# Datalake souverain : dbt-duckdb dans k8s

26 Mars 2026

Antoine GIRAUD et Cédric OLIVIER

Data Days Lille 2026

# Présentations »

Antoine GIRAUD

ippon

Cédric OLIVIER

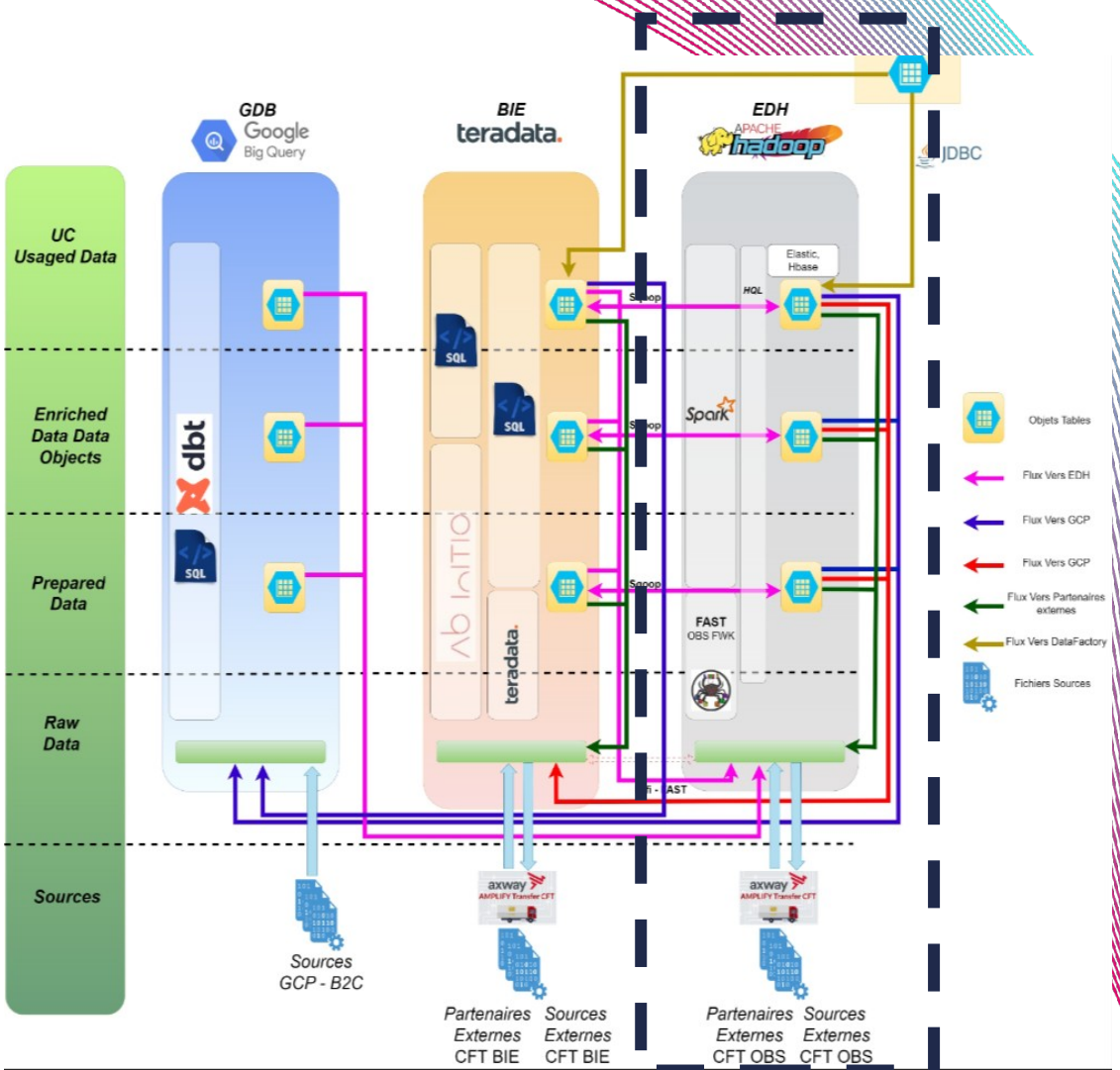
orange™

# 01

# Contexte

Rénovation de notre patrimoine data

# Architecture actuelle



# Nos données bientôt SDF

**Nos constats :**

Sortie du datacenter très proche

Complexité croissante

Satisfaction utilisateur en baisse

**Le besoin:**

Continuité de service

Réduire nos coûts

Simplifier

Harmoniser nos méthodes de travail

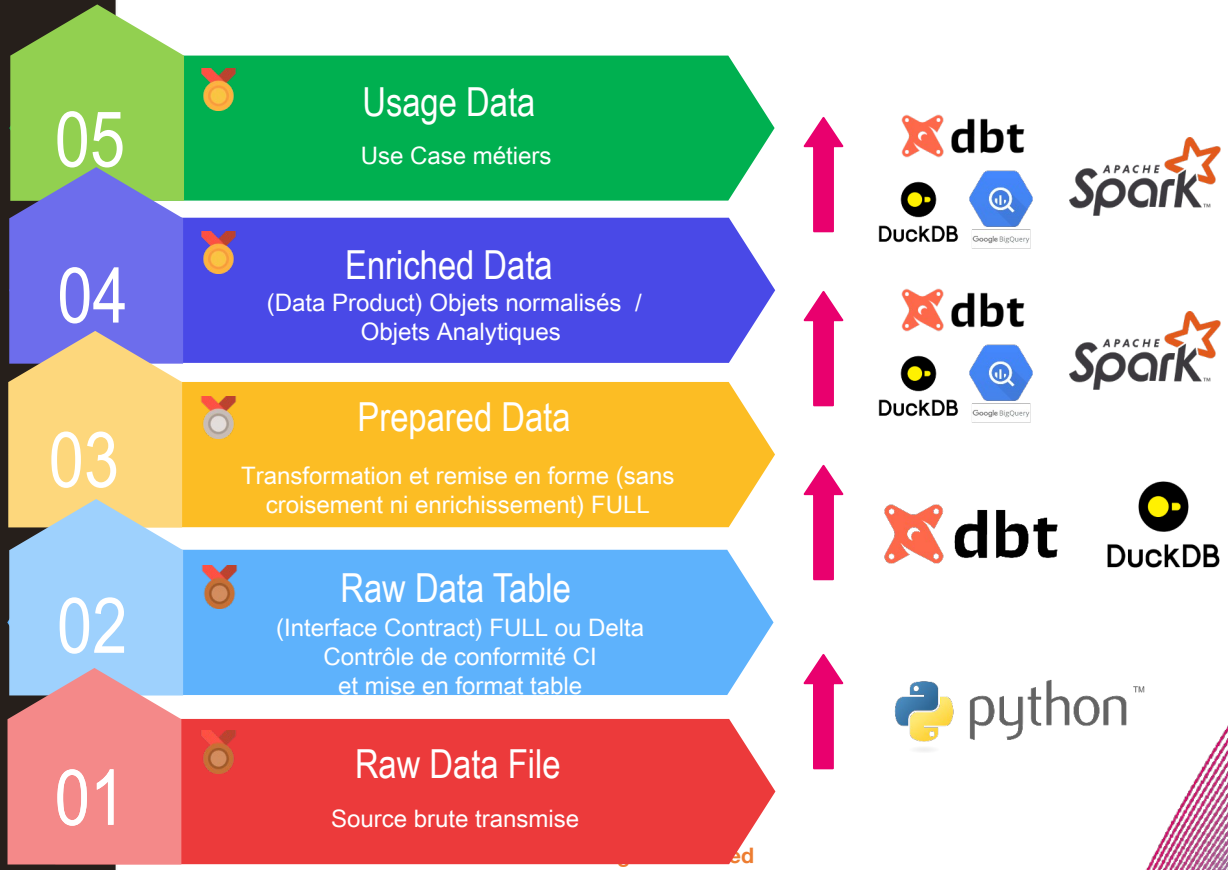
# 02

# Architecture

# Nos Choix

- Sortir de Cloudera/HDFS/kafka/Hive etc ...
- Séparer le compute du stockage pour pouvoir changer la méthode de compute sans impact
- Eviter le vendor lock-in d’où le full open-source
- Scalabilité avec k8s
- Automatiser en partie notre migration

