Neura**lake**

Simple systems for complex data

GAUTHAM ACHARYA

Software Engineering Lead @ Neuralink

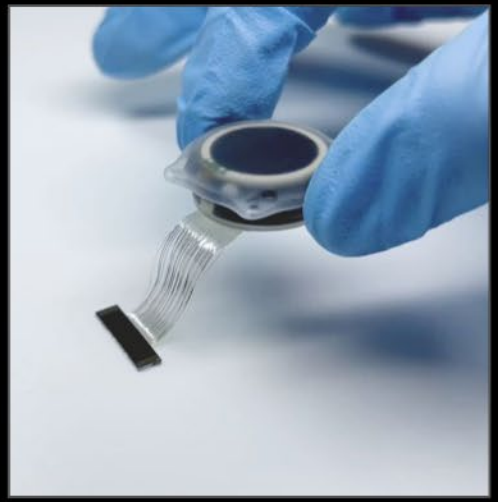
- Data engineering
- Full-stack development
- Manufacturing software
- Lab management tools

gautacharya@gmail.com

NEURALINK'S MISSION

Create a generalized brain interface to restore autonomy to those with unmet medical needs today and unlock human potential tomorrow.

THE NEURALINK STACK

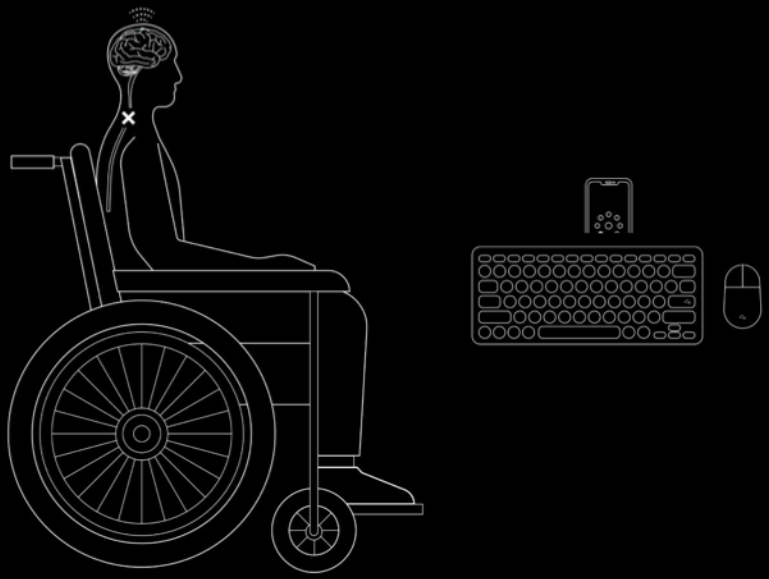


Invisible device with all power and compute on head

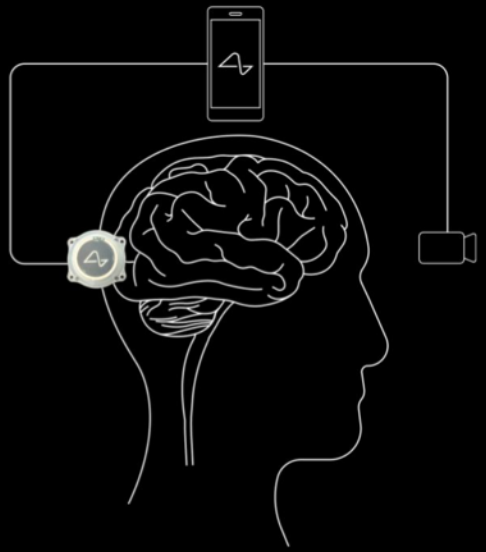
Surgical robot to implant threads with micron precision

Neural decoding software to translate brain signals into useful outputs

RESTORE AUTONOMY



Telepathy

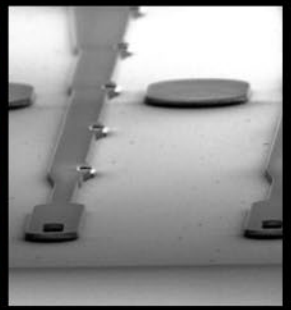


Blindsight

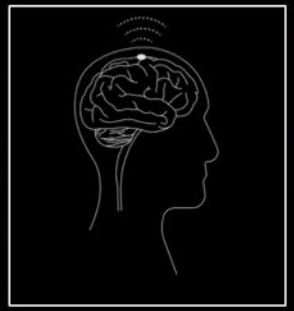
Data @ Neuralink

COMPLEX DATA SOURCES

HISTOPATHOLOGY



MANUFACTURING



BCI SESSIONS

SURGERIES

...AND MORE

SIMPLE SYSTEMS

Our **design philosophy**:

1. **Systems that scale down to** a single developer machine and up to stateless clusters.
2. **Prioritize local development experience** - use composable libraries instead of distributed services.
3. **No large, stateful distributed clusters** for data lakes/warehousing.
4. **Code as a catalog** – define tables in code, generate a catalog and APIs without databases

1. Ingestion
2. Discovery
3. Access

Low-latency streaming ingestion pipelines.

Elegant schema versioning and backfilling of data lakes.

1. Ingestion
2. Discovery
3. Access

Simple data catalog, generated from source

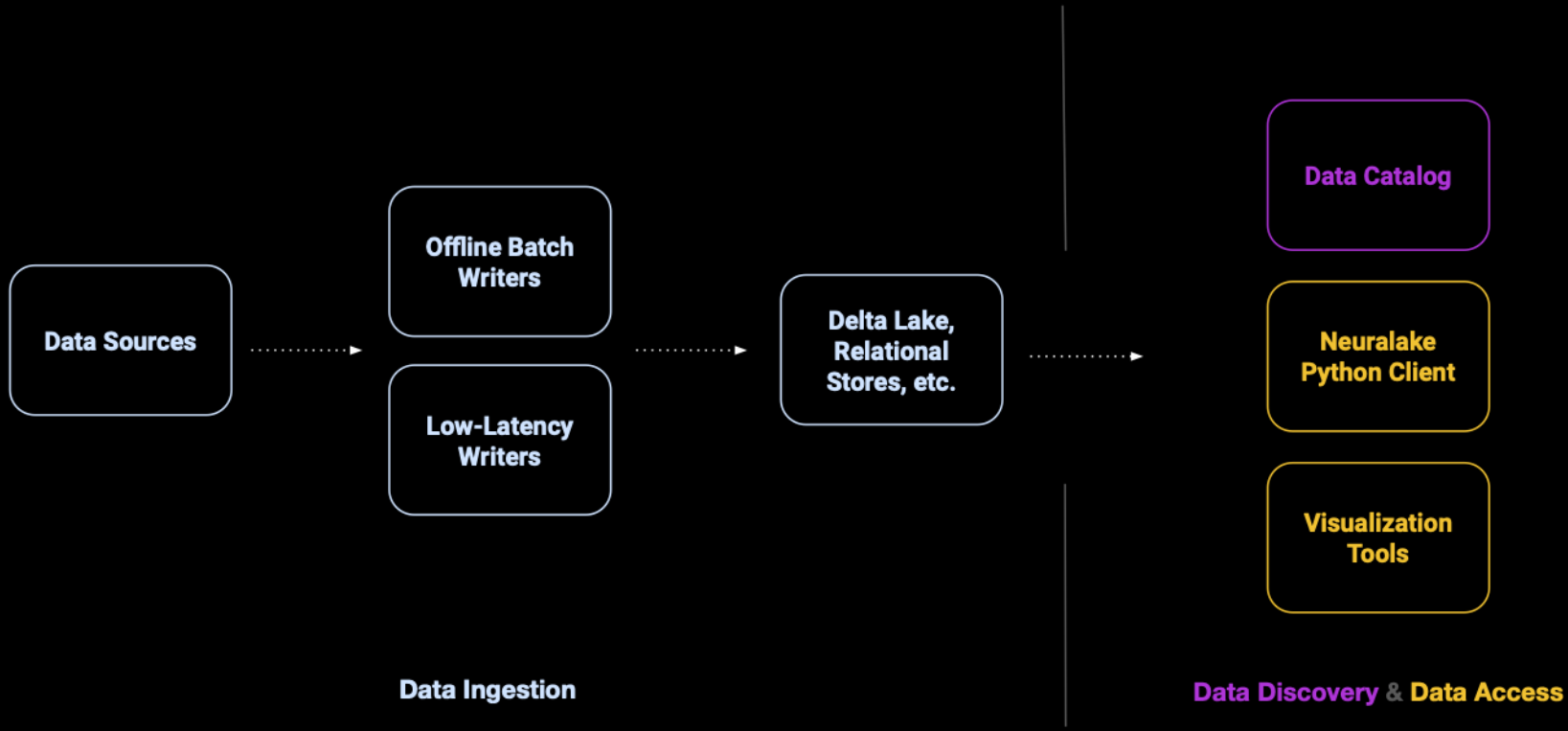
Low-code dashboards generated from catalog definitions

