

Documentation sur la création d'un data warehouse pour l'analyse des ventes

1. Création d'un data lake avec amazon S3

On part sur AWS , ensuite on recherche le service S3 qui se présente comme ça :

The screenshot shows the Amazon S3 console interface. On the left, there's a sidebar with navigation links like 'Buckets', 'Access management and security', and 'Storage management and insights'. The main area is titled 'General purpose buckets' and shows three existing buckets: 'aws-glue-assets-355142185625-eu-west-3', 'ensae-student-data', and 'project-olist-ensae'. Each bucket entry includes its name, AWS Region (Europe (Paris) eu-west-3), and creation date. Below the table are two buttons: 'Account snapshot' and 'External access summary - new'. At the bottom right of the main area, there's a prominent orange 'Create bucket' button.

On crée un compartiment en créant sur le bouton **create bucket**

This screenshot is identical to the one above, showing the Amazon S3 console with the 'General purpose buckets' list. A red arrow points specifically to the orange 'Create bucket' button located at the bottom right of the main content area.

Après avoir crée notre bucket on charge les tables de notre base de données dans le bucket

Le bucket a déjà été créé et a été nommé projet-olist-ensae

Lorsque on clique sur projetct-olist-ensae on se retrouve sur cette partie ou on voit notre base de données et les différentes tables se trouvent dans le dossier bronze.

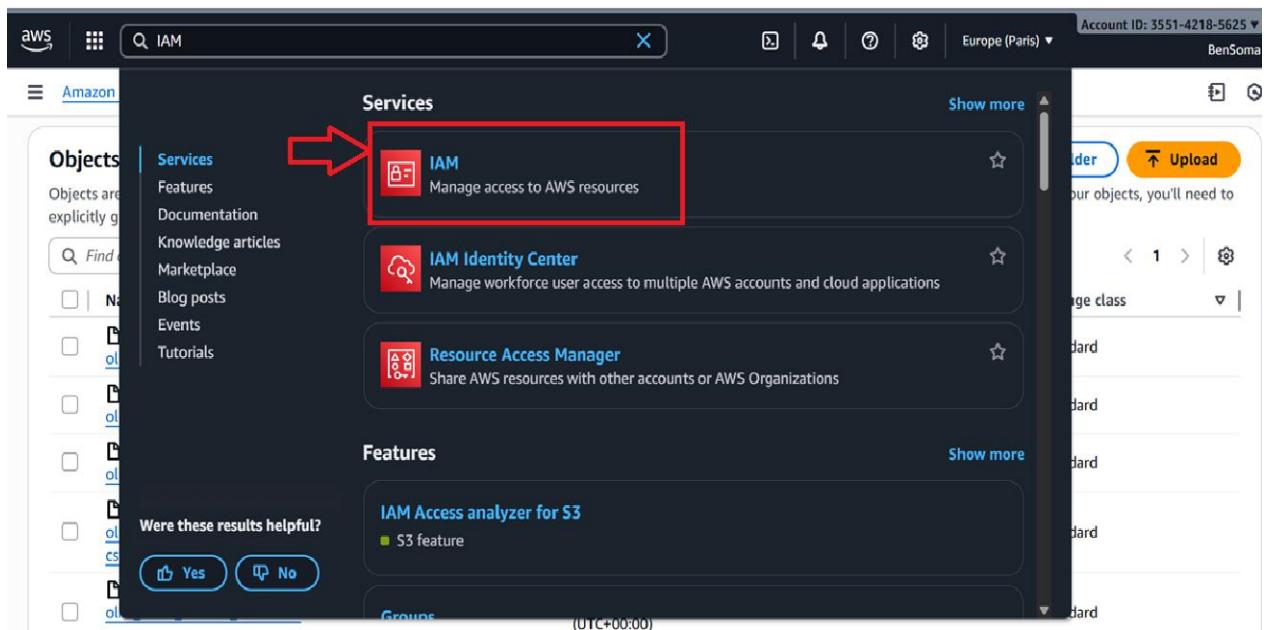
The screenshot shows the AWS S3 console interface. At the top, the path is 'Amazon S3 > Buckets > project-olist-ensae'. Below the path, the bucket name 'project-olist-ensae' is displayed with an 'Info' link. A navigation bar below the path includes tabs for 'Objects', 'Metadata', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Objects' tab is selected. In the main content area, the heading 'Objects (2)' is shown, followed by a toolbar with actions like 'Copy S3 URI', 'Copy URL', 'Download', 'Open L2', 'Delete', 'Actions', 'Create folder', and 'Upload'. A note below the toolbar states: '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](#)'. A search bar labeled 'Find objects by prefix' is present. The object list table has columns for Name, Type, Last modified, Size, and Storage class. Two entries are listed: 'bronze-parquet/' (Folder) and 'bronze/' (Folder). The entry 'bronze/' is circled in red.

Voici comment les différentes tables se présentent :

The screenshot shows the AWS S3 console interface, similar to the previous one but at a deeper level. The path is 'Amazon S3 > Buckets > project-olist-ensae > bronze/'. The heading 'Objects (9)' is shown, followed by a toolbar with actions like 'Copy S3 URI', 'Copy URL', 'Download', 'Open L2', 'Delete', 'Actions', 'Create folder', and 'Upload'. A note below the toolbar states: '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](#)'. A search bar labeled 'Find objects by prefix' is present. The object list table has columns for Name, Type, Last modified, Size, and Storage class. Five CSV files are listed: 'olist_customers_dataset.csv', 'olist_geolocation_dataset.csv', 'olist_order_items_dataset.csv', 'olist_order_payments_dataset.csv', and 'olist_order_reviews_dataset.csv'. All files are of type 'csv', created on December 11, 2025, at 18:59:41 (UTC+00:00), and have a size between 5.5 MB and 14.7 MB, all in 'Standard' storage class.

2. Crédation d'un groupe et de plusieurs IAM ID (5 au total)

Partir au niveau de la barre de recherche de la console et rechercher IAM



On clique sur IAM

On clique à présent sur User_groups pour créer les différents IAM users avec un accès complet aux ressources

La creation des users va permettre aux différents membres de l'équipe d'accéder à tout le travail , faire des modifications .

3. Création d'un bucket sur AWS Glue

On recherche Glue dans la barre de recherche

A screenshot of the AWS IAM search results page. The search bar at the top contains the text "glue". The left sidebar shows navigation options like Services, Features, Resources, Documentation, Knowledge articles, Marketplace, Blog posts, and Events. A red arrow points to the "AWS Glue" service card, which is highlighted with a red box. The card describes AWS Glue as a serverless data integration service. Below the services section, there's a "Features" section with a card for "AWS Glue Studio". At the bottom, there are "Were these results helpful?" buttons for "Yes" and "No".

On clique sur AWS Glue

Après avoir cliqué sur ça on clique sur l'onglet à droite sur databases

A screenshot of the AWS Glue Welcome page. The left sidebar has a "Data Catalog" section with a "Databases" link, which is highlighted with a red box and a red arrow pointing to it. The main content area includes sections for "Welcome to AWS Glue", "Prepare your account for AWS Glue", "Catalog and search for datasets", "Move and transform data", "Resources and tutorials", and "Data integration and management".

AWS Glue

Databases (1)

Last updated (UTC) December 13, 2025 at 01:27:42

Add database

Name	Description	Location URI	Created on (UTC)
olist-database	-	-	December 11, 2025 at 19:07:30

Filter databases

Getting started
ETL jobs
Visual ETL
Notebooks
Job run monitoring
Data Catalog tables
Data connections
Workflows (orchestration)

Data Catalog

Databases
Tables
Stream schema registries
Schemas
Connections
Crawlers
Classifiers
Catalog settings

On clique à présent sur « Add database » pour créer la base de données sur Glue qui va nous permettre de charger les tables depuis S3

On crée un CRAWLER en cliquant sur CRAWLER dans la barre latérale

AWS Glue > Crawlers

Crawlers

Last updated (UTC) December 13, 2025 at 01:32:42

Create crawler

Name	State	Schedule	Last run	Last run ...	Log	Table cha...
olist	Ready	-	-	-	-	-

Filter crawlers

Job run monitoring
Data Catalog tables
Data connections
Workflows (orchestration)

Data Catalog

Databases
Tables
Stream schema registries
Schemas
Connections
Crawlers
Classifiers
Catalog settings

Data Integration and ETL

Legacy pages

What's New Documentation

Après la creation de notre crawler on clique sur « run a crawler » et apres ca nous verrons les différentes tables qui ont été importées sur GLUE .

La base a été créée et se nomme olis-database on clique sur ca

The screenshot shows the AWS Glue Data Catalog interface. On the left, a sidebar menu includes 'AWS Glue' at the top, followed by 'Getting started', 'ETL jobs', 'Visual ETL', 'Notebooks', 'Job run monitoring', 'Data Catalog tables', 'Data connections', and 'Workflows (orchestration)'. Below this, under 'Data Catalog', are 'Databases' (which is currently selected and highlighted in blue), 'Tables', 'Stream schema registries', 'Schemas', 'Connections', 'Crawlers', 'Classifiers', and 'Catalog settings' (which has a red box drawn around it). The main content area is titled 'Databases (1)' and contains a sub-instruction: 'A database is a set of associated table definitions, organized into a logical group.' A search bar with the placeholder 'Filter databases' is present. The database list table has columns: 'Name' (with a checkbox), 'Description' (with an up arrow icon), 'Location URI' (with a down arrow icon), and 'Created on (UTC)' (with a down arrow icon). A single database entry, 'olist-database', is listed. The 'Created on (UTC)' column for this entry shows 'December 11, 2025 at 19:07:30'. At the top right of the main area are 'Edit', 'Delete', and 'Add database' buttons, with the 'Add database' button being highlighted with a red box and a red arrow pointing to it from the bottom. The top right corner also features a gear icon.

4. Transformation des tables CSV en parquet

Pourquoi parquet ?

Pour pouvoir gérer de grandes bases de données

Pour faire ça nous , toujours au niveau de GLUE dans la barre latérale à gauche nous cliquons sur « ETL JOB »

AWS Glue > Crawlers

Crawlers

A crawler connects to a data store, progresses through a prioritized list of classifiers to determine the schema for your data, and then creates metadata tables in your data catalog.

Crawlers (1) Info

Last updated (UTC) December 13, 2025 at 01:37:13

Name	State	Schedule	Last run	Last run ...	Log	Table cha...
olist	Ready		Succeeded	December 1...	View log	9 created

Dans ETL Jobs on clique sur script Editor

AWS Glue > Jobs

AWS Glue Studio Info

Create job Info

- Author in a visual interface focused on data flow. **Visual ETL**
- Author using an interactive code notebook. **Notebook**
- Author code with a script edit** **Script editor**

Your jobs (1) Info

Job name	Type	Created by	Last modified	AWS Glue version	Action
csv_to_parquet	Glue ETL	Script	12/11/2025, 7:34:48 PM	5.0	-

On choisit l'option spark pour le code

Le code suivant a été utilisé pour transformer les fichiers CSV en parquet :
Le code en question se trouve dans les fichiers annexes

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
```

```

from awsglue.job import Job

# -----
# Initialisation du job Glue
# -----
args = getResolvedOptions(sys.argv, ["JOB_NAME"])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# -----
# Emplacements S3
# -----
input_path = "s3://your-bucket-name/bronze/"      # tes CSV Olist
output_path = "s3://your-bucket-name/bronze-parquet/" # dossier Parquet à créer

# -----
# Lecture des CSV
# -----
df_csv = (
    spark.read
        .option("header", "true")
        .option("inferSchema", "true")  # (optionnel)
        .option("sep", ",")
        .csv(input_path)
)

print("Nombre de lignes chargées :", df_csv.count())
print("Schéma détecté :")
df_csv.printSchema()

# -----
# Écriture en Parquet
# -----
(
    df_csv.write
        .mode("overwrite")      # écrase si le dossier existe
        .option("compression", "snappy")
        .parquet(output_path)
)

print("Transformation CSV → Parquet terminée.")

