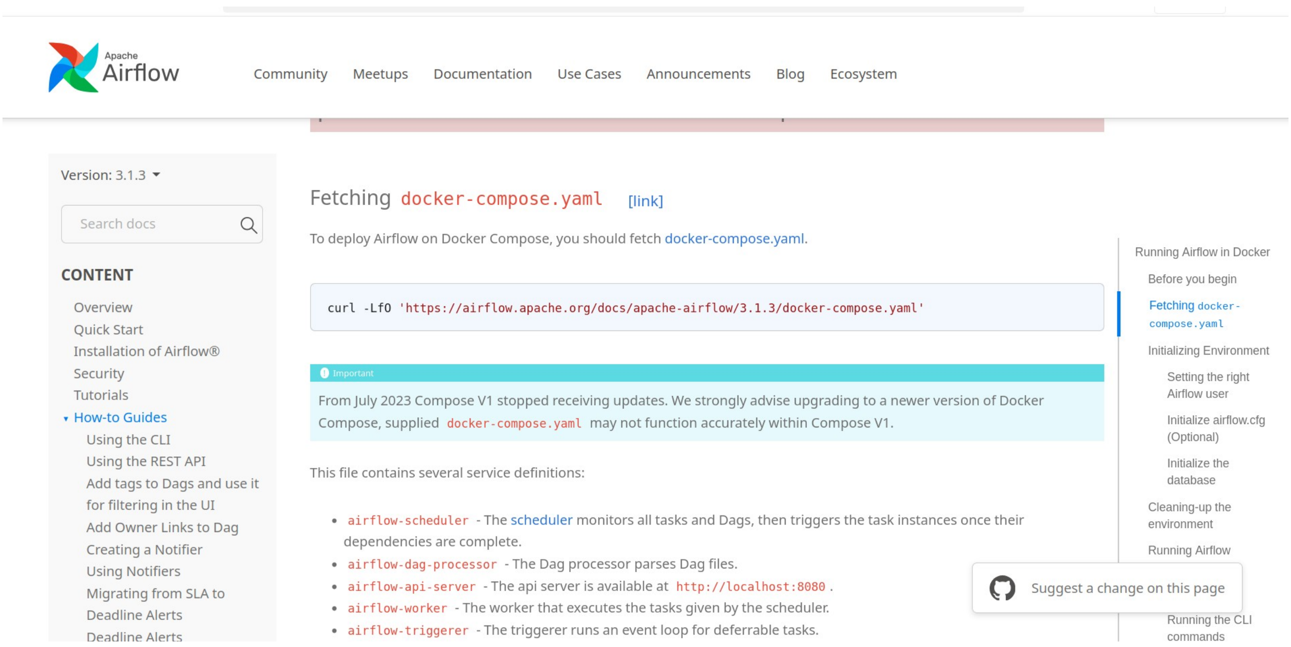
Grandes lignes du projet

Prérequis: - Avoir un compte AWS
- petites notions sur linux, nous travaillerons sur Ubuntu 24.04
- connaissance de base docker, docker-compose

lien, si vous n’avez pas de compte AWS et vous voulez en créé : https://smart.dhgate.com/step-by-step-guide-to-creating-and-securing-your-aws-account-for-optimal-use/

un pipeline Airflow orchestrant des extractions de données depuis un bucket S3 de Amazon (weather(météo), COVID) et ensuite des transformations utilisant duckdb via dbt.

- Installation de airflow via le script docker compose que vous pouvez trouver sur le site officiel de airflow: https://airflow.apache.org/docs/apache-airflow/stable/howto/docker-compose/index.html#fetching-docker-compose-yaml,



Fetching `docker-compose.yaml` [link]

To deploy Airflow on Docker Compose, you should fetch docker-compose.yaml.

```
curl -LfO 'https://airflow.apache.org/docs/apache-airflow/3.1.3/docker-compose.yaml'
```

Important

From July 2023 Compose V1 stopped receiving updates. We strongly advise upgrading to a newer version of Docker Compose, supplied `docker-compose.yaml` may not function accurately within Compose V1.

This file contains several service definitions:

- `airflow-scheduler` - The scheduler monitors all tasks and Dags, then triggers the task instances once their dependencies are complete.
- `airflow-dag-processor` - The Dag processor parses Dag files.
- `airflow-api-server` - The api server is available at `http://localhost:8080`.
- `airflow-worker` - The worker that executes the tasks given by the scheduler.
- `airflow-triggerer` - The triggerer runs an event loop for deferrable tasks.

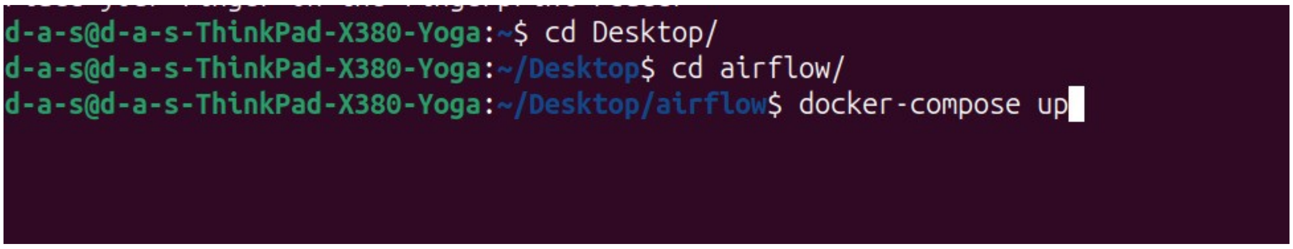
- cliquez sur le lien en bleu dans l’image ou directement ici : docker-compose.yml pour telecharger le script docker compose.

- Créé un dossier airflow et mettre le script docker-compose.yml téléchargé, dans ce dossier

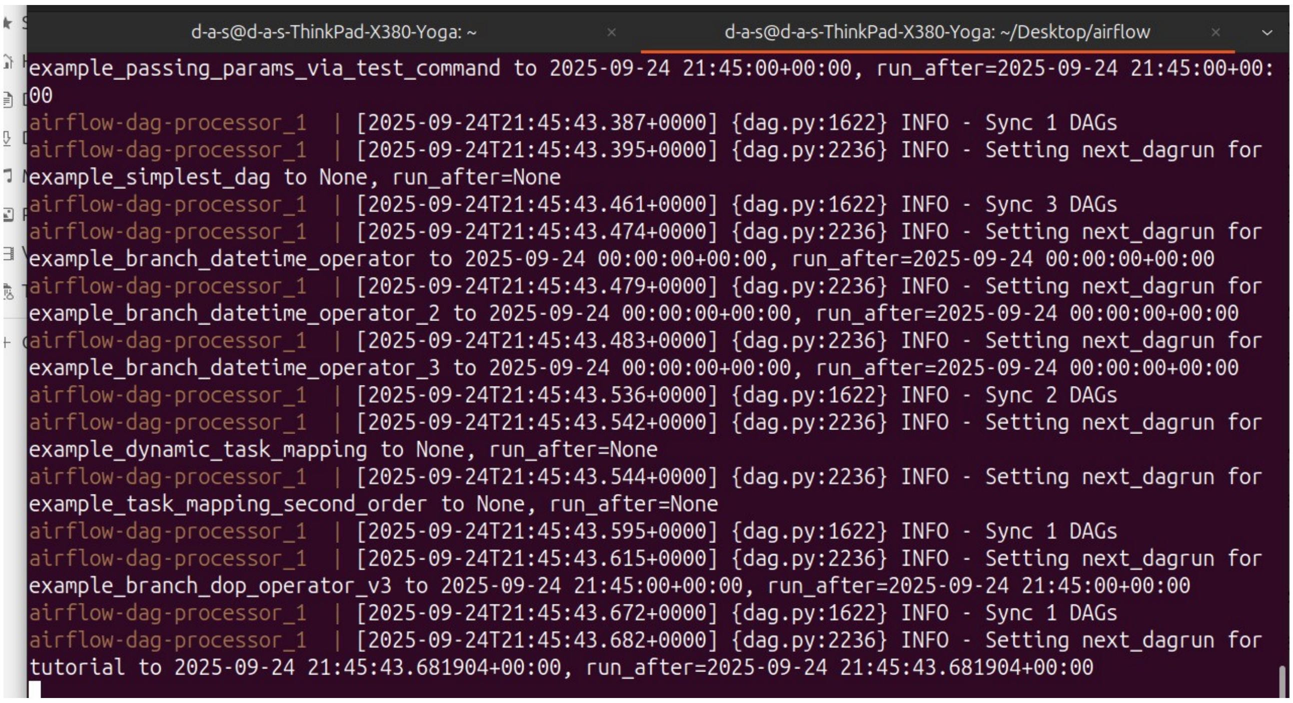
- créé cette structure de base à l’interieur du dossier airflow avec cette commande : mkdir -p dags plugins scripts config data/{raw,processed} tests

ensuite :

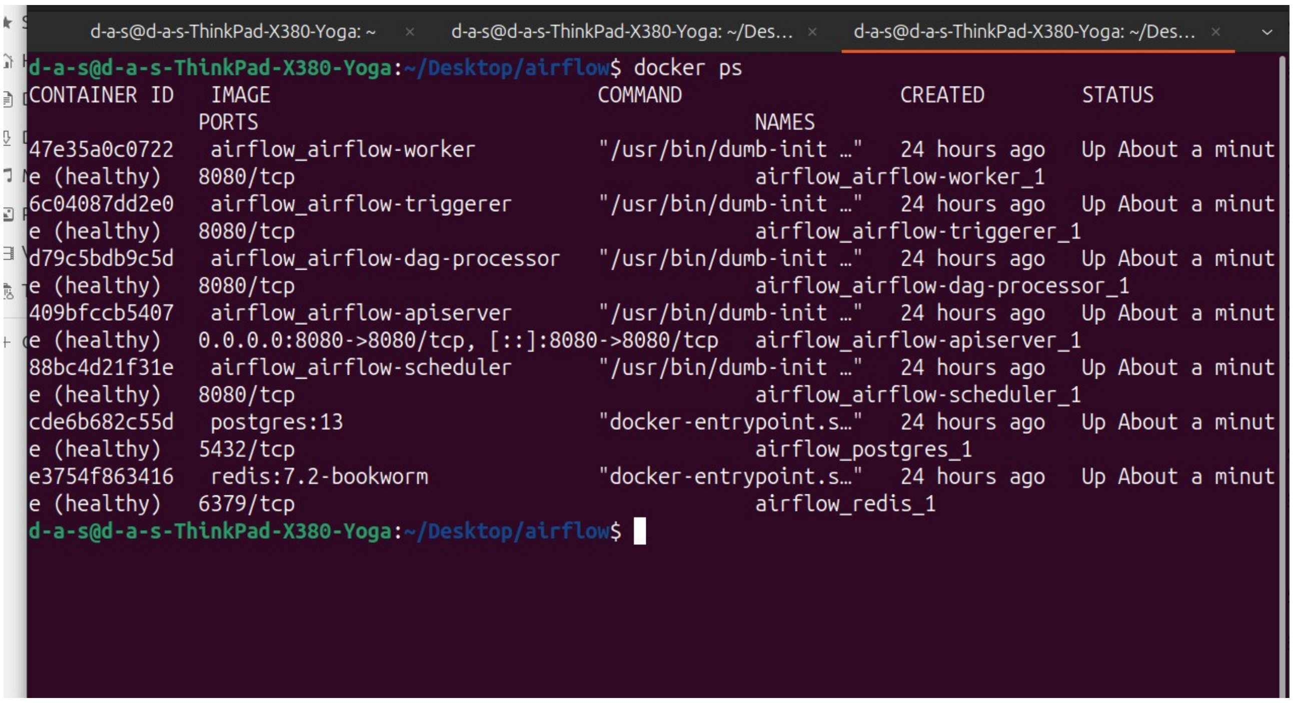
- on se deplace dans le dossier où se trouve le script et on exécute la commande : docker-compose up -d
il installera tout les conteneurs liés à docker et va les lancer.



```
d-a-s@d-a-s-ThinkPad-X380-Yoga:~$ cd Desktop/
d-a-s@d-a-s-ThinkPad-X380-Yoga:~/Desktop$ cd airflow/
d-a-s@d-a-s-ThinkPad-X380-Yoga:~/Desktop/airflow$ docker-compose up
```