1. Ingestion
2. Discovery
3. Access

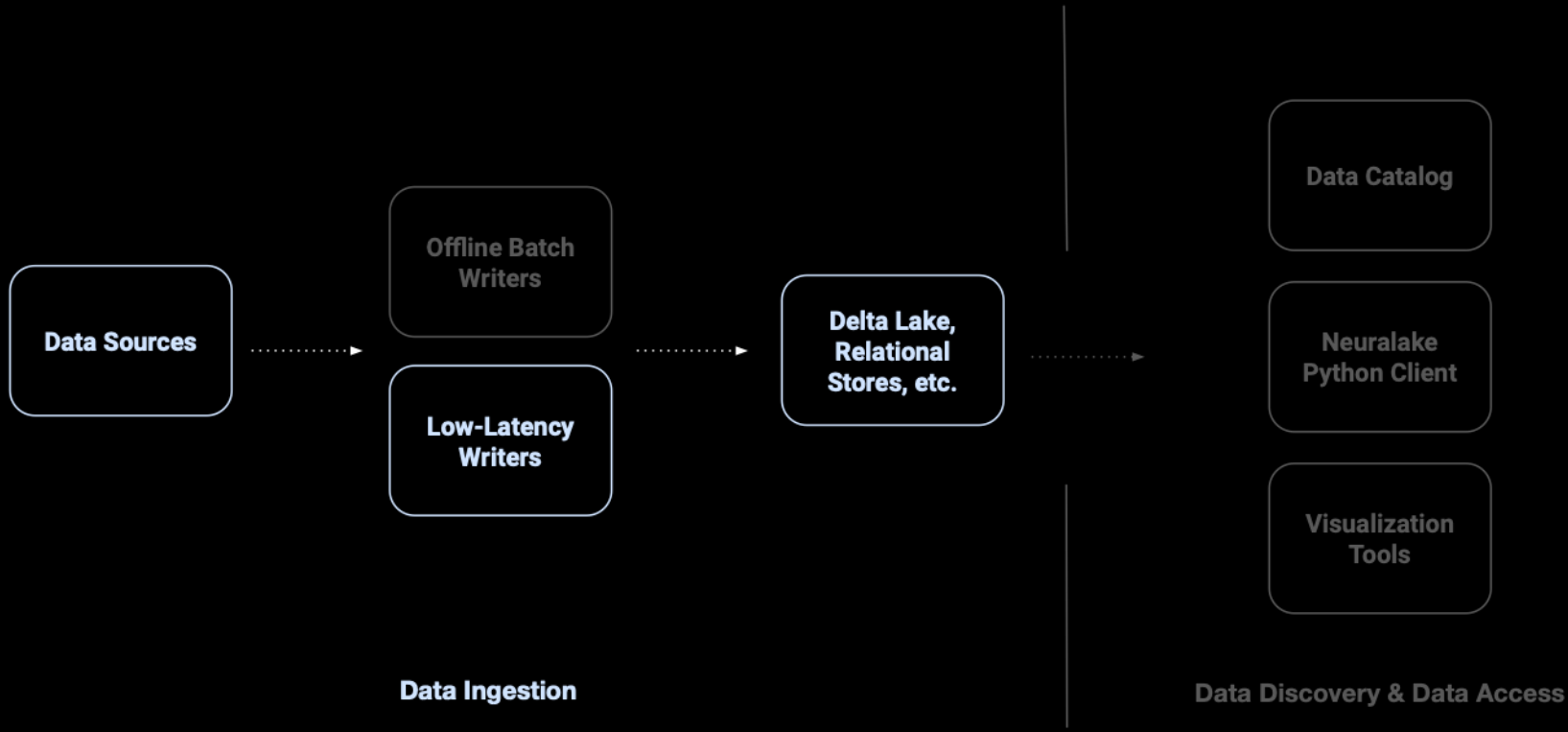
One-click to get a dataframe from data catalog

Auto-generated SQL and REST APIs from data catalog

The Neura**lake** Data Platform



The Neura**lake** Data Platform



Real-time Data Ingestion in Neuralake

Delta Lake is an open source project that enables a Lakehouse architecture on top of data lakes.

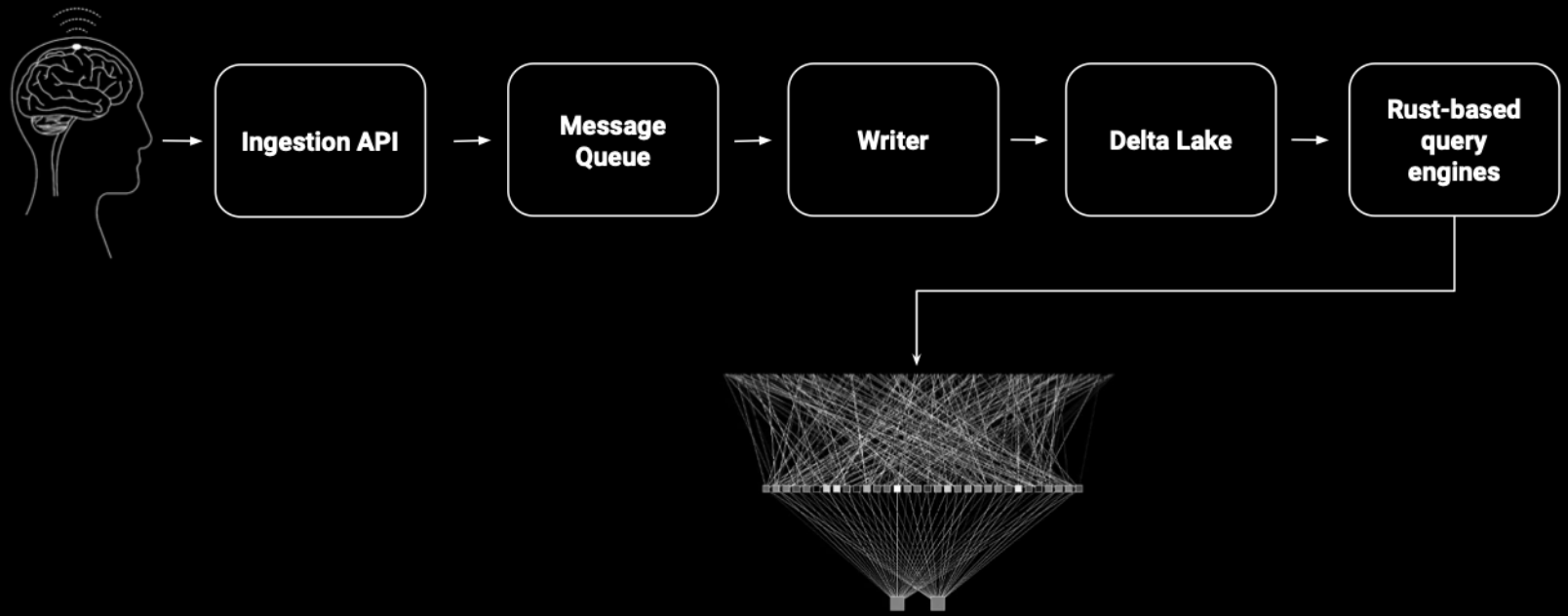
Delta Lake provides ACID transactions, scalable metadata handling, and unifies streaming and batch data processing on top of existing blob stores.

