Test technique

I - Objectif

L'objectif de test est de concevoir un flux passant par GCP, allant des données source à un rapport pour les métiers. La conception du flux se concrétisera par un schéma d'architecture et une partie de code obligatoire (ainsi que quelques bonus).

Une réflexion devra être menée sur certains sujets qui seront approfondis lors du debrief.

Les fichiers à nous retourner :

- Ce document word complété (questions et commentaires si vous le souhaitez en rouge gras)
- Un dossier zippé contenant un projet python, le schéma d'architecture et les documents que vous jugerez pertinents – si les données de sortie sont trop grosses, écrivez une seule ligne

II - Mise en place de l'environnement

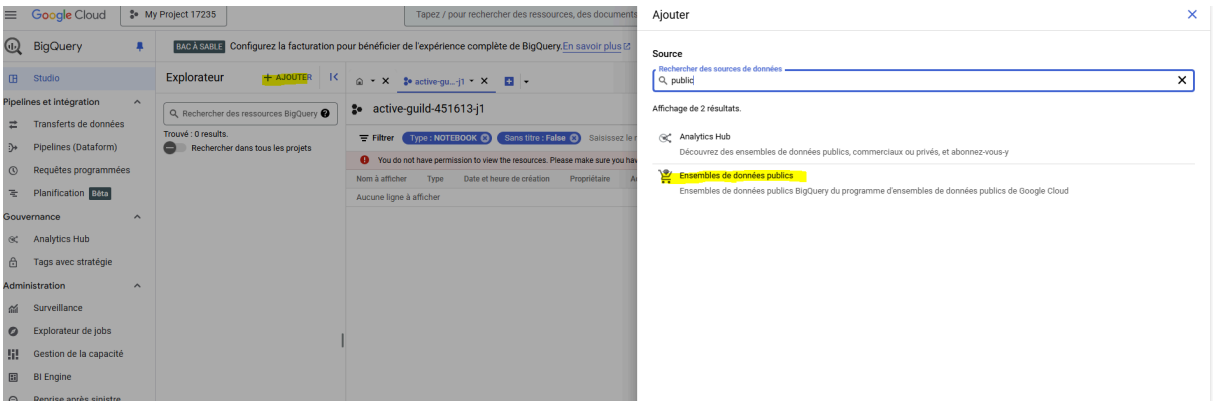
Lors de ce test, vous allez avoir besoin :

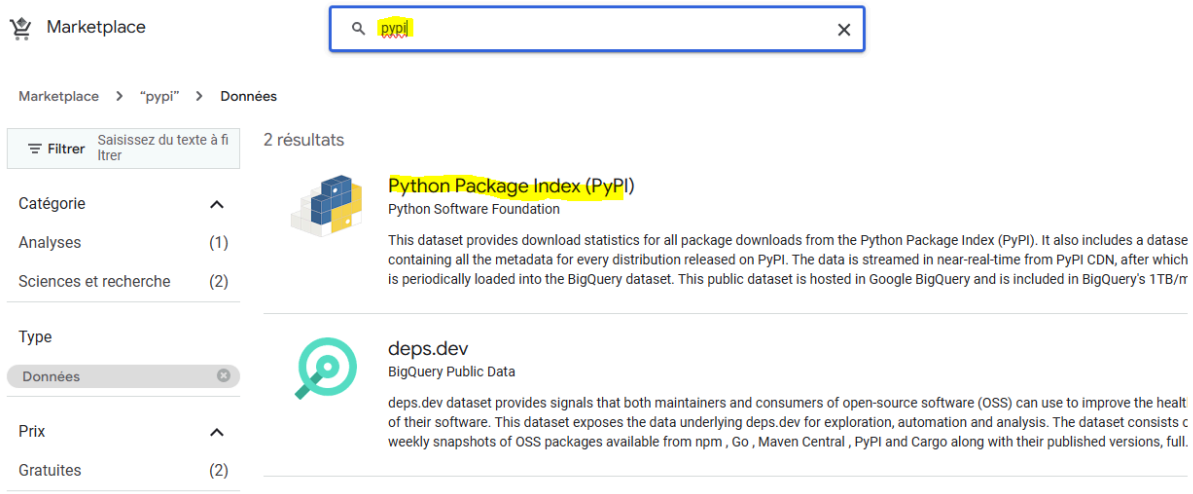
- D'installer Python (3.12 de préférence)
- De créer un projet dans le bac à sable BigQuery (gratuit, avec des quotas qui ne seront normalement pas atteints lors de ce test)
- D'installer g cloud et de le configurer avec vos credentials locaux
- D'utiliser les API Python clientes Google (BQ : google-cloud-bigquery)

