Algorithmic Trading for Beginners

Companion Materials

Ash Cole Trading Series — Book 4

Bot Configuration Template • Pre-Deployment Checklist
Backtesting Results Template • Troubleshooting FAQ

These materials are designed to be edited. Adjust every value to match your trading account, risk tolerance, and strategy parameters.

Bot Configuration Template

risk_config.json

This is the master configuration file for your trading bot. Every risk parameter, symbol setting, and operational limit lives here. Copy this template, save it as risk_config.json in your project folder, and adjust every value to match your account and risk tolerance.

Do not use these default values without thinking about them. A 1% risk per trade on a $10,000 account means $100 at risk per position. On a $50,000 account, that same 1% means $500. The percentage is the same; the dollar impact is very different. Set each value deliberately.

The Template

```
{
    "risk_per_trade_pct": 1.0,
    "max_daily_loss_pct": 2.0,
    "max_trades_per_day": 6,
    "max_positions_per_group": 1,
    "min_account_equity": 5000,
    "max_errors_before_kill": 5,
    "max_spread_points": 5.0,
    "risk_reward_ratio": 2.0,
    "slippage_points": 1.0,
    "commission": 4.50,

    "symbols": {
        "NQ": {
            "point_value": 20.0,
            "max_size": 3,
            "min_size": 1,
            "pip_size": 0.25,
            "trail_points": 15,
            "breakeven_trigger": 20
        },
        "MNQ": {
            "point_value": 2.0,
            "max_size": 20,
            "min_size": 1,
            "pip_size": 0.25,
            "trail_points": 15,
            "breakeven_trigger": 20
        },
        "ES": {
            "point_value": 50.0,
            "max_size": 2,
            "min_size": 1,
            "pip_size": 0.25,
            "trail_points": 10,
```

```
            "breakeven_trigger": 15
        },
        "EURUSD": {
            "point_value": 1.0,
            "max_size": 2.0,
            "min_size": 0.01,
            "pip_size": 0.00001,
            "trail_points": 150,
            "breakeven_trigger": 200
        }
    },

    "telegram": {
        "bot_token": "YOUR_BOT_TOKEN",
        "chat_id": "YOUR_CHAT_ID"
    },

    "heartbeat_interval_minutes": 30,
    "log_retention_days": 30
}
```

Key-by-Key Reference

risk_per_trade_pct — The percentage of your account equity risked on each trade. Start at 0.5% to 1.0%. Never exceed 2% per trade as a beginner. This value feeds directly into the position sizing calculation in Chapter 7.

max_daily_loss_pct — The maximum percentage your account can lose in a single day before the kill switch activates. At 2%, a $10,000 account will shut down after losing $200 in one day. This is your safety net.

max_trades_per_day — The maximum number of trades the bot can take per day. Six is a reasonable starting point for ICT setups during killzones. If you are trading one symbol in one session, three to four may be more appropriate.

max_positions_per_group — Limits exposure to correlated markets. At 1, the bot holds only one US index futures position at a time (NQ or ES, not both). At 2, it can hold two. Start at 1.

min_account_equity — The absolute floor for your account balance. If equity drops below this number, the kill switch fires regardless of daily P&L. Set it to 80% of your starting balance as a hard stop.

max_errors_before_kill — How many consecutive errors the bot tolerates before shutting down. Five is conservative enough to survive a brief data hiccup but aggressive enough to stop before real damage from a persistent problem.

max_spread_points — The maximum acceptable spread before the bot skips a trade. On NQ during the New York session, spreads are typically 0.25–0.75 points. Set this to reject trades during illiquid periods when spreads widen.

risk_reward_ratio — The minimum reward-to-risk ratio for trade entry. At 2.0, the bot only takes trades where the take-profit distance is at least twice the stop-loss distance. This comes directly from the confluence scoring in Chapter 6.

slippage_points — Expected slippage per trade, used in backtesting cost calculations. One point is a reasonable starting assumption for liquid futures. Adjust based on your demo trading results.

commission — Round-trip commission per lot/contract. Check your broker's fee schedule and enter the exact value. This feeds into the backtesting cost model in Chapter 10.

symbols — Per-symbol configuration. point_value is the dollar value of one point of movement per contract (NQ = $20, MNQ = $2, ES = $50). max_size and min_size control position sizing limits. pip_size is the minimum price increment. trail_points and breakeven_trigger control the trailing stop and breakeven logic from Chapter 9 — derive these values from your backtesting, not from guesswork.

telegram — Your Telegram bot credentials for trade alerts and daily summaries. See Chapter 11 for setup instructions.

