From REST API to Iceberg Lakehouse: ELT with Python, dlt and DuckDB

dltHub

Introduction

- 👋 My name is Marcin. I'm the CTO at dltHub

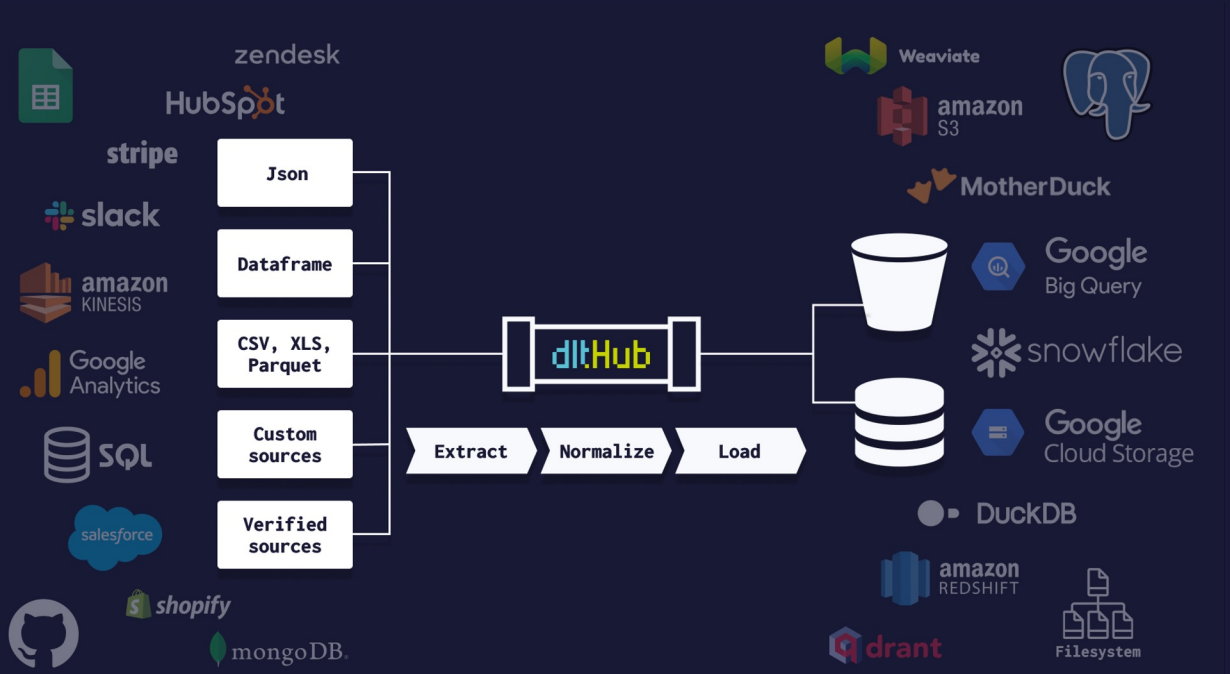
- This talk:
  - What is dlt?
  - Our journey with DuckDB & why we work well together
  - Add "T" to "EL"? Why "T" in Python?
  - Demo

dltHub

What is dlt?

dlt - data loading tool

- an open-source Python library
- automates schema evolution, normalization, and data loading.



dltHub

dlt is code

- Any Python developer can do data engineering tasks

```python
import dlt

@dlt.resource(table_name="foo_data", primary_key="id", write_disposition="merge")
def foo():
    for i in range(10):
        yield {"id": i, "name": f"This is item {i}"}

pipeline = dlt.pipeline(
    pipeline_name="python_data_example",
    destination="duckdb",
)

load_info = pipeline.run(foo)

# print load info and the "foo_data" table as dataframe
print(load_info)
print(pipeline.dataset().foo_data.df())
```

dltHub

dlt + DuckDB: how it started

- Struggling with onboarding and being understood :)
- We see DuckDB in early 2023
- Next week all our docs converted to DuckDB and new destination implemented (using INSERT statement, that was before dlt supported parquet!)
- Week later we are at duckcon#2 to see who builds it!

dltHub

Why we fit together

- Library, not a platform. No backend, no containers.
- Runs everywhere.
- Data first! See and touch data all the time.
- Local workflows.
- How we use DuckDB?
  - Onboarding, docs, education - this is how we grow.
  - Local analytics, local testing
  - Data destination + MotherDuck + DuckLake (coming in days)
  - Data source: fast and robust csv, parquet and json reading
  - NEW! (**dlt.transformation**) Query Engine for cloud storage

dltHub

Why we add “T” to “EL” in Python?

