Using dlt to move data from DuckDB to ClickHouse®

Using dlt to move data from DuckDB to ClickHouse®
From Pocket Calculator to Kubernetes in Prod

Altinity dltHub

Introductions

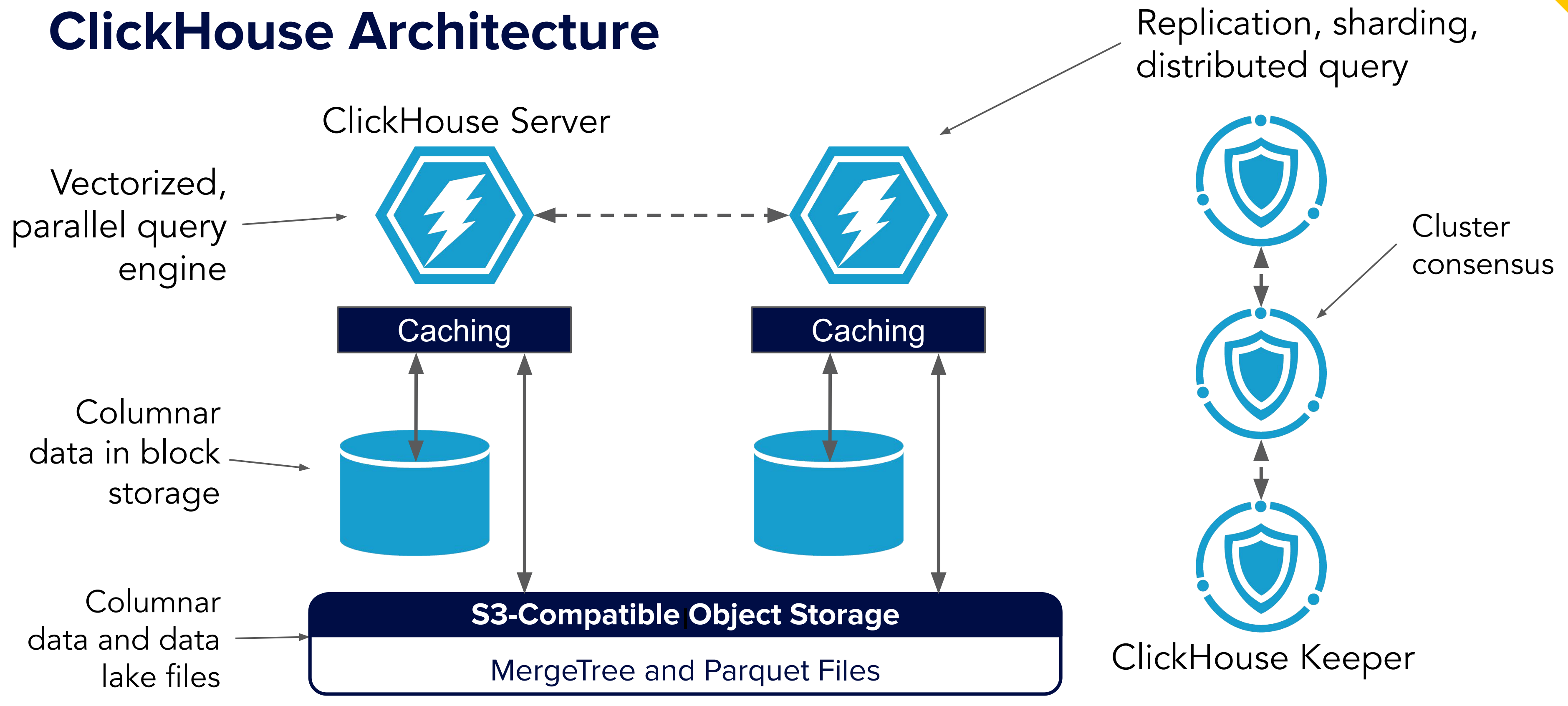
Josh Lee
Open Source Advocate @ Altinity

