Commerçants
autrement

Automatisation du rafraîchissement d'Univers

18/03/2024

Sommaire

1. Contexte
2. Outils
3. Articulation
4. Emplacements des outils
5. Démonstration

Contexte

- Bascule Cloud Datalake et Datawarehouse terminée ⇒ Lakehouse GCP Bigquery
- Besoin d'environnements en DEV et REC à l'image de la production pour les évolutions et corrections des projets Data
- Besoin d'industrialiser la création et l'alimentation en données de ces environnements hors production
- Côté Décisionnel, un outil existait sur Oracle à remplacer pour offrir un service similaire
- Point d'attention FINOPS : le Cloud permet de créer facilement de nouveaux environnements, de stocker des volumes de données importants, mais ce n'est pas gratuit

Contexte - Univers

Un **univers** = Environnement complet correspondant à la somme des **données** et des **traitements**

Univers = **Données** + **Traitements**

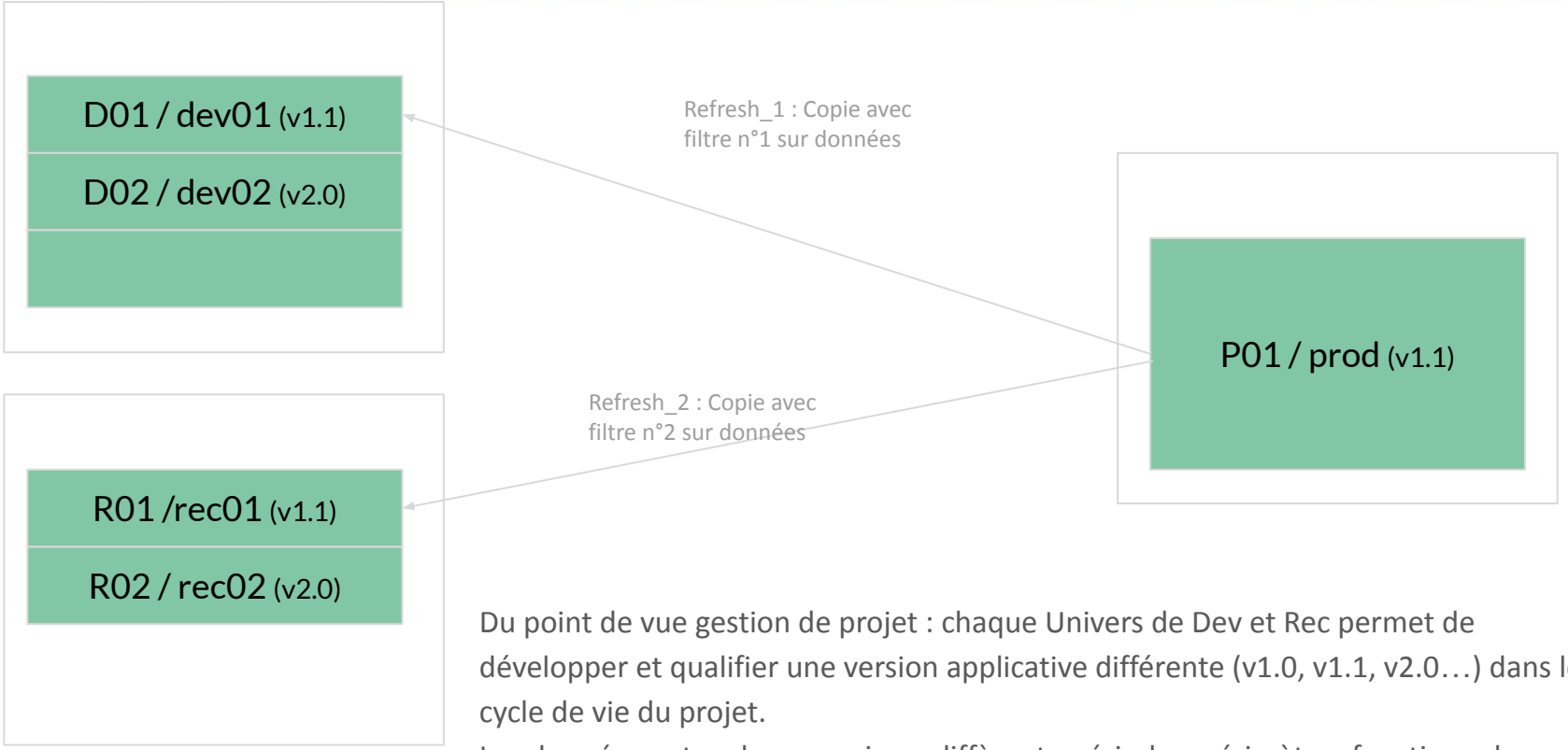
L'**univers** de **référence** est l'environnement de production, qui sera décliné en plusieurs versions en DEV et REC