• • •

Save this file as risk_config.json in the same directory as your bot script. The load_config() function from Chapter 8 reads it at startup and validates all required keys before the bot begins trading.

Pre-Deployment Safety Checklist

Complete every item before starting your bot on a demo account

This checklist is not optional. Every item must be verified before the bot makes its first trade on a demo account. Print this page, work through each item, and check it off. If any item fails, fix it before proceeding.

☐ **1.** Demo account is funded and connected. Log in to MT5 or IBKR with your demo credentials. Verify that the account balance is visible and that market data is flowing for every symbol in your config.

☐ **2.** Config file is correct. Open risk_config.json and verify every value: risk_per_trade_pct, max_daily_loss_pct, max_trades_per_day, and all symbol-specific parameters (point_value, max_size, min_size, pip_size). A misconfigured point_value can turn a $100 risk trade into a $2,000 risk trade. IBKR users: also confirm your connection port is 4002 (IB Gateway, recommended for deployment) rather than 7497 (TWS).

☐ **3.** Logging is active. Start the bot, wait one minute, and check that the log file exists in the logs directory. Open it and verify that the startup message, config echo, and heartbeat are all present.

☐ **4.** Telegram alerts work. Before the bot enters its main loop, add a test call: send_telegram_alert("Bot startup test", tg_token, tg_chat_id). Confirm that the message arrives on your phone within 10 seconds. If it does not arrive, check your bot token and chat_id in the config. Do not use the alert() shorthand for this test — that function is only available inside the bot loop after the AlertManager is initialized.

☐ **5.** Kill switch is configured. Verify that min_account_equity and max_errors_before_kill are set to values appropriate for your demo account. On a $10,000 demo, a min_equity of $8,000 and max_errors of 5 are reasonable starting points.

☐ **6.** Kill switch test. Temporarily set max_daily_loss_pct to 0.1 in risk_config.json. Start the bot. On the first iteration, the risk manager should immediately trigger the kill switch. Confirm three things: (a) all open positions are closed, (b) the Telegram KILL SWITCH alert arrives within 30 seconds, and (c) the bot stops placing new trades. Reset max_daily_loss_pct to your intended value before proceeding.

☐ **7.** Time zone is correct. Print datetime.datetime.now(ny_tz) at bot startup and verify that it matches the current New York time. The daily reset, killzone filtering, and session detection all depend on this being correct. If running on a VPS, also run timedatectl to verify the server clock is synced via NTP.

☐ **8.** Backtest baseline is saved. Export your most recent backtest results to CSV using export_backtest(). Record the key metrics below (use the Backtesting Results Template on the next page). You will compare demo performance against this baseline.

☐ **9.** Trade log is initialized. Call setup_trade_log() and confirm that trades.csv exists with the correct column headers. This file becomes your automated trading journal.

☐ **10.** Heartbeat interval is set. During the first week, set heartbeat_interval_minutes to 30 in your config. After the first week with no issues, you can extend to 60. You need frequent confirmation that the bot is alive during initial deployment.

☐ **11.** Manual intervention plan is tested. If running on a VPS, test the SSH connection and verify you can kill the Python process remotely. If using a trading VPS on Windows, verify that you can access Remote Desktop from your phone or laptop. You must be able to stop the bot from anywhere.

• • •

Passing Criteria for Live Trading

Your bot must meet all five of these gates before you move from demo to live money:

Gate 1: Minimum 3 months on demo with no manual interventions required.

Gate 2: Minimum 100 completed trades (entries and exits) to provide a statistically meaningful sample.

Gate 3: Profit factor above 1.2 (total gross profits divided by total gross losses).

Gate 4: Maximum drawdown below 15% of starting demo equity.

Gate 5: 30 consecutive days with no system failures, no missed heartbeats, and no unexpected shutdowns.

Backtesting Results Template

Record and compare every backtest run

Use this template to record the results of every backtest you run. Keeping a written record prevents cherry-picking — you cannot claim a strategy works based on the one good backtest if the previous four were failures. Fill in one column per backtest run.

Run Information

Field	Run 1	Run 2	Run 3
Date of backtest			
Symbol(s)			
Timeframe			
Data period			
Number of bars			
Config file version			
Notes / changes since last run			

Core Metrics

Metric	Run 1	Run 2	Run 3
Total trades			
Win rate (%)			
Average R:R (actual)			
Profit factor			
Total net P&L ($)			
Max drawdown ($)			
Max drawdown (%)			
Max consecutive losses			
Average trade duration (bars)			

Validation Checks

Check	Run 1	Run 2	Run 3
Lookahead bias audit passed?			
Walk-forward test passed?			
Cost sensitivity: still profitable at High costs?			
Cross-symbol validation: consistent across symbols?			
Sample size > 100 trades?			
Equity curve: smooth or jagged? (Jagged = sharp drops and recoveries, few consistent runs — suggests luck. Smooth = small drawdowns, steady recovery — suggests edge.)			
Profit factor > 2.0? (If yes, investigate why)			

Comparison Notes

After each run, write a brief note comparing results to the previous run. What changed? Did the metric improvements make sense given the parameter adjustments? Are you seeing genuine improvement or curve-fitting?