job.commit()

```

Lorsque nous partons dans S3 nous constatons que y a un nouveau dossier qui a été créé et qui contient nos fichiers parquet .

Pourquoi des tables sous format parquet ?

Parce que le format Parquet est plus performant que le CSV : il est orienté colonnes, permet une **compression efficace**, réduit l'espace de stockage sur S3, **accélère les lectures avec Spark**, conserve le **schéma des données** et est mieux adapté aux **analyses Big Data**.

Résumé des services utilisés et leur rôle jusqu'à présent :

Service AWS	Utilité dans le projet
Amazon S3	Stockage scalable et sécurisé des données brutes et transformées.
IAM	Gestion des accès et des permissions pour les utilisateurs et services.
AWS Glue	Catalogage des données, crawling automatique, et transformation ETL serverless.
Glue Data Catalog	Catalogue centralisé des métadonnées pour interroger les données avec Athena/Redshift.
Glue Crawler	Découverte automatique des schémas et mise à jour du catalogue.
Glue ETL Job	Exécution de scripts PySpark pour transformer les données (CSV → Parquet).
Parquet	Format de stockage optimisé pour l'analytique Big Data.

Après toutes ces étapes nous sommes allés sur amazon redshift

The screenshot shows the AWS search interface. The search bar at the top contains the query 'redshift Serverless'. Below the search bar, there's a sidebar with a 'Services' section containing links to 'Fonctionnalités (15)', 'Ressources (7)', 'Publications de blog (3)', 'Documentation (2 075)', 'Articles de connaissances (107)', 'Tutoriels (2)', and 'Marketplace (863)'. The main content area is titled 'Services' and shows a list of services. The 'Amazon Redshift' service card is highlighted with a red arrow and a dashed border. It has a purple icon, the name 'Amazon Redshift', and the description 'Entreposage de données rapide, simple et rentable'. Other visible cards include 'AWS Glue DataBrew' and 'Amazon SageMaker'. At the bottom of the main content area, there's a 'Fonctionnalités' section with a 'Redshift Serverless' card.

The screenshot shows the Amazon Redshift dashboard. The top navigation bar includes the account name 'diloma10 (3551-4218-5625)' and the region 'Europe (Paris)'. The main header says 'Amazon Redshift sans serveur' and 'Tableau de bord sans serveur'. On the right, there's a large orange button labeled 'Créer un groupe de travail'. The dashboard features a section titled 'Présentation de l'espace de noms' with a table showing metrics like 'Nombre total d'instantanés' (0), 'Unités de partage des données dans mon compte' (2), 'Unités de partage des données nécessitant une autorisation' (0), 'Unités de partage des données d'autres comptes' (0), and 'Unités de partage des données nécessitant une association' (1). There are also filters for 'Filtrer l'espace de noms' and 'All namespaces'.

Après la création du groupe il faut cliquer sur le nom dans l'espace de groupes

The screenshot shows the AWS Redshift console interface. At the top, there's a banner with instructions about registering Redshift table definitions to AWS Glue Data Catalog. Below the banner, the breadcrumb navigation shows: Amazon Redshift sans serveur > Configuration d'espace de noms > olistspace. The main content area displays the 'olistspace' namespace details under the 'Informations générales' tab. The 'Actions' dropdown menu is open, and the 'Données de requête' button is highlighted with a red circle and arrow. The bottom of the screen shows standard AWS navigation links like CloudShell, Commentaires, and Console de l'application mobile.

On clique sur données de request ou request editor pour pouvoir ajouter des scripts , importer des tables et faire les transformations

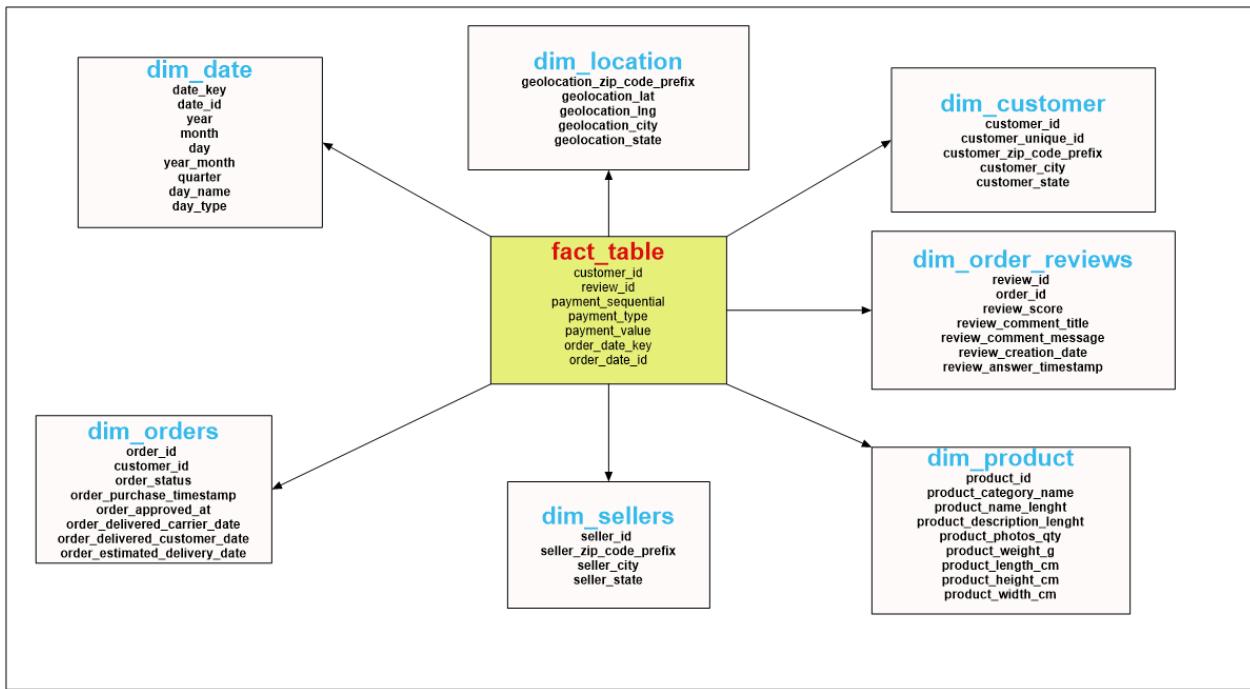
Dans cette partie nous avons fait de nombreuses transformations sur notre base de données

Avant de transformer nous avons importé les données de S3 vers redshift . Les codes pour cette tache se trouvent dans les annexes

Apres importation nous avons transformé les données que nous avons mis dans un dossier silver . Less codee pour ces transformations se trouvent dans les annexes

Nous sommes ensuite passés au schema de notre data warehouse

Apres cela nous avons crée un dossier dw pour stocker nos tables de dimension et de faits

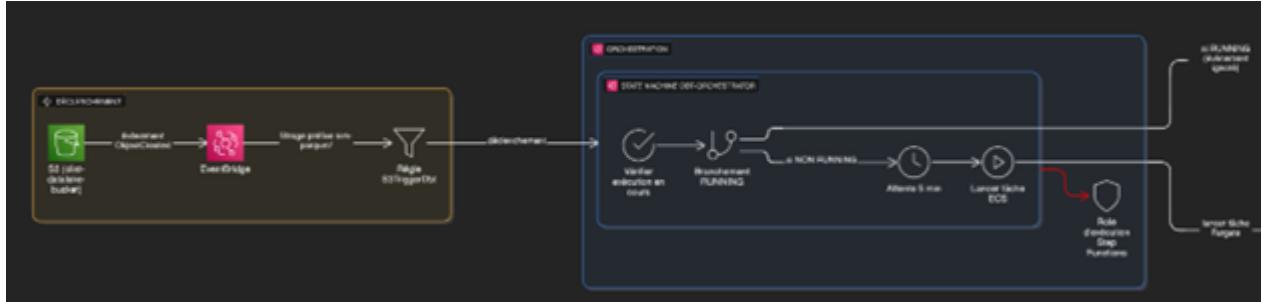
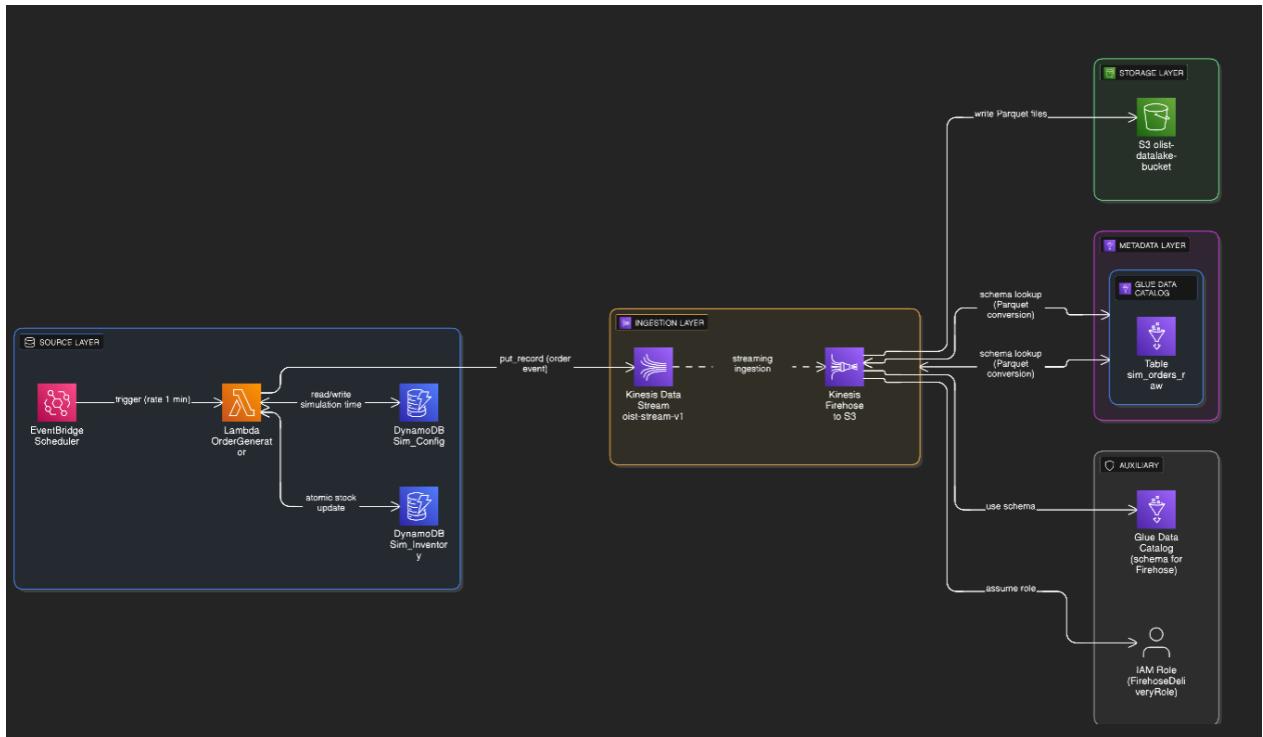


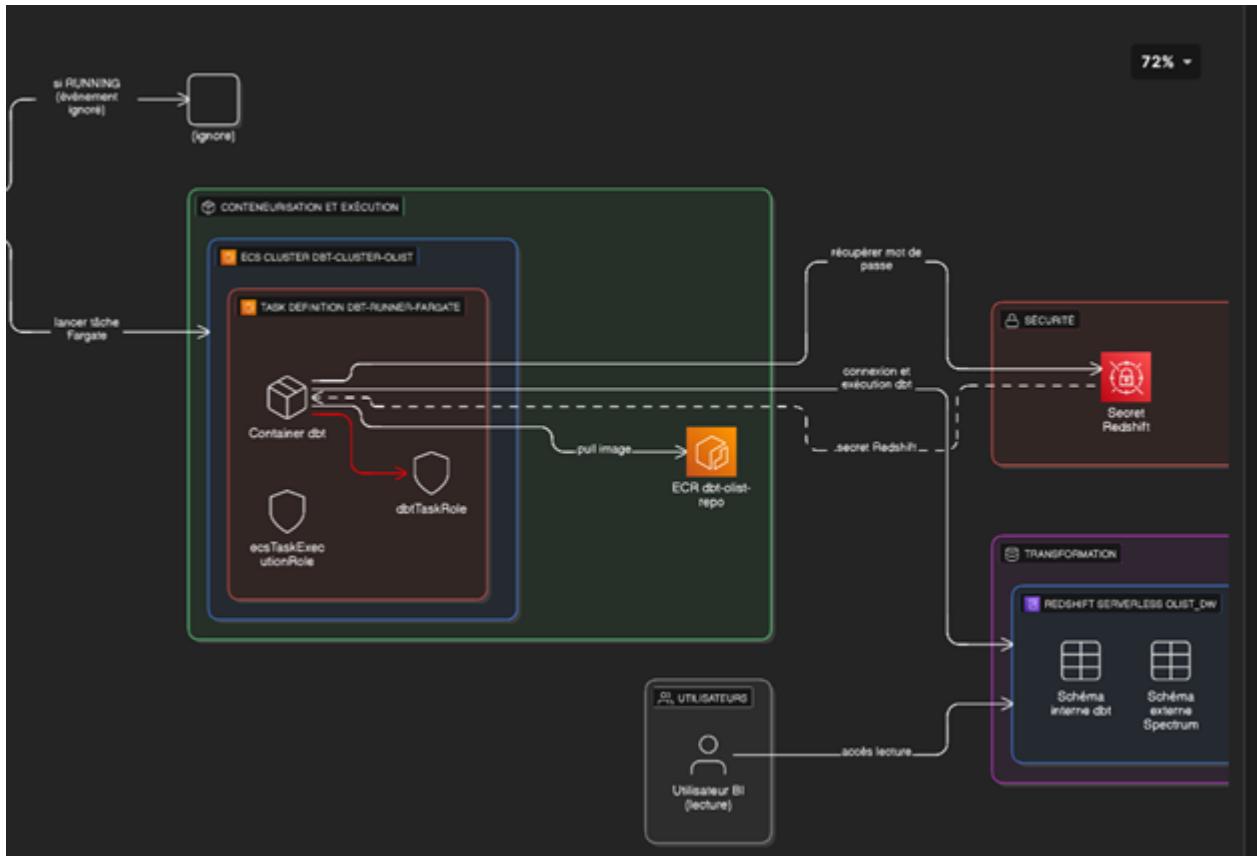
Les codes pour générer ces tables se trouvent dans les annexes

Nous avons utilisé le modèle étoile . Pourquoi le modèle étoile ?

Nous avons utilisé le **modèle en étoile** car il permet une **modélisation simple et lisible**, optimise les **performances des requêtes analytiques**, facilite les **jointures entre tables de faits et dimensions**, et est particulièrement adapté aux **outils de BI** comme Power BI pour l'analyse et la visualisation des données.

LA PARTIE TEMPS REELLE DU PROJET





1- Problématique du temps réel

Dans l'univers de la donnée, le terme "temps réel" peut recouvrir deux réalités distinctes que nous avons dû arbitrer :

- ⊕ **Le Real-Time "Pur" (Streaming)** : Le traitement se fait message par message avec une latence de l'ordre de la milliseconde. C'est l'idéal pour la détection de fraude immédiate, mais cela nécessite des ressources de calcul actives 24h/24 (coûteuses) et une complexité de gestion des transactions très élevée.
- ⊕ **Le Near Real-Time (Micro-batch)** : Les données sont collectées en continu mais traitées par petits groupes (toutes les quelques minutes). Cette approche offre un excellent compromis entre fraîcheur des données et efficacité analytique.

Pour ce projet, nous avons opté pour une approche **Near Real-Time** basée sur des déclenchements planifiés et des buffers de stockage. Ce choix est justifié par deux facteurs majeurs :

1. **Contraintes Financières** : L'utilisation de services "Serverless" (Lambda, Firehose, Redshift Serverless) permet de ne payer qu'à l'usage. Une

infrastructure streaming "pure" aurait engendré des coûts fixes importants, peu justifiables pour un prototype de simulation.

2. **Temps de Développement :** L'absence d'une API de production réelle nous a poussés à créer notre propre générateur. L'approche par batch permet d'orchestrer plus facilement la transformation complexe (via dbt) sans avoir à gérer des états de flux complexes.

2- Le moteur de simulation et la gestion des stocks:

Pour comprendre cette partie, imaginez que nous ne voulons pas simplement lire un fichier Excel, mais recréer un site e-commerce qui "vit" en temps réel. Pour cela, nous utilisons deux services piliers : AWS Lambda (le cerveau) et Amazon DynamoDB (la mémoire).

2.1 Amazon DynamoDB

DynamoDB est une base de données NoSQL orientée clé-valeur, conçue pour les charges transactionnelles à grande échelle. Contrairement à une base relationnelle classique (comme MySQL ou PostgreSQL), elle ne repose ni sur des jointures complexes ni sur des scans complets de tables. Chaque accès se fait via une clé primaire, ce qui lui permet de garantir une latence faible et constante, même lorsque le nombre de requêtes ou de clients augmente fortement. C'est précisément ce type de comportement que l'on attend d'un système opérationnel temps réel.

Dans notre projet, DynamoDB a été choisie pour jouer le rôle de socle transactionnel, séparé du monde analytique (S3, Redshift). Ce choix est cohérent avec les architectures e-commerce modernes, où les bases opérationnelles doivent rester rapides, fiables et simples, tandis que les traitements analytiques sont déportés vers un Data Warehouse.

Nous l'avons retenue pour deux raisons critiques.

Gestion de la configuration de la simulation (Sim_Config)

Cette table stocke l'état global de la simulation, notamment :

- simulated_time, qui représente l'heure courante dans notre monde virtuel,
- speed_factor, qui contrôle l'accélération du temps.

Cette information doit être persistante et immédiatement accessible. Si la fonction Lambda est arrêtée, redéployée ou temporairement désactivée, la simulation doit reprendre exactement là où elle s'était arrêtée. DynamoDB répond parfaitement à ce besoin grâce à :

- des lectures et écritures rapides sur clé primaire,
- une haute disponibilité native,

- l'absence de gestion d'infrastructure (pas de serveur, pas de maintenance).

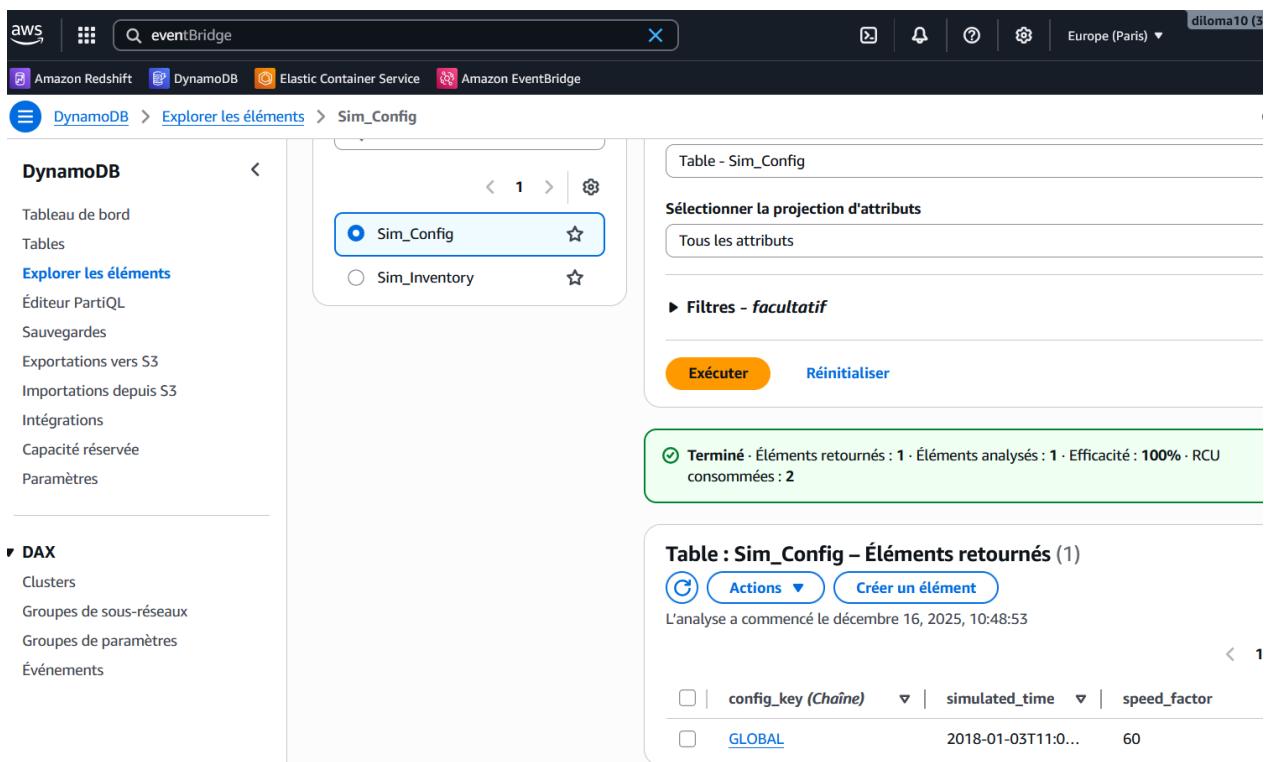