L'**univers** est composé de sous ensembles applicatifs et/ou fonctionnels **cohérents**

Un **rafraîchissement Data** : Copie des données depuis l'environnement de production

⇒ vers un-sous environnement de développement ou de recette

Contexte - Rafraîchissement d'univers



Du point de vue gestion de projet : chaque Univers de Dev et Rec permet de développer et qualifier une version applicative différente (v1.0, v1.1, v2.0…) dans le cycle de vie du projet.

Les données entre chaque univers diffèrent : périodes, périmètres fonctionnels

IRIS

Contexte - Qui fait quoi

- Une boîte à outils développée par l'équipe Socle Data
- Composée d'un ensemble d'outils permettant une alimentation partielle ou complète d'un univers
- Proposés aux équipes projets pour être autonome sur leur besoin de rafraîchissement
- Formation, accompagnement et support par l'équipe Socle
- Proposition d'un arbre de décision pour guider et sensibiliser sur le coût les réplications (FINOPS)

		Réplication données			**Réplication structure + données**			**Réplication complète**
Besoin	**Equipe Acteur**	Table	Dataset	Projet	Table	Dataset	Projet	Tous les datasets / tables
Refresh partiel données	Projet	X						
Refresh partiel données	Projet		X					
Refresh partiel données	Projet			X				
Refresh partiel structure+données	Projet				X			
Refresh partiel structure+données	Projet					X		
Refresh partiel structure+données	Projet						X	
Refresh complet d'un Univers	Socle / Projet							X

Contexte - FINOPS

Le service BigQuery est facturé sur 2 métriques :

- le **stockage de données** dans les tables / vues matérialisées / snapshots
- **à l'usage** lors de la consultation des données (scan de données)

Au 1er Janvier 2024, parcourir **1 To de données** en Europe coûte environ **7€**

Optimiser les coûts d'usage BigQuery lors des rafraîchissements

- limiter le nombre d'Univers
- limiter le nombre de tables
- limiter le volume de données par tables ⇒ filtre
- limiter la fréquence de rafraîchissement