Elvis Kahoro
DevX @ dltHub

ALTINITY

dltHub

ClickHouse Architecture

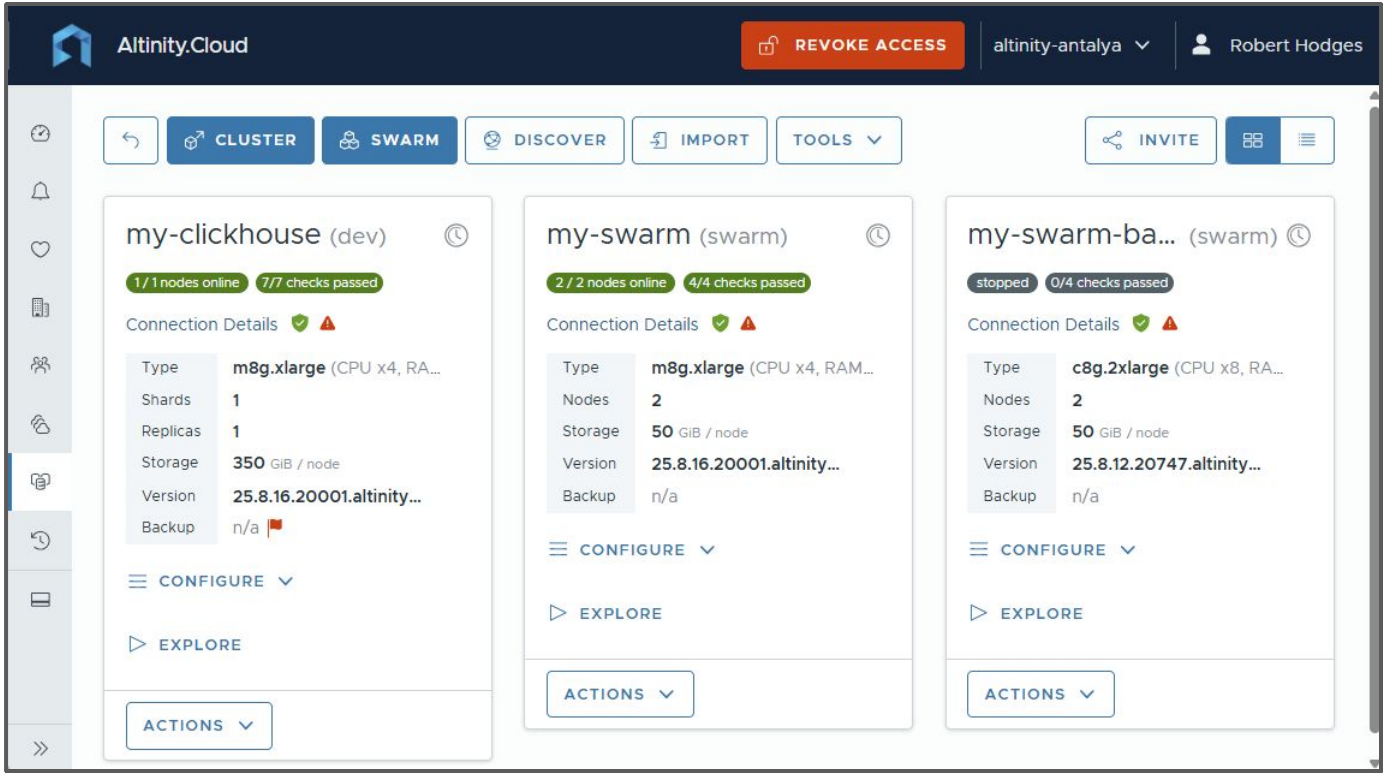

Replication, sharding, distributed query
ClickHouse Server
Vectorized, parallel query engine
Caching
Caching
Cluster consensus
Columnar data in block storage
Columnar data and data lake files
S3-Compatible Object Storage
MergeTree and Parquet Files
ClickHouse Keeper

ALTINITY®

Run Open Source ClickHouse® Better

Altinity.Cloud   Enterprise Support

Altinity Cloud (Managed, BYOC, BYOK)



my-clickhouse (dev)

1 / 1 nodes online · 7/7 checks passed

Connection Details

Type	m8g.xlarge (CPU x4, RA...
Shards	1
Replicas	1
Storage	350 GiB / node
Version	25.8.16.20001.altinity...
Backup	n/a

CONFIGURE · EXPLORE · ACTIONS

my-swarm (swarm)

2 / 2 nodes online · 4/4 checks passed

Connection Details

Type	m8g.xlarge (CPU x4, RAM...
Nodes	2
Storage	50 GiB / node
Version	25.8.16.20001.altinity...
Backup	n/a

CONFIGURE · EXPLORE · ACTIONS

my-swarm-ba... (swarm)

