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

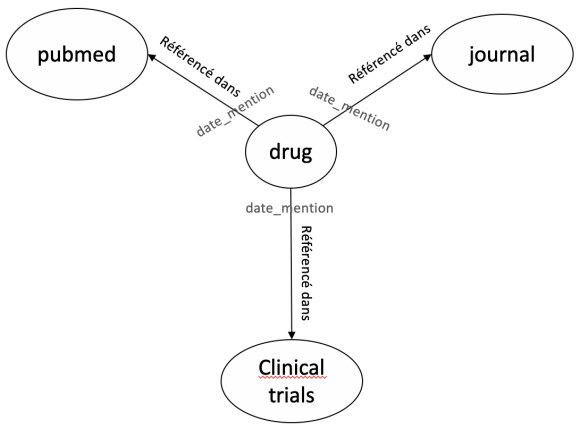
clinical_trials.csv : contient des publications scientifiques avec un titre (scientific_title), un id (id), un journal (journal) et une date (date).

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

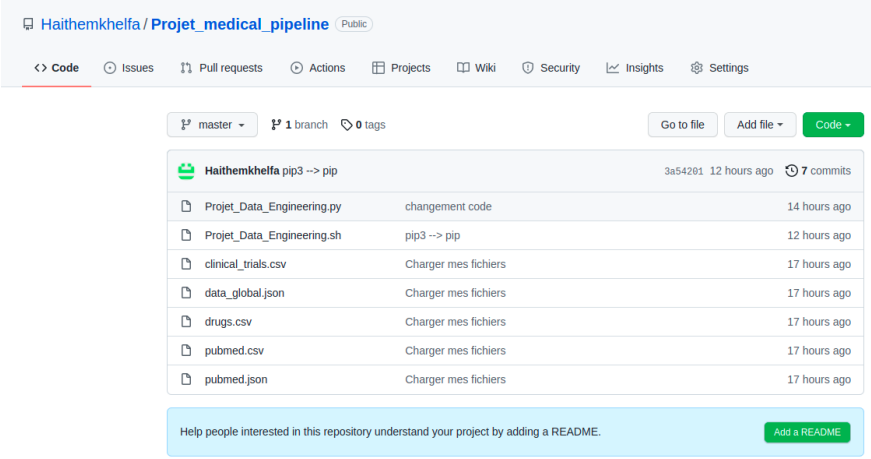
Le dossier projet_medical_pipeline contient les 7 fichiers suivants :

- 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 pipline CI/CD avec Jenkins*

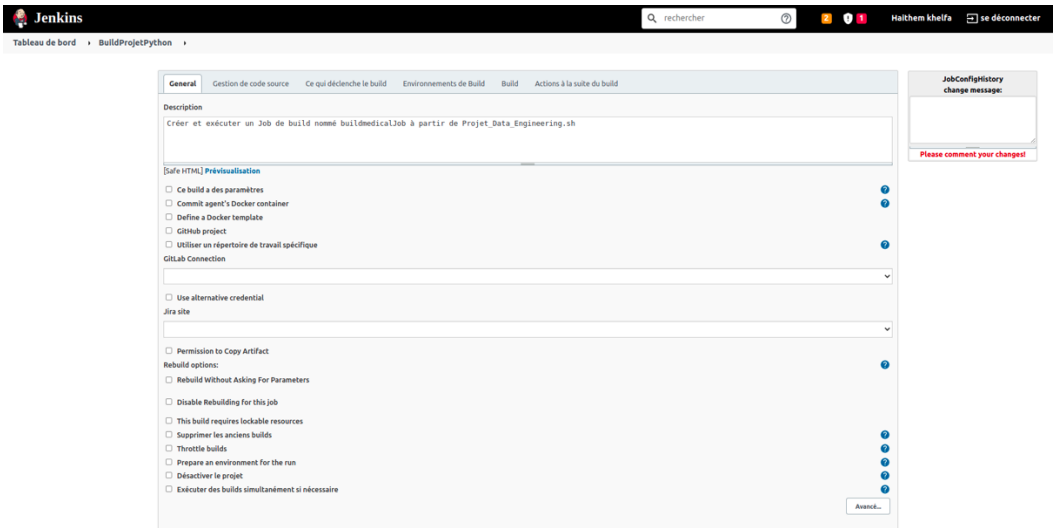
Étape 1 : Créer un répertoire projet_medical_pipeline dans notre compte github

