

# Projet data Engineering

## I) Python et Data Engineering

**Objectif** : Réaliser un code clair et proprement structuré. Mettre en avant les éléments considérés comme essentiels pour du code utilisable dans un environnement de production. Mettre l'accent sur vos connaissances en conception de jobs de manipulation de données ainsi que les bonnes pratiques python.

### 1. Les données

Vous avez à votre disposition les 4 fichiers de données suivants :

**drugs.csv** : contient les noms de drugs (des médicaments) avec un id (atccode) et un nom (drug)

**pubmed.csv** : contient des titres d'articles PubMed (title) associés à un journal (journal) à une date donnée (date) ainsi qu'un id (id)

**pubmed.json** : même structure que pubmed.csv mais en format JSON

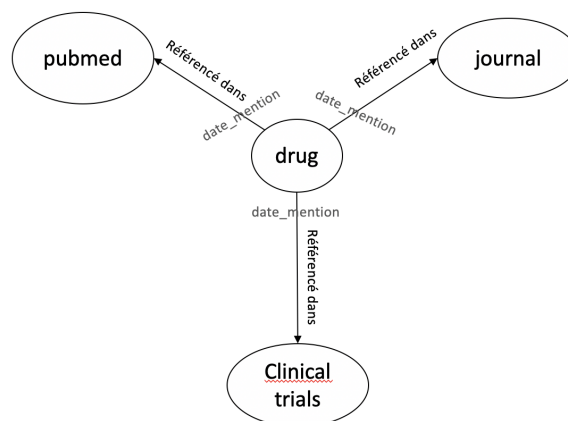
**clinical\_trials.csv** : contient des publications scientifiques avec un titre (scientific\_title), un id (id), un journal (journal) et une date (date).

### 2. Le travail à réaliser

L'objectif est de construire une data pipeline permettant de traiter les données définies dans la partie précédente afin de générer le résultat décrit dans la partie 3.

### 3. Data pipeline

Votre data pipeline doit produire en sortie un unique fichier JSON qui représente un graphe de liaison entre les différents médicaments et leurs mentions respectives dans les différentes publications PubMed, les différentes publications scientifiques et enfin les journaux avec la date associée à chacune de ces mentions.



## Mise en place d'un pipeline du projet Python

### 1. Réaliser un code clair et proprement structuré en Python

Récupérer le dossier projet\_medical\_pipeline à partir de l'URL : [https://github.com/Haithemkhelfa/Projet\\_medical\\_pipeline.git](https://github.com/Haithemkhelfa/Projet_medical_pipeline.git)

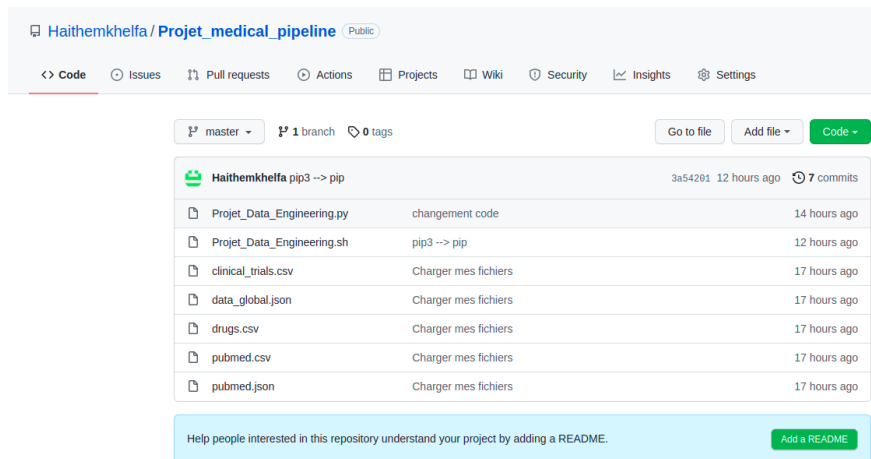
Le dossier projet\_medical\_pipeline contient les 7 fichiers suivants :