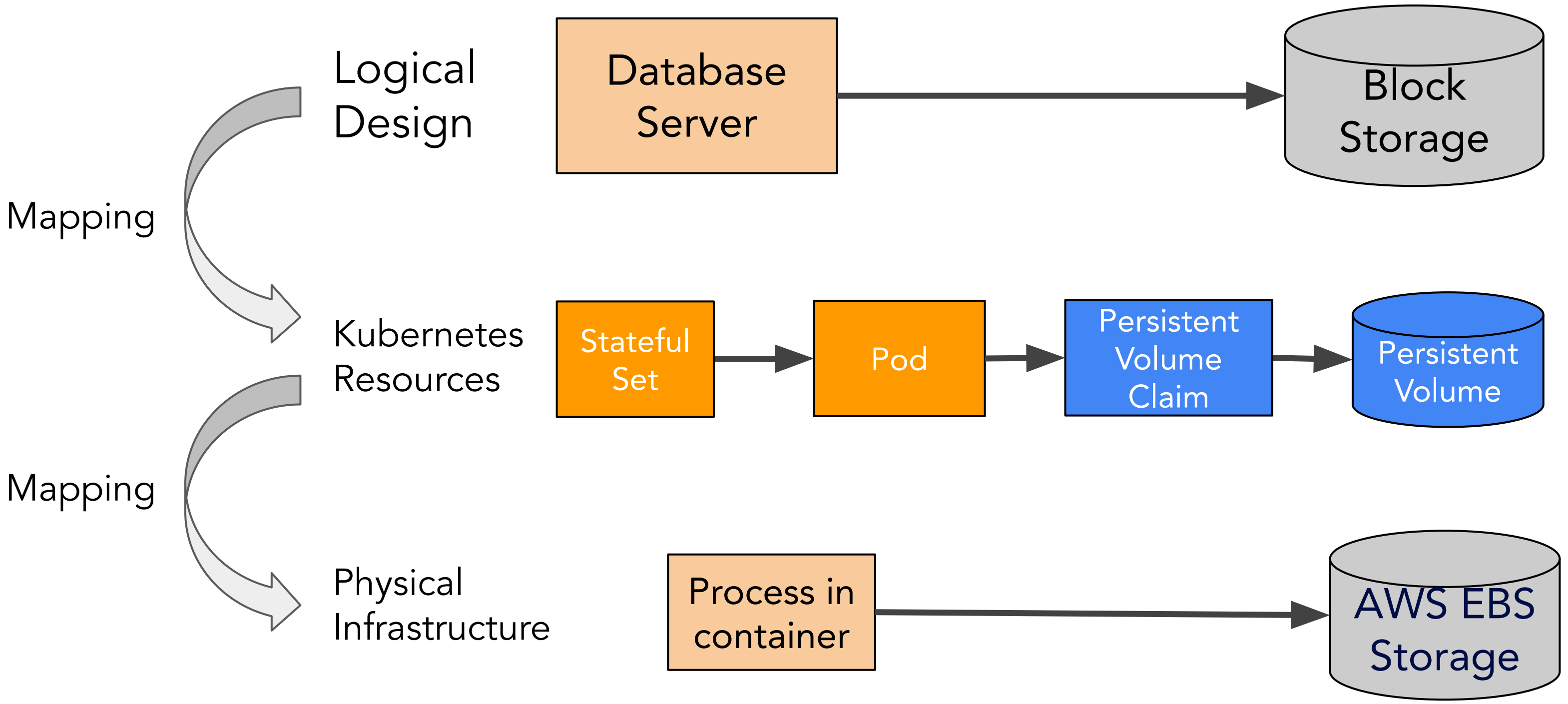
stopped · 0/4 checks passed

Connection Details

Type	c8g.2xlarge (CPU x8, RA...
Nodes	2
Storage	50 GiB / node
Version	25.8.12.20747.altinity...
Backup	n/a

CONFIGURE · EXPLORE · ACTIONS

The Altinity Operator for ClickHouse



Altinity

TRUSTED BY

flatiron. PostHog harness Taktile ARTSY SpareBank 1 SR-BANK

6M+
PyPi Downloads

8,000+
OSS companies in production

600+
Snowflake customers in production

OPEN SOURCE

pip install dlt and go

dlt (data load tool) is the most popular production-ready open source Python library for moving data. It loads data from various and often messy data sources into well-structured, live datasets.

Unlike other non-Python solutions, with dlt library, there's no need to use any backends or containers. We do not replace your data platform, deployments, or security models. Simply import it into your favorite AI code editor, or add it to your Jupyter Notebook. You can load data from any source that produces Python data structures, including APIs, files, databases, and more.

Get started

Cloud storage or files | REST APIs | **SQL databases** | Python data structures

```python
from dlt.sources.sql_database import sql_database

source = sql_database(
    "mysql+pymysql://rfamro@mysql-rfam-public.ebi.ac.uk:4497/Rfam"
)

pipeline = dlt.pipeline(
    pipeline_name="sql_database_example",
    destination="duckdb",
    dataset_name="sql_data",
)

pipeline.run(source)
```

