Tutorial: It's in the Air Tonight.

Sensor Data in RAG

Real-Time AI with Open Source

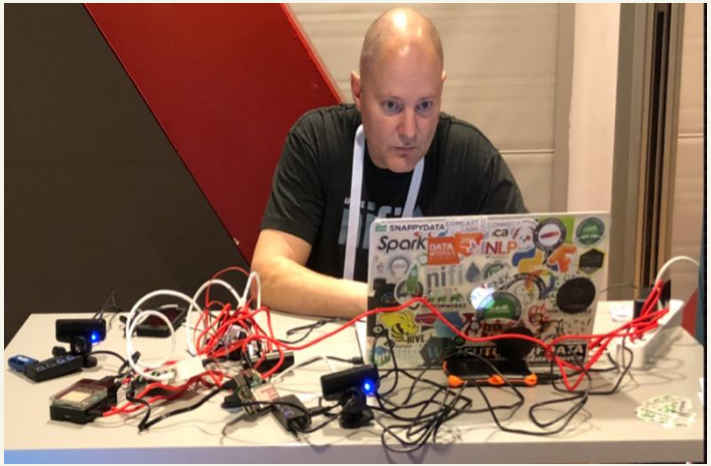
In this talk we will discuss the various ways real-time intersects with AI and some practical steps in building your own real-time AI enhanced applications. For data we will work with air quality readings and sensors.

While building it we will see the practical reasons we choose what indexes make sense, what to vectorize, how to query multiple vectors even when one is an image and one is text. We will see why we do filtering. We will then use our vector database of Air Quality readings to feed our LLM and get proper answers to Air Quality questions. I will show you how to all the steps to build a RAG application with Milvus, LangChain, Ollama, Python and Air Quality Reports. Finally after demos I will answer questions, provide the source code and additional resources including articles.

https://medium.com/@tspann/whats-in-the-air-tonight-mr-milvus-fbd42f06e482

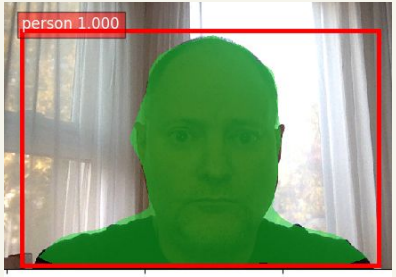
These Slides

Tim Spann

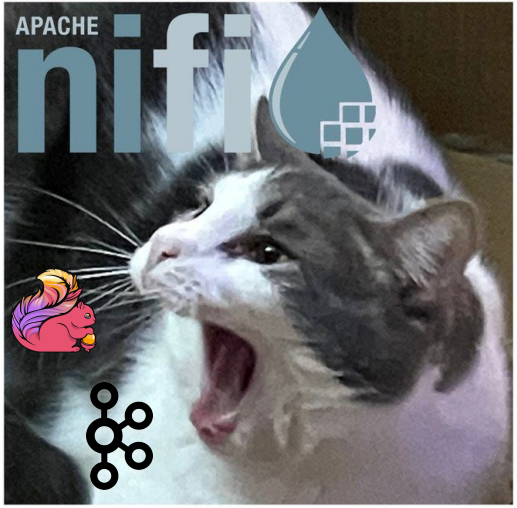


Twitter: @**PaasDev** // Blog: **datainmotion.dev**
Principal Developer Advocate / Field Engineer
NY/NJ/Philly Data + AI Meetups
ex-Zilliz, ex-Pivotal, ex-Cloudera,
ex-StreamNative, ex-PwC, ex-HPE, ex-E&Y.

https://medium.com/@tspann
https://github.com/tspannhw



AI + Streaming Weekly by Tim Spann



https://bit.ly/32dAJft
\

This week in Apache NiFi, Apache Flink, Apache Kafka, ML, AI, Streamlit, Jupyter, Apache Iceberg, Python, Java, LLM, GenAI, Vector DB and Open Source friends.