d-a-s@d-a-s-ThinkPad-X380-Yoga: ~
d-a-s@d-a-s-ThinkPad-X380-Yoga: ~/Desktop/airflow

```
example_passing_params_via_test_command to 2025-09-24 21:45:00+00:00, run_after=2025-09-24 21:45:00+00:
00
airflow-dag-processor_1  | [2025-09-24T21:45:43.387+0000] {dag.py:1622} INFO - Sync 1 DAGs
airflow-dag-processor_1  | [2025-09-24T21:45:43.395+0000] {dag.py:2236} INFO - Setting next_dagrun for
example_simplest_dag to None, run_after=None
airflow-dag-processor_1  | [2025-09-24T21:45:43.461+0000] {dag.py:1622} INFO - Sync 3 DAGs
airflow-dag-processor_1  | [2025-09-24T21:45:43.474+0000] {dag.py:2236} INFO - Setting next_dagrun for
example_branch_datetime_operator to 2025-09-24 00:00:00+00:00, run_after=2025-09-24 00:00:00+00:00
airflow-dag-processor_1  | [2025-09-24T21:45:43.479+0000] {dag.py:2236} INFO - Setting next_dagrun for
example_branch_datetime_operator_2 to 2025-09-24 00:00:00+00:00, run_after=2025-09-24 00:00:00+00:00
airflow-dag-processor_1  | [2025-09-24T21:45:43.483+0000] {dag.py:2236} INFO - Setting next_dagrun for
example_branch_datetime_operator_3 to 2025-09-24 00:00:00+00:00, run_after=2025-09-24 00:00:00+00:00
airflow-dag-processor_1  | [2025-09-24T21:45:43.536+0000] {dag.py:1622} INFO - Sync 2 DAGs
airflow-dag-processor_1  | [2025-09-24T21:45:43.542+0000] {dag.py:2236} INFO - Setting next_dagrun for
example_dynamic_task_mapping to None, run_after=None
airflow-dag-processor_1  | [2025-09-24T21:45:43.544+0000] {dag.py:2236} INFO - Setting next_dagrun for
example_task_mapping_second_order to None, run_after=None
airflow-dag-processor_1  | [2025-09-24T21:45:43.595+0000] {dag.py:1622} INFO - Sync 1 DAGs
airflow-dag-processor_1  | [2025-09-24T21:45:43.615+0000] {dag.py:2236} INFO - Setting next_dagrun for
example_branch_dop_operator_v3 to 2025-09-24 21:45:00+00:00, run_after=2025-09-24 21:45:00+00:00
airflow-dag-processor_1  | [2025-09-24T21:45:43.672+0000] {dag.py:1622} INFO - Sync 1 DAGs
airflow-dag-processor_1  | [2025-09-24T21:45:43.682+0000] {dag.py:2236} INFO - Setting next_dagrun for
tutorial to 2025-09-24 21:45:43.681904+00:00, run_after=2025-09-24 21:45:43.681904+00:00
```


d-a-s@d-a-s-ThinkPad-X380-Yoga: ~
d-a-s@d-a-s-ThinkPad-X380-Yoga: ~/Des...
d-a-s@d-a-s-ThinkPad-X380-Yoga: ~/Des...

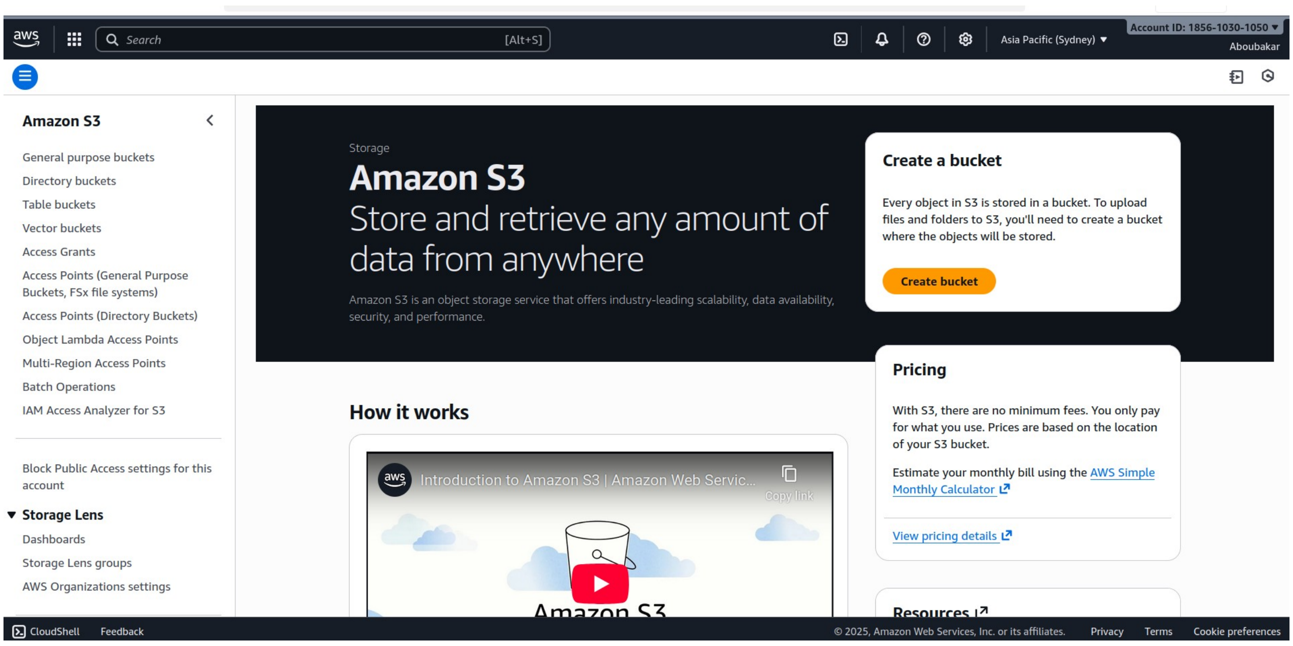
```
d-a-s@d-a-s-ThinkPad-X380-Yoga:~/Desktop/airflow$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED        STATUS
              PORTS                                     NAMES
47e35a0c0722   airflow_airflow-worker         "/usr/bin/dumb-init …"   24 hours ago   Up About a minut
e (healthy)   8080/tcp                                  airflow_airflow-worker_1
6c04087dd2e0   airflow_airflow-triggerer      "/usr/bin/dumb-init …"   24 hours ago   Up About a minut
e (healthy)   8080/tcp                                  airflow_airflow-triggerer_1
d79c5bdb9c5d   airflow_airflow-dag-processor  "/usr/bin/dumb-init …"   24 hours ago   Up About a minut
e (healthy)   8080/tcp                                  airflow_airflow-dag-processor_1
409bfccb5407   airflow_airflow-apiserver      "/usr/bin/dumb-init …"   24 hours ago   Up About a minut
e (healthy)   0.0.0.0:8080->8080/tcp, [::]:8080->8080/tcp   airflow_airflow-apiserver_1
88bc4d21f31e   airflow_airflow-scheduler      "/usr/bin/dumb-init …"   24 hours ago   Up About a minut
e (healthy)   8080/tcp                                  airflow_airflow-scheduler_1
cde6b682c55d   postgres:13                    "docker-entrypoint.s…"   24 hours ago   Up About a minut
e (healthy)   5432/tcp                                  airflow_postgres_1
e3754f863416   redis:7.2-bookworm             "docker-entrypoint.s…"   24 hours ago   Up About a minut
e (healthy)   6379/tcp                                  airflow_redis_1
d-a-s@d-a-s-ThinkPad-X380-Yoga:~/Desktop/airflow$
```

- Installer dbt-core + l’adaptateur DuckDB car on ne va pas utiliser Postgres mais DuckDB vu que

S3 est un outil de stockage objet qui permet de faire des laques de données qui n’ ont pas une structure relationnelle pour pouvoir effectuer des requetes sur S3, on ne peut pas utiliser Postgres car gérant les db relationnelles, on va utiliser DuckDB qui peut requeté sur des fichiers Json, CSV et à des connecteurs pour S3 qu’on va utiliser.
On le fera plus tard vers la fin dans un conteneur airflow pour qu’il soit visible à l’interieur d’ airflow.

- Créé un bucket S3, on l’a nommé dans notre cas data1pipeline vous pouvez le changé

- Allez sur votre console AWS, connectez vous, tapez dans la barre de recherche S3, vous allez être ici


Search [Alt+S]
Asia Pacific (Sydney)
Account ID: 1856-1030-1050
Aboubakar
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Vector buckets
Access Grants
Access Points (General Purpose Buckets, FSx file systems)
Access Points (Directory Buckets)
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings
CloudShell
Feedback

Storage

Amazon S3

Store and retrieve any amount of data from anywhere

Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

Create a bucket

Every object in S3 is stored in a bucket. To upload files and folders to S3, you'll need to create a bucket where the objects will be stored.

Create bucket

Pricing

With S3, there are no minimum fees. You only pay for what you use. Prices are based on the location of your S3 bucket.

Estimate your monthly bill using the AWS Simple Monthly Calculator

View pricing details

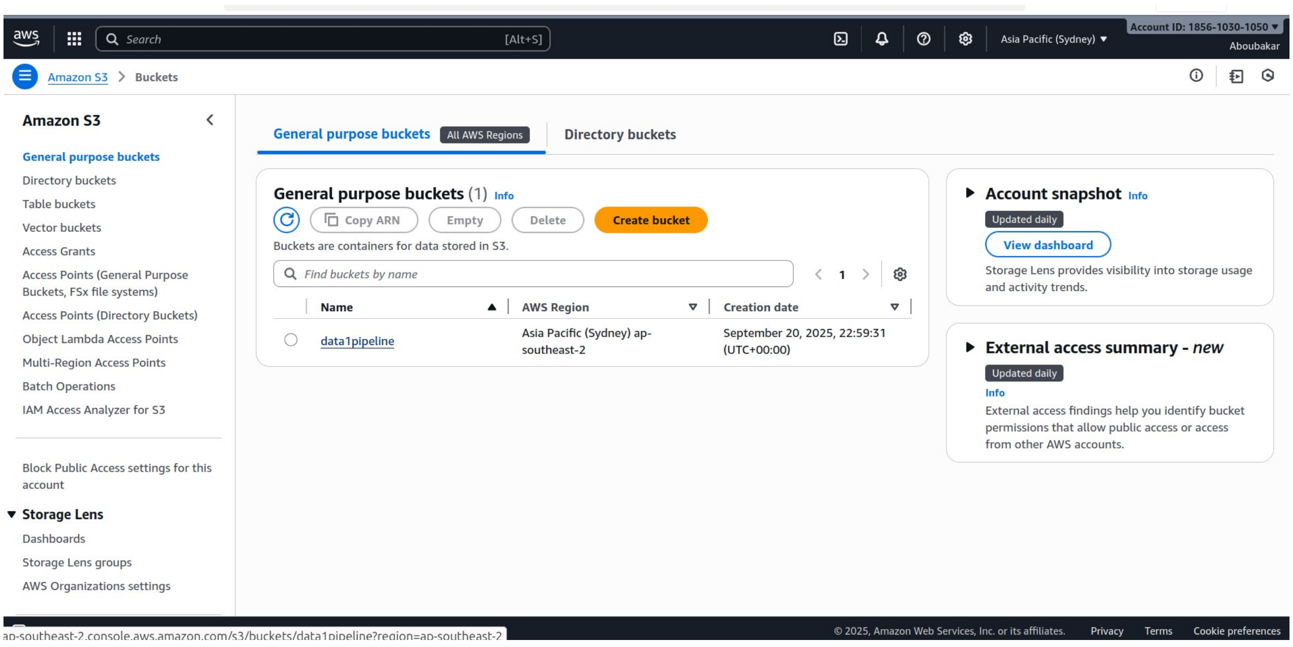
How it works

Introduction to Amazon S3 | Amazon Web Servic...

Resources

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

-Cliquez sur general purpose bucket


Search [Alt+S]
Asia Pacific (Sydney)
Account ID: 1856-1030-1050
Aboubakar
Amazon S3 > Buckets
Amazon S3
General purpose buckets
Directory buckets
Table buckets
Vector buckets
Access Grants
Access Points (General Purpose Buckets, FSx file systems)
Access Points (Directory Buckets)
Object Lambda Access Points
Multi-Region Access Points
Batch Operations
IAM Access Analyzer for S3
Block Public Access settings for this account
Storage Lens
Dashboards
Storage Lens groups
AWS Organizations settings

General purpose buckets | All AWS Regions | Directory buckets

General purpose buckets (1) Info

Copy ARN | Empty | Delete | Create bucket

Buckets are containers for data stored in S3.

