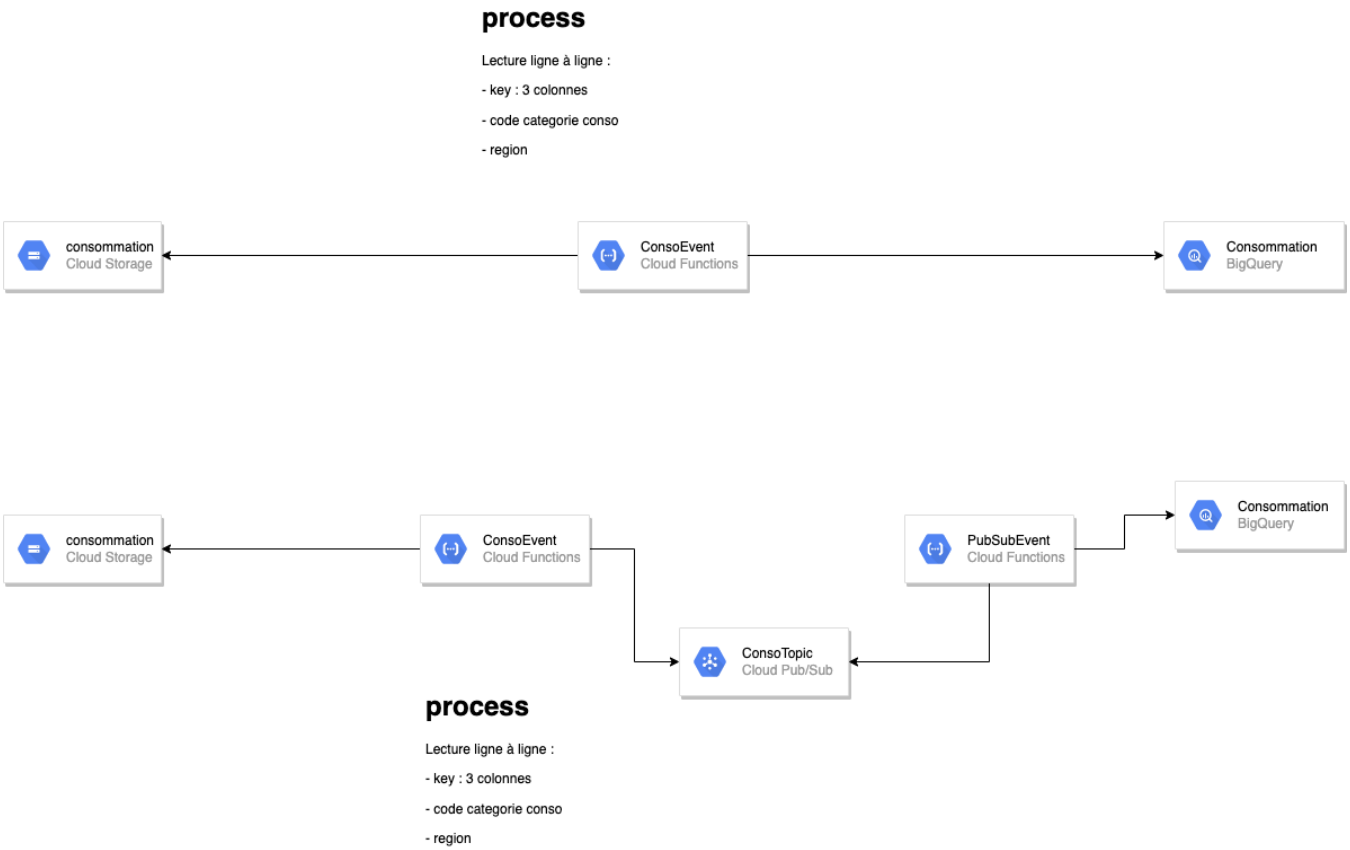
Usecases

Stockage froid, évènement, code serverless et stockage à chaud

La source de données se trouve ici : https://www.data.gouv.fr/fr/datasets/consommation-annuelle-delectricite-et-gaz-par-region-et-par-code-naf/

1. Stocker le fichier sur Cloud storage dans un bucket (nous en avons besoin pour lecture occasionnelle à régulière, pas de redondance).
2. Créer la possibilité d'avoir un évènement lors de l'ajout/création d'un fichier (à la fin de l'écriture fichier)
3. Attacher une cloud function à cet évènement :
    a. Elle doit lire le fichier ligne à ligne
    b. Construire un objet json avec une clé composée et le reste des colonnes.
    c. Déverser dans BigQuery
    d. Visualiser sous DataStudio
4. Faire une alternative :
    a. La structure Json est envoyéd vers la solution Pub/Sub
    b. Une autre function est abonnée à cette source de donnée (notion de subscription et de topic).
    c. La cloud function envoie le résultat dans big query (cf 3a)
    d. Visualiser sous Datastudio
5. Dans les premiers items, vous pouvez déployer "manuellement" avec la CLI Gcloud.
6. Ensuite, refaire cette partie de développement en utilisant : Cloud source repositories et Cloud Build si besoin (le but est d'automatiser vos déploiements).
7. Pensez à logguer vos opérations dans le code de manière intelligente.
8. Visualisez les métriques d'exécution pour vous familiariser avec la solution GCP de monitoring et de logging.



Stockage froid, pipeline complexe et stockage à chaud

En partant de la source de données précédente, ajouter la récupération des codes NAF : https://www.insee.fr/fr/information/2120875

1. Créer un fichier csv propre avec ces codes NAF.
2. Créer un pipeline Dataflow mode Batch (utiliser d'abord le local runner avant de déployer sur GCP !) :
    a. Lisez la source de données des consommations d'énergie.
    b. Une première branche :

i. prend le fichier en entrée
ii. créé un objet JSON par ligne
iii. Écrit un fichier JSON d'export global dans un bucket de sortie (premier output)
iv. Écrit chaque ligne (clé composée) dans une base de données de type SQL (pas de réplication, performance low, optimisation de coûts) (second output).
v. Créer une cloud function sécurisée de manière basic (token en header simple, vous pouvez chercher comment utiliser Google Secrets si besoin, sinon dans le code).
    1. Cette cloud function ne prend qu'une opération : GET sur la resource .../consumption/{id} et renvoie le résultat stocké auparavant dans cloud SQL.

c. Une deuxième branche :
i. lit aussi le fichier des codes NAF
ii. Réalisation d'une jointure selon la clé du code NAF (fouiller cette documentation : https://beam.apache.org/documentation/programming-guide/#core-beam-transforms)
iii. Enrichissez l'enregistrement original des informations NAF
iv. Écrire chaque enregistrement dans Bigquery.

Il est demandé de ne pas laisser allumer les instances Cloud SQL : prévoyez de les éteindre en HNO à minima manuellement, sinon prévoyez un système automatisé pour le faire (piste : cloud scheduler + cloud function + Google API 😉).

Attention à Dataflow :

1. borner le nombre de machines max à 1, choisir une petite machine,
2. borner le temps d'exécution max de votre batch (issue de vos lancements en local + overhead d'instanciation de la machine environ 1 minute voir moins)
3. Visualisez toujours dans la console ce que vous lancez : une erreur ou une boucle n'est pas forcément détectée par le Runtime Dataflow et peut engendrer des coûts supplémentaires inutiles.