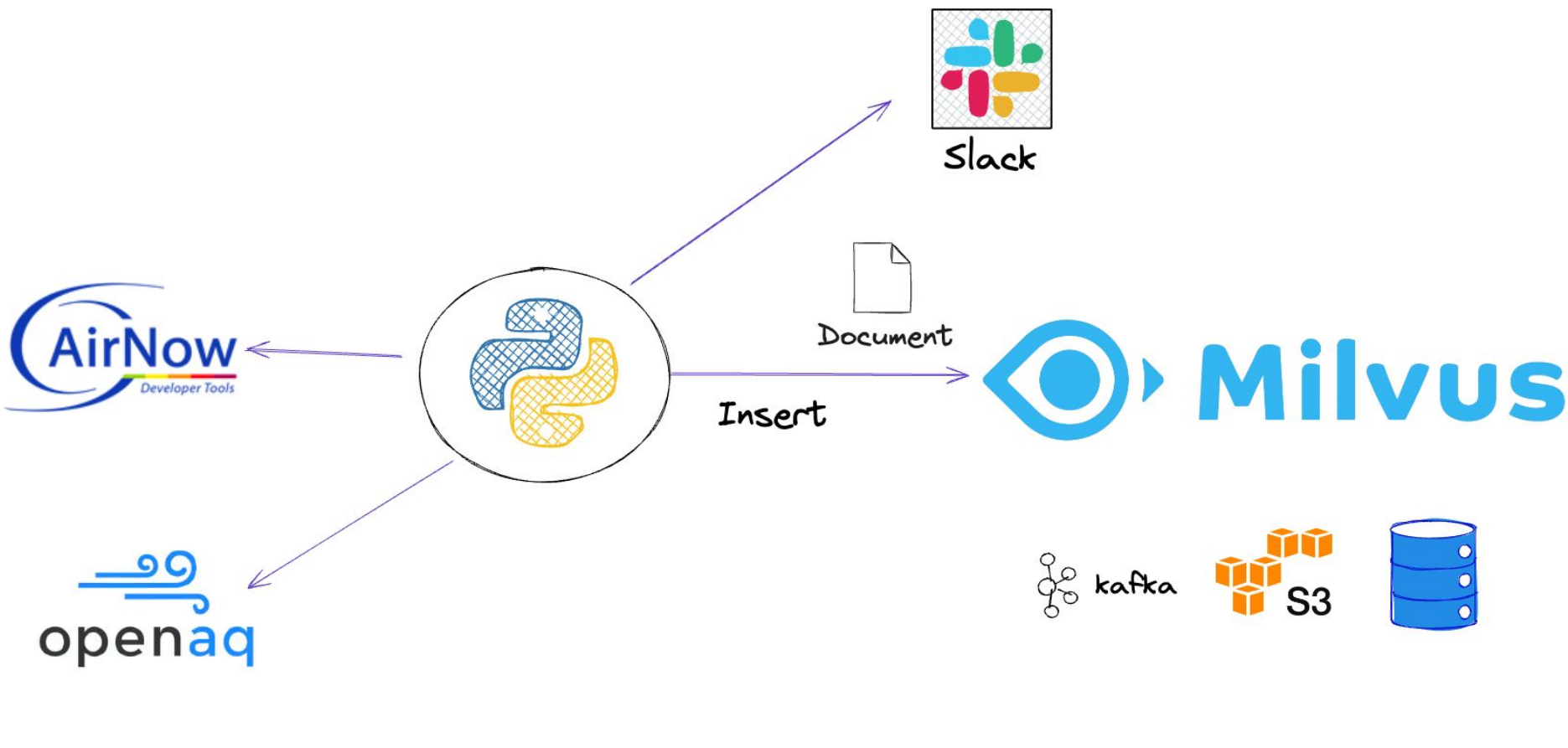
Introduction

Overview

GenAI Architecture

Walk Through Build

Q&A


Slack
AirNow
Developer Tools
Document
Insert
Milvus
openaq
kafka
S3

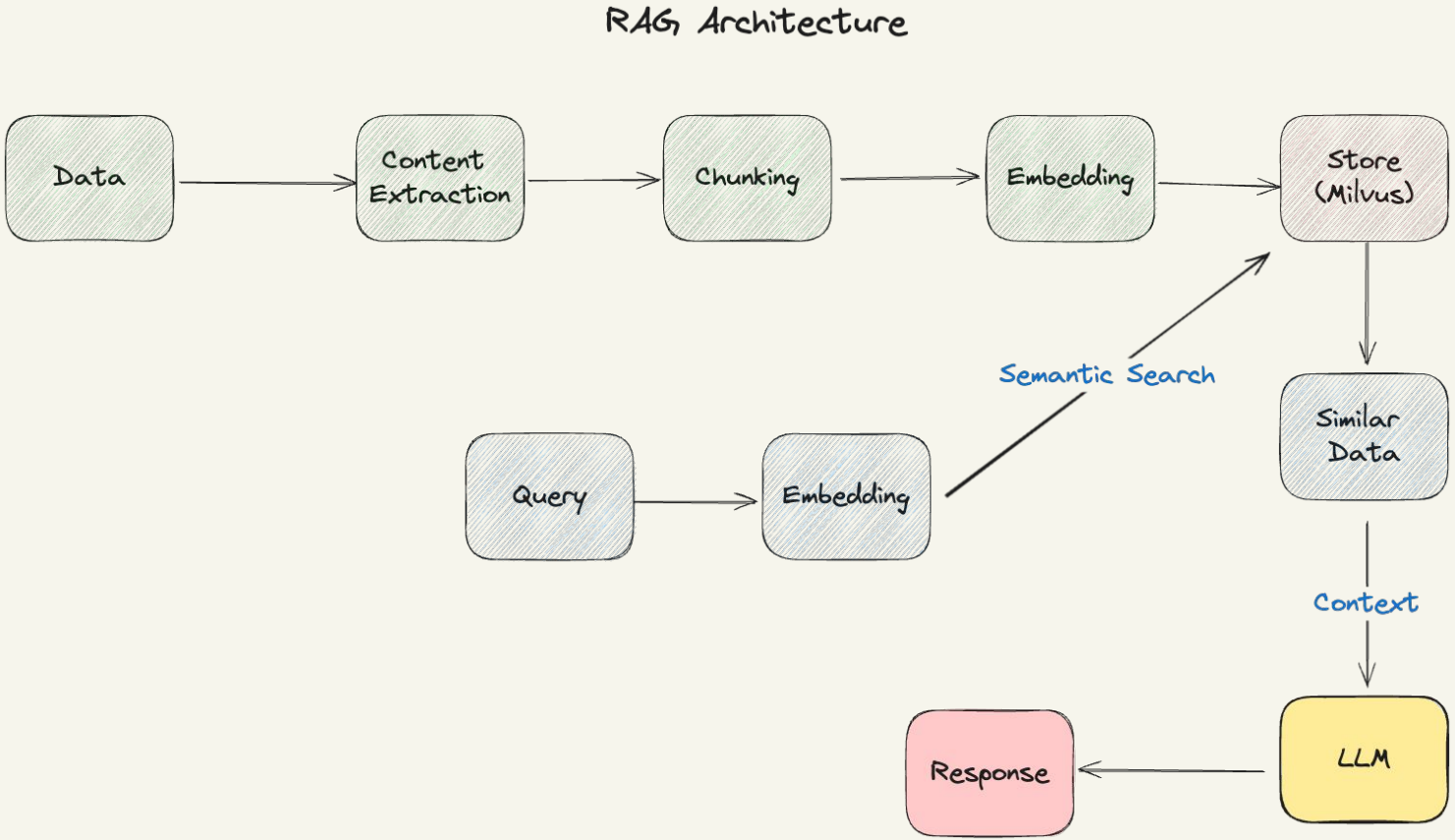
Let’s build RAG for Sensor Data

- Acquire Sensor Data
- Connect Ollama
- Install Model
- Create RAG Flow
- Run

Basic Idea

Use RAG to **force** the **LLM to work with your data** by injecting it via a vector database like **Milvus**

Basic RAG Architecture

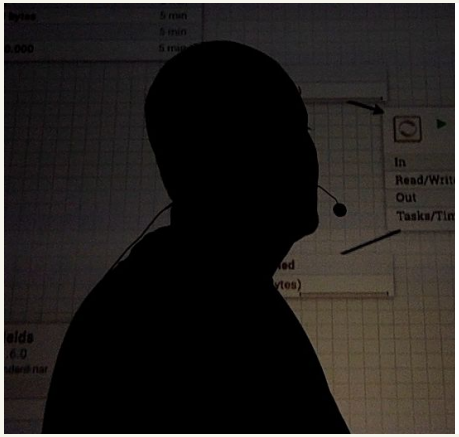


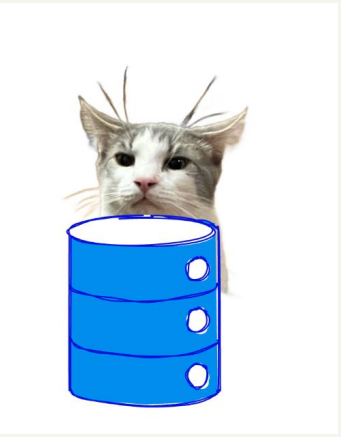
What?

openaq

- Open Data
- REST API
- Open AQ - Air Quality Data
- Stream of Data
- Location
- Time
- Sensor Readings