DynamoDB devient ainsi la mémoire fiable de notre horloge virtuelle.

Gestion de l'inventaire (Sim_Inventory)

Cette table contient les produits Olist et leur niveau de stock. Pour rester réaliste, une commande ne peut être validée que si le stock du produit est strictement supérieur à zéro. DynamoDB permet ici d'utiliser des mises à jour conditionnelles atomiques :

si deux commandes arrivent quasi simultanément pour le dernier article disponible, DynamoDB garantit qu'une seule décrémentation du stock réussira.

Ce mécanisme assure une cohérence transactionnelle forte, sans avoir recours à des verrous applicatifs ou à une logique complexe côté Lambda. C'est un point essentiel pour simuler un comportement e-commerce crédible sous concurrence.



The screenshot shows the AWS DynamoDB console interface. The left sidebar navigation includes 'Tableau de bord', 'Tables', 'Explorer les éléments', 'Éditeur PartiQL', 'Sauvegardes', 'Exportations vers S3', 'Importations depuis S3', 'Intégrations', 'Capacité réservée', and 'Paramètres'. Below this is a 'DAX' section with 'Clusters', 'Groupes de sous-réseaux', 'Groupes de paramètres', and 'Événements'. The main content area is titled 'Table - Sim_Config'. It shows two items: 'Sim_Config' (selected) and 'Sim_Inventory'. A button bar at the bottom right includes 'Exécuter' (orange) and 'Réinitialiser'. A green status bar at the bottom indicates 'Terminé - Éléments retournés : 1 · Éléments analysés : 1 · Efficacité : 100% · RCU consommées : 2'. Below this is a table titled 'Table : Sim_Config – Éléments retournés (1)'. It shows one row with columns: config_key (Chain), simulated_time, and speed_factor. The data row is: GLOBAL, 2018-01-03T11:0..., 60. A note says 'L'analyse a commencé le décembre 16, 2025, 10:48:53'.

config_key (Chain)	simulated_time	speed_factor
GLOBAL	2018-01-03T11:0...	60

DynamoDB

Tableau de bord

Tables

Explorer les éléments

Éditeur PartiQL

Sauvegardes

Exportations vers S3

Importations depuis S3

Intégrations

Capacité réservée

Paramètres

DAX

Clusters

Groupes de sous-réseaux

Groupes de paramètres

Événements

Table - Sim_Config

Sélectionner la projection d'attributs

Tous les attributs

Filtres - facultatif

Exécuter Réinitialiser

Terminé · Éléments retournés : 1 · Éléments analysés : 1 · Efficacité : 100% · RCU consommées : 2

Table : Sim_Config – Éléments retournés (1)

Actions Crée un élément

L'analyse a commencé le décembre 16, 2025, 10:48:53

config_key (Chainé)	simulated_time	speed_factor
GLOBAL	2018-01-03T11:0...	60

Avantages du choix DynamoDB

- Très forte scalabilité et latence constante.
- Mises à jour atomiques et conditionnelles simples à mettre en œuvre.
- Service entièrement managé, sans administration de serveurs.
- Intégration native avec Lambda, Kinesis, Glue et Redshift.

Limites et contreparties

- Modélisation des données plus rigide (tout doit être pensé autour des clés).
- Pas de jointures ni de requêtes analytiques complexes.
- Peu adaptée aux besoins de reporting ou d'agrégation massive.

Ces limites sont toutefois assumées : DynamoDB n'est pas utilisée pour l'analyse, mais uniquement pour le temps réel transactionnel.

Si l'on souhaite exploiter les données de DynamoDB pour de l'analytique, plusieurs solutions existent, chacune avec ses avantages et ses limites. Une approche

courante consiste à activer DynamoDB Streams, qui capture toutes les modifications apportées aux tables en temps quasi réel. Ces flux peuvent être consommés par une fonction Lambda ou un Kinesis Data Stream pour transférer les données vers un stockage analytique, comme S3 ou Redshift. Cette méthode permet de disposer de données très récentes sans impacter les performances transactionnelles de la table principale et s'intègre naturellement dans une architecture event-driven. Cependant, elle ajoute de la complexité au pipeline et nécessite de gérer le débit des shards, ainsi que la transformation des données (format Parquet, enrichissement, normalisation) avant ingestion.

On peut naturellement se demander si DynamoDB est vraiment le meilleur choix pour une entreprise comme Olist, surtout si l'on souhaite suivre en temps réel instantané toutes les actions des clients sur le site. En pratique, même avec sa latence extrêmement faible et ses mises à jour atomiques, DynamoDB ne permet pas de traiter chaque événement à la milliseconde pour l'ensemble des utilisateurs sans saturer le système ou générer des coûts prohibitifs. C'est pourquoi, dans les architectures e-commerce modernes, DynamoDB est utilisé pour le plan transactionnel, là où la cohérence et la rapidité sont critiques — validation des commandes, mise à jour des stocks, notifications clients — tandis que l'analytique se fait en quasi temps réel via des exports ou des streams vers S3 et Redshift. Cette séparation assure performance, fiabilité et scalabilité, tout en fournissant des données fraîches pour le reporting et l'analyse, sans tenter de suivre chaque clic ou mouvement client en direct.

Pour pallier cette limitation et obtenir un suivi plus fin des événements utilisateurs, une entreprise comme Olist pourrait compléter DynamoDB avec un système de streaming dédié, tel qu'Apache Kafka, Redis Streams, ou même Amazon QLDB. Ces solutions ne remplacent pas DynamoDB pour les transactions critiques, mais permettent de capturer l'ensemble des événements utilisateur à très haute fréquence, offrant la possibilité de traitements temps réel plus fins ou d'analyses ultra-réactives. En combinant DynamoDB pour la cohérence transactionnelle et un système de streaming pour le suivi instantané, Olist pourrait disposer d'une architecture à la fois fiable, scalable et capable de restituer un quasi temps réel sur l'ensemble des comportements clients, ce que DynamoDB seule ne peut garantir.

En parallèle, nous avons mis en place la table Sim_Inventory pour gérer les stocks de produits. Plutôt que de reprendre l'intégralité du dataset Olist, nous avons sélectionné un échantillon pertinent pour démontrer la faisabilité du concept. Cette table contient les identifiants produits (product_id), ainsi que leur niveau de stock actuel (stock_level), la catégorie et le prix. Nous avons utilisé un script Python, exécuté via l'AWS CLI, pour peupler initialement cette table. Ce script prépare le terrain pour que notre générateur de commandes puisse interagir avec des données concrètes.

2.2 La génération de commandes via AWS Lambda

AWS Lambda est un service serverless qui permet d'exécuter du code à la demande, sans gestion de serveurs ni d'infrastructure. Dans notre projet, Lambda joue un rôle central : elle simule le comportement des clients du site Olist en générant des événements de commande de manière contrôlée et réaliste.

La fonction est déclenchée chaque minute par un planificateur Amazon EventBridge. À chaque exécution, elle commence par lire l'état de la simulation dans DynamoDB, en particulier l'horloge virtuelle (simulated_time) et le facteur d'accélération du temps (speed_factor). Sur cette base, elle génère un nombre aléatoire de commandes (entre 3 et 20), chacune associée à un produit, un client et un timestamp cohérent avec le temps simulé.

Pour chaque commande, la Lambda applique une logique transactionnelle réaliste : elle décrémente le stock du produit correspondant dans DynamoDB à l'aide d'une mise à jour conditionnelle atomique, garantissant qu'aucune vente n'est validée si le stock est insuffisant. Une fois les commandes générées, l'horloge virtuelle est avancée et persistée, ce qui permet à la simulation de reprendre correctement lors de l'exécution suivante, même en cas d'arrêt ou de redéploiement de la fonction.

Afin de limiter les coûts et la latence liés aux accès DynamoDB, un cache en mémoire est utilisé au sein de la Lambda. La liste des identifiants de produits est chargée une seule fois et conservée en mémoire entre les invocations, évitant ainsi de relancer un scan complet de la table à chaque exécution. Ce mécanisme illustre une optimisation couramment utilisée en environnement serverless pour réduire le nombre de requêtes, améliorer les performances et maîtriser les coûts.

The screenshot shows the AWS Lambda function editor interface. On the left, the file tree displays a project named 'OLIST-ORDER-GENERATOR' containing a single file 'lambda_function.py'. On the right, the code editor shows the following Python script:

```

11 dynamodb = boto3.resource('dynamodb')
12 kinesis = boto3.client('kinesis')
13 fake = Faker('pt_BR') # On utilise la locale Brésilien pour Olist !
14
15 # Constants
16 STREAM_NAME = 'olist-stream-v1'
17 TABLE_INVENTORY = 'Sim_Inventory'
18 TABLE_CONFIG = 'Sim_Config'
19
20 def get_simulation_state():
21     """Récupère l'heure virtuelle actuelle"""
22     table = dynamodb.Table(TABLE_CONFIG)
23     resp = table.get_item(Key={'config_key': 'GLOBAL'})
24     # Si pas de config, on met une valeur par défaut
25     if 'Item' not in resp:
26         return datetime(2018, 1, 1), 60
27
28     item = resp['Item']
29     sim_time = datetime.fromisoformat(item['simulated_time'])
30     speed = int(item['speed_factor'])
31     return sim_time, speed
32
33 def update_simulation_time(current_time, minutes_to_add):

```

3. Le Pipeline d'Ingestion

Une fois la commande générée par la fonction Lambda, celle-ci doit être transportée de manière fiable depuis le monde transactionnel vers le monde analytique, sans impacter les performances du système opérationnel. C'est précisément le rôle de la famille Amazon Kinesis, qui sert de colonne vertébrale à notre pipeline d'ingestion temps réel batché.

3.1 Kinesis Data Streams

Amazon Kinesis Data Streams peut être vu comme un bus d'événements durable et hautement disponible, comparable à un tapis roulant très rapide sur lequel la Lambda dépose chaque commande. Une fois publiée dans le stream, la donnée est conservée pendant une durée configurable (24 heures dans notre cas), indépendamment des consommateurs.

Ce choix apporte deux avantages majeurs. D'une part, il garantit la résilience du pipeline : si un composant en aval (Firehose, S3, traitement analytique) est temporairement indisponible, les événements ne sont pas perdus et peuvent être relus ultérieurement. D'autre part, il permet de découpler les producteurs des consommateurs. Plusieurs systèmes peuvent consommer le même flux en parallèle, chacun à son rythme, sans interférer les uns avec les autres.

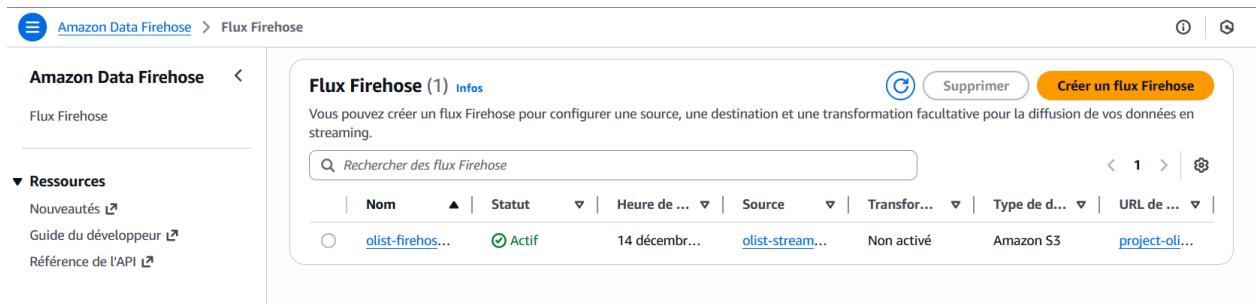
Dans un contexte réel, on pourrait ainsi imaginer un consommateur chargé d'envoyer des emails de confirmation aux clients, un autre alimentant des tableaux de bord temps réel, et un troisième — celui de notre projet — dédié à l'ingestion vers le Data Warehouse. Kinesis Data Streams joue donc le rôle de tampon de sécurité et de point de convergence pour l'ensemble des événements e-commerce.

3.2 Kinesis Data Firehose

Kinesis Data Firehose est le service chargé de consommer les événements depuis Kinesis Data Streams et de les déposer dans le Data Lake S3, en les rendant immédiatement exploitables pour l'analytique. Contrairement à un simple mécanisme de copie, Firehose effectue plusieurs opérations clés qui optimisent fortement la chaîne analytique.

Tout d'abord, Firehose applique un buffering contrôlé : il attend d'avoir accumulé suffisamment de données — soit 300 secondes, soit 150 Mo pour notre choix — avant d'écrire un fichier dans S3. Cette stratégie évite la création de milliers de petits fichiers, qui dégraderaient fortement les performances de lecture dans des moteurs analytiques comme Redshift Spectrum ou Athena.

Ensuite, Firehose réalise la conversion de format, en transformant les événements JSON, verbeux et peu optimisés pour l'analyse, en fichiers Parquet, un format colonne compressé, spécialement conçu pour les charges analytiques. Cette conversion repose sur AWS Glue Data Catalog, qui décrit précisément le schéma des données Olist. Grâce à ce catalogue, Firehose sait comment interpréter les champs JSON et les convertir dans une structure analytique cohérente.



The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with navigation links: 'Amazon Data Firehose' (selected), 'Flux Firehose', 'Ressources' (expanded), 'Nouveautés', 'Guide du développeur', and 'Référence de l'API'. The main area has a header 'Flux Firehose (1) Infos' with a 'Supprimer' button and a 'Créer un flux Firehose' button. Below the header is a message: 'Vous pouvez créer un flux Firehose pour configurer une source, une destination et une transformation facultative pour la diffusion de vos données en streaming.' A search bar says 'Rechercher des flux Firehose'. A table lists the existing flux: 'olist-firehose' (Status: Actif, Last updated: 14 décembre...). The table has columns: Nom, Statut, Heure de..., Source, Transform..., Type de d..., URL de... .

4. Stratégie de Data Warehousing : Redshift Serverless et Spectrum

Une fois nos fichiers Parquet déposés dans le Data Lake (S3) par Firehose, ils ne sont pas encore intégrés dans une base de données. Pour les exploiter, nous avons choisi Amazon Redshift Serverless, une solution d'entrepôt de données qui s'adapte automatiquement à la charge de travail sans avoir à gérer de clusters complexes.

4.1. L'approche Hybride : Tables Externes vs Tables Internes

Notre architecture repose sur un concept puissant : la séparation du stockage et du calcul.

- ✚ Les Tables Externes : Nous ne copions pas les données brutes dans Redshift. Nous utilisons Redshift Spectrum pour lire directement les fichiers Parquet stockés sur S3. C'est notre couche "Bronze". Cela réduit les coûts de stockage drastiquement.
- ✚ Les Tables Internes : Pour les données transformées et agrégées (Silver et Gold), nous créons de véritables tables dans Redshift. Cela garantit des performances maximales pour les requêtes des utilisateurs finaux.

5. Transformation et Modélisation avec dbt

Pour transformer la donnée brute (fichiers Parquet) en indicateurs métiers, nous utilisons dbt (data build tool). C'est l'outil standard du marché qui nous permet d'appliquer les bonnes pratiques du génie logiciel (versioning, tests, environnements) au monde de la donnée (SQL).

5.1. Du développement Local à la Production

Le développement des modèles dbt a commencé en local. Sur nos machines, nous avons configuré un profil de connexion (profiles.yml) pointant vers notre Redshift workspace. C'est indispensable pour tester les requêtes, débugger les erreurs SQL et visualiser le graphe de dépendance (Lineage) avant de déployer quoi que ce soit.

Une fois le code validé localement, nous ne pouvons pas laisser nos ordinateurs faire tourner la production. Nous devons conteneuriser le projet. Nous avons donc créé une image Docker contenant tout le projet dbt et ses dépendances. Cette image est stockée sur Amazon ECR (Elastic Container Registry), prête à être déployée.

The screenshot shows the AWS ECR console interface. On the left, there's a sidebar with navigation links for 'Amazon Elastic Container Registry', 'Private registry' (selected), 'Public registry', and links to 'ECR public gallery', 'Amazon ECS', and 'Amazon EKS'. The main content area has a blue header bar with the message: 'La signature générée est désormais disponible. Signez automatiquement vos images de conteneurs lors de leur envoi pour en vérifier l'authenticité et sécuriser votre chaîne d'approvisionnement. Configuration de la signature d'images'. Below this, a table lists a single private repository:

Nom du référentiel	URI	Créé à	Immuabilité des identifications	Type de chiffrement
olist_dw_dbt_run_sim	355142185625.dkr.ecr.eu-west-3.amazonaws.com/olist_dw_dbt_run_sim	16 décembre 2025, 00:11:41 (UTC-00)	Mutable	AES-256

5.2. Le Modèle Silver : nettoyage et aplatissement

La première étape de la transformation consiste à nettoyer la donnée brute issue du JSON converti en Parquet. La structure initiale est complexe : chaque commande contient une liste d'articles imbriqués. Pour rendre ces informations exploitables, nous devons les aplatis. Cela signifie que l'on extrait les éléments imbriqués afin qu'une ligne corresponde à un produit vendu, plutôt qu'à une commande entière.

Ensuite, comme les données arrivent progressivement au fil du temps, il est essentiel de mettre en place une gestion incrémentale. Au lieu de retraiter tout l'historique à chaque exécution, le modèle ne prend en compte que les nouvelles commandes arrivées depuis la dernière mise à jour. Cette logique permet de réduire considérablement le temps de calcul et les coûts, tout en maintenant la base de données à jour.

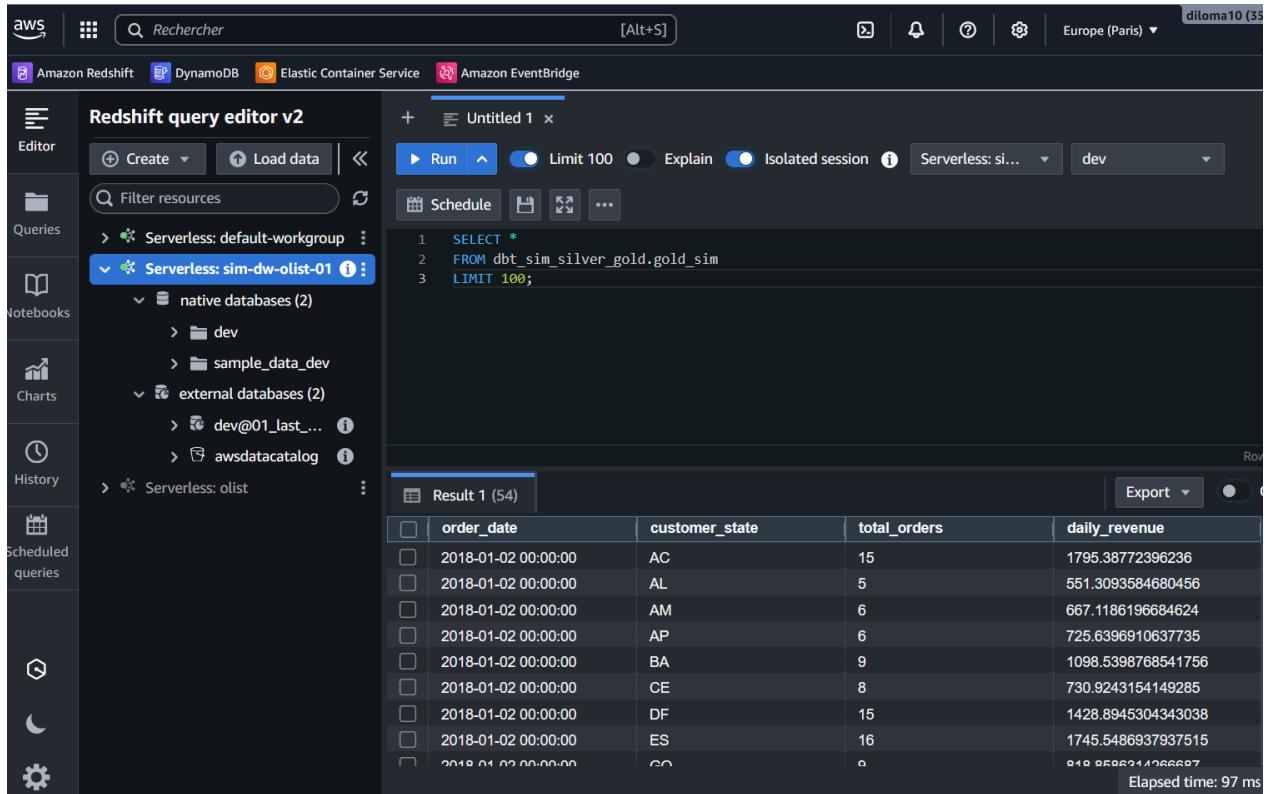
5.3. Le Modèle Gold : Agrégation Métier

Après avoir nettoyé et structuré les données, le modèle Gold constitue la dernière étape de la chaîne de transformation. Son rôle est de produire des indicateurs métiers

directement exploitables pour le pilotage et la visualisation. Dans ce cas précis, il s'agit de calculer le chiffre d'affaires quotidien par état brésilien.

Chaque journée est ainsi associée à deux informations essentielles : le nombre total de commandes validées et le revenu généré. Ces agrégations permettent de suivre l'évolution des ventes dans le temps et d'identifier les différences de performance entre régions.

Le modèle est conçu pour fonctionner de manière incrémentale. Plutôt que de recalculer l'ensemble de l'historique à chaque exécution, il ne met à jour que les jours concernés par l'arrivée de nouvelles données. Cette optimisation réduit considérablement le temps de traitement et les coûts, tout en garantissant que les indicateurs restent fiables et à jour.



The screenshot shows the AWS Redshift query editor interface. On the left, there's a sidebar with various navigation options like Editor, Queries, Notebooks, Charts, History, Scheduled queries, and a gear icon. The main area has a title bar "Untitled 1" with a "Run" button, "Limit 100" checkbox, "Explain" checkbox, "Isolated session" checkbox, and a dropdown for "Serverless: si... dev". Below this is a "Schedule" section with a calendar icon and three dots. The code editor contains the following SQL query:

```
1  SELECT *
2  FROM dbt_sim_silver_gold.gold_sim
3  LIMIT 100;
```

Below the code editor is a table titled "Result 1 (54)" showing the query results:

	order_date	customer_state	total_orders	daily_revenue
□	2018-01-02 00:00:00	AC	15	1795.38772396236
□	2018-01-02 00:00:00	AL	5	551.3093584680456
□	2018-01-02 00:00:00	AM	6	667.1186196684624
□	2018-01-02 00:00:00	AP	6	725.6396910637735
□	2018-01-02 00:00:00	BA	9	1098.5398768541756
□	2018-01-02 00:00:00	CE	8	730.9243154149285
□	2018-01-02 00:00:00	DF	15	1428.8945304343038
□	2018-01-02 00:00:00	ES	16	1745.5486937937515
□	2018-01-02 00:00:00	GO	9	919.8586211066697

At the bottom right of the results table, it says "Elapsed time: 97 ms".

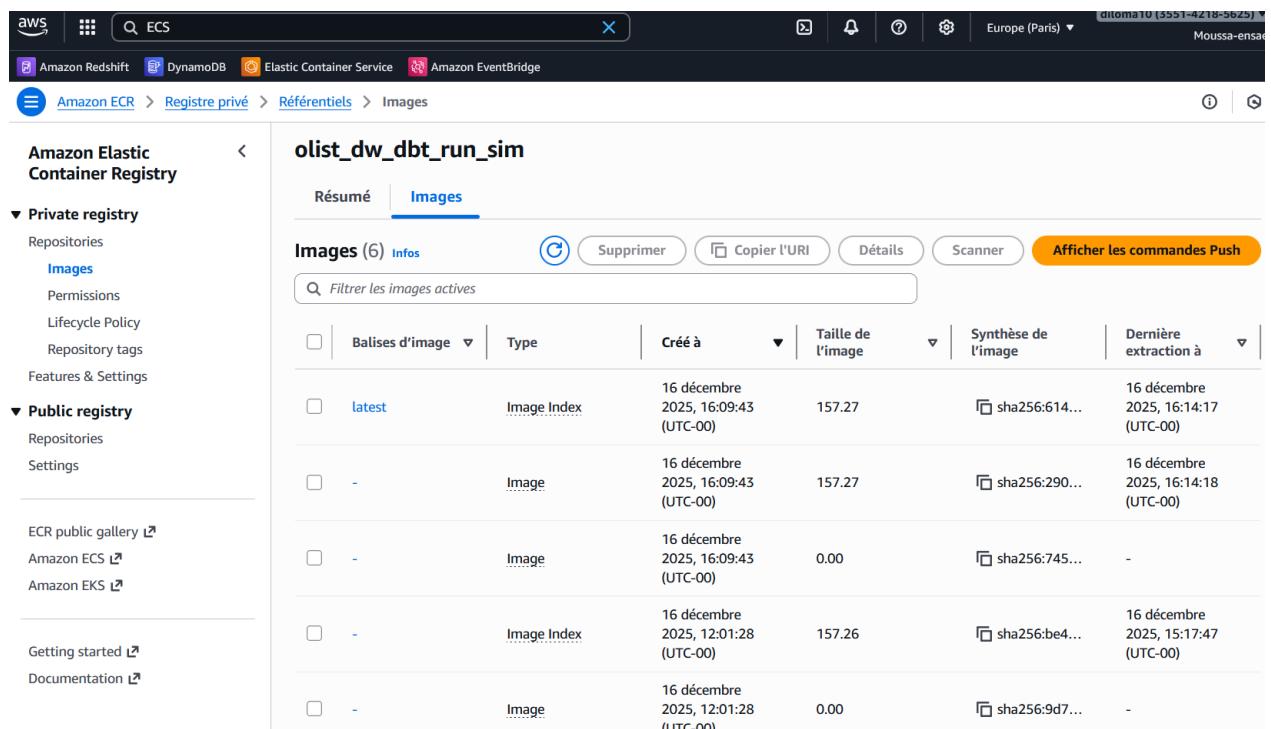
6. Automatisation de la transformation

Le développement des modèles de transformation de données s'effectue initialement sur un poste de travail local, ce qui permet une itération rapide et un débogage immédiat. Cependant, pour garantir la pérennité et la fiabilité du pipeline de données, il est impératif de migrer cette logique vers une infrastructure Cloud robuste et automatisée. Cette section détaille comment nous sommes passés d'une exécution manuelle à un environnement de production autonome et sécurisé sur AWS.

6.1. La Conteneurisation avec Docker et Amazon ECR

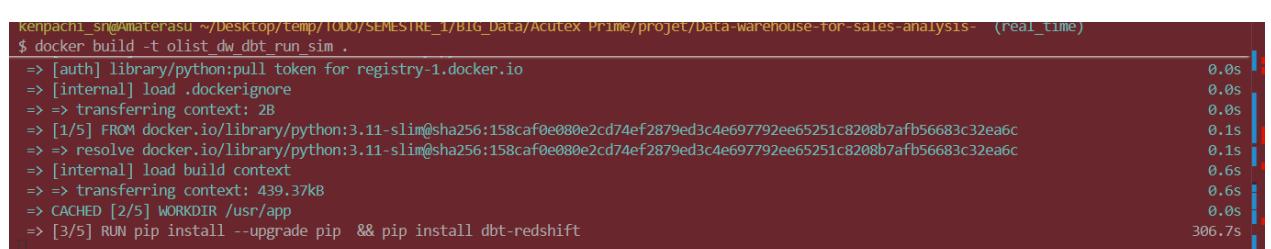
La première étape de cette industrialisation consiste à figer l'environnement technique. En local, dbt dépend de versions spécifiques de Python, de bibliothèques et de configurations systèmes qui peuvent varier d'une machine à l'autre. Pour résoudre ce problème classique de portabilité, nous avons encapsulé l'ensemble du projet dbt, incluant le code SQL et les dépendances logicielles, dans une image Docker.

Cette approche garantit que le code s'exécutera exactement de la même manière en production que sur la machine de développement. Une fois cette image construite, elle est stockée dans Amazon Elastic Container Registry (ECR). Ce service agit comme un entrepôt privé et sécurisé pour nos images Docker, rendant notre application accessible aux services de calcul d'AWS sans l'exposer publiquement.



The screenshot shows the AWS ECR console interface. On the left, there's a sidebar with navigation links for Amazon Redshift, DynamoDB, Elastic Container Service, and Amazon EventBridge, followed by 'Amazon ECR > Registre privé > Référentiels > Images'. The main area is titled 'olist_dw_dbt_run_sim' and has tabs for 'Résumé' and 'Images'. The 'Images' tab is selected, showing a table with six rows. The columns are 'Balises d'image', 'Type', 'Créé à', 'Taille de l'image', 'Synthèse de l'image', and 'Dernière extraction à'. The rows are:

Balises d'image	Type	Créé à	Taille de l'image	Synthèse de l'image	Dernière extraction à
latest	Image Index	16 décembre 2025, 16:09:43 (UTC-00)	157.27	sha256:614...	16 décembre 2025, 16:14:17 (UTC-00)
-	Image	16 décembre 2025, 16:09:43 (UTC-00)	157.27	sha256:290...	16 décembre 2025, 16:14:18 (UTC-00)
-	Image	16 décembre 2025, 16:09:43 (UTC-00)	0.00	sha256:745...	-
-	Image Index	16 décembre 2025, 12:01:28 (UTC-00)	157.26	sha256:be4...	16 décembre 2025, 15:17:47 (UTC-00)
-	Image	16 décembre 2025, 12:01:28 (UTC-00)	0.00	sha256:9d7...	-



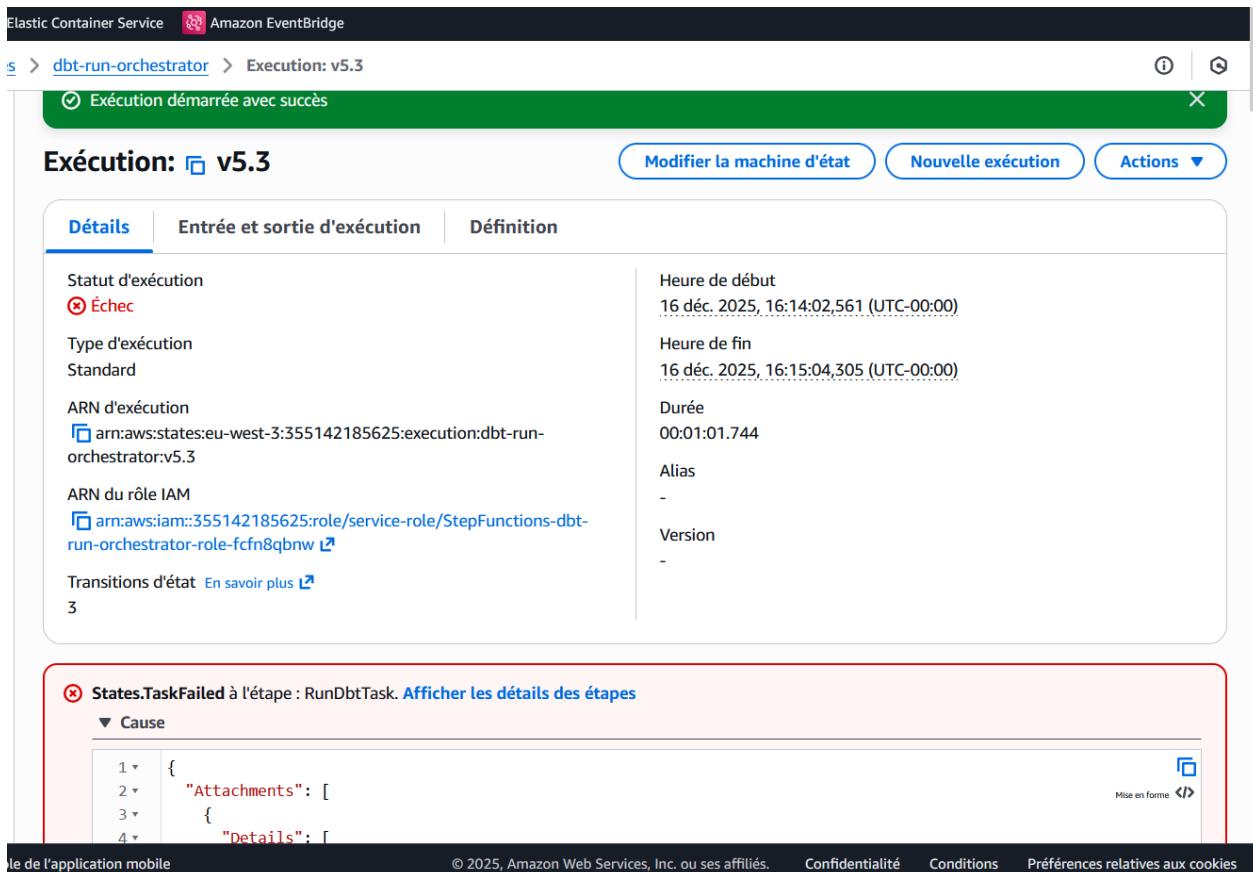
```
kenpacni-sg@kenpacni-sg:~/Desktop/temp/SEMESTRE_1/BIG_Data/AcuteX_Projeto/projet/Data-warehouse-for-sales-analysis-(real_time)$ docker build -t olist_dw_dbt_run_sim .
=> [auth] library/python:pull token for registry-1.docker.io
=> [internal] load .dockerrignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/python:3.11-slim@sha256:159caf0e080e2cd74ef2879ed3c4e697792ee65251c8208b7afb56683c32ea6c
=> => resolve docker.io/library/python:3.11-slim@sha256:158caf0e080e2cd74ef2879ed3c4e697792ee65251c8208b7afb56683c32ea6c
=> [internal] load build context
=> => transferring context: 439.37kB
=> CACHED [2/5] WORKDIR /usr/app
=> [3/5] RUN pip install --upgrade pip && pip install dbt-redshift
0.0s
0.0s
0.0s
0.1s
0.1s
0.6s
0.6s
0.0s
306.7s
```

6.2. Déploiement Serverless avec AWS ECS et Fargate

Une fois notre image Docker sécurisée dans le registre ECR, nous devons trouver un moyen de l'exécuter de manière autonome. Plutôt que de gérer manuellement des serveurs EC2 qui auraient nécessité des mises à jour de sécurité et une surveillance

constante, nous avons opté pour Amazon ECS (Elastic Container Service) avec le mode de lancement Fargate. Fargate abstrait totalement la couche serveur : nous définissons simplement les ressources nécessaires (CPU et RAM) et AWS se charge de trouver la capacité de calcul au moment précis de l'exécution.

Le cœur de cette configuration repose sur ce qu'on appelle une Définition de Tâche (Task Definition). C'est le plan technique de notre application. Dans cette définition, j'ai spécifié l'URI de l'image stockée sur ECR ainsi que les contraintes de ressources. Pour un job dbt standard sur notre volume de données, une configuration légère de 0.5 vCPU et 1 Go de mémoire s'est avérée suffisante, ce qui contribue à la maîtrise des coûts. C'est également dans cette définition que nous configurons la journalisation. En activant le driver de logs awslogs, nous demandons au conteneur de rediriger toutes les sorties standards (ce qui s'affiche habituellement dans le terminal lors d'un dbt run) directement vers Amazon CloudWatch. Cela nous permet de surveiller à distance le succès des transformations ou de lire les messages d'erreur sans jamais avoir à nous connecter à une machine.



The screenshot shows the AWS CloudWatch Metrics interface. At the top, there are navigation links for 'Elastic Container Service' and 'Amazon EventBridge'. Below that, a breadcrumb trail shows the path: 'dbt-run-orchestrator > Execution: v5.3'. A green banner at the top indicates a successful execution. The main area is titled 'Exécution: v5.3'. It contains three tabs: 'Détails', 'Entrée et sortie d'exécution', and 'Définition'. The 'Détails' tab is selected. It displays various execution details:

Statut d'exécution	Heure de début
Échec	16 déc. 2025, 16:14:02,561 (UTC-00:00)
Type d'exécution	Heure de fin
Standard	16 déc. 2025, 16:15:04,305 (UTC-00:00)
ARN d'exécution	Durée
arn:aws:states:eu-west-3:355142185625:execution:dbt-run-orchestrator:v5.3	00:01:01.744
ARN du rôle IAM	Alias
arn:aws:iam::355142185625:role/service-role/StepFunctions-dbt-run-orchestrator-role-fcfn8qbnw ↗	-
Transitions d'état	Version
En savoir plus ↗	-
3	

Below this, a red-bordered box highlights an error message: 'States.TaskFailed à l'étape : RunDbtTask. [Afficher les détails des étapes](#)'. A 'Cause' section is expanded, showing a JSON-like stack trace:

```
1 { "Attachments": [ 2 { 3 { "Details": [ 4 {
```

At the bottom of the page, there are footer links: 'de l'application mobile', '© 2025, Amazon Web Services, Inc. ou ses affiliés.', 'Confidentialité', 'Conditions', and 'Préférences relatives aux cookies'.

AWS Rechercher [Alt+S] Europe (Paris) diloma10 (3551-4218-5625) Moussa-ensae

Amazon Redshift DynamoDB Elastic Container Service Amazon EventBridge

Amazon Elastic Container Service Définitions de tâches run_container_olist Révision 2 Conteneurs

Amazon Elastic Container Service run_container_olist:2 Dernière mise à jour à 18 décembre 2025, 09:58 (UTC) Déployer Actions Crée une révision

Présentation Infos

ARN arn:aws:ecs:eu-west-3:355142185625:task-definition/run_container_olist:2	Statut ACTIVE	Heure de création 17 décembre 2025, 11:37 (UTC)	Environnement de l'application Fargate
Rôle de la tâche dbt-ecs-task-role-olist	Rôle d'exécution de tâche dbt-ecs-task-role-olist	Système d'exploitation/Architecture Linux/X86_64	Mode réseau awsvpc
Injection de pannes Désactivé			

Conteneurs JSON Placement de tâche Volumes (0) Requiert des attributs Balises

Taille de la tâche

CPU de tâche unités 1 024 (1 vCPU) Mémoire de tâche 3 072 Mio (3 Gio)

Allocation maximale de CPU de tâche pour les conteneurs

Mémoire de tâche 3 072 Mio (3 Gio)

Allocation maximale de mémoire de tâche pour la réservation de mémoire de conteneur

Mémoire (Mio)

docker desktop PERSONAL

Ask Gordon BETA Containers Images Volumes Kubernetes Builds Models MCP Toolkit BETA Docker Hub Docker Scout Extensions

Last refresh: 4 minutes ago

Run a new container 355142185625.dkr.ecr.eu-west-3.amazonaws.com/olist_dw_dbt_run_sim:latest

Optional settings Container name teste_version6.4 A random name is generated if you do not provide one.

Ports No ports exposed in this image

Volumes Host path sim-dw-olist-01.3551421856... Container path 5439 +

Environment variables Variable BT_REDSHIFT_USER Value dbt_user - Variable DBT_REDSHIFT_PASSWORD Value Passer123 +

Cancel Run

Engine running RAM 1.24 GB CPU 0.00% Disk: 2.70 GB used (limit 1006.85 GB) Update available

AWS | Rechercher [Alt+S] | Europe

Amazon Redshift | DynamoDB | Elastic Container Service | Amazon EventBridge

CloudWatch > Log management > /ecs/run_container_olist > ecs/dbt-container-olist/44882548a65a44859477a8afeb...

CloudWatch

Favoris et récents

▶ **Signaux d'application (APM)** Nouveau

▼ Infrastructure Monitoring

- Container Insights
- Bases de données Insights
- Lambda Insights
- État des ressources EC2

▼ Journaux

Log Management Nouveau

- Anomalies dans les journaux
- Queue en direct
- Logs Insights Nouveau
- Contributor Insights

▶ Métriques

Événements de journaux

Vous pouvez utiliser la barre de filtre ci-dessous pour rechercher et faire correspondre des termes, de événements de journal. [En savoir plus sur les modèles de filtre](#)

Filtrer les événements : appuyez sur Entrée pour rechercher.

1m 1h Effacer Fuseau horaire UTC Affichage

Message

```
[0m18:32:37 retries: 1
[0m18:32:37 retry_all: False
[0m18:32:37 autocommit: True
[0m18:32:37 access_key_id: None
[0m18:32:37 is_serverless: None
[0m18:32:37 serverless_work_group: None
[0m18:32:37 serverless_acct_id: None
[0m18:32:37 Registered adapter: redshift=1.9.5
[0m18:32:37 Connection test: [[32mOK connection ok[0m
[0m18:32:37 [31m1 check failed:[0m
[0m18:32:37 Error from git --help: Could not find command, ensure it is in the user's PATH:
```

Amazon Redshift DynamoDB Elastic Container Service Amazon EventBridge

CloudWatch > Log management > /ecs/run_container_olist > ecs/dbt-container-olist/ccded294fec0497f87b5586e42...

CloudWatch

- Favoris et récents
- ▶ Signaux d'application (APM)
- ▼ Infrastructure Monitoring
 - Container Insights
 - Bases de données Insights
 - Lambda Insights
 - État des ressources EC2
- ▼ Journaux
 - Log Management** Nouveau
 - Anomalies dans les journaux
 - Queue en direct
 - Logs Insights Nouveau
 - Contributor Insights
- ▶ Métriques
- ▶ Surveillance du réseau

Événements de journaux

Vous pouvez utiliser la barre de filtre ci-dessous pour rechercher et faire correspondre des termes, d'événements de journal. [En savoir plus sur les modèles de filtre](#)

Filtrer les événements : appuyez sur Entrée pour rechercher.

1m 1h Effacer Fuseau horaire UTC Affichage

Message

```
[0m21:21:26 role: None
[0m21:21:26 retries: 1
[0m21:21:26 retry_all: False
[0m21:21:26 autocommit: True
[0m21:21:26 access_key_id: None
[0m21:21:26 is_serverless: None
[0m21:21:26 serverless_work_group: None
[0m21:21:26 serverless_acct_id: None
[0m21:21:26 Registered adapter: redshift=1.9.5
[0m21:21:26 Connection test: [[32mOK connection ok[0m
[0m21:21:26 [32mAll checks passed![0m]
```

Aucun nouvel événement pour le moment *Nouvelle tentative automatique suspendue.* [Reprendre](#)

Amazon Redshift DynamoDB Elastic Container Service Amazon EventBridge

/ecs/run_container_olist

Amazon Redshift

Entreposage de données rapide, simple et rentable

CloudWatch

- Favoris et récents
- ▶ Signaux d'application (APM)
- ▼ Infrastructure Monitoring
 - Container Insights
 - Bases de données Insights
 - Lambda Insights
 - État des ressources EC2
- ▼ Journaux
 - Log Management** Nouveau
 - Anomalies dans les journaux
 - Queue en direct
 - Logs Insights Nouveau
 - Contributor Insights
- ▶ Métriques
- ▶ Surveillance du réseau

Classe de journal	Informations	Filtres de métriques	Protection des données
Standard	ARN arn:aws:logs:eu-west-3:355142185625:log-group:/ecs/run_container_olist*	0	Nombre de données sensibles
	Heure de création Il y a 3 heures	Filtres d'abonnement 0	Index de champs personnalisés
	Conservation Ne jamais arriver à expiration	Règles de Contributor Insights	Configurer
	Octets stockés -	ID de clé KMS	Transformer
		Protection contre la suppression Off	Détection des anomalies

Flux de journaux

Flux de journaux (1) Supprimer Crée un flux de journaux Rechercher tous les flux de journaux

Par défaut, nous chargeons uniquement les flux de journaux les plus récents.

Filtrer les flux de journaux ou essayer la recherche Correspondance exacte Afficher les résultats expirés Informations

Flux de journaux	Heure du dernier événement
ecs/dbt-container-olist/002ef0d5f5b44447a12fd2742a35d3d3	2025-12-16 15:22:54 (UTC)

L'un des aspects les plus délicats de la configuration a été la gestion du réseau. Le conteneur dbt a une double contrainte : il doit être capable de communiquer avec Redshift (situé dans notre réseau privé) pour exécuter les requêtes SQL, mais il a aussi besoin d'un accès à l'internet public pour télécharger les dépendances Python et dbt au démarrage. Nous avons donc dû configurer le VPC (Virtual Private Cloud). La tâche est lancée dans un sous-réseau spécifique et se voit attribuer une adresse IP publique temporaire. Nous avons associé un Groupe de Sécurité (Security Group) strict qui agit comme un pare-feu virtuel. Ce groupe autorise le trafic sortant vers le port 5439 (le port standard de Redshift) tout en bloquant les accès entrants non sollicités, garantissant ainsi l'intégrité de notre pipeline de transformation.

The screenshot shows the AWS Management Console with the EC2 service selected. In the left navigation pane, under 'Instances', 'Instances' is selected. The main content area displays the 'Groupes de sécurité' (Security Groups) page. It shows a table with one row for a security group named 'sg-0a70f9ab7004d77e1'. The table includes columns for Name, ID du groupe de sécurité, Nom du groupe de sécurité, and ID de VPC. Below the table, a detailed view for the 'sg-0a70f9ab7004d77e1 - default' security group is shown, specifically the 'Règles entrantes' (Inbound Rules) tab. This tab lists two rules: one for TCP port 5439 (IPv4) and another for 'Tout le trafic' (All traffic).

Name	ID du groupe de sécurité	Nom du groupe de sécurité	ID de VPC
<input checked="" type="checkbox"/> sg-0a70f9ab7004d77e1	sg-0a70f9ab7004d77e1	default	vpc-0fc6694118fde500f

Règles entrantes (2)					
Name	ID de règle de groupe	Version IP	Type	Protocole	
-	sgr-0a81d93a5a5e4742c	IPv4	TCP personnalisé	TCP	
-	sgr-0dcbb1bb4b95569ed7	-	Tout le trafic	Tous	

6.3. Sécurité Avancée : Rôles IAM et Secrets Manager

L'automatisation ne doit jamais se faire au détriment de la sécurité. Dans un environnement de développement local, il est courant (bien que risqué) de stocker les mots de passe dans des fichiers de configuration. Sur AWS, cette pratique est à bannir en production. Nous avons donc utilisé AWS Secrets Manager pour stocker les identifiants de connexion à Redshift Serverless (mot de passe uniquement, les autres sont injecté dans les variables d'environnement au moment du run de la task definition) sous forme chiffrée.

Le défi technique ici résidait dans l'intégration entre ECS et Secrets Manager. Au lieu de passer les mots de passe en clair dans les variables d'environnement de la Définition de Tâche, nous avons utilisé la fonctionnalité ValueFrom. Cela permet d'injecter la

référence de l'ARN (Amazon Resource Name) du secret. Au moment du lancement du conteneur, l'agent ECS récupère dynamiquement la valeur déchiffrée et la présente à l'application comme une variable d'environnement standard. Pour que cette magie opère, nous avons dû configurer des rôles IAM (Identity and Access Management) très granulaires .

Nous avons défini un Rôle d'Exécution de Tâche (Task Execution Role) spécifique. Ce rôle est une carte d'identité numérique qui donne à notre service ECS des permissions explicites : le droit de tirer l'image depuis ECR, le droit de lire la valeur spécifique du secret dans Secrets Manager, et le droit d'écrire des flux de logs dans CloudWatch. Sans ce rôle correctement configuré, le conteneur échouerait au démarrage avec une erreur d'autorisation, invisible pour l'utilisateur lambda.

The screenshot shows the AWS IAM console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and various service links like Amazon Redshift, DynamoDB, Elastic Container Service, and Amazon EventBridge. On the far right, it shows the user 'diloma10 (3551)' and some global settings. Below the navigation, the path 'IAM > Rôles > dbt-ecs-task-role-olist > Créer une politique' is visible. The main area is titled 'Spécifier les autorisations' (Step 1). It includes a 'JSON' tab and a 'Actions' dropdown. A code editor displays the following JSON policy:

```
1 ▼ {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": "secretsmanager:GetSecretValue",
7       "Resource": "arn:aws:secretsmanager:eu-west-3:355142185625:secret:dbt-redshift-password"
8     }
9   ]
10 }
```

To the right of the editor, there are sections for 'Modifier l'instruction' (Edit instruction) and 'Sélectionner une instruction' (Select instruction), both currently empty. A button '+ Ajouter une nouvelle instru' (Add new instruction) is at the bottom of the selection section.

Screenshot of the AWS Secrets Manager console showing the details of a secret named "dbt-redshift-password".

Détails du secret

Clé de chiffrement: aws/secretsmanager

Nom du secret: dbt-redshift-password

ARN du secret: arn:aws:secretsmanager:eu-west-3:355142185625:secret:dbt-redshift-password-ekoQLC

Description du secret: -

Type de secret: -

Screenshot of the AWS Elastic Container Service (ECS) console showing the creation of a task definition.

Créer une révision Infos

Créez ou modifiez un fichier JSON définissant les définitions de conteneur et de volume pour une tâche Amazon ECS.

