



# Hamilton is a python micro-framework for describing dataflows

👉 an OS tool to add to your LLM App toolbelt

➤ DAGWORKS (YCW23) Stefan Krawczyk CEO  
⚡ talk @ LLM Avalanche June 2023



# Context: Production[ization] Pains

Two things I've heard at conferences and from peers:

1. Observability is tough
2. Iterations & debugging can be very painful

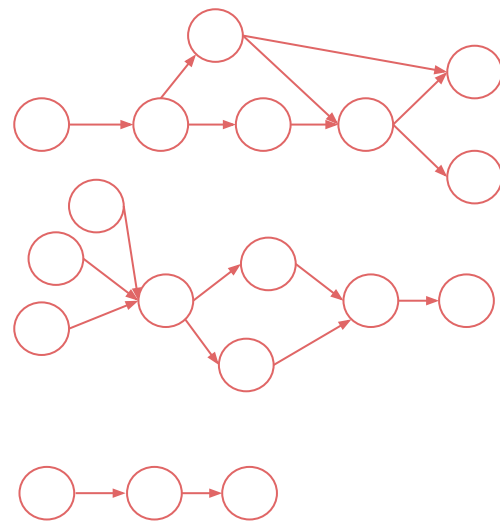
**My take:** it comes down to the software engineering!

**TL;DR:**

**Hamilton can help you here**, especially if you're rolling your own code.



# LLM Powered Apps are just a sequence of dataflows [DAGs]



**Use Hamilton here**

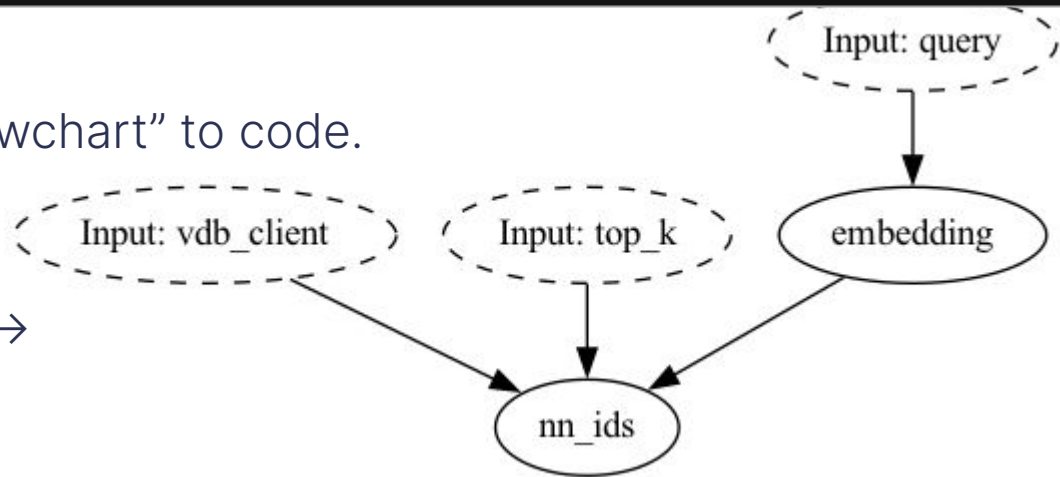


# What is Hamilton?

**Paradigm:** write declarative python functions → get a dataflow/DAG:

```
def embedding(query: str) -> List[float]:  
    response = openai.Embedding.create(input=query, model="HARDCODED")  
    return response["data"][0]["embedding"]  
  
def nn_ids(embedding: List[float], vdb_client: Client, top_k: int) -> List[int]:  
    return vdb_client.top_k(embedding, top_k=top_k)
```

The most direct way to go from “flowchart” to code.



**BTW:** Hamilton created this image →



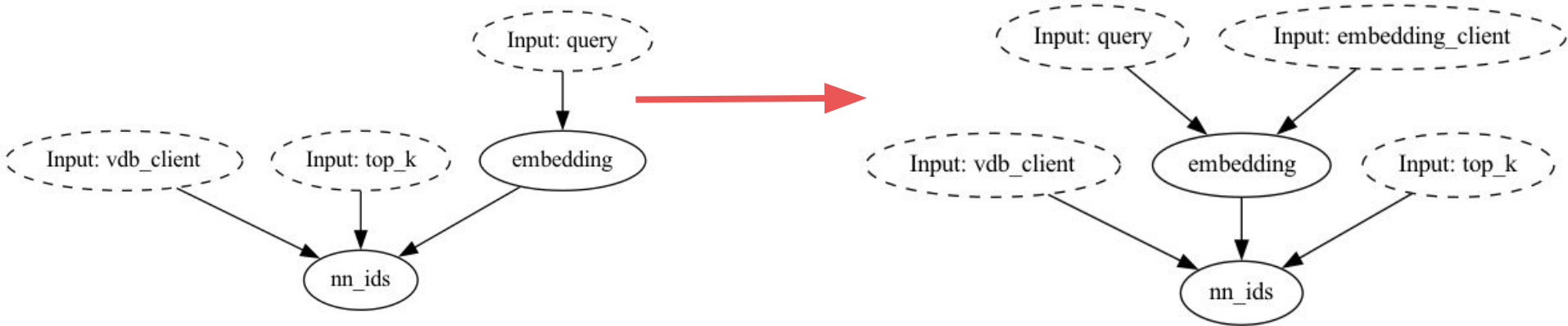
# Why Hamilton?