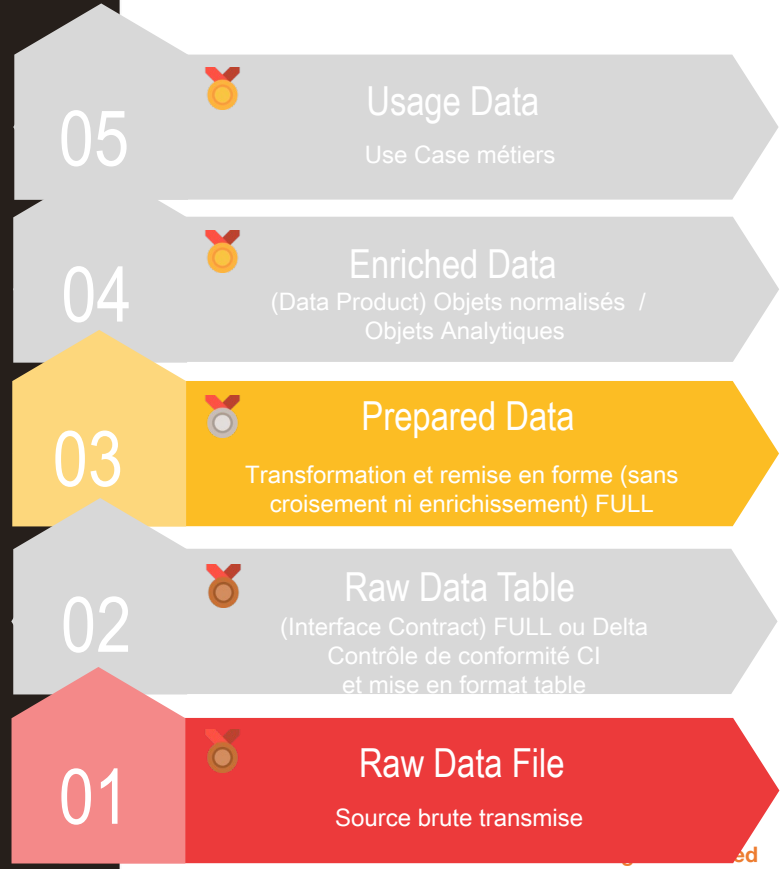
# Chaine valorisation données




ARCHITECTURE DATA MEDAILLON

- Agréger pour le business
- Filtrer, Nettoyer Améliorer
- Intégrer brut

# Focus collecte




ARCHITECTURE DATA MEDAILLON

- Agréger pour le business
- Filtrer, Nettoyer Améliorer
- Intégrer brut

Apache Airflow

kubernetes

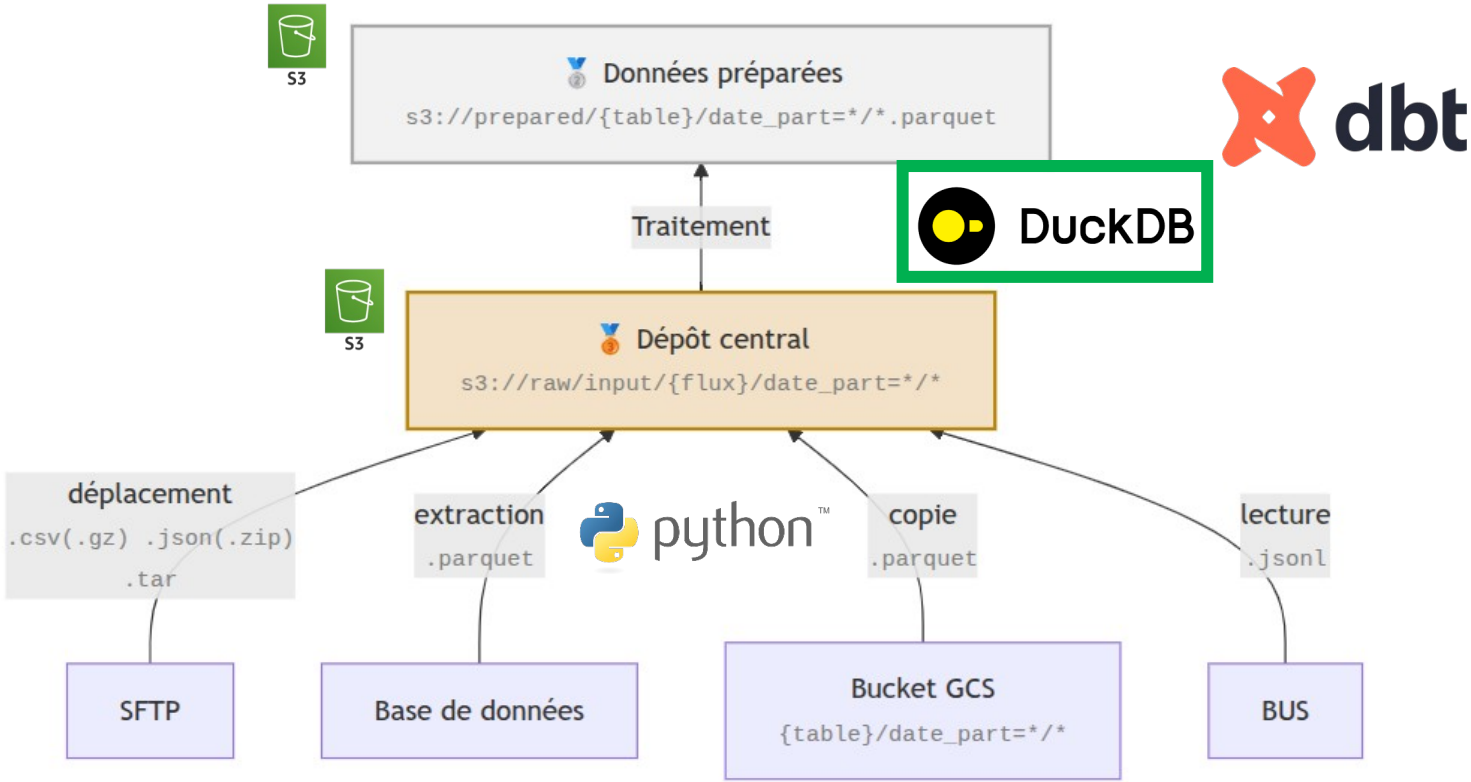
S3

dbt

DuckDB

Data Contract

# Collecte des flux



# 03


# Nouveau moteur de calcul et ETL  SQL

# Présentation DuckDB


## Un moteur de calcul analytique SQL

- Très **performant** (C++)
- **Open source** (Licence MIT)
- Très **léger** (~30 Mo)
- 1 seul nœud (in-process)
- **OLAP** (et non OLTP)
- SQL moderne

| Lecture & écriture | Lecture seule |
|---|---|
| • **JSON, CSV**, Excel, gsheet, **xml**<br>• **Parquet**<br>• Iceberg, Delta Lake, DuckLake<br>• MySQL, **PostgreSQL**, SQLite, **BigQuery**, Teradata, SQL Server<br>• Arrow, Pandas, Polars<br>• GIS (.shp, .geojson) via GDAL | • Http (url)<br>• **Zip** (zipfs) |
| • **S3**, **GCS**, Azure | • Hugging Face |

# Bixi rentals 2014 – 2021

## 35 million

DuckDB

Load 49 .csv ~25s

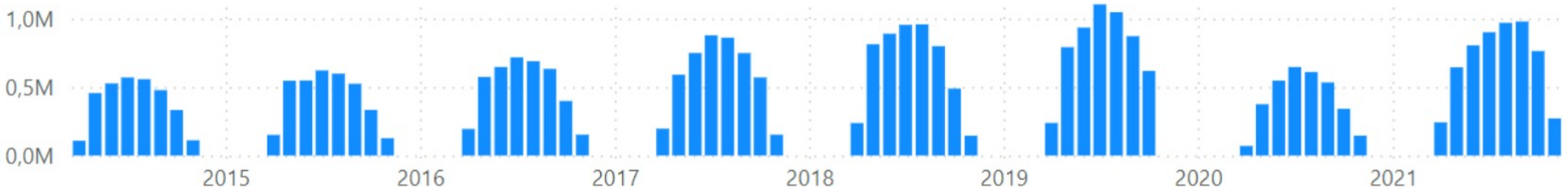
```
create or replace table raw_bixi_rentals as
SELECT * replace( -- 👇 truncate datetime to minutes & cast
  concat(left(start_date, 16),':00')::datetime as start_date,
  concat(left(end_date, 16),':00')::datetime as end_date
)
from read_csv( '**\OD_20*.csv',
    header=true, AUTO_DETECT=TRUE, ignore_errors=true)
```

Offload to Parquet ~ 4s

```
copy raw_bixi_rentals -- 4s
  to 'bixi_rentals_2014_2021.parquet';
copy raw_bixi_rentals -- 10s
  to 'bixi_rentals_2014_2021.csv' ;
copy raw_bixi_rentals -- 34s
  to 'bixi_rentals_2014_2021.json' ;
```

| start_date | start_station_code | end_date | end_station_code | duration_sec | is_member |
|---|---|---|---|---|---|
| 2014-04-15 07:52:00.000 | 6202 | 2014-04-15 07:56:00.000 | 6067 | 262 | 1 |
| 2014-04-15 07:52:00.000 | 6380 | 2014-04-15 08:13:00.000 | 6173 | 1246 | 1 |
| 2014-04-15 07:53:00.000 | 6163 | 2014-04-15 08:08:00.000 | 6009 | 892 | 1 |
| 2014-04-15 07:53:00.000 | 6434 | 2014-04-15 08:17:00.000 | 6053 | 1443 | 1 |
| 2014-04-15 07:53:00.000 | 6184 | 2014-04-15 08:02:00.000 | 6015 | 517 | 1 |

| | | |
|---|---|---|
| raw_bixi_rentals_2014_2021.parquet | 456 796 Ko | x 4.2 |
| raw_bixi_rentals_2014_2021.csv | 1 914 873 Ko | x 2.7 |
| raw_bixi_rentals_2014_2021.json | 5 232 867 Ko | |



