Trader v1 — User Manual

Last updated: October 8, 2025

Welcome to **Trader v1**. This guide explains what the program does, how to install it, how to configure it, and how to run, monitor, and troubleshoot it. It is written so that a first-time user can get from zero to a safe paper-trading runtime.

---

1) What this program does

Trader v1 is an automated trading runtime that pulls market data and news, engineers features, trains and evaluates a lightweight model, and (optionally) places paper or live trades through Alpaca. It uses:

- A **scheduled data layer** that keeps a broad universe fresh on predictable intervals for feature engineering and training
- A **real-time stream layer** that watches a smaller focus list using Alpaca's feed for tick-by-tick updates
- A **decision engine and trade executor** that combine signals, sentiment, and safety limits to create or skip trades
- A **PostgreSQL database** where market bars, headlines, signals, positions, trades, and run metadata are stored
- Optional **Discord notifications** and light **OpenAI sentiment** on headlines

  Safety first: live trading is disabled by default. Paper trading is on in `config.yaml`. You can also force autonomy off via `AI_AUTONOMOUS_TRADING=false` in `.env`.

---

2) Repository layout

You will primarily work with these paths:

```
trader/
├─ app/                  # Python source code
│  ├─ orchestrator.py    # Main all-in-one runtime (recommended entry point)
│  ├─ run_scheduler.py   # Alternative pulse scheduler entry point
│  ├─ data_ingest.py     # Market + news ingestion utilities
│  ├─ data_stream.py     # Alpaca live stream (focus tickers)
│  ├─ decision_engine.py # Decision logic and cadence controls
│  ├─ trade_executor.py  # Order submission via Alpaca
│  ├─ model_trainer.py   # Periodic training
│  ├─ model_evaluator.py # Periodic evaluation
│  ├─ evaluate_model.py  # Manual evaluation script
│   ├─ sentiment.py       # Optional OpenAI headline scoring
│   ├─ news_fetch.py      # Headline providers (GDELT primary)
│   ├─ storage.py         # DB access layer on SQLAlchemy
│   ├─ db_utils.py        # DB engine helpers and dotenv loading
│   ├─ scheduler.py       # Simple async scheduler utilities
│   ├─ backfill_data.py   # Manual historical backfill
│   ├─ scrub_duplicates.py# Find and clean duplicates
│   └─ utils/             # Logging, Discord, cooldown helpers
├─ config.yaml            # Application scheduling, risk, and universe
├─ schema.sql             # Database schema bootstrap
├─ .env.example           # Environment variables template
├─ requirements.txt       # Python dependencies
├─ models/                # Saved models (created at runtime)
└─ logs/                  # Rotating logs (created at runtime)
```

> Tip: your zip may include a `.venv/` directory. You do not need this if you create your own virtual environment.

---

3) System requirements

- macOS, Linux, or Windows
- Python 3.10 or newer (tested with 3.13)
- PostgreSQL 14 or newer
- An Internet connection for APIs

Optional providers: - Alpaca account for data streaming and paper or live trading - Discord webhook URL for alerts - OpenAI API key for optional sentiment - Keys for alternative data vendors if you enable them later

---

4) Installation

1. **Unzip** the project so the `trader/` folder is on your machine.
2. **Create a virtual environment** inside the project:

```
cd trader
python3 -m venv .venv
source .venv/bin/activate            # Windows: .venv\Scripts\activate
```

3. **Install Python packages**:

```
pip install -r requirements.txt
```

4. **Create your environment file** from the template:

```
cp .env.example .env
```

---

5) Configure environment variables (`.env`)

Open `.env` in a text editor. The most important entries are:

Database - `PG_DSN` — full SQLAlchemy DSN. Example: `postgresql+psycopg2://postgres:trader123@localhost:5432/trader` - You can also set `PGHOST`, `PGPORT`, `PGDATABASE`, `PGUSER`, `PGPASSWORD`. The DSN takes priority when present.