```
21     "value": "/usr/app/\" -"
22   },
23   "essential": true,
24   "image": "355142185625.dkr.ecr.eu-west-3.amazonaws.com/olist_dw_dbt_run_sim@sha256:eef9b9c06f9e2eb7aa56cf09934fa5a5a16384
25   "logConfiguration": {
26     "logDriver": "awslogs",
27     "options": {
28       "awsLogs-group": "/ecs/run_container_olist",
29       "awsLogs-create-group": "true",
30       "awsLogs-region": "eu-west-3",
31       "awsLogs-stream-prefix": "ecs"
32     },
33     "secretOptions": []
34   },
35   "mountPoints": [],
36   "name": "dbt-container-olist",
37   "portMappings": [
38     {
39       "appProtocol": "http",
40       "containerPort": 80,
41       "hostPort": 80,
42       "name": "dbt-container-olist-80-tcp",
43       "protocol": "tcp"
44     }
45   ],
46   "secrets": [
47     {
48       "name": "DBT_REDSHIFT_PASSWORD",
49       "valueFrom": "arn:aws:secretsmanager:eu-west-3:355142185625:secret:dbt-redshift-password-ekoQLC:password::"
50     }
51   ]
52 }
```

Annuler

AWS Rechercher [Alt+S] Europe (Paris) ▾

Amazon Redshift DynamoDB Elastic Container Service Amazon EventBridge

AWS Secrets Manager > Secrets > dbt-redshift-password

Détails du secret

Clé de chiffrement	Description du secret
aws/secretsmanager	-
Nom du secret	Type de secret
dbt-redshift-password	-
ARN du secret	
arn:aws:secretsmanager:eu-west-3:355142185625:secret:dbt-redshift-password-ekoQIC	

Vue d'ensemble Rotation Versions Réplication Balises

Valeur du secret Infos

Récupérez et affichez la valeur du secret.

Clé/valeur Texte brut

Clé du secret	Valeur du secret
username	dbt_user
password	Passer123@0
dbname	dev
host	sim-dw-olist-01.355142185625.eu-west-3.redshift-serverless.amazonaws.com
port	5439
namespaceName	namespace-olist-dw-01
engine	redshift

AWS Rechercher [Alt+S] Europe (Paris) ▾

Amazon Redshift DynamoDB Elastic Container Service Amazon EventBridge

Amazon Elastic Container Service > Définitions de tâches > run_container_olist > Révision 1 > Créer une révision

Amazon Elastic Container Service

- Mode Express Nouveau
- Clusters
- Espaces de noms
- Définitions de tâches**
- Paramètres du compte

- Amazon ECR
- Référentiels

- AWS Batch

- Documentation
- Découvrir les produits
- Abonnements

Laissez-nous un commentaire.

Supprimer

Clé Type de valeur
DBT_REDSHIFT_HOST Valeur

Valeur Supprimer

Clé Type de valeur
DBT_REDSHIFT_PASSWORD Valeur

Valeur 625:secret:dbt-redshift-password:password:

Le password sera pas dans les variables d'environnement mais dans les 'secrets'

Clé Type de valeur
DBT_REDSHIFT_USER Valeur

Valeur dbt_user

CloudShell Commentaires Console de l'application mobile © 2025, Amazon Web Services, Inc. ou ses affiliés. Confidentialité Conditions

6.4. Orchestration Temporelle avec AWS Step Functions

Avoir un conteneur capable de transformer les données est une chose, savoir quand le lancer en est une autre. Notre flux de données est continu, avec des fichiers arrivant via Firehose à des intervalles irréguliers. Lancer une tâche ECS à chaque fichier serait inefficace et coûteux (temps de démarrage du conteneur, surcharge des connexions Redshift). Nous avions besoin d'un chef d'orchestre intelligent.

```
1 {
2   "source": ["aws.s3"],
3   "detail-type": ["Object Created"],
4   "detail": {
5     "bucket": {
6       "name": ["project-olist-ensae"]
7     },
8     "object": {
9       "key": [
10         {
11           "prefix": "bronze-sim/"
12         }
13       ]
14     }
15   }
16 }
```

C'est le rôle d'AWS Step Functions. Nous avons dessiné une machine à états (State Machine) qui gère la logique de déclenchement. Le processus, défini en langage ASL (Amazon States Language), commence par une étape de vérification : CheckIfAnotherRunIsActive. La machine interroge l'API AWS pour voir si une exécution est déjà en cours. Cette étape est cruciale pour éviter les conditions de concurrence où deux transformations essaieraient de modifier les mêmes tables en même temps.

Si le système est libre, la Step Function passe dans un état d'attente nommé WaitFiveMinutes. Ce délai agit comme un tampon temporel (buffer), permettant à plusieurs fichiers Parquet de s'accumuler dans le Data Lake. Une fois ce délai écoulé, l'étape RunDbtTask est déclenchée. Elle utilise l'intégration native avec ECS pour lancer notre tâche Fargate avec les paramètres réseaux définis précédemment (Cluster, Subnets, Security Groups). Ce mécanisme de "micro-batching" orchestré par Step Functions nous permet de simuler un traitement quasi temps réel tout en optimisant l'utilisation des ressources et en stabilisant les performances de notre Data Warehouse.

dbt-run-orchestrator Standard

Concevoir | Code | Configuration

Quitter Actions Exécuter Enregistrer

Annuler Rétablir Format Copier Commandes Zoom avant Zoom arrière Centre

```

15
16
17 "IsAlreadyRunning": {
18   "Type": "Choice",
19   "Choices": [
20     {
21       "Variable": "$.running.executions[0]",
22       "IsPresent": true,
23       "Next": "IgnoreEvent"
24     }
25   ],
26   "Default": "WaitFiveMinutes"
27 },
28
29 "IgnoreEvent": {
30   "Type": "Succeed"
31 },
32
33 "WaitFiveMinutes": {
34   "Type": "Wait",
35   "Seconds": 300,
36   "Next": "RunDbtTask"
37 },
38
39 "RunDbtTask": {
40   "Type": "Task".

```

```

graph TD
    Start((Start)) --> ListExecutions[Step Functions: ListExecutions  
CheckIfAnotherRunIsActive]
    ListExecutions --> Choice{Choice state  
IsAlreadyRunning}
    Choice -- "$.running.executions[0] is present" --> Succeed[Success state  
IgnoreEvent]
    Choice -- Default --> Wait[Wait state  
WaitFiveMinutes]
    Succeed --> End((End))
    Wait --> RunDbtTask[ECS: RunTask  
RunDbtTask]
    RunDbtTask --> End

```

Step Functions > State machines > dbt-run-orchestrator > Execution: version_8.9.1_rebuild_task_for_maj

Step Functions

- Tableau de bord
- Machines d'état
- Activités

Ressources pour développeurs

- Inspecteur d'exécution
- Atelier d'apprentissage en ligne
- Développement local
- Simulateur de flux de données
- Coup de projecteur sur la fonction
- Documentation

Rejoindre notre panneau de commentaires

Événements (17)

ID	Type	Step	Resource	Démarré après
1	ExecutionStarted			0
2	TaskStateEntered	CheckIfAnotherRunIsActive	aws-sdk:sfn:listExecutions	00:00:00.036
3	TaskScheduled	CheckIfAnotherRunIsActive	aws-sdk:sfn:listExecutions	00:00:00.036
4	TaskStarted	CheckIfAnotherRunIsActive	aws-sdk:sfn:listExecutions	00:00:00.085
5	TaskSucceeded	CheckIfAnotherRunIsActive	aws-sdk:sfn:listExecutions	00:00:00.182
6	TaskStateExited	CheckIfAnotherRunIsActive		00:00:00.207

j-west-3.console.aws.amazon.com/states/home?region=eu-west-3#/homepage © 2025, Amazon Web Services, Inc. ou ses affiliés. Confidentialité Conditions Préférences relatives aux cookies

aws | Rechercher [Alt+S] | Europe (Paris) | Moussa-ensa

Amazon Redshift | DynamoDB | Elastic Container Service | Amazon EventBridge

Amazon EventBridge > Planifications > Olist-Ticker

Olist-Ticker

La planification Olist-Ticker a été désactivée.

Activer | Modifier | Supprimer

Détails de la planification

Nom de la planification Olist-Ticker	Status Désactivé	Heure de début de la planification	Fenêtre horaire flexible
Description Lance notre lambda order_generator	ARN de planification arn:aws:scheduler:eu-west-3:355142185625:schedule/default/Olist-Ticker	Heure de fin de la planification Jan 02, 2026, 00:01:00 (UTC+00:00)	Date de création Dec 14, 2025, 21:02:53 (UTC+00:00)
Nom du groupe de planifications default	Action après l'achèvement NONE	Fuseau horaire d'exécution UTC	Date de dernière modification Dec 19, 2025, 00:19:02 (UTC+00:00)

Planification | Cible | Politique de nouvelles tentatives | File d'attente de lettres mortes

Planification

Fréquence fixe Infos
rate (1 minutes)

aws | Rechercher [Alt+S] | Europe (Paris) | Moussa-ensa

Amazon Redshift | DynamoDB | Elastic Container Service | Amazon EventBridge

Amazon EventBridge > Règles

Selectionnez un bus à événements

Bus d'événements

Sélectionner ou entrer un nom de bus d'événements

default

Règles relatives aux bus d'événements default (2)

Supprimer | Activer | Modifier | Modèle CloudFormation | Crée une règle

Nom	Statut	Type	Bus d'événements	ARN	Description
dbt-run-orchestrator	Activé	Standard	default	arn:aws:events:eu-west-3:355142185625:rule/dbt-run-orchestrator	-
StepFunctionsGetEventsForECSTaskRule	Activé	Géré	default	arn:aws:events:eu-west-3:355142185625:rule/StepFunctionsGetEventsForECSTaskRule	This rule is used to notify Step Functions regarding AWS ECS Tasks

AWS | Rechercher [Alt+S] | Europe (Paris) | diloma10 (355)

Amazon Redshift | DynamoDB | Elastic Container Service | Amazon EventBridge

Step Functions > State machines > dbt-run-orchestrator > Execution: version_8.9.1_rebuild_task_for_maj

Step Functions

- Tableau de bord
- Machines d'état**
- Activités

▼ Ressources pour développeurs

- Inspecteur d'exécution
- Atelier d'apprentissage en ligne ↗
- Développement local
- Simulateur de flux de données
- Coup de projecteur sur la fonction
- Documentation ↗

Rejoindre notre panneau de commentaires ↗

Exécution démarlée avec succès

Exécution: version_8.9.1_rebuild_task_for_maj

Modifier la machine d'état Nouvelle exécution Actions ▾

Détails	Entrée et sortie d'exécution	Définition
Statut d'exécution Opération réussie	Heure de début 18 déc. 2025, 21:16:04,180 (UTC-00:00)	
Type d'exécution Standard	Heure de fin 18 déc. 2025, 21:21:53,618 (UTC-00:00)	
ARN d'exécution arn:aws:states:eu-west-3:355142185625:execution:dbt-run-orchestrator:version_8.9.1_rebuild_task_for_maj	Durée 00:05:49.438	
ARN du rôle IAM arn:aws:iam::355142185625:role/service-role/StepFunctions-dbt-run-orchestrator-role-fcfn8qbnw ↗	Alias -	
Transitions d'état En savoir plus ↗ 6	Version -	

AWS | Rechercher [Alt+S] | Europe (Paris) | diloma10 (355)

Amazon Redshift | DynamoDB | Elastic Container Service | Amazon EventBridge

Step Functions > State machines > dbt-run-orchestrator > Execution: version_8.9.1_rebuild_task_for_maj

Step Functions

- Tableau de bord
- Machines d'état**
- Activités

▼ Ressources pour développeurs

- Inspecteur d'exécution
- Atelier d'apprentissage en ligne ↗
- Développement local
- Simulateur de flux de données
- Coup de projecteur sur la fonction
- Documentation ↗

Rejoindre notre panneau de commentaires ↗

Exécution démarlée avec succès

Exécution: version_8.9.1_rebuild_task_for_maj

Modifier la machine d'état Nouvelle exécution Actions ▾

Détails	Entrée et sortie d'exécution	Définition
Statut d'exécution Opération réussie	Heure de début 18 déc. 2025, 21:16:04,180 (UTC-00:00)	
Type d'exécution Standard	Heure de fin 18 déc. 2025, 21:21:53,618 (UTC-00:00)	
ARN d'exécution arn:aws:states:eu-west-3:355142185625:execution:dbt-run-orchestrator:version_8.9.1_rebuild_task_for_maj	Durée 00:05:49.438	
ARN du rôle IAM arn:aws:iam::355142185625:role/service-role/StepFunctions-dbt-run-orchestrator-role-fcfn8qbnw ↗	Alias -	
Transitions d'état En savoir plus ↗ 6	Version -	

Amazon Redshift DynamoDB Elastic Container Service Amazon EventBridge

Step Functions > Machines d'état > Machine d'état: dbt-run-orchestrator

Step Functions

- Tableau de bord
- Machines d'état
- Activités
- Ressources pour développeurs
 - Inspecteur d'exécution
 - Atelier d'apprentissage en ligne
 - Développement local
 - Simulateur de flux de données
 - Coup de projecteur sur la fonction
 - Documentation

Rejoindre notre panneau de commentaires

	Nom	Statut	Heure de début (lo...)	Heure de fin (locale)
<input type="checkbox"/>	version_8.9.1_rebuild_task_for_maj	Opération réussie	18 déc. 2025, 21:16:04	18 déc. 2025, 21:21:5
<input type="checkbox"/>	version_8.9_rebuild_image	Échec	18 déc. 2025, 20:28:38	18 déc. 2025, 20:34:2
<input type="checkbox"/>	version_8.8_add_git_in_dockerfile	Échec	18 déc. 2025, 19:25:19	18 déc. 2025, 19:31:0
<input type="checkbox"/>	version_8.7_switch_password_to_bloc_secr...	Échec	18 déc. 2025, 18:27:14	18 déc. 2025, 18:33:0
<input type="checkbox"/>	version_8.6_print_password	Échec	18 déc. 2025, 17:59:52	18 déc. 2025, 18:05:5
<input type="checkbox"/>	version_8.5	Échec	18 déc. 2025, 17:42:24	18 déc. 2025, 17:48:1
<input type="checkbox"/>	version_7.3	Échec	18 déc. 2025, 10:19:32	18 déc. 2025, 10:20:2
<input type="checkbox"/>	Version_7.2	Échec	18 déc. 2025, 09:43:04	18 déc. 2025, 09:43:5
<input type="checkbox"/>	version_6.5	Échec	17 déc. 2025, 11:55:04	17 déc. 2025, 11:55:5
<input type="checkbox"/>	version_6.3_modif_password_patiente	Échec	16 déc. 2025, 16:48:50	16 déc. 2025, 16:49:4
<input type="checkbox"/>	version_6_changement_pass_word_dbt_u...	Échec	16 déc. 2025, 16:34:04	16 déc. 2025, 16:35:0
<input type="checkbox"/>	v5.3	Échec	16 déc. 2025, 16:14:02	16 déc. 2025, 16:15:0
<input type="checkbox"/>	v5.2	Échec	16 déc. 2025, 15:22:33	16 déc. 2025, 15:23:2
<input type="checkbox"/>	version_4.1	Échec	16 déc. 2025, 15:17:34	16 déc. 2025, 15:18:2
<input type="checkbox"/>	version_3.1	Échec	16 déc. 2025, 15:12:01	16 déc. 2025, 15:12:5
<input type="checkbox"/>	v2_last	Échec	16 déc. 2025, 15:06:28	16 déc. 2025, 15:07:0

aws Rechercher [Alt+5] Europe (Paris)

Amazon Redshift DynamoDB Elastic Container Service Amazon EventBridge

Redshift query editor v2

Editor Create Load data Filter resources Serverless: s... dev Run Limit 100 Explain Isolated session

Queries Notebooks Charts History scheduled queries

Serverless: default-workgroup Serverless: sim-dw-olist-01 Serverless: olist

Schedule Result 1 (10) Export