IRIS

Commerçants autrement

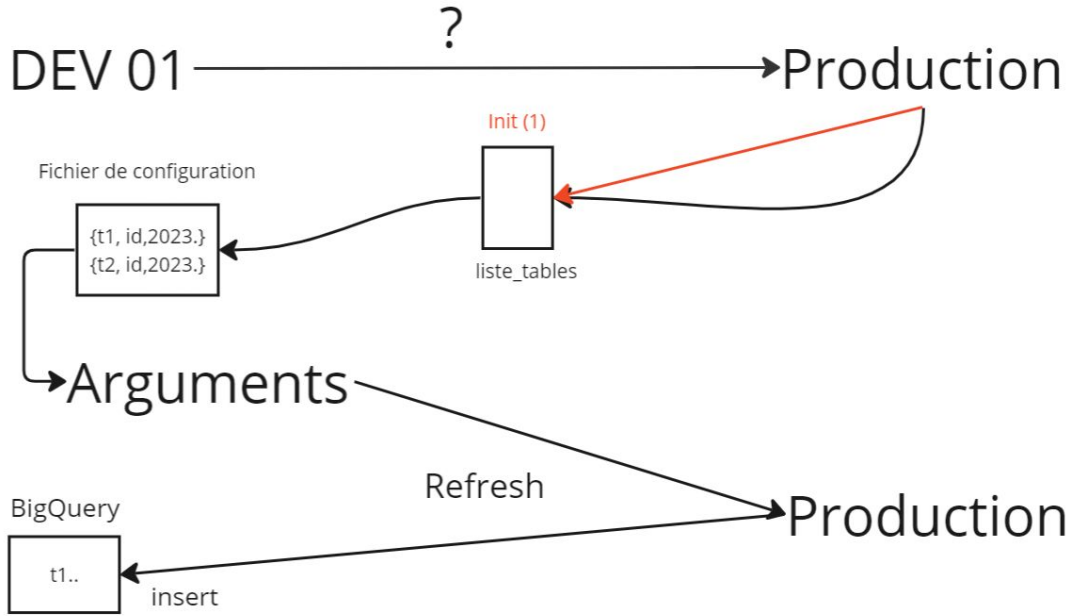
Contexte - Rafraîchissement d'univers

Voici les différentes étapes du rafraîchissement :

1. Interrogation de l'environnement de Production afin de récupérer ses tables et de créer un fichier de configuration

2. Nettoyage de l'environnement de développement

3. Rafraîchissement des environnements BigQuery de dev et recette à partir de celui de production



Outils - refreshSubEnv

L'outil refreshSubEnv permet le **rafraîchissement** d'un univers.

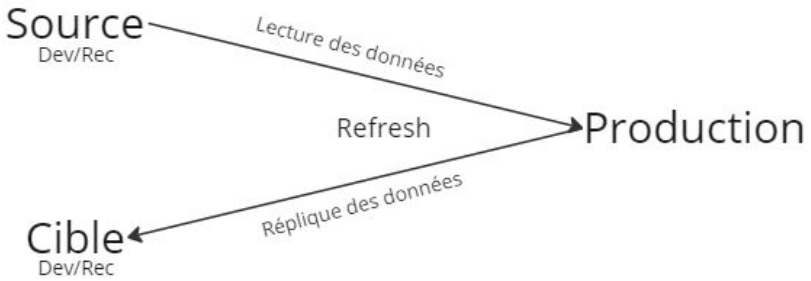
Il permet la **réplique de données** (dataset -tables) depuis un environnement **source** vers un environnement **cible**.

Cet outil prend en arguments

- l'identifiant du projet
- le sous-environnement cible (dev 01 / rec 02)
- en option : dataset / table / période à rafraîchir

Voici quelques exemples d'usages possibles :

- Refresh de tous les datasets du projet GCP EDH Ingestion
- Refresh de toutes les tables d'un dataset particulier du projet GCP EDH Ingestion
- Refresh de toutes les tables partitionnées d'un dataset du projet GCP SIDE filtrées sur une période



Outils - prepareRefreshDatasets

Dans une démarche de **création ou réinitialisation d'un univers** :

⇒ l'outil permet de supprimer tous les objets (Tables, vues, procédures stockées…) des datasets d'un univers afin de préparer le rafraîchissement ISO Production

Il prend en arguments :

- l'environnement cible du refresh (l'univers),
- le numéro de cet environnement
- l'authentification au projet GCP Source (Production)
- l'authentification au projet GCP cible (Dev ou Rec)
- Une option Dryrun pour tester avant suppression définitive de l'univers cible

Outils - createRefreshConfigFile

Cet outil crée un **fichier de configuration** à partir de la liste des tables de l'environnement de production.

Ce fichier contient plusieurs informations pour chaque table :