Alpaca trading and data - `ALPACA_API_KEY_ID` and `ALPACA_API_SECRET_KEY` — required for trading or Alpaca data client - `ALPACA_BASE_URL` — paper default is `https://paper-api.alpaca.markets` - `ALPACA_DATA_FEED` — `IEX` or `SIP` (IEX by default)

Notifications - `DISCORD_WEBHOOK` or `DISCORD_WEBHOOK_URL` — post alerts to a channel - `DISCORD_ALERT_LEVEL` — `info`, `warn`, or `error` to control noise

Autonomy and focus - `FOCUS_TICKERS` — comma-separated list for the live stream layer (for example `AAPL,MSFT,NVDA,...`) - `AI_AUTONOMOUS_TRADING` — set `false` to keep the decision engine parked

Optional API keys - `OPENAI_API_KEY` for sentiment - Vendor keys such as `FINNHUB_API_KEY`, `NEWS_API_KEY`, `FMP_API_KEY`, `MARKETSTACK_API_KEY`, and others if you enable those sources

> If a required value is missing, the program degrades gracefully. Example: no Alpaca keys means the executor is disabled and the stream uses a fallback path where possible.

---

6) Configure application settings (`config.yaml`)

Key sections you may want to adjust:

Universe

```
universe:
  name: "SP100"
  tickers:
    - AAPL; MSFT; NVDA; TSLA; META; AMZN; GOOGL; GOOG; LLY; BRK-B
    - JNJ; UNH; V; JPM; WMT; XOM; PG; CVX; MA; HD
```

- These are processed into a list, semicolon separated within each line. Edit to fit your research set.

Focus universe

```
focus_universe:
  mode: "realtime"
  tickers: [AAPL, MSFT, NVDA, TSLA, META, AMZN, GOOGL, GOOG, LLY, BRK-B]
```

- This list drives the Alpaca live stream layer.

Scheduling (seconds between pulses)

```
scheduling:
  focus_stream: true
  focus_stream_seconds: 15
  sp100_update_seconds: 300
  news_poll_seconds: 300
  technicals_poll_seconds: 300
  fundamentals_poll_seconds: 900
  data_fetch_seconds: 300
  model_train_seconds: 1800
  model_eval_seconds: 3600
  trade_exec_seconds: 60
  heartbeat_seconds: 900
```

Risk

```
risk:
  max_position_pct_equity: 0.10
  max_daily_loss_pct: 0.03
  max_open_positions: 10
  paper_trading: true
```

Thresholds and notifications

```
thresholds:
  enabled: true
  daily_loss_limit: 0.03
  max_connection_failures: 3
  min_heartbeat_interval: 600
  auto_restart_after_minutes: 10

notifications:
  discord_enabled: true
  mention_role_id: "123456789012345678"
```

---

7) Initialize the database

Create the database once, then apply the schema.

```
# Create a database named "trader" (change if you prefer)
createdb trader

# Apply tables and indices
psql trader < schema.sql
```

Tables the app uses include: `minute_bars`, `hour_bars`, `daily_bars`, `news_headlines`, `signals`, `positions`, `trades`, `model_runs`, and `provider_payloads`.

---

8) How to run

A. All-in-one orchestrator (recommended)

Starts all loops: data ingestion, news, fundamentals, model training and evaluation, trade execution, live stream, and heartbeat.

```
source .venv/bin/activate
python -m app.orchestrator
```

You will see log lines like:

```
📊 Starting orchestrator pipeline
🔗 All subsystems initialized
   Heartbeat active (900s)
```

Stop with `Ctrl+C`.

B. Pulse scheduler (advanced alternative)

This variant registers named jobs on a small scheduler. Use it if you prefer explicit job registration and shorter runs.

```
python -m app.run_scheduler
```

---

9) Backfill and data hygiene

Backfill historical data

Populate market bars and news for a chosen period. Run this while the orchestrator is stopped.

```
python -m app.backfill_data --start 2025-09-01 --end 2025-10-08 --universe SP100
```

The script applies uniqueness constraints during inserts to avoid duplicates.

Scrub duplicates

Audit and optionally delete duplicate rows if you suspect earlier duplication from testing.

```
# Report only
python -m app.scrub_duplicates --report

# Remove duplicates (irreversible) — run only if you reviewed the report
python -m app.scrub_duplicates
```

Targets tables like `market_bars` and `news_headlines` using unique keys such as `(symbol, timestamp)` or `(ticker, source, published_at, headline_hash)`.

---

10) Models: training, storage, and evaluation

- The orchestrator periodically trains and evaluates. Models are saved into `models/` with timestamped names such as:
- `models/daily/trader_model-daily_YYYYMMDDTHHMMSS.pkl`
- `models/trader_model_YYYYMMDD_HHMM/` with metadata
- To run a manual evaluation:

  ```
  python -m app.evaluate_model
  ```

- Feature engineering lives in `app/features.py`. The default classifier is a lightweight tree-based model. XGBoost is installed for future use.