Find buckets by name

Name	AWS Region	Creation date
data1pipeline	Asia Pacific (Sydney) ap-southeast-2	September 20, 2025, 22:59:31 (UTC+00:00)

Account snapshot Info

Updated daily

View dashboard

Storage Lens provides visibility into storage usage and activity trends.

External access summary - *new*

Updated daily

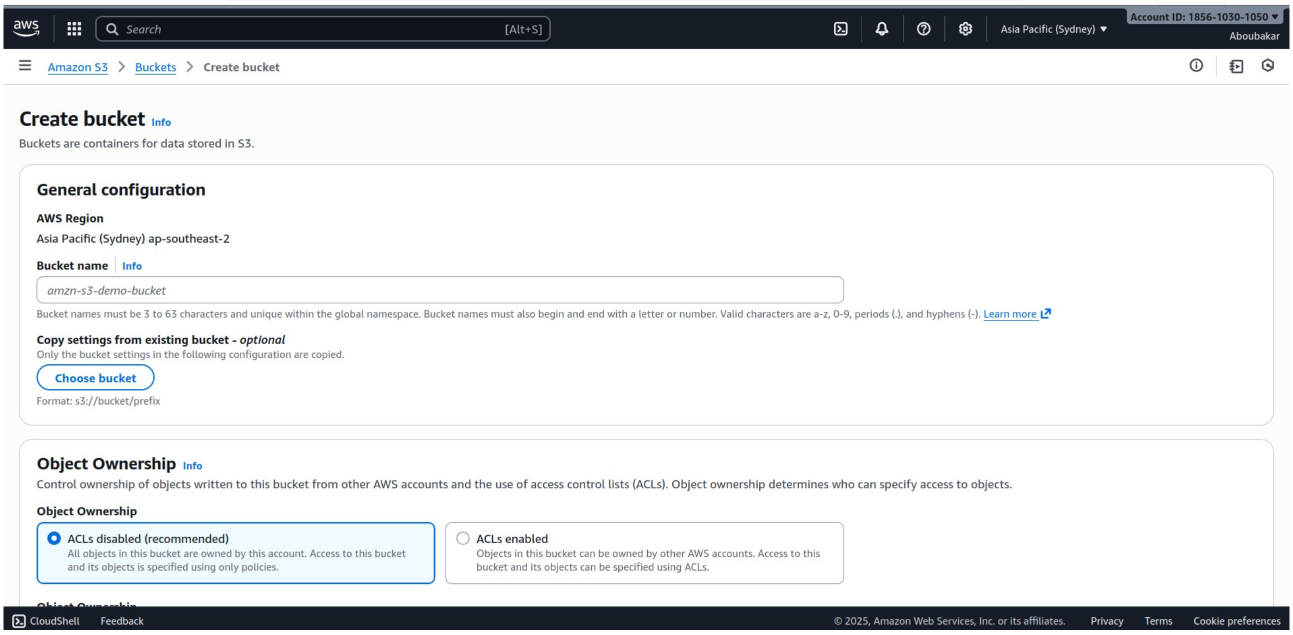
Info

External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.

ap-southeast-2.console.aws.amazon.com/s3/buckets/data1pipeline?region=ap-southeast-2

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

- Cliquez sur create bucket et entrez le nom laissez les parametre ar defaut et cliquez tout en bas sur create bucket


Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region

Asia Pacific (Sydney) ap-southeast-2

Bucket name Info

amzn-s3-demo-bucket

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). Learn more

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

Object Ownership Info

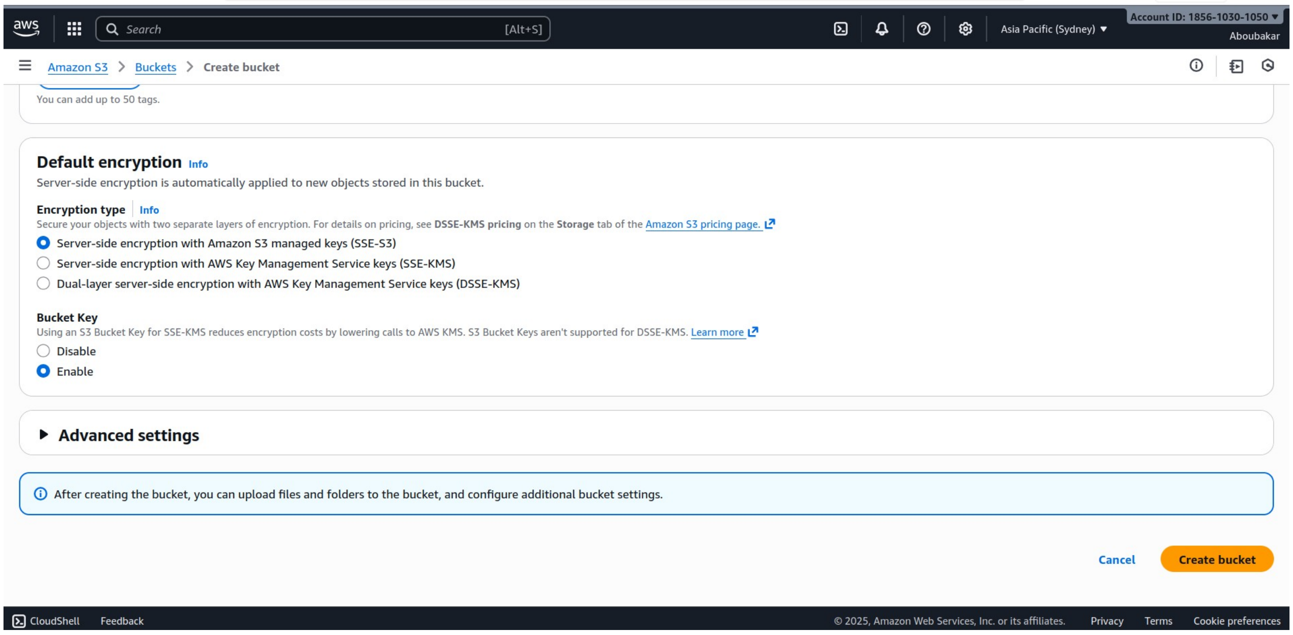
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

Object Ownership

[✓ ACLs disabled (recommended)] All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

ACLs enabled: Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


Amazon S3 > Buckets > Create bucket

You can add up to 50 tags.

Default encryption Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type Info

Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Storage** tab of the Amazon S3 pricing page.

[✓ Server-side encryption with Amazon S3 managed keys (SSE-S3)]
Server-side encryption with AWS Key Management Service keys (SSE-KMS)
Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more

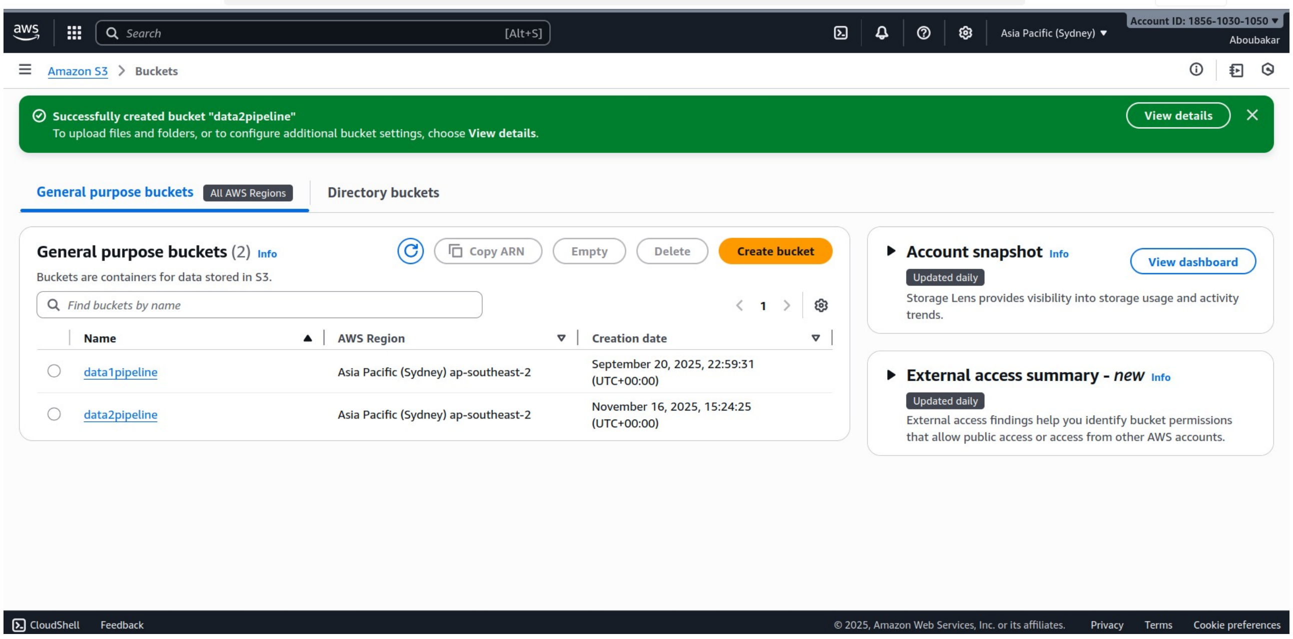
Disable [✓ Enable]

Advanced settings

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel Create bucket

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



Successfully created bucket "data2pipeline"
To upload files and folders, or to configure additional bucket settings, choose **View details**.

General purpose buckets (2) Info
Buckets are containers for data stored in S3.

Name	AWS Region	Creation date
data1pipeline	Asia Pacific (Sydney) ap-southeast-2	September 20, 2025, 22:59:31 (UTC+00:00)
data2pipeline	Asia Pacific (Sydney) ap-southeast-2	November 16, 2025, 15:24:25 (UTC+00:00)

Account snapshot Info
Updated daily
Storage Lens provides visibility into storage usage and activity trends.

External access summary - *new* Info
Updated daily
External access findings help you identify bucket permissions that allow public access or access from other AWS accounts.

- Créé S3 Access Key et Secret Key
1- Ces identifiants permettent à dbt (ou tout autre outil) d’accéder à votre bucket S3.
2- Ils sont générés depuis la console AWS IAM (Identity and Access Management) :
3- Créer un utilisateur IAM avec accès programmatique (Programmatic access).
4- Attacher une politique (Policy) permettant l’accès à S3 (ex: AmazonS3FullAccess pour test ou une policy restreinte pour la production).
6- Cliquez sur votre utilisateur et cherchez : create access key et generé les
7- La console génère alors une Access Key ID et une Secret Access Key.
Ces clés peuvent être utilisées dans dbt, Airflow, ou tout script pour se connecter à S3.

- Placer les fichiers data_extraction_dag.py et dbt_pipeline_dag.py dans le dossier airflow créé par docker compose

- Créé dans le dossier airflow, le dossier dbt et à l interieur de ce dossier, le dossier “votre projet”

```
# Aller dans ton dossier airflow (si tu es sur ton Desktop par exemple)
cd ~/Desktop/airflow

# Créer le dossier dbt à l'intérieur du dossier airflow
mkdir -p dbt

# Aller dans ce dossier dbt
cd dbt

# Créer le dossier pour ton projet dbt (exemple : my_dbt_project)
mkdir -p my_dbt_project

# Initialiser un projet dbt à l'intérieur
cd my_dbt_project
dbt init my_dbt_project
```

Volumes montés dans le conteneur Airflow via Docker Compose :
- /home/d-a-s/Desktop/airflow/dbt:/opt/airflow/dbt: on le met dans la partie volume de notre fichier docker compose

→ Le dossier local ~/airflow/dbt est monté dans le conteneur à /opt/airflow/dbt.
→ Sert à stocker les projets dbt (models, macros, analyses, seeds, etc.) de manière persistante.
→ Toute modification dans le conteneur sera visible sur la machine hôte et vice-versa.

- /home/d-a-s/.dbt:/opt/airflow/.dbt : on le met dans la partie volume de notre fichier docker compose

→ Le dossier local ~/.dbt, il contient le fichier profile, vous pouvez le mettre ailleurs mais il faut specifier le dossier et monté dans le conteneur à /opt/airflow/.dbt.
→ Contient les fichiers de configuration dbt, notamment profiles.yml qui définit les connexions aux bases de données.
→ Permet au conteneur d'utiliser les mêmes configurations que celles définies sur la machine hôte.

Configuration dbt pour le projet my_dbt_project (profiles.yml) :

- target: dev

→ Définit l'environnement de travail par défaut pour dbt.

