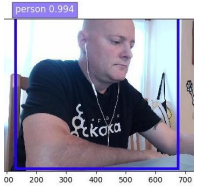
CLOUDERA

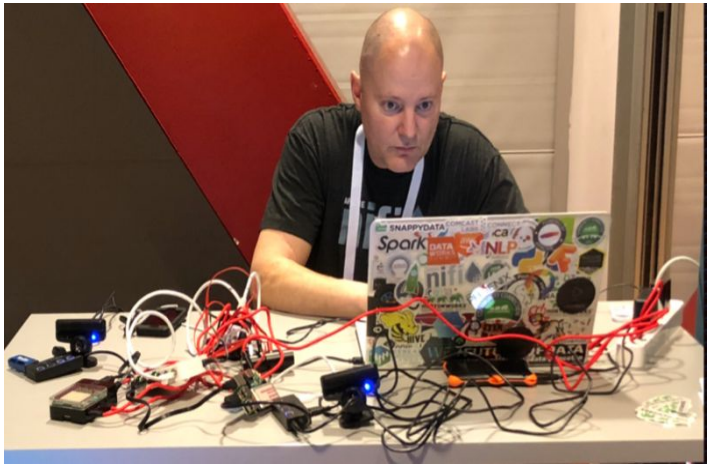
Building Apache NiFi 2.0 Python Processors

Tim Spann
Principal Developer Advocate

Feb 29, 2024



CONF42



Tim Spann

Twitter: @**PaasDev** // Blog: **datainmotion.dev**

Principal Developer Advocate.

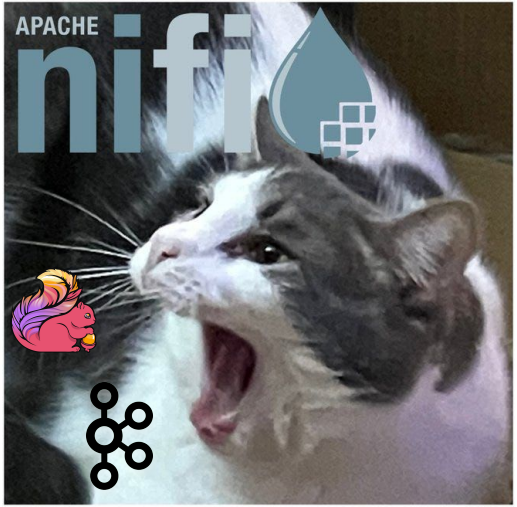
Princeton Future of Data Meetup.

ex-Pivotal, ex-Hortonworks, ex-StreamNative, ex-PwC

https://medium.com/@tspann

https://github.com/tspannhw

FLaNK Stack Weekly by Tim Spann



https://bit.ly/32dAJft

https://www.meetup.com/futureofdata-princeton/

This week in Apache NiFi, Apache Flink, Apache Kafka, ML, AI, Apache Spark, Apache Iceberg, Python, Java and Open Source friends.

Future of Data - NYC + NJ + Philly + Virtual

https://www.meetup.com/futureofdata-princeton/

From Big Data to AI to Streaming to Containers to Cloud to Analytics to Cloud Storage to Fast Data to Machine Learning to Microservices to ...

ICEBERG

@PaasDev

Apache NiFi has emerged as a robust and flexible platform for designing data integration and flow management solutions. With the release of Apache NiFi 2.0, the community has introduced a host of new features, making it even more powerful and extensible. One exciting enhancement is the ability to create custom processors using Python, providing a seamless integration of Python scripts into your data flow.

In this talk, I will delve into the world of Apache NiFi 2.0 Python processors, exploring the capabilities they offer and demonstrating how to build custom processors to enhance your data processing pipelines. Attendees will gain a deep understanding of the integration points between NiFi and Python, enabling them to leverage the extensive libraries and frameworks available in the Python ecosystem. – Introduction to Apache NiFi 2.0 – Python Processors Deep Dive – Build your own custom Python Processor – Integrating Python Libraries and Frameworks – Debugging and Troubleshooting

By the end of this talk, participants will have a comprehensive understanding of building and optimizing Apache NiFi 2.0 Python processors, enabling them to integrate Python seamlessly into their data processing workflows. This session is suitable for data engineers, architects, and anyone interested in harnessing the combined power of Apache NiFi and Python for efficient data integration and flow management.

Let’s enhance real-time streaming pipelines with smart Python code. Adding code for vector databases and LLM.

The event is a **series of pre-recorded videos**, broadcasted on our YouTube channel.
Your talk can go from **15 minutes** (lighting talk), through **30 min** (regular talk), up to **60+ min** (in-depth/with demo). We are open to any type of session

Once you are approved as a speaker, **we will request you to record your talk**. You can use any available recording tool, although we recommend OBS Studio (open source and free, compatible with iOS, Windows, and Linux). We'll provide you with a tutorial for your convenience (set everything up in under 15 minutes)
If you'd like to deliver a workshop, you can present it at a conf42 event. Due to pre-recorded nature of our events, you will need to transform it into a hands-on tutorial.

Please **avoid submitting the exact same talk** that has already been presented at a past conf42. Instead, we encourage you to **provide a continuation of your previous content** (2.0 version) or **offer a fresh new perspective on the same topic**. For example, if you previously made a theoretical overview of a tool, consider giving a hands-on demo this time.

The clarity of your voice is the most important technical aspect of a talk, lots of people will listen to it as a podcast / in the background. We recommend you use the best microphone available