Nous détaillons les étapes (2) et (3)

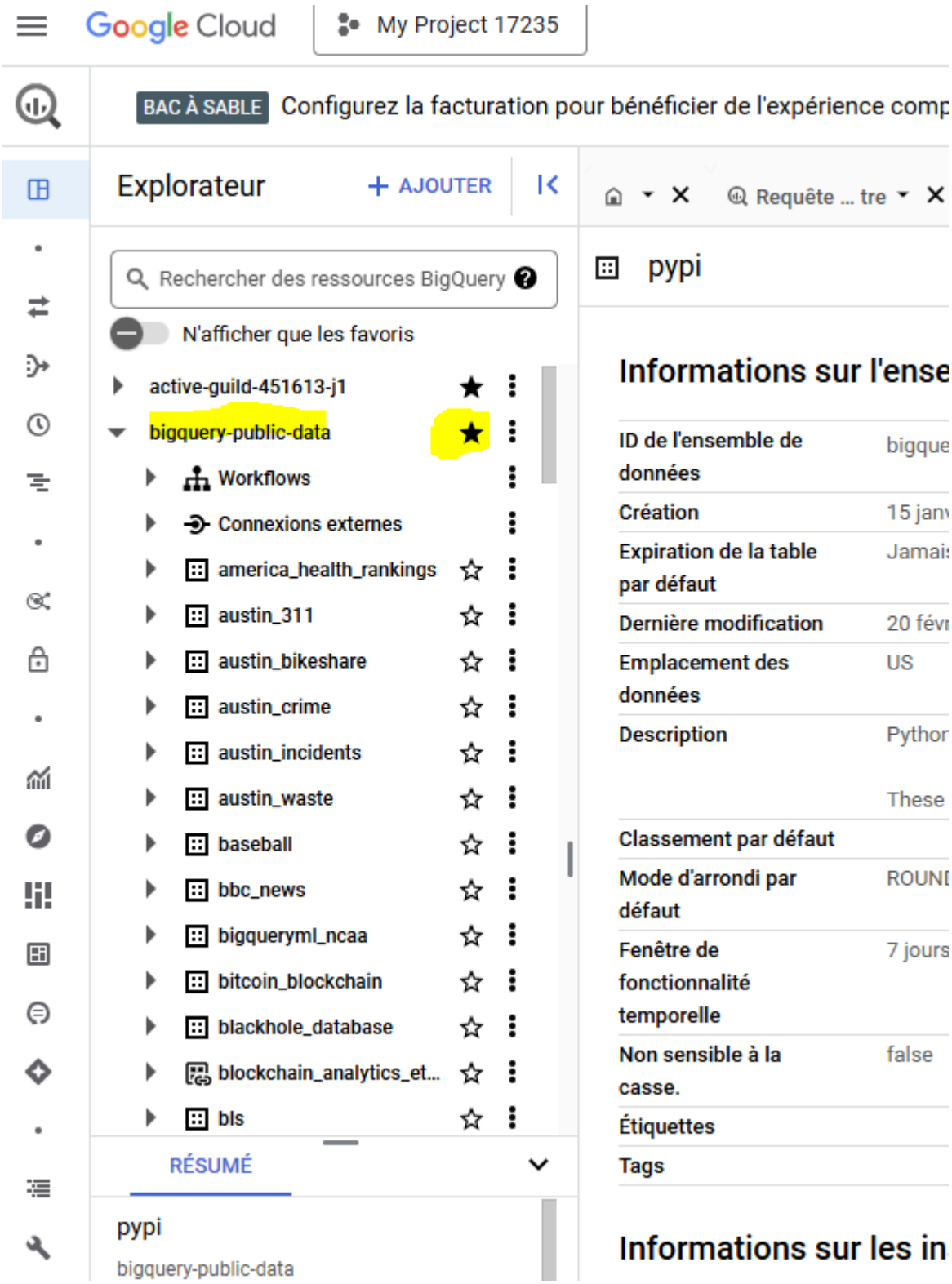
1) Configuration du bac à sable BigQuery

- Créer un compte google si ce n'est pas déjà fait (ou uniquement pour ce test) : Google Compte
- Suivre les instructions de ce lien pour parvenir créer un projet dans le bac à sable BigQuery (gratuit sans période limitée) : https://cloud.google.com/bigquery/docs/sandbox?hl=fr
- Suivre ces instructions pour accéder aux données publiques : Ensembles de données publics de BigQuery | Google Cloud





Vous pouvez épingler les données publiques après avoir accédé aux données pour vous faciliter la tâche par la suite.