- E(t)L vs. ELT

- Convert Python scripts into robust pipelines with minimal changes
- Keep using pandas, dask, arrow (eager transformations)
- Single tool: Python code. All your libraries available
- Universal schema, query in ibis, narwhals, SQL, SQLGlot…
- Column lineage, annotation propagation

dltHub

Introduction

- 👋 My name is Shreyas. I'm a Developer Advocate at dltHub

- Previous Experience:

  Data Engineer

  Data Scientist

  Data Analyst

dltHub

Today's Takeaways

- Tackling common ELT challenges
- Build and test prod-ready ELT pipelines using dlt and DuckDB
- Transformation of raw data using Ibis, SQL and Arrow
- Load data to Iceberg or Delta Lake tables in local filesystem
- Load data into Iceberg tables in cloud with dlt+ using built-in catalog support

dltHub

Challenges of ELT

Data complexity

- Deeply nested data
- Large datasets


```
{
  "school": "Columbia University",
  "city": "New York City",
  "semester": "Fall 2024",
  "courses": [
    {
      "course_id": "c1",
      "modules": [
        {
          "module_id": "m1",
          "lessons": [
            { "lesson_id": "l1", "title": "Introduction" },
            { "lesson_id": "l2", "title": "Getting Started" }
          ]
        },
        {
          "module_id": "m2",
          "lessons": [
            { "lesson_id": "l3", "title": "Advanced Topics" }
          ]
        }
      ]
    },
    {
      "course_id": "c2",
      "modules": [
        {
          "module_id": "m3",
          "lessons": [
            { "lesson_id": "l4", "title": "Basics" }
          ]
        }
      ]
    }
  ]
}
```

Challenges of ELT

Transformation complexity

- Hard to reuse SQL scripts-tied to schemas/pipelines
- Limited Support for Non-SQL Logic

dltHub

Challenges of ELT

Cost management

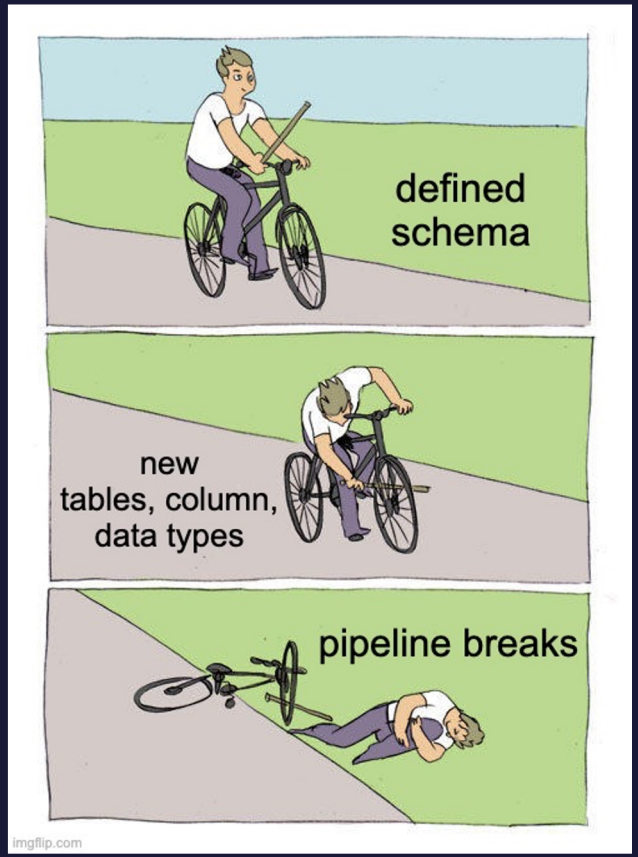
- Compute-intensive transforms
- Repeated transformations during testing

