while it was down.

How does the bot know a stop was hit?

Each cycle compares its state with IBKR's positions. If a position is gone, it looks for an execution whose permanent order ID matches that position's stop and logs a StopFilled sale with the real price and R. It never guesses by share count, which caused false stop-outs after partial sells in an earlier design.

FAQ / CONTINUED

Is my Telegram token safe in the logs?

Yes. When a send fails, the requests library puts the full URL, token included, into the error text. notify.py redacts the token (and anything shaped like bot<digits>:<key>) before writing logs/notify_errors.log. Still: never paste the token into a chat or commit .env.

How is relative volume (RVOL) calculated?

Volume traded so far today in regular hours, through the bar being checked, divided by the average volume traded by the same time of day over the previous 14 sessions. Comparing at the same time of day matters: compared with a full day's volume, almost nothing would pass at 10:05.

What hours does the bot trade?

Cycles run every 5 minutes from 10:01 to 15:56 ET on weekdays. New entries only between 10:05 and 15:30 ET; from 10:00 to 10:05 and after 15:30 it only manages open trades; at 15:51 it closes everything. On half-days (13:00 close) those times shift earlier by the same amount, and on market holidays it does nothing.

Where is the stop loss, and how is it set?

The initial stop is the day's low minus 1%, sent to IBKR as a real SELL stop order the moment the entry fills, sized to exactly the shares bought. It rests at the broker, so it works even if your computer is off. Later the bot only ever moves it up: breakeven at +1R, then under 5-minute swing lows.

If several stocks qualify, which does it buy first?

Candidates that pass all six filters are ranked by relative volume, highest first, then bought one by one until a limit is hit: 5 open positions, 5 new entries a day, the 3% daily loss stop, or a failed live safety check. Each stock is entered at most once a day.

Where do I put the Telegram token and chat ID?

In the .env file in your project folder: TELEGRAM_BOT_TOKEN= and TELEGRAM_CHAT_ID=. Get the token from @BotFather (/newbot) and the chat ID from @userinfobot, then press Start in your bot's chat. Never paste either into a chat window or commit .env.



WANT IT READY TO RUN?

SKIP THE BUILD

Everything in this guide is free. If you would rather not assemble it prompt by prompt, the finished, tested source code and this guide are packaged for \$29.

● ● ● BUILD IT YOURSELF: FREE



Every prompt, every fix, every check in this guide. About a day with Claude Code.

● ● ● READY-MADE PACKAGE: \$29



The complete source from this build, configured for an IBKR paper account, plus this guide. Available on Gumroad and Lemon Squeezy.

DISCLAIMER

Educational software for paper trading. Not investment advice, not a recommendation to buy or sell any security, and no guarantee of results. Simulated and paper results do not reflect real trading costs, slippage or risk. Trading involves risk of loss.

Workflow Factory / workflowfactory.tools. Strategy structure credit: Humbled Trader's Claude + IBKR tutorial. Interactive Brokers, IBKR, TWS, Telegram, Yahoo and Claude are trademarks of their owners; no affiliation implied.