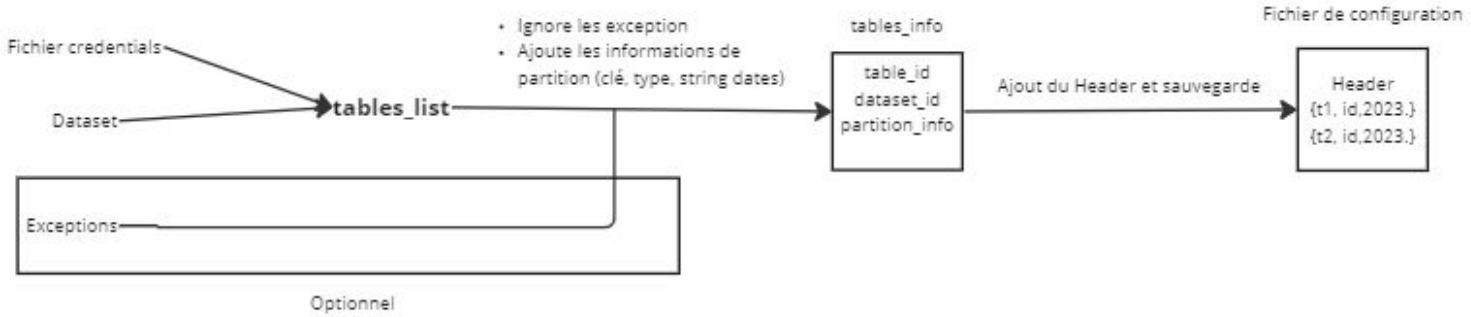
- son identifiant,
- l'identifiant de son dataset
- si elle est partitionnée, on indique sa clé de partition et le type de données et des chaînes de caractères date_debut et date_fin associées au type

Ce fichier peut être mis à jour s'il y a des nouvelles tables dans l'environnement de production.
Il est également possible d'indiquer des tables qu'on ne veut pas avoir dans le fichier de configuration.

Outils - createRefreshConfigFile

L'outil prend en arguments

- un dataset source,
- l'authentification au projet GCP Source (Production)
- le nom du répertoire de configuration, celui du fichier de configuration souhaité
- en option, une liste d'exclusion de tables à ignorer



Outils - createRefreshConfigFile

Ce fichier de configuration contient également une partie “**Header**” avec des variables début et fin.

L’entête du fichier de configuration doit donc être personnalisé pour indiquer la période à extraire en production pour rafraîchir l'univers cible

```
---HEADER---
debut_DAY=
fin_DAY=
debut_NUM=
fin_NUM=
-----------
```

Exemple de personnalisation :

debut_DAY = 2023-01-01
fin_DAY = 2023-03-31

Outils - autoRefresh

Cet outil parcourt le fichier de configuration précédemment créé et personnalisé
puis crée un **fichier exécutable bash** comprenant

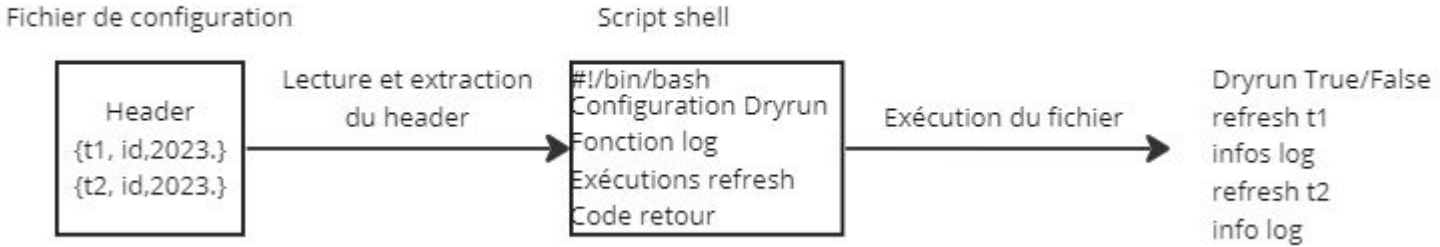
- des lignes d'exécution de l'outil refreshSubEnv pour chaque table du dataset, avec les périodes de données à rafraîchir
- une gestion des erreurs Bash
- une gestion de log pour tracer le rafraîchissement

