## **Macro Fusion Signal — Minimal-Viable Build**

### **0. Guiding principles**

* One embedded engine (**DuckDB**) handles storage **and** analytics — no external metastore needed. ([bauplanlabs.com](https://www.bauplanlabs.com/blog/blending-duckdb-and-iceberg-for-optimal-olap?utm_source=chatgpt.com))
* Use DuckDB’s **native Delta-Lake extension** for ACID versioning & time-travel. ([duckdb.org](https://duckdb.org/2024/06/10/delta.html?utm_source=chatgpt.com))
* First deliver **daily data + probabilities**; dashboards are optional sugar; no PDFs or model cards yet.

### **1. Set up the lake (Day 1-2)**

1. Install DuckDB + duckdb\_extension\_manager install delta.
2. Create a single Delta table mfs\_raw with partition column d\_snap (YYYY-MM-DD).
3. For each ingest run, COPY new parquet/CSV into @mfs\_raw using MERGE to avoid duplicates.
4. Add **Great Expectations** suite for nulls, ranges, schema drift (works with DuckDB driver). ([pran-kohli-1990.medium.com](https://pran-kohli-1990.medium.com/duckdb-dbt-great-expectations-awesome-data-pipelines-8b459ccd7afc?utm_source=chatgpt.com))

### **2. Ingest public feeds (Day 1-6)**

| **Feed** | **Pull time (ET)** | **CLI / lib** | **File → staging** | **Notes** |
| --- | --- | --- | --- | --- |
| Cleveland Fed CPI/PCE nowcast | 08:10 | httpx | clev\_nowcast\_{d}.csv | Free redistributable feed ([reddit.com](https://www.reddit.com/r/dataengineering/comments/1im5kgl/when_is_duckdb_and_iceberg_enough/?utm_source=chatgpt.com)) |
| UST par-yield curve | 18:05 | Treasury CSV | ust\_curve\_{d}.csv | 30-yr history ([dataengineeringcentral.substack.com](https://dataengineeringcentral.substack.com/p/delta-lake-vs-apache-iceberg-the?utm_source=chatgpt.com)) |
| NY Fed GSCPI | 09:00 (if new) | HTTP JSON | gscpi\_{m}.csv | Lag ≤10 d ([plotly.com](https://plotly.com/solutions/finance/?utm_source=chatgpt.com)) |
| Policy- & World-Uncertainty | 09:15 | CSV download | epu\_{m}.csv | Headline + categorical ([medium.com](https://medium.com/datamatiks/building-an-iot-platform-using-modern-data-stack-part-1-02c5460e3f6c?utm_source=chatgpt.com)) |
| Baltic Dry Index | 07:00 | TradingEconomics RSS | bdi\_{d}.csv | Free RSS ([pran-kohli-1990.medium.com](https://pran-kohli-1990.medium.com/duckdb-dbt-great-expectations-awesome-data-pipelines-8b459ccd7afc?utm_source=chatgpt.com), [duckdb.org](https://duckdb.org/2024/06/10/delta.html?utm_source=chatgpt.com)) |
| Commodities (WTI, Gold, CRB) | 07:05 | fredapi | commod\_{d}.csv | WTI =DCOILWTICO etc. ([github.com](https://github.com/MCG-NJU/VFIMamba?utm_source=chatgpt.com)) |
| Google-Trends keywords | 07:15 | pytrends | gtrend\_{d}.csv | GDP-nowcast keywords set ([bauplanlabs.com](https://www.bauplanlabs.com/blog/blending-duckdb-and-iceberg-for-optimal-olap?utm_source=chatgpt.com)) |

*Wrap each pull in Prefect/cron; after QC, INSERT into Delta.*

### **3. Harmonise & transform (Day 7-9)**

1. Build a **global-trading-day calendar** with pandas\_market\_calendars to align dates.
2. High-freq → daily; low-freq (GSCPI/EPU) use **state-space MIDAS interpolation** (see Xu Chen notes). ([faculty.washington.edu](https://faculty.washington.edu/chx/teaching/python/state-space-basics/?utm_source=chatgpt.com))
3. Convert each series to **rolling 5-yr percentile rank** to stabilise regimes (robust to outliers). ([youtube.com](https://www.youtube.com/watch?v=4HoWSchC4D4&utm_source=chatgpt.com))
4. Materialise a wide table mfs\_aligned in DuckDB.

### **4. Model & output (Day 10-15)**

1. **Bayesian Dynamic Factor Model** (three factors) in statsmodels → latent scores. ([youtube.com](https://www.youtube.com/watch?v=4HoWSchC4D4&utm_source=chatgpt.com))
2. Feed latest factors + lagged series into **LightGBM** (simpler than full GBM/SHAP stack) to predict:  
   * p\_cpi\_surprise\_1m, p\_cpi\_surprise\_3m, p\_cpi\_surprise\_12m
   * p\_curve\_steepen\_30d
3. Store predictions in Delta table mfs\_pred.

### **5. Validation (Day 16-18)**

* Walk-forward expanding window (2008-2025).
* Target ROC-AUC ≥ 0.70 for CPI; ≥ 0.65 for curve.
* Quick structural-break sanity: check factor loadings pre-/post-2020.

### **6. Serve (Day 19-22)**

1. **FastAPI** endpoint /mfs/latest → returns JSON from mfs\_pred ORDER BY d\_snap DESC LIMIT 1.
2. Use **DuckDB connection per request** to avoid concurrency stalls (see GitHub issue). ([github.com](https://github.com/duckdb/duckdb/discussions/13719?utm_source=chatgpt.com))
3. Simple **Plotly Dash** page (optional) with gauge + bar chart (Plotly widely used in finance). ([plotly.com](https://plotly.com/solutions/finance/?utm_source=chatgpt.com))

### **7. Final checklist (Day 23-24)**

* All feeds load & pass expectations.
* duckdb + Delta queries run in < 200 ms locally.
* FastAPI returns valid JSON via curl.
* Back-test metrics logged to /reports/metrics.csv.
* README with pip install, python ingest.py, python serve.py — **no extra docs for now**.

**Outcome:** a lean, DuckDB-powered macro signal that asset managers can hit via REST or query directly with DuckDB’s Delta reader—delivering daily probabilities without the overhead of Iceberg, heavy dashboards, or compliance docs (yet).