- outputs:

dev:
type: duckdb
path: /opt/airflow/dbt/my_dbt_project/my_duckdb.duckdb
→ Fichier DuckDB stocké dans le dossier du projet dbt monté dans le conteneur Airflow.
→ Persistant : toutes les modifications restent sur la machine hôte via le volume Docker.
extensions:

- httpfs

→ Permet à DuckDB d'accéder à des fichiers stockés sur S3 directement depuis dbt.
threads: 1
→ Nombre de threads utilisés pour l'exécution des modèles dbt.
schema: main
→ Schéma par défaut utilisé dans DuckDB.
s3_region: ap-southeast-2
s3_access_key_id: AKIASWNZ4NZ5JMMRMLOV
s3_secret_access_key: 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
→ Informations de connexion pour accéder au bucket S3 depuis dbt/DuckDB.

Fichier sources.yml (dans models/staging)

Rôle :

• Définit les sources de données externes que dbt va utiliser dans les modèles.

• Permet à dbt de référencer les fichiers S3 (ou d'autres sources) de manière centralisée et documentée.

• Facilite la traçabilité et la gestion des données avant toute transformation (staging).

Contenu et explication :

version: 2

sources:

- name: raw

```
# Nom logique de la source de données
tables:
- name: weather_data # Table ou fichier météo
description: "Fichiers météo depuis S3"
external:
# Données externes (non transformées)
location: "s3://data1pipeline/raw/weather/" # Chemin S3
format: json
# Format des fichiers
pattern: "*.json" # Pattern pour filtrer les fichiers
- name: covid_data # Table ou fichier COVID
description: "Fichiers COVID depuis S3"
external:
location: "s3://data1pipeline/raw/covid/"
format: csv
pattern: "*.csv"
```

Résumé :

• Ce fichier n'est pas strictement obligatoire pour faire fonctionner dbt, mais il est fortement recommandé.

• Il sert à documenter les sources et permettre aux modèles dbt de les référencer facilement via source('raw', 'weather_data') ou source('raw', 'covid_data').

• Cela centralise la configuration S3, donc si le chemin change, il suffit de modifier ici.

- Étapes pour ajouter tes modèles

1. Aller dans ton projet dbt :
```
cd ~/Desktop/airflow/dbt/my_dbt_project
```

2. Créer les dossiers nécessaires :

```
mkdir -p models/staging

mkdir -p models/marts
```

mettez y les fichiers
-- models/staging/stg_weather.sql
-- models/staging/stg_covid.sql
-- models/mart/daily_weather_metrics.sqlCréation et récupération de la clé API OpenWeatherMap

1. Créer un compte :

Aller sur https://home.openweathermap.org/users/sign_up

2. Connexion :

Se connecter à https://home.openweathermap.org

3. Générer une clé API :

• Aller dans le menu API keys.

• Cliquer sur "Generate" ou utiliser la clé par défaut fournie.

• Donner un nom à la clé (exemple : airflow_project_key).
Vous utiliserez cette clé dans le data_extraction_dag.py pour recupérer les données de meteo(weather)

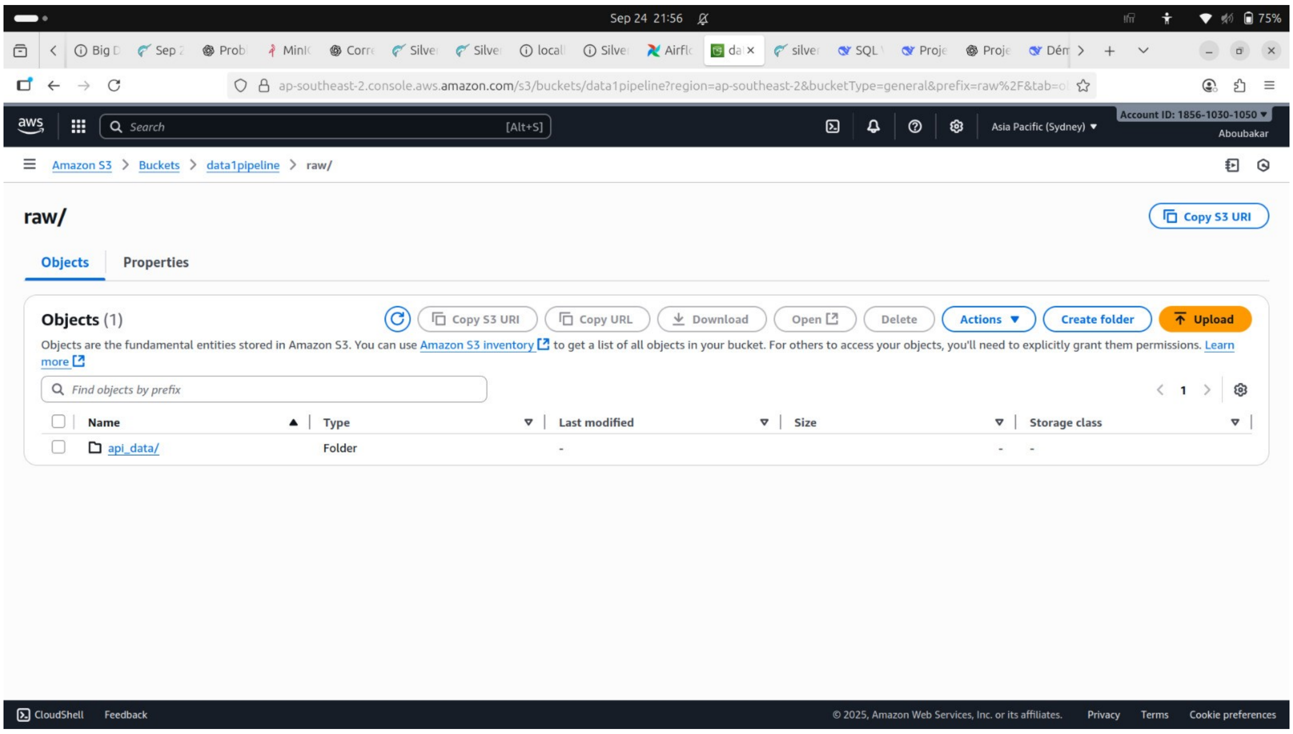
4.
1️⃣ Accéder au conteneur Airflow
Identifie d'abord le nom du conteneur Airflow en cours d'exécution :
avec docker ps
Ensuite, connecte-toi au conteneur (par ex. le scheduler ou le webserver)
avec la commande : docker exec -it airflow-scheduler-1 bash

5.
2️⃣ Installer dbt et dbt avec l adaptateur duckdb dans le
conteneur avec : pip install --no-cache-dir dbt-core dbt-
duckdb

6. Vérifier l'installation avec
dbt –version
vous dois voir quelque chose comme :
installed version: 1.8.x
plugins:
- duckdb: 1.8.x

7. ouvrir la page web de airflow http://localhost:8080/
ET executé les dag : data_extraction_dag et dbt_pipeline_dag

Resultats



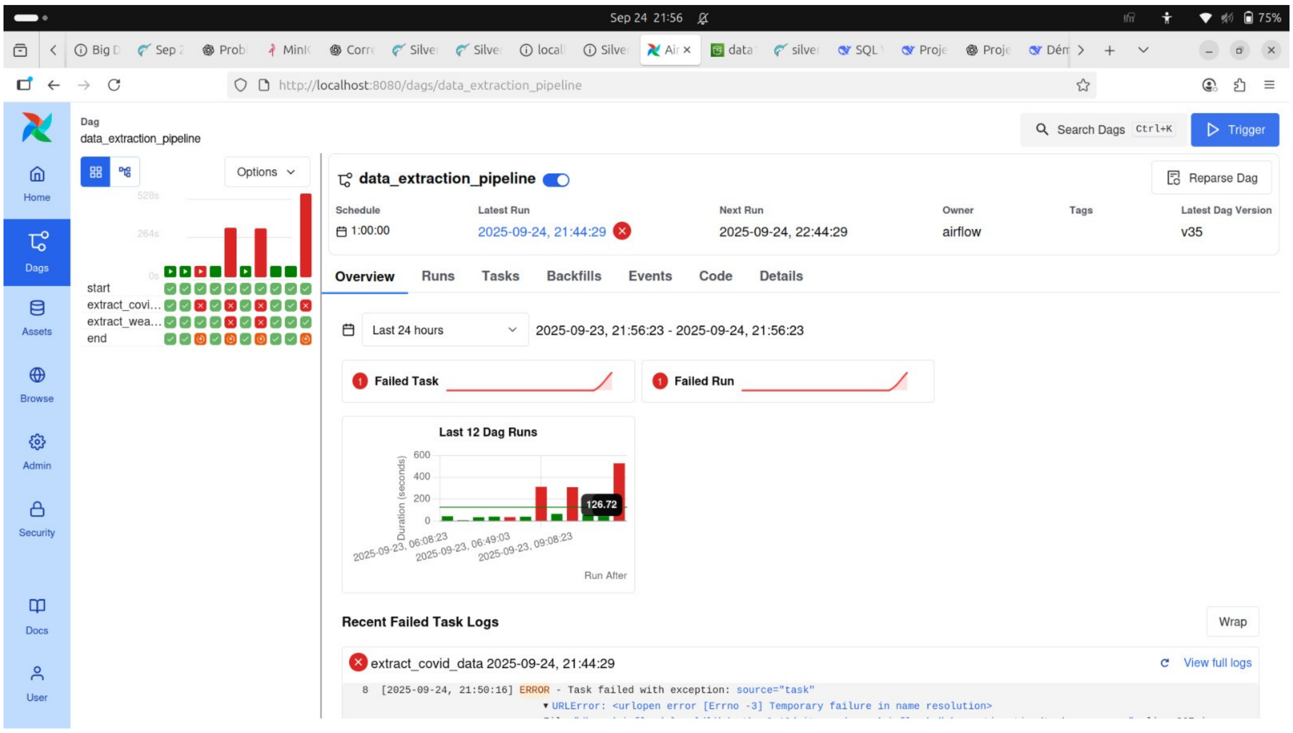
raw/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
api_data/	Folder	-	-	-

On verifie qu'il y a des données dans le bucket



Dag data_extraction_pipeline

Schedule	Latest Run	Next Run	Owner	Tags	Latest Dag Version
1:00:00	2025-09-24, 21:44:29	2025-09-24, 22:44:29	airflow		v35

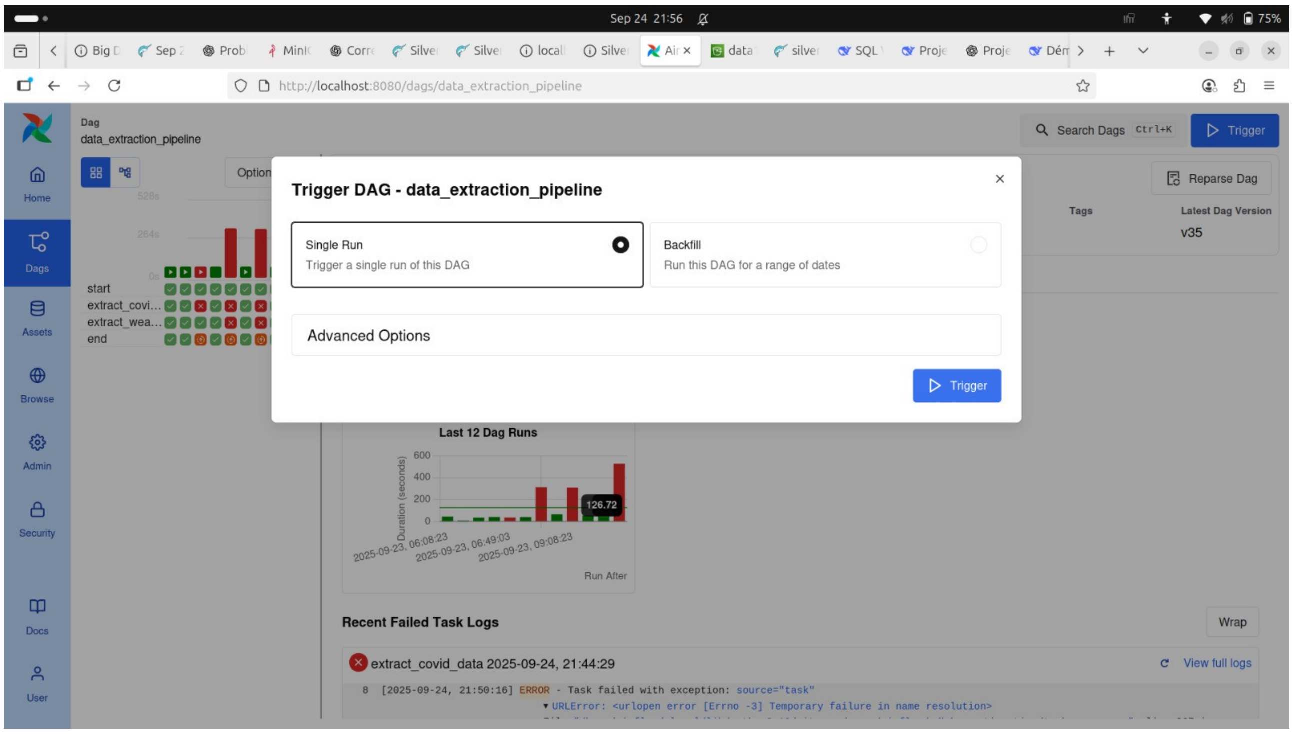
Last 24 hours 2025-09-23, 21:56:23 - 2025-09-24, 21:56:23