How?





- REST Calls
- Python
- Vector Storage
- Embeddings

When?

- Continuously
- Query on recent

Key Challengers

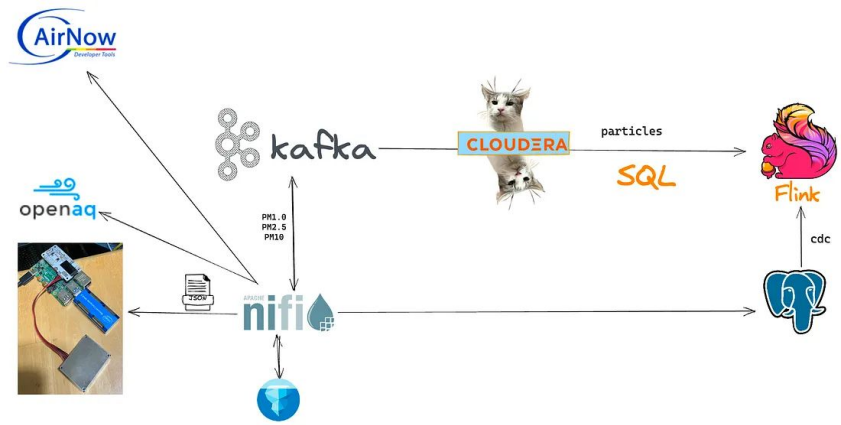
- Lots of data
- Sensor Readings
- Unstructured Text
- Geospatial Data
- Time Series
- Sparse Data
- Not part of LLM

Unstructured Data

Solve with Vector Embeddings

- Lots of formats
- Text, Documents
- Images
- Videos
- Logs
- Audio
- Email
- Multiple Languages

Continuous Ingest



- REST Feeds to Kafka
- Local Sensors to Kafka
- Apache Flink SQL via Kafka
- IBM Data Prep Kit on Ray Fast Ingest

Tiered Sensor Storage Options

- Apache Iceberg
- Redis
- Apache Pinot
- Object Storage
  - MiNio
  - S3
  - ADLSv2
  - GCS