Le script bash est nommé avec un rappel du dataset et les périodes traitées

Outils - autoRefresh

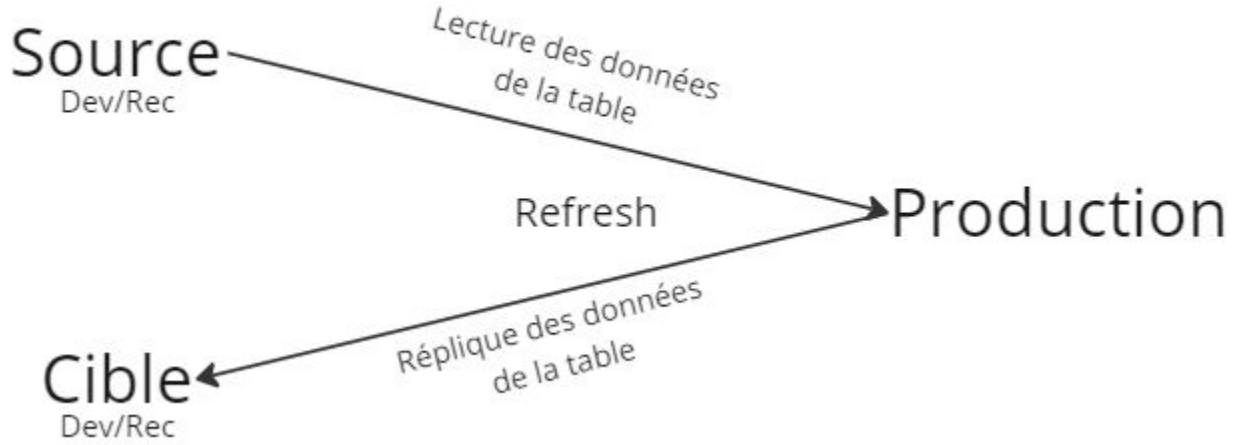
L'outil prend en arguments suivants :

- l'identifiant du projet,
- l'environnement cible
- le chemin du fichier de configuration
- la chemin cible du fichier d'exécution (shell) généré
- une option appelée Dryrun permet de tester l'exécution du shell généré dans exécution des commandes de copie des données



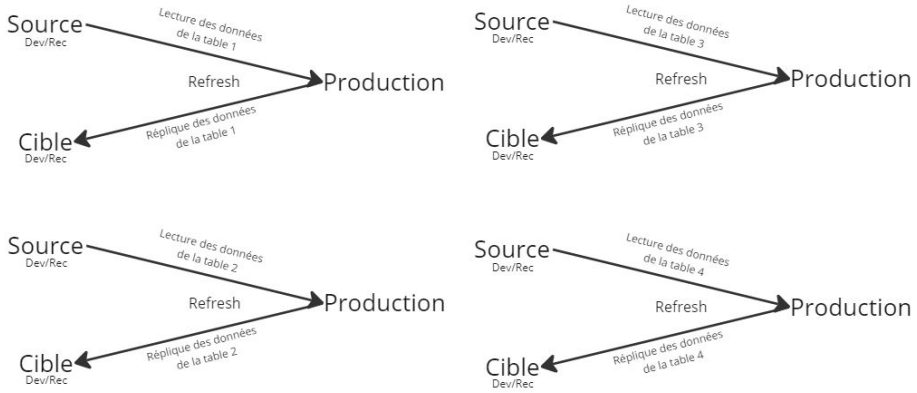
Articulation - Cas 1 : refresh d'une table

Voici le fonctionnement de refreshSubEnv dans le cas du rafraîchissement d'une seule table.