Recent Failed Task Logs

extract_covid_data 2025-09-24, 21:44:29

```
[2025-09-24, 21:50:16] ERROR - Task failed with exception: source="task"
URLError: <urlopen error [Errno -3] Temporary failure in name resolution>
```

Le pipeline d'extraction des données des API et le stockage vers S3



Trigger DAG - data_extraction_pipeline

Single Run [✓] — Trigger a single run of this DAG

Backfill — Run this DAG for a range of dates

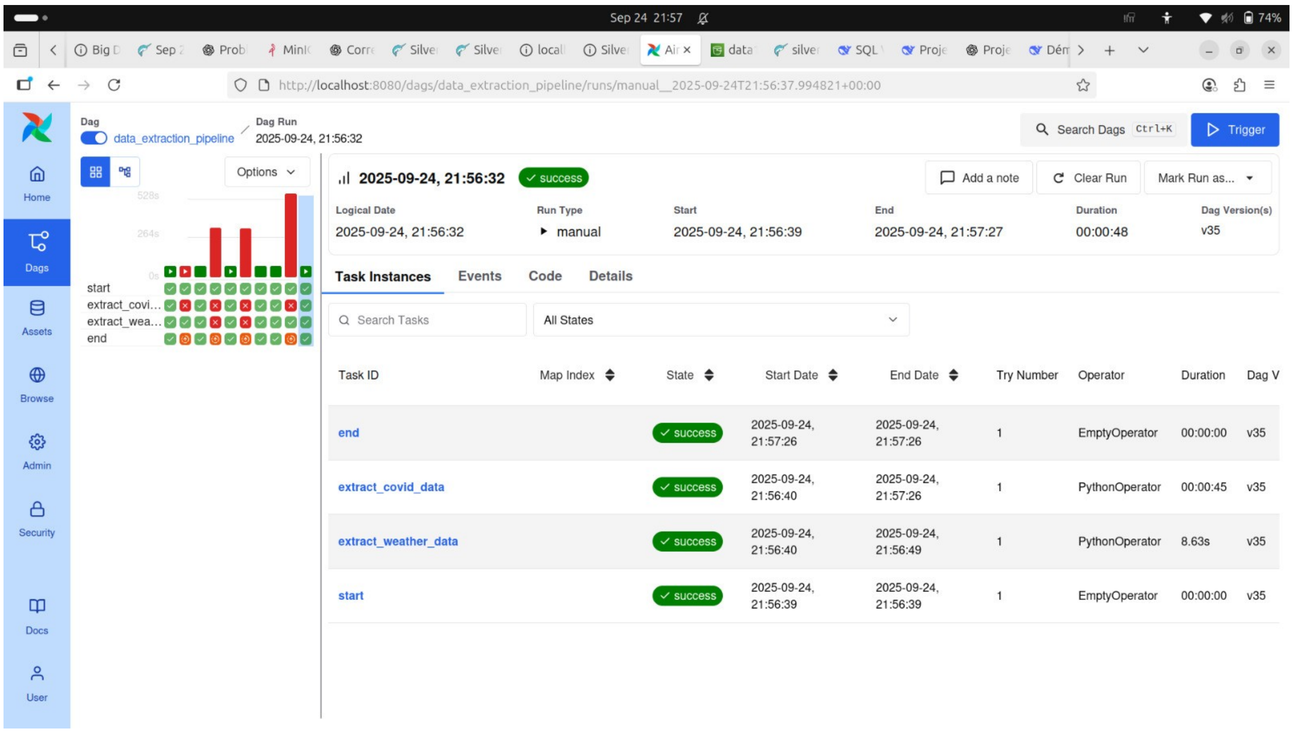
Advanced Options

Recent Failed Task Logs

extract_covid_data 2025-09-24, 21:44:29

```
8  [2025-09-24, 21:50:16] ERROR - Task failed with exception: source="task"
                          ▼URLError: <urlopen error [Errno -3] Temporary failure in name resolution>
```

On lance le pipeline d’extraction de données des API de Covid et Météo et on envoie dans le bucket S3 Selon le script data_extraction_dag.py_py qui dans dags du dossier airflow



Dag data_extraction_pipeline / Dag Run 2025-09-24, 21:56:32

2025-09-24, 21:56:32 ✓ success

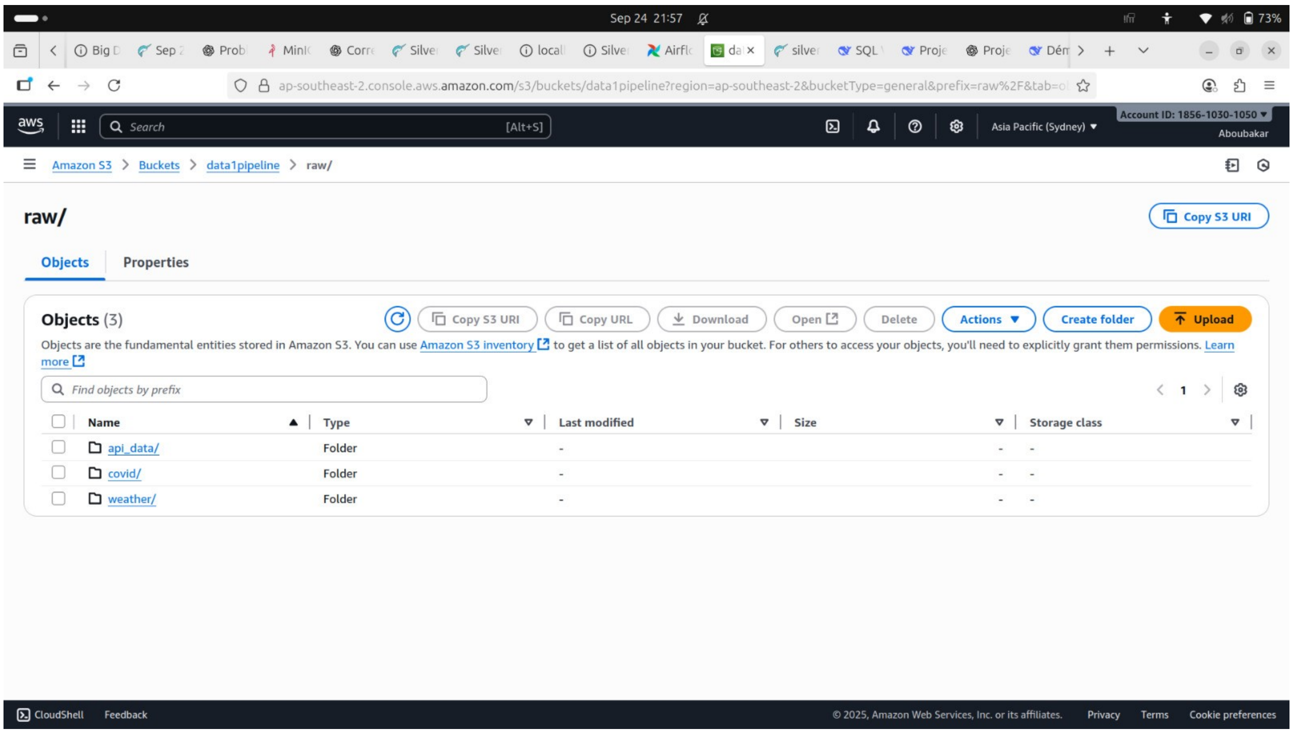
Logical Date	Run Type	Start	End	Duration	Dag Version(s)
2025-09-24, 21:56:32	manual	2025-09-24, 21:56:39	2025-09-24, 21:57:27	00:00:48	v35

Task Instances · Events · Code · Details

Task ID	Map Index	State	Start Date	End Date	Try Number	Operator	Duration	Dag V
end		✓ success	2025-09-24, 21:57:26	2025-09-24, 21:57:26	1	EmptyOperator	00:00:00	v35
extract_covid_data		✓ success	2025-09-24, 21:56:40	2025-09-24, 21:57:26	1	PythonOperator	00:00:45	v35
extract_weather_data		✓ success	2025-09-24, 21:56:40	2025-09-24, 21:56:49	1	PythonOperator	8.63s	v35
start		✓ success	2025-09-24, 21:56:39	2025-09-24, 21:56:39	1	EmptyOperator	00:00:00	v35

http://localhost:8080/dags/data_extraction_pipeline/runs/manual__2025-09-24T21:56:37.994821+00:00
Add a note
Clear Run
Mark Run as...
Search Tasks
All States

Le pipeline éxécuté avec succès


Sep 24 21:57
ap-southeast-2.console.aws.amazon.com/s3/buckets/data1pipeline?region=ap-southeast-2&bucketType=general&prefix=raw%2F&tab=o
Amazon S3 > Buckets > data1pipeline > raw/
Copy S3 URI
Objects
Properties
Copy URL
Download
Open
Delete
Actions
Create folder
Upload

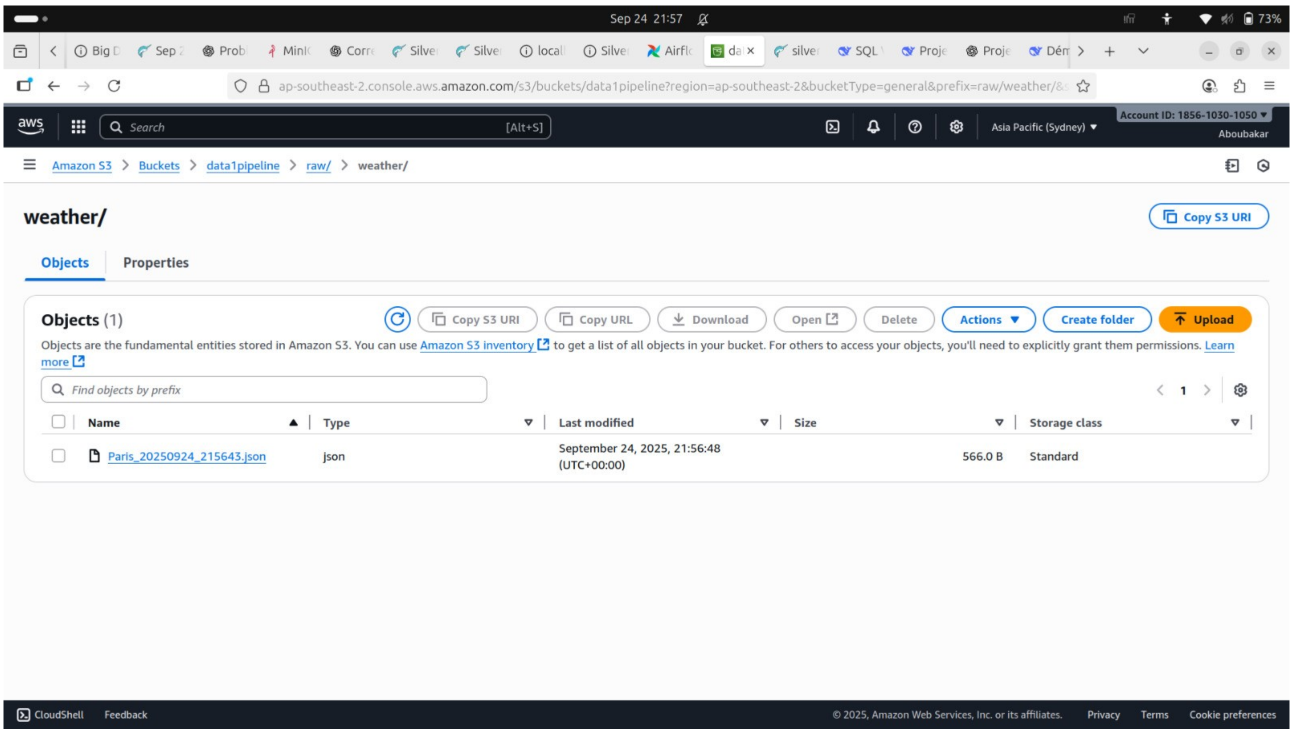
raw/

Objects (3)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
api_data/	Folder	-	-	-
covid/	Folder	-	-	-
weather/	Folder	-	-	-

