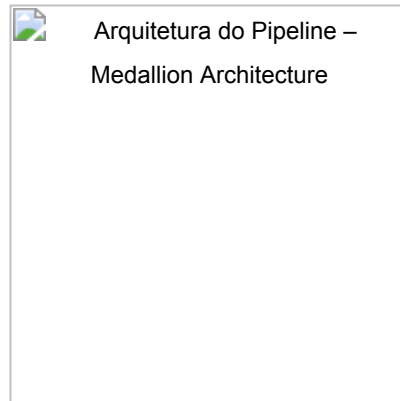


# Arquitetura Proposta do Pipeline de Dados

A arquitetura implementa um pipeline baseado na **Medallion Architecture (Bronze → Silver → Gold)**, garantindo **backfill histórico e observabilidade**.



**Figura:** Arquitetura do pipeline com camadas Bronze, Silver e Gold, observabilidade e backfill.

## Visão Geral do Fluxo

### 1. Fontes Públicas (ONS, CCEE, ANEEL)

Dados em CSV, JSON, XML ou APIs são consumidos diretamente pelo pipeline.

### 2. Agendamento e Orquestração

- **Cloud Scheduler** executa o pipeline via cron (ex.: diariamente às 03h).
- **Cloud Composer (Airflow)** orquestra a DAG de ingestão:
  - baixar dados
  - salvar raw
  - processar e validar
  - gerar Parquet
  - carregar no DW

### 3. Camada Landing/Raw – Bronze (Cloud Storage)

- Armazena **dados brutos exatamente como vieram**, preservando estrutura e formato.
- Serve para **auditoria** e **backfill**, permitindo reproduzir qualquer processamento passado.

### 4. Camada de Processamento

Executada em:

- **Cloud Run** (containers serverless)
- **Cloud Functions** (funções leves)
- **Compute Engine** (máquinas para processamento pesado)

Responsável por:

- validação de schema
- limpeza e normalização
- unificação de schemas (carga + geração)
- regras de data quality

#### 5. Camada de Curadoria – Silver

Dados tratados são gravados como **arquivos Parquet padronizados** no Storage, garantindo schema consistente e pronto para análise.

#### 6. DW / Lakehouse – Gold (BigQuery)

Dados Silver são carregados em tabelas analíticas (fatos/dimensões), usando operações **idempotentes** (**MERGE/UPSERT**) para atualizar registros antigos.

## 2. Backfill (Reprocessamento Histórico)

A arquitetura suporta backfill de forma nativa:

- O **Composer** possui um fluxo dedicado de backfill.
- A DAG aceita parâmetros de período (`start_date / end_date`).
- Para cada período reprocessado:
  1. Lemos os arquivos **raw (Bronze)** ou baixamos a fonte novamente.
  2. Reprocessamos a camada Silver (tratamento + Parquet).
  3. Carregamos no Gold usando **UPSERT**, substituindo valores antigos pelos corrigidos.

**Resultado:** o pipeline é **idempotente** — rodar duas vezes para o mesmo período sempre produz o mesmo estado final.

## 3. Observabilidade e Monitoramento

A arquitetura possui uma camada de observabilidade independente que cobre:

### 3.1. Logs Técnicos

- Todos os serviços (Composer, Run, Functions, Compute Engine) escrevem no **Cloud Logging**.
- O código registra logs estruturados contendo:
  - início da ingestão
  - quantidade de linhas
  - erros de schema
  - erros de data quality

### 3.2. Detecção de Falhas de Execução

- **Cloud Monitoring** coleta métricas de erros (HTTP 5xx, falhas no Composer, exceptions).
- Políticas de alerta monitoram:
  - falha de DAG

- falha de tasks
- exceções em Cloud Run/Functions

### 3.3. Detecção de Problemas de Dados (Dado Zerado ou Anômalo)

- A camada de processamento aplica regras de qualidade:
  - DataFrame vazio
  - colunas obrigatórias ausentes
- Quando uma regra falha:
  - O job lança uma exceção de **data\_quality\_error**
  - Um **alerta automático** é disparado para o time

### 3.4. Canais de Notificação

- Alertas podem ser enviados para:
  - Slack
  - E-mail
  - SMS