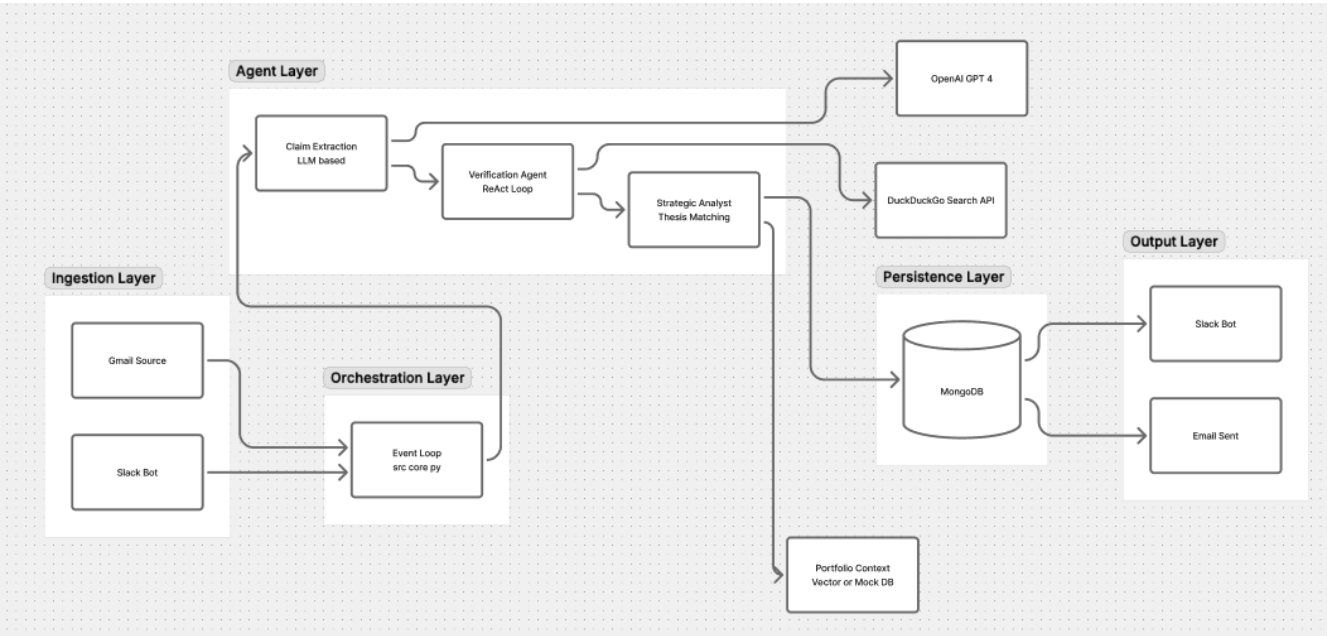
Current Prototype

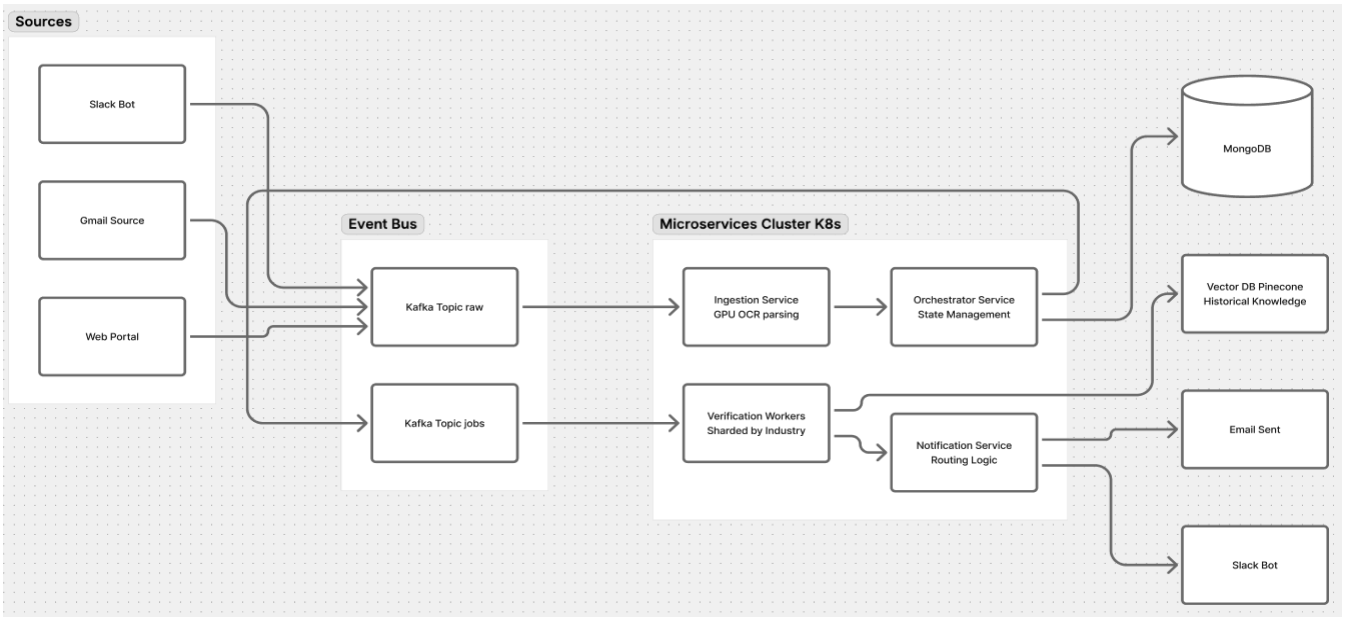


The system architecture follows a modular, event-driven pattern designed for resilience and clear separation of concerns.

Core Design Decisions: Three foundational choices drive this architecture:

1. **Agentic Reasoning over static RAG**: Investment verification requires active investigation ("Verify then Trust"), mimicking a human analyst's multi-step research process rather than simple passive retrieval.
2. **Event-Driven Ingestion**: Venture deal flow is highly asynchronous and bursty; using an event loop decouples high-volume ingestion (e.g., email floods) from high-latency verification tasks, preventing system lockup.
3. **Modular Monolith**: This structure balances rapid development velocity with strict boundary enforcement, allowing for independent scaling of components (ingestion vs. analysis) without the immediate operational overhead of a full microservices mesh.
4. **Ingestion Layer**: Asynchronous connectors (Ingestion Sources) monitor channels like Gmail and Slack, normalizing incoming pitch decks into standardized event payloads.
5. **Event Loop**: A central, non-blocking bus orchestrated by src/core.py manages the state machine, routing events to the appropriate analysis agents without tight coupling.
6. **Agent Layer**: Specialized agents execute the core logic:
    - **Claim Extraction**: Uses LLMs to parse unstructured PDF text into verifiable claims.
