**1. In simple terms**

Every thirty minutes your system:

1. **Collects chatter** about Bitcoin from Reddit.
2. **Collects the latest Bitcoin price moves** from an exchange feed.
3. **Builds a quick price forecast** with a machine‑learning model.
4. **Asks an AI language model** to read the Reddit posts that were just fetched and to explain, in plain English, why the model thinks the price is likely to go up or down.
5. **Shows the forecast and the explanation** (and saves everything for later checks).

So users don’t just get “up” or “down” – they also see the top Reddit sentences that make the AI believe that call.

**2. In technical terms**

| **Stage** | **Tooling** | **What happens** |
| --- | --- | --- |
| **Ingestion** | *Producers*• Python + Reddit API (r/Bitcoin, r/CryptoCurrency) → reddit\_raw topic• Async WebSocket client (e.g. Coinbase) → btc\_ticks topic | Streams raw JSON for every new Reddit post/comment and every trade quote into Kafka. |
| **Aggregation** | *Kafka Streams / KSQL* | • 30‑minute tumbling windows.• reddit\_raw ⇒ concatenate cleaned text → reddit\_30m\_batch.• btc\_ticks ⇒ OHLC + technical indicators → btc\_30m\_ohlc. |
| **Price prediction** | *Dockerised PyTorch service* (LSTM / Temporal Fusion Transformer) | Subscribes to btc\_30m\_ohlc, outputs next‑window price + σ to model\_preds. Metrics and parameters logged in MLflow. |
| **Retrieval‑Augmented Generation (RAG)** | *Vector store* (pgvector / FAISS) + GPT‑4o (or fine‑tuned Llama‑3‑8B) behind FastAPI | 1. Embeds every Reddit batch and loads into the vector DB.2. For each prediction, retrieves top‑k similar Reddit passages.3. Prompts the LLM: “Given these posts and the model output, state whether sentiment is bullish, bearish, or neutral and explain why.”4. Publishes {direction, rationale, evidence} to rag\_outputs. |
| **Storage & monitoring** | ClickHouse or S3‑Parquet; Grafana dashboards | Persist all batches, predictions, and LLM outputs for back‑tests; live boards track MAPE, latency, token‑cost, and consumer lag. |
| **Consumers** | Streamlit/React dashboard, alert bot, or downstream trading sim | Display price curve, forecast arrow, and highlighted Reddit evidence; optionally trigger paper‑trade orders for research. |

**Key properties**

* **Latency budget:** ~2–3 s end‑to‑end (most of it is the LLM call).
* **Throughput:** a few hundred Reddit docs + thousands of price ticks per 30‑min window – trivial for MSK Serverless.
* **Cost control:** embed‑then‑filter pipeline lets you call the heavy LLM only once per window; token spend ≈ $30‑$50 per month.
* **Evaluation:** walk‑forward testing with embargo gaps prevents look‑ahead bias; LLM explanations kept faithful by returning the exact Reddit sentences alongside the summary.

This architecture demonstrates real‑time data engineering, online ML, and explainable LLM reasoning—all tied together with Kafka’s streaming backbone.