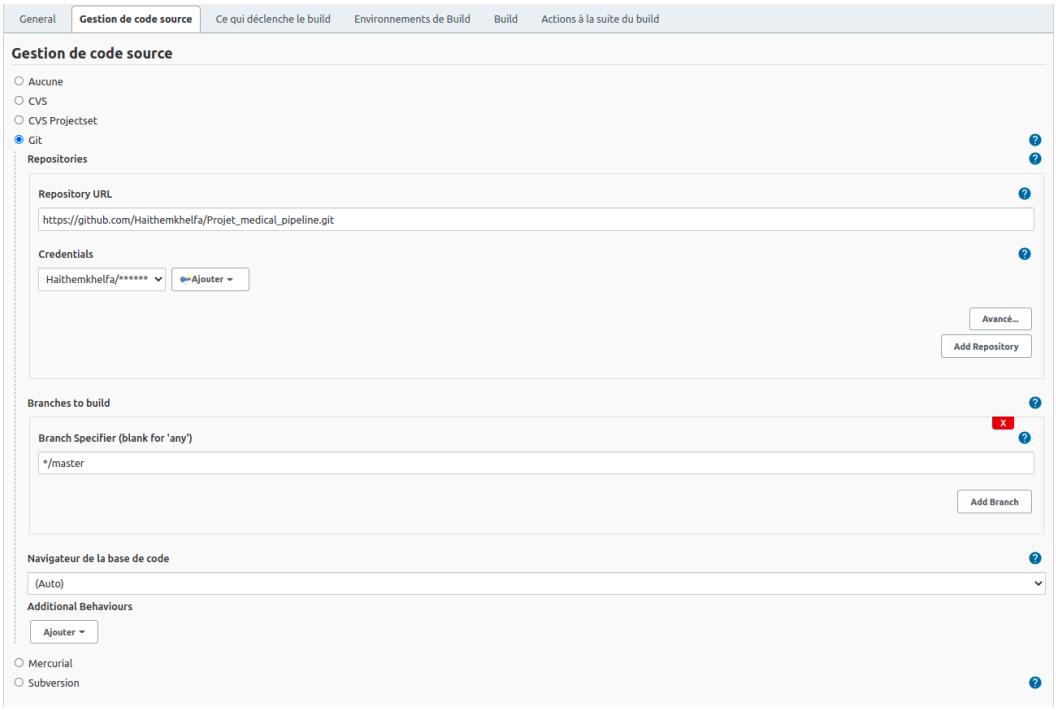
Étape 2 : Remonter notre répertoire projet_python dans le répertoire projet_medical_pipeline situé dans notre github