Data loading - Examples

REST API >> BigQuery

- Salesforce / Hubspot / CRM >> Snowflake - RevOps

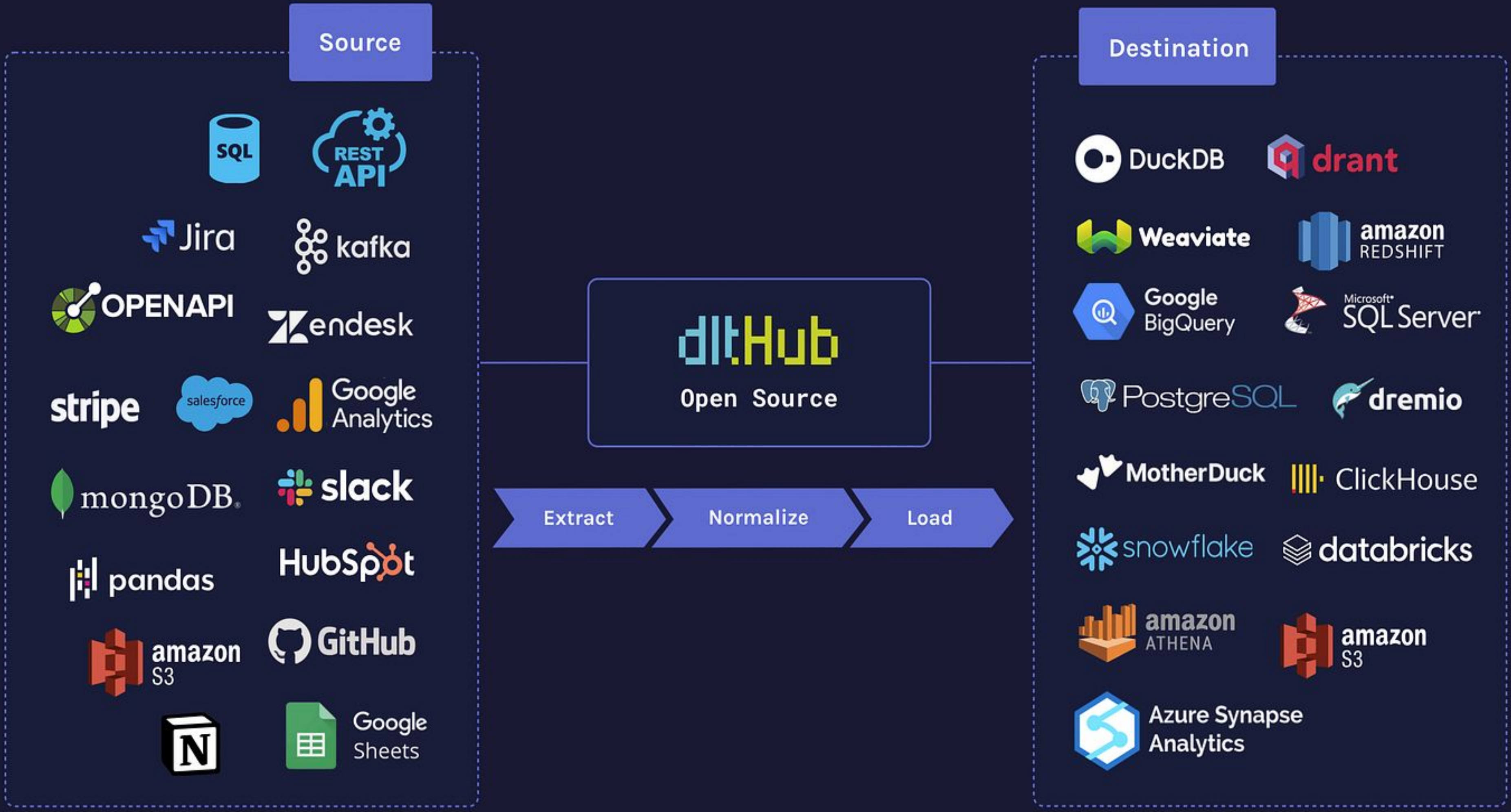
SQL Source

- Snowflake (subset) >> Local (DuckDB) - exploration
- Local (DuckDB) >> Snowflake staging - test before commit

Filesystem (S3 / local)

- CSV, JSONL, Parquet >> Iceberg - backups / archive

Custom sources


Source
SQL
REST API
Jira
kafka
OPENAPI
Zendesk
stripe
salesforce
Google Analytics
mongoDB.
slack
pandas
HubSpot
amazon S3
GitHub
N
Google Sheets
dltHub
Open Source
Extract
Normalize
Load
Destination
DuckDB
Qdrant
Weaviate
amazon REDSHIFT
Google BigQuery
Microsoft SQL Server
PostgreSQL
dremio
MotherDuck
ClickHouse
snowflake
databricks
amazon ATHENA
amazon S3
Azure Synapse Analytics

Alternatives - Managed data platform

Expensive

- Pay per row
- Extra for faster syncs i.e. 15m >> 1m
- Connectors
    - Extra for access to enterprise connectors
    - 25 connectors make up 90% of revenue

Alternatives - Managed data platform

Expensive

- Pay per row
- Extra for faster syncs i.e. 15m >> 1m
- Connectors
    - Extra for access to enterprise connectors
    - 25 connectors make up 90% of revenue

Rigid

- Subject to the limits and config of the platform

Privacy & data governance

- PII - Fintech, Health, etc.

Alternatives - Custom Python scripts

Inevitably need workarounds when

- Missing connector
- Need more control

Engineers start building custom pipelines

- Just want to solve a problem or answer a question

Alternatives - Custom Python scripts

Inevitably need workarounds when

- Missing connector
- Need more control

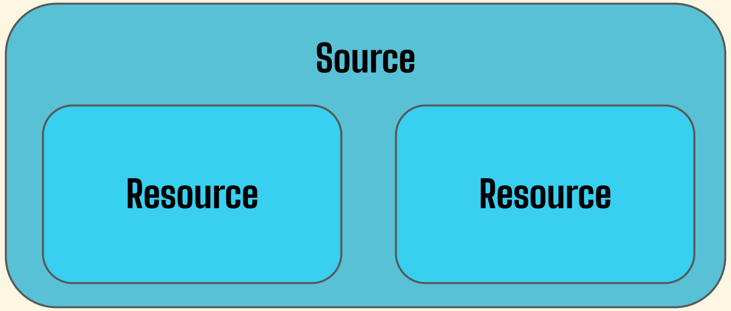
Engineers start building custom pipelines

- Just want to solve a problem or answer a question
- Not interested in building a data platform
  - Data normalization
  - Schema evolution
  - Data contracts
  - Incremental loading
  - Execution engine and infrastructure > Spark, Kubernetes
- FAANG >> Platform teams >> What about the rest of us?

dltHub - How

```
p = dlt.pipeline(
    pipeline_name="9zero_demo",
    destination="snowflake",
    dataset_name="dataset",
)
p.run(data_source())
```

A **pipeline** sends a **source** to a **destination**



```python
import pandas as pd


df = pd.DataFrame({
    "order_id": [1, 2, 3],
    "customer_id": [1, 2, 3],
    "order_amount": [100.0, 200.0, 300.0],
})
p = dlt.pipeline("orders_pipeline", destination="snowflake")
p.run(df, table_name="orders")
```

DataFrames

- Passed in directly
- Pandas, Polars + Arrow

```
import dlt
from dlt.sources.filesystem import filesystem, read_parquet

filesystem_resource = filesystem(
  bucket_url="s3://my-bucket/files",
  file_glob="**/*.parquet",
  incremental=dlt.sources.incremental("modification_date")
)
filesystem_pipe = filesystem_resource | read_parquet()

# We load the data into the table_name table
p = dlt.pipeline(pipeline_name="my_pipeline", destination="duckdb")
p.run(filesystem_pipe.with_name("table_name"))
```

