dpf-core — CI/CD dbt vers Cloud Run Jobs (DEV/REC/PROD)

Ce dépôt déploie une image Docker contenant dbt vers **Cloud Run Jobs** sur GCP, avec une chaîne CI/CD GitHub Actions basée sur **Workload Identity Federation (OIDC)** — **sans clés JSON**.

Objectif

- 3 environnements : **DEV**, **REC**, **PROD**
- 3 projets GCP distincts (recommandé)
- Déclenchements :
  - `develop` → déploiement DEV
  - `release/*` → déploiement REC
  - tag `v*` (ex: `v1.0.0`) → déploiement PROD

Pré-requis

Outils locaux

- `gcloud` (authentifié)
- `docker`
- `gh` (GitHub CLI) — optionnel mais recommandé

Accès IAM

Vous devez pouvoir :

- créer SA / donner des rôles IAM
- créer Artifact Registry
- créer Cloud Run Job
- créer Workload Identity Pool/Provider

Paramétrage des projets / environnements

Éditer : `scripts/01_env.sh`

Vous devez définir :

- `PROJECT_ID` dev/rec/prod (3 projets)
- `DATASET` dev/rec/prod

Exemple :

- dev: `dpf-client-dev`, dataset `clients_dev`
- rec: `dpf-client-rec`, dataset `clients_rec`
- prod: `dpf-client-prod`, dataset `clients_prod`

---

1) (Optionnel) Créer les projets GCP

Si vos projets n'existent pas :

```
./scripts/02_create_project.sh dev
./scripts/02_create_project.sh rec
./scripts/02_create_project.sh prod
```

---

2) Créer les prérequis (CI/CD + Docker + Cloud Run Job) sur chaque env

```
./scripts/03_bootstrap_prereq.sh dev
./scripts/03_bootstrap_prereq.sh rec
./scripts/03_bootstrap_prereq.sh prod
```

En sortie, le script affiche les éléments importants :

- `WIF_PROVIDER` (à mettre dans GitHub secrets)
- `SA_BUILDER_EMAIL` (à mettre dans GitHub secrets)
- `GAR_REPO`, `CLOUD_RUN_JOB`, `SA_RUNNER_EMAIL`, etc.

Note PROD (sécurité)

Pour PROD, le provider WIF est configuré pour autoriser uniquement :

- repo `jcbrun/dpf-client`
- refs tags commençant par `v` (ex: `refs/tags/v1.0.0`)

---

3) Configurer GitHub Environments (dev/rec/prod)

Dans GitHub UI :

- Settings → Environments → créer : `dev`, `rec`, `prod`
- (recommandé) sur `prod`: activer **Required reviewers**

Option A — Manuellement dans GitHub UI

Pour chaque env, définir :

Secrets

- `WIF_PROVIDER` : `projects/<PROJECT_NUMBER>/locations/global/workloadIdentityPools/github-pool/providers/github-provider`
- `GCP_SA_EMAIL` : `sa-builder-<PROJECT_ID>@<PROJECT_ID>.iam.gserviceaccount.com`

Variables

- `GCP_PROJECT_ID`
- `GCP_REGION`
- `GAR_REPO`
- `CLOUD_RUN_JOB`
- `SA_RUNNER_EMAIL`
- `DATASET`
- `BQ_LOCATION`
- `DBT_CMD`

Option B — Automatique via GitHub CLI (recommandé)

Prereq: `gh auth login`

```
./scripts/05_configure_github_envs.sh jcbrun/dpf-client
```

---

4) CI/CD GitHub Actions

CI (Pull Request)

- Se déclenche sur `pull_request`
- Fait `dbt parse/compile` + `docker build`

CD (déploiement)

- `develop` → environment `dev`
- `release/*` → environment `rec`
- tag `v*` → environment `prod`

Déclencher REC

Créer une branche release :

```
git checkout -b release/1.0.0
git push -u origin release/1.0.0
```

Déclencher PROD

Créer un tag :

```
git tag v1.0.0
git push origin v1.0.0
```

Vérifier par ligne de commande

GitHub Actions

```
gh run list --repo jcbrun/dpf-client
gh run view <RUN_ID> --log-failed
```

Cloud Run Jobs

```
gcloud run jobs executions list job-dbt-<PROJECT_ID> --project
<PROJECT_ID> --region europe-west9
gcloud logs read "resource.type=cloud_run_job AND
resource.labels.job_name=job-dbt-<PROJECT_ID>" --project <PROJECT_ID> --
limit 100
```

Test local (build + push + update + execute)

```
./scripts/04_build_push_update_execute.sh dev docker
./scripts/04_build_push_update_execute.sh rec docker
./scripts/04_build_push_update_execute.sh prod cloudbuild
```

Points fréquents si ça bloque

- WIF : erreur "attribute condition" → vérifier condition + repo + ref
- BigQuery : rôle projet OK mais manque IAM au niveau dataset (fréquent)
- Cloud Run Job : vérifier SA runtime (`SA_RUNNER_EMAIL`) et les env vars `PROJECT_ID/DATASET/BQ_LOCATION`

```
---

## Ce qu'il te reste à faire (très concret)

1) **Remplacer** tes scripts par ceux ci-dessus (`scripts/`)
2) **Éditer** `scripts/01_env.sh` avec tes vrais IDs projets/datasets
3) Lancer :
```bash
./scripts/03_bootstrap_prereq.sh dev
./scripts/03_bootstrap_prereq.sh rec
./scripts/03_bootstrap_prereq.sh prod
```
```

4. Créer les **GitHub Environments** (`dev`, `rec`, `prod`)
5. (optionnel) pousser la config automatiquement :

```
./scripts/05_configure_github_envs.sh jcbrun/dpf-client
```

Si tu me donnes tes **3 Project IDs exacts** (dev/rec/prod) + si tu veux **dataset identique** (ex `clients`) ou **suffixé** (`clients_dev`…), je te renvoie une version de `01_env.sh` “prête prod” sans placeholders.