Generative AI

https://github.com/tspannhw/FLaNK-HuggingFace-DistilBert-SentimentAnalysis

a BigScience initiative

BLOOM

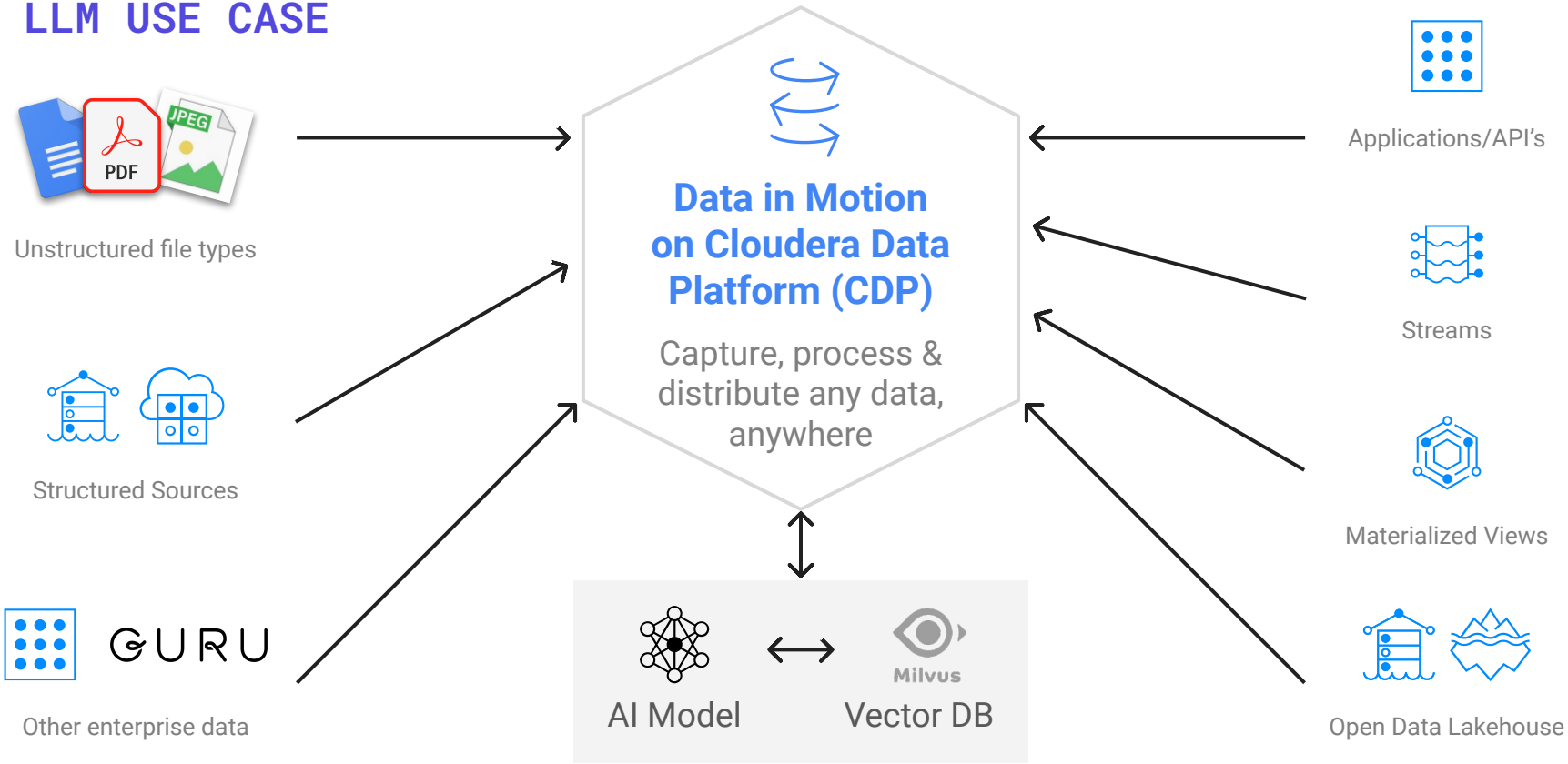
176B params · 59 languages · Open-access

watsonx.ai

CLOUDERA
Machine Learning

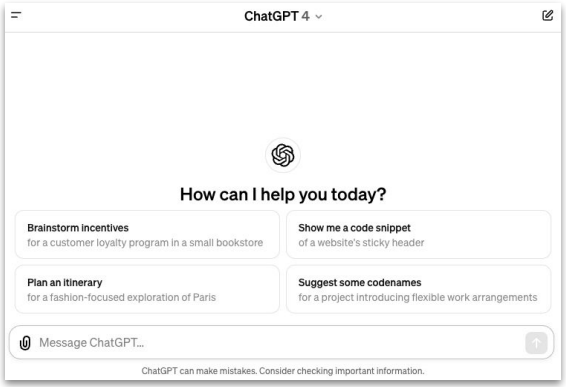
https://github.com/tspannhw/FLaNK-LLM

LLM USE CASE



GENERATIVE AI CAPABILITY

Common Families of Generative AI Capability



Text-to-Text

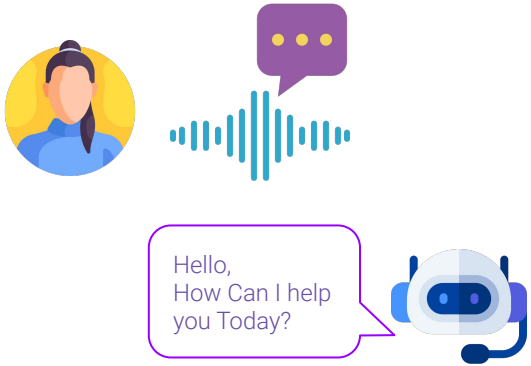
CONVERSATIONAL CHATBOTS

e.g. ChatGPT, Falcon, LLaMA

Text-to-Image

AI ASSISTED DESIGN

e.g. DALL-E, Midjourney



Text-to-Speech

INTERACTIVE VOICE RESPONSE

RAPID INNOVATION IN THE LLM SPACE

Too much to cover today.. but you should know the common LLMs, Frameworks, Tools

Notable LLMs

Closed Models

OpenAI — GPT3.5, GPT4

Claude2

Open Models

Meta AI — Llama2, Code Llama

MISTRAL AI_ — Mistral7B, Mixtral8x7B

++ 100s more… check out the HuggingFace LLM Leaderboard (pretrained, domain fine-tuned, chat models, …)

Popular LLM Frameworks

LangChain

Langchain is a framework for developing apps powered by LLMs

- Python and JavaScript Libraries
- Provides modules for **LLM Interface, Retrieval, & Agents**

LlamaIndex

LLamaIndex is a framework designed specifically for RAG apps

- Python and JavaScript Libraries
- Provides built in optimizations / techniques for advanced RAG

When to use one over the other? Use Langchain if you need a general-purpose framework with flexibility and extensibility. Consider LlamaIndex if you're building a RAG only app (retrieval/search)

Some common Vector DBs

Chroma, Pinecone, Solr, pgvector, Milvus, Weaviate

Open Source vs Self Hosted vs SaaS option

Open Community & Open Models

Hugging Face

HuggingFace is an ML community for hosting & collaborating on **models, datasets**, and **ML applications**