- drugs.csv
- pubmed.csv
- pubmed.json
- clinical\_trials.csv
- Projet\_data\_Enginnering.py (un code structuré en python)
- Projet\_data\_Enginnering.sh (un fichier pour créer un fichier Dockerfile )
- Data\_global.json (un fichier JSON qui représente les différents médicaments et leurs mentions respectives dans les différentes publications PubMed, les différentes publications scientifiques et enfin les journaux avec la date associée à chacune de ces mentions)

### 2. Mise en place d'un pipeline CI/CD avec Jenkins

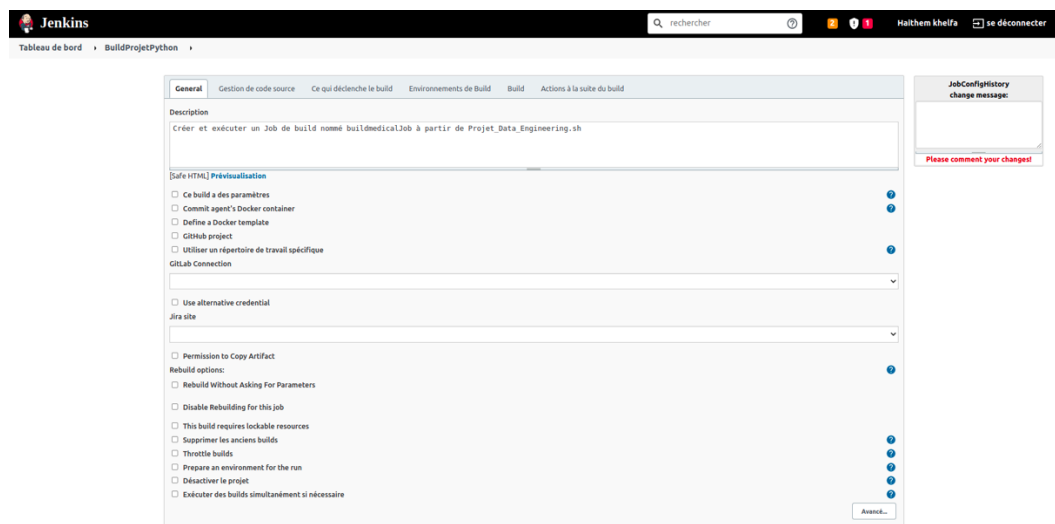
**Étape 1 :** Créer un répertoire projet\_medical\_pipeline dans notre compte github

**Étape 2 :** Remonter notre répertoire projet\_python dans le répertoire projet\_medical\_pipeline situé dans notre github



**Étape 3 :** Configurer le Jenkins **BuildProjectPython** pour pointer sur le répertoire projet\_medical\_pipeline sous github

