**Delrapport: Agentisk Tradingplattform för Råvaror**

**Författare**: Martin Björkquist

**Datum**: September 26, 2025

**Kurs**: Fördjupning i Pythonprogrammering (+projektkurs)

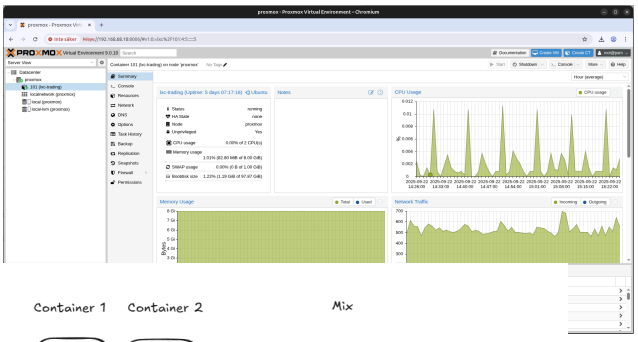
## Delrapport: Agentisk Tradingplattform för Råvaror

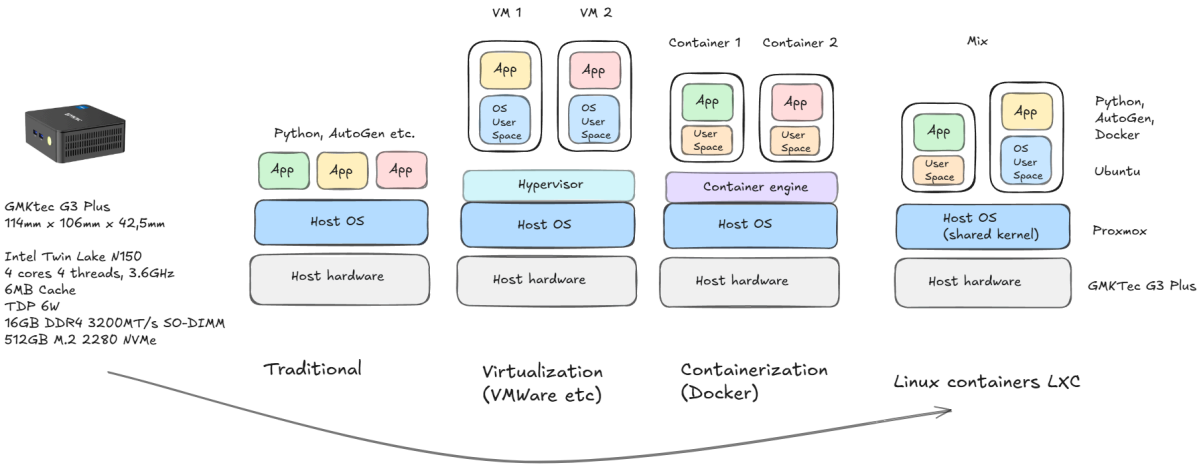
Det här blir en kort delrapport eftersom jag kommer fortsätta med projektet i nästkommande kurs. Jag redogör här för hur jag tänkt och byggt projektet och hur det hänger samman.

Jag valde det här området för att jag länge varit intresserad av råvaruhandel, speciellt kaffe futures (KC=F), som jag handlat manuellt med hjälp av teknisk analys i TradingView. De grafer och signaler som jag använder där ville jag automatisera i Python för att göra det mer skalbart, och på sikt lägga till AI för sentimentanalys och agentiska team (via AutoGen) för att optimera strategin dynamiskt och i slutändan slå min manuella handel genom bättre datahantering och konsekventa beslut. Jag ville också driftsätta plattformen lokalt på en strömsnål mini-PC som alltid kunde vara igång för tillfällen då jag själv inte har möjlighet med skärmtid.

Projektet startade med en veckas POC (kk1) för att lära mig vad som skulle krävas att bygga en ETL inom finans och börs: Hämtade data för guld och silver via API, lagrade lokalt, automatiserade och testade med lyckat utfall.
Nu till kk2, efter ytterligare fyra veckor, har jag en komplett bas för intradagshandel med kaffe: En robust ETL-pipeline, indikatorer, strategi, backtesting, visualisering, automatisering och tester. Fokus i detta steget har varit på ren kod, modularitet och tillförlitlighet – inte på att finjustera strategin ännu. Det sparar jag till nästa kurs med AI-agenter.

