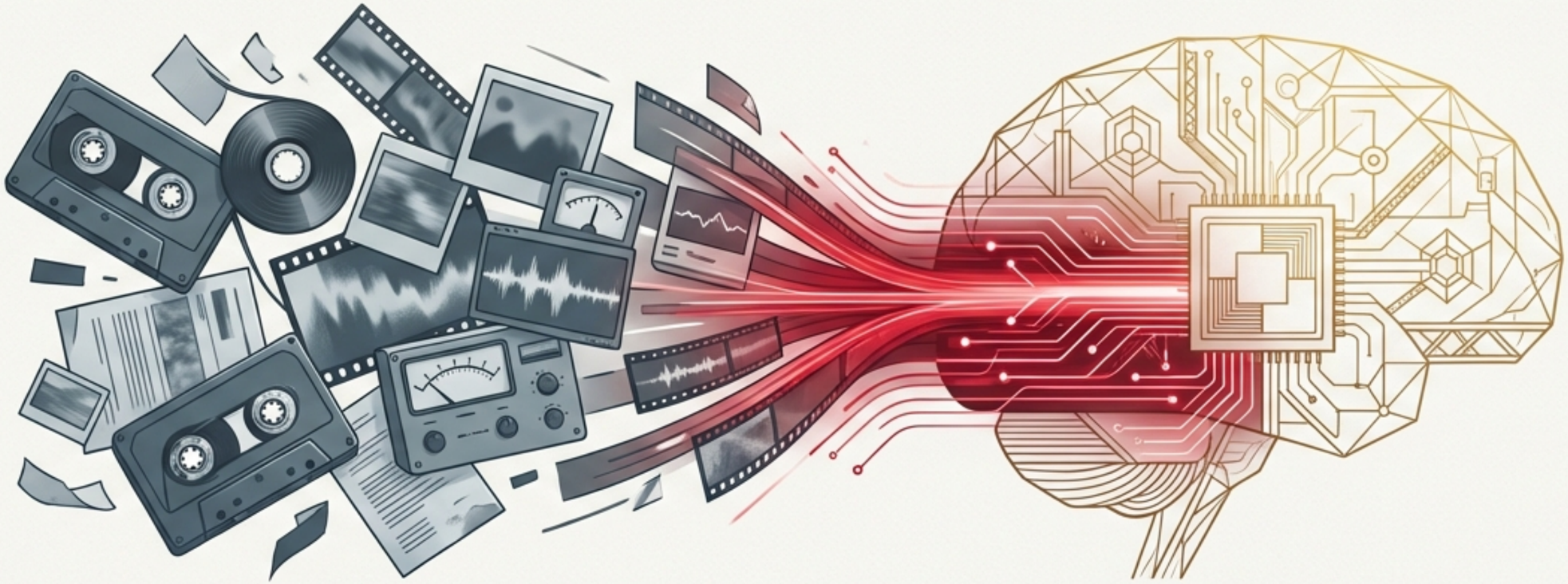
2025: The Year of the Real-Time Renaissance.



The industry didn't just stream data this year; it turned the entire physical world into an intelligent, event-driven pipeline. We've hit an inflection point where two massive forces collide:

1. **The Data Deluge**: An explosion of unstructured, real-time data from the physical world—images, sensor feeds, logs, videos, and documents.
2. **The Generative Leap**: The rise of powerful, accessible LLMs ready to understand, summarize, and act on that data.

This deck is the playbook for harnessing this collision. It's a "Wrapped" summary of one expert's prolific year building the future of real-time AI.

Meet the Headliner: Tim Spann.

The Unstructured Orchestrator

Role: Principal Developer Advocate, Senior Solutions Engineer (Snowflake, Zilliz, Cloudera).

Expertise: A prolific speaker, author, and data engineer specializing in real-time streaming, IoT, and GenAI.

Community: Runs meetups in NYC, Princeton, and Philly; DZone Big Data Zone Leader.

2025 By The Numbers

14
Conference Talks

85
Articles & Blog Posts

53
New Code Repositories

52
Issues of "All Data and AI Weekly" Newsletter

> Treat everything as a stream. Connect the physical world to the digital. Make data intelligent, in real-time.

The Backing Band: The FLaNK Stack for Real-Time Data

FLaNK is the open-source rhythm section for every project. It's a powerful, resilient pattern for moving and processing data at scale.

Apache NiFi (The Universal Connector)

The visual ETL and orchestration layer. Ingests data from hundreds of sources (REST, files, sensors, social media), handles routing, transformation, and enrichment with a drag-and-drop UI. Manages backpressure and provides full data provenance.

Apache Kafka (The Distributed Bus)

The highly scalable, fault-tolerant event log. Acts as the central nervous system, decoupling producers from consumers and ensuring data durability. (Can be swapped with Apache Pulsar).