Ingest For Today


```
TODAYS_DATE = str(
datetime.today().strftime('%Y-%m-%d') )

YESTERDAYS_DATE = (datetime.now() -
timedelta(1)).strftime('%Y-%m-%d')

url =
'https://api.openaq.org/v2/measurements?country=U
S&date_from={0}&date_to={1}&limit=1000&page={2}
&offset=0&sort=desc&radius=1000&order_by=dateti
me'.format(str(YESTERDAYS_DATE),
str(TODAYS_DATE), str(aqpage) )

headers = {"accept": "application/json", "x-api-key":
str(API_KEY)}
```

Ingest into Vectors

- RedisVL
- Milvus
- Snowflake
- Apache Pinot
- PgVector

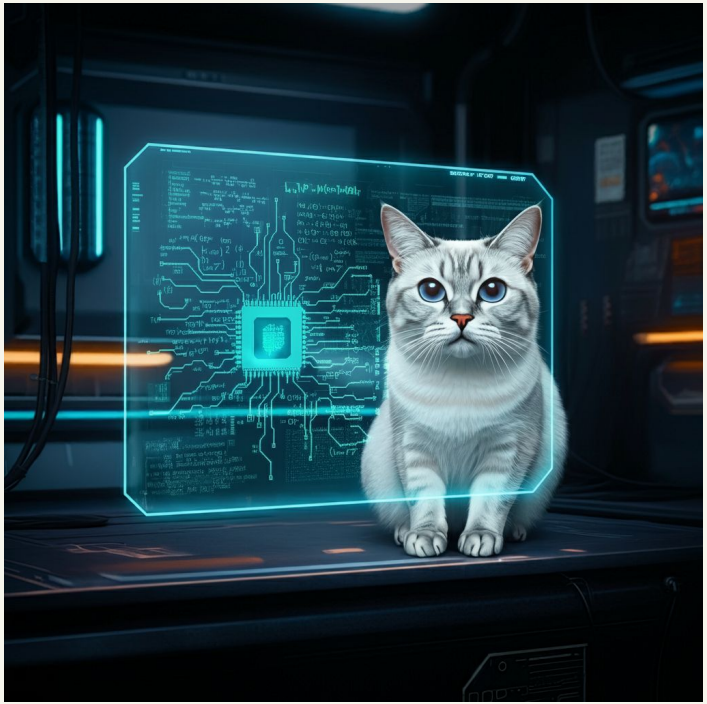
Schema - Sensor Value

```
schema.add_field(field_name='parameter',
datatype=DataType.VARCHAR, max_length=255)
schema.add_field(field_name="value",
datatype=DataType.FLOAT)
schema.add_field(field_name='datelocal',
datatype=DataType.VARCHAR, max_length=255)
schema.add_field(field_name="unit",
datatype=DataType.VARCHAR, max_length=255)
schema.add_field(field_name="isAnalysis",
datatype=DataType.VARCHAR, max_length=12)
schema.add_field(field_name='entity',
datatype=DataType.VARCHAR, max_length=255)
schema.add_field(field_name='sensorType',
datatype=DataType.VARCHAR, max_length=255)
```

Schema - Geo / Location


```
schema.add_field(field_name='locationId', datatype=DataType.INT32)

schema.add_field(field_name='location',
datatype=DataType.VARCHAR, max_length=255)

schema.add_field(field_name="latitude", datatype=DataType.FLOAT)

schema.add_field(field_name="longitude", datatype=DataType.FLOAT)

schema.add_field(field_name="country",
datatype=DataType.VARCHAR, max_length=255)

schema.add_field(field_name="city", datatype=DataType.VARCHAR,
max_length=255)

schema.add_field(field_name="isMobile",
datatype=DataType.VARCHAR, max_length=12)
```

Schema - Vectors, Large Strings & ID

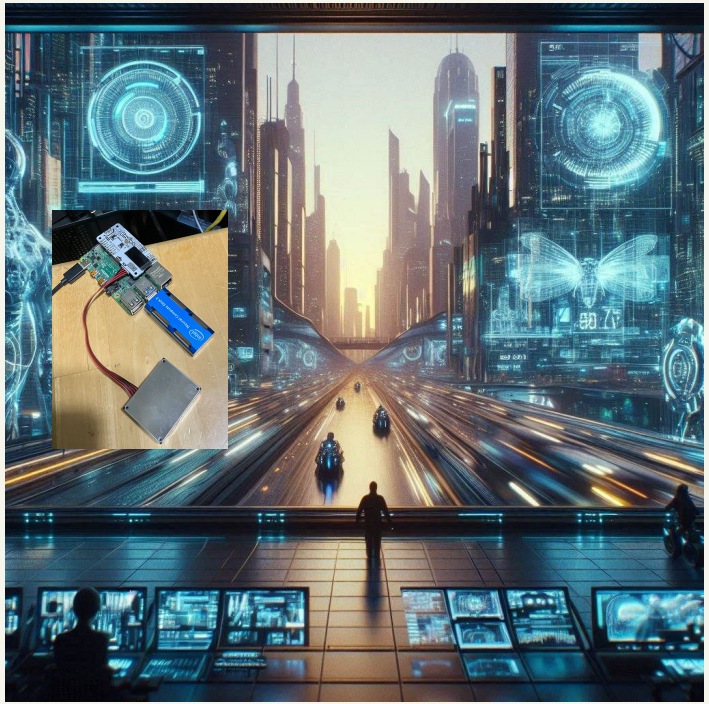


```
schema.add_field(field_name='id', datatype=DataType.INT64,
is_primary=True, auto_id=True)

schema.add_field(field_name="vector",
datatype=DataType.FLOAT_VECTOR, dim=DIMENSION)

schema.add_field(field_name="details",
datatype=DataType.VARCHAR, max_length=8000)

schema.add_field(field_name="fulllocation",
datatype=DataType.VARCHAR, max_length=2000)
```