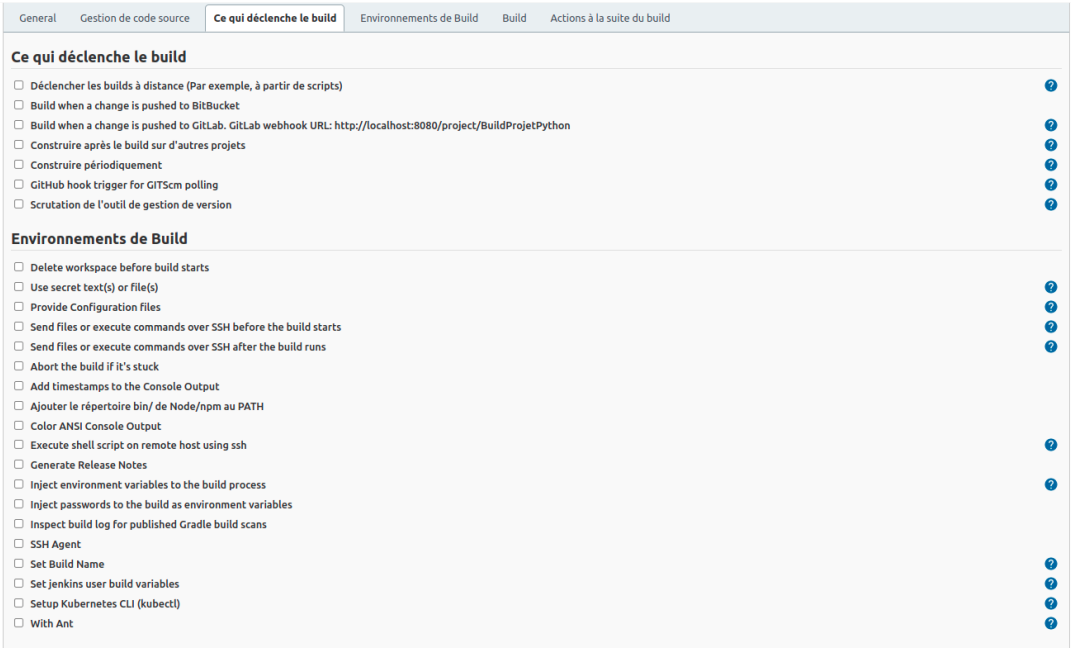


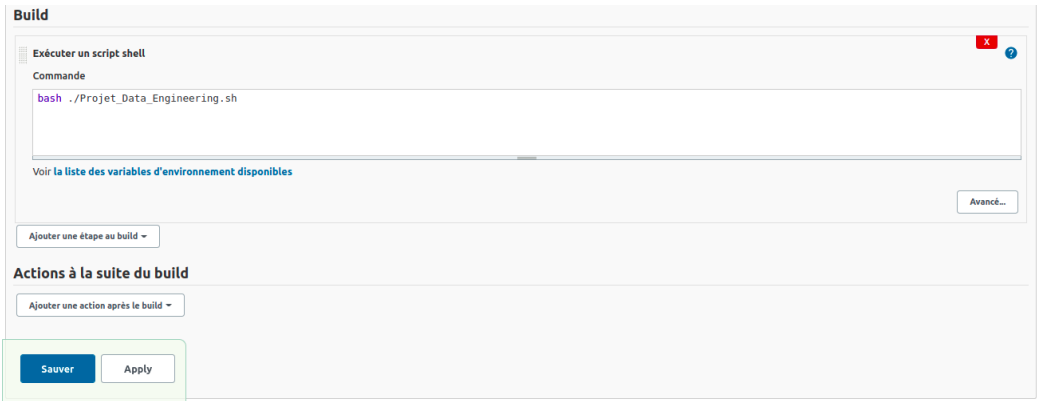
Étape 3 : Configurer le Jenkins **BuildProjetPython** pour pointer sur le répertoire projet_medical_pipeline sous github

Étape 4 : Créer et exécuter un Job de build nommé **BuildProjetPython** à partir de Projet_data_Enginnering.sh