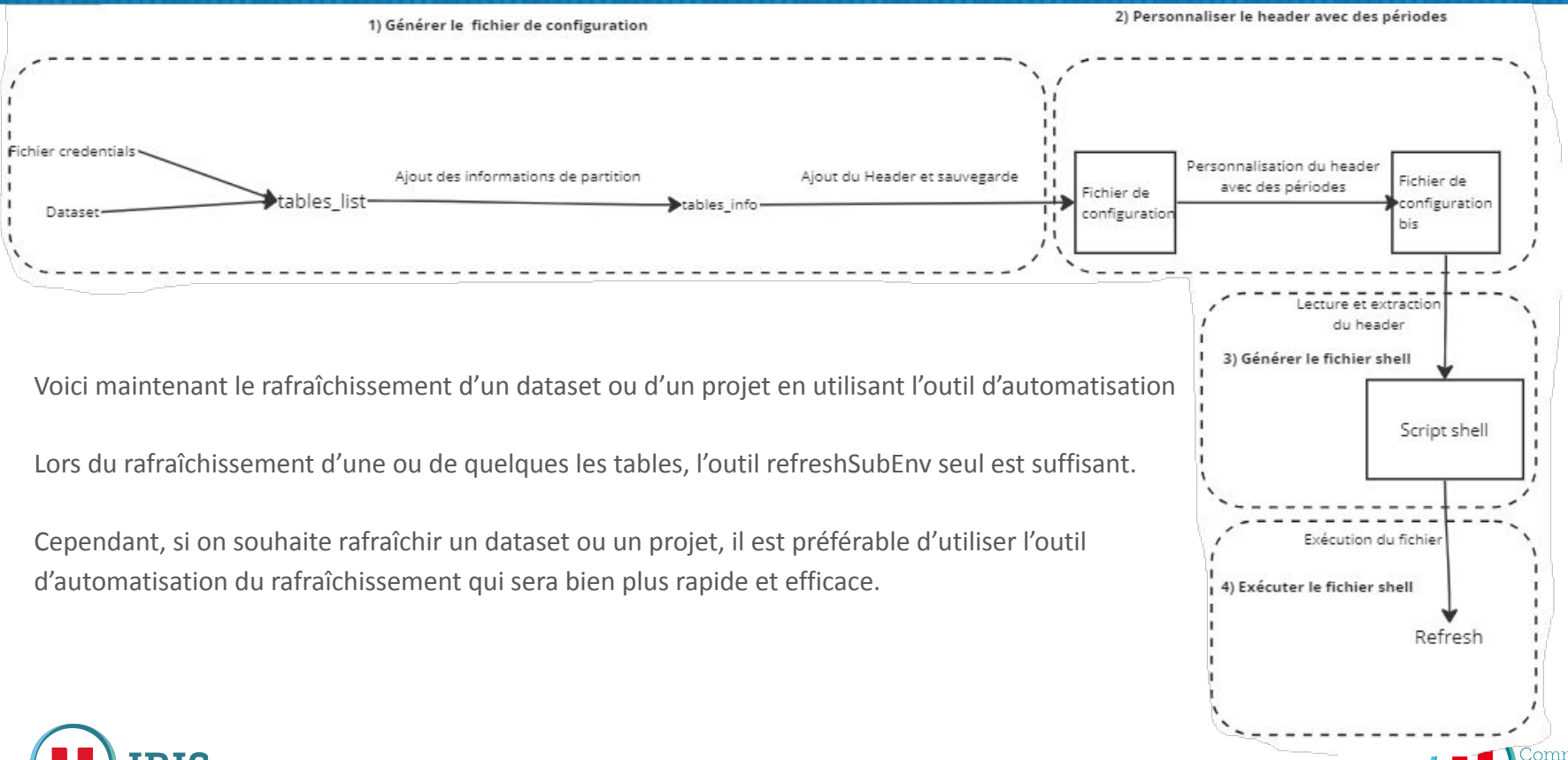
Articulation - Cas 2 : refresh d'un dataset / projet

Voici maintenant le fonctionnement général de l'outil refreshSubEnv dans le cas du refresh d'un dataset ou d'un projet.



Il existe une option appelée schema_only afin de n'avoir que la structure du dataset ou du projet.

Articulation - Cas 3 : refresh personnalisé



Voici maintenant le rafraîchissement d'un dataset ou d'un projet en utilisant l'outil d'automatisation

Lors du rafraîchissement d'une ou de quelques les tables, l'outil refreshSubEnv seul est suffisant.

Cependant, si on souhaite rafraîchir un dataset ou un projet, il est préférable d'utiliser l'outil d'automatisation du rafraîchissement qui sera bien plus rapide et efficace.

