Guia de Implementação de Transformações com DBT no GCP

Este documento descreve, de forma estruturada, como implementar um pipeline de **transformação com DBT** no ecossistema GCP.

O objetivo é criar um fluxo **reprodutível, auditável e escalável** para rodar DBT localmente e em produção (Cloud Run Jobs).

Visão Geral deste Fluxo

O processo funciona em 4 macro-etapas:

1. **Setup Local (DBT funcionando)**
2. **Contratos (Sources + primeiros modelos Silver)**
3. **Qualidade (Dedup + Testes mínimos)**
4. **Produção (Docker + Cloud Run Job + execução/rollback)**

> DBT não coleta dados. DBT transforma dados já disponíveis no BigQuery.

Etapa 1 — Estrutura do Projeto

Crie o repositório do DBT e aplique a estrutura mínima:

```
/meu_projeto_dbt
├── dbt_project.yml
├── profiles.yml
├── models/
│   └── nome_fonte/
│       ├── silver/
│       └── gold/
├── macros/
├── tests/
└── docker/
```

Só siga adiante se:

- a pasta de domínio (ex: `models/nome_fonte/`) existir
- `silver/` e `gold/` estiverem **dentro** do domínio

Etapa 2 — Setup Local (ambiente DBT)

1. Crie ambiente virtual:

```
python -m venv dbt_env
source dbt_env/bin/activate
```

Windows:

```
.\dbt_env\Scripts\activate
```

2. Instale DBT BigQuery:

```
pip install dbt-bigquery
```

3. Inicialize o projeto:

```
dbt init meu_projeto_dbt
cd meu_projeto_dbt
```

4. Autentique no GCP:

```
gcloud auth login
gcloud auth application-default login
```

5. Valide o projeto:

```
gcloud config get-value project
```

6. Rode debug:

```
dbt debug
```

Só siga adiante se:

- `dbt debug` passar 100%
- o project/dataset do BigQuery estiver correto

Etapa 3 — Criar Sources (contrato com RAW/Bronze)

Crie o arquivo:

models/silver/nome_fonte/schema.yml

Exemplo mínimo:

version: 2

sources:
  - name: nome_fonte_raw
    database: SEU_PROJECT_ID
    schema: nome_fonte_raw
    tables:
      - name: raw_solicitacoes

Valide que o source resolve:

dbt debug
dbt ls --select source:nome_fonte_raw.raw_solicitacoes

Só siga adiante se:

- `dbt ls` listar a source corretamente

Etapa 4 — Criar Modelo Silver (staging confiável)

Crie um primeiro modelo em `models/nome_fonte/silver/`:

models/nome_fonte/silver/silver_nomefonte_nometabela.sql

Boas Práticas:

- implemente a deduplicação antes de fazer a tipagem (ler etapa seguinte)
- a última CTE **lista colunas explicitamente**
- nada de `select *` no final

Rode:

dbt run --select path:models/nome_fonte/silver

Só siga adiante se:

- o modelo criar tabela/view no BigQuery
- não houver erro de parse/tipagem

Etapa 5 — Deduplicar

Implemente deduplicação com:

- `ROW_NUMBER()`
- `PARTITION BY` na chave de negócio
- `ORDER BY dt_extract_utc DESC`
- filtro `row_num = 1`

Rode novamente:

dbt run --select silver_nome_fonte__solicitacoes

Só siga adiante se:

- a tabela final tiver 1 linha por chave
- você conseguir explicar qual registro “sobrevive”

Etapa 6 — Criar Testes (obrigatório)

No próprio `schema.yml` do modelo (no mesmo diretório):

models/nome_fonte/silver/schema.yml

Exemplo mínimo:

```
version: 2

models:
  - name: silver_nome_fonte__solicitacoes
    columns:
      - name: solicitacao_id
        tests:
          - not_null
          - unique
```

Execute:

dbt test --select path:models/nome_fonte/silver

Só siga adiante se:

- `not_null` e `unique` passarem

Etapa 7 — Criar 1º Modelo Gold (consumo)

Crie Gold dentro do domínio:

models/nome_fonte/gold/fct_nomefonte_nometabela.sql

Use apenas `ref()` para consumir Silver:

```
select
  solicitacao_id,
  empresa_id,
  status,
  data_hora_abertura
from {{ ref('silver_nome_fonte__solicitacoes') }}
```

Rode:

dbt run --select path:models/nome_fonte/gold

Só siga adiante se:

- Gold for construído **somente** a partir de Silver (`ref()`)

Etapa 8 — Dockerizar o DBT (para produção)

Crie um Dockerfile por fonte (padrão que você já usa):

docker build -t dbt-nome_fonte:latest -f docker/nome_fonte/Dockerfile .

Tag + push (Artifact Registry):

```
docker tag dbt-nome_fonte:latest
us-east1-docker.pkg.dev/PROJECT/repo-dbt-nome_fonte/dbt-nome_fonte:latest
docker push us-east1-docker.pkg.dev/PROJECT/repo-dbt-nome_fonte/dbt-nome_fonte:latest
```

Só siga adiante se:

- a imagem estiver no Artifact Registry

Etapa 9 — Atualizar Cloud Run Job (DBT em produção)

Atualize a job:

gcloud run jobs update job-dbt-nome_fonte --image us-east1-docker.pkg.dev/PROJECT/repo-dbt-nome_fonte/dbt-nome_fonte:latest --region us-east1

Só siga adiante se:

- a job atualizar sem erro
- os logs mostrarem execução do dbt

Etapa 10 — Execução Operacional (runbook)

Execução seletiva (padrão):

dbt run --select path:models/nome_fonte
dbt test --select path:models/nome_fonte/silver

Quando quebrar:

1. olhar logs do Cloud Run
2. olhar erro do BigQuery (SQL/permissão)
3. olhar logs do dbt (modelo/teste)

Rollback padrão:

- corrigir modelo
- rebuild do modelo afetado (rodar `dbt run -s nome_modelo`)
- reexecutar job