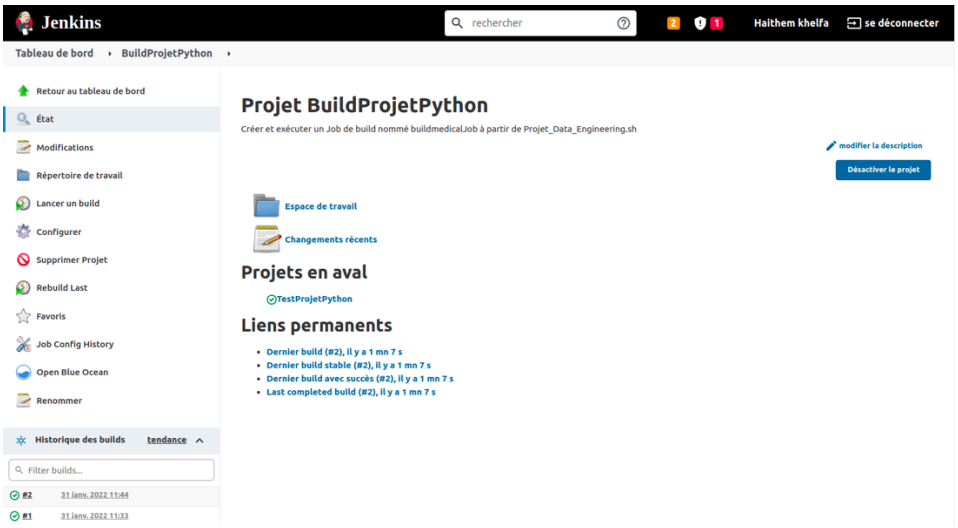



General
Gestion de code source
Ce qui déclenche le build
Environnements de Build
Build
Actions à la suite du build
Gestion de code source
Aucune
CVS
CVS Projectset
Git
Repositories
Repository URL
https://github.com/Haithemkhelfa/Projet_medical_pipeline.git
Credentials
Haithemkhelfa/******
Ajouter
Avancé...
Add Repository
Branches to build
Branch Specifier (blank for 'any')
*/master
Add Branch
Navigateur de la base de code
(Auto)
Additional Behaviours
Ajouter
Mercurial
Subversion

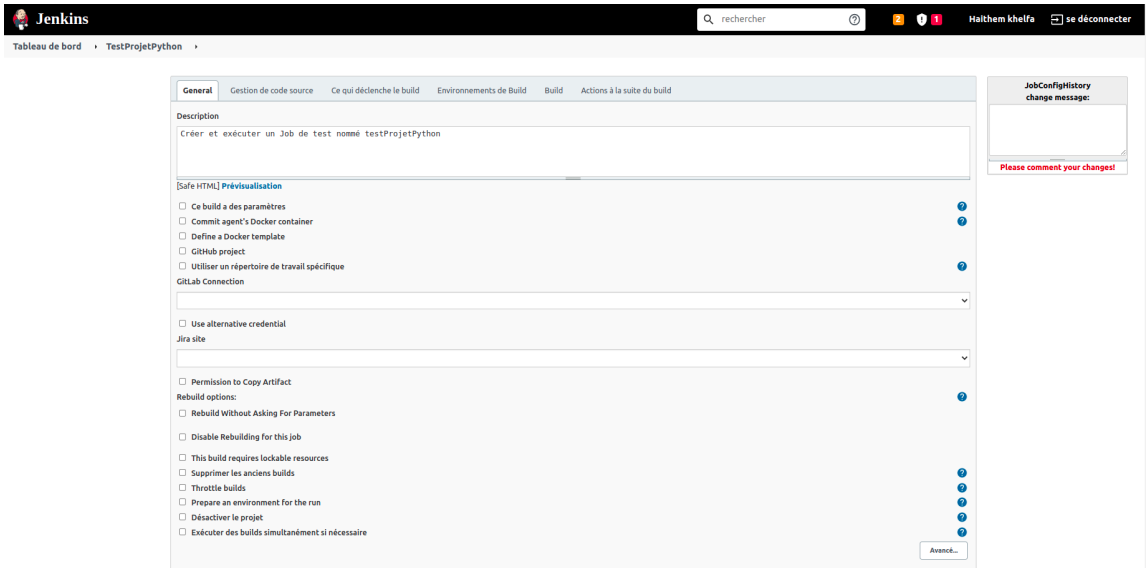

General
Gestion de code source
Ce qui déclenche le build
Environnements de Build
Build
Actions à la suite du build
Ce qui déclenche le build
Déclencher les builds à distance (Par exemple, à partir de scripts)
Build when a change is pushed to BitBucket
Build when a change is pushed to GitLab. GitLab webhook URL: http://localhost:8080/project/BuildProjetPython
Construire après le build sur d'autres projets
Construire périodiquement
GitHub hook trigger for GITScm polling
Scrutation de l'outil de gestion de version
Environnements de Build
Delete workspace before build starts
Use secret text(s) or file(s)
Provide Configuration files
Send files or execute commands over SSH before the build starts
Send files or execute commands over SSH after the build runs
Abort the build if it's stuck
Add timestamps to the Console Output
Ajouter le répertoire bin/ de Node/npm au PATH
Color ANSI Console Output
Execute shell script on remote host using ssh
Generate Release Notes
Inject environment variables to the build process
Inject passwords to the build as environment variables
Inspect build log for published Gradle build scans
SSH Agent
Set Build Name
Set jenkins user build variables
Setup Kubernetes CLI (kubectl)
With Ant