**Étape 4 :** Créer et exécuter un Job de build nommé **BuildProjectPython** à partir de Projet\_data\_Enginnering.sh



| General   | Gestion de code source | Environnements de Build | Build | Actions à la suite du build |
|---|------------------------|-------------------------|-------|-----------------------------|
| <h2>Ce qui déclenche le build</h2> <ul style="list-style-type: none"> <li><input type="checkbox"/> Déclencher les builds à distance (Par exemple, à partir de scripts) <span>?</span></li> <li><input type="checkbox"/> Build when a change is pushed to BitBucket <span>?</span></li> <li><input type="checkbox"/> Build when a change is pushed to GitLab, GitLab webhook URL: <code>http://localhost:8080/project/BuildProjetPython</code> <span>?</span></li> <li><input type="checkbox"/> Construire après le build sur d'autres projets <span>?</span></li> <li><input type="checkbox"/> Construire périodiquement <span>?</span></li> <li><input type="checkbox"/> GitHub hook trigger for GITScm polling <span>?</span></li> <li><input type="checkbox"/> Scrutation de l'outil de gestion de version <span>?</span></li> </ul>   |                        |                         |       |                             |
| <h2>Environnements de Build</h2> <ul style="list-style-type: none"> <li><input type="checkbox"/> Delete workspace before build starts <span>?</span></li> <li><input type="checkbox"/> Use secret text(s) or file(s) <span>?</span></li> <li><input type="checkbox"/> Provide Configuration files <span>?</span></li> <li><input type="checkbox"/> Send files or execute commands over SSH before the build starts <span>?</span></li> <li><input type="checkbox"/> Send files or execute commands over SSH after the build runs <span>?</span></li> <li><input type="checkbox"/> Abort the build if it's stuck <span>?</span></li> <li><input type="checkbox"/> Add timestamps to the Console Output <span>?</span></li> <li><input type="checkbox"/> Ajouter le répertoire bin/ de Node/npm au PATH <span>?</span></li> <li><input type="checkbox"/> Color ANSI Console Output <span>?</span></li> <li><input type="checkbox"/> Execute shell script on remote host using ssh <span>?</span></li> <li><input type="checkbox"/> Generate Release Notes <span>?</span></li> <li><input type="checkbox"/> Inject environment variables to the build process <span>?</span></li> <li><input type="checkbox"/> Inject passwords to the build as environment variables <span>?</span></li> <li><input type="checkbox"/> Inspect build log for published Gradle build scans <span>?</span></li> <li><input type="checkbox"/> SSH Agent <span>?</span></li> <li><input type="checkbox"/> Set Build Name <span>?</span></li> <li><input type="checkbox"/> Set Jenkins user build variables <span>?</span></li> <li><input type="checkbox"/> Setup Kubernetes CLI (kubectl) <span>?</span></li> <li><input type="checkbox"/> With Ant <span>?</span></li> </ul> |                        |                         |       |                             |

Build

Exécuter un script shell

Commande

```
bash ./Projet_Data_Engineering.sh
```

Voir la liste des variables d'environnement disponibles

Avancé...

Actions à la suite du build

Ajouter une action après le build

Sauver

Apply

Dis à Jenkins de construire le fichier JSON. Sur le côté gauche, cliquez sur Lancer un build pour démarrer le travail. Jenkins will download notre Git repository and execute the build command *Projet\_data\_Enginnering.sh*.

The screenshot shows the Jenkins dashboard for the job 'Projet BuildProjetPython'. The left sidebar contains navigation links: 'Retour au tableau de bord', 'État', 'Modifications', 'Répertoire de travail', 'Lancer un build', 'Configurer', 'Supprimer Projet', 'Rebuild Last', 'Favoris', 'Job Config History', 'Open Blue Ocean', and 'Renommer'. Below these is the 'Historique des builds' section with a search bar and a table showing two builds: #2 (31 Janv. 2022 11:44) and #1 (31 Janv. 2022 11:33). The main content area shows the job name, a description, and buttons for 'modifier la description' and 'Désactiver le projet'. It also displays 'Espace de travail', 'Changements récents', 'Projets en aval' (listing 'TestProjetPython'), and 'Liens permanents' with links to the last build, stable build, successful build, and last completed build.

**Étape 5 :** Créer et exécuter un Job de test nommé **TestProjetPython**. Le test réussit si le fichier data.json a été créé avec succès dans le répertoire : /home/myapp/data.json

The screenshot shows the Jenkins configuration page for the job 'TestProjetPython'. The tabs at the top are 'General', 'Gestion de code source', 'Ce qui déclenche le build', 'Environnements de Build', 'Build', and 'Actions à la suite du build'. The 'General' tab is selected, showing the job description, a 'Safe HTML' preview, and various checkboxes for build options like 'Ce build a des paramètres', 'Commit agent's Docker container', 'Define a Docker template', 'GitHub project', 'Utiliser un répertoire de travail spécifique', 'GitLab Connection', 'Use alternative credential', 'Jira site', 'Permission to Copy Artifact', 'Rebuild options', 'Disable Rebuilding for this job', 'This build requires lockable resources', 'Supprimer les anciens builds', 'Throttle builds', 'Prepare an environment for the run', 'Désactiver le projet', and 'Exécuter des builds simultanément si nécessaire'. A 'Please comment your changes!' message is visible on the right.