Index

```
    field_name="vector",
    index_type="IVF_FLAT",
    metric_type="L2",
    params={"nlist": 100}
```

Collection – “openaqmeasurements”



- vector IVF_FLAT (384)
- id - STL_SORT

Similarity Search

```
results = milvus_client.search(“aq”,
        data=[model(details)],
        filter='location like "%H%"',
        output_fields=["location", "details",
    "parameter","value", "datelocal","fulllocation"])
```

Horicon Wildlife Are Current Air Quality Reading for co is 0.2 ppm for Location 1285: Horicon Wildlife Are, None, US @ 43.46611000000001,-88.621109 at 2024-11-17T20:00:00-06:00. Location 1285: Horicon Wildlife Are, None, US @ 43.46611000000001,-88.621109

Non Vector Query

```
res = milvus_client.query(
    collection_name=”aq”,
    filter='location like "%Tr%" && parameter ==
            "pm25"',
    group_by_field="location",
    output_fields=["location", "parameter","value"],
    limit=250
)

for result in res:
    print(result)
```

LangChain for RAG

```
embeddings =
HuggingFaceEmbeddings(model_name=
        "all-MiniLM-L6-v2")

vector_store = Milvus(
    embedding_function=embeddings,
    collection_name=”aq”,
    primary_field = "id",
    text_field="details",
    connection_args={"uri": MILVUS_URL},
)
```

Ollama + LLama 3.1


```
llm = Ollama(
        model="llama3.1",

callback_manager=CallbackManager([StreamingStdOut
CallbackHandler()]),
        stop=["<|eot_id|>"],
    )
query = input("\nQuery: ")
    qa_chain = RetrievalQA.from_chain_type(
        llm, retriever=vector_store.as_retriever(collection
=  “aq”))
result = qa_chain.invoke({"query": query})
```

RAG Results to Slack

Tim Spann Not Bot SparkDev 8:33 PM

Give me a detailed air quality report for Perch River

Based on the provided context, I can generate a detailed air quality report for Perch River. Here's the report:

Location: Perch River, None, US @ 44.0878,-75.9744

Sensor Type: Reference grade (Governmental Organization)

Air Quality Parameters:

* **Ozone (o3):**
  + Current reading: 0.03 ppm at 2024-11-18T14:00:00-05:00
  + Historical readings:
    - 0.018 ppm at 2024-11-18T05:00:00-05:00
    - 0.02 ppm at 2024-11-18T02:00:00-05:00
    - 0.016 ppm at 2024-11-18T08:00:00-05:00

Note that the report only provides data for ozone (o3) and does not include information on other air quality parameters such as particulate matter (PM2.5), nitrogen dioxide (NO2), etc.

Please note that this report is based solely on the provided context and may not reflect the actual current air quality conditions, as it is subject to change based on various environmental factors.

/tspannhw/AIM-BecomingAnAIEngineer: AIM - Becoming An AI Engineer

https://bit.ly/3BV4IKX

What’s in the Air Tonight Mr. Milvus?

https://bit.ly/4fQhBog

Choose a Language

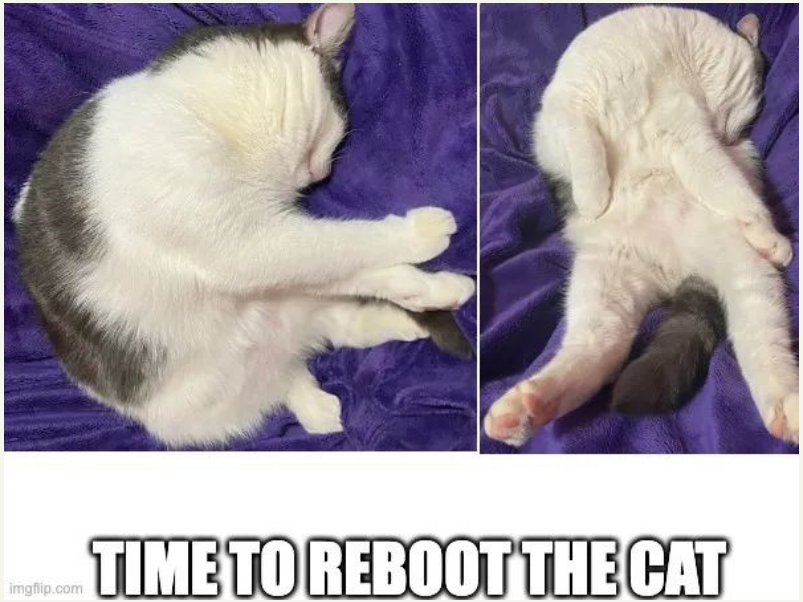
Python ←Default. Today

Java (Spring, Langchain4J, ...)