Whitepaper

delta-rs implements the Delta Lake protocol in Rust

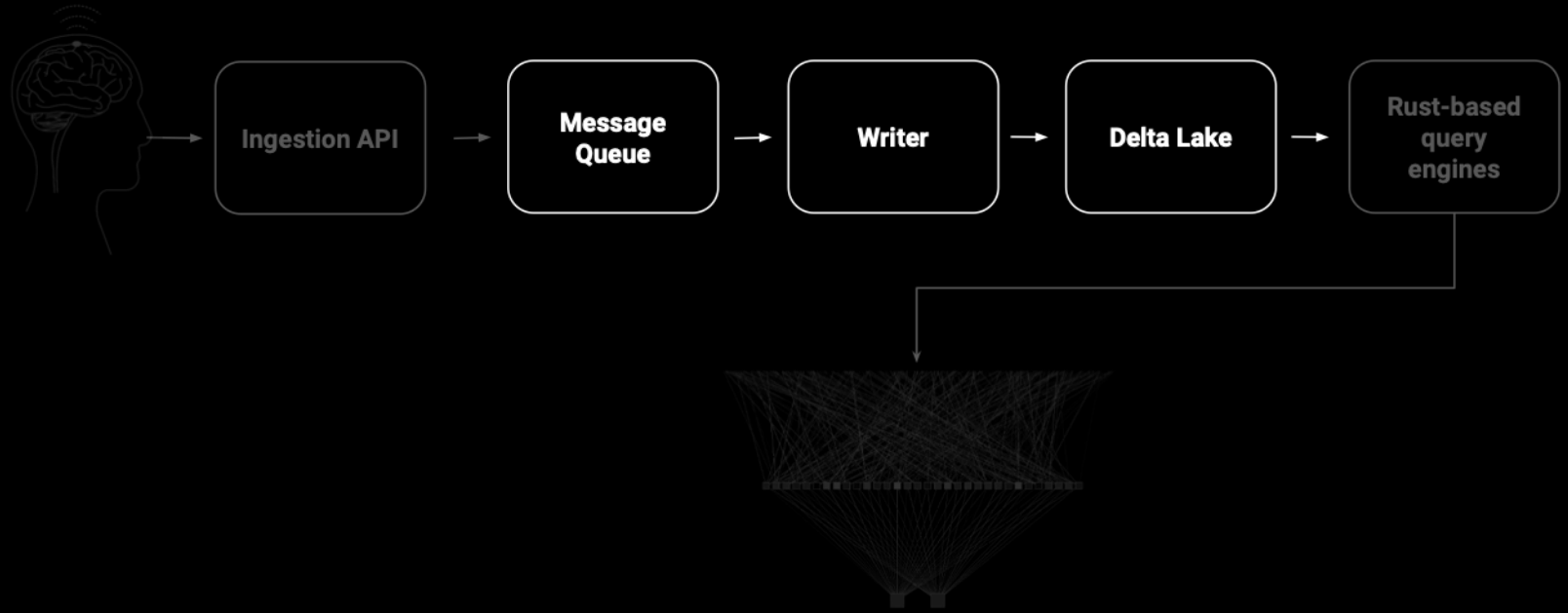
Github

Real-time Data Ingestion in Neuralake



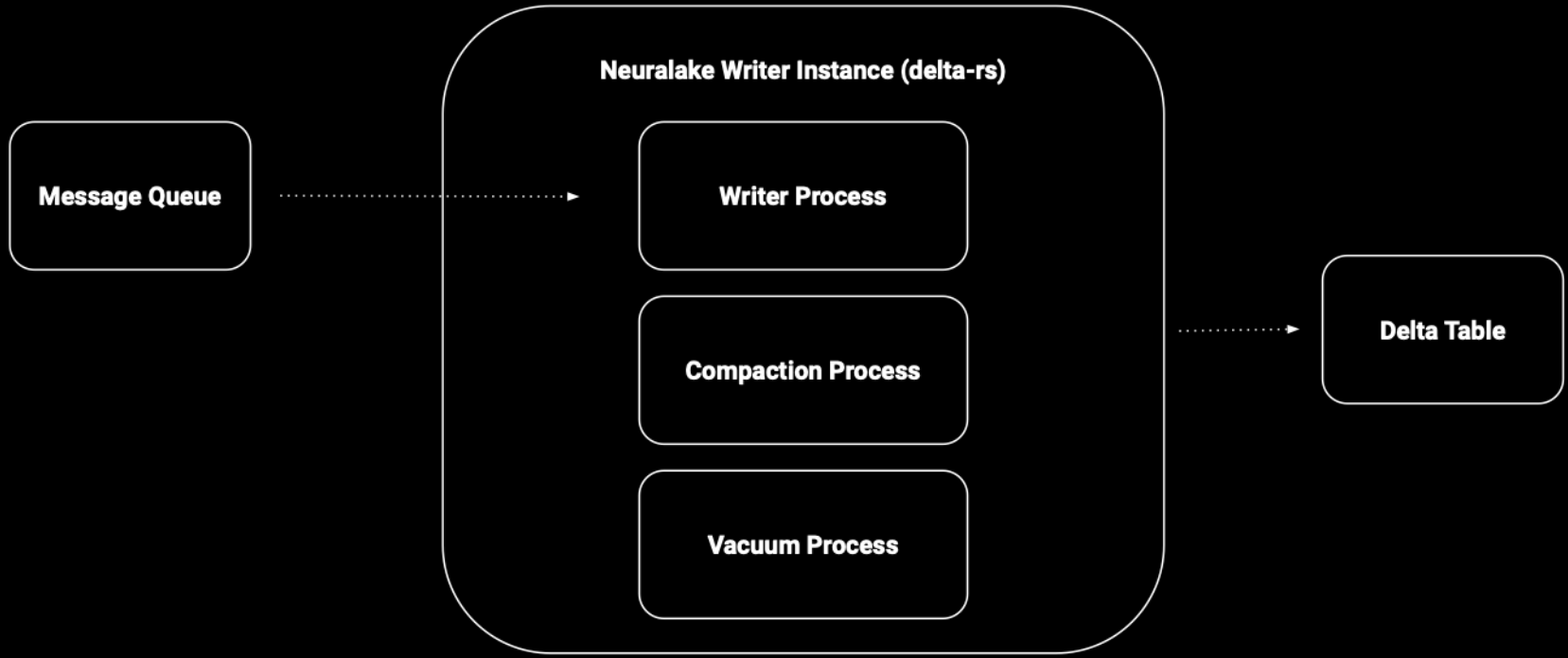
Labeled data available for model training