dltHub

Challenges of ELT

Governance and observability

- Schema evolution
- Pipeline trace-monitor, debug, metrics
- Track PII, lineage, contracts



dltHub

Challenges of ELT

Developer experience

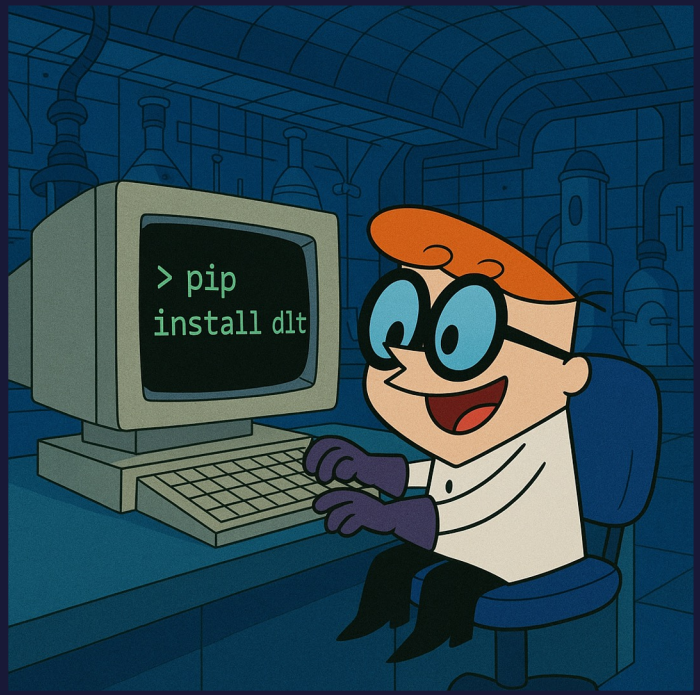
- ETL tools are declarative
- Hard to test SQL functions-no breakpoints, print
- No local dev Loop-pipelines tested in prod env

dltHub

ELT, but without the pain

Local-first Dev experience

- dlt is imperative, pythonic
- Extract, load, transform (DuckDB engine)-local FS as Iceberg, Delta Lake or Parquet
- Test pipelines locally-no cost



dltHub

ELT, but without the pain

Schema and Data Contract

```python
# Define the API resource for Rick and Morty Characters
@dlt.resource(name="characters",
              max_table_nesting=3,
              primary_key="id",
              write_disposition={"disposition": "merge", "strategy": "scd2"},
              columns={"episode": {"data_type": "json"}},
              schema_contract={"tables": "evolve", "columns": "freeze", "data_type": "discard_row"} )
def characters(last_created=dlt.sources.incremental("created", initial_value="2010-01-01T00:00:00Z")):
```

dltHub

ELT, but without the pain

Governance and observability

- dlt trace-detailed execution record of ELT
- apply_hints() to tag PII fields, enforce constraints
- Metadata flows automatically through ELT

dltHub

ELT, but without the pain

Seamless transition to production

```python
#define pipeline to load jaffle data to Snowflake
sf_pipeline = dlt.pipeline(
        pipeline_name="jaffle_pipeline",
        destination='snowflake',
        dataset_name="jaffle_dataset"
    )
load_info_sf = sf_pipeline.run(jaffle_shop())
print(load_info_sf)
```

```python
pipeline = dlt.pipeline(
    pipeline_name="smart_plug",
    destination="duckdb",
    dataset_name="smart_plug_data",
)
```