### Val av Teknik och Bibliotek

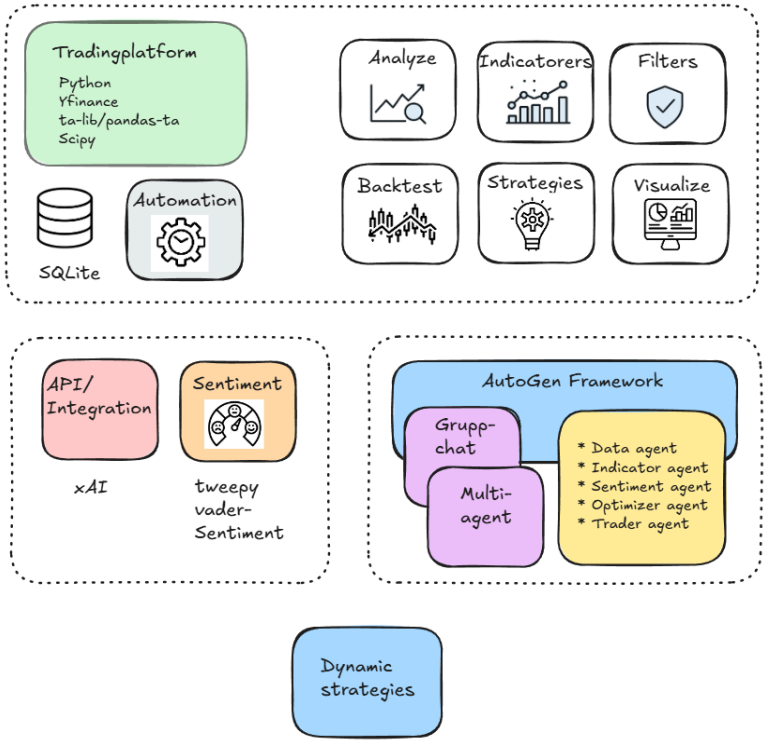





Jag installerade allt i en Ubuntu server inuti en LXC-container i Proxmox för resurseffektivitet, säkerhet och snapshot-teknik (backups), så att jag kan återställa om något går fel. Hårdvaran blev en GMKTec G3 plus som är en mycket strömsnål liten apparat, hyfsat billig men ändå med tillräckligt bra prestanda och ett omtyckt val för homelab-lösningar och Proxmox. Till koden använder jag  Python 3.12 som bas.

Biblioteken som används:

- **yfinance**: För att hämta rådata (OHLCV: Open, High, Low, Close, Volume) från Yahoo Finance. Gratis, enkelt API, och perfekt för intradag (30-minutersdata).
- **pandas och pandas_ta**: Pandas för att hantera data som tabeller (DataFrames) – rensa, sortera, beräkna. Pandas_ta lägger till tekniska indikatorer som ADX och ATR, vilket sparar tid.
- **backtrader**: För backtesting – simulera handel baserat på historisk data. Det är händelsestyrt, som en simuleringsmotor, och passar bra för att testa strategier.
- **SQLite**: Enkel databas för att lagra data lokalt. Inga externa servrar behövs, perfekt för homelab-setup.
- **CRON**: För automatisering kör jag ETL dagligen efter kaffehandelns stängningsdags via CRON i LXC-containern.
- **pytest och Pydantic**: Pytest för tester säkerställer att koden funkar. Pydantic för config-validering med type hints, vilket gör koden säkrare.
- **matplotlib och mplfinance**: För visualisering av grafer och trades. Bra för att se resultat grafiskt men fokus är inte här att skapa avancerade visuella plots.




*Översiktsdiagram över systemet, med ETL-bas och framtida AI-agenter för sentiment och optimering.*

**ETL-Pipelinen**

ETL är basen i projektet. Jag byggde det linjärt med flera moduler för att det ska vara robust och lätt att felsöka och data flödar från källa till analys utan manuella steg.