# Un grand lecteur & écrivain

DuckDB

l'etl sql

Multitude de sources

- .json(.gz)
- .jsonl(.gz)
- .csv(.gz|.zip)
- format fixe(.tar)
- .parquet

```
from read_csv('s3://chemin/*.csv{.gz}',
              sep='þ', encoding='ISO_8859_15')

from read_json('s3://chemin/*.jsonl{.gz}',
              maximum_depth=1)

from read_parquet('gs://chemin/*.parquet')



from read_gsheet('1B4RFuOnZ4ITZ-nR9givZ7vWV',
              sheet='dim_magasin', range='A2,F')

from read_excel('s3://chemin/fichier.xlsx',
              sheet='dim_magasin', range='A2,F')



from read_html('s3://chemin/*.html',
          record_element='tr', all_varchar=true)

from read_xml('s3://chemin/*.xml',
          root_element='dataroot', record_element='clients')

from 's3://chemin/*.yaml
```

# Travailler avec des json

## un sql moderne

DuckDB

l'etl sql

```
select
        date_part::date as date_part,
        id::uuid as id,

        {
                'version': info->>'version',
                'type': info->>'type',
                'publicationDate': (info->>'publicationDate')::timestamp,
        } as info,
        --> struct(version varchar, type varchar, publicationDate timestamp)

        unnest( data::json[] ) as data,
        --> json
        data->>'id' as data_id,

        filename,

from read_json('s3://chemin/*.jsonl*', maximum_depth=1)
```

# Travailler avec des json

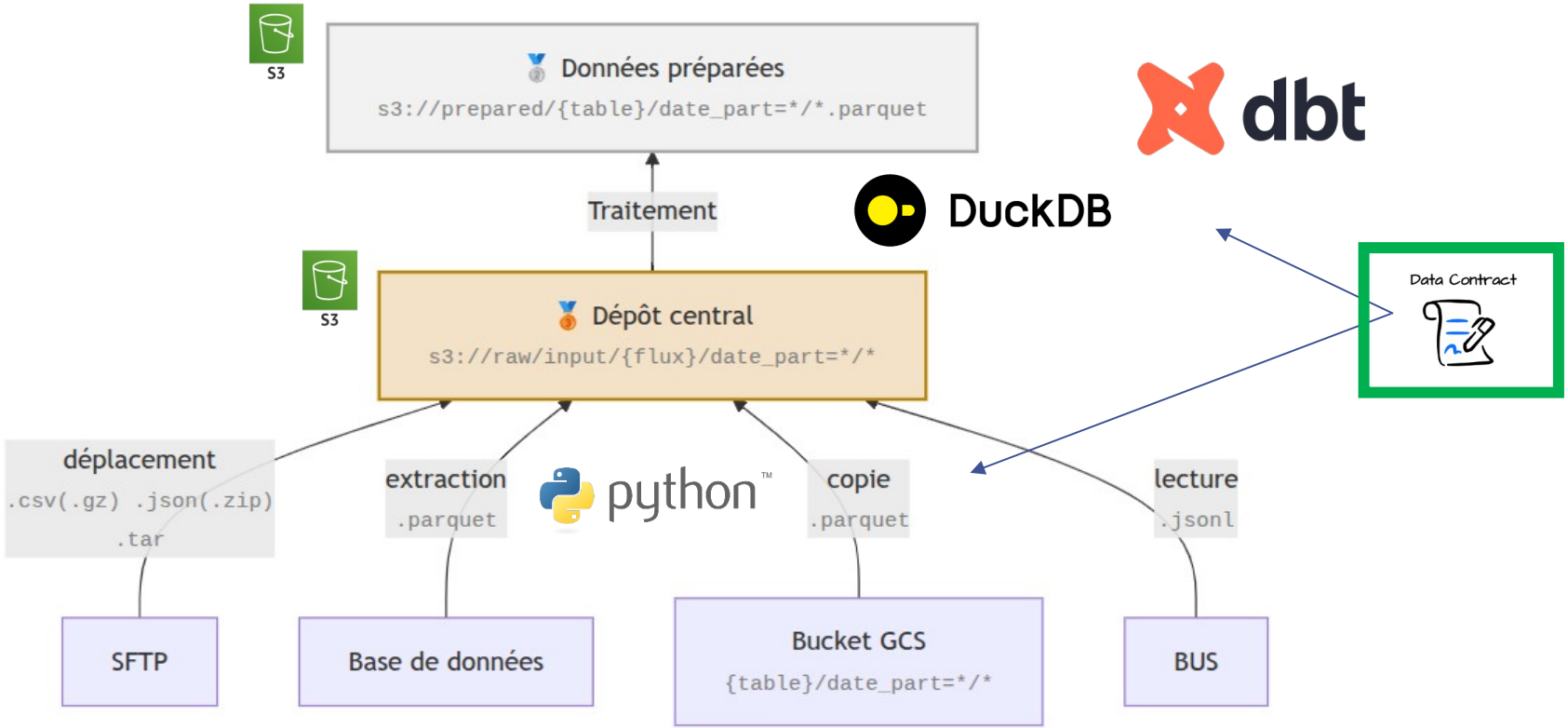
## un sql moderne

DuckDB

**l'etl sql**

```
select
        date_part,   id,   data_id,   info,

        -- installedOffer
        unnest(data->'$.partenaire[*].installedOffer[*]') as installedOffer,

        offer_status: list_distinct(installedOffer->>'$.status[*]'),
        status_count: length(offer_status),
        status_codes: list_distinct(offer_status->>'$.code'),
        status_latest: list_max(data->>'$.status[*].date')::date,

        planif_acteur: list_first(list_filter( offer_status, lambda x: x->>'code' = 'planif' ))->>'acteur',
        planif_date: list_first(list_filter( offer_status, lambda x: x->>'code' = 'planif' ))->>'date',

        install_acteur: list_first(list_filter( offer_status, lambda x: x->>'code' = 'install' ))->>'acteur',
        install_date: xxxx

from {{ ref('int_ord_f_cust_product_order_unnested') }}
```

# Collecte des flux

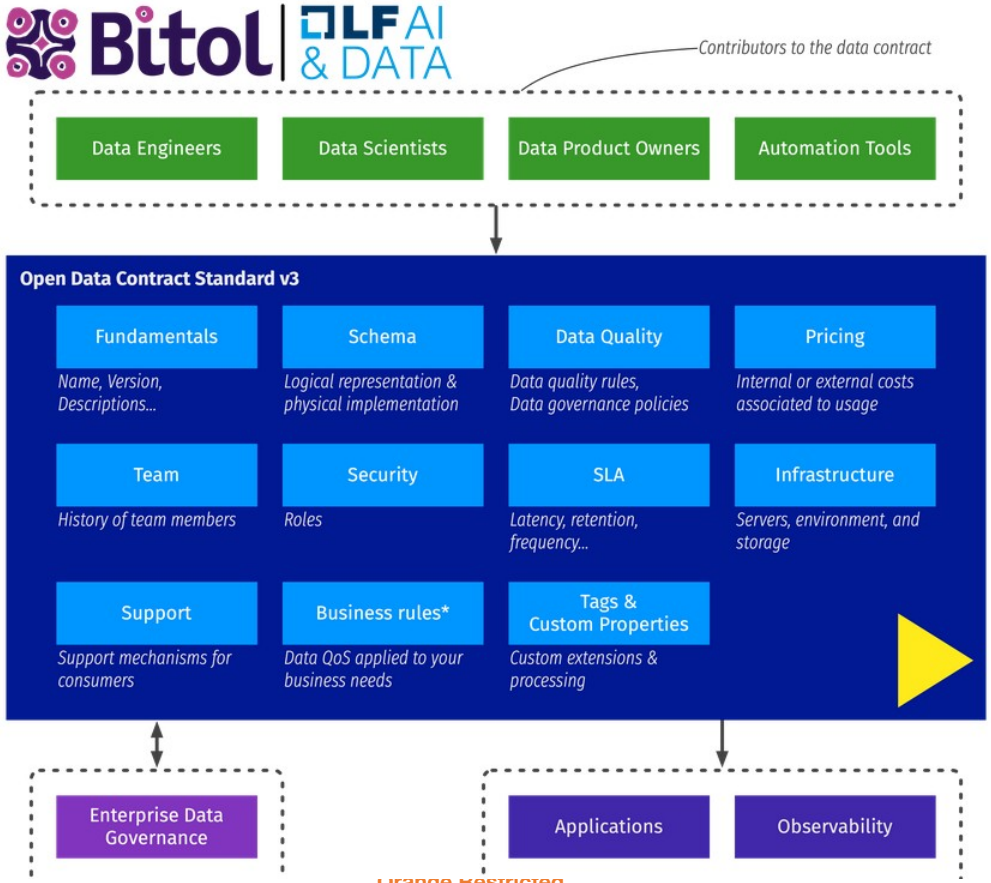


# 03

# Accélérer la migration grâce aux contrats de données **ODCS**

de contrats historiques
à des contrats **ODCS**
aux **artefacts dbt-duckdb**

# **ODCS** Open Data Contract Standard




# Data Contract

Like OpenAPI, but for data

Verified

165 followers  http://datacontract.com

Entropy Data

Alignement au contrat ODCS

Overview  Repositories 14  Discussions  Projects  Packages  People

## Pinned

**datacontract-cli** Public

Enforce Data Contracts

Python 830 207

**datacontract-editor** Public

Data Contract Editor

JavaScript 73 18

**open-data-contract-standard-python** Public

Python library to read and write YAML files using the Open Data Contract Standard