dltHub

ELT, but without the pain

Flexible transformations

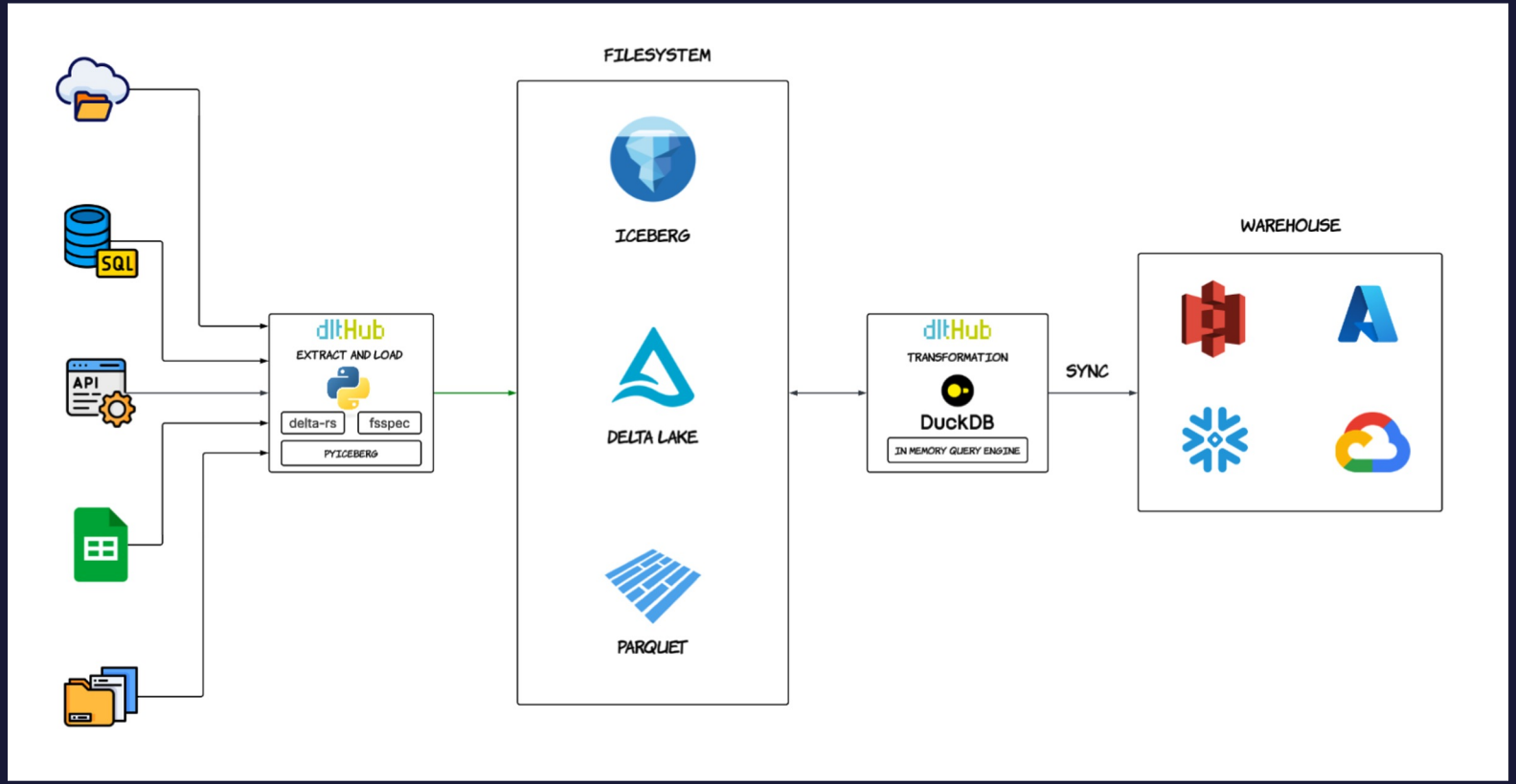
```python
#this transformation uses Ibis expression
@dlt.transformation(write_disposition="replace")
def orders_per_store(dataset: dlt.Dataset) -> Any:
    orders = dataset["orders"]
    stores = dataset["stores"]
    return (
      orders.join(stores, orders.store_id == stores.id)
      .group_by(stores.name)
      .aggregate(order_count=orders.id.count())
  )
```

```python
#this transformation uses arrow tables
@dlt.transformation(write_disposition="replace")
def ordered_stores(dataset: dlt.Dataset) -> Any:
  stores = dataset.stores.arrow()
  sorted_stores = stores.sort_by([("name", "ascending")])
  yield sorted_stores.slice(0, 5)
```

```python
#this transformation uses SQL expression
@dlt.transformation(write_disposition="replace")
def ordered_customers(dataset: dlt.Dataset) -> Any:
    customers_table = dataset("SELECT * FROM customers ORDER BY name LIMIT 5")
    return customers_table
```