- Nous allons utiliser les données du dataset : **bigquery-public-data.pypi** (PyPi dans l'explorateur de données)

2) Configuration de l'outil gcloud CLI

L'outil gcloud CLI permet d'interagir avec GCP, en particulier avec BigQuery. Il permet aussi de stocker ses identifiants localement.

- Installez gcloud CLI en suivant Installer gcloud CLI | Google Cloud CLI Documentation
- Lancez la commande **gcloud init**
- Stockez vos identifiants via **gcloud auth application-default login**

II - Test – Dashboard des téléchargement PyPi

1) Besoin métier

La direction IT a besoin d'un dashboard permettant de monitorer les téléchargements PyPi sur une période donnée (pour l'exemple, nous allons dire 15 jours).

Les KPIs suivants sont attendus (somme et moyenne) :

- Nombre de téléchargements
- Taille des paquets téléchargés
- Nombre de dépendances

Le dashboard permet de voir ces KPIs :

- Par pays (ou pas de pays sélectionné)
- Par projet PyPi (ou pas de projet sélectionné)
- Par version d'un projet PyPi (ou pas de version sélectionnée)

Les agrégats peuvent se faire à l'heure jusqu'à la semaine.

Une dernière analyse est demandée :

- Un graphe doit présenter le poids relatif de chaque jour de la semaine pour chaque métrique (sélecteur, superposition ou autre) - Si vous n'y arrivez pas, au moins le nombre de téléchargements
- Un graphe doit présenter le nombre moyen de téléchargements par heure

2) Besoin interne

Pour d'autres analyses, un fichier journalier doit être écrit et mis à disposition pour d'autres pipelines. Sous le dossier data/ pour l'exemple.

Le flux doit être orchestré sur Google Cloud Composer/Airflow et correctement monitoré.

3) Tâches

- Proposez un schéma d'architecture répondant aux questions et impératifs suivants (en plus de ceux exprimés dans les besoins métier et interne) :
    - Les données partent de BigQuery :
      table bigquery-public-data.pypi.file_downloads (téléchargements) +
      bigquery-public-data.pypi.distribution_metadata (metadonnées des paquets)
    - **Outre la requête originale pour acquérir les données, le reste des opérations est fait en Python (via l'api cliente entre autres et d'autres libraires si vous le jugez nécessaire)**
    - Le flux doit être conçu pour tourner périodiquement
    - Décrivez également comment le code est déployé (au moins les outils pour le faire)
    - Vous pouvez expliquer des étapes textuellement
    - Il est sous-entendu que nous restons exclusivement sur GCP en termes d'outils en ligne (vous pourrez proposer d'autres solutions en debrief si vous le souhaitez)

- Quelques questions préliminaires (répondre sur ce document en dessous de chaque question **en rouge et en gras**) :
    - Auriez-vous des questions à poser au métier ou en interne pour adapter votre solution ?
        - **Avons-nous besoin des données historiques au-delà des 15 jours ?**
        - **Quelle fréquence de rafraîchissement des données pour les dashboards ?**
        - **Avons-nous une liste de package précis à suivre ?**
        - **Avons-nous plus de filtres spécifique ? Car c'est trop de données et cela nécessite beaucoup de compute.**
    - Nous nous sommes authentifiés via gcloud CLI, comment devrait-on faire dans le cas d'un flux de production orchestré ?
        - **Utilisation d'un compte de services**
    - Que se passe-t-il si le code ne tourne pas pendant une journée ?
        - **Si le flux ne tourne pas pendant une journée les données des dashboard ne seront plus fraîches avec un retard d'une journée.**
        - **Un système d'alerte devra être mis en place pour notifier les DE**
    - Comment le gérez-vous ?