```

1 SELECT
2   order_id,
3   order_purchase_timestamp
4 FROM
5   olist_spectrum_schema.table_order_sim
6 ORDER BY order_purchase_timestamp DESC
7 LIMIT 10;
  
```

order_id	order_purchase_time...
33285aeb-91e0-4a8e-881...	2018-02-02 17:27:10
802f706d-21ac-4e45-ab8...	2018-02-02 16:00:44
49650b8b-6a23-4094-b76...	2018-02-02 14:42:48
8729522d-74e4-4f61-bf33...	2018-02-02 12:38:30
a9aad474-8509-4cec-974...	2018-02-02 11:21:41
a4e6c0ee-c267-47dc-b6a...	2018-02-02 08:56:42
06b3da5e-9910-4598-b83...	2018-02-02 08:48:58
657652c9-5b35-450d-93f...	2018-02-02 07:48:55
441a900e-4f0e-4d0e-84b...	2018-02-02 07:02:24

Query ID 2036111 Elapsed time: 758 ms

AWS dilma10 [35]

Rechercher

Amazon Redshift Amazon EventBridge Elastic Container Service DynamoDB

Redshift query editor v2

Editor [Alt+5] 🔍

Queries Create Load data

Filter resources

Serverless: sl... dev

Schedule ...

```
1 SELECT
2     order_id,
3     order_purchase_timestamp
4 FROM
5     olist_spectrum_schema.table_order_sim
6 ORDER BY order_purchase_timestamp DESC
7 LIMIT 10;
```

Result 1 (10)

	order_id	order_purchase_time...
1	33285aeb-91e0-4a8e-881...	2018-02-02 17:27:10
2	802f706d-21ac-4e45-ab8...	2018-02-02 16:00:44
3	49650b8b-6a23-4094-b76...	2018-02-02 14:42:48
4	8729522d-74e4-4f81-bf33...	2018-02-02 12:38:30
5	a9aad474-8809-4cec-974...	2018-02-02 11:21:41
6	a4e6c0ee-c267-47dc-b6a...	2018-02-02 08:56:42
7	06b3da5e-9910-4598-b83...	2018-02-02 08:48:56
8	657652c9-5b35-459d-93f...	2018-02-02 07:48:55
9	AMAZONAE-AN00-A00E-SAN	2018-02-02 07:02:24
10	AMAZONAE-AN00-A00E-SAN	2018-02-02 07:02:24

Export

Query ID 2036111 Elapsed time: 758 ms

The screenshot shows the AWS Redshift query editor v2 interface. On the left is a sidebar with icons for Editor, Queries, Notebooks, Charts, History, Scheduled queries, and Settings. The main area has tabs for 'last_element_bronze_S3' and 'Last_elements_silver'. The current tab is 'last_element_bronze_S3'. The top navigation bar includes 'Rechercher', 'Amazon Redshift', 'DynamoDB', 'Elastic Container Service', and 'Amazon EventBridge'. The top right shows 'Europe (Paris)' and a user icon. The main content area contains a query editor with a code editor showing a SELECT statement and a results table titled 'Result 1 (10)' with 10 rows of data. The bottom right shows 'Query ID 2036111' and 'Elapsed time: 758 ms'.

```
1  SELECT
2    order_id,
3    order_purchase_timestamp,
4    customer_city,
5    total_price
6   FROM
7    dt_tms_silver_gold.silver_sim
8   ORDER BY
9    order_purchase_timestamp DESC
10   LIMIT 10;
```

	order_id	order_purchase_timestamp	customer_city	total_price
1	8cc0a279-1e72-4722-887...	2018-01-11 09:01:58	Rezende de da Paz	117.13808687707422
2	5b09670-5537-4c1f-b680...	2018-01-11 07:36:35	Mendonça de Rodrigues	121.3851717919708
3	a75385d-9c71-4677-900...	2018-01-11 06:17:12	Pereira	144.60171434177624
4	ecd27c24-a234-4579-806...	2018-01-11 03:46:46	Rezende do Campo	50.04586437231542
5	bb1950c8-e93c-47d3-ae9...	2018-01-11 03:13:18	Brilo Alegre	142.4790736413283
6	20e3ba1d-d23e-47e3-ba8...	2018-01-11 02:58:30	Sá	150.106128350022
7	c6303e46-57e1-48ab-9e2...	2018-01-11 01:36:00	Câmara	39.18213280994516
8	2ef1696b-b8a3-4248-a3e...	2018-01-11 01:17:24	Fernera	76.73919860121273
9	fe90e949-7402-4d92-ab3...	2018-01-11 00:18:46	Araújo de Fonsêca	97.56967134528199
10	db68c4e2-e3f2-435d-888...	2018-01-10 23:11:24	Pacheco da Prata	177.38349912368864

7- La connexion avec Power bi

Maintenant que nous avons automatisé le process il suffit de créer un utilisateur avec redshift et de se brancher sur redshift depuis cette utilisateur .On donne a l'utilisateur que les accès de lecture.

```
-- RUN ON active connection | -- Select block
1 CREATE USER bi_analyst WITH PASSWORD 'P@SSw0rd78945@#$%&';
2
3 -- Autoriser l'utilisateur à se connecter à la base de données
4 GRANT CONNECT ON DATABASE dev TO bi_analyst;
5
6 -- Autoriser l'utilisateur à voir le schéma externe créé
7 GRANT USAGE ON SCHEMA olist_spectrum_schema TO bi_analyst;
8
9
10 -- Donner accès en lecture seule à la table
11 GRANT SELECT ON ALL TABLES IN SCHEMA olist_spectrum_schema TO bi_analyst;
12
13 ALTER DEFAULT PRIVILEGES IN SCHEMA olist_spectrum_schema
14 GRANT SELECT ON TABLES TO bi_analyst;
15
16 -- Nouveaux droits pour Silver/Gold
17
18 GRANT USAGE ON SCHEMA dbt_sim_silver_gold TO bi_analyst;
19
20 GRANT SELECT ON ALL TABLES IN SCHEMA dbt_sim_silver_gold TO bi_analyst;
21
22 ALTER DEFAULT PRIVILEGES IN SCHEMA dbt_sim_silver_gold
23 GRANT SELECT ON TABLES TO bi_analyst;
24
25
26 -- Nouveaux droit pour silver/gold apres changement :
27
28 -- 1. Autoriser l'accès au schéma
29 GRANT USAGE ON SCHEMA dbt_sim_silver_gold TO bi_analyst;
30
31 -- 2. Donner les droits SELECT sur les tables actuelles (gold_sim vient d'être recréée)
32 GRANT SELECT ON ALL TABLES IN SCHEMA dbt_sim_silver_gold TO bi_analyst;
33
34 -- 3. Définir les droits par défaut pour que les prochains "dbt run" donnent automatiquement les droits
35 ALTER DEFAULT PRIVILEGES IN SCHEMA dbt_sim_silver_gold
36 GRANT SELECT ON TABLES TO bi_analyst;
37
38
39 --- pour le dbt user
40
41
```

Untitled - Power BI Desktop

File Home Insert Modeling View Optimize Help

Clipboard

Navigator

silver_sim

	order_purchase_timestamp	product_id
d-4c3e-bc35-7d6fbabbff9c	02/01/2018 14:00:00	ea999c97061d17b7fe7e85d842
34-45ca-b409-065964f9134e	02/01/2018 14:00:00	fb3fae6c3a74bc9fd48a65d00e
b-439a-b01e-40d5628eb0bd	02/01/2018 14:00:00	66a5f1151e8a4215ba3ce6e59
1-4011-aa40-d457237774b4	02/01/2018 15:00:00	e065f51f1e3050bd276cabbb4d
9-497a-b03e-5b044871d786	02/01/2018 15:00:00	5f1d25eacf1c3821a3a53a52da
t-4c01-9981-fa7a417e36c	02/01/2018 16:00:00	2c7dec02b1eecc3secaf62d0e
2-4760-9cb0-a5aa5d5d7d31	02/01/2018 16:00:00	fef526cd8279b98e8e83d3e6b5
'-4f82-80f9-b2af9b036c701	02/01/2018 16:00:00	13c7f50d037a0c12e243d2e6b
b-4907-aecb-541608942d8f	02/01/2018 16:00:00	244811270c7b8b6cce69e34a0
i-4bd7-88b8-62818d5c7996	02/01/2018 18:00:00	078be1a1d40808274672847d88c
b-432e-bc14-59e7ca09dc	02/01/2018 18:00:00	a32a25c59034c4b034031798e
7-45fc-8303-65519a15a70c	02/01/2018 18:00:00	7950263daee3988ed39eeb271
c-4c9d-9209-dcc3f1f63bba	02/01/2018 19:00:00	3b91132e7eaba17b615edbd
14-4f57-b4fb-bd181fc9234	02/01/2018 19:00:00	dfdd4a0d70650d74a92d3d31b
'-492a-8c02-3f0d02d96f64	02/01/2018 19:00:00	ce04dcb7a9649d4953f5d4a
b-4fe3-8201-ec0d7f659040	02/01/2018 19:00:00	11774988372a9ee0f836c4c7
7-455e-9521-df18ae789ac	02/01/2018 20:00:00	7d2e2a2c7096151cd14274d42
i-417f-a07b-5fc1a184395be	02/01/2018 20:00:00	1823142d83a24b19beb008b
30-4e06-a6ad-05d59f9d4fc	02/01/2018 20:00:00	fb40a08ee4c3aa79a86833c838
1-41fb-9c24-8af423a2bf9	03/01/2018 03:00:00	2942e5008ea1e055a5d893
ib-4ce8-a786-41a726793a7d	03/01/2018 03:00:00	33c59c2bf23093baae87f60a49
'-4e70-9e44-24bf273583c3	03/01/2018 03:00:00	08ef02b6aae73d3a385f1342f
id-4eac-bdd5-2600e18e779b	03/01/2018 03:00:00	fe4dece0c63ba4f19f204d3b

Load Transform Data Cancel

Untitled - Power BI Desktop

File Home Insert Modeling View Optimize Help

Clipboard

Navigator

gold_sim

	order_date	customer_state	total_orders	daily_revenue
02/01/2018 00:00:00	"AP"	6	725,6396911	
02/01/2018 00:00:00	"PA"	7	471,8314998	
02/01/2018 00:00:00	"RO"	7	896,2623351	
02/01/2018 00:00:00	"AL"	5	551,3093585	
02/01/2018 00:00:00	"RN"	11	960,5999842	
02/01/2018 00:00:00	"GO"	9	818,856314	
02/01/2018 00:00:00	"MA"	6	633,9273882	
02/01/2018 00:00:00	"AM"	6	667,1186197	
02/01/2018 00:00:00	"SP"	17	1853,071607	
02/01/2018 00:00:00	"BA"	9	1098,539877	
02/01/2018 00:00:00	"TO"	10	1099,369265	
02/01/2018 00:00:00	"MG"	12	1140,44655	
02/01/2018 00:00:00	"RR"	10	952,1066565	
02/01/2018 00:00:00	"PI"	8	874,3274784	
02/01/2018 00:00:00	"ES"	16	1745,548694	
02/01/2018 00:00:00	"SE"	14	1468,197146	
02/01/2018 00:00:00	"RU"	13	1553,920756	
02/01/2018 00:00:00	"DF"	15	1428,89453	
02/01/2018 00:00:00	"MS"	8	612,5592916	
02/01/2018 00:00:00	"PB"	9	907,9103784	
02/01/2018 00:00:00	"PR"	13	1294,728624	
02/01/2018 00:00:00	"RS"	18	1743,972284	
02/01/2018 00:00:00	"CE"	8	730,9243154	
02/01/2018 00:00:00	"MT"	13	1170,913866	

