Signal Memory Engine — Ingestion, Coherence, and Storage Flow

This document describes the major pipeline stages that support the Signal Memory Engine: Ingestion, Coherence, and Storage. These processes work together to ensure signals, queries, and memories are properly normalized, persisted, and retrievable for downstream analytics.

Stage	Description
1. Ingestion	Raw inputs (documents, user queries, batch files) are ingested via the `ingestion/` modules. Scripts like `batch_loader.py` and `ingest_memories.py` handle loading structured and unstructured data. Embeddings are generated (OpenAI or HuggingFace) and inserted into Pinecone indexes. Drift monitoring (`scripts/drift_monitor.py`) can also run to periodically scan data health.
2. Coherence	After retrieval, raw vector hits are passed into `coherence/commons.py`. This stage transforms them into normalized memory events with consistent structure: id, timestamp, content, score, and metadata. It may also add flags for emotional recursion, rerouting, or compliance. This ensures heterogeneous sources are unified into a shared schema.
3. Storage	Finalized signals and events are persisted into SQLite (`storage/sqlite_store.py`). Each entry includes query, agent routing decision, biometric context, drift score, and escalate flag. Parallel JSONL trace logs (`utils/tracing.py`) maintain a lightweight audit trail. Dashboard hooks (`utils/dashboard.py`) allow real-time visualization. This dual persistence ensures both long-term durability (DB) and quick analytics/debugging (logs).