<u>Sous bucket créés</u>


Sep 24 21:57
ap-southeast-2.console.aws.amazon.com/s3/buckets/data1pipeline?region=ap-southeast-2&bucketType=general&prefix=raw/weather/&
Amazon S3 > Buckets > data1pipeline > raw/ > weather/
Copy S3 URI
Objects
Properties
Copy URL
Download
Open
Delete
Actions
Create folder
Upload

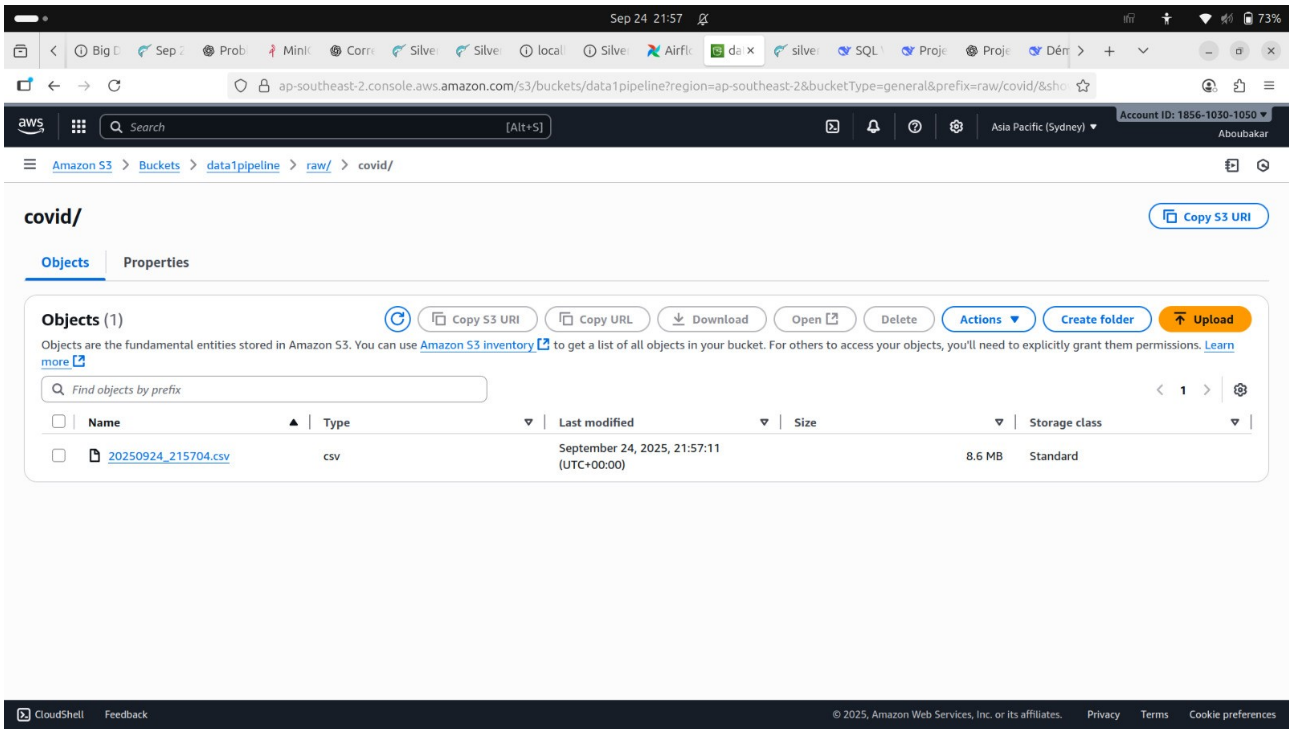
weather/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
Paris_20250924_215643.json	json	September 24, 2025, 21:56:48 (UTC+00:00)	566.0 B	Standard

<u>données dans le sous bucket weather</u>


Amazon S3 > Buckets > data1pipeline > raw/ > covid/

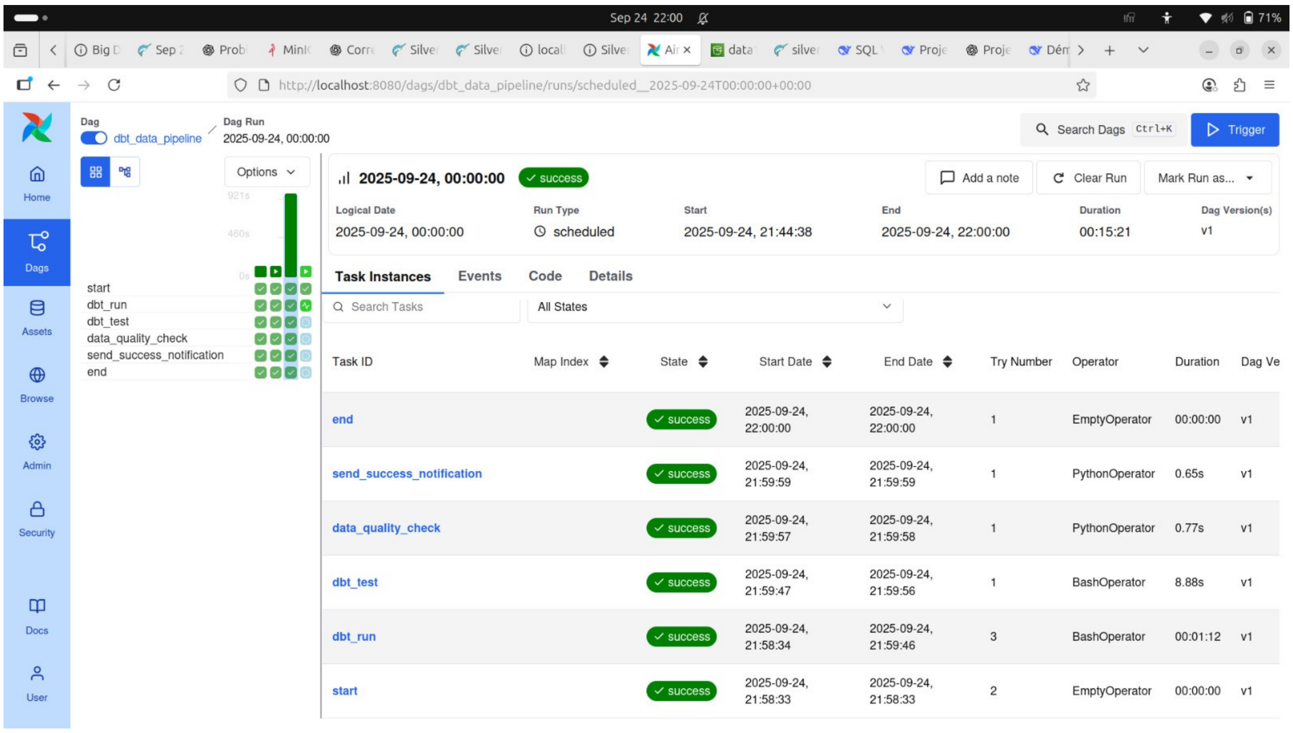
covid/

Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
20250924_215704.csv	csv	September 24, 2025, 21:57:11 (UTC+00:00)	8.6 MB	Standard

données dans le sous bucket covid


http://localhost:8080/dags/dbt_data_pipeline/runs/scheduled__2025-09-24T00:00:00+00:00

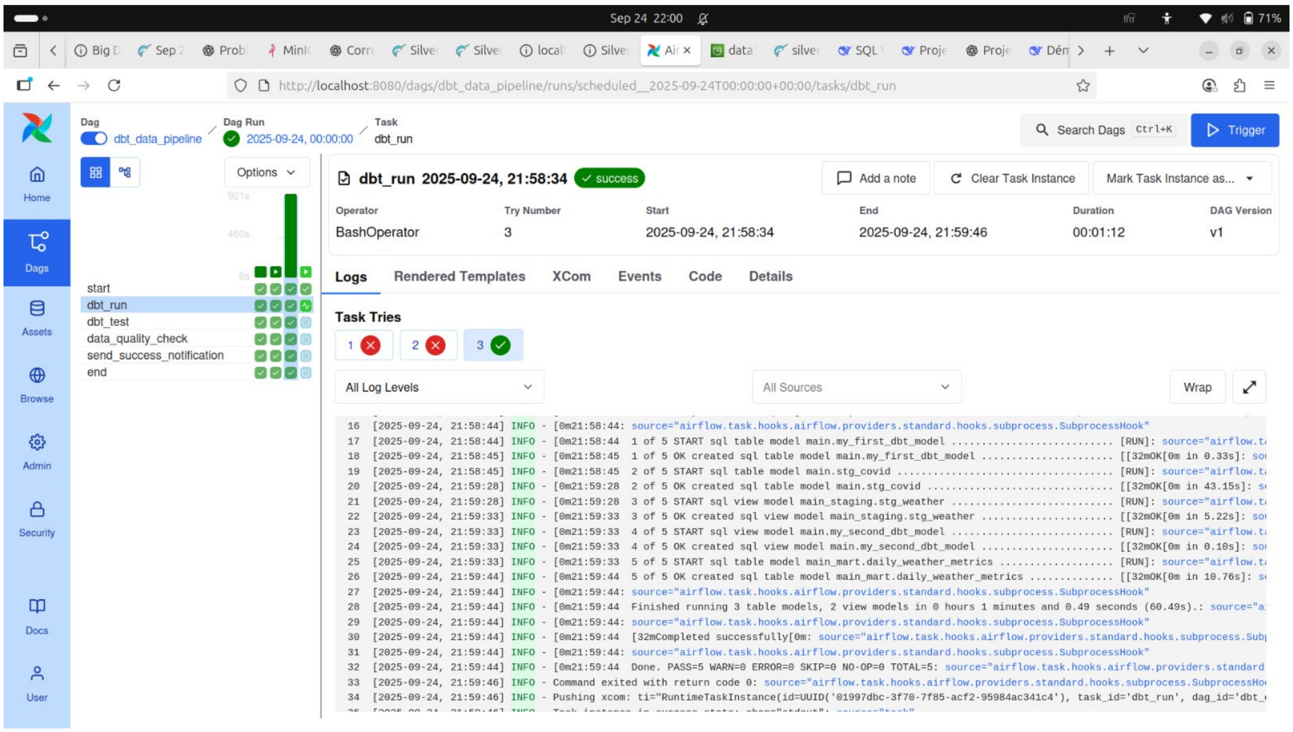
Dag: dbt_data_pipeline / Dag Run: 2025-09-24, 00:00:00

2025-09-24, 00:00:00 success

Logical Date	Run Type	Start	End	Duration	Dag Version(s)
2025-09-24, 00:00:00	scheduled	2025-09-24, 21:44:38	2025-09-24, 22:00:00	00:15:21	v1

Task ID	Map Index	State	Start Date	End Date	Try Number	Operator	Duration	Dag Ve
end		success	2025-09-24, 22:00:00	2025-09-24, 22:00:00	1	EmptyOperator	00:00:00	v1
send_success_notification		success	2025-09-24, 21:59:59	2025-09-24, 21:59:59	1	PythonOperator	0.65s	v1
data_quality_check		success	2025-09-24, 21:59:57	2025-09-24, 21:59:58	1	PythonOperator	0.77s	v1
dbt_test		success	2025-09-24, 21:59:47	2025-09-24, 21:59:56	1	BashOperator	8.88s	v1
dbt_run		success	2025-09-24, 21:58:34	2025-09-24, 21:59:46	3	BashOperator	00:01:12	v1
start		success	2025-09-24, 21:58:33	2025-09-24, 21:58:33	2	EmptyOperator	00:00:00	v1

Execution du pipeline de transformation dbt des données stocker dans le sous bucket covid et weather sur S3



