Fluxograma de Execucao — Pipeline do Laboratorio

Gerado em: 2025-12-18 14:15

Objetivo

Este documento descreve, de forma simples, a ordem de execucao do laboratorio e como os componentes (containers, ingestor, Terraform, Glue, Athena e API) se encadeiam.

Fluxo — visao simples

```
[1] Subir o lab local
    Arquivo: data_bases/docker-compose.yaml
    Acao: docker compose up -d
        |
        v
[2] Bancos inicializam dados de exemplo (1o start / volume vazio)
    Postgres: data_bases/docker/postgres/init/*.sql
    MySQL:    data_bases/docker/mysql/init/*.sql
    Mongo:    data_bases/docker/mongo/init/seed_suporte.js
        |
        v
[3] Ingestao RAW (container ingestor)
    Arquivo: data_bases/ingestor/main.py
    Faz: le Postgres/MySQL/Mongo -> grava Parquet + manifest no S3 (raw)
        |
        v
[4] S3 recebe camada RAW
    Saidas:
      - raw/<dominio>/<entidade>/ingestion_date=.../run_id=.../part-0000.parquet
      - raw/_manifests/ingestion_date=.../run_id=.../manifest.json
        |
        v
[5] Pipeline IaC + Glue
    Arquivo: data_bases/iac/run_pipeline.sh
    Faz: (a) terraform init/plan/apply
         (b) start RAW crawler -> start Glue Job -> start CURATED crawler
        |
        v
[6] Glue Job gera CURATED
    Script: data_bases/iac/scripts/unified_job.py
```

```
    Saida:
      - curated/clientes_unificados/ano=YYYY/mes=MM/...
        |
        v
[7] Catalogo e consumo
    - Glue Crawler (curated) cataloga no Data Catalog
    - Athena consulta clientes_unificados
    - Power BI conecta via ODBC (Simba Athena)
        |
        v
[8] (Opcional) API
    API Gateway -> Lambda (data_bases/iac/lambda/clientes_api.py) -> Athena -> JSON
```

Ordem de execucao — arquivo por arquivo

A) Lab local (Docker)

- **Arquivo:** data_bases/docker-compose.yaml
  - **Quem executa:** Voce
  - **Quando:** Inicio do laboratorio (docker compose up -d)
  - **Efeito:** Sobe containers e redes (Postgres/MySQL/Mongo/ingestor/terraform etc.).
- **Arquivo:** data_bases/docker/postgres/init/*.sql
  - **Quem executa:** Container Postgres
  - **Quando:** Primeira inicializacao (volume vazio)
  - **Efeito:** Cria tabelas e insere dados do dominio financeiro (customers/ledger).
- **Arquivo:** data_bases/docker/mysql/init/*.sql
  - **Quem executa:** Container MySQL
  - **Quando:** Primeira inicializacao (volume vazio)
  - **Efeito:** Cria tabelas e insere dados do dominio vendas (orders/order_items).
- **Arquivo:** data_bases/docker/mongo/init/seed_suporte.js
  - **Quem executa:** Container Mongo
  - **Quando:** Primeira inicializacao (volume vazio)
  - **Efeito:** Cria colecao/indexes e insere tickets do dominio suporte.
- **Arquivo:** data_bases/ingestor/main.py
  - **Quem executa:** Container ingestor
  - **Quando:** Apos os bancos estarem acessiveis

- **Efeito:** Extrai dados e grava Parquet + manifest no S3 (raw).

B) Infra AWS (Terraform) + orquestracao (shell)

O script run_pipeline.sh aplica a infraestrutura e dispara os jobs do Glue.

- **Orquestracao principal:** data_bases/iac/run_pipeline.sh
- **Terraform (recursos AWS):** data_bases/iac/*.tf (Glue Database, Crawlers, Job, Lambda, API Gateway).

C) Processamento e consumo (AWS)

- **Glue Job:** data_bases/iac/scripts/unified_job.py
  - Le Parquet do raw, agrega/junta e grava curated/clientes_unificados (particionado por ano/mes).
- **Lambda:** data_bases/iac/lambda/clientes_api.py
  - Recebe /clientes?nome=..., executa query no Athena e devolve JSON.
- **Athena:** (console ou clientes)
  - Consulta clientes_unificados; resultados vao para s3://<bucket>/athena-results/.
- **Power BI:** (Windows) ODBC Simba Athena
  - Conecta no Athena via DSN e importa tabelas para dashboard.

Saidas geradas — onde olhar

- - S3 RAW: raw/... (Parquet + _manifests)
- - S3 CURATED: curated/clientes_unificados/ano=YYYY/mes=MM/...
- - S3 Athena results: athena-results/ (saida de queries)
- - Glue Data Catalog / Athena: base e tabelas catalogadas pelos crawlers
- - API: API Gateway invoca Lambda, que consulta Athena