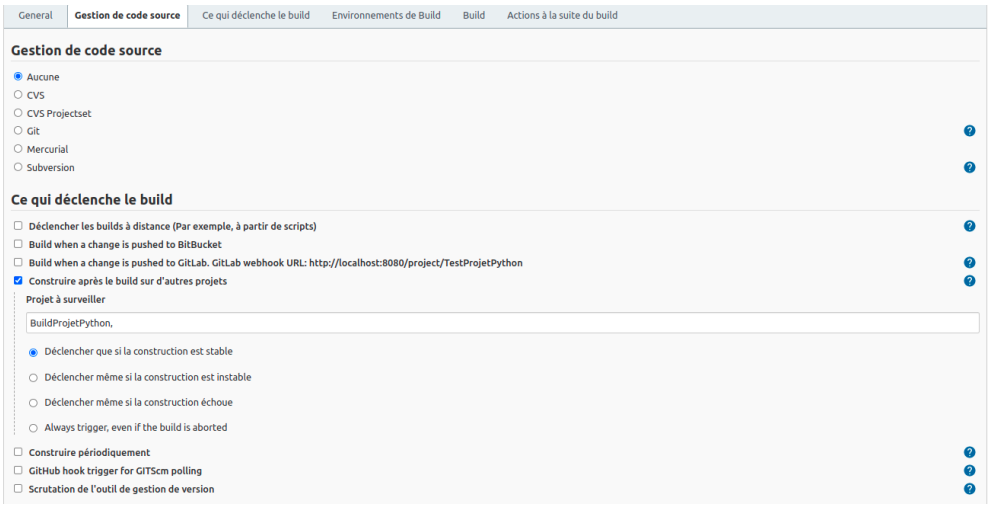

Build
Exécuter un script shell
Commande
bash ./Projet_Data_Engineering.sh
Voir la liste des variables d'environnement disponibles
Avancé...
Ajouter une étape au build
Actions à la suite du build
Ajouter une action après le build
Sauver
Apply