Filesystem

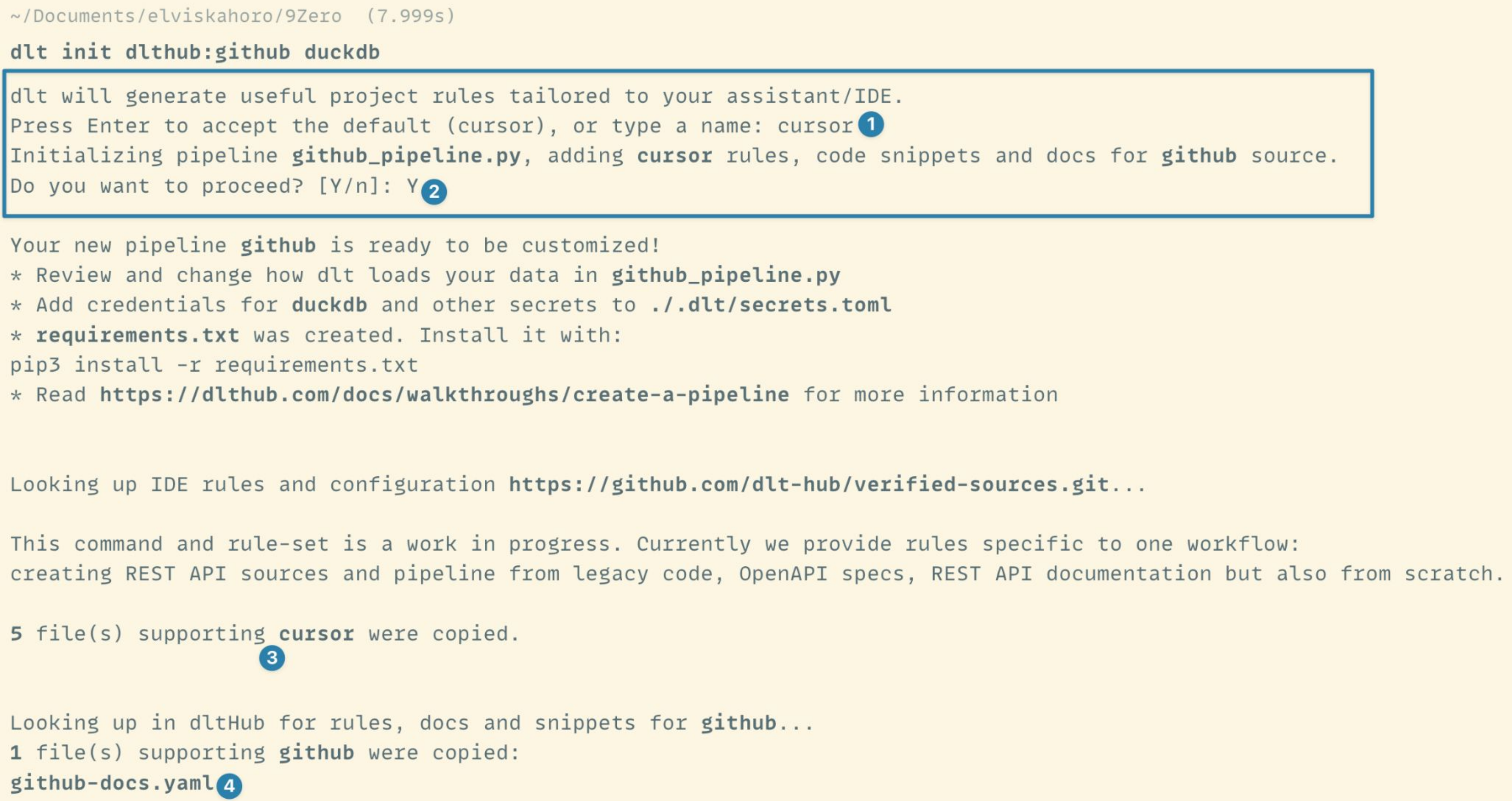
- bucket_url which can be local
- Configure what column to use as our incremental variable
- JSONL, CSV, Parquet

```python
@dlt.source
def github_rest_api_source(access_token: str = dlt.secrets.value):
    """Define dlt resources from REST API endpoints."""
    config: RESTAPIConfig = {
        "client": {
            "base_url": "https://api.github.com",
            "auth": {},
            "headers": {},
        },
        "resource_defaults": {
            "endpoint": {
                "params": {
                    "per_page": 30,
                },
            },
        },
        "resources": [
            {
                "name": "issues",
                "endpoint": {
                    "path": "/repos/warpdotdev/warp/issues",
                    "method": "GET",
                },
            },
        ],
    }
    yield from rest_api_resources(config)
```