  Tip: keep older models for comparison. Do not delete the `models/` subfolders if you want evaluation history.

11) Trading execution

- Trading happens only if all of the following are true: 1) `risk.paper_trading=false` for live mode or `true` for paper mode 2) Alpaca keys are present in `.env` 3) `AI_AUTONOMOUS_TRADING=true` in `.env` and the decision engine reports confidence above the threshold in `config.yaml`
- Orders are submitted as day market orders via Alpaca. Position limits, cooldowns, and max open positions apply.
- If keys are missing, you will see a log like: `Alpaca credentials missing — trade executor disabled` and no orders are sent.

12) Logs and monitoring

- Log files are created in `logs/` for normal operation and error cases
- Heartbeat posts a periodic status and can publish summary metrics to Discord if configured
- Discord embeds are color coded: ok, warn, error, info
- To lower noise, adjust `DISCORD_ALERT_LEVEL` or toggle `notifications.discord_enabled` in `config.yaml`

13) Common operations quick reference

```
# Start the orchestrator
python -m app.orchestrator

# Start the alternative scheduler
python -m app.run_scheduler

# Manual historical backfill
python -m app.backfill_data --start YYYY-MM-DD --end YYYY-MM-DD

# Scrub duplicates (report only or delete)
python -m app.scrub_duplicates --report
python -m app.scrub_duplicates

# Manual model evaluation
python -m app.evaluate_model
```

14) Troubleshooting

Database connection fails - Make sure PostgreSQL is running. Check `PG_DSN` in `.env`. The program can fall back to an in-memory SQLite engine, but you will lose persistence when the process stops.

No OpenAI key - Sentiment scoring is disabled and headlines get a neutral score. This is fine for early testing.

Alpaca free plan or no keys - Trade executor is disabled and the live stream may not start. You can still run scheduled ingestion and model training.

Duplicate data detected - Run `python -m app.scrub_duplicates --report` to identify and review. Then run it without `--report` to clean.

Model accuracy seems low - Extend your training window, adjust features in `features.py`, or refine the universe. Keep paper trading while you iterate.

"1h needs ≥ 1 day window" warnings - Some intervals require minimum lookback windows. Backfill a longer period or relax the interval configuration.

"multiple primary keys for table \"sync_log\"" - You applied a conflicting schema alteration during tests. Re-apply `schema.sql` cleanly or drop the extra constraint.

---

15) Data schema overview (simplified)

- `minute_bars`, `hour_bars`, `daily_bars` — OHLCV market bars. Unique by `(ts, ticker)`
- `news_headlines` — ticker, headline, source, url, published_at, sentiment, and a `headline_hash` used for dedupe
- `signals` — model decisions captured for auditability
- `positions` and `trades` — current holdings and submitted orders
- `model_runs` — run summaries such as bars inserted, news inserted, signals created
- `provider_payloads` — raw JSON payloads to aid debugging and reproducibility

---

16) Safety checklist before enabling live trading

- Confirm your `.env` has valid Alpaca keys and `ALPACA_BASE_URL` for live or paper
- Leave `risk.paper_trading=true` while testing
- Verify `AI_AUTONOMOUS_TRADING=false` until you are confident
- Review Discord alerts and logs for at least a full session
- If and only if comfortable: set `risk.paper_trading=false` and `AI_AUTONOMOUS_TRADING=true`

---

17) FAQ

Can I run this without any keys? Yes. You can ingest with Yahoo Finance and run the model offline. Trading and advanced feeds will be disabled.

Where do models get saved? The `models/` folder. Daily snapshots use a timestamp in the filename.

Can I change the universe? Yes. Edit `config.yaml` under `universe` and `focus_universe`.

How do I change cadences? Edit `scheduling` in `config.yaml`. Values are in seconds.

How do I keep this running 24x7? Use a process manager such as `tmux`, `screen`, or `systemd`. On macOS or Linux you can also run inside a detached `tmux` session.

---

18) Next steps

- Tune thresholds in `config.yaml`
- Add providers if you have keys
- Expand Discord embeds or add a dashboard
- Keep everything on paper until you are confident with your workflow

If you want a one-page quick start or a printable PDF, say the word and I will generate it.