Où sont installés les outils (Git + Serveur de dev)

Des outils archivés dans Github :
https://github.com/ugieiris/edh/tree/feature/refresh_subenv_lot1/tools/esg/esg_refreshSubEnv
Déployés en cible sur les Edges GCP de **DEV** et **REC** et opérationnel pour tous les projets GCP

⇒ Objectif : Basculer les outils sur un repo git dédié avec un CI/CD python (solution proposée par Valentin)

Ils sont accessibles sur le serveur edge de développement pour dev et edge de recette pour recette via les commandes suivantes :
Connexion avec le compte de service svc_edh_deploy
cd /u01/app/esg_refresh_univers
source ./venv/bin/activate

```
Mar 13 16:18 bash
Feb 28 09:40 config
Feb 21 14:43 createRefreshConfigFile.py -> /u01/app/edh/edh-current/tools/esg/esg_refreshSubEnv/extract_list_tables.py
Mar 13 10:32 logs
Mar 11 16:16 outputs
Feb 12 14:48 prepareRefreshDatasets.py -> /u01/app/edh/edh-current/tools/esg/esg_refreshSubEnv/prepare_refresh_datasets.py
Feb 12 14:47 README.md -> /u01/app/edh/edh-current/tools/esg/esg_refreshSubEnv/README.md
Feb 12 14:47 refreshSubEnv.py -> /u01/app/edh/edh-current/tools/esg/esg_refreshSubEnv/refreshSubEnv.py
Feb 21 14:51 venv
```

Démonstration - Mode “DryRun”

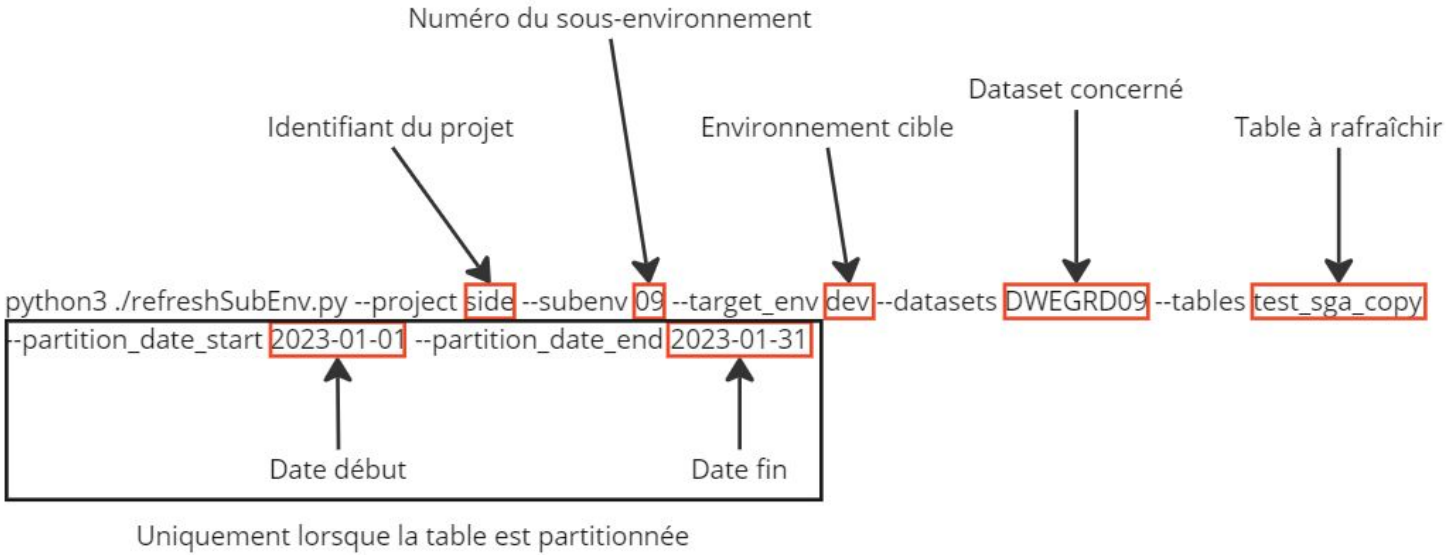
Il existe un mode Dryrun qui est **activé par défaut**.
Il faut explicitement indiquer ‘False’ après la commande d’exécution afin de le désactiver.