Apache Flink (The Processing Engine)

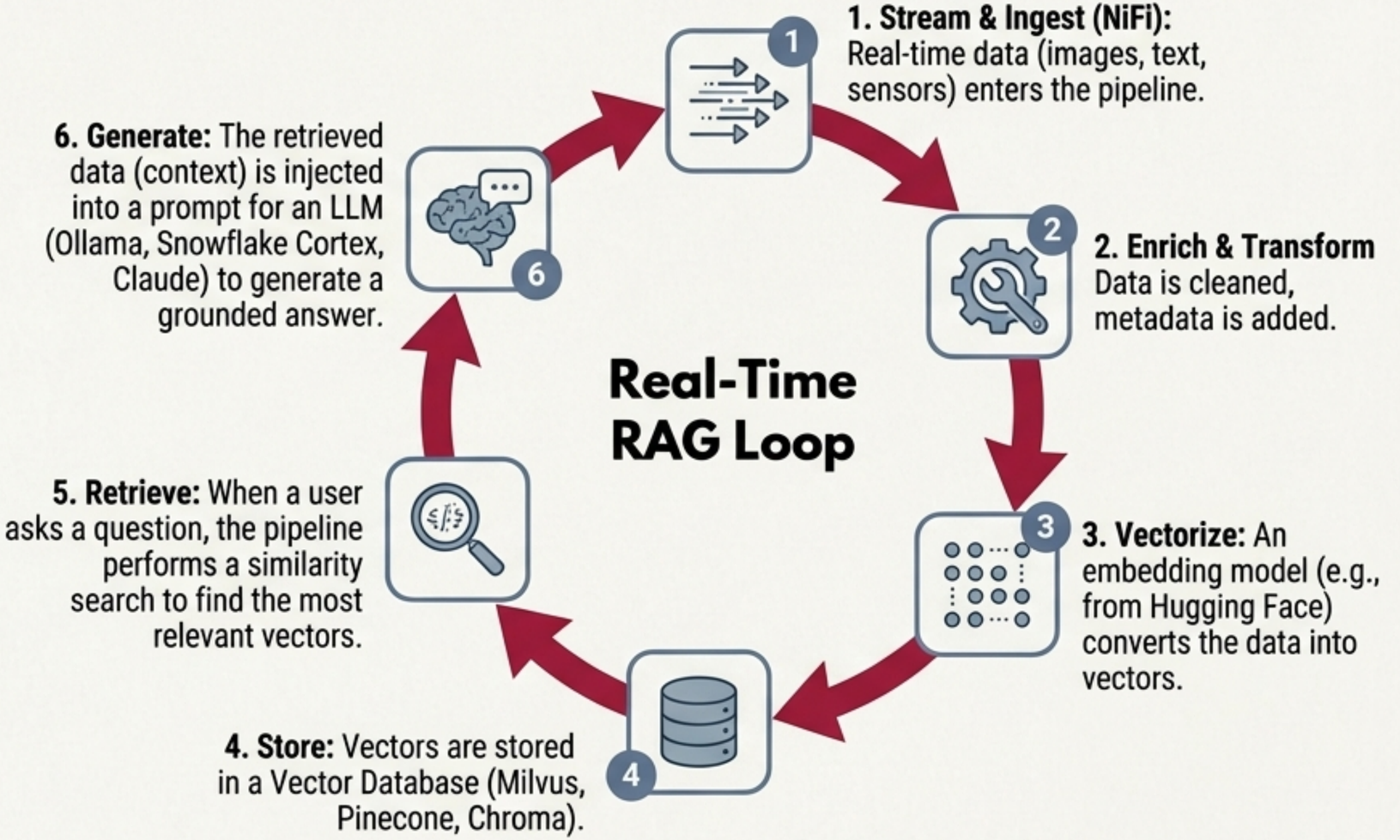
The engine for stateful computations over data streams. Enables real-time analytics, complex event processing (CEP), and continuous SQL queries on data in motion.

The Headliner: Adding a Generative AI & RAG Vocalist.

FLaNK provides the real-time data. Generative AI makes that data interactive and intelligent. Retrieval-Augmented Generation (RAG) is the bridge that connects them, grounding LLMs with fresh, external context.