Real-time Data Ingestion in Neuralake

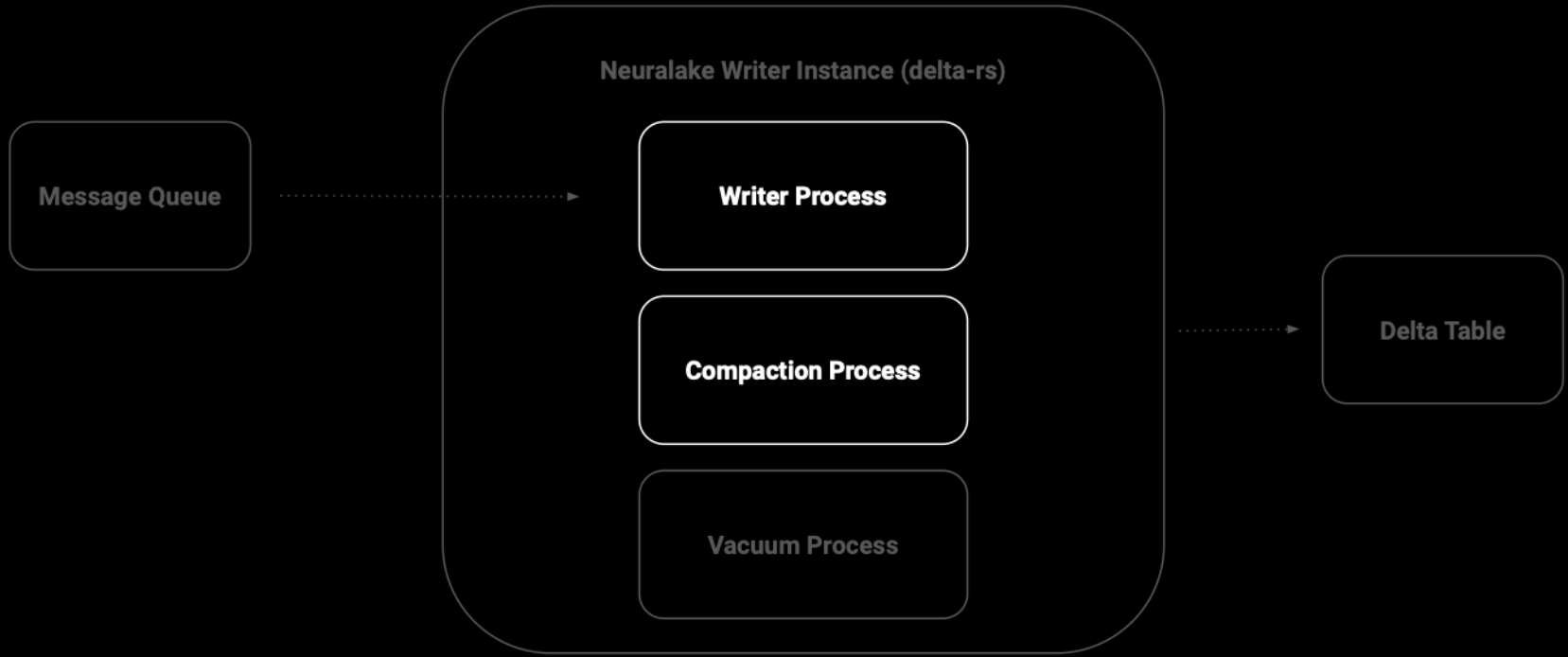


Labeled data available for model training

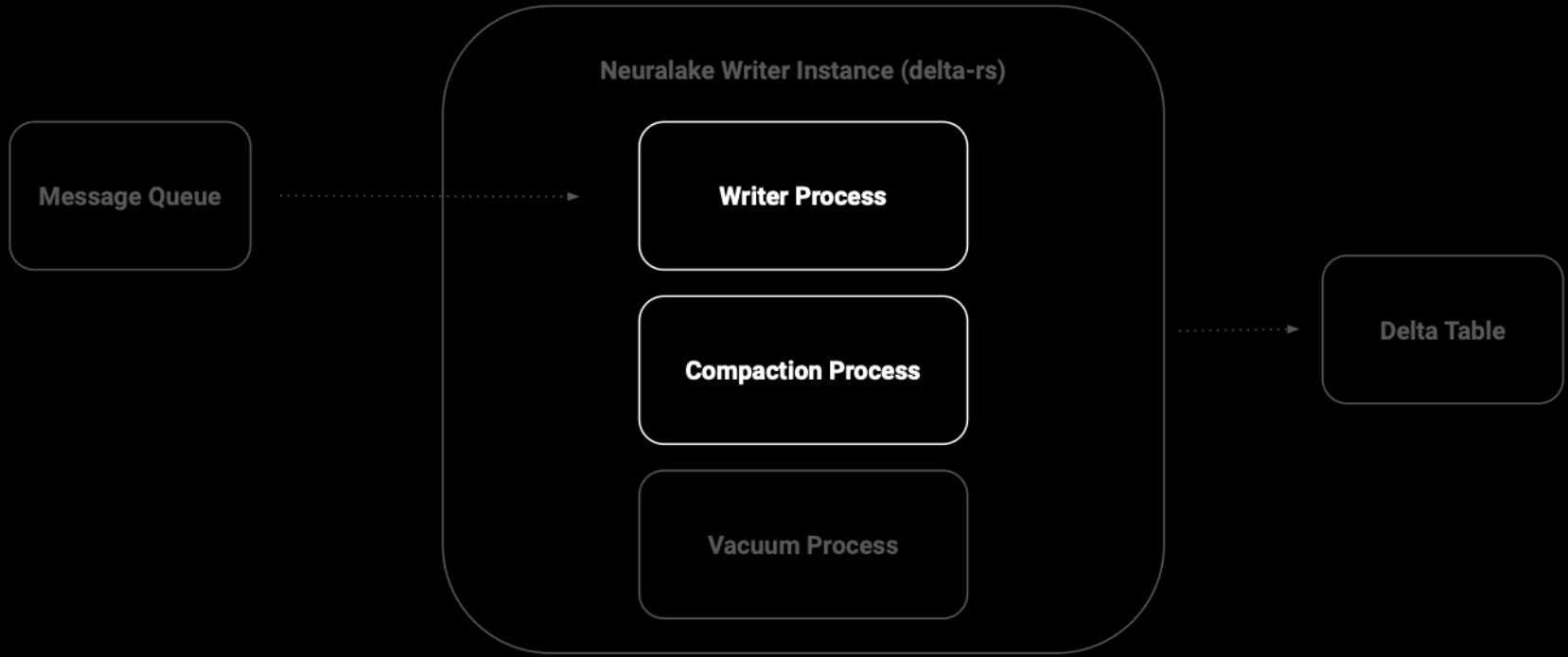
Real-time Data Ingestion in Neuralake



Writer and compaction write async, and only acquire an interprocess lock to commit

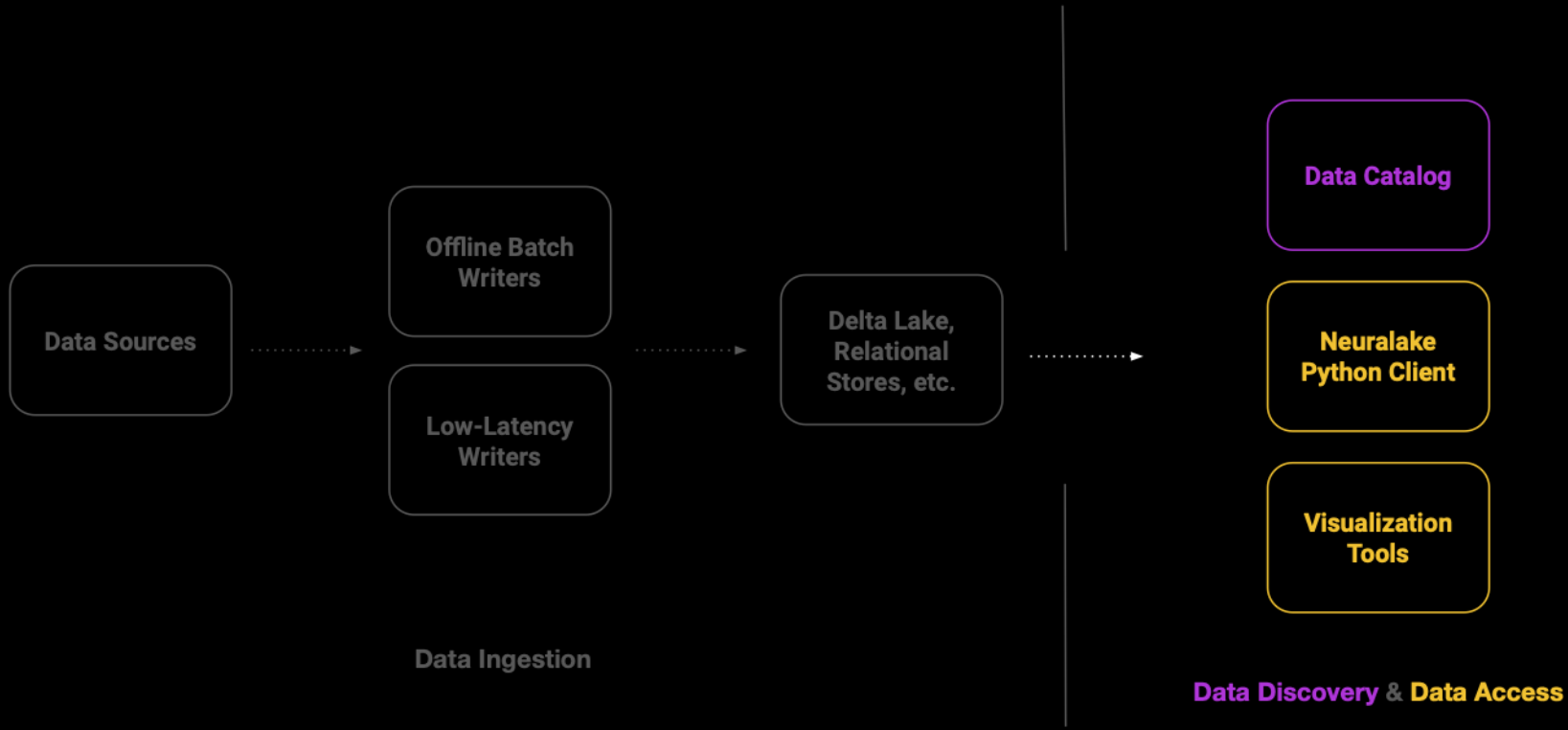


Partial writes are resolved via a custom PUT-if-absent semantic.



See this talk on the use case for PUT-if-absent.

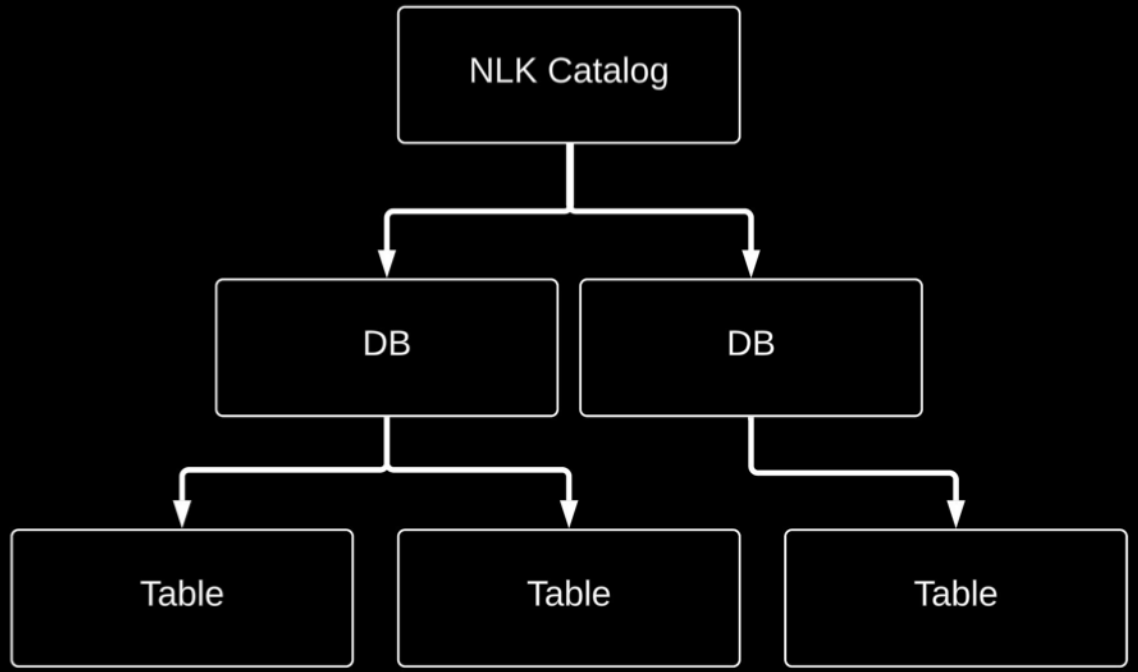
The Neura**lake** Data Platform



The Neuralake Python client consists of catalogs, databases, and tables for easy discovery and querying