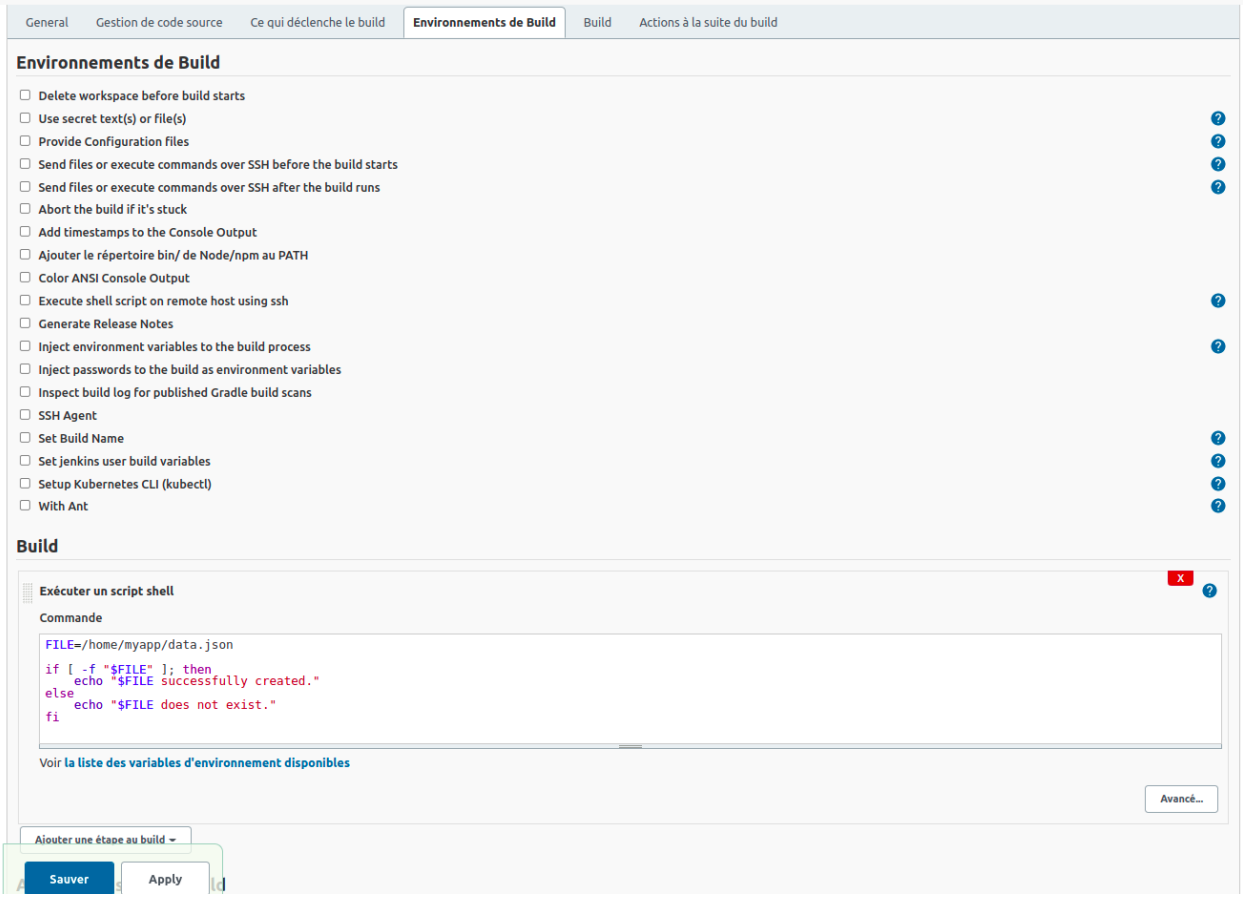
Dis à Jenkins de construire le fichier JSON. Sur le côté gauche, cliquez sur Lancer un build pour démarrer le travail. Jenkins will download notre Git repository and execute the build command *Projet_data_Enginnering.sh*.



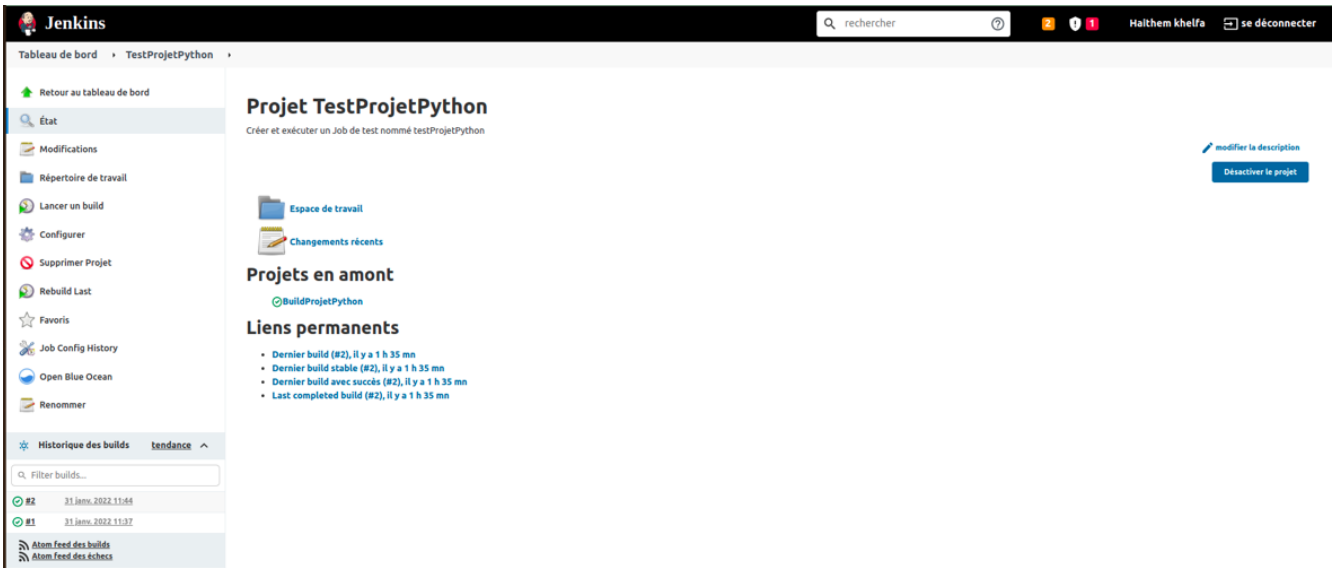
Étape 5 : Créer et exécuter un Job de test nommé **TestProjetPython.** Le test réussit si le fichier data.json a été créé avec succès dans le répertoire : /home/myapp/data.json