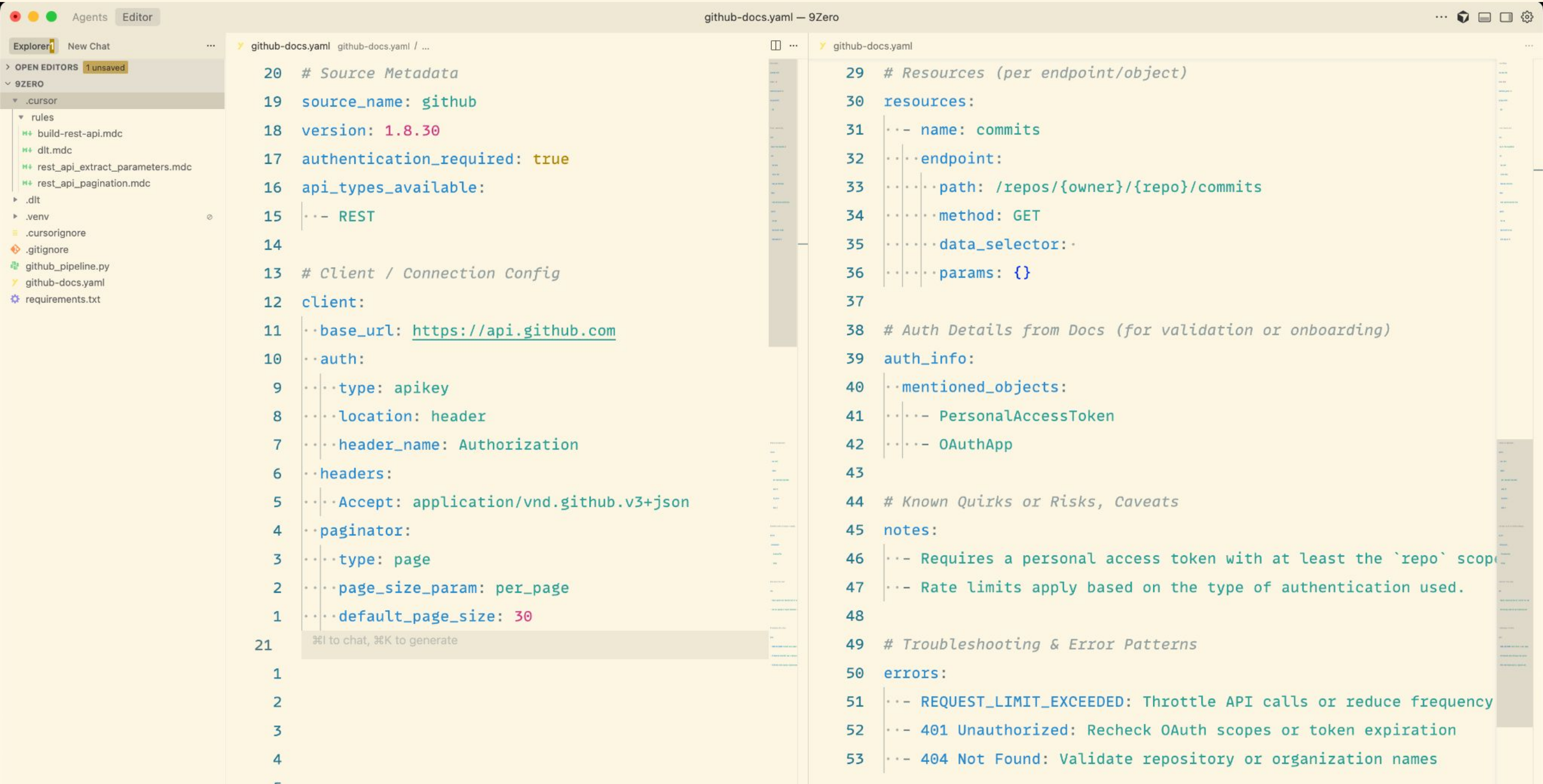
REST API

- Client
- Params
- Resources
  - Name
  - Path
  - Method

DEMO

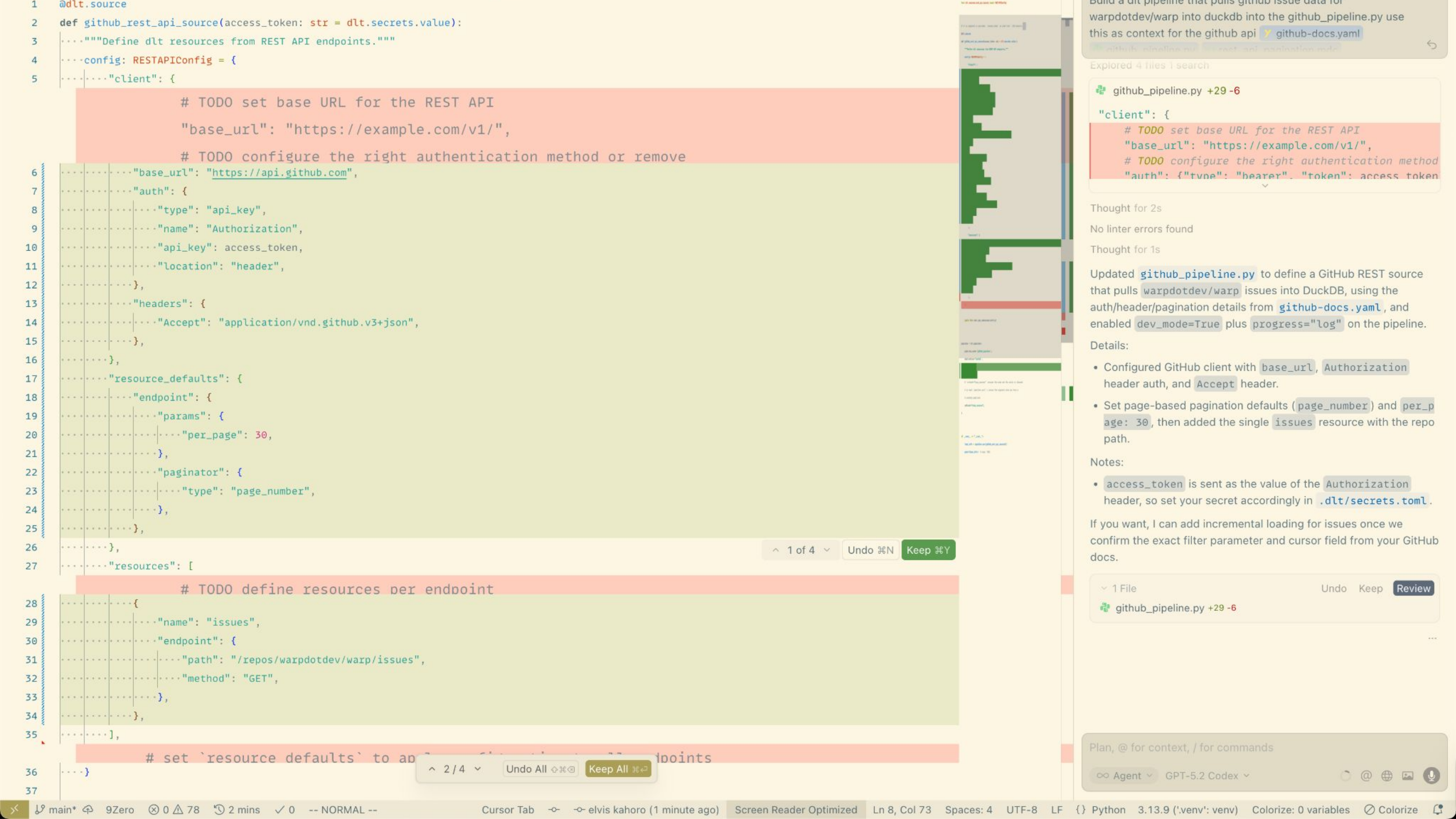


```
~/Documents/elviskahoro/9Zero  (7.999s)
dlt init dlthub:github duckdb
dlt will generate useful project rules tailored to your assistant/IDE.
Press Enter to accept the default (cursor), or type a name: cursor 1
Initializing pipeline github_pipeline.py, adding cursor rules, code snippets and docs for github source.
Do you want to proceed? [Y/n]: Y 2

Your new pipeline github is ready to be customized!
* Review and change how dlt loads your data in github_pipeline.py
* Add credentials for duckdb and other secrets to ./.dlt/secrets.toml
* requirements.txt was created. Install it with:
pip3 install -r requirements.txt
* Read https://dlthub.com/docs/walkthroughs/create-a-pipeline for more information



Looking up IDE rules and configuration https://github.com/dlt-hub/verified-sources.git...

This command and rule-set is a work in progress. Currently we provide rules specific to one workflow:
creating REST API sources and pipeline from legacy code, OpenAPI specs, REST API documentation but also from scratch.

5 file(s) supporting cursor were copied.
                    3


Looking up in dltHub for rules, docs and snippets for github...
1 file(s) supporting github were copied:
github-docs.yaml 4
```



```yaml
# Source Metadata
source_name: github
version: 1.8.30
authentication_required: true
api_types_available:
  - REST

# Client / Connection Config
client:
  base_url: https://api.github.com
  auth:
    type: apikey
    location: header
    header_name: Authorization
  headers:
    Accept: application/vnd.github.v3+json
  paginator:
    type: page
    page_size_param: per_page
    default_page_size: 30
```

```yaml
# Resources (per endpoint/object)
resources:
  - name: commits
    endpoint:
      path: /repos/{owner}/{repo}/commits
      method: GET
      data_selector: 
      params: {}

# Auth Details from Docs (for validation or onboarding)
auth_info:
  mentioned_objects:
    - PersonalAccessToken
    - OAuthApp

# Known Quirks or Risks, Caveats
notes:
  - Requires a personal access token with at least the `repo` scope
  - Rate limits apply based on the type of authentication used.

