

README_dbt — Mise en place dbt (multi-projets) + profils + Docker

Ce document décrit les actions à mener côté **dbt** dans le repo **dpf-client** pour :

- installer dbt localement
 - gérer **plusieurs projets dbt** (**DBT_PROJECT**)
 - initialiser Git
 - créer **profiles/profiles.yml** adapté **BigQuery + ADC (Cloud Run)**
 - créer le **Dockerfile + entrypoint.sh** pour exécution en Cloud Run Job
-

1) Installer dbt localement

Prérequis

- Python 3
- pip
- (optionnel) **virtualenv / venv**

Installation (recommandée via venv)

Depuis la racine du repo :

```
./scripts/06_dbt_install.sh
source .venv/bin/activate
dbt --version
```

Versions par défaut : **dbt-core==1.10.3, dbt-bigquery==1.10.3** (modifiable via variables d'environnement).

2) Gérer plusieurs projets dbt (**DBT_PROJECT**)

Pourquoi plusieurs projets ?

Cela permet de séparer :

- un projet dbt par domaine fonctionnel
- ou un projet par client / use-case tout en conservant un seul pipeline CI/CD.

Convention

Chaque projet dbt vit dans un dossier dédié :

```
|- uc_01/  
|- uc_02/  
| ...
```

Création d'un projet dbt

```
./scripts/07_dbt_init_project.sh uc_02
```

Note : `dbt init` est interactif. Si tu veux du non-interactif, crée un squelette `dbt_project.yml` manuellement ou demande une version scriptée.

3) Initialiser Git (`git init`) et premier commit

Si le repo n'est pas déjà initialisé :

```
git init  
git add .  
git commit -m "Initial commit dbt + CI/CD"
```

4) Créer un répertoire `profiles/` dans le projet

Dans dbt, un `profiles.yml` décrit la connexion (ici BigQuery).

Pour Cloud Run Job, on utilise :

- `method: oauth`
- **ADC** (Application Default Credentials) via le **service account runtime** du job Cloud Run

Création automatique :

```
./scripts/08_dbt_create_profiles.sh uc_02
```

Cela crée :

- `uc_02/profiles/profiles.yml`

Contenu du `profiles.yml` (résumé)

- `PROJECT_ID` via env var
- `DATASET` via env var
- `BQ_LOCATION` via env var

- **DBT_TARGET** via env var (**dev | rec | prod**)
-

5) Créer le Dockerfile + entrypoint (Cloud Run)

Génération automatique

```
./scripts/09_dbt_create_docker_assets.sh uc_02
```

Cela crée :

- **uc_02/Dockerfile**
- **uc_02/entrypoint.sh**

Points importants

- Le Dockerfile installe dbt + dbt-bigquery
 - **DBT_PROFILES_DIR=/app/profiles**
 - L'entrypoint lance :
 - **dbt deps**
 - **dbt debug**
 - puis la commande **DBT_CMD** (par défaut **dbt build**)
-

6) Tester dbt en local (optionnel)

Avec authentification gcloud locale

```
export PROJECT_ID="xxx"
export DATASET="clients_dev"
export BQ_LOCATION="EU"
export DBT_TARGET="dev"
export DBT_PROFILES_DIR=". ./uc_02/profiles"

cd uc_02
dbt debug
dbt build
```

En local, **method: oauth** utilise tes identifiants gcloud (si tu es loggué et que tu as les droits).

7) Construire l'image Docker localement (optionnel)

Dans le dossier du projet dbt :

```
cd uc_02
docker build -t dbt-uc_02:local .
docker run --rm \
-e PROJECT_ID="$PROJECT_ID" \
-e DATASET="$DATASET" \
-e BQ_LOCATION="$BQ_LOCATION" \
-e DBT_TARGET="$DBT_TARGET" \
-e DBT_CMD="dbt build" \
dbt-uc_02:local
```

8) Intégration avec CI/CD Cloud Run Job

Le pipeline CD déploie l'image dans Artifact Registry et met à jour le Cloud Run Job en passant les env vars :

- PROJECT_ID
- DATASET
- BQ_LOCATION
- DBT_CMD

Dans Cloud Run, dbt utilise automatiquement la SA runtime du Job (ADC).

9) Adapter à plusieurs projets dbt dans CI/CD

Il y a 2 approches :

A) 1 repo = 1 projet dbt (simple)

Le Dockerfile est à la racine du repo.

B) 1 repo = plusieurs projets dbt (recommandé ici)

Chaque projet a son **Dockerfile**. Dans ce cas, le workflow CD doit construire depuis un sous-dossier :

Exemple :

```
docker build -t "$IMAGE" ./uc_02
```

➡ Tu peux gérer ça avec une variable GitHub par environnement :

- DBT_PROJECT_DIR=uc_02

Et dans le workflow :

```
docker build -t "$IMAGE" "${{ vars.DBT_PROJECT_DIR }}"
```

Si tu veux, je te fournis la variante `cd.yml` pour sélectionner le projet dbt à déployer.

10) Checklist

- dbt installé (`dbt --version`)
- projet dbt créé (`<DBT_PROJECT>/dbt_project.yml`)
- `profiles/profiles.yml` présent
- `Dockerfile` + `entrypoint.sh` présents
- variables Cloud Run Job : `PROJECT_ID/DATASET/BQ_LOCATION/DBT_CMD`
- SA runtime Cloud Run Job a les droits BigQuery

3) Option importante : "multi-projets dbt" dans le CD

Comme tu veux "configurer plusieurs projet dbt : DBT_PROJECT", la CI/CD doit savoir **quel dossier builder**.

Reco : ajouter une variable GitHub par environment, par exemple :

- `DBT_PROJECT_DIR` = `uc_02` (ou `uc_01`, etc.)

Puis modifie les steps build dans `cd.yml` :

```
```yaml
- name: Build & Push image
 run: |
 IMAGE="{{ vars.GCP_REGION }}-docker.pkg.dev/{{ vars.GCP_PROJECT_ID }}/{{ vars.GAR_REPO }}/dbt:${{ GITHUB_SHA }}"
 docker build -t "$IMAGE" "{{ vars.DBT_PROJECT_DIR }}"
 docker push "$IMAGE"
 echo "IMAGE=$IMAGE" >> $GITHUB_ENV
```

Et tu ajoutes `DBT_PROJECT_DIR` dans tes variables d'environnement GitHub.

Si tu veux, je peux te donner la version complète du `cd.yml` avec `DBT_PROJECT_DIR` et aussi `DBT_TARGET` passé au job.

## Si tu me confimes

- est-ce que tu veux **un dbt project différent par environnement** (ex: `uc_dev`, `uc_rec`, `uc_prod`)  
**ou** le même projet dbt déployé sur 3 environnements ?

Je te fournis la variante finale du workflow CD (avec `DBT_PROJECT_DIR` et `DBT_TARGET`) directement.