Python 15 4

**open-data-contract-standard-excel-template** Public

Edit Open Data Contract Standard in Excel

35 5

# Classes Pydantic ODCS [url]

```python
class OpenDataContractStandard(pyd.BaseModel):
    version: str | None = None
    kind: str | None = None
    apiVersion: str | None = None
    id: str | None = None
    name: str | None = None
    servers: list[Server] | None = None
    dataProduct: str | None = None
    description: Description | None = None
    schema_: list[SchemaObject] | None = pyd.Field(default=None, alias="schema")
    support: list[Support] | None = None
```

```python
class Server(pyd.BaseModel):
    id: str | None = None
    server: str | None = None
    type: str | None = None
    description: str | None = None
```

```python
class SchemaObject(pyd.BaseModel):
    id: str | None = None
    name: str | None = None
    physicalType: str | None = None
    description: str | None = None
```

Applicable to Elements (either Objects or Properties)

| Key | UX label | Required | Description |
|---|---|---|---|
| id | ID | No | A unique identifier for the element used to create stable, refactor-safe references. Recommended for elements that will be referenced. See References for more details. |
| name | Name | Yes | Name of the element. |
| physicalName | Physical Name | No | Physical name. |

# Classes Pydantic ODCS [url]

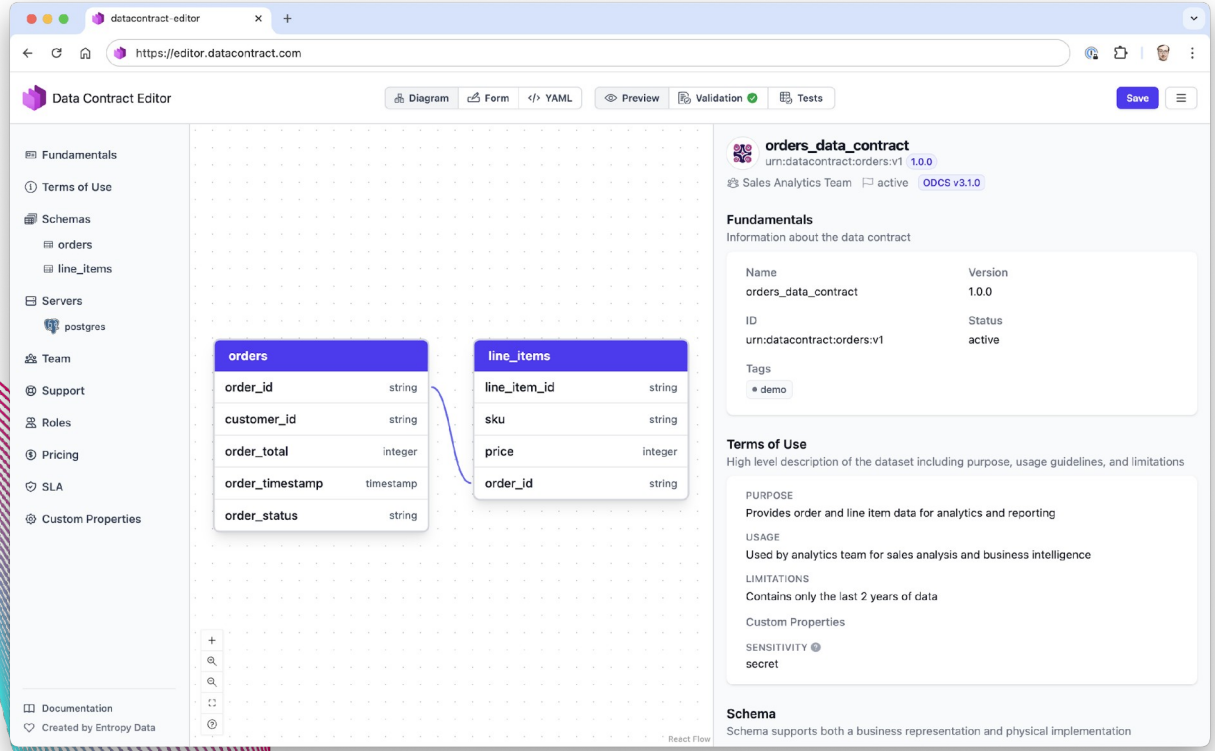
## Usage

```python
from open_data_contract_standard.model import OpenDataContractStandard

# Load a data contract specification from a file
data_contract = OpenDataContractStandard.from_file('path/to/your/data_contract.yaml')
# Print the data contract specification as a YAML string
print(data_contract.to_yaml())
```

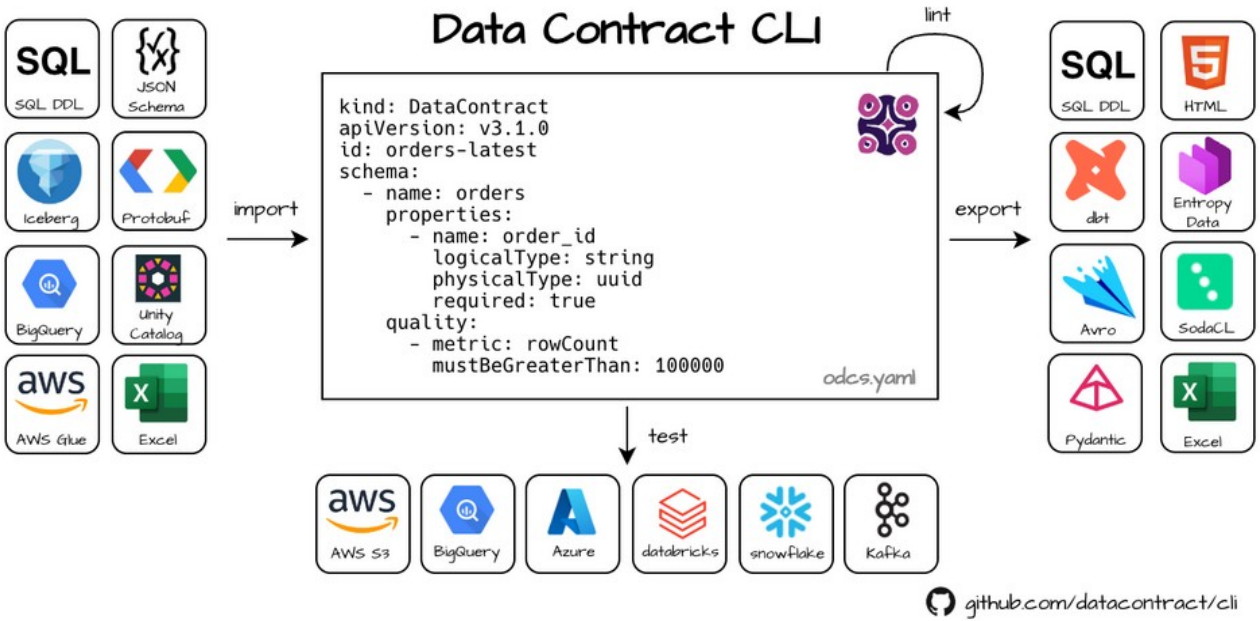
```python
# Traduction d'une colonne FAST vers ODCS
column = SchemaProperty(
    name=nom.lower(),
    logicalType=logicalType,
    # physicalType=logicalType,
    description=(data.get("targetDescription")
                 or data.get("sourceDescription", "")).strip() or None,
    primaryKey=is_pk or None,
    primaryKeyPosition=pk_position if is_pk else None,
    required=data.get("sourceNotNull") == "X" or None,
    transformLogic=data.get("logic"),
    logicalTypeOptions=logicalTypeOptions or None,
    classification="sensitive" if data.get("sensitive") else None,
)
```