        - **Il faut mettre en place des alertes (mail, teams, slack) pour notifier les DE que la pipeline est down. Ceci pourra ce faire via des services GCP (exemple pubsub)**

- Codez ce qui est détaillé dans la partie suivante
    - Ce code doit être prêt pour la production, ce n'est pas un simple script oneshot
    - N'hésitez pas à commenter ce que vous faites, les "bonnes" pratiques de commentaire ne seront pas jugées

- En fin de test, des thèmes de réflexion seront listés, vous n'avez pas à répondre ici (certains éléments peuvent être ajoutés à l'architecture et détaillés en commentaire ceci dit). Ces éléments seront développés lors du debrief

4) Code obligatoire

Nous rappelons les tables source :

- bigquery-public-data.pypi.file_downloads (téléchargements)
- bigquery-public-data.pypi.distribution_metadata (metadonnées des paquets)

Les contraintes :

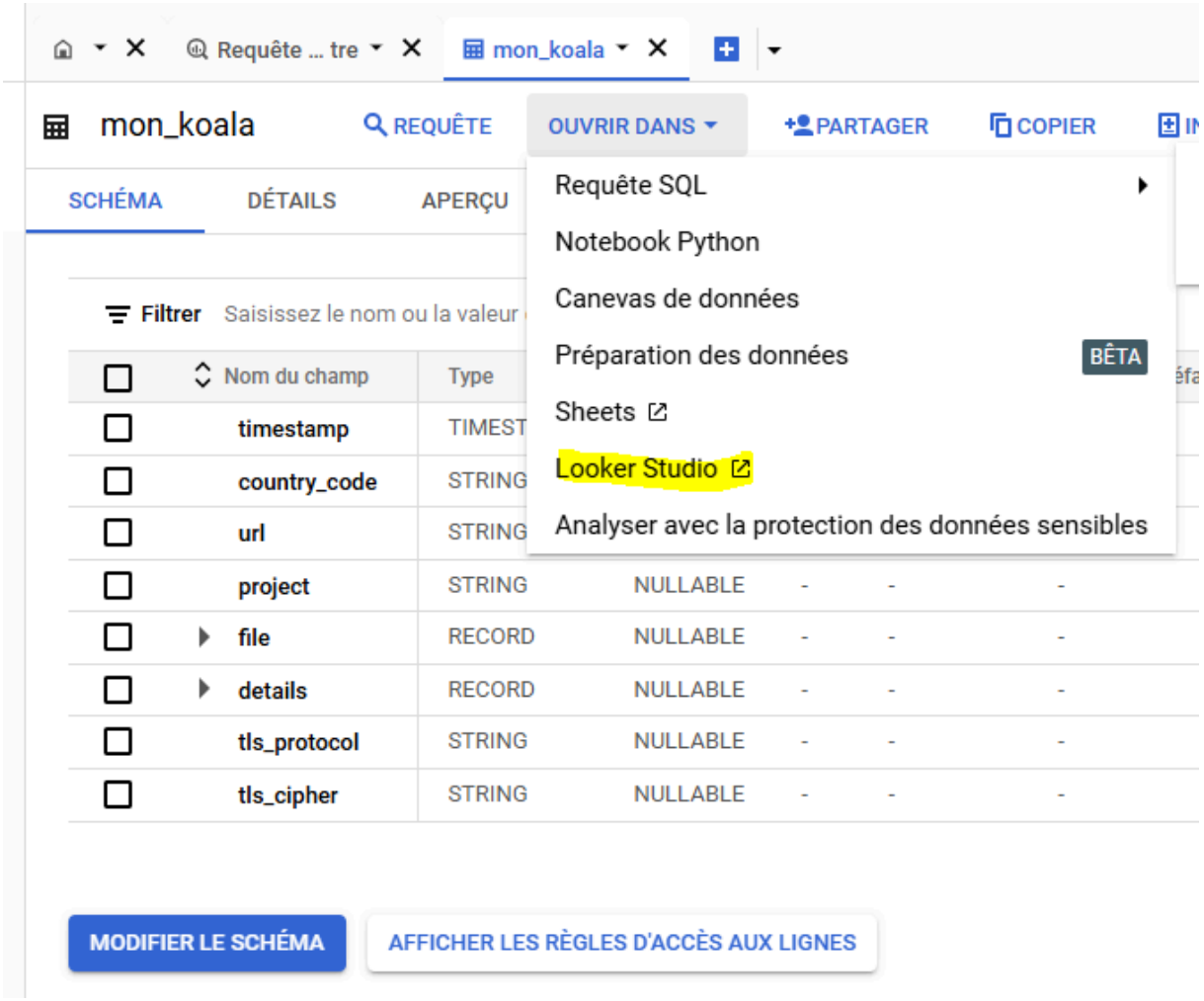
- Le code doit être réalisé en Python (API BQ Python)
- Il se matérialise par un projet Python avec un seul point d'entrée
- Il peut être créé avec poetry ou être un projet simple avec un main.py et un fichier de dépendances (i.e. soit poetry, soit les outils basiques standards pip, setuptools, venv)
- Nous rappelons que le code doit ressembler à un code de production, il doit donc être bien organisé avec des bonnes pratiques associées