1. **Extract (data_fetch.py)**: Hämtar rådata från yfinance för KC=F (kaffe-futures). Jag använder yf.Ticker för att skapa ett objekt som frågar Yahoo Finance efter OHLCV-data över 60 dagar, 30-minutersintervall. Ingen API-nyckel behövs för

yfinance och är mycket enkelt att använda. Data returneras som pandas DataFrame. Inget fylls i för eventuella luckor (natt och helg) för att hålla det autentiskt för backtesting, vilket är ett medvetet val och regel vid hantering av börshandelsdata.

2. **Load (database.py)**: Sparar rådata i SQLite-databasen (trading.db). Tabellen ohlcv_data har kolumner för date (primärnyckel), open, high, low, close, volume och ticker. Jag använder INSERT OR IGNORE för att undvika dubletter här. Hämtning sker med SQL-frågor för datumintervall.
3. **Transform (transform.py och indicators.py)**: Bearbetar datan från DB. Först rensar clean_data: Tar bort NaN, ta bort ogiltiga rader som high=low), outliers, (>5 standardavvikelser), och sorterar. Sen lägger compute_all_indicators till indikatorer för modellering i strategin.

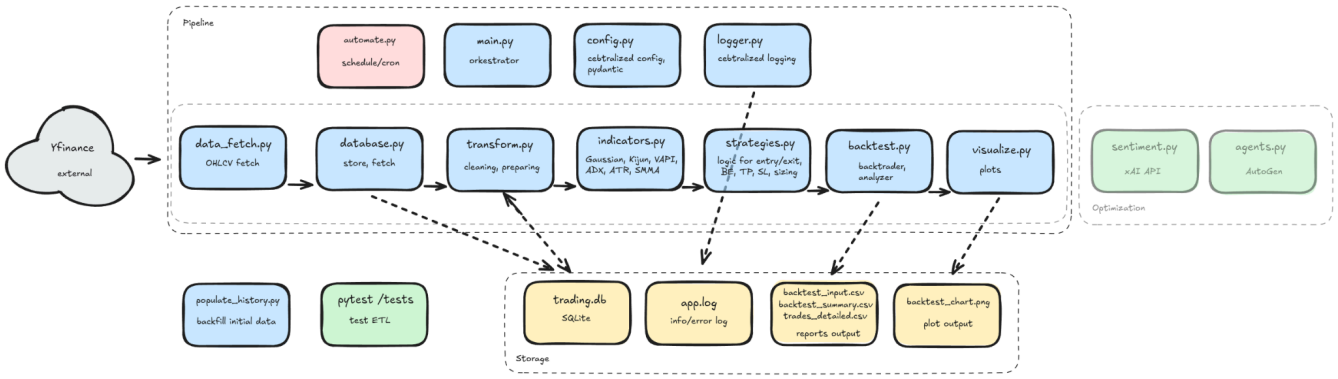



*Diagram över ETL-flödet, från hämtning till backtest.*

Allt körs i main.py: Hämta, spara, transformera, backtesta. Loggning via logger.py spårar varje steg för felsökning. Centraliserad konfig för parametrar och sökvägar i config.py.

**Indikatorer:**

Indikatorerna hjälper strategin att förutsäga rörelser baserat på historik, volym och volatilitet. Jag valde dessa för att de kompletterar varandra bra i denna typ av handel: Några är trendfokuserade, andra mäter styrka eller volym. Här är en enkel förklaring:

- **Gaussian Channel (med triple EMA)**: En kanal runt priset, som en tunnel. Mittlinjen är ett glidande medelvärde (EMA, exponentiellt vägd teknik för mjukhet). Jag använder triple EMA för extra mjukhet. Banden baseras på ATR (volatilitet).
- **Kijun-Sen**: Mittpunkten mellan högsta och lägsta pris över 125 perioder. Som ett balansvärde som hjälper att se om trenden är stark (pris över Kijun = uppåt). En enkel rolling beräkning görs ungefär som glidande medelvärde.
- **VAPI (Volume Accumulation Percentage Indicator)**: Mäter tryck bakom priset genom volym. Formel: EMA(pris * volym) / EMA(volym) och är en variant tagen från det jag använder i min nuvarande plattform Tradingview.
- **ADX (Average Directional Index)**: Mäter trendstyrka (0-100). Över 25 = stark trend. Baseras på riktad rörelse (plus/minus) och är en indikator på volatilitet.
- **ATR (Average True Range)**: Genomsnittlig prisrörelse per period som mäter kursens svängighet. Används för stopp-loss och riskhanteringen.