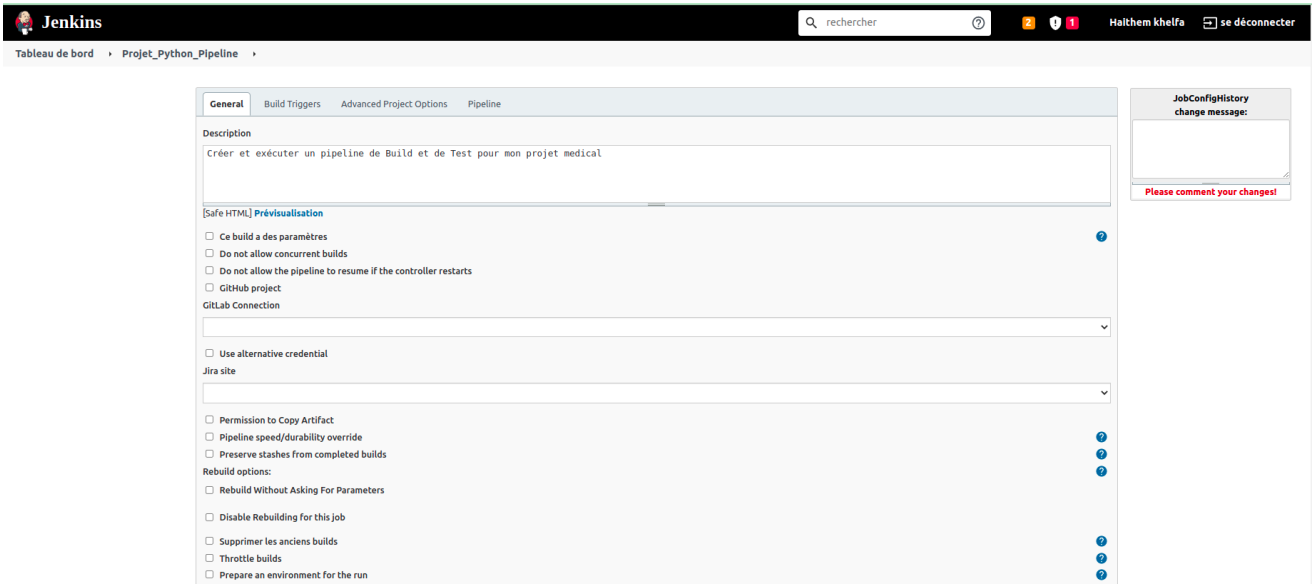
Dis à Jenkins d'exécuter le travail **BuildProjetPython** à nouveau. On devrait maintenant voir que les deux tâches **BuildProjetPython** & **TestProjetPython** sont terminées. Cela signifie que notre code pour les deux tâches s'est exécuté sans erreur.