- Latest open source LLMs are in HuggingFace
- + great learning resources / demos

https://huggingface.co/

Cloudera Generative AI Stack

APPLICATIONS

Applied Machine Learning Prototypes (AMPs)

CLOSED-SOURCE FOUNDATION MODELS

APIs: OpenAI (GPT-4 Turbo)
Amazon Bedrock: Anthropic (Claude 2), Cohere…

MODEL HUBS

Hugging Face

FINE-TUNED MODELS

OPEN SOURCE FOUNDATION MODELS

Meta (Llama 2)

MANAGED VECTOR STORE

Pinecone

PRIVATE VECTOR STORE

Milvus, Solr*

CLOUDERA Open Data Lakehouse

REAL-TIME DATA INGEST & ROUTING → DATA WRANGLING → DATA STORE & VISUALIZATION → AI MODEL TRAINING & SERVING

CLOUD INFRASTRUCTURE

aws, Azure, Google Cloud, Red Hat

SPECIALIZED HARDWARE

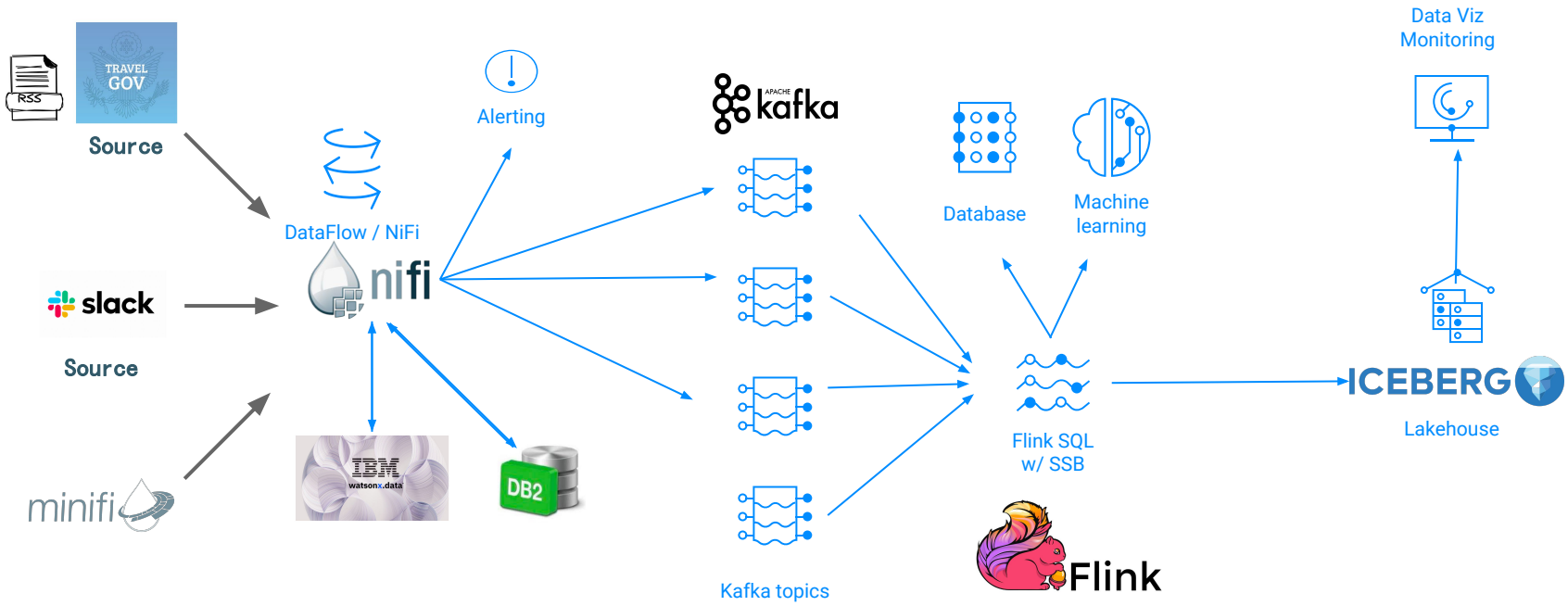
NVIDIA, Google, AMD, IBM, intel, DELL Technologies