- **SMMA (Smoothed Moving Average)**: Ett ännu mjukare medelvärde över 200 perioder, rekursivt beräknat för att se långa trender.

Dessa beräknas i indicators.py och läggs till DataFramen i transform.py och blir signaler för beslut.

### Strategi och Backtesting

Tradingstrategin i strategies.py (GaussianKijunStrategy) integrerar sedan alla indikatorerna. T.ex. Entry när: Alla uppåt (Gaussian upp, VAPI upp, ADX>25, pris > SMMA/Kijun) = köp. Exit: Trendbrott (pris under Kijun), TP1 (sälj 40% vid 0.75x risk), breakeven (+0.4x risk), trailing (ATR*3).

| LONG Entry | Tillstånd | SHORT Entry |
|---|---|---|
| ✅ Gaussian (34) > [1] | Momentum upp | ❌ Gaussian < [1] |
| ✅ Close > Kijun (125) | Pris över equilibrium | ❌ Close < Kijun |
| ✅ VAPI (13) > [1] | Volym ackumulerar upp | ❌ VAPI < [1] |
| ✅ ADX (14) > 25 | Stark trend | ✅ ADX > 25 |
| ✅ Close > SMMA 200 | Lång trend upp | ❌ Close < SMMA |
| ❌ No-trade-zon | Gaussian ≠ VAPI direction | ❌ No-trade-zon |

Riskmodell: 0.9% per trade, max 5 trades/dag
Position: 40% Del 1 (snabb Take Profit),
60% Del 2 (trailing till ATR*3)

Backtesting i backtest.py: Använder backtrader för simulering, sparar metrics som winrate och PnL i CSV. Visualisering i visualize.py: Candlestick-grafer med indikatorer och trade-markörer (long/short, exits).

backtest_summary.csv

projects > agentic-tradingplatform > results > reports > backtest_summary.csv > data

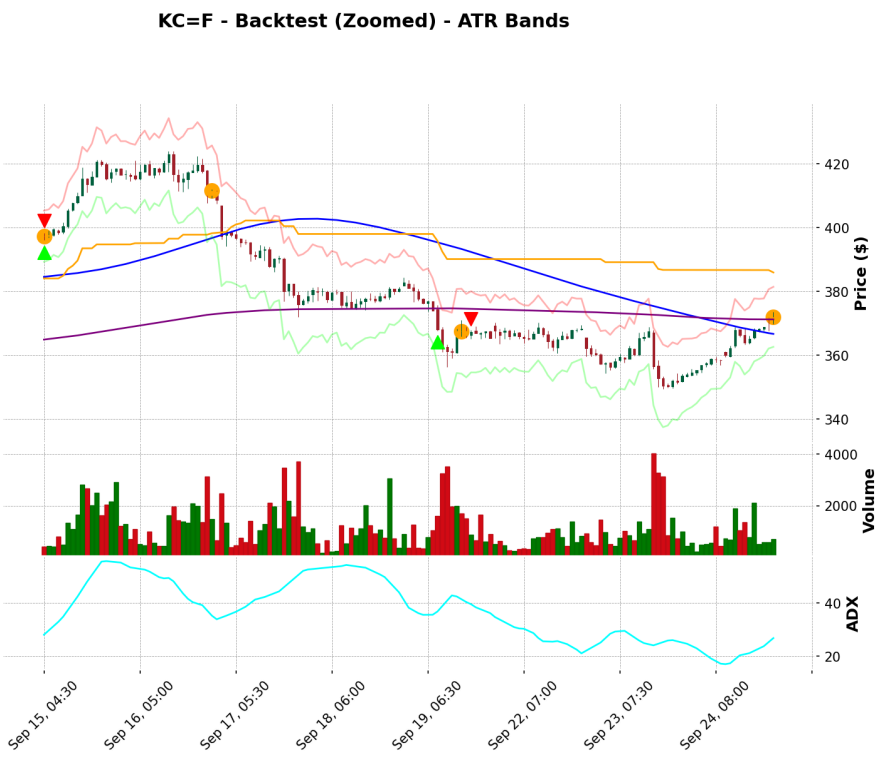
```
final_value,pnl,pnl_percent,max_drawdown_percent,total_trades,percent_profitable,profit_factor
105837.42451477051,5837.424514770508,5.8374245147705075,0.9474048172433629,10,80.0,16.066035163658 82
```