Choose a Framework

- LangChain <- Pick
- LlamaIndex
- IBM - Data-prep-kit
- NVIDIA NIM

Start a Jupyter Lab



Python SDK Connect to Milvus

From:

> *client = MilvusClient(“db/milvus_demo.db”)*

To:

> *client = MilvusClient(uri=”http://server:19530")*

Or:

> *client = MilvusClient(uri=”https://server12345.serverless.gcp-us-west1.cloud.zilliz.com", token=”tokenX”)*

Simple Retrieval-Augmented Generation (RAG) with LangChain

Summary

By the end of this application, you'll have a comprehensive understanding of using Milvus, data ingest object semi-structured and unstructured data, and using Open Source models to build a robust and efficient data retrieval system.

🐍 AIM Stack - Easy Local Free Open Source RAG

- Ollama / Llama 3.2 / Milvus-Lite / LangChain
- Python / Jupyter Notebooks

🔥 Install Ollama

🛩️ Download for Mac, Linux, Windows

https://ollama.com/download

👽 Install Open Source Meta Llama Model

https://ollama.com/library/llama3.2:3b-instruct-fp16

ollama run llama3.2:3b-instruct-fp16

>>> /bye

Running the model will download many gigabytes of model and weights for you. When it is complete it will put you interactive chat mode.
You can test it, or type */bye* to exit.

🤷 List all of your models

ollama list

NAME	ID	SIZE	MODIFIED
llava:7b	8dd30f6b0cb1	4.7 GB	40 hours ago
mistral-nemo:latest	994f3b8b7801	7.1 GB	9 days ago
gemma2:2b	8ccf136fdd52	1.6 GB	10 days ago
nomic-embed-text:latest	0a109f422b47	274 MB	10 days ago
llama3.2:3b-instruct-fp16	195a8c01d91e	6.4 GB	2 weeks ago
llama3.2:latest	a80c4f17acd5	2.0 GB	2 weeks ago
reader-lm:latest	33da2b9e0afe	934 MB	3 weeks ago

📊 Let's use it

🦙 First, let's import all the libraries we will need

```
import os
from pymilvus import MilvusClient
from langchain_community.document_loaders import PyPDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.chains import RetrievalQA
from langchain_milvus import Milvus
from langchain_community.embeddings import HuggingFaceEmbeddings
from langchain.callbacks.manager import CallbackManager
from langchain.callbacks.streaming_stdout import StreamingStdOutCallbackHandler
from langchain import hub

#### Constant For PDF Downloads
path_pdfs = "talks/"

#### Initialize Our Documents
documents = []
```

🐦 Download some PDFs of talks

1. Build a directory for the talks
2. Download PDFs

Note:

You can use your own PDFs, download more from

- https://github.com/tspannhw/SpeakerProfile
- https://www.slideshare.net/bunkertor/presentations

Iterate through PDFs and load into documents

```
for file in os.listdir(path_pdfs):
    if file.endswith(".pdf"):
        pdf_path = os.path.join(path_pdfs, file)
        print(pdf_path)
        loader = PyPDFLoader(pdf_path)
        documents.extend(loader.load())
```

Connect to Milvus

Use **Milvus-Lite** for local database

This is ./milvusrag101.db

```
client = MilvusClient(uri= "./rag101.db")
if client.has_collection("LangChainCollection"):
    print("Collection exists")
else:
    client.drop_collection("LangChainCollection")
```

🐍 AIM Stack - Easy Local Free Open Source RAG

Choose Your Model -->

Free, Hugging Face Hosted, Open Source

```
embeddings = HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2")
```

Next step, chunk up our big documents

```
text_splitter = RecursiveCharacterTextSplitter(chunk_size=500, chunk_overlap=100)

all_splits = text_splitter.split_documents(documents)
```

Choose Your Embedding Function

Embedding Function	Type	API or Open-sourced
openai	Dense	API
sentence-transformer	Dense	Open-sourced
bm25	Sparse	Open-sourced
Splade	Sparse	Open-sourced
bge-m3	Hybrid	Open-sourced
voyageai	Dense	API
jina	Dense	API
cohere	Dense	API
Instructor	Dense	Open-sourced
Mistral AI	Dense	API
Nomic	Dense	API
mGTE	Hybrid	Open-sourced