    - **Verification**: Performs a "Re-Act" loop (Reason-Act-Observe) using search APIs to validate claims against external data.
    - **Strategic Analysis**: Synthesizes verified data with Portfolio Context to assess thesis fit.
7. **Output Layer**: The Notification Service consumes final reports and routes them back to the user via the originating channel (e.g., proper email reply or Slack thread), completing the feedback loop.

Proposed Architecture For Scaling



The scaled architecture replaces the monolithic event loop with distributed infrastructure to handle high throughput:

1. **Decoupled Ingestion**: Sources push raw data to a **Kafka** topic (raw), allowing the ingestion layer to scale independently of processing capacity.
2. **Microservices Cluster**:
    - **Ingestion Service**: Specialized GPU-enabled workers handle OCR and document parsing.
    - **Verification Workers**: Stateless inputs/outputs allow these to be horizontally scaled on Kubernetes and sharded by vertical (e.g., distinct worker pools for BioTech vs. SaaS).
3. **Semantic Memory**: A **Vector Database (Pinecone)** acts as the long-term knowledge store, enabling the system to perform retrieval-augmented generation (RAG) against historical deal flow.
4. **Routing Logic**: A dedicated **Notification Service** subscribes to the analysis-completed topic, handling channel-specific formatting and delivery to ensure reliability.