# DataContract editor [url]

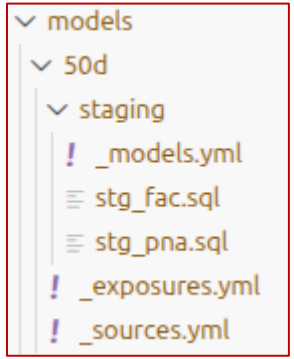
Entropy Data



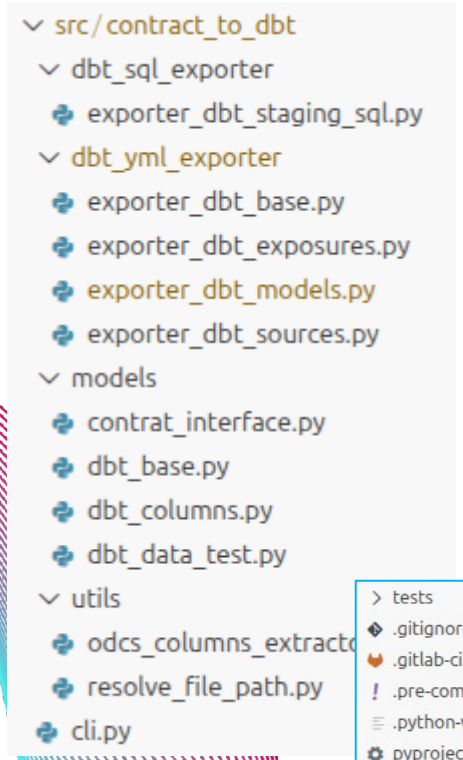
# DataContract CLI

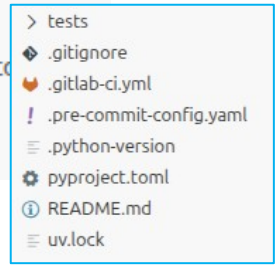



Non suffisant
à notre besoin

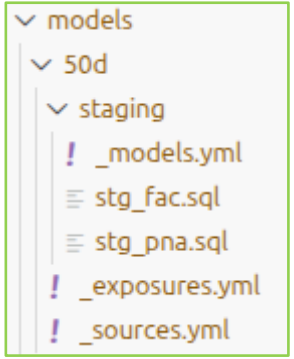


# Contrat ODCS -> artefacts dbt



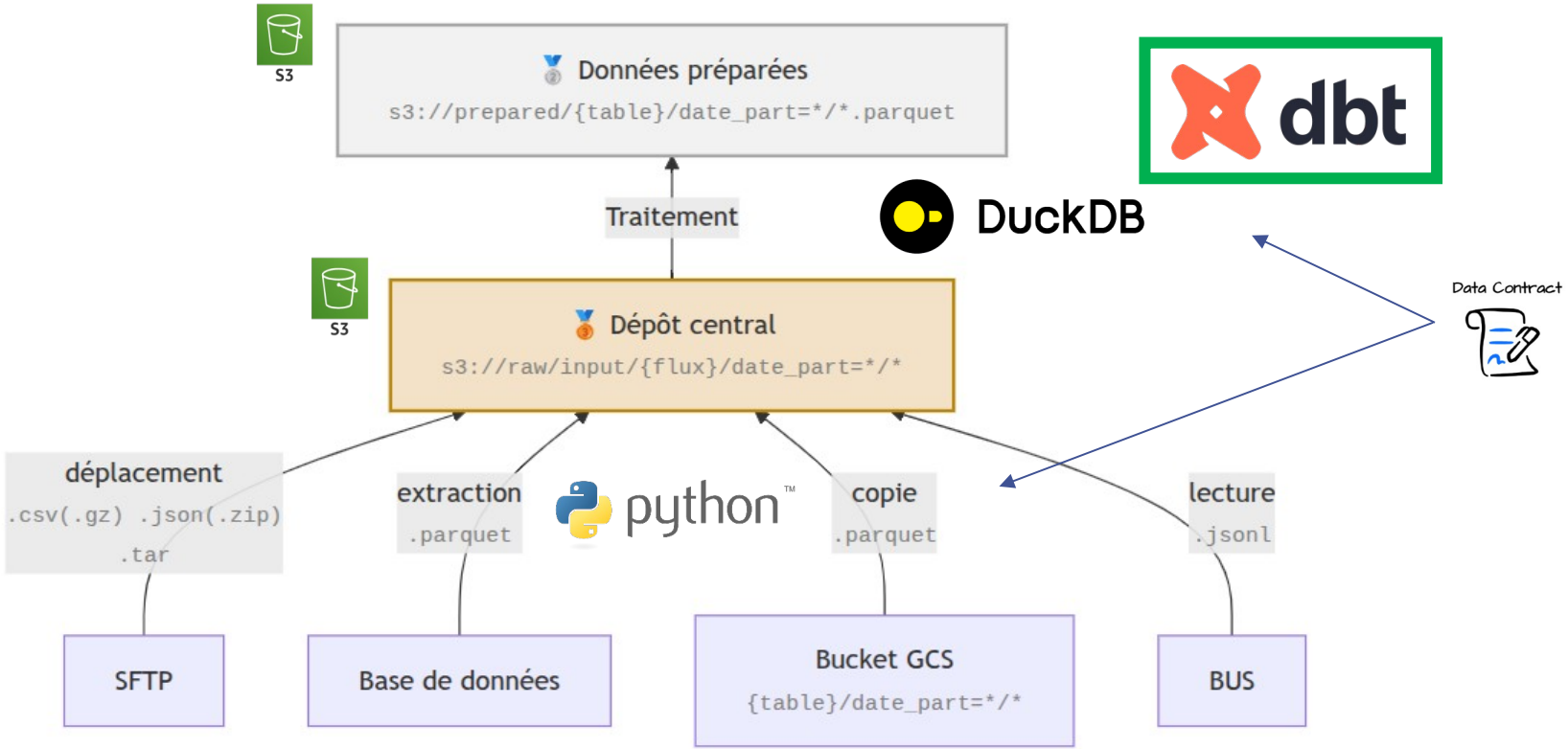




```
root → /workspaces/contract_to_dbt (main)
  $ uv run contract-to-dbt contracts/odcs_50d.yml

Bytecode compiled 2718 files in 249ms
🚀 Export du domaine : 50d
   🔧 Export SQL pour modèle : fac
   🔧 Export SQL pour modèle : pna
🎉 Export terminé avec succès (2 models)
```




```python
@staticmethod
def to_dbt_column(col: SchemaProperty) -> DbtModelColumn:
    '''Convertir une colonne ODCS (SchemaProperty) en dbt'''

    output = {
        "name": col.name or col.physicalName,
        "description": col.description or "🚨 TODO #urbanismeDonnées",
        "data_type": (col.physicalType or col.logicalType).lower(),
        "tags": col.tags,
    }
    # if col.required:  # on fait 1 fois le data_tests: 'not_null' à la source
    #     output["constraints"] = [{"type": "not_null"}]
    return DbtModelColumn(**output)
```

# Collecte des flux



# 03


# Analytics Engineering Industrialisation

## Sommes-nous bien seuls avec dbt-duckdb ?

# Présentation dbt core

requête SQL select
+choix materialisation
*vue, table…*

tests qualité de données
documentation & stewardship

versionage git
workflow dev/prod
CI/CD

# Packages dbt pratiques

dbt docs & dbt **colibri** (catalogue, lineage table/colonne)

dbt **project evaluator** (Gouvernance)

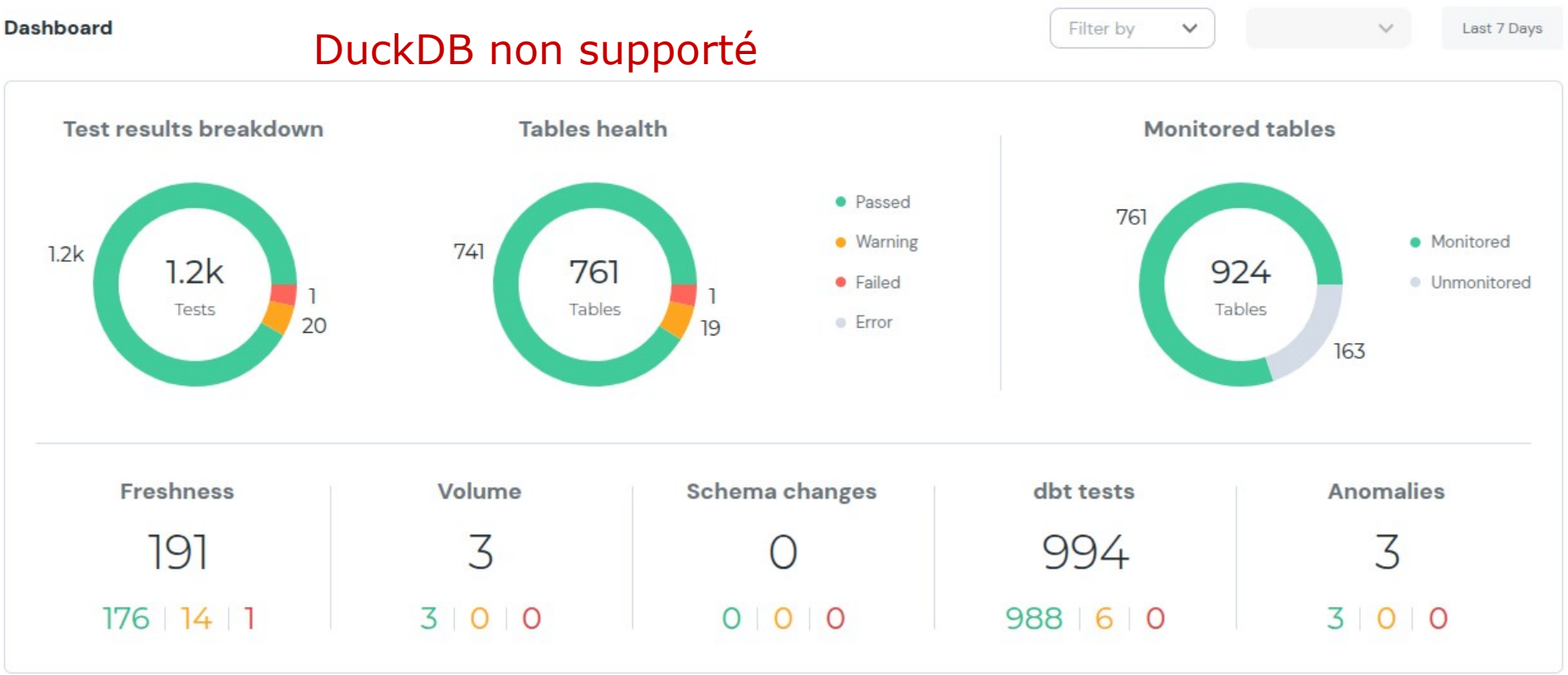
dbt **assertions** (Axel Thevenot) - 1 colonne exceptions

dbt **power user**

dbt **elementary** (Observabilité)
dbt artefacts

dbt utils, recce (MR data diff), dbt loom (mesh)