Troubleshooting FAQ

Common errors and how to fix them

This FAQ covers the errors you are most likely to encounter while building and running your trading bot. Each entry includes the error message, its plain-English explanation, and the fix.

Connection and Setup

Error: *ModuleNotFoundError: No module named "MetaTrader5."*

What it means: Python cannot find the MetaTrader5 library. Either it is not installed, or you are running the wrong Python environment.

Fix: Run pip install MetaTrader5 in your terminal. If you are using a virtual environment, make sure it is activated first (you should see the environment name in parentheses at the start of your terminal prompt). If pip install succeeds but the error persists, your VS Code may be using a different Python interpreter — press Ctrl+Shift+P, type "Python: Select Interpreter," and choose the one inside your virtual environment.

Error: *mt5.initialize() returns False / "IPC initialize failed"*

What it means: Python cannot connect to the MetaTrader 5 terminal. The terminal is either not running, not logged in, or not configured for API access.

Fix: Open MT5 manually and log in to your demo account. Go to Tools → Options → Expert Advisors and check "Allow algorithmic trading." If MT5 is running but you still get this error, check that only one instance of MT5 is open — the Python API can only connect to one instance at a time.

Error: *ConnectionRefusedError with ib_async / "API connection failed"*

What it means: Python cannot reach Interactive Brokers' Trader Workstation (TWS) or IB Gateway.

Fix: Open TWS or IB Gateway and go to Edit → Global Configuration → API → Settings. Enable "Enable ActiveX and Socket Clients" and set the port to 7497 (TWS paper trading), 7496 (TWS live), 4002 (IB Gateway paper), or 4001 (IB Gateway live). Make sure "Read-Only API" is unchecked if you want to place trades. Restart TWS after changing settings.

Error: *Symbol not found / mt5.symbol_info() returns None*

What it means: The symbol name in your code does not match your broker's exact naming convention.

Fix: Brokers use different names for the same instrument — "EURUSD", "EURUSD.i", "EURUSDm", "EUR.USD." Open MT5's Market Watch panel (Ctrl+M),

right-click, select "Show All," and search for the correct symbol name. Copy it exactly, including any suffixes. For IBKR, use the contract details on their website to find the exact conId and exchange.

Error: *PermissionError: [Errno 13] Permission denied*

What it means: Your script is trying to write to a directory where it does not have permission, or another process has the file locked.

Fix: On Windows, this often means the CSV file is open in Excel. Close it and try again. On Linux/Mac, check the directory permissions with ls -la. If running on a VPS, make sure your bot runs under a user account that owns the logs and data directories.

• • •

Data and Timezone Issues

Error: *DataFrame is empty / "No data returned."*

What it means: Your data request returned zero rows. The market may be closed, the date range may be invalid, or the symbol may not have data for the requested timeframe.

Fix: Check three things: (1) Is the market currently open? Forex closes Friday 5 PM ET to Sunday 5 PM ET. Futures have daily maintenance windows. (2) Is your date range valid? The start date must be before the end date, and both must be in the past for historical data. (3) Does your broker provide the timeframe you requested? Some brokers do not offer M1 data for all instruments.

Error: *Killzone detections firing at the wrong times*

What it means: Your timezone conversion is incorrect. The bot thinks it is a different time than it actually is.

Fix: Add a debug print at bot startup: print(f"Current NY time: {datetime.datetime.now(ny_tz)}"). Compare this to the actual New York time. If they do not match, check your ny_tz definition and ensure pytz is installed. On a VPS, the server's system clock may be wrong — run timedatectl to verify.

Error: *KeyError: "close" or "high" / Column name mismatch*

What it means: Your DataFrame column names do not match what the detection functions expect. MT5 returns lowercase columns; CSV files may have different capitalization.

Fix: After loading data, add df.columns = df.columns.str.lower() to normalize all column names to lowercase. Verify that the required columns (open, high, low, close, tick_volume or volume) are all present.

• • •

Order Execution

Error: *Order rejected: "No money" / Insufficient margin*

What it means: Your account does not have enough free margin to open the position at the requested size.