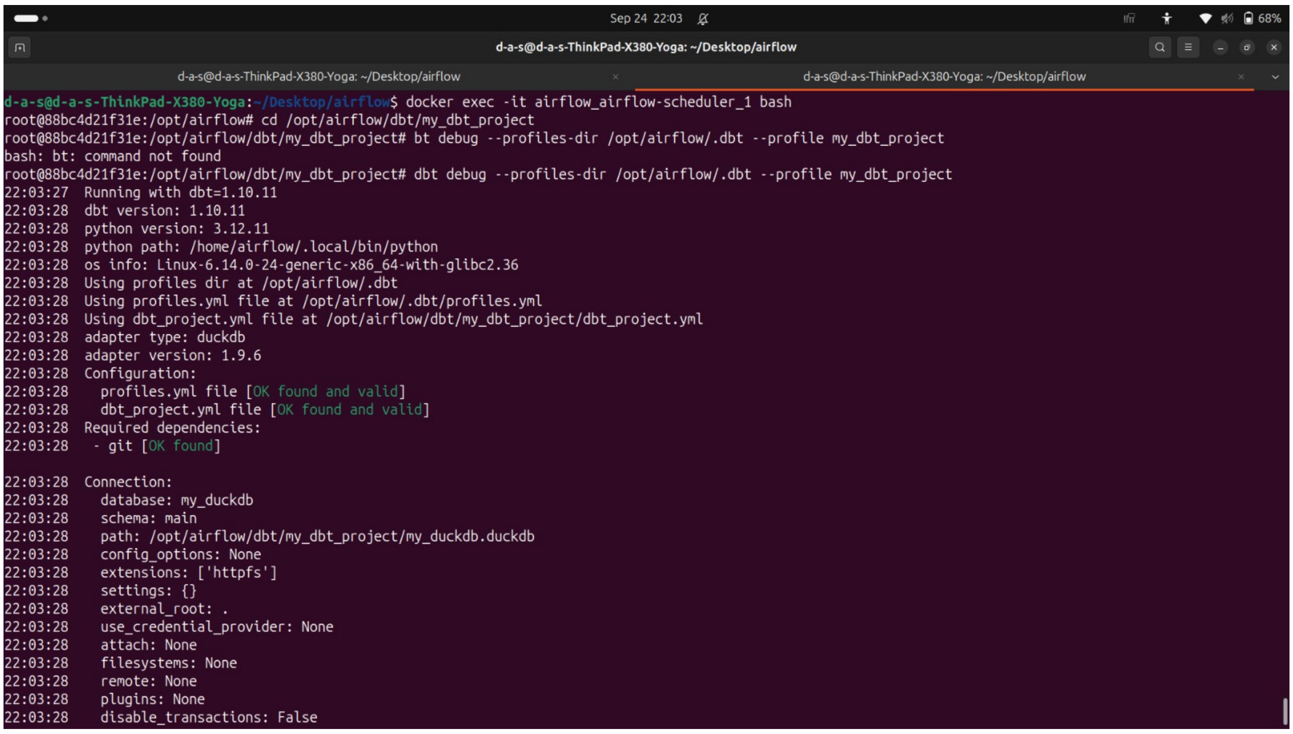
Dag: dbt_data_pipeline / Dag Run: 2025-09-24, 00:00:00 / Task: dbt_run

dbt_run 2025-09-24, 21:58:34 success

Operator	Try Number	Start	End	Duration	DAG Version
BashOperator	3	2025-09-24, 21:58:34	2025-09-24, 21:59:46	00:01:12	v1

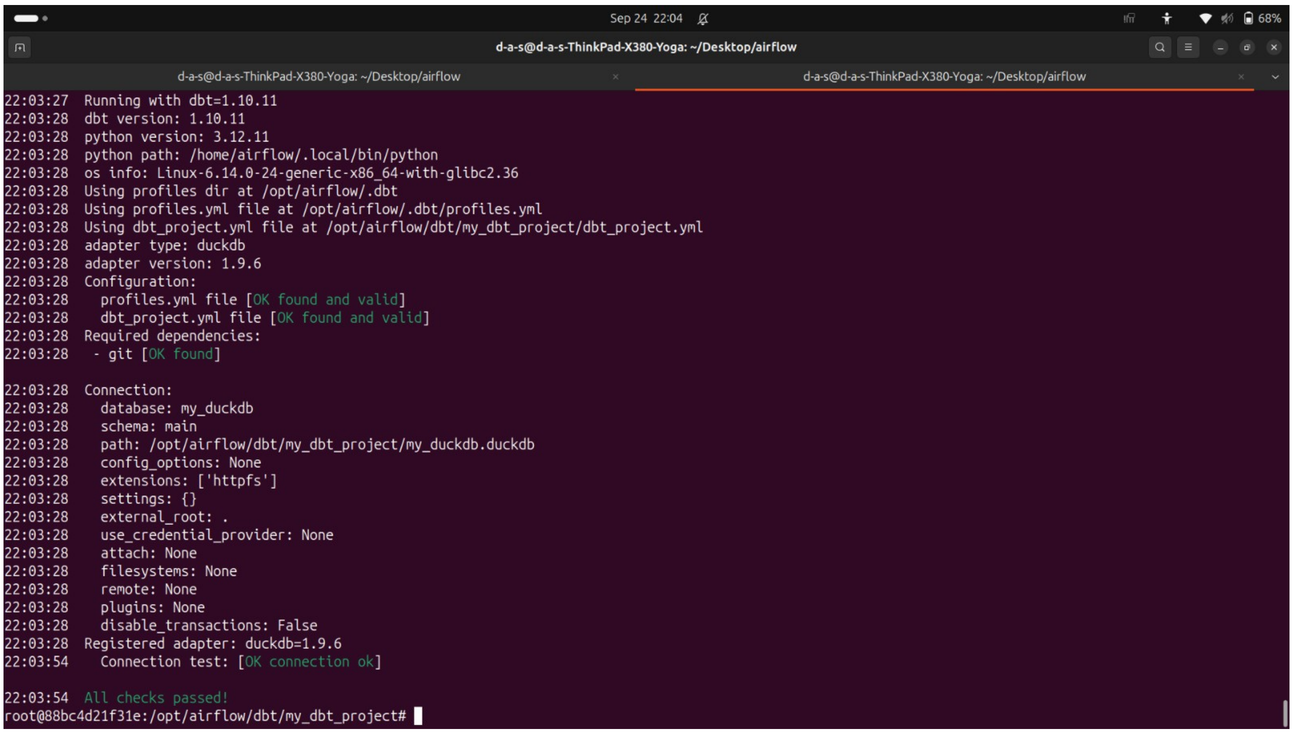
```
[2025-09-24, 21:58:44] INFO - [0m21:58:44: source="airflow.task.hooks.airflow.providers.standard.hooks.subprocess.SubprocessHook"
[2025-09-24, 21:58:44] INFO - [0m21:58:44  1 of 5 START sql table model main.my_first_dbt_model .......................... [RUN]: source="airflow.t
[2025-09-24, 21:58:45] INFO - [0m21:58:45  1 of 5 OK created sql table model main.my_first_dbt_model ..................... [[32mOK[0m in 0.33s]: so
[2025-09-24, 21:58:45] INFO - [0m21:58:45  2 of 5 START sql table model main.stg_covid ................................... [RUN]: source="airflow.t
[2025-09-24, 21:59:28] INFO - [0m21:59:28  2 of 5 OK created sql table model main.stg_covid .............................. [[32mOK[0m in 43.15s]: s
[2025-09-24, 21:59:28] INFO - [0m21:59:28  3 of 5 START sql view model main_staging.stg_weather .......................... [RUN]: source="airflow.t
[2025-09-24, 21:59:33] INFO - [0m21:59:33  3 of 5 OK created sql view model main_staging.stg_weather ..................... [[32mOK[0m in 5.22s]: so
[2025-09-24, 21:59:33] INFO - [0m21:59:33  4 of 5 START sql view model main.my_second_dbt_model .......................... [RUN]: source="airflow.t
[2025-09-24, 21:59:33] INFO - [0m21:59:33  4 of 5 OK created sql view model main.my_second_dbt_model ..................... [[32mOK[0m in 0.10s]: so
[2025-09-24, 21:59:33] INFO - [0m21:59:33  5 of 5 START sql table model main_mart.daily_weather_metrics .................. [RUN]: source="airflow.t
[2025-09-24, 21:59:44] INFO - [0m21:59:44  5 of 5 OK created sql table model main_mart.daily_weather_metrics ............. [[32mOK[0m in 10.76s]: s
[2025-09-24, 21:59:44] INFO - [0m21:59:44: source="airflow.task.hooks.airflow.providers.standard.hooks.subprocess.SubprocessHook"
[2025-09-24, 21:59:44] INFO - [0m21:59:44  Finished running 3 table models, 2 view models in 0 hours 1 minutes and 0.49 seconds (60.49s).: source="a
[2025-09-24, 21:59:44] INFO - [0m21:59:44: source="airflow.task.hooks.airflow.providers.standard.hooks.subprocess.SubprocessHook"
[2025-09-24, 21:59:44] INFO - [0m21:59:44  [32mCompleted successfully[0m: source="airflow.task.hooks.airflow.providers.standard.hooks.subprocess.Sub
[2025-09-24, 21:59:44] INFO - [0m21:59:44: source="airflow.task.hooks.airflow.providers.standard.hooks.subprocess.SubprocessHook"
[2025-09-24, 21:59:44] INFO - [0m21:59:44  Done. PASS=5 WARN=0 ERROR=0 SKIP=0 NO-OP=0 TOTAL=5: source="airflow.task.hooks.airflow.providers.standard
[2025-09-24, 21:59:46] INFO - Command exited with return code 0: source="airflow.task.hooks.airflow.providers.standard.hooks.subprocess.SubprocessHo
[2025-09-24, 21:59:46] INFO - Pushing xcom: ti="RuntimeTaskInstance(id=UUID('01997dbc-3f70-7f85-acf2-95984ac341c4'), task_id='dbt_run', dag_id='dbt_
```

Tout les modèles sont éxécutés



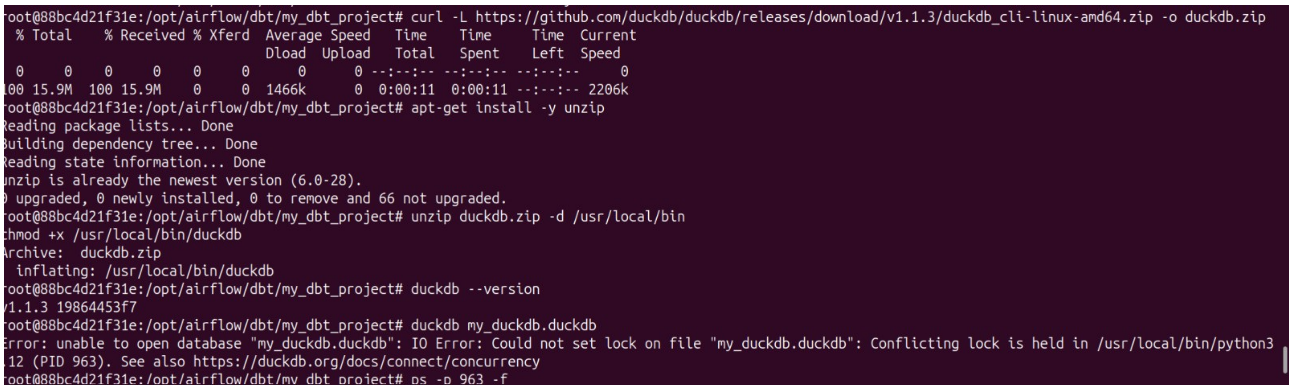
```
d-a-s@d-a-s-ThinkPad-X380-Yoga:~/Desktop/airflow$ docker exec -it airflow_airflow-scheduler_1 bash
root@88bc4d21f31e:/opt/airflow# cd /opt/airflow/dbt/my_dbt_project
root@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# bt debug --profiles-dir /opt/airflow/.dbt --profile my_dbt_project
bash: bt: command not found
root@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# dbt debug --profiles-dir /opt/airflow/.dbt --profile my_dbt_project
22:03:27  Running with dbt=1.10.11
22:03:28  dbt version: 1.10.11
22:03:28  python version: 3.12.11
22:03:28  python path: /home/airflow/.local/bin/python
22:03:28  os info: Linux-6.14.0-24-generic-x86_64-with-glibc2.36
22:03:28  Using profiles dir at /opt/airflow/.dbt
22:03:28  Using profiles.yml file at /opt/airflow/.dbt/profiles.yml
22:03:28  Using dbt_project.yml file at /opt/airflow/dbt/my_dbt_project/dbt_project.yml
22:03:28  adapter type: duckdb
22:03:28  adapter version: 1.9.6
22:03:28  Configuration:
22:03:28    profiles.yml file [OK found and valid]
22:03:28    dbt_project.yml file [OK found and valid]
22:03:28  Required dependencies:
22:03:28   - git [OK found]

22:03:28  Connection:
22:03:28    database: my_duckdb
22:03:28    schema: main
22:03:28    path: /opt/airflow/dbt/my_dbt_project/my_duckdb.duckdb
22:03:28    config_options: None
22:03:28    extensions: ['httpfs']
22:03:28    settings: {}
22:03:28    external_root: .
22:03:28    use_credential_provider: None
22:03:28    attach: None
22:03:28    filesystems: None
22:03:28    remote: None
22:03:28    plugins: None
22:03:28    disable_transactions: False
```