🐍Load Text embedding model via HuggingFace

Then we load all of the splits and embeddings to Milvus

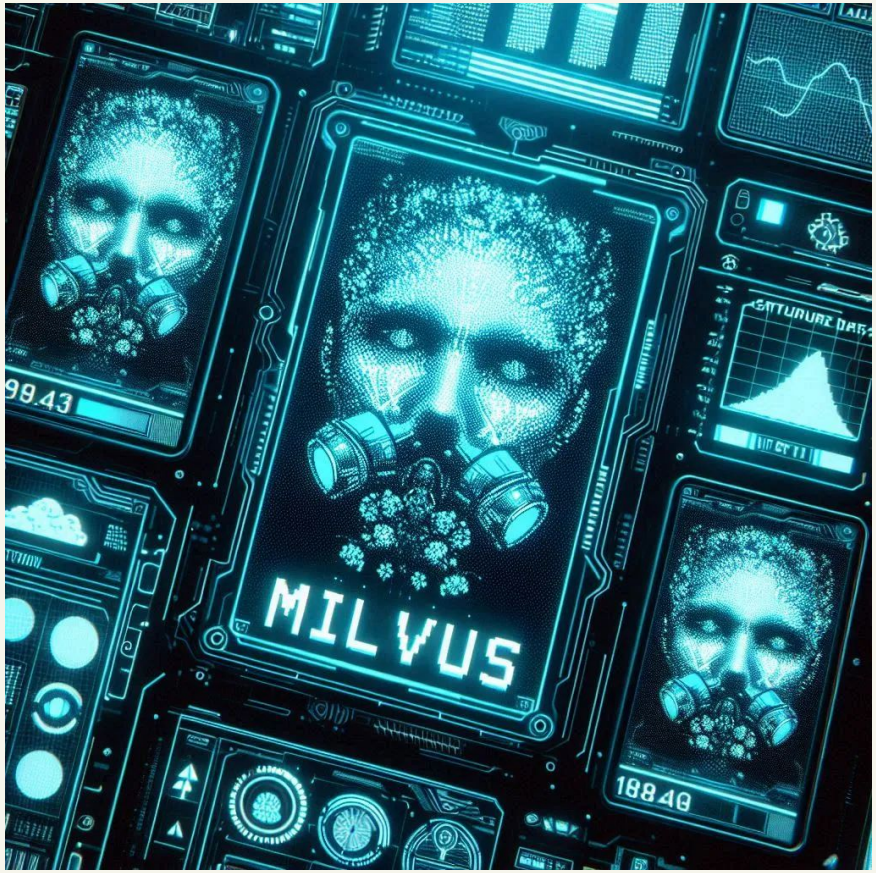
Verify Documents are Loaded

```
from langchain_huggingface import HuggingFaceEmbeddings
model_kwargs = {"device": "cpu", "trust_remote_code": True}

embeddings =
HuggingFaceEmbeddings(model_name="all-MiniLM-L6-v2",
model_kwargs=model_kwargs)

vectorstore = Milvus.from_documents(
    documents=documents,
    embedding=embeddings,
    connection_args={
        "uri": MILVUS_URL,
    },
    drop_old=False,
)
```

```
from langchain_ollama import OllamaLLM

def run_query() -> None:
    llm = OllamaLLM(
        model="llama3.2:3b-instruct-fp16",
callback_manager=CallbackManager([StreamingStdOutCallbackHandler
()]), stop=["<|eot_id|>"],)

    query = input("\nQuery: ")
    prompt = hub.pull("rlm/rag-prompt")

    qa_chain = RetrievalQA.from_chain_type(
        llm, retriever=vectorstore.as_retriever(),
chain_type_kwargs={"prompt": prompt}
    )
```

Loop Your Prompt

```
if __name__ == "__main__":
    while True:
        run_query()
```



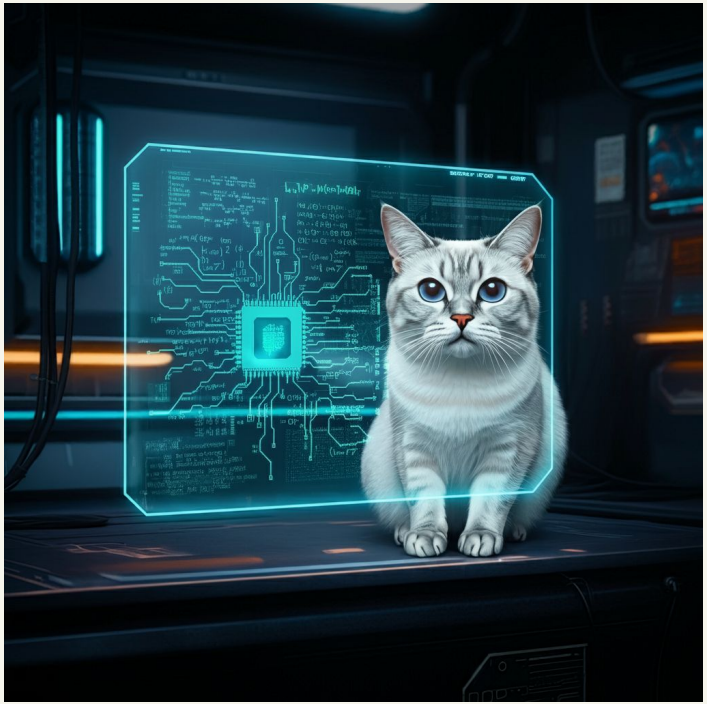

SPIRIT
Business
Cat
ADULT
Size Costume
ONE SIZE FITS MOST


AUTOMATE EVERYTHING
imgflip.com

Street Cameras

https://medium.com/cloudera-inc/streaming-street-cams-to-yolo-v8-with-python-and-nifi-to-minio-s3-3277e73723ce