# **Elementary** – edr report, alert



# Stratégies incrémentales

## Raffinement des données

**Full quotidien**
ce n'est pas de l'incrémental,
mais on veut traiter que le dernier full SCD1

**Append only** - nouvelles données du jour (~ stream)

**Delta** glissant
stratégie : upsert / delete-insert
défis : hard delete, reprise histo, full hebdo

**Change Data Capture**

**Besoins d'historisation ?** SDC2

# **Notre solution** – external "incremental"

dbt > models > mercure > staging > stg_acteurs.sql

Add documentation or tests | Datapilot Notebooks | You, 6 minutes ago | 3 authors (Antoine Giraud and others)

```
{{- config(
    materialized='external',
    location=get_s3_domain_path('mercure') +'/'+ this.name,
    options={"partition_by": "date_part", "OVERWRITE_OR_IGNORE": True},
) -}}

select
    try_cast(date_part as date) as date_part,
    dr_name as dr_name,
    cact as cact,
    lactno as lactno,
    lactpr as lactpr,
    lactali as lactali,
    filename as file_cd,
    try_cast(null as boolean) as is_legal_filter,
    {{ dbt_assertions.assertions() }},
from {{ source('mercure', 'acteurs') }}
where 1=1
    {{ filter_if_partition_exists(config.get('location')) }}
```

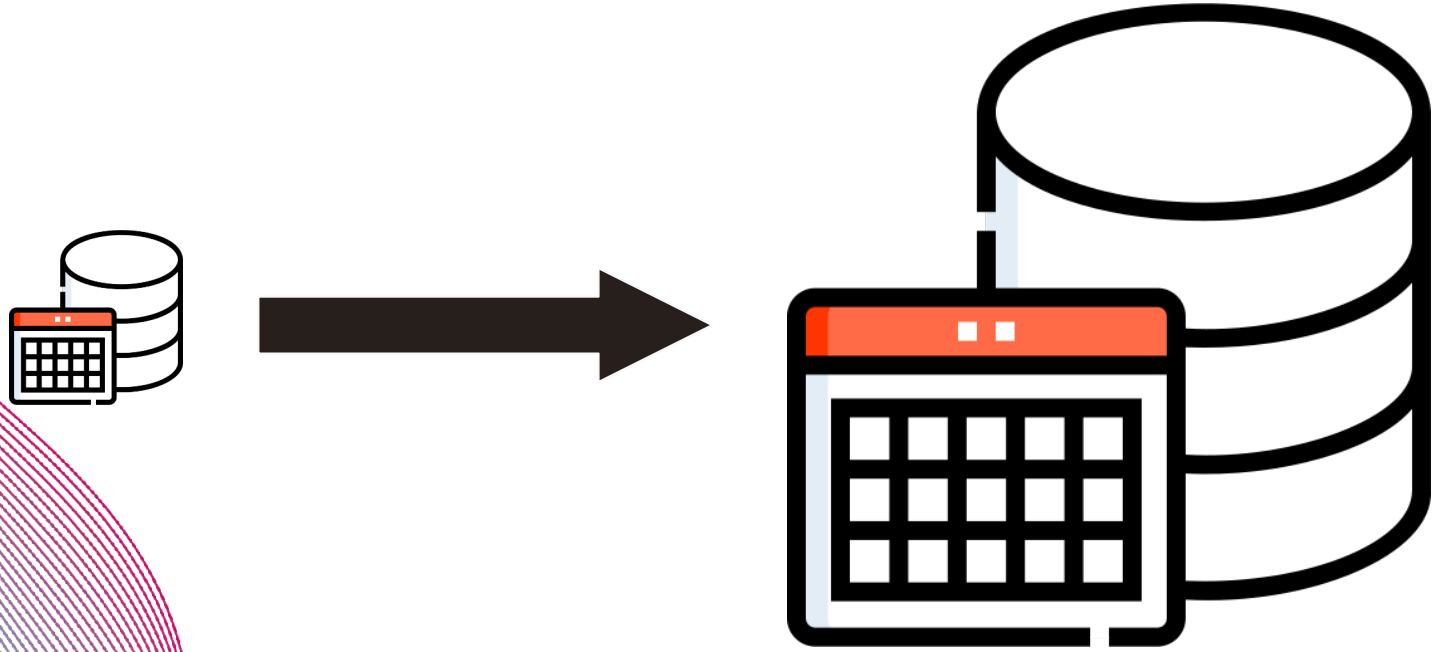
# dbt profiles.yml

dbt > ! profiles.yml

You, 2 days ago | 3 authors (Antoine Giraud and others)

```yaml
dbt_duckdb_k8s:
  target: dev
  outputs:
    dev: &base_profile
      type: duckdb
      path: "{{ env_var('DUCKDB_BDD_PATH', '.') }}/dev.duckdb"
      external_root: "s3://ofr-d2b-prepared-shine-dev"
      threads: 1
      extensions:
        - httpfs
        - postgres
        - ducklake
        - parquet
        - encodings
      settings:
        # memory_limit: "7.8GB"
        # preserve_insertion_order: false
        custom_extension_repository: "/opt/duckdb"
      secrets:
        - type: s3
          key_id: "{{ env_var('S3_ACCESS_KEY_ID', '') }}"
          secret: "{{ env_var('S3_SECRET_ACCESS_KEY', '') }}"
          endpoint: "{{ env_var('S3_ENDPOINT', '') }}"
        # - type: postgres
        #   name: pg_secret
        #   host: host.docker.internal
        #   port: 5432
        #   user: duckuser
        #   password: "duckpass"
        #   database: ducklake_catalog_s3
      # attach:
      #   - path: "ducklake:postgres:dbname=ducklake_catalog_s3"
      #     alias: ducklake
      #     options:
      #       meta_secret: pg_secret
      #       data_path: "s3://ofr-d2b-prepared-shine-int"
```