On se connecte au conteneur airflow scheduler où on a installé dbt et on éxécute la commande pour voir que la connexion de DuckDB à S3 est fait



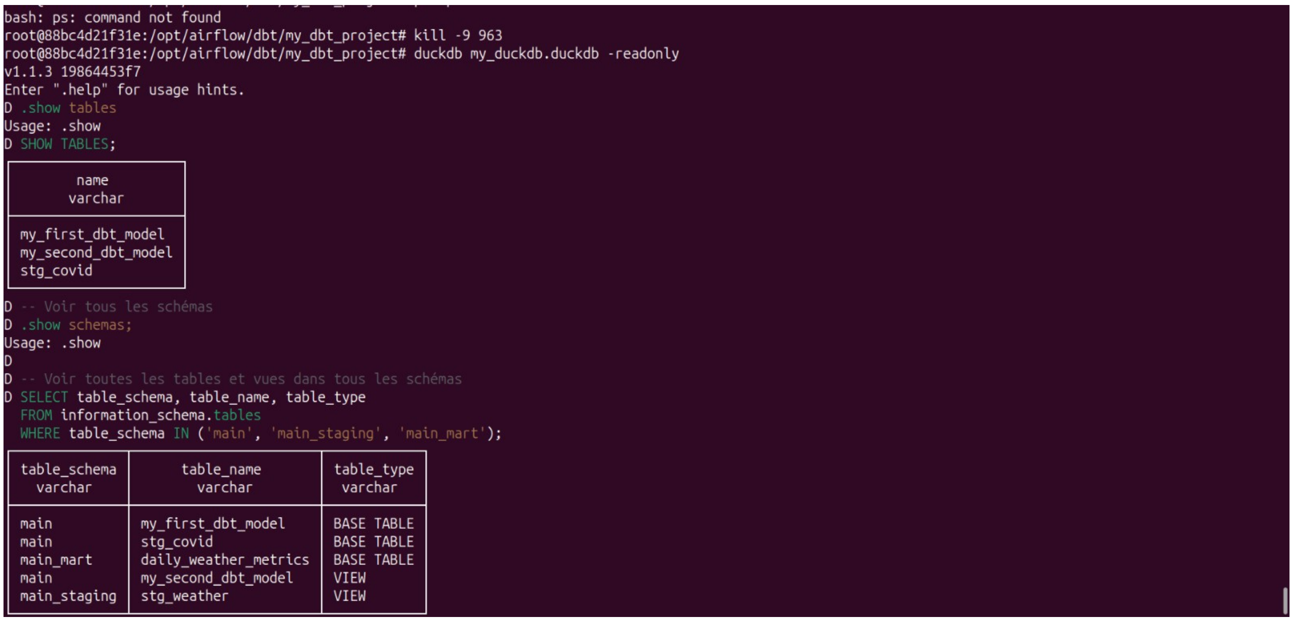
```
22:03:27  Running with dbt=1.10.11
22:03:28  dbt version: 1.10.11
22:03:28  python version: 3.12.11
22:03:28  python path: /home/airflow/.local/bin/python
22:03:28  os info: Linux-6.14.0-24-generic-x86_64-with-glibc2.36
22:03:28  Using profiles dir at /opt/airflow/.dbt
22:03:28  Using profiles.yml file at /opt/airflow/.dbt/profiles.yml
22:03:28  Using dbt_project.yml file at /opt/airflow/dbt/my_dbt_project/dbt_project.yml
22:03:28  adapter type: duckdb
22:03:28  adapter version: 1.9.6
22:03:28  Configuration:
22:03:28    profiles.yml file [OK found and valid]
22:03:28    dbt_project.yml file [OK found and valid]
22:03:28  Required dependencies:
22:03:28   - git [OK found]

22:03:28  Connection:
22:03:28    database: my_duckdb
22:03:28    schema: main
22:03:28    path: /opt/airflow/dbt/my_dbt_project/my_duckdb.duckdb
22:03:28    config_options: None
22:03:28    extensions: ['httpfs']
22:03:28    settings: {}
22:03:28    external_root: .
22:03:28    use_credential_provider: None
22:03:28    attach: None
22:03:28    filesystems: None
22:03:28    remote: None
22:03:28    plugins: None
22:03:28    disable_transactions: False
22:03:28  Registered adapter: duckdb=1.9.6
22:03:54    Connection test: [OK connection ok]

22:03:54  All checks passed!
root@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# 
```

Connexion réussi



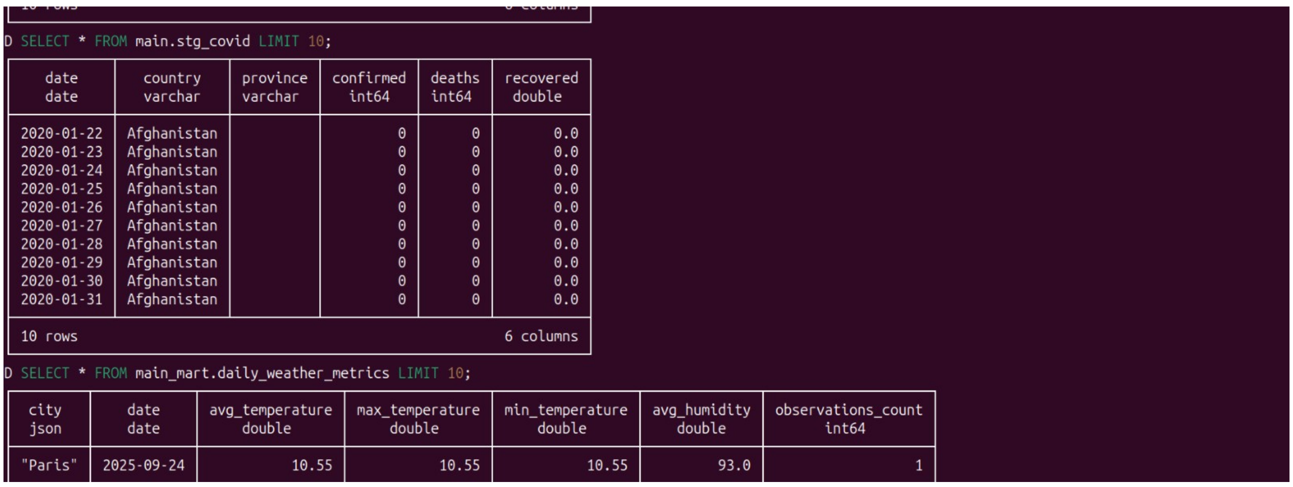
```
oot@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# curl -L https://github.com/duckdb/duckdb/releases/download/v1.1.3/duckdb_cli-linux-amd64.zip -o duckdb.zip
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 15.9M  100 15.9M    0     0  1466k      0  0:00:11  0:00:11 --:--:-- 2206k
oot@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# apt-get install -y unzip
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
unzip is already the newest version (6.0-28).
0 upgraded, 0 newly installed, 0 to remove and 66 not upgraded.
oot@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# unzip duckdb.zip -d /usr/local/bin
chmod +x /usr/local/bin/duckdb
Archive:  duckdb.zip
  inflating: /usr/local/bin/duckdb
oot@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# duckdb --version
v1.1.3 19864453f7
oot@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# duckdb my_duckdb.duckdb
Error: unable to open database "my_duckdb.duckdb": IO Error: Could not set lock on file "my_duckdb.duckdb": Conflicting lock is held in /usr/local/bin/python3
.12 (PID 963). See also https://duckdb.org/docs/connect/concurrency
oot@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# ps -p 963 -f
```

Installation de DuckDB dans le conteneur pour verifier qu'il y a effectiovement les tables liés à l'éxécution de dbt run



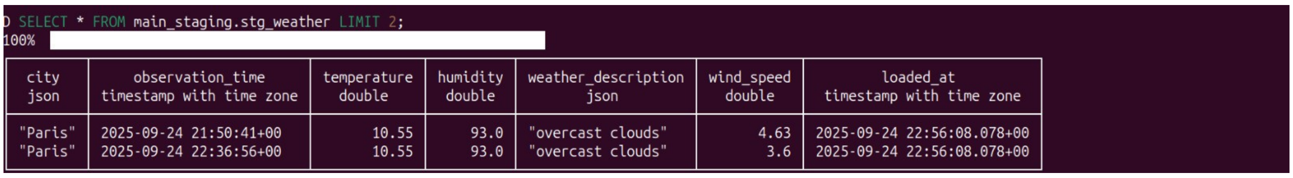
```
bash: ps: command not found
root@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# kill -9 963
root@88bc4d21f31e:/opt/airflow/dbt/my_dbt_project# duckdb my_duckdb.duckdb -readonly
v1.1.3 19864453f7
Enter ".help" for usage hints.
D .show tables
Usage: .show
D SHOW TABLES;
┌───────────────────┐
│       name        │
│      varchar      │
├───────────────────┤
│ my_first_dbt_model│
│ my_second_dbt_model│
│ stg_covid         │
└───────────────────┘
D -- Voir tous les schémas
D .show schemas;
Usage: .show
D
D -- Voir toutes les tables et vues dans tous les schémas
D SELECT table_schema, table_name, table_type
  FROM information_schema.tables
  WHERE table_schema IN ('main', 'main_staging', 'main_mart');
┌──────────────┬───────────────────────┬────────────┐
│ table_schema │      table_name       │ table_type │
│   varchar    │        varchar        │  varchar   │
├──────────────┼───────────────────────┼────────────┤
│ main         │ my_first_dbt_model    │ BASE TABLE │
│ main         │ stg_covid             │ BASE TABLE │
│ main_mart    │ daily_weather_metrics │ BASE TABLE │
│ main         │ my_second_dbt_model   │ VIEW       │
│ main_staging │ stg_weather           │ VIEW       │
└──────────────┴───────────────────────┴────────────┘
```

Verification qu'on a toutes les tables



```
D SELECT * FROM main.stg_covid LIMIT 10;
┌────────────┬─────────────┬──────────┬───────────┬────────┬───────────┐
│    date    │   country   │ province │ confirmed │ deaths │ recovered │
│    date    │   varchar   │ varchar  │   int64   │ int64  │  double   │
├────────────┼─────────────┼──────────┼───────────┼────────┼───────────┤
│ 2020-01-22 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-23 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-24 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-25 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-26 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-27 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-28 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-29 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-30 │ Afghanistan │          │         0 │      0 │       0.0 │
│ 2020-01-31 │ Afghanistan │          │         0 │      0 │       0.0 │
├────────────┴─────────────┴──────────┴───────────┴────────┴───────────┤
│ 10 rows                                                    6 columns │
└──────────────────────────────────────────────────────────────────────┘
D SELECT * FROM main_mart.daily_weather_metrics LIMIT 10;
┌─────────┬────────────┬─────────────────┬─────────────────┬─────────────────┬──────────────┬────────────────────┐
│  city   │    date    │ avg_temperature │ max_temperature │ min_temperature │ avg_humidity │ observations_count │
│  json   │    date    │     double      │     double      │     double      │    double    │       int64        │
├─────────┼────────────┼─────────────────┼─────────────────┼─────────────────┼──────────────┼────────────────────┤
│ "Paris" │ 2025-09-24 │           10.55 │           10.55 │           10.55 │         93.0 │                  1 │
```

<u>**données dans stagging covid et metrics de météo**</u>



```
D SELECT * FROM main_staging.stg_weather LIMIT 2;
100% ▕████████████████████████████████████████████████████████████▏
┌─────────┬──────────────────────────┬─────────────┬──────────┬─────────────────────┬────────────┬────────────────────────────┐
│  city   │     observation_time     │ temperature │ humidity │ weather_description │ wind_speed │         loaded_at          │
│  json   │ timestamp with time zone │   double    │  double  │        json         │   double   │  timestamp with time zone  │
├─────────┼──────────────────────────┼─────────────┼──────────┼─────────────────────┼────────────┼────────────────────────────┤
│ "Paris" │ 2025-09-24 21:50:41+00   │       10.55 │     93.0 │ "overcast clouds"   │       4.63 │ 2025-09-24 22:56:08.078+00 │
│ "Paris" │ 2025-09-24 22:36:56+00   │       10.55 │     93.0 │ "overcast clouds"   │        3.6 │ 2025-09-24 22:56:08.078+00 │
└─────────┴──────────────────────────┴─────────────┴──────────┴─────────────────────┴────────────┴────────────────────────────┘
```

<u>**données dans stg_weather**</u>