Fix: Check your position sizing calculation. If risk_per_trade_pct is 1% but the calculated lot size requires more margin than available, the order will fail. Reduce the lot size, close existing positions to free margin, or fund the demo account with more capital. Also, check that your point_value and min_size config values are correct for the symbol.

Error: *Order fills at a different price than expected (slippage)*

What it means: The market moved between when your bot detected the setup and when the broker executed the order. This is normal.

Fix: Some slippage is unavoidable in live and demo trading. If slippage consistently exceeds 1–2 points on liquid instruments during active sessions, check your execution timing — your detection pipeline may be taking too long. Update slippage_points in your config to match the observed slippage, then rerun your backtest to confirm the strategy remains profitable.

Error: *Trade opened, but stop-loss/take-profit not set*

What it means: The order was sent without SL/TP parameters, or the SL/TP was rejected by the broker.

Fix: For MT5, verify that your OrderSendRequest includes sl and tp fields with valid prices. SL must be below the entry for buys and above the entry for sells. Some brokers enforce minimum stop distances — the stop cannot be closer than a certain number of points from the current price. For IBKR, bracket orders must be placed as an OCA group (see Chapter 7).

• • •

Risk Management and Kill Switch

Error: *Kill switch triggered unexpectedly*

What it means: Your max_daily_loss_pct or min_account_equity threshold was hit. Or max_errors_before_kill was reached due to repeated connection or data errors.

Fix: Check the log file for the exact trigger. If it was a loss limit, review whether the limit is set appropriately for your account size and strategy volatility. If it was an error count, check for connection stability issues. The kill switch is doing its job — investigate the cause before restarting.

Error: *Bot takes more trades than max_trades_per_day*

What it means: The daily trade counter may not be resetting at midnight, or the counter is not being incremented correctly.

Fix: Verify that a new DailyRiskManager instance is created at midnight New York time (the daily reset block in your bot loop). The pattern is risk_mgr = DailyRiskManager(starting_balance=account.equity, ...) — not a .reset() call. Check that current_trading_day is updating correctly and that the daily reset block is reached on each new day. Also verify that record_trade() is called for every trade, not just winning trades.

• • •

Logging and Monitoring

Error: *Telegram alerts not arriving*

What it means: The bot token or chat_id is wrong, or the Telegram API is unreachable from your VPS.

Fix: Test manually: open a browser and go to https://api.telegram.org/bot<YOUR_TOKEN>/sendMessage?chat_id=<YOUR_ID>&text=test. If you get a JSON response with "ok": true, your credentials work. If the bot is on a VPS, check that outbound HTTPS traffic is allowed (some corporate or restricted networks block Telegram).

Error: *Log files growing too large*

What it means: The bot is logging at DEBUG level in production, or clean_old_logs() is not running.

Fix: Set log level to INFO for production (DEBUG is for development only). Verify that clean_old_logs() is called daily (typically at the midnight reset). Check the log_retention_days value in your config — 30 days is a reasonable default. On Windows, log files may not be deleted if the logging handler still has them open; the clean_old_logs function handles this by catching PermissionError.

Error: *Heartbeat not firing / "Bot may be down" alert*

What it means: The bot's main loop has stalled, crashed, or is blocked waiting for a response that never arrives.

Fix: Check the log file for the last entry before the heartbeat stopped. Common causes: an unhandled exception in the main loop, a network timeout with no timeout parameter set, or the bot waiting for market data that never arrives during off-hours. Make sure all network calls have explicit timeout parameters.

• • •

Backtesting

Error: *Backtest results look too good (PF > 2.0)*

What it means: Suspiciously good results usually indicate lookahead bias, insufficient transaction costs, or curve-fitting to a specific data period.

Fix: Run audit_lookahead() to check for lookahead bias. Run cost_sensitivity_test() to see whether the results hold under realistic costs. Run walk_forward_test() to check out-of-sample performance. If the strategy only works on one specific data period, it is curve-fitted. A profit factor between 1.2 and 1.8 is realistic for a retail algo strategy. Above 2.0 warrants investigation — though confluence-rich entries during news killzones can legitimately produce higher ratios.

Error: *Walk-forward test shows much worse out-of-sample results*

What it means: Your strategy parameters are overfit to the training period. The rules work on historical data but do not generalize.

Fix: Widen your parameter ranges. If your strategy only works with a swing lookback of exactly 7 and a displacement multiplier of exactly 1.3, it is fragile. A robust strategy should work reasonably well across a range of similar parameters. Also, check that your train/test split does not accidentally separate different market regimes (trending vs ranging).

• • •

End of Companion Materials