dltHub

OSS ELT Architecture



dltHub

End to end data pipeline using dlt and DuckDB demo

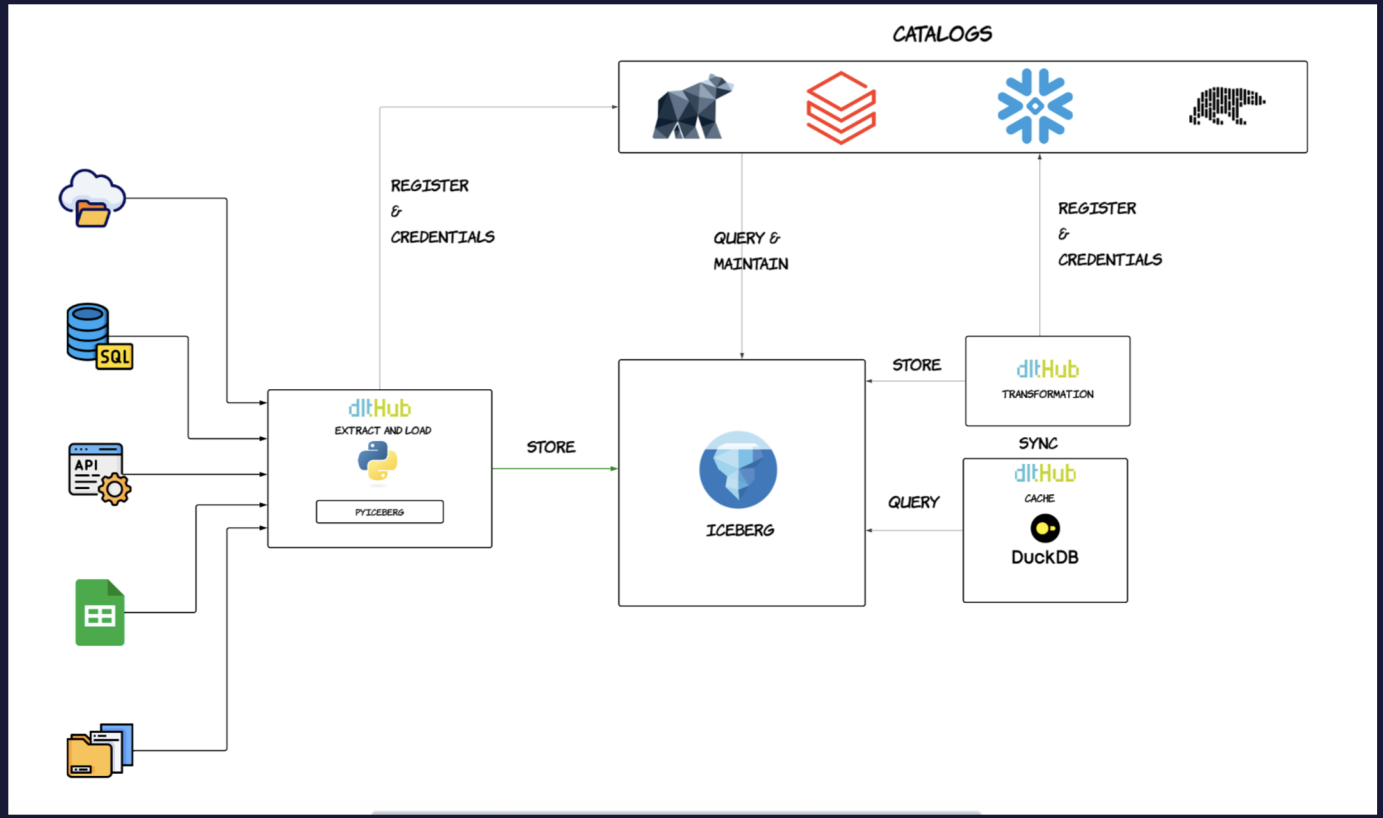
dltHub

Implementation status: EXPERIMENTAL

- Still incorporating friendly testing feedback.
- Part of the library, user interface not public
- Link to our demo:
- Read the docs: https://dlthub.com/docs/general-usage/transformations

dltHub

dlt+ ELT Architecture



dltHub

Thank You!
Any Questions?

dltHub