Neuralake Python Client

- **Catalog, DB, and table structure**
- Uniform API to retrieve data as dataframes
- Flexible syntax for defining tables
- Auto-generated catalog
- Auto-generated API for visualization tools



Clean, uniform API for retrieving tables

Neuralake Python Client

- Catalog, DB, and table structure
- **Uniform API to retrieve data as dataframes**
- Flexible syntax for defining tables
- Auto-generated catalog
- Auto-generated API for visualization tools

```python
from neuralake.catalogs import NlkCatalog
from neuralake.core import Filter

df = NlkCatalog.db('bci').table(
    'normalized_band_power',
    (
        Filter('implant_id', '==', 4595),
        Filter('date', '==', '2024-04-28'),
        Filter('hour', '==', 23),
    ),
)
print(df.collect())
```

Developers can easily add new tables with a declarative syntax

Neuralake Python Client

- Catalog, DB, and table structure
- Uniform API to retrieve data as dataframes
- **Flexible syntax for defining tables**
- Auto-generated catalog
- Auto-generated API for visualization tools

```
normalized_binned_spikes = ParquetTable(
    name="normalized_binned_spikes",
    uri="s3://neuralake-bucket/spikes",
    partitioning=(
        Partition("implant_id", pa.string()),
        Partition("date", pa.string()),
        Partition("hour", pa.string())
    ),
    partitioning_scheme=PartitioningScheme.HIVE,
    docs_filters=[
        Filter("implant_id", "==", 4595),
        Filter("date", "==", "2024-04-28"),
        Filter("hour", "==", 14),
    ],
)
```

Developers can easily add new tables with custom function definitions

Neuralake Python Client

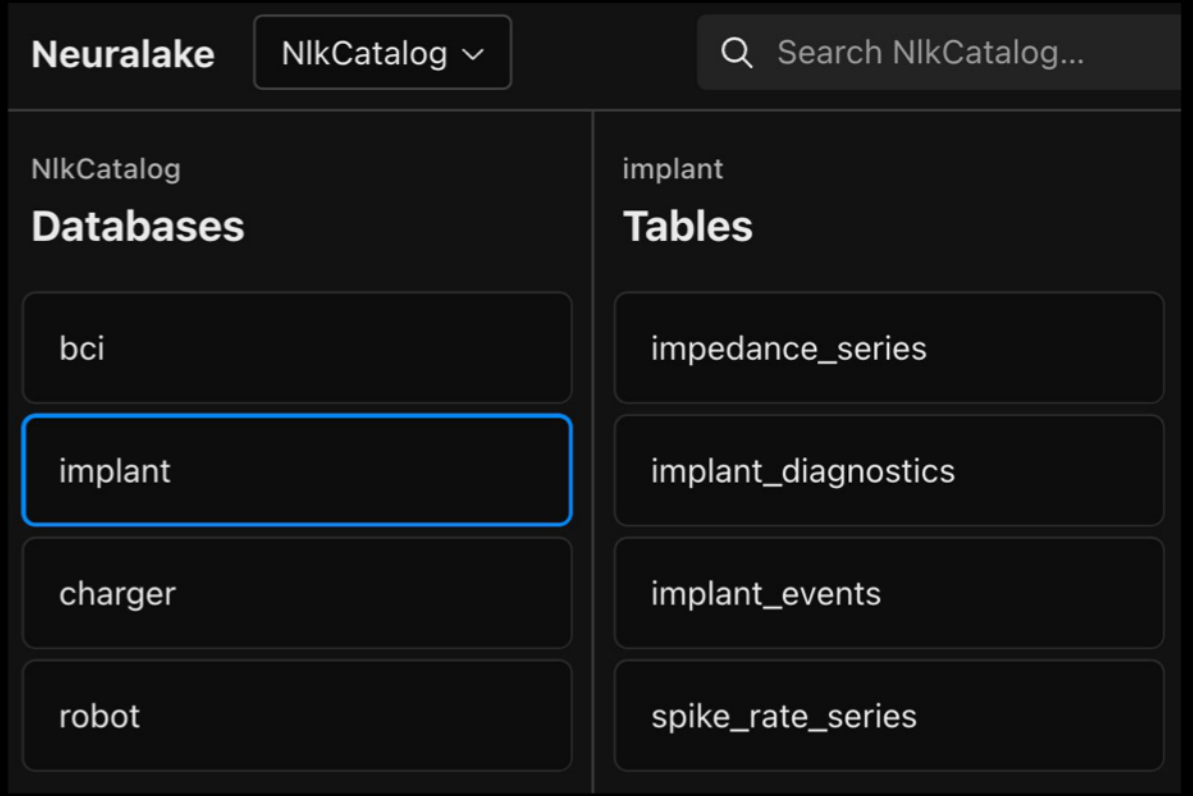
- Catalog, DB, and table structure
- Uniform API to retrieve data as dataframes
- **Flexible syntax for defining tables**
- Auto-generated catalog
- Auto-generated API for visualization tools

```python
@table
def robot_insertions() -> NlkDataFrame:
    """
    Context-driven table for Robot insertions, backed by data from logs.

    Available data includes insertion parameters, locations,
    and notes from data review.

    The LIMS-portion of this data source is refreshed every 30 minutes.
    The Robot logs portion of this data source is refreshed daily.
    """

    logs_lf: NlkDataFrame = robot_logs()
    insertion_notes_lf: pl.LazyFrame = pl.scan_parquet(
        "s3://n7k-neuralake-bucket/insertion-notes/data.parquet"
    )

    return (
        logs_lf.filter(
            pl.col("message_type") == "INSERTION_EVENT",
            pl.col("NLInsertionSummary").is_not_null()
        )
        .select("surgery_id", "timestamp", "NLInsertionSummary")
        .explode("NLInsertionSummary")
        .with_columns(
```

Auto-generated catalog allows for easy browsing.

Neuralake Python Client

- Catalog, DB, and table structure
- Uniform API to retrieve data as dataframes
- Flexible syntax for defining tables
- **Auto-generated catalog**
- Auto-generated API for visualization tools



Code can be copied into JupyterHub with a single click.

Neuralake Python Client

- Catalog, DB, and table structure
- Uniform API to retrieve data as dataframes
- Flexible syntax for defining tables
- **Auto-generated catalog**
- Auto-generated API for visualization tools

implant_diagnostics

Online telemetry collected from implants while they are actively connected to a client.

Copy

```
from neuralake.catalogs import NlkCatalog
from neuralake.core import Filter

df = NlkCatalog.db('implant').table(
    'implant_diagnostics',
    (
        Filter('implant_id', '==', 4595),
        Filter('date', '==', '2024-04-14'),
    ),
)
print(df.collect())
```