QUESTIONS

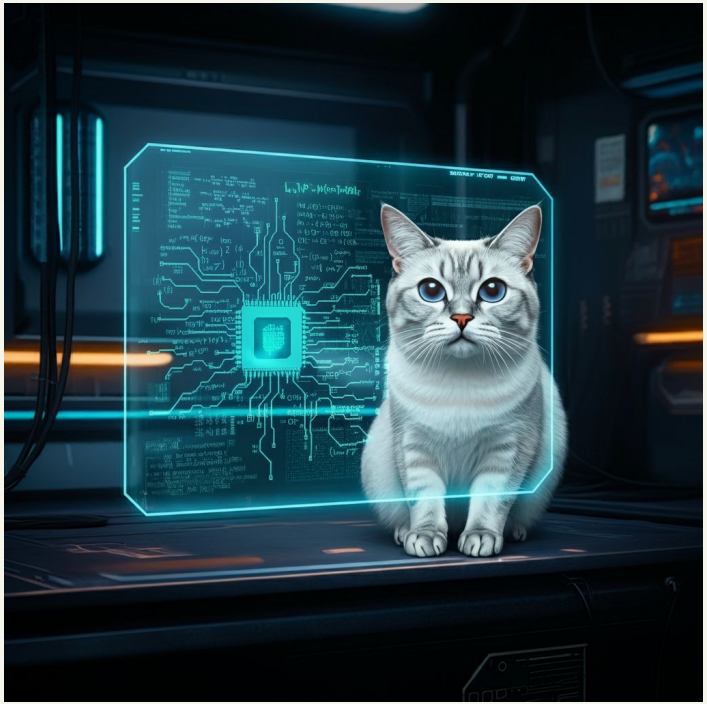


Jupyter

Vector Database

Ollama

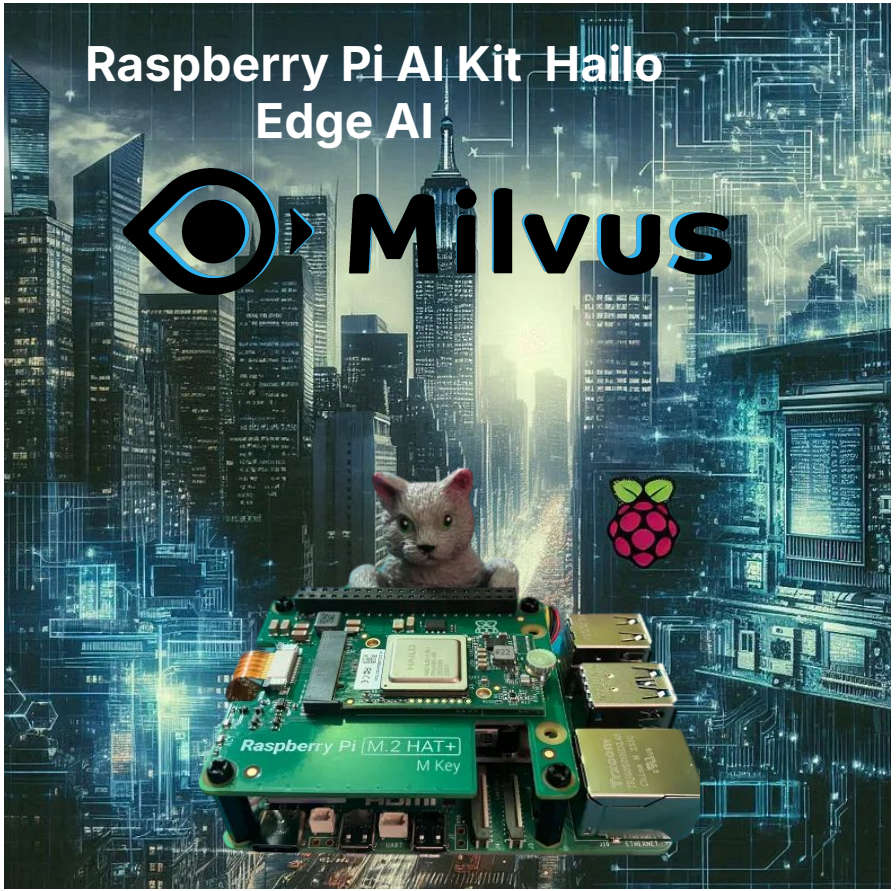
RESOURCES



https://www.datainmotion.dev/2019/12/iot-series-minifi-agent-on-raspberry-pi.html

https://medium.com/cloudera-inc/wildfires-air-quality-time-to-fire-up-the-sensors-and-start-flanking-12ea0ba33f63

https://medium.com/@tspann/not-every-field-is-just-text-numbers-or-vectors-976231e90e4d



https://medium.com/@tspann/unstructured-data-processing-with-a-raspberry-pi-ai-kit-c959dd7fff47

AIM Weekly by Tim Spann

https://bit.ly/32dAJft
https://github.com/milvus-io/milvus

This week in Milvus, Towhee, Attu, GPT Cache, Gen AI, LLM, Apache NiFi, Apache Flink, Apache Kafka, ML, AI, Apache Spark, Apache Iceberg, Python, Java, Vector DB and Open Source friends.

 |

zilliz