// MODULE · PLATFORM COVERAGE

Sixteen platforms, one decision.

A field guide to where retail automation actually lives in 2026.

16	08	06	03
EQUITY / FUTURES	CRYPTO VENUES	LANGUAGES	CODE SAMPLES

NEXURAL · SAGE IDEAS LLC
PUBLISHED 2026 · PERPLEXITY COMPUTER
nexural.com/learn/automation

// contents

Contents

// 01 · how to read this pdf

Pick a platform under time pressure.

This document is a reference, not a tutorial. It exists for one moment: you have an idea you want to automate this week, and you need to choose a platform without burning a weekend reading docs. Open to the matrix, find your asset class, scan the Where it breaks column, and decide.

Three rules before you commit

1. Match the platform to the asset. A platform written for FX will fight you on equities. A platform written for retail equities will fight you on futures.

2. Pick the language you already know. Pine Script, EasyLanguage, MQL5, NinjaScript, thinkScript, ACSIL, cAlgo, Python — they look similar in slides and feel completely different at 3 a.m. when an order rejects.

3. Stress-test the Where it breaks column first. Every platform on the next pages has a known failure mode. Most retail automation horror stories are consequences of ignoring that one column.

Color encoding (consistent with the website)

- Equities / futures / general
- Crypto CEX
- Crypto DEX / wallet-only venues
- Funded / prop futures programs

// 02 · the 16-platform matrix

Sixteen platforms, side by side.

Same 16 rows shown on nexural.com/learn/automation. Source citations on the final page.

Platform	Asset focus	Language	Cost	Order API
TradingView	Stocks / FX / Crypto	Pine Script v5	$15-60/mo + alerts	Webhook POST
NinjaTrader	Futures / FX	NinjaScript (C#)	Free sim, ~$50/mo live or one-time license	Native .NET
cTrader / cAlgo	Forex / CFD	C# (cAlgo)	Free with broker	Native + REST/FIX
Sierra Chart	Futures / Stocks	ACSIL (C++)	$36-$56/mo	DTC protocol
MultiCharts	Futures / Stocks	PowerLanguage / .NET	$1,497 one-time or $97/mo	Native + IB/Rithmic
TradeStation	Stocks / Options / Futures	EasyLanguage	$0 stocks, $1.50/contract futures	REST + WebSocket
MetaTrader 4	Forex	MQL4	Free with broker	EA only (DLL bridge for REST)
MetaTrader 5	FX / CFD / Futures	MQL5	Free with broker	MetaApi / WebRequest

Strengths and failure modes — first 8

TradingView. Chart-first, huge community. Where it breaks: Cannot place orders directly — needs a webhook bridge.

NinjaTrader. Futures-grade backtest engine, low-latency. Where it breaks: Steeper C# learning curve; data feed sold separately.

cTrader / cAlgo. ECN-style execution, clean automation API. Where it breaks: Broker selection narrower than MT5.

Sierra Chart. Microsecond charting, very low latency. Where it breaks: Windows-only, dense UI, ACSIL is C++.

MultiCharts. Portfolio backtester, walk-forward built-in. Where it breaks: Windows-only; smaller community than NT8.

TradeStation. All-in-one broker + platform, full historical data. Where it breaks: Tight EasyLanguage lock-in; harder to migrate strategies.

MetaTrader 4. Largest FX ecosystem, every broker supports it. Where it breaks: Single-threaded EA model; legacy code.

MetaTrader 5. Multi-asset, multi-threaded, modern strategy tester. Where it breaks: Smaller broker pool than MT4 in some regions.

Platform	Asset focus	Language	Cost	Order API
ThinkOrSwim	Stocks / Options	thinkScript	$0 with TD/Schwab	None for orders
QuantConnect	Multi-asset	Python / C#	$8-$80/mo per node	REST + Python SDK
Interactive Brokers	Multi-asset	Python / Java / C++ / C#	Per-trade commissions	TWS API + Web API
Alpaca	Stocks / Crypto	Python / JS / Go	$0 stocks (US-only equities)	REST + WebSocket
Tradier	Stocks / Options	Python / JS	$0 + $10/mo pro	REST + WebSocket
MEXC / Bybit (perps)	Crypto perps	Python (CCXT)	Maker/taker bps	REST + WebSocket
Tradovate	Futures	Python / JS (REST)	$0.79-$1.79/contract	REST + WebSocket
Project X / TopstepX	Futures (sim funded)	Python / JS (REST)	Eval fee + per-contract	REST + WebSocket

Strengths and failure modes — second 8

ThinkOrSwim. Best-in-class options analytics, free. Where it breaks: thinkScript cannot place orders — alerts only.

QuantConnect. Cloud backtests, free institutional data, live brokerage links. Where it breaks: Vendor lock-in to LEAN; cloud minutes have cost.