# Mise à l'échelle ?

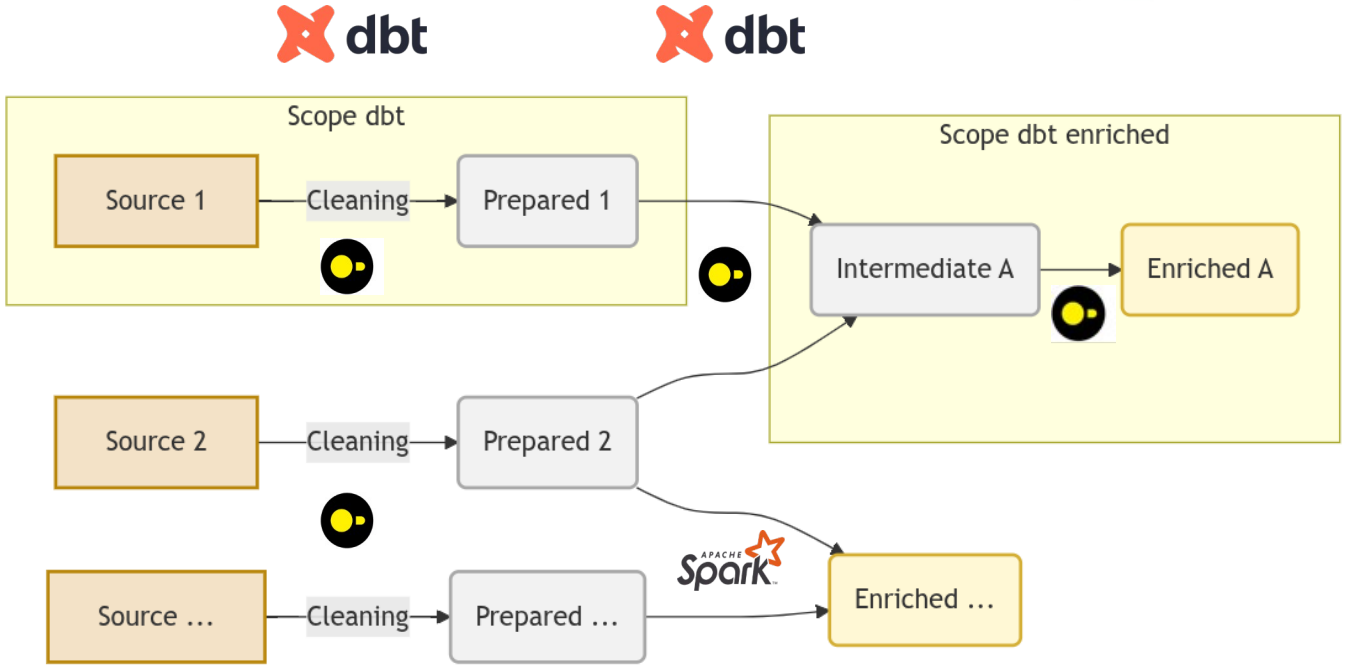


# Vers la mise en prod

kubernetes

Apache Airflow

python

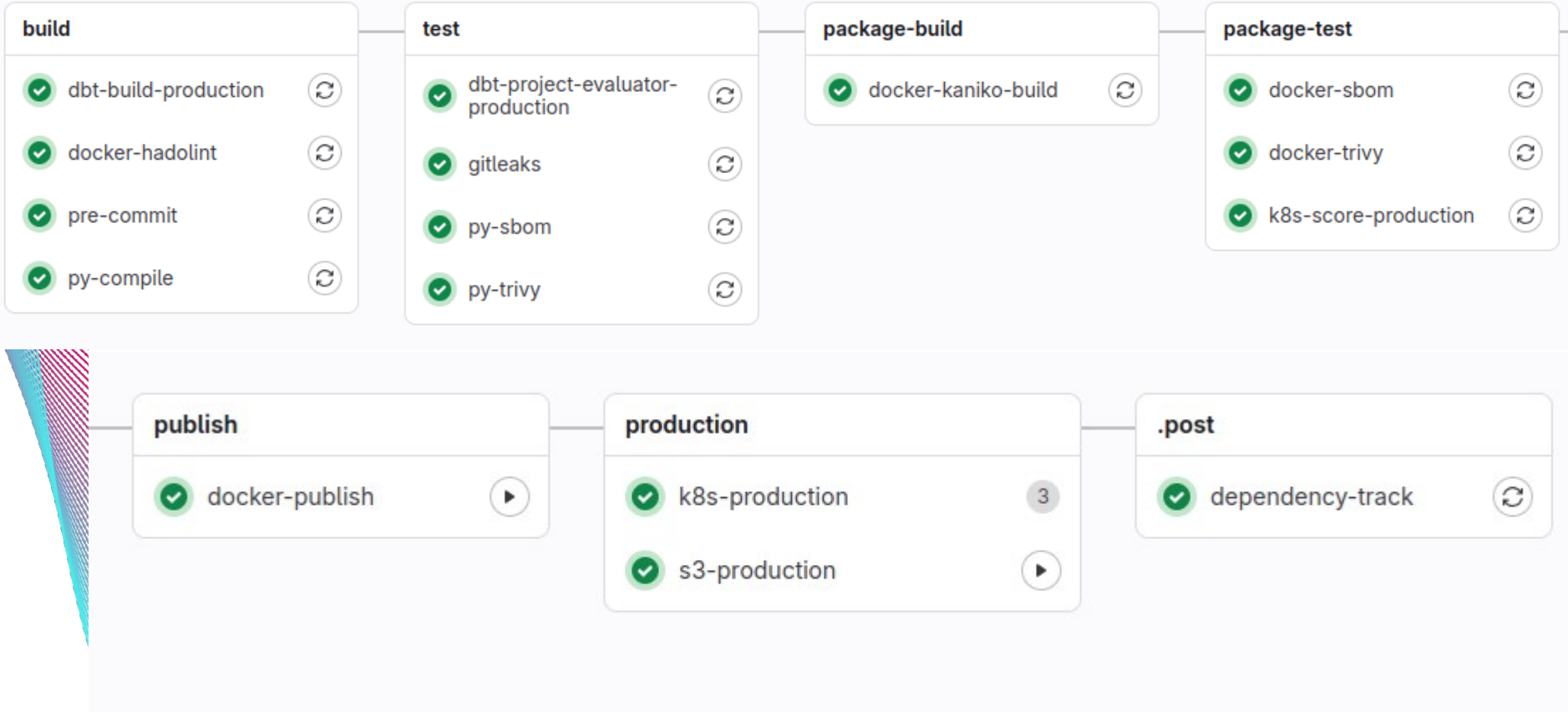


# 03

# Infra & Déploiement kubernetes

## DataOps, CICD GitLab, To Be Continuous

# Déploiement automatisé



# **Mais … DuckDB** étant in-process **il faut gérer**

La **gouvernance** et modélisation

La gestion des **habilitations**

**L’exposition** des données

**L’intégration** au SI

# k9s – cli pratique

```
Context: production-ch1 [RW]      <ctrl-d> Delete          <t> Trigger
Cluster:                          <d>      Describe        <w> Warp To Namespace
User:    antoine                  <e>      Edit            <y> YAML
K9s Rev: v0.50.16 ⚡v0.50.18       <?>      Help
K8s Rev: v1.25.16                 <shift-j> Jump Owner
CPU:     n/a                      <s>      Suspend/Resume
MEM:     n/a
```

cronjobs(d2bprd-batch)[45]

| NAME↑ | SCHEDULE | SUSPEND | LAST SCHEDULE | AGE |
|---|---|---|---|---|
| dbt-duckdb-shine-shinedata | 15 2 * * * | false | 13h | 20d |
| dbt-duckdb-s | | | | |
| stargate-54y | | | | |

```
Context: review [RW]              <ctrl-d> Delete          <t> Trigger
Cluster:                          <d>      Describe        <w> Warp To Namespace
User:    antoine                  <e>      Edit            <y> YAML
K9s Rev: v0.50.16 ⚡v0.50.18       <?>      Help
K8s Rev: v1.32.8                  <shift-j> Jump Owner
CPU:     n/a                      <s>      Suspend/Resume
MEM:     n/a
```

cronjobs(d2bdev-batch)[20]

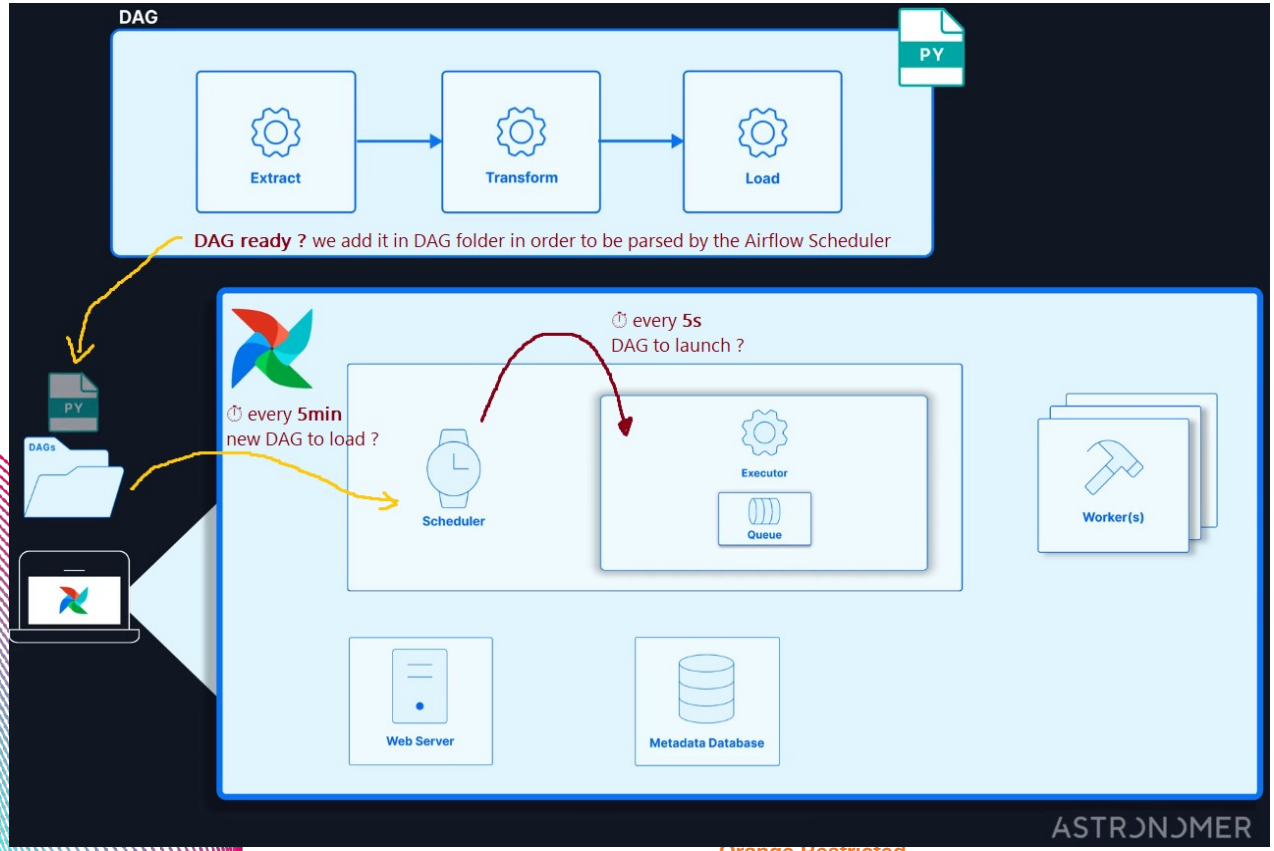
| NAME↑ | SCHEDULE | SUSPEND | LAST SCHEDULE | AGE |
|---|---|---|---|---|
| dbt-duckdb-shine-shinedata | 15 2 * * * | true | <none> | 2d1h |
| dbt-duckdb-shine-shinelfo | 15 2 * * * | true | <none> | 2d1h |
| dbt-duckdb-siu-comptes | 15 2 * * * | true | <none> | 2d |