Key Tech Callouts

- Ollama
- Milvus/Zilliz
- Pinecone
- Snowflake Cortex AI
- Hugging Face



1. Stream & Ingest (NiFi): Real-time data (images, text, sensors) enters the pipeline.

2. Enrich & Transform Data is cleaned, metadata is added.

3. Vectorize: An embedding model (e.g., from Hugging Face) converts the data into vectors.

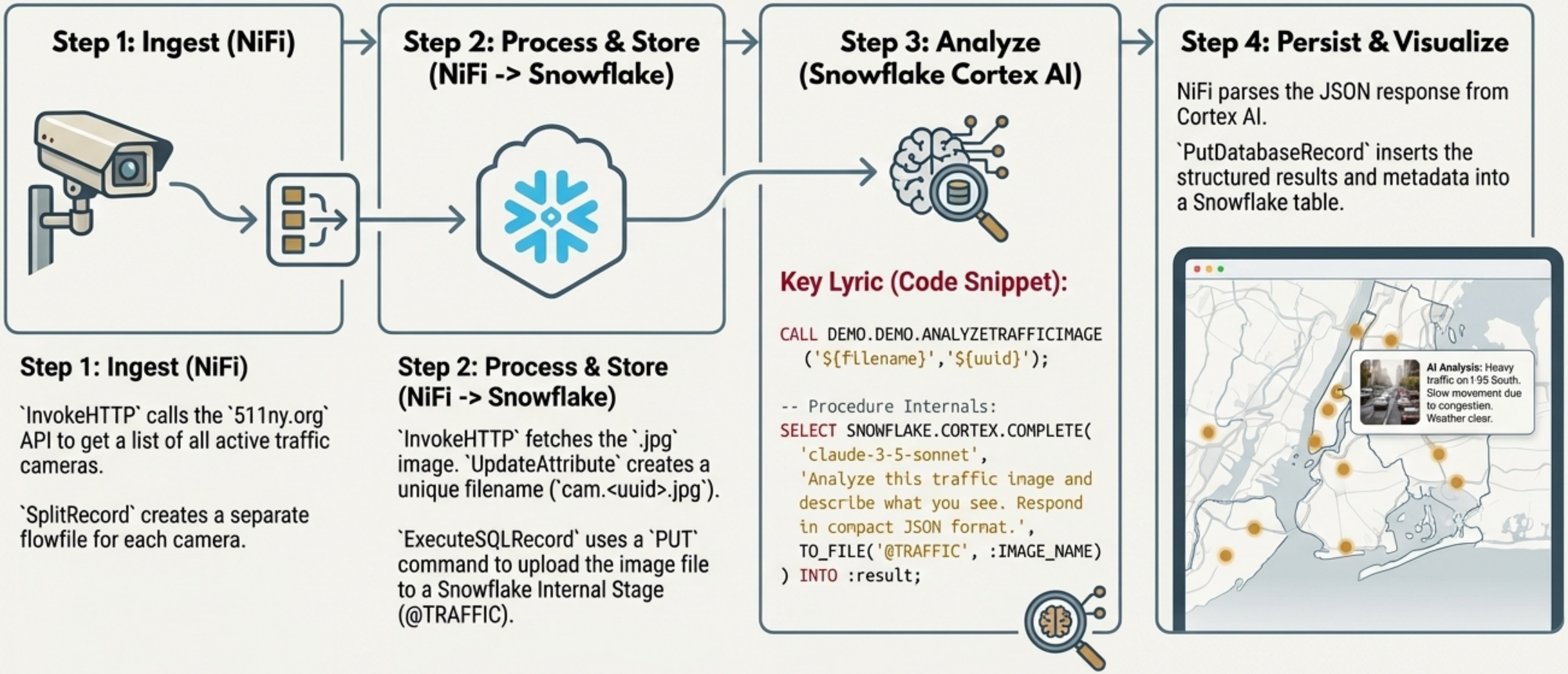
4. Store: Vectors are stored in a Vector Database (Milvus, Pinecone, Chroma).

5. Retrieve: When a user asks a question, the pipeline performs a similarity search to find the most relevant vectors.

6. Generate: The retrieved data (context) is injected into a prompt for an LLM (Ollama, Snowflake Cortex, Claude) to generate a grounded answer.

Top Track #1: "TrafficAI' - Real-Time Vision Meets Cortex AI.

Build an application to analyze live traffic camera images from NYC to assess road conditions.



Step 1: Ingest (NiFi)

`InvokeHTTP` calls the `511ny.org` API to get a list of all active traffic cameras.

`SplitRecord` creates a separate flowfile for each camera.

Step 2: Process & Store (NiFi -> Snowflake)

`InvokeHTTP` fetches the `.jpg` image. `UpdateAttribute` creates a unique filename (`cam.<uuid>.jpg`).

`ExecuteSQLRecord` uses a `PUT` command to upload the image file to a Snowflake Internal Stage (@TRAFFIC).

Top Track #2: “NiFi Man” - Should I Have Come?