Interactive Brokers. Cheapest commissions, deepest products. Where it breaks: API is famously verbose; many small footguns.

Alpaca. Modern dev experience, paper account is free. Where it breaks: US equities only; thinner instrument coverage.

Tradier. Cheap options, clean REST. Where it breaks: Smaller community than IBKR / Alpaca.

MEXC / Bybit (perps). Deep perp liquidity, low fees. Where it breaks: Region-restricted in US; KYC tier limits.

Tradovate. Modern futures broker, no platform fee. Where it breaks: Smaller historical data set; futures only.

Project X / TopstepX. Funded-trader API endpoints, modern auth. Where it breaks: Strict daily-loss / drawdown rules outside your control.

// 03 · reading the matrix

A decision tree, not a popularity contest.

Five questions, in order

01. What instrument?

Equities → TradingView, Alpaca, Tradier, IBKR, TradeStation. Options → ThinkOrSwim (charting), Tradier, IBKR (execution). Futures → NinjaTrader, Sierra Chart, MultiCharts, Tradovate, TradeStation, Project X. FX → MT5, cTrader, IBKR. Crypto → see Section 04.

02. Do you need to place real orders?

If your platform cannot — like Pine Script or thinkScript — you need a webhook bridge to an executing broker. Budget that bridge as production infrastructure, not a weekend script.

03. What language do you actually know?

Python lovers: Alpaca, QuantConnect, IBKR, Tradovate, Tradier, CCXT. C# lovers: NinjaTrader, cAlgo, QuantConnect. Pine Script: TradingView. EasyLanguage: TradeStation. MQL: MetaTrader 4/5. Skip platforms whose language you would have to learn under deadline.

04. What is your latency tolerance?

Cloud webhooks add 200-1500 ms. Native APIs add 1-50 ms. Sierra Chart, NinjaTrader, and IBKR sit at the low end. If you're trading mean-reversion intraday futures, this is a first-class concern, not an afterthought.

05. Where does your money actually live?

Some platforms only chart — execution lives at the broker. Always confirm where the order is sent and what happens when that connection drops.

// 04 · crypto venues

Eight venues, eight different KYC stories.

Crypto is a separate world. CEXes look like brokers; DEXes look like wallets. Same eight rows shown on the on-site CryptoMatrix.

Venue	Type	Products	API	KYC	Taker fee
Coinbase Advanced	CEX	Spot, perp	REST + WS	Full	0.40%
Kraken	CEX	Spot, perp, futures	REST + WS	Full	0.26-0.40%
Binance	CEX	Spot, perp, options	REST + WS + FIX	Full	0.10%
Bybit	CEX	Spot, perp, options	REST + WS	Tiered	0.055-0.10%
OKX	CEX	Spot, perp, options	REST + WS	Tiered	0.05-0.10%
CCXT	Library	100+ exchanges	Unified REST	Per venue	Per venue
Hyperliquid	DEX (L1)	Perp DEX	REST + WS	Wallet only	0.035%
dYdX v4	DEX (Cosmos)	Perp DEX	REST + WS + gRPC	Wallet only	0.05%

Where each venue breaks

Coinbase Advanced. Rate limit: 30 req/s. Where it breaks: Perp not available in many US states.

Kraken. Rate limit: 1 call/s tier. Where it breaks: Futures via separate Kraken Futures domain.

Binance. Rate limit: 1,200 weight/min. Where it breaks: Geo-blocked for US (use Binance.US, fewer products).

Bybit. Rate limit: 120 req/s. Where it breaks: US-restricted; KYC affects withdrawal limits.

OKX. Rate limit: 20 req/2s endpoint. Where it breaks: Passphrase required for trade endpoints.

CCXT. Rate limit: Per venue. Where it breaks: Each venue still has its own quirks under the hood.

Hyperliquid. Rate limit: Soft per-IP. Where it breaks: Withdraw routing depends on bridge state.

dYdX v4. Rate limit: Per-IP. Where it breaks: v4 is a Cosmos chain — RPC node selection matters.

// 05 · language quick-pick

Which language buys which platform.

Python

Runs on: Alpaca, QuantConnect, IBKR (ib_insync), Tradovate, Tradier, Project X, every crypto venue via CCXT.

Why it matters: Easiest to hire for, deepest data ecosystem (pandas / polars / numpy).

C# / .NET

Runs on: NinjaTrader (NinjaScript), cTrader (cAlgo), QuantConnect, IBKR (CSharpAPI).

Why it matters: Fast, strongly typed, native on Windows trading desktops.

Pine Script

Runs on: TradingView only.

Why it matters: Best in class for chart logic; cannot place orders by itself.

EasyLanguage

Runs on: TradeStation only.