Ce qu'il faut faire :

- Produire une/des requête(s) permettant d'acquérir les données de la table cible et d'alimenter le dashboard exprimé par le besoin (à vous de déterminer le niveau d'agrégation etc. selon le besoin)

- Produire un fichier par jour dans le dossier data/ de votre projet (à vous de choisir le type de fichier pertinent)
- Produire une vue ou une table bigquery qui extrait le dernier téléchargement par pays, par projet et par jour (mêmes bornes temporelles que le besoin)
- Produire une vue ou une table bigquery qui donne pour chaque ligne de la table des téléchargements :
    - La durée depuis le téléchargement précédent du même projet
    - Un booléen qui indique si le fichier est gzippé ou non (sur la base des informations disponibles, on ne vérifiera évidemment pas s'il l'est physiquement)
- Produisez un dashboard Looker avec trois ou quatre graphes (deux sont obligatoires) décrits dans le besoin
    - Si vous n'arrivez pas à utiliser Looker, comme le poste n'est pas celui d'un Data Analyst, vous pouvez produire des graphes dans un notebook ou autre outil de votre choix (même Excel). Cependant un Looker (ou outil gratuit similaire) serait préférable.
    - Nous ne jugerons pas l'esthétique des graphes ou du dashboard
    - En dernier recours, vous pouvez au moins agréger les données pour servir un graphe ad-hoc
    - L'objectif ici est de bien valider que vous avez servi le besoin en l'ayant bien compris et avec les données qu'il faut

Voici comment accéder à Looker :



Si vous rencontrez des problèmes d'API, vous pouvez exceptionnellement télécharger les données à la main et les charger à la main (mais une tentative de code doit être présente).

Si vraiment vous n'arrivez pas à accéder aux données, le test peut être fait en local, nous vous transmettrons les fichiers de base (là encore une tentative de code doit être présente et il faudra bien expliciter ce qui devrait être fait).

5) Code bonus

- Un Dockerfile qui crée une image exécutable de votre projet
- Un fichier de DAG Airflow **dag.py** illustratif (nous voulons les étapes principales, pas besoin que cela fonctionne vraiment, du pseudo code ou des commentaires peuvent être ajoutés pour les liens avec la config)

6) Sujets de réflexion et questions

Nous rappelons que ces questions (du moins une partie d'entre elles) seront discutées en face à face (ou sont peut-être déjà explicitées dans votre architecture), il n'y a pas besoin d'y répondre ici (vous pouvez mettre quelques éléments si vous le souhaitez).

- Comment organiseriez-vous BigQuery pour un projet de ce genre qui va de l'ingestion ou à une dashboard ?
- Où écririez-vous les fichiers si vous aviez un compte GCP payant dans le cadre d'un vrai projet de production ? Cherchez la méthode pour le faire et notez la si pertinent (les fonctions/librairies utilisées)
- Comment avez-vous optimisé (ou comment vous optimiseriez) les coûts dans votre travail ?
- Comment vous assuriez-vous de la qualité des données de sortie ? Notez quelques indicateurs et comment faire (au moins la méthode générique)
- Comment feriez-vous pour traiter le même problème avec des millions de ligne par jour ? Cela peut aller de petites optimisations à de grandes
- Quelles sont les possibilités pour mettre à disposition vos données :
    - Pour des services IT ?
    - Pour les utilisateurs ?
- Imaginez le même problème avec l'accès à un topic Kafka listant les créations de paquet et les téléchargements, en d'autres termes les données des deux tables manipulées (qui ne seraient donc pas disponibles sur BigQuery). Comment feriez-vous pour aboutir au même résultat ?
- Auriez-vous une autre résolution de ce problème qui n'inclue pas le code obligatoire ?

Des solutions fonctionnelles et technologiques sont attendues.