Partitions

implant_id	int
date	str

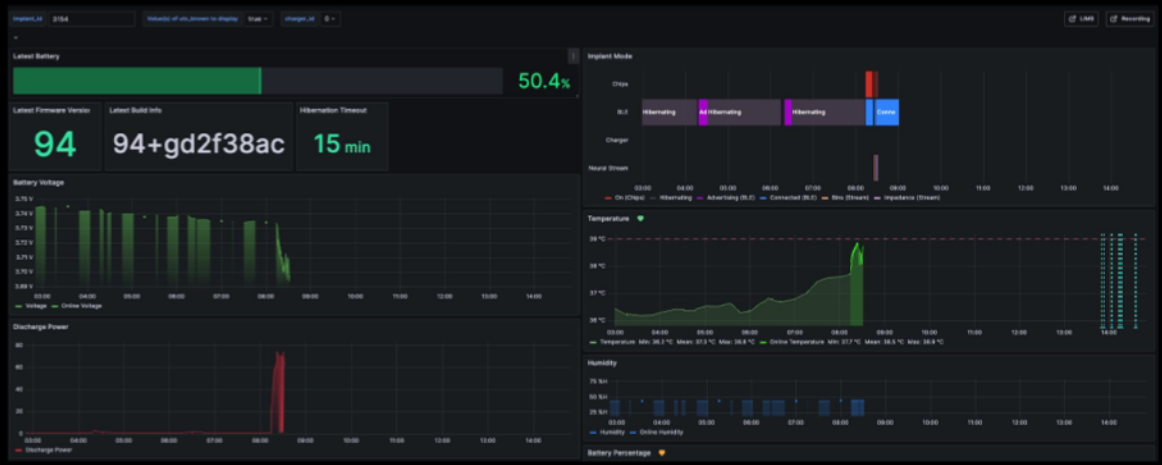
Schema

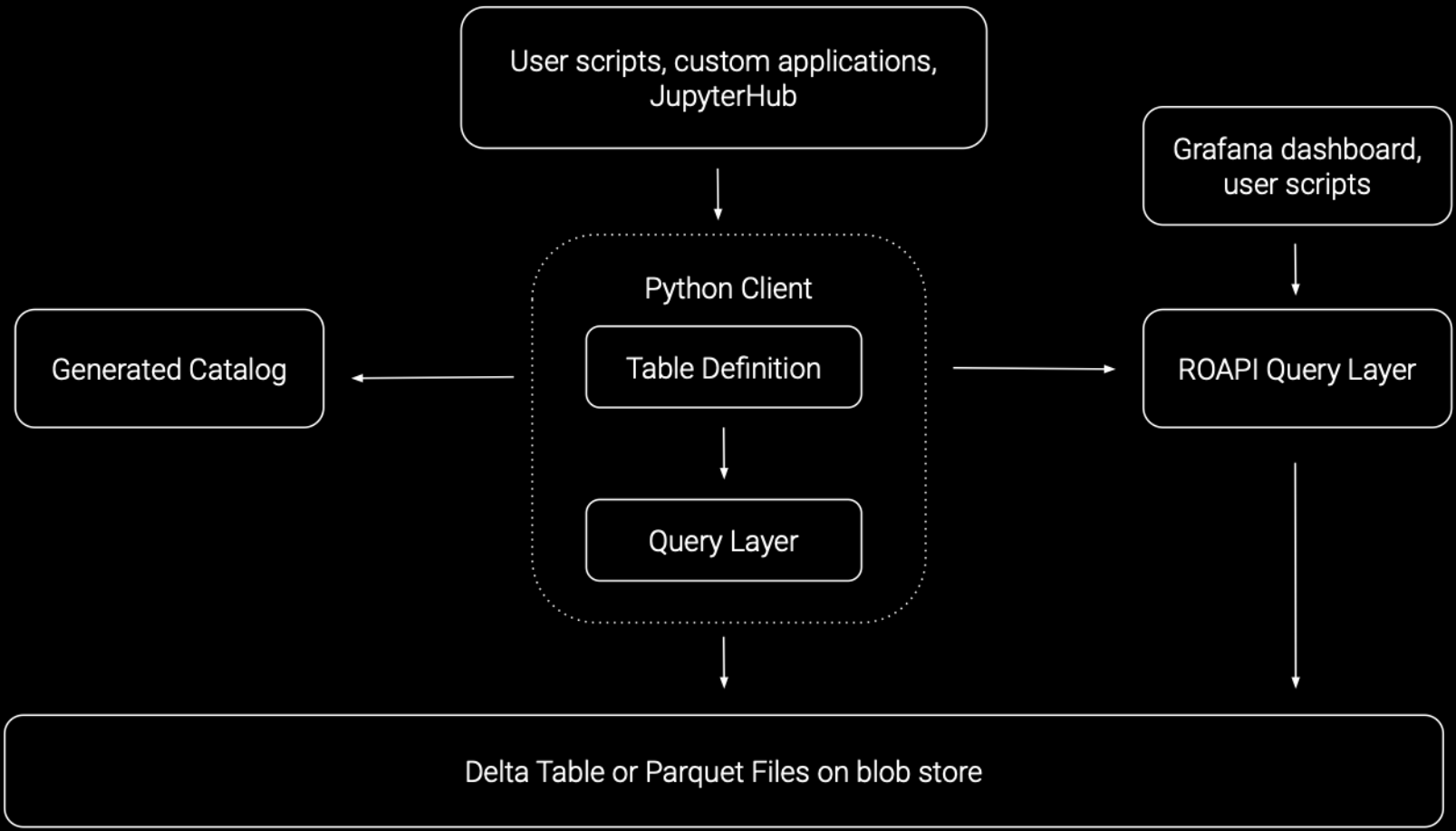
local_ts_us	Datetime(time_unit='ns', time_zone=None)
ticks_since_boot	Int64

Read-only APIs are generated using ROApi, allowing for SQL queries and visualization via tools such as Grafana

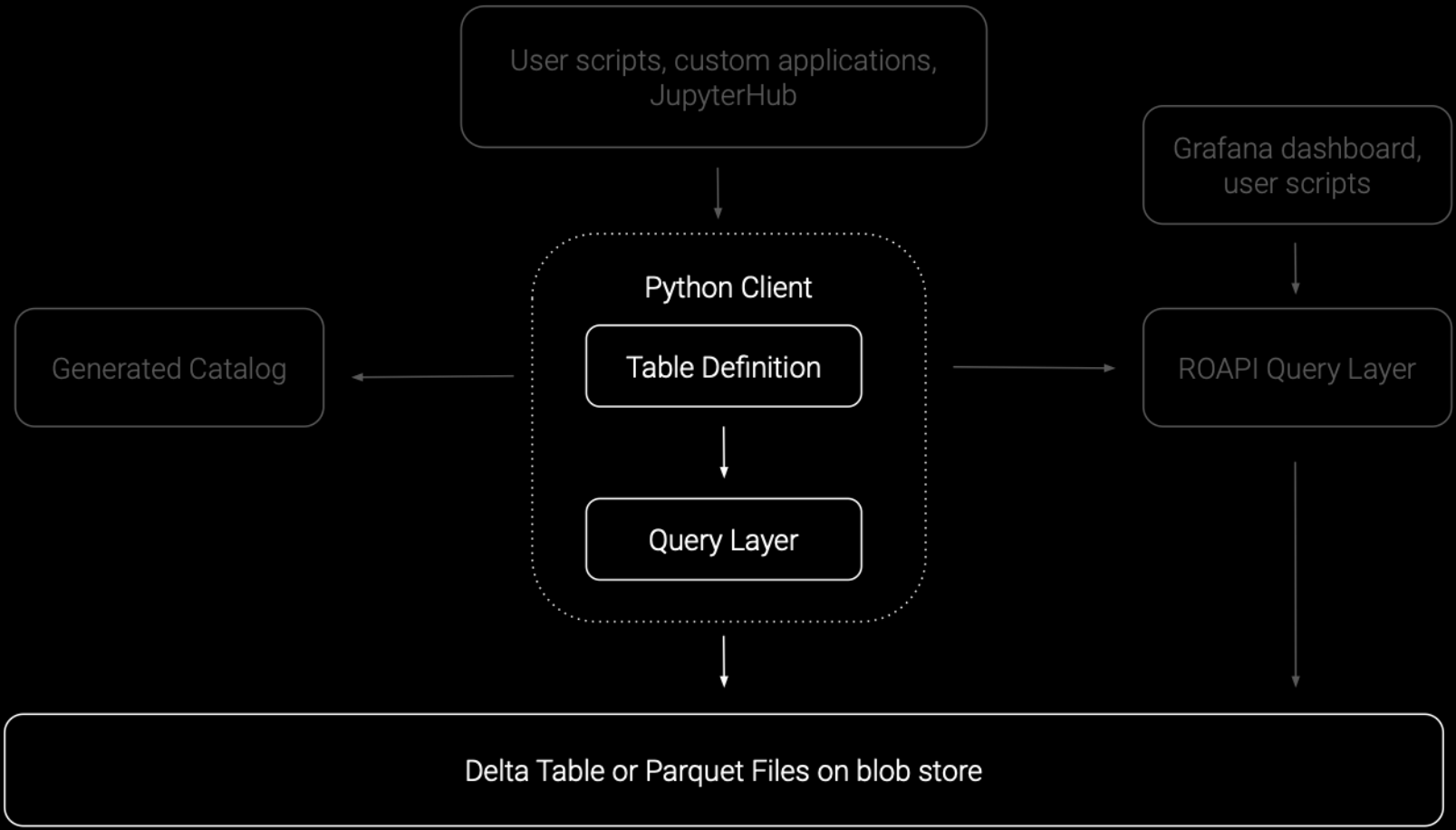
Neuralake Python Client

- Catalog, DB, and table structure
- Uniform API to retrieve data as dataframes
- Flexible syntax for defining tables
- Auto-generated catalog
- **Auto-generated API for visualization tools**