Why it matters: Powerful but proprietary — strategies don't port off TradeStation.

MQL4 / MQL5

Runs on: MetaTrader 4 / MetaTrader 5.

Why it matters: Standard for retail FX brokers worldwide. Single-threaded EA model on MT4.

thinkScript

Runs on: ThinkOrSwim only — read/charting only.

Why it matters: Excellent for options analysis; pair with a real broker API for execution.

// 06 · sample · ctrader calgo

cTrader cAlgo (C#) — RSI bracket bot.

cAlgo is cTrader's native automation API. Robots inherit from `Robot` and react to bar/tick events. The example below reads RSI on every closed bar and submits a market order with stop-loss and take-profit in pips on extremes. Notice three production patterns:

1. A `Label` field prevents duplicate positions from the same robot.
2. Stop and take-profit are submitted as part of the order — no orphaned exits.
3. `OnStop()` flattens any open positions so a shutdown never leaves naked risk.

```
// RsiBracketBot.cs — cTrader cAlgo
using cAlgo.API;
using cAlgo.API.Indicators;

[Robot(TimeZone = TimeZones.UTC, AccessRights = AccessRights.None)]
public class RsiBracketBot : Robot
{
    [Parameter("RSI Periods", DefaultValue = 14)] public int RsiPeriods { get; set; }
    [Parameter("Lots", DefaultValue = 0.10)]      public double Lots { get; set; }
    [Parameter("SL pips", DefaultValue = 25)]     public double SL { get; set; }
    [Parameter("TP pips", DefaultValue = 50)]     public double TP { get; set; }

    private RelativeStrengthIndex _rsi;
    private const string Label = "RsiBracketBot";

    protected override void OnStart() {
        _rsi = Indicators.RelativeStrengthIndex(Bars.ClosePrices, RsiPeriods);
    }

    protected override void OnBar() {
        if (Positions.FindAll(Label, SymbolName).Length > 0) return;
        var volume = Symbol.QuantityToVolumeInUnits(Lots);
        var prev   = _rsi.Result.Last(1);
        if (prev < 30) ExecuteMarketOrder(TradeType.Buy,  SymbolName, volume, Label, SL, TP);
        else if (prev > 70) ExecuteMarketOrder(TradeType.Sell, SymbolName, volume, Label, SL, TP);
    }

    protected override void OnStop() {
        foreach (var p in Positions.FindAll(Label, SymbolName)) ClosePosition(p);
    }
}
```

// 07 · sample · ccxt crypto

CCXT (Python) — spot & perp order.

CCXT is the unified Python/JS library for over 100 crypto venues. The same code targets Coinbase Advanced, Kraken, Bybit, OKX, Binance — you change one env var. Below: a spot market-buy by quote-cost on one client, then a separate `defaultType: "swap"` client for a leveraged perpetual long with isolated margin. The two error classes (`RateLimitExceeded`, `InsufficientFunds`) are split because they need different retry strategies — back off vs. abort.

```
# ccxt_order.py
import os, ccxt
VENUE      = os.environ["VENUE"]
API_KEY    = os.environ["API_KEY"]
API_SECRET = os.environ["API_SECRET"]
PASSPHRASE = os.environ.get("API_PASSPHRASE")

exchange = getattr(ccxt, VENUE)({
    "apiKey": API_KEY, "secret": API_SECRET, "password": PASSPHRASE,
    "enableRateLimit": True,
    "options": {"defaultType": "spot"},
})

def place_spot_buy(symbol, quote_amount):
    try:
        bal = exchange.fetch_balance()
        free_quote = bal["free"].get(symbol.split("/")[1], 0)
        if free_quote < quote_amount:
            raise RuntimeError(f"Insufficient quote balance: {free_quote}")
        return exchange.create_market_buy_order(
            symbol, amount=None, params={"cost": quote_amount},
        )
    except ccxt.RateLimitExceeded:
        raise   # caller backs off
    except ccxt.InsufficientFunds as e:
        raise RuntimeError(f"Exchange rejected order: {e}")

def place_perp_long(symbol, contracts, leverage=3):
    perp = getattr(ccxt, VENUE)({
        "apiKey": API_KEY, "secret": API_SECRET, "password": PASSPHRASE,
        "enableRateLimit": True,
        "options": {"defaultType": "swap"},
    })
    perp.set_leverage(leverage, symbol, params={"marginMode": "isolated"})
    return perp.create_market_buy_order(symbol, contracts)
```

// 08 · sample · tradovate

Tradovate (Python) — futures REST + WS.