Apache Airflow

# 03

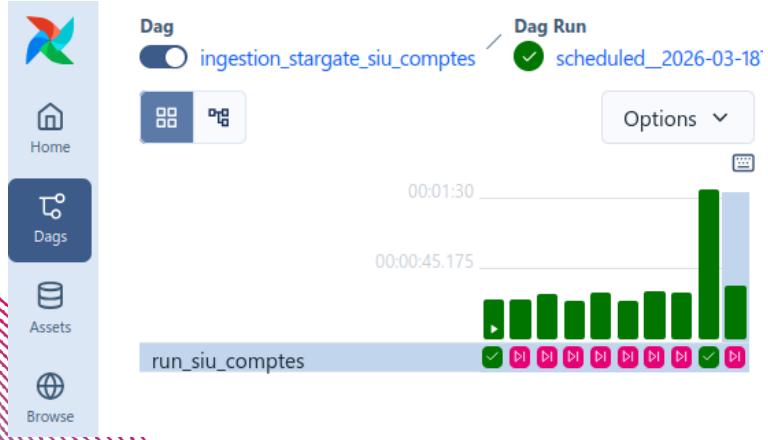

# Orchestration

# Déploiement Airflow sur k8s



# Skip ou pas skip ?

On tourne qd on n'a pas de données ?




kubernetes

# k8sCronJobTriggerOperator

Création d’un custom operator pour déclencher un cronjob k8s

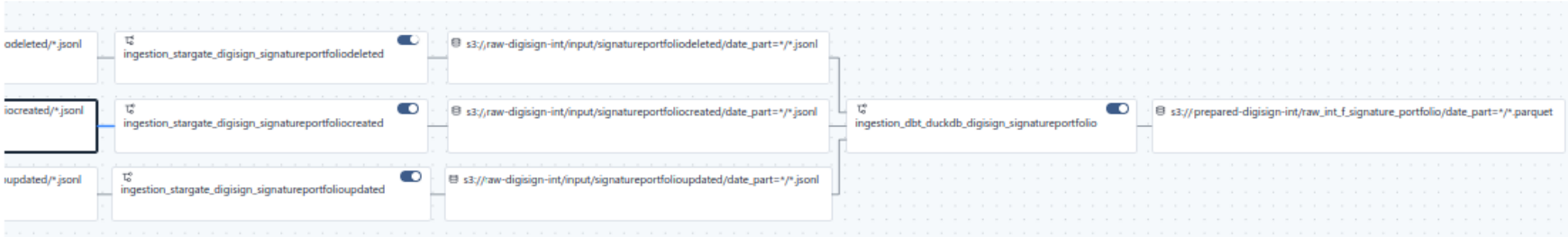
```
17  INFO - 📅 Trigger type: Time/Manual (Date: 2026-03-18 00:10:00+00:00)
18  INFO - 🚀 Triggering CronJob: stargate-siu-comptes in namespace=d2bint-batch
19  INFO -    🤖 Hostname: airflow-edge-worker-batch-[redacted]q
20  INFO -       ENV: INT
21  INFO -    job.batch/stargate-siu-comptes-260318-0010 created
22  INFO -    ⏳ Waiting for pod creation…
23  INFO -    📦 Found pod: stargate-siu-comptes-260318-0010-q9nqj
24  INFO -    ⏳ Waiting for pod stargate-siu-comptes-260318-0010-q9nqj to be ready for logs…
25  INFO -    ✔️ Container is running — logs available
26  INFO - 📜 Streaming logs for pod stargate-siu-comptes-260318-0010-q9nqj…
27  INFO    Démarrage du traitement : siu (env integration)
28  INFO    🚀 Début du traitement pour siu (flux: comptes)
29  INFO    Chargement du contrat depuis S3 : interfacecontract-siu.yml
30  INFO    Contrat d'interface chargé avec succès en mémoire.
31  INFO    Recherches des fichiers dans la bucket [redacted]-raw-siu-int et le répertoire S3 : /reception/comptes/
32  INFO    Liste des fichiers trouvés : []
33  INFO    Aucun fichier avec replace_mode à traiter.
34  INFO    Fraicheur min : None, fraicheur max : None, nombre de fraicheurs différentes : 0
35  WARNING - 🔍 Detection: No files to process. This task will skip.
36  INFO - ⏳ Waiting for Job completion…
37  INFO - ✅ Job stargate-siu-comptes-260318-0010 succeeded
```

# Orchestration via Assets

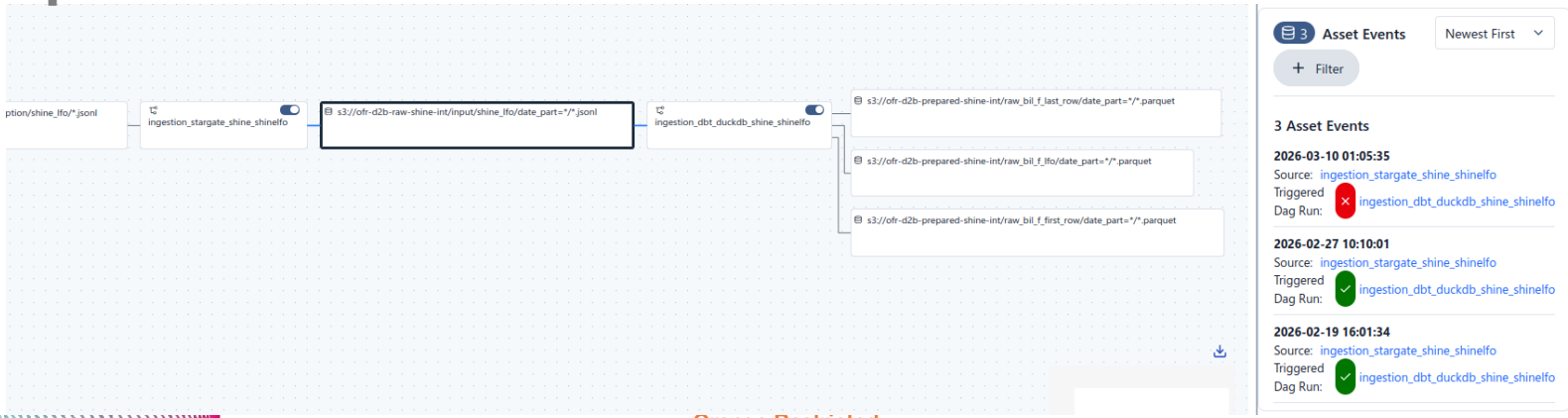
## Concentrateur ?

**Asset**

s3://ofr-d2b-raw-digisign-int/reception/signatureportfoliocreated/*.jsonl




## Split ?



# Assets

On tourne qd on n'a pas de donnée ?

```python
assets_in = [
    Asset(f"s3://raw-{domain}-{ENV}/input/{table}/date_part=*/{input_file}")
    for table in input_flux
]

cron_expr = row.get("cron", "15 2 * * *")
timezone = row.get("timezone", "Europe/Paris")

schedule = (
    assets_in
    if len(input_flux) == 1
    else AssetOrTimeSchedule(
        timetable=CronTriggerTimetable(cron_expr, timezone=timezone),
        assets=assets_in,
    )
)

with DAG(
    dag_id=f"ingestion_dbt_duckdb_{domain}_{flux_short}",
    schedule=schedule,
    tags=["ingestion", "dbt-duckdb", domain],
    default_args=default_executor_args,
    dagrun_timeout=timedelta(hours=0.33),
):
```


# Bonus

# IHM - Métiers

## Vos référentiels métiers

### Référentiel Djingo

Tables du contrat

| name | description | colonnes |
|---|---|---|
| motif_abandon | Table de correspondance dernière question posée et motif abandon | question_poser |
| provided_event_exclus | Table des provided Event exclus des réponses données | provided_event |
| ref_source | Table de correspondance caller_id et canal source | caller_id sou |
| univers_intent | Table de correspondance intention et univers | intention un |

Gérer ce contrat

## Charger un référentiel de djingo

motif_abandon | provided_event_exclus | ref_source | univers_intent

### Table motif_abandon

En attente d'un fichier...

Table de correspondance dernière question posée et motif abandon

Choisir un fichier pour motif_abandon

Drag and drop file here
Limit 30MB per file
Browse files

### Colonnes attendues

| name | type | description |
|---|---|---|
| question_poser | string | La question posée par Djingo |
| motif_abandon | string | None |

# Qualité par construction grain à la table


## Non fiable

Qualité de donnée imprévisible.

**Aucun contrôle** automatisé, découverte des erreurs via les utilisateurs.

Données non fiables et non suivies.


## Informel

Quelques tests disparates, pas de couverture systématique.

Des tests sont en place mais **non documentés de façon adéquate dans les spécifications**.


## Formalisé

Base indispensable.

- **Unicité** : Clé primaire définie et testée.

- **Non-nullité** : Colonnes critiques obligatoires.

- **Typage** : Formats strictement respectés (ex : date, timezone).

- Stratégie **historisation** (SCD1, SCD2)


## Évalué

Contrôles de cohérence métier.

- **Tests de Plage (Range)** : Valeurs dans des bornes logiques.
- **Tests de Nomenclature** : Valeurs conformes à des listes de référence.
- **Relations** : Vérification des clés étrangères.


## Optimisé

Surveillance du comportement statistique de la table.

- **Anomalie de Volume** : Alerte si la table reçoit 50% de lignes en moins qu'habituellement.

- **Écart Type** : Détection de valeurs aberrantes.


# Conclusion