trades_detailed.csv

projects > agentic-tradingplatform > results > reports > trades_detailed.csv > data

```
trade_id,entry_date,exit_date,duration_bars,duration_hours,entry_price,exit_price,size,pnl,pnl_comm,pnl_percent,is_winner,close_reason
1,2025-08-05 12:00:00,2025-08-05 13:00:00,2,1.0,297.6000061035156,304.8500061035156,69,500.25,500.25,2.436155864015069,True,
2,2025-08-06 04:00:00,2025-08-06 04:30:00,1,0.5,290.1000061035156,289.0,69,-75.90042114257812,-75.90042114257812,-0.37918168920103806,False,
3,2025-08-06 05:00:00,2025-08-08 05:30:00,39,19.5,290.8999938964844,291.29815067964444,68,27.074661254882812,27.074661254882812,0.13687067429149685,True,
4,2025-08-08 08:00:00,2025-08-12 05:00:00,32,16.0,298.54998779296875,309.6507486371852,67,743.7509765625,743.7509765625,3.7182251877746877,True,
5,2025-08-13 10:00:00,2025-08-18 07:00:00,51,25.5,315.6000061035156,330.6936471121652,63,950.8993835449219,950.8993835449219,4.7825224070809735,True,
6,2025-08-18 08:00:00,2025-08-25 12:30:00,97,48.5,331.25,366.57000122070315,60,2119.2000732421875,2119.2000732421875,10.662641877948113,True,
7,2025-08-26 09:00:00,2025-08-28 10:30:00,41,20.5,373.54998779296875,381.50659525169516,53,421.7001953125,421.7001953125,2.1299980507926497,True,
8,2025-09-09 04:30:00,2025-09-09 09:00:00,9,4.5,389.8999938964844,383.8999938964844,51,-306.0,-306.0,-1.538856141042402,False,
9,2025-09-09 10:30:00,2025-09-16 12:30:00,99,49.5,386.04998779296875,406.05588247261795,51,1020.3006286621094,1020.3006286621094,5.182203163383593,True,
10,2025-09-19 07:30:00,2025-09-19 10:30:00,6,3.0,374.3500061035156,381.0,53,352.4496765136719,352.4496765136719,1.7764107888502376,True,
```

*Exempel på sammanställning av metrics och detaljer på trades från backtest.*




*Plot över trades tillsammans med indikatorer efter backtest.*

**Automatisering, Tester**

Automatisering i automate.py: Kör main.py dagligen kl 20:00 via CRON i LXC.

populate_historical.py fyller DB med bakfill historik.

Tester i tests/: pytest för att verifiera fetch, clean, DB och indikatorer. T.ex. mock yfinance för att testa datahämtning.

Requirements.txt visar vilka paket som krävs vid installation i Python 3.12

Mer information om mappstruktur, ETL och installation går att läsa i README på https://github.com/mabjq/ds24_agentic_tradingplatform

Docstringarna följer Google Python Style Guide för konsistens och läsbarhet, med sektioner för Args och Returns för att tydligt beskriva input/output.

**Slutsats och Tankar**

Projektet ger en stark bas för att gå vidare med agentisk AI-trading: ETL hanterar dataflödet, indikatorer och strategin/backtest validerar idéerna. Jag ser att den modulära designen bör kunna utökas med AI (AutoGen för agenter, xAI för sentiment) i nästa kurs.