> Iteration speed, clarity, & testability

```
def embedding(query: str) -> List[float]:  
    response = openai.Embedding.create(input=query, model="HARDCODED")  
    return response["data"][0]["embedding"]
```



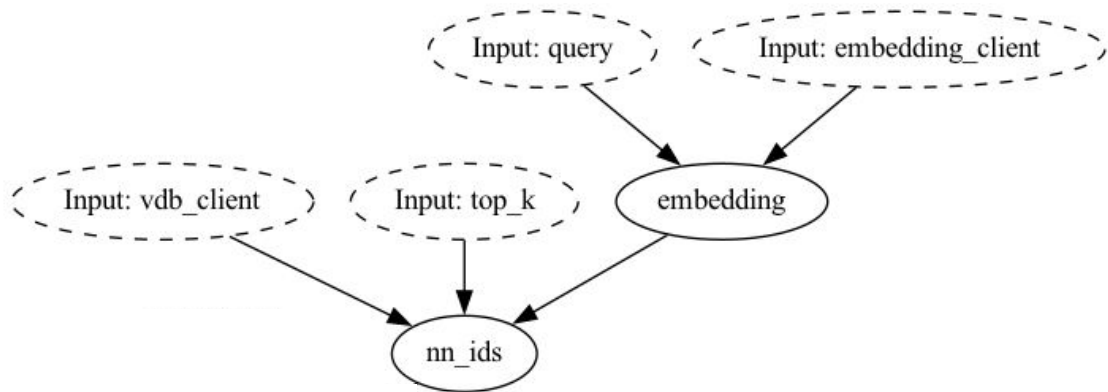
```
def embedding(query: str, embedding_client: object) -> List[float]:  
    return embedding_client.get_embedding(query)
```