At the “Community over Code” conference, Tim turned his travel decision into a real-time data problem, framed by the quote: *“We’re here – but should we have come?”*

The Goal

Use the FLaNK stack to ingest diverse, real-time data sources to logically determine if traveling to a conference is worth it.

Air Travel: Public APIs for flight arrivals and delays.

Local Transit: GTFS-RT feeds for Minneapolis's “Metro Transit” system to track buses live.

Environmental: Open APIs for Air Quality Index (AQI) readings.

Weather: NOAA feeds for forecasts and conditions.

On-Site Conditions: Live sensor data from the conference floor itself (temperature, humidity, pressure, PM2.5) streamed via phone to Kafka.

The project demonstrates how to create a personal decision-making engine by treating life as a series of event streams.

Top Track #3: From Sensor RAG to Paranormal Analytics

The same real-time RAG architecture that analyzes air quality can also track... ghosts. It proves the pattern's power in handling any form of unstructured or semi-structured data.

Use Case A: Sensor RAG for Air Quality (PyData Global)

Goal: Build a RAG application to answer questions about real-time environmental conditions.

- Python script calls the `openAQ` REST API for sensor readings.
- Data is formatted into a human-readable report string: "Air quality reading for Ozone (O3) is 26 µg/m³ in Trenton, NJ."
- The text is embedded and stored in Milvus Lite.
- A user asks, "Give me a detailed air quality report for Trenton."
- LangChain retrieves context from Milvus and uses Ollama (Llama 3.1) to generate a detailed, formatted report.

Use Case B: The Ghost Registry (DEV Community Article)

Goal: Treat ghost sightings as unstructured data events for analysis.

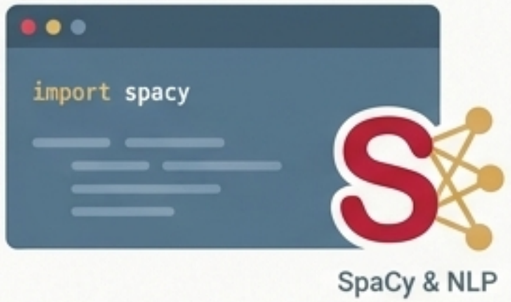
- Ghost reports (text descriptions, PDFs) are ingested.
- The data is vectorized and stored.
- A Streamlit app allows users to query the "Ghost Registry."

Memorable Example: An analysis tracking a 'Spectre Octopus' that attacks pirates in New Jersey.

If you can describe it, you can stream it, vectorize it, and build an AI application on top of it.

Writing Your Own Setlist: Custom Python Processors in NiFi 2.0

Apache NiFi 2.0 makes Python a first-class citizen, allowing you to inject custom code directly into your data flows. This opens up the entire Python ecosystem for enrichment, transformation, and analysis.



"Extract Company Names"

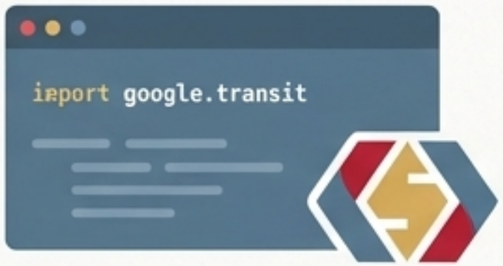
Uses SpaCy and NLP to parse text and identify organization names before calling an LLM, saving time and money. For example, it can intercept "What is the stock price for Amazon?" and route it to a stock API instead of ChatGPT.

"Address To Lat/Long"

Uses the `geopy` library to convert textual addresses into geographic coordinates, enriching data for geospatial analysis.

"Image Captioning"

Uses Salesforce BLIP and PyTorch to automatically generate descriptive captions for images flowing through the pipeline.



"Parse GTFS"

A custom processor to parse complex `google.transit` protobuf formats for real-time transit data.

You are not limited by the built-in processors.
If you can write it in Python, you can run it in NiFi.

The Real-Time AI Playbook: Your Turn.

1. 

Orchestrate with NiFi

Use NiFi as the universal entry point and control plane for all data. Its visual interface, provenance, and backpressure are critical for managing complex flows.

2. 

Ground with Real-Time RAG

Don't let LLMs operate in a vacuum. Continuously feed them fresh context from real-time streams via a vector database to get accurate, relevant answers.

3. 

Treat Everything as an Event Stream

From a traffic jam to a travel decision, model your problem domains as streams of events. This makes them compatible with modern data stacks like FLaNK.

4. 

Extend with Open Source

Leverage the vast ecosystem. Use Python to write custom logic, Ollama to run models locally, and Milvus for scalable vector search.

Call to Action

GitHub Repositories

github.com/tspannhw

Newsletter & Blog

datainmotion.dev

Social & Community

LinkedIn: in/timothyspann
Meetups: Future of Data NYC/NJ/Philly