# Troubleshooting & Error Patterns
errors:
  - REQUEST_LIMIT_EXCEEDED: Throttle API calls or reduce frequency
  - 401 Unauthorized: Recheck OAuth scopes or token expiration
  - 404 Not Found: Validate repository or organization names
```



```
@dlt.source
def github_rest_api_source(access_token: str = dlt.secrets.value):
    """Define dlt resources from REST API endpoints."""
    config: RESTAPIConfig = {
        "client": {
            # TODO set base URL for the REST API
            "base_url": "https://example.com/v1/",
            # TODO configure the right authentication method or remove
            "base_url": "https://api.github.com",
            "auth": {
                "type": "api_key",
                "name": "Authorization",
                "api_key": access_token,
                "location": "header",
            },
            "headers": {
                "Accept": "application/vnd.github.v3+json",
            },
        },
        "resource_defaults": {
            "endpoint": {
                "params": {
                    "per_page": 30,
                },
                "paginator": {
                    "type": "page_number",
                },
            },
        },
        "resources": [
            # TODO define resources per endpoint
            {
                "name": "issues",
                "endpoint": {
                    "path": "/repos/warpdotdev/warp/issues",
                    "method": "GET",
                },
            },
        ],
        # set `resource defaults` to ap                      dpoints
    }
```

Build a dlt pipeline that pulls github issue data for warpdotdev/warp into duckdb into the github_pipeline.py use this as context for the github api `github-docs.yaml`

Explored 4 files 1 search

github_pipeline.py +29 -6

```
"client": {
    # TODO set base URL for the REST API
    "base_url": "https://example.com/v1/",
    # TODO configure the right authentication method
    "auth": {"type": "bearer", "token": access_token
```

Thought for 2s

No linter errors found

Thought for 1s

Updated `github_pipeline.py` to define a GitHub REST source that pulls `warpdotdev/warp` issues into DuckDB, using the auth/header/pagination details from `github-docs.yaml`, and enabled `dev_mode=True` plus `progress="log"` on the pipeline.