The screenshot shows the Jenkins configuration page for the job 'TestProjetPython', specifically the 'Gestion de code source' tab. It displays options for source code management: 'Aucune', 'CVS', 'CVS Projectset', 'Git', 'Mercurial', and 'Subversion'. Below this is the 'Ce qui déclenche le build' section with checkboxes for 'Déclencher les builds à distance', 'Build when a change is pushed to BitBucket', 'Build when a change is pushed to GitLab', and 'Construire après le build sur d'autres projets'. The 'Construire après le build sur d'autres projets' option is selected, showing a list of projects to watch, including 'BuildProjetPython'. Other options include 'Déclencher que si la construction est stable', 'Déclencher même si la construction est instable', 'Déclencher même si la construction échoue', 'Always trigger, even if the build is aborted', 'Construire périodiquement', 'GitHub hook trigger for GITScm polling', and 'Scrutation de l'outil de gestion de version'.



## Étape 6 : Créer et exécuter un pipeline de Build et de Test pour notre projet python dans Jenkins

Dans cette partie, vous allons écrire un pipeline dans Jenkins pour exécuter nos deux tâches

- Configurez le travail **Projet\_Python\_Pipeline**

The screenshot shows the Jenkins configuration page for a job named 'Projet\_Python\_Pipeline'. The page is divided into several tabs: 'General', 'Build Triggers', 'Advanced Project Options', and 'Pipeline'. The 'General' tab is currently selected.

**General Tab:**

- Description:** Créer et exécuter un pipeline de Build et de Test pour mon projet medical
- Safe HTML:** Prévisualisation
- Options:**
  - ☐ Ce build a des paramètres
  - ☐ Do not allow concurrent builds
  - ☐ Do not allow the pipeline to resume if the controller restarts
  - ☐ GitHub project
  - ☐ GitLab Connection
  - ☐ Use alternative credential
  - ☐ Jira site
  - ☐ Permission to Copy Artifact
  - ☐ Pipeline speed/durability override
  - ☐ Preserve stashes from completed builds
- Rebuild options:**
  - ☐ Rebuild Without Asking For Parameters
  - ☐ Disable Rebuilding for this job
  - ☐ Supprimer les anciens builds
  - ☐ Throttle builds
  - ☐ Prepare an environment for the run

**Build Triggers Tab:**

- ☐ Build when a change is pushed to BitBucket
- ☐ Build when a change is pushed to GitLab. GitLab webhook URL: http://localhost:8080/project/Projet\_Python\_Pipeline
- ☐ Build whenever a SNAPSHOT dependency is built
- ☐ Construire après le build sur d'autres projets
- ☐ Construire périodiquement
- ☐ GitHub hook trigger for GITScm polling
- ☐ Scrutation de l'outil de gestion de version
- ☐ Désactiver le projet
- ☐ Période d'attente
- ☐ Déclencher les builds à distance (Par exemple, à partir de scripts)

**Advanced Project Options Tab:**

**Pipeline Tab:**

**Definition**

**Pipeline script**

```
1 node {
2   stage('Preparation') {
3     catchError(buildResult: 'SUCCESS') {
4       sh 'docker stop testproject'
5       sh 'docker rm testproject'
6     }
7   }
8 }
9 stage('Build JSON file') {
10  build 'BuildProjetPython'
11 }
12
13 stage('Result : Output JSON file') {
14  build 'TestProjetPython'
15 }
16 }
```

☒ Use Groovy Sandbox

**Pipeline Syntax**

**Buttons:** Sauver, Apply

Dans la section Pipeline, ajoutez le script suivant :

```
node {
  stage('Preparation') {
    catchError(buildResult: 'SUCCESS') {
      sh 'docker stop testproject'
      sh 'docker rm testproject' } }
  stage ('Build JSON file') { build 'BuildProjectPython' }
  stage('Result : Output JSON file') { build 'TestProjectPython' }
}
```

Ce script effectue les opérations suivantes :

- Il crée une construction de nœud unique par opposition à un nœud distribué ou multi-nœud.
- Dans la phase de préparation, **Projet\_Python\_Pipeline** vérifiera d'abord que toutes les instances précédentes du conteneur docker **BuildProjectPython** sont arrêtées et supprimées. Mais s'il n'y a pas encore de conteneur en cours d'exécution, on obtiendra une erreur. Par conséquent, on utilise la fonction `catchError` pour attraper les erreurs et renvoyer une valeur "SUCCESS". Cela permettra de faire en sorte que le pipeline passe à l'étape suivante.
- Dans l'étape Build, **Projet\_Python\_Pipeline** construira notre **BuildProjectPython** (Build JSON file).
- Dans l'étape Résultat, **Projet\_Python\_Pipeline** va construire votre **TestProjectPython** (Output JSON file).