Tradovate is one of the most retail-friendly futures brokers and exposes a clean access-token REST API plus a WebSocket fill feed. Three concerns this snippet addresses on purpose:

1. Token-bearer auth with explicit `401` surfacing — tokens expire silently otherwise.
2. The `isAutomated: true` flag is required by Tradovate for bot-placed orders.
3. A self-healing `on_close` reconnect for the fill stream — every long-lived WebSocket eventually drops.

```
# tradovate_order.py
import os, time, requests
from websocket import WebSocketApp

BASE = "https://demo.tradovateapi.com/v1"
WS   = "wss://demo.tradovateapi.com/v1/websocket"

def get_access_token() -> str:
    payload = {
        "name":       os.environ["TRADOVATE_USERNAME"],
        "password":   os.environ["TRADOVATE_PASSWORD"],
        "appId":      os.environ["TRADOVATE_APP_ID"],
        "appVersion": "1.0",
        "cid":        os.environ["TRADOVATE_CID"],
        "sec":        os.environ["TRADOVATE_SECRET"],
    }
    r = requests.post(f"{BASE}/auth/accesstokenrequest", json=payload, timeout=10)
    r.raise_for_status()
    return r.json()["accessToken"]

def place_market_order(tok, account_id, symbol, qty, action="Buy"):
    body = {
        "accountSpec": os.environ["TRADOVATE_USERNAME"],
        "accountId":   account_id,
        "action":      action,
        "symbol":      symbol,
        "orderQty":    qty,
        "orderType":   "Market",
        "isAutomated": True,
    }
    r = requests.post(f"{BASE}/order/placeorder", json=body,
                      headers={"Authorization": f"Bearer {tok}"}, timeout=10)
    if r.status_code == 401: raise PermissionError("Token expired")
    r.raise_for_status()
    return r.json()

def stream_fills(tok, on_fill):
    def _open(ws):   ws.send(f"authorize\n1\n\n{tok}")
    def _msg(ws, m): on_fill(m)
    def _close(ws, code, reason):
        print(f"[WS] closed {code} {reason} — reconnecting in 3s")
        time.sleep(3); stream_fills(tok, on_fill)
```

```
    WebSocketApp(WS, on_open=_open, on_message=_msg,
                 on_close=_close).run_forever()
```

// 09 · pick your first platform

Three honest recommendations.

There is no universal best platform. There is a best fit for your asset, language, and risk tolerance. These three picks cover 90% of new automation users.

If you trade US stocks or crypto and want Python.

Pick: Alpaca (stocks) or CCXT + a CEX (crypto).

Modern REST APIs, free paper accounts, the smallest gap between idea and live trade. Pair with QuantConnect if you want a hosted backtester.

If you trade futures and want a real backtest.

Pick: NinjaTrader 8 + Rithmic data, or Tradovate for REST.

NinjaTrader gives you the most mature retail futures backtester; Tradovate gives you the cleanest REST API. Many traders pair them: NT8 to research, Tradovate to execute.

If you trade options and live in TradingView.

Pick: TradingView for charts + Tradier for execution.

TradingView's alerting and charting are unmatched. Tradier's options API is cheap and REST-clean. Build a webhook bridge in 50 lines and you're done.

// 10 · footnotes & sources

Sources & further reading.

Pricing, products, and API surface change. Verify on each provider's site before committing capital. Links below are clickable.

01	TradingView	https://www.tradingview.com/pricing/
02	NinjaTrader	https://ninjatrader.com/pricing/
03	cTrader / cAlgo	https://help.ctrader.com/ctrader-automate/
04	Sierra Chart	https://www.sierrachart.com/index.php?page=doc/Compare.php
05	MultiCharts	https://www.multicharts.com/buy/
06	TradeStation	https://www.tradestation.com/pricing/
07	MetaTrader 4	https://www.metatrader4.com/en/automated-trading/mql4-ide
08	MetaTrader 5	https://www.metatrader5.com/en/automated-trading/mql5-ide
09	ThinkOrSwim	https://www.schwab.com/trading/thinkorswim
10	QuantConnect	https://www.quantconnect.com/pricing
11	Interactive Brokers TWS API	https://www.interactivebrokers.com/campus/ibkr-api-page/twsapi-doc/
12	Alpaca	https://docs.alpaca.markets/
13	Tradier	https://documentation.tradier.com/
14	Tradovate API	https://api.tradovate.com/
15	CCXT (crypto unified API)	https://docs.ccxt.com/
16	Coinbase Advanced	https://docs.cdp.coinbase.com/advanced-trade-api/
17	Kraken	https://docs.kraken.com/api/
18	Binance	https://developers.binance.com/docs
19	Bybit	https://bybit-exchange.github.io/docs/
20	OKX	https://www.okx.com/docs-v5/en/
21	Hyperliquid	https://hyperliquid.gitbook.io/hyperliquid-docs/
22	dYdX v4	https://docs.dydx.exchange/

Continue reading → Next module: Backtesting