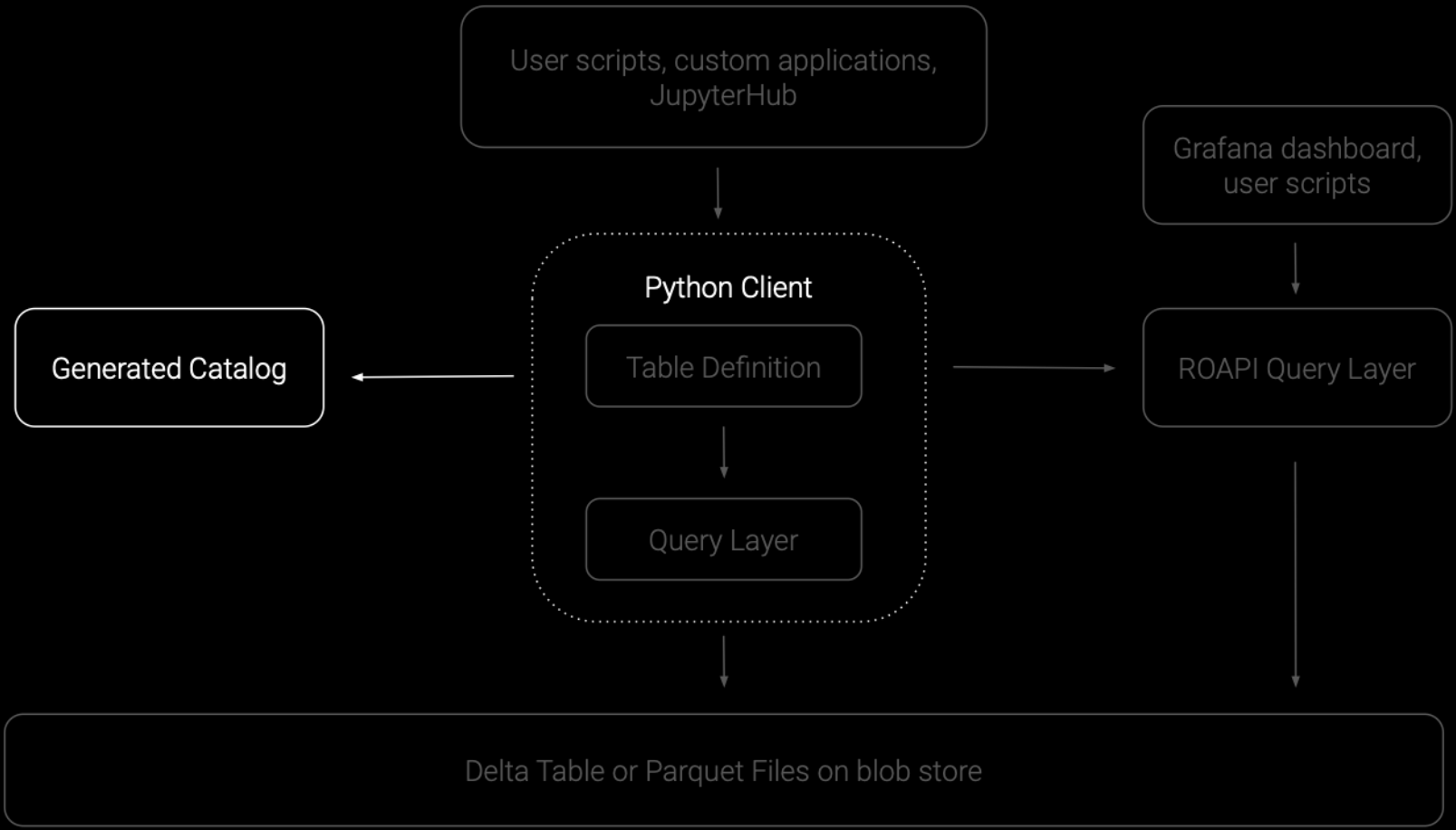



User scripts, custom applications, JupyterHub
Grafana dashboard, user scripts
Python Client
Generated Catalog
Table Definition
ROAPI Query Layer
Query Layer
Delta Table or Parquet Files on blob store

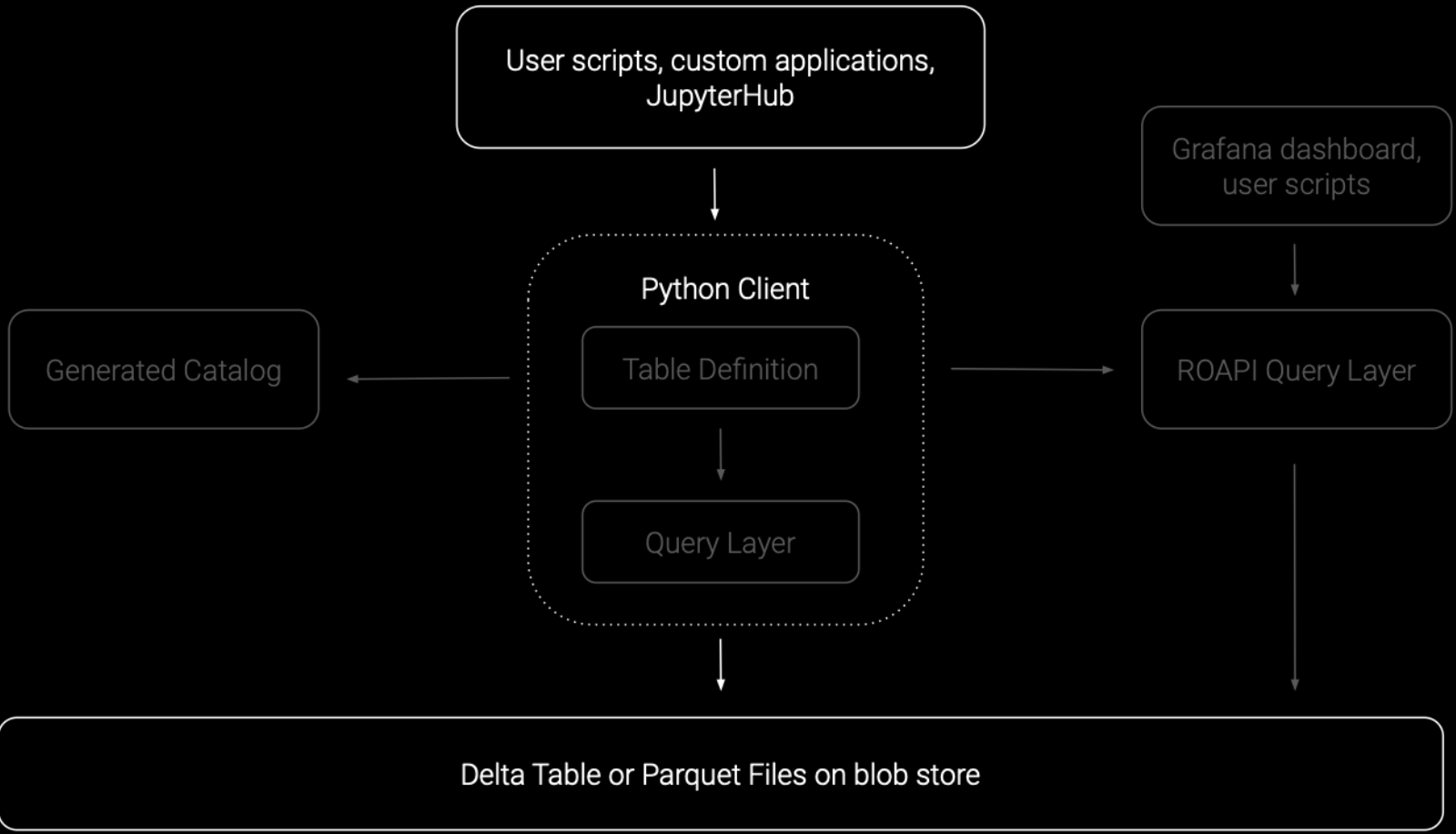
The Python client uses *polars* and *pyarrow* to query delta tables and parquet files. Developers can create tables with a single declarative command or use custom functions.