Une fois on a exécuté notre pipeline « **Projet\_Python\_Pipeline** », la **Stage View** doit afficher trois zones vertes avec le nombre de secondes que chaque étape a pris pour construire.

## Résultats

Notre data pipeline produit en sortie un unique fichier JSON « data.json » qui représente les différents médicaments et leurs mentions respectives dans les différentes publications PubMed, les différentes publications scientifiques et enfin les journaux avec la date associée à chacune de ces mentions.

The screenshot shows the Jenkins web interface for a pipeline named 'Projet\_Python\_Pipeline'. The left sidebar contains navigation links: 'Back to Dashboard', 'Status', 'Changes', 'Lancer un build' (highlighted), 'Configurer', 'Supprimer Pipeline', 'Full Stage View', 'Job Config History', 'Open Blue Ocean', 'Renommer', and 'Pipeline Syntax'. Below these is the 'Historique des builds' section, showing a list of builds with columns for build number, date, and status. The main content area displays the 'Pipeline View' for 'Projet\_Python\_Pipeline', with a description: 'Créer et exécuter un pipeline de Build et de Test pour mon projet medical'. It includes a 'Recent Changes' section and a 'Stage View' table. The 'Stage View' table shows the following data:

|                               | Preparation | Build JSON file | Result : Output JSON file |
|-------------------------------|-------------|-----------------|---------------------------|
| Average stage times:          | 429ms       | 12s             | 6s                        |
| (Average full run time: ~20s) |             |                 |                           |
| #1 Jan 31 11:44 No Changes    | 429ms       | 12s             | 6s                        |

Below the 'Stage View' table, there is a 'Liens permanents' section. The top right of the interface shows the user 'Haithem khelfa' and a 'se déconnecter' button.

## II) SQL

**Objectif :** Réaliser des requêtes SQL claires et facilement compréhensibles.

### 1. Les données

Nous avons les tables suivantes :

#### TRANSACTIONS

Cette table contient des données transactionnelles avec les infos suivantes :

- **date** : date à laquelle la commande a été passée
- **order\_id** : identifiant unique de la commande
- **client\_id** : identifiant unique du client
- **prod\_id** : identifiant unique du produit acheté
- **prod\_price** : prix unitaire du produit
- **prod\_qty** : quantité de produit achetée

Echantillon de la table TRANSACTION :

| date     | order_id | client_id | prod_id | prod_price | prod_qty |
|----------|----------|-----------|---------|------------|----------|
| 01/01/20 | 1234     | 999       | 490756  | 50         | 1        |
| 01/01/20 | 1234     | 999       | 389728  | 3,56       | 4        |
| 01/01/20 | 3456     | 845       | 490756  | 50         | 2        |
| 01/01/20 | 3456     | 845       | 549380  | 300        | 1        |
| 01/01/20 | 3456     | 845       | 293718  | 10         | 6        |

#### PRODUCT\_NOMENCLATURE

Cette table contient le référentiel produit c'est à dire les méta-données du produit. On y trouve les infos suivantes :

- **product\_id** : identifiant unique du produit
- **product\_type** : type de produit (DECO ou MEUBLE)
- **product\_name** : le nom du produit

Echantillon de la table PRODUCT\_NOMENCLATURE :

| product_id | product_type | product_name  |
|------------|--------------|---------------|
| 490756     | MEUBLE       | Chaise        |
| 389728     | DECO         | Boule de Noël |
| 549380     | MEUBLE       | Canapé        |
| 293718     | DECO         | Mug           |



## **2. Première partie du test**