Ce mode est un **test à blanc**, il permet d’afficher les lignes générées afin que le développeur puisse avoir un aperçu du résultat sans réelle exécution.
En effet, l’exécution du rafraîchissement peut avoir des **conséquences** non désirées et a également un **coût**.

```
(venv) [svc_edh_deploy-dev@g0016dedge0 esg_refresh_univers]$ . ./otherTest.sh
/DryRun=True
python3 ./refreshSubEnv.py --project side --subenv 09 --target_env dev --schema_only True --datasets DWEGRD09 --tables DE_CLIENT_CENTRALE_COMPLEMENT
python3 ./refreshSubEnv.py --project side --subenv 09 --target_env dev --schema_only True --datasets DWEGRD09 --tables DE_PARAMETRES_GLOBAUX
python3 ./refreshSubEnv.py --project side --subenv 09 --target_env dev --schema_only True --datasets DWEGRD09 --tables test_sga --partition_date_start 2024-01-01 --partition_date_end 2024-01-31
python3 ./refreshSubEnv.py --project side --subenv 09 --target_env dev --schema_only True --datasets DWEGRD09 --tables test_sga_copy --partition_date_start 2024-01-01 --partition_date_end 2024-01-31
```

Démonstration 1 - Exécution de refreshSubEnv

Cet outil réplique des données (dataset -tables) depuis un environnement source vers un environnement cible.



Next

Actuellement

- une première version de l'outillage de rafraîchissement est terminée
- il permet d'automatiser le rafraîchissement d'un univers en dev ou rec
- il est disponible pour réaliser des refresh unitaire ou d'ensemble de tables : à vous de jouer !!
- un backlog a été créé pour recueillir les retours d'utilisation : Outil refresh Data - Retour d'utilisation

A	B	C	D	E	F	G	H
Qui	Quand	Outil refresh officiellement présenté ?	Cas d'usage / Projet	Fonctionnalité testée	Statut	Amélioration / anomalie rencontrée	Revue PO
Stéphane GAUDIN	16/10/2023	Oui	Validation Socle Data	Refresh des données d'une table SIDE	Remarque	Afficher nombre de lignes répliquées	Oui

Rappel FINOPS

- La création d'un nouvel univers doit être validé par les managers de la ligne Data

Démonstration 2 - Exécution de autoRefresh

Annexe - Structure du fichier d'exécution

```
#!/bin/bash

# Détecte si le DryRun est désactivé ou non
if [ "$1" = "False" ]; then
    vDryRun=False
    echo "Mode Dryrun désactivé : Rafraîchissement des tables"
else
    vDryRun=True
    echo "Mode Dryrun activé : Test à blanc, affiche uniquement les informations"
fi

# Activation de l'environnement virtuel
source ./venv/bin/activate
```

```
# Teste si mode dryrun est actif ou non : si oui => affichage seulement de la commande sans exécuter le refresh
if [ "$vDryRun" = "True" ]; then
    echo "python3 ./refreshSubEnv.py --project side --subenv 09 --target_env dev --datasets DWEGRD09 --tables test_sga --partition_date_start 2023-01-01 --partition_date_end 2023-01-31 "
else
    python3 ./refreshSubEnv.py --project side --subenv 09 --target_env dev --datasets DWEGRD09 --tables test_sga --partition_date_start 2023-01-01 --partition_date_end 2023-01-31
    if [ $? -ne 0 ]; then
        log "ERR" "Erreur de connexion au projet side, au sous-environnement 09 ou à l'environnement cible dev"
        error=$((error + 1))
    else
        log "INFO" "Connexion au projet side, le sous-environnement est 09 et l'environnement cible est dev"
    fi
fi
```

```
if [ $error -eq 0 ]; then
    # Code retour signifiant que tous les rafraîchissements ont fonctionné
    exit 0
else
     # Code retour signifiant la présence d'erreurs dans un ou plusieurs rafraîchissements
    echo "Nombre d'erreurs : $error"
    exit 3
fi
```

Annexe - Refresh personnalisé avec détails