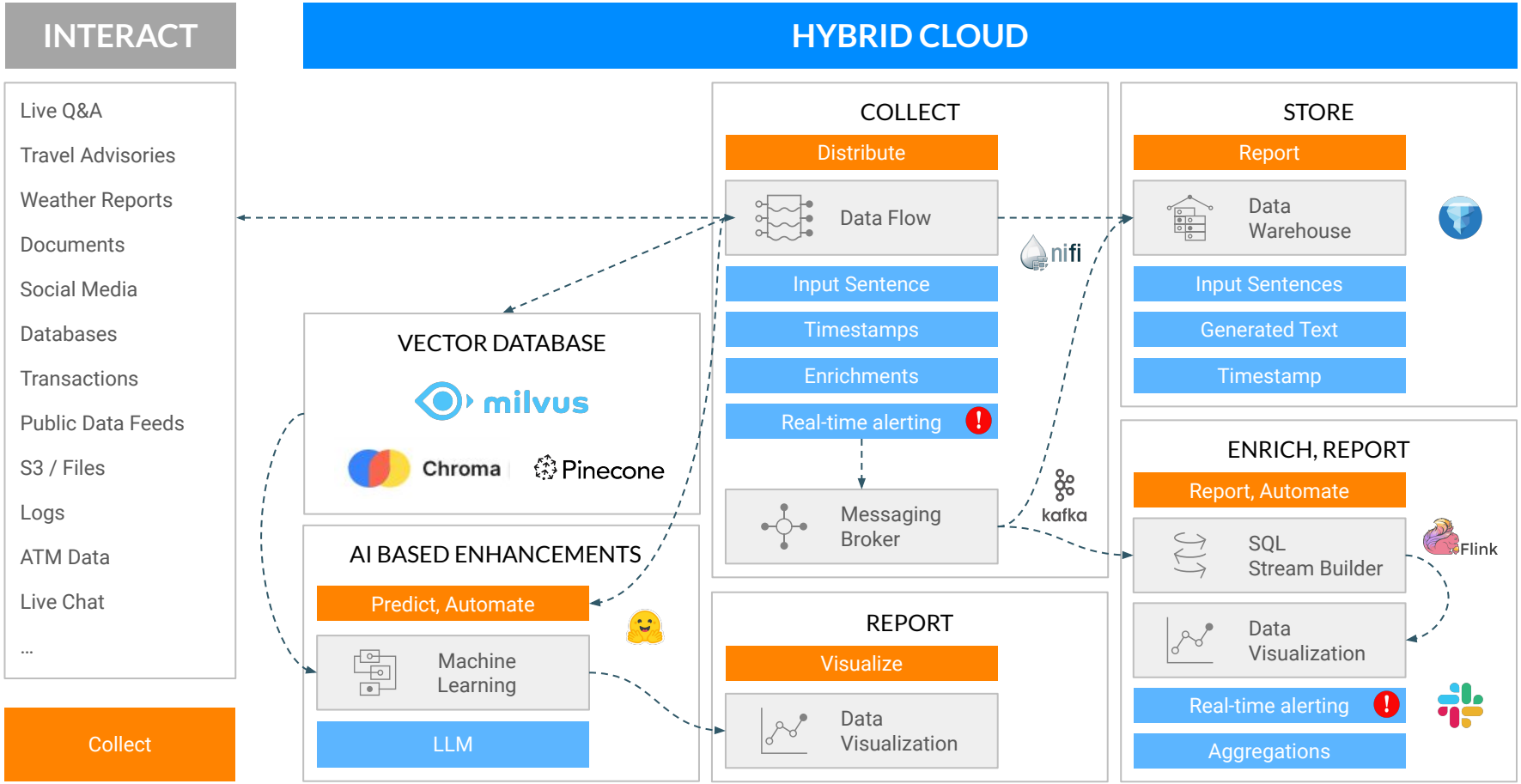
Generative AI

Apache NiFi And Real-Time GenAI

Architecture in the context of Travel Advisories

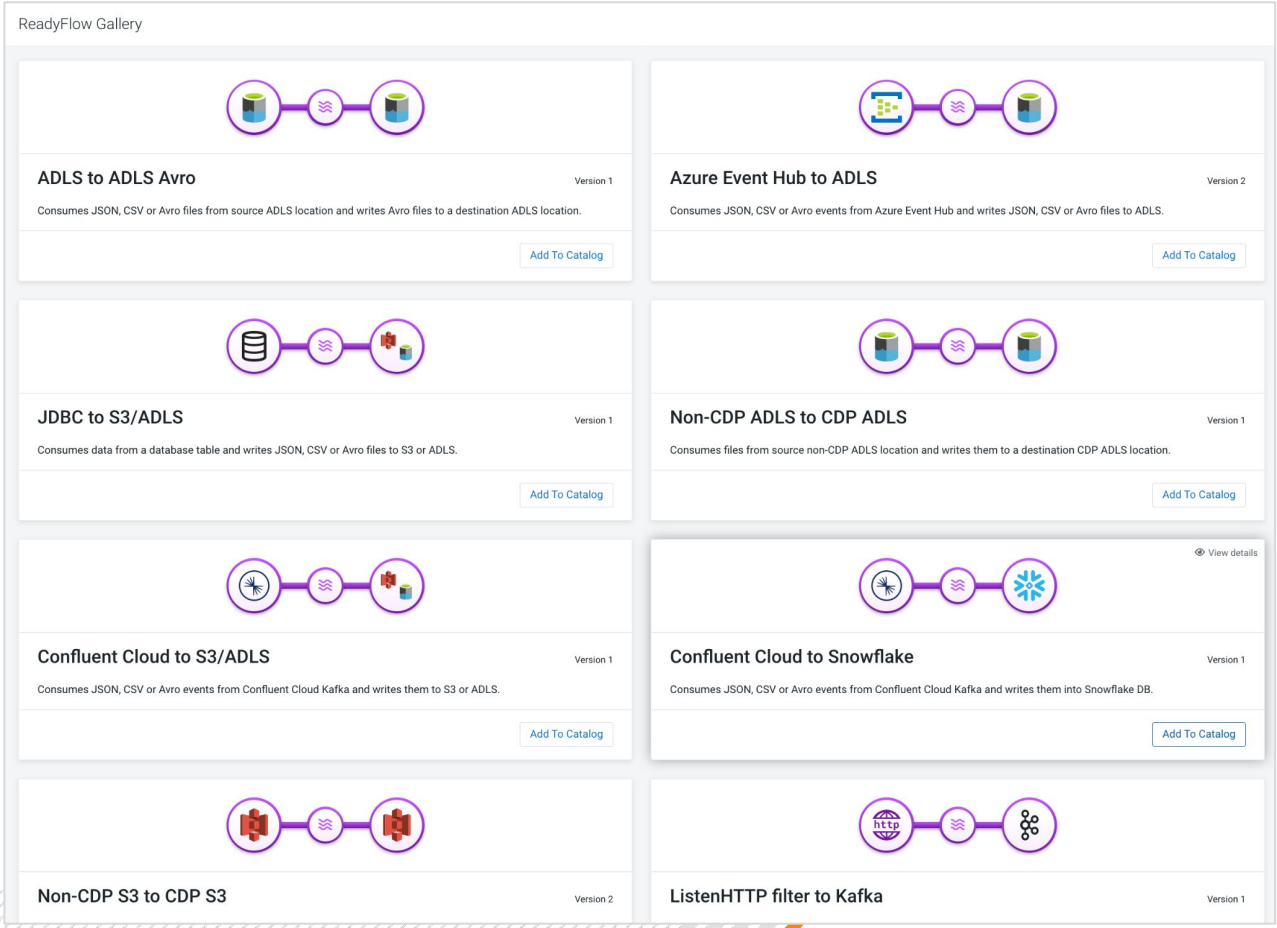
WatsonX.AI Granite LLM, NiFi, Kafka & Flink





ReadyFlow Gallery

Leverage pre-built flow templates to quickly customize and deploy new data flows



Cloudera + LLMs

LLM Serving
Serving Framework

LLM Fine Tuning Process
Training Framework

Vector DB

Data Preparation
Data Engineering

Knowledge Repository
Data Storage / Management

CML RAY NVIDIA LangChain Streamlit python

CDF nifi

Streaming Classification
Real-Time Model Deployment

milvus Pinecone Chroma

CDE Spark python DASK RAY

minifi

CDP CLOUDERA

Key:
GPU Task
CPU Task

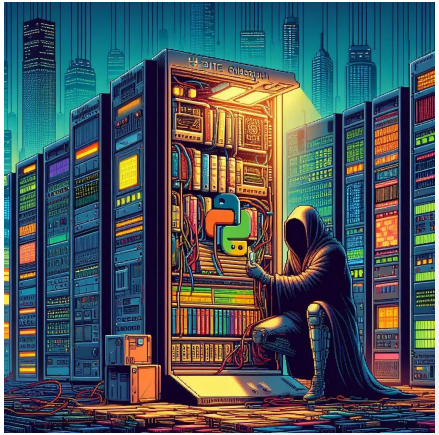
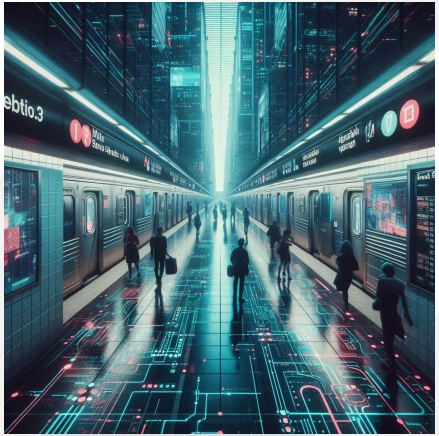
NiFi 2.0.0 Features

- Python Integration
- Parameters
- JDK 21+
- JSON Flow Serialization
- Rules Engine for Development Assistance
- Run Process Group as Stateless
- flow.json.gz

https://cwiki.apache.org/confluence/display/NIFI/NiFi+2.0+Release+Goals

https://medium.com/cloudera-inc/getting-ready-for-apache-nifi-2-0-5a5e6a67f450

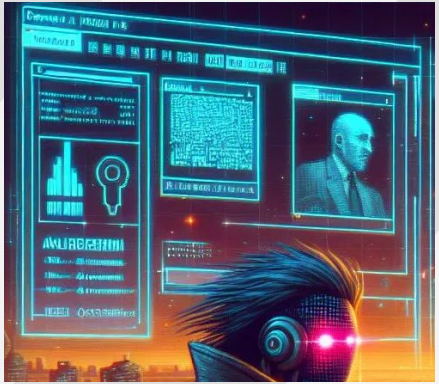


nifi

Python Processors





Basics

```python
from nifiapi.flowfiletransform import FlowFileTransform, FlowFileTransformResult
from nifiapi.properties import PropertyDescriptor, StandardValidators, ExpressionLanguageScope

class CallWatsonXAI(FlowFileTransform):
    class Java:
        implements = ['org.apache.nifi.python.processor.FlowFileTransform']

    class ProcessorDetails:
        version = '2.0.0-SNAPSHOT'
        description = """Output results of call to WatsonX.AI """
        tags = ["ibm", "WatsonX", "WatsonXAI", "generativeai", "ai", "artificial intelligence", "ml", "mach
```

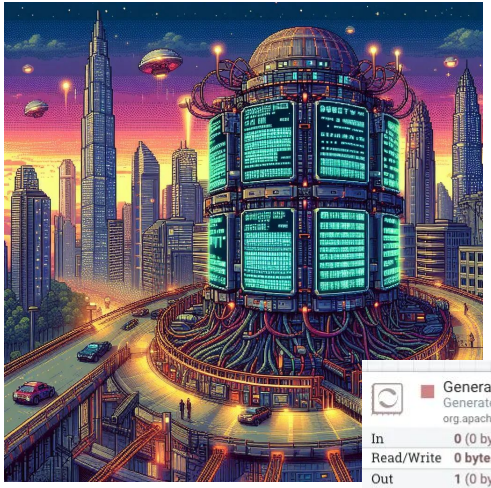
```python
  property_descriptors = [
      PROMPT_TEXT,
      WATSONXAI_API_KEY,
      WATSONXAI_PROJECT_ID
  ]

  def __init__(self, **kwargs):
      super().__init__()
      self.property_descriptors.append(self.PROMPT_TEXT)
      self.property_descriptors.append(self.WATSONXAI_API_KEY)
      self.property_descriptors.append(self.WATSONXAI_PROJECT_ID)

  def getPropertyDescriptors(self):
      return self.property_descriptors
```

Basics

```
PROMPT_TEXT = PropertyDescriptor(
    name="Prompt Text",
    description="Specifies whether or not the text (including full prompt with context) to send",
    required=True,
    validators=[StandardValidators.NON_EMPTY_VALIDATOR],
    expression_language_scope=ExpressionLanguageScope.FLOWFILE_ATTRIBUTES
)
WATSONXAI_API_KEY = PropertyDescriptor(
    name="WatsonX AI API Key",
    description="The API Key to use in order to authentication with IBM WatsonX",
    sensitive=True,
    required=True,
    validators=[StandardValidators.NON_EMPTY_VALIDATOR],
    expression_language_scope=ExpressionLanguageScope.FLOWFILE_ATTRIBUTES
)
WATSONXAI_PROJECT_ID = PropertyDescriptor(
    name="WatsonX AI Project ID",
    description="The Project ID for WatsonX",
    sensitive=True,
    required=True,
    validators=[StandardValidators.NON_EMPTY_VALIDATOR],
    expression_language_scope=ExpressionLanguageScope.FLOWFILE_ATTRIBUTES
)
```

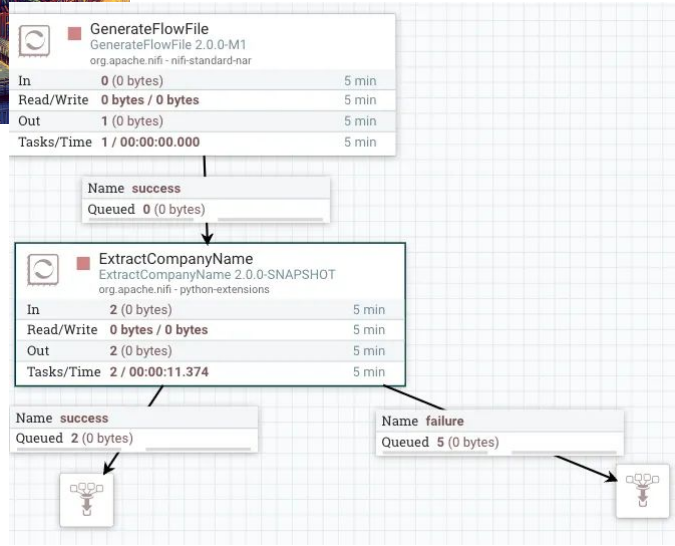
Basics

```
def transform(self, context, flowfile):
    from ibm_watson_machine_learning.foundation_models.utils.enums import ModelTypes
    from ibm_watson_machine_learning.foundation_models import Model

    prompt_text = context.getProperty(self.PROMPT_TEXT).evaluateAttributeExpressions(flowfile).getValue(
    watsonx_api_key = context.getProperty(self.WATSONXAI_API_KEY).evaluateAttributeExpressions(flowfile)
    project_id = context.getProperty(self.WATSONXAI_PROJECT_ID).evaluateAttributeExpressions(flowfile).g

    my_credentials = {
        "url"    : "https://us-south.ml.cloud.ibm.com",
        "apikey" : watsonx_api_key
    }

    model_id    = ModelTypes.LLAMA_2_70B_CHAT
    gen_parms   = None
    project_id  = project_id
    space_id    = None
    verify      = False

    model = Model( model_id, my_credentials, gen_parms, project_id, space_id, verify )
    gen_parms_override = None
    generated_response = model.generate( prompt_text, gen_parms_override )

    attributes = {"mime.type": "application/json"}
    output_contents = json.dumps(generated_response)

    self.logger.debug(f"Prompt: {prompt_text}")

    return FlowFileTransformResult(relationship = "success", contents=output_contents, attributes=attri
```



Extract Company Names

- Python 3.10+
- HuggingFace, NLP, SpaCY, PyTorch



Attribute Values

companylist
["Amazon", "Microsoft", "Cloudera", "DataSQLR", "Google", "IBM"]

filename
36fb4ae6-701a-4e1d-b890-c93b44f2200b

parsedcompany
Amazon

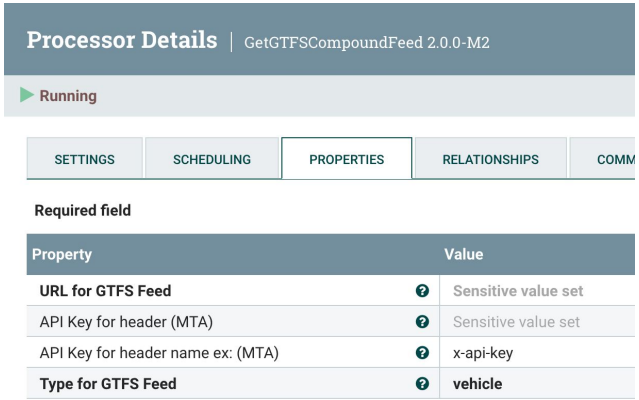
path
./

uuid
6366a2c9-3dd4-4e8f-8825-83189d403b92

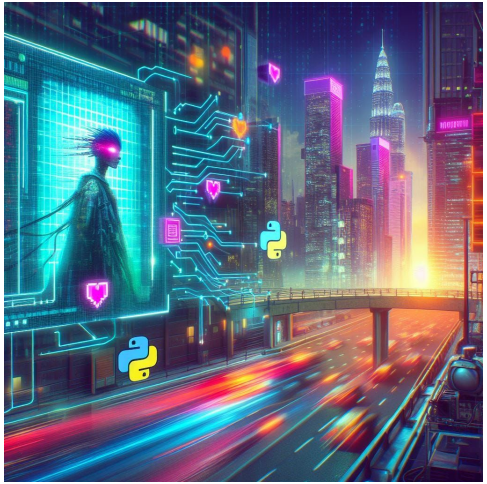
https://github.com/tspannhw/FLaNK-python-ExtractCompanyName-processor

Get Compound GTFS Data

- Python 3.10+
- GTFS to JSON



https://github.com/tspannhw/FLaNK-python-processors/blob/main/GetGTFSCompoundFeed.py



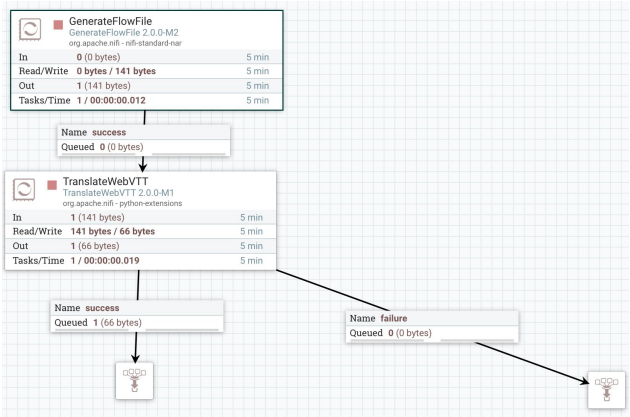
Extract Text from Web VTT

- Python 3.10+
- Web VTT to Text
- Web Video Text Tracks Format Extractor

WEBVTT

1
00:00:06.066 --> 00:00:07.166
Now let's talk about

2
00:00:07.166 --> 00:00:12.033
data retrieval, views,
and materialized views.

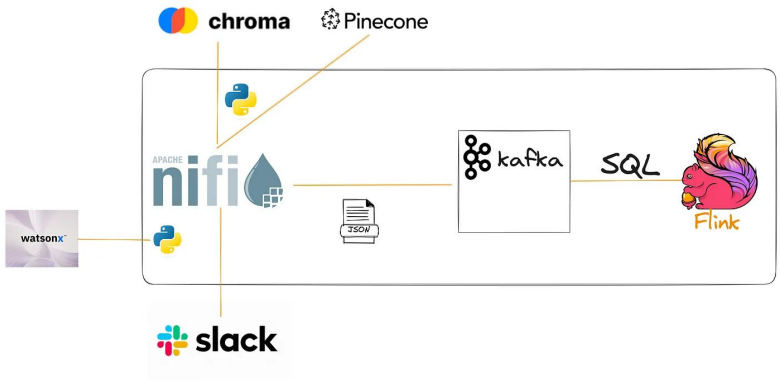


https://developer.mozilla.org/en-US/docs/Web/API/WebVTT_API

https://github.com/tspannhw/FLaNK-python-processors/blob/main/TranslateWebVTT.py

WatsonX SDK To Foundation

- Python 3.10+
- LLM
- WatsonX.AI Foundation Models
- Inference
- Secure
- Official SDK from IBM



https://github.com/tspannhw/FLaNK-python-watsonx-processor

System / Process Monitoring

- Python 3.10+
- psutil
- Swap memory, disk, networks

Attribute Values

cpu
18.1

diskusage
67583.5 MB

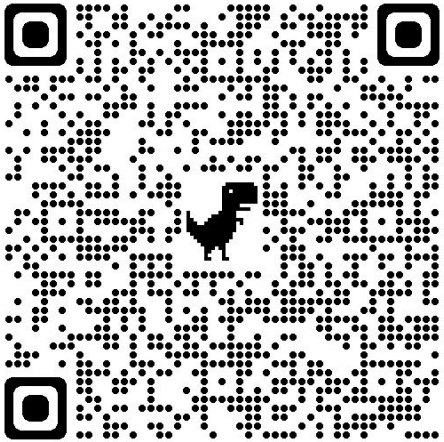
filename
f2975a45-28ab-4cea-bb04-ec25fed2efae

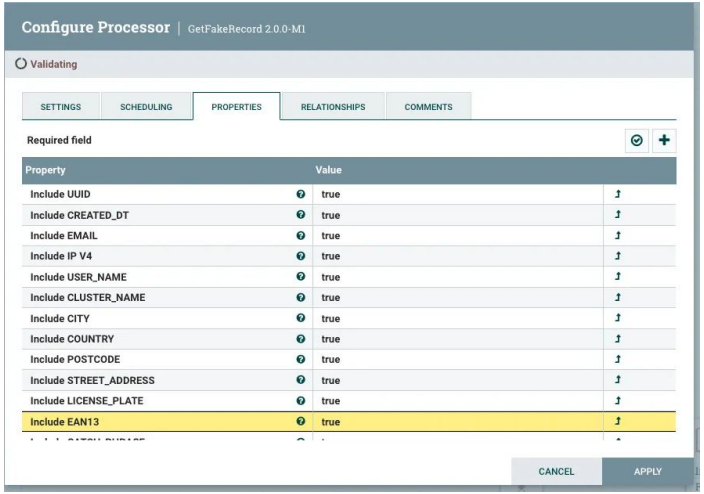
memory
67.2

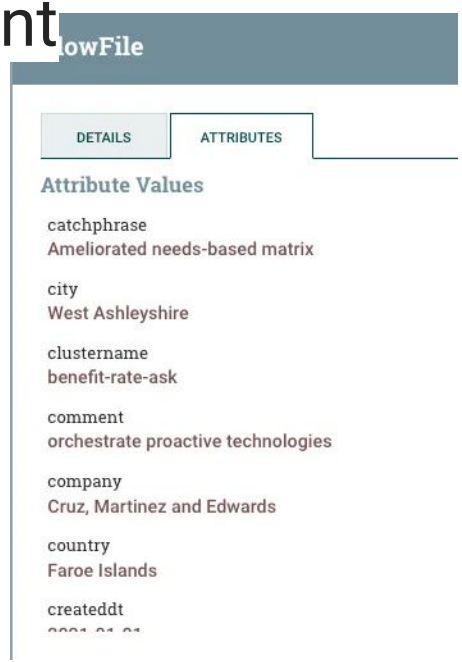
netaddr
{"lo0": [[2, "127.0.0.1", "255.0.0.0", null, null], [30, "::1", "ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff", null, null], [30, "fe80::1%lo0", "ffff:ffff:ffff:ffff::", null, null]], "en0": [[2, "192.168.1.153", "255.255.255.0", "192.168.1.255", null], [18, "bc:d0:74:65:4b:da", null, null, null], [30, "fe80::821:6f53:8208:ee72%en0", "ffff:ffff:ffff:ffff::", null, null]], "anpi2": [[18, "9a:b8:70:7d:1e:1d", null, null, null], [30, "fe80::98b8:70ff:fe7d:1e1d%anpi2", "ffff:ffff:ffff:ffff::", null, null]], "anpi1": [[18, "9a:b8:70:7d:1e:1c", null, null, null], [30, "fe80::98b8:70ff:fe7d:1e1c%anpi1", "ffff:ffff:ffff:ffff::", null, null]], "anpi0": [[18, "9a:b8:70:7d:1e:1b", null, null, null], [30, "fe80::98b8:70ff:fe7d:1e1b%anpi0", "ffff:ffff:ffff:ffff::", null, null]], "en4": [[18, "9a:b8:70:7d:1e:fb", null, null, null]], "en5": [[18, "9a:b8:70:7d:1e:fc", null, null, null]], "en7": [[18, "9a:b8:70:7d:1e:fd", null, null, null]], "en1": [[18, "36:6d:e8:f0:47:00", null,

Generate Synthetic Records w/ Faker

- Python 3.10+
- faker
- Choose as many as you want
- Attribute output

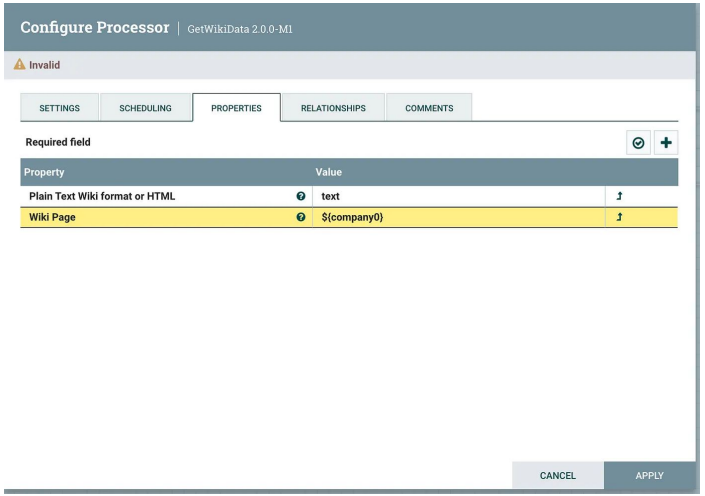


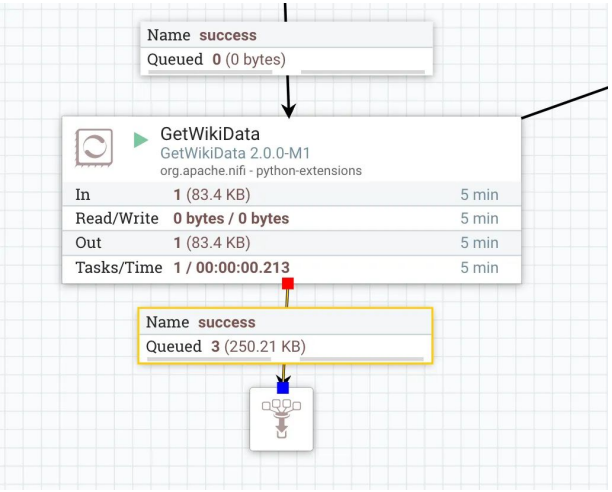




Download a Wiki Page as HTML or WikiFormat (Text)

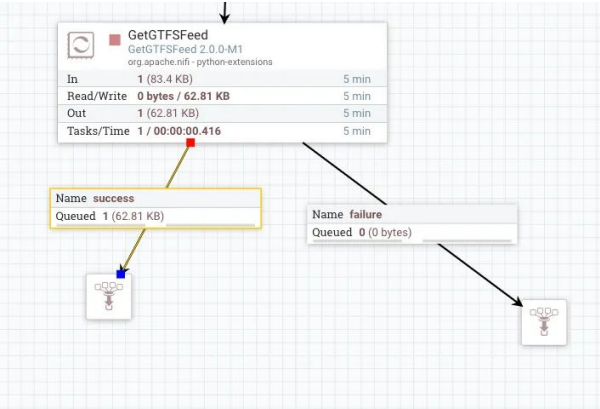
- Python 3.10+
- Wikipedia-api
- HTML or Text
- Choose your wiki page dynamically





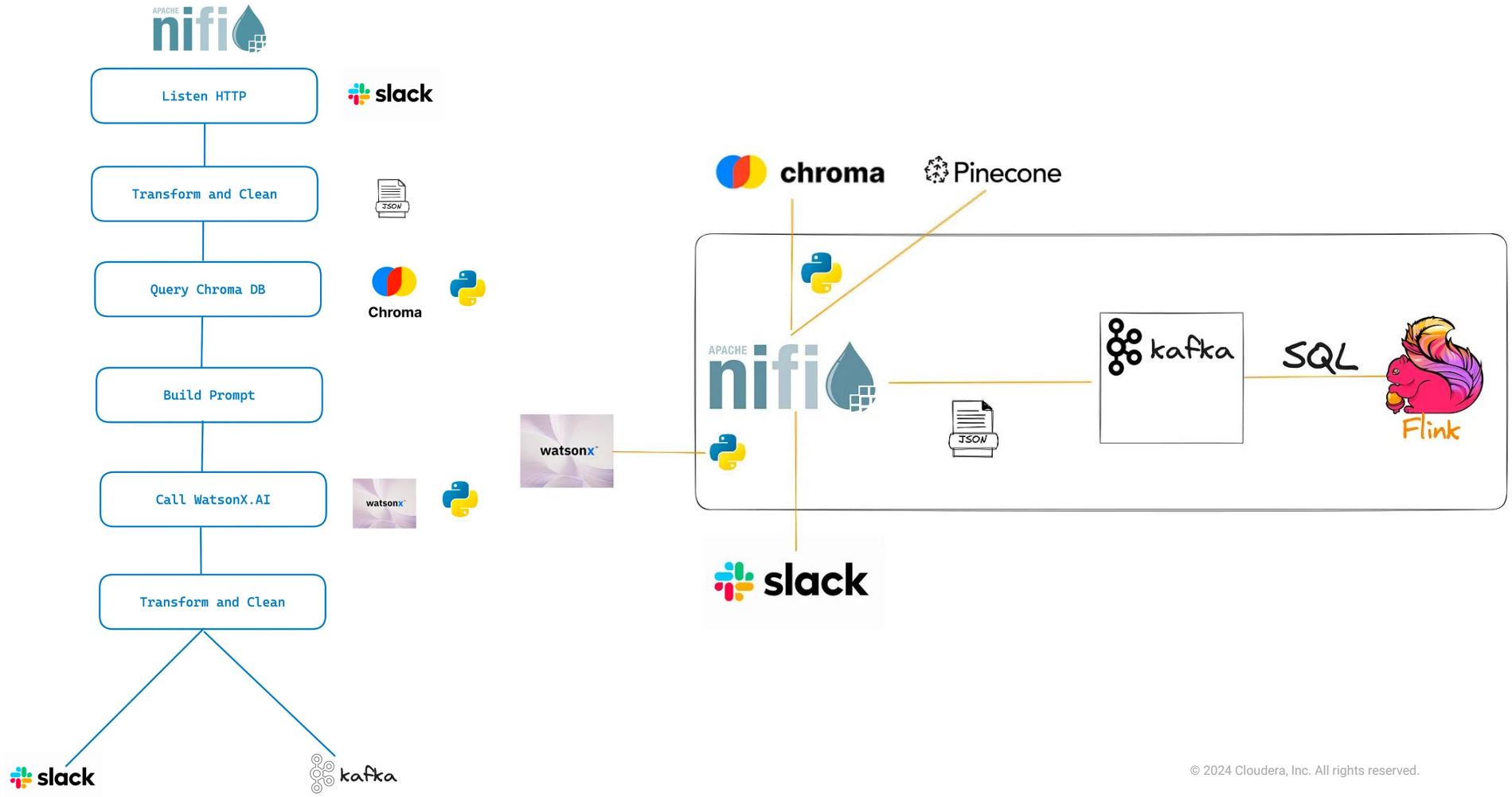
Get GTFS Data

- Python 3.10+
- GTFS from Transit URL
- Alerts, Trip Updates or Vehicle Positions
- Returns JSON
- google.transit and google.protobuf



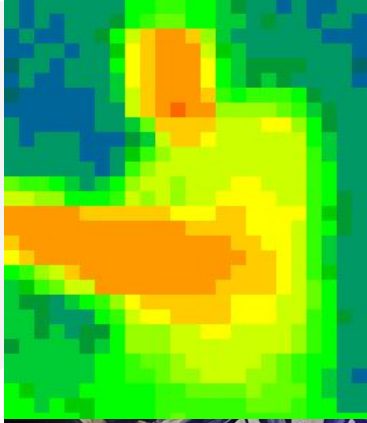
Other Python Processors

- Updated Pinecone (Vector DB Interface)
- ChunkDocument, ParseDocument
- ConvertCSVtoExcel
- DetectObjectInImage
- PromptChatGPT
- PutChroma, QueryChroma (Vector DB Interface)



DEMO

THANK YOU



APACHE nifi