Details:

- Configured GitHub client with `base_url`, `Authorization` header auth, and `Accept` header.
- Set page-based pagination defaults (`page_number`) and `per_page: 30`, then added the single `issues` resource with the repo path.

Notes:

- `access_token` is sent as the value of the `Authorization` header, so set your secret accordingly in `.dlt/secrets.toml`.

If you want, I can add incremental loading for issues once we confirm the exact filter parameter and cursor field from your GitHub docs.

1 File
Undo
Keep
Review
github_pipeline.py +29 -6
Plan, @ for context, / for commands
Agent
GPT-5.2 Codex

```
main*  9Zero  0 ⚠ 78  2 mins  ✓ 0  -- NORMAL --  Cursor Tab  elvis kahoro (1 minute ago)  Screen Reader Optimized  Ln 8, Col 73  Spaces: 4  UTF-8  LF  {} Python  3.13.9 ('.venv': venv)  Colorize: 0 variables  Colorize
```

Browse over 10,100 LLM-native scaffoldings for dlt pipelines

Filter By

Category

[✓ CRM & Sales]
☐ Customer Support
☐ Operations & Logistics
☐ Project Management
☐ Marketing & Email
☐ E-commerce & Payments
☐ Communication & Messaging
☐ Analytics & Tracking
☐ Social Media & Advertising
☐ Development & DevOps
☐ HR & Workforce Management
☐ Cloud Services & Infrastructure
☐ Finance & Accounting

Clear All Filters

Search LLM-native scaffoldings...

‹ 1 2 3 ... 6 › Showing 1-9 of 47 results

Activecampaign

API Overview

Source

Affinity

Introduction – Affinity API Reference

Source

AgileCRM

Source

Apptivo

Source

Avid CRM

Avid CRM API Documentation

Source

Capsule-crm

Capsule API - Capsule API

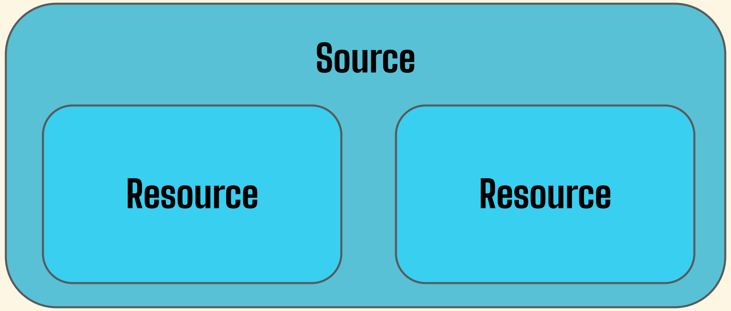
Source

dltHub - How

```python
@dlt.resource
def customers():
    """Iterate over customers in CRM"""
    yield from get_customers()


@dlt.resource
def companies():
    """Iterate over companies in CRM"""
    yield from get_companies()


@dlt.source
def crm():
    """Entities registered in CRM"""
    return [customers(), companies()]
```

A **resource** is the unit of data input

A **source** is a logical grouping of **resources**



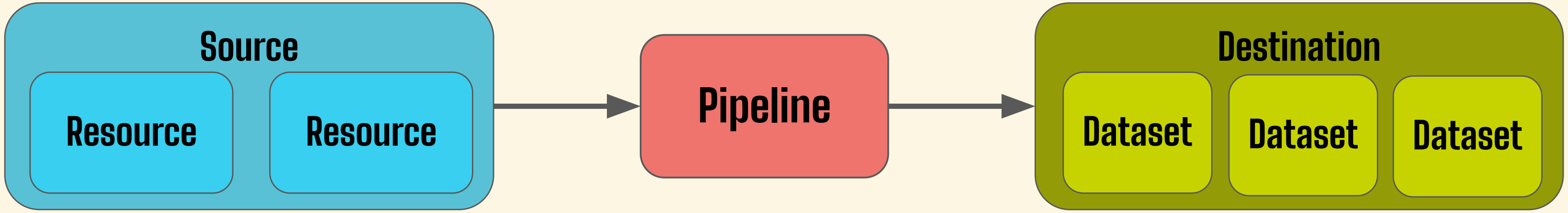
dltHub - How

```
p = dlt.pipeline(
    pipeline_name="9zero_demo",
    destination="snowflake",
    dataset_name="dataset",
)
p.run(data_source())
```

A **resource** is the unit of data input

A **source** is a logical grouping of **resources**

A **pipeline** sends a **source** to a **destination**



Thank you!
Questions?

https://altinity.com/slack

https://dub.sh/dlthub-education

https://dub.sh/dlthub-slack