# Why Hamilton?

## > Modularity/Composability

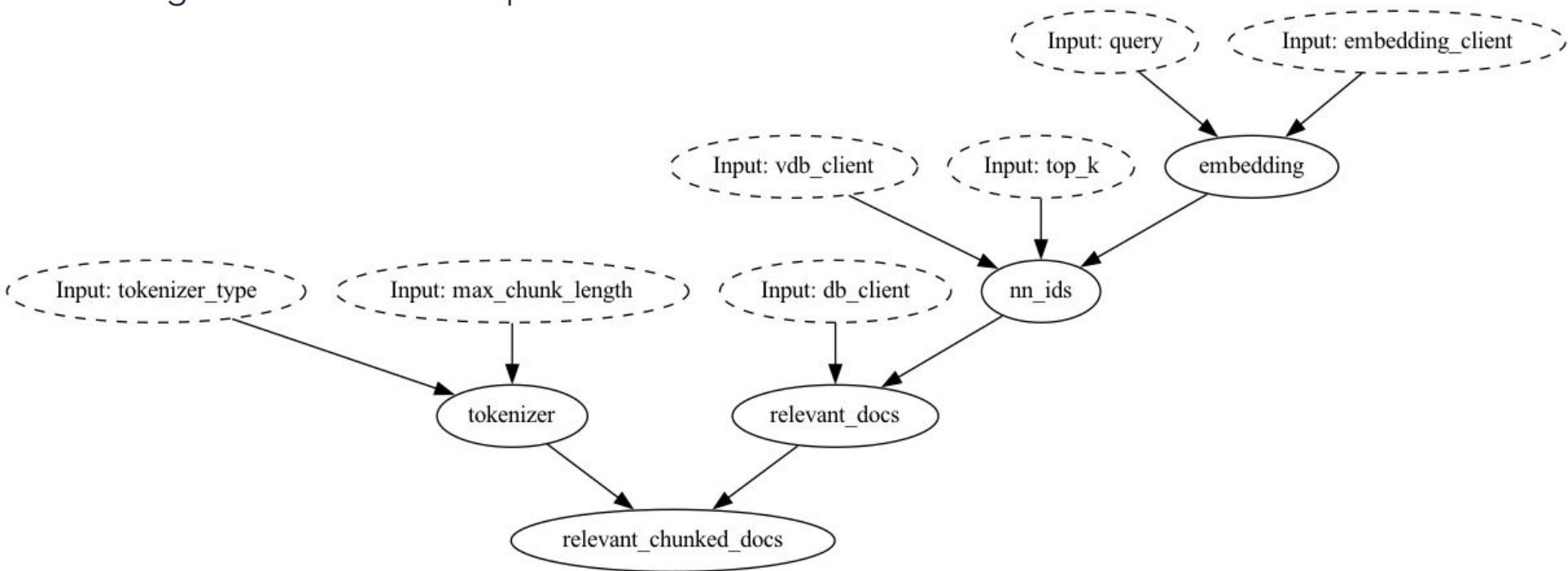




# Why Hamilton?

## > Modularity/Composability

1. Straightforward to compose & reuse flows.

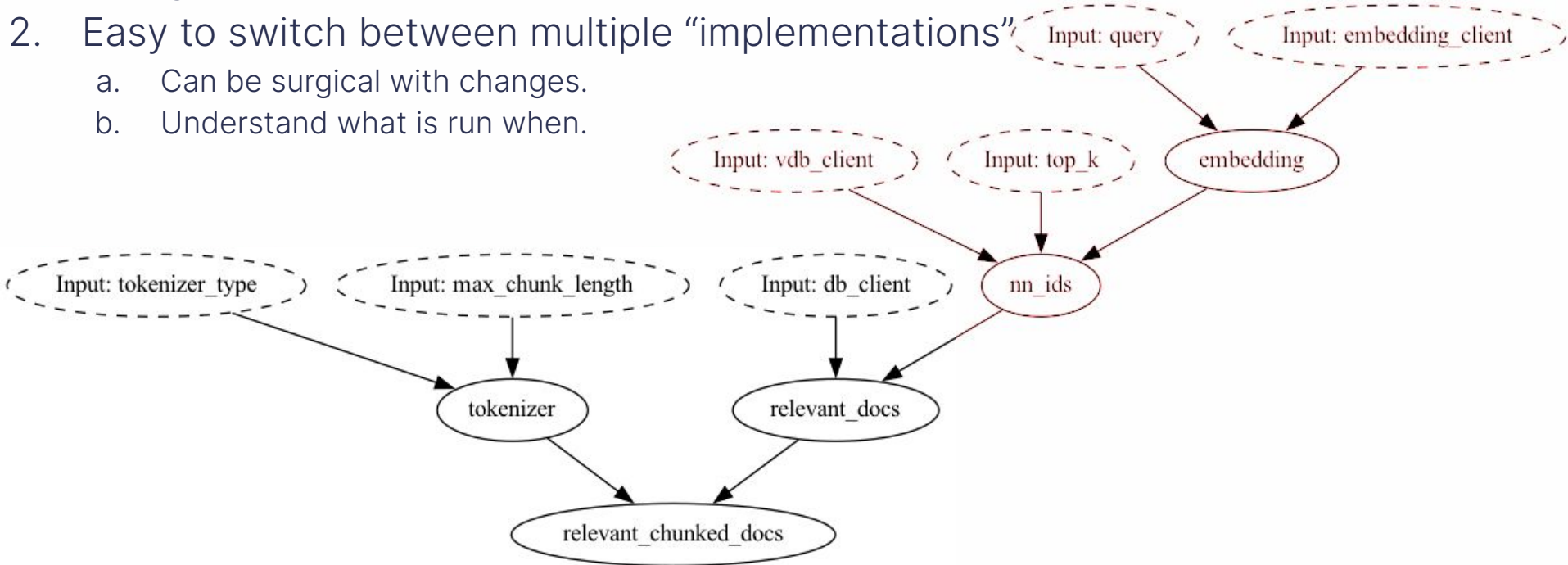




# Why Hamilton?

## > Modularity/Composability

1. Straightforward to compose & reuse flows.
2. Easy to switch between multiple “implementations”
  - a. Can be surgical with changes.
  - b. Understand what is run when.










# Hamilton: a tool to add to your LLM App toolbelt

## Summary + a few things:

1. Write python functions → get a dataflow; use to model “**actions**”.
  - a. Great iteration & testing story.
  - b. Easy to version with git.
2. Runs everywhere Python runs: Jupyter  Airflow  FastAPI 
3. Built to be extended; comes with more than I showed
  - a. Can add custom caching, decorators for observability, add metadata to functions, etc.
4. Can be used for data & ML pipelines
  - a. One tool to use for all your “dataflow” needs.



# Get started:



```
pip install sf-hamilton
```

▶: [tryhamilton.dev](https://tryhamilton.dev) ← runs 🦆 in the browser!

★: <https://github.com/dagworks-inc/hamilton> (see examples)

🚩: Join us on [slack](#)

🐦: [https://twitter.com/hamilton\\_os](https://twitter.com/hamilton_os)

🐦: <https://twitter.com/stefkrawczyk>

👤: <https://www.linkedin.com/in/skrawczyk/>



<https://www.dagworks.io> (sign up! We're building on top of Hamilton!)