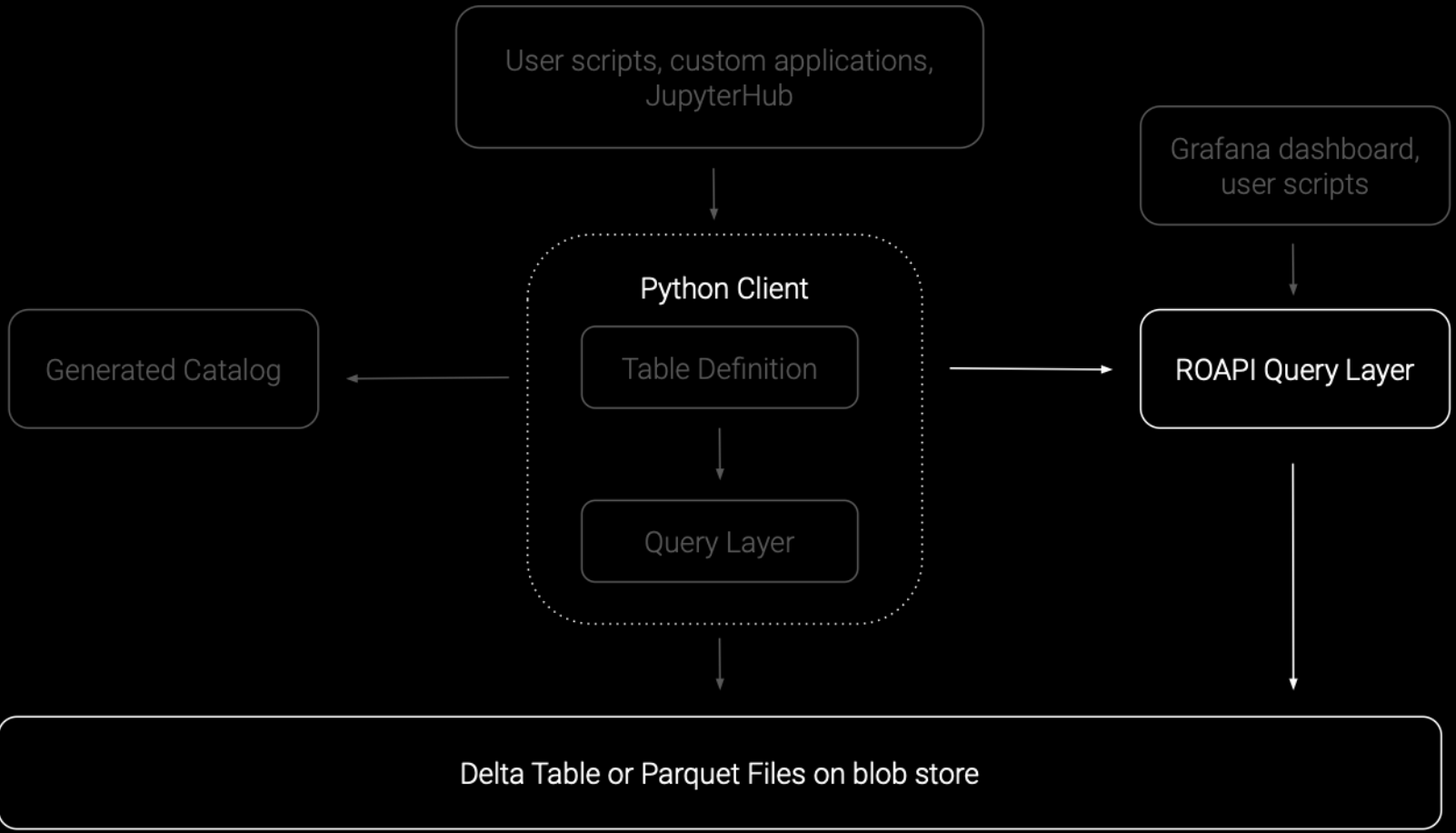
The catalog is auto-generated from the table definitions. Table definitions are serialized to JSON, which generates a stateless web app



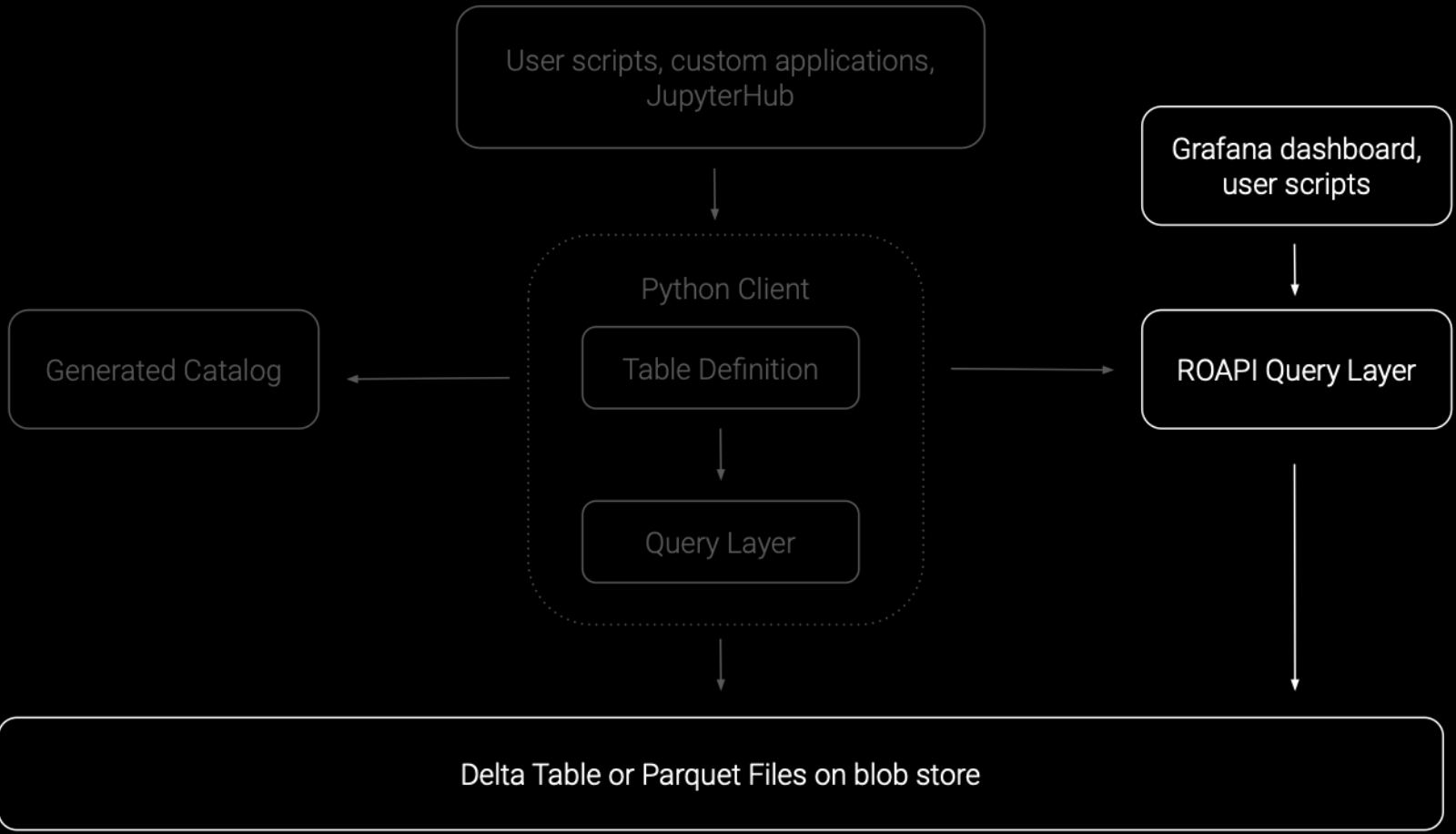
Users can import the *neuralake* library into their scripts and applications.
JupyterHub is loaded with the *neuralake* library.



ROAPI config is generated from table definitions.
ROAPI executes interactive FlightSQL queries with Apache Datafusion. See it on github.com/roapi.



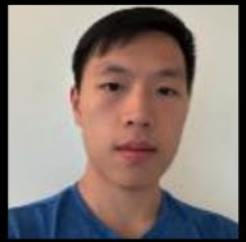
ROAPI can be queried via PostgreSQL clients, SQL over HTTP, or GraphQL.
Neuralake is configured as a Grafana data source for low-code dashboarding.



Conclusion

- **Define tables in code**, auto-generate a catalog, API, and dashboarding tools without needing to maintain a catalog database.
- **Read and write scale down to a laptop** without Java Virtual Machine (JVM) overhead, and scale up to a stateless cluster.
- **Flexible design allows for easy extensions**, e.g. adding offline batch processing and image/video storage.
- **Rust-based systems** such as *polars*, Apache Datafusion, and *delta-rs* allow for **high-performance data access**.
- **Delta Lake allows for ACID transactionality** on blob stores without the overhead of a database server.

The Neuralake team


Peter Ke


Natalie Cygan


Emilienne Repak


Ameer Syedibrahim


Tomas Vancura


QP Hou


Lexi Mattick

“[The Link] has helped me reconnect with the world, my friends, and my family. It's given me the ability to do things on my own again without needing my family at all hours of the day and night.”

— PRIME Study participant

PRIME Study Progress Update

“Y'all are giving me too much, it's like a luxury overload, I haven't been able to do these things in 8 years and now I don't know where to even start allocating my attention.”

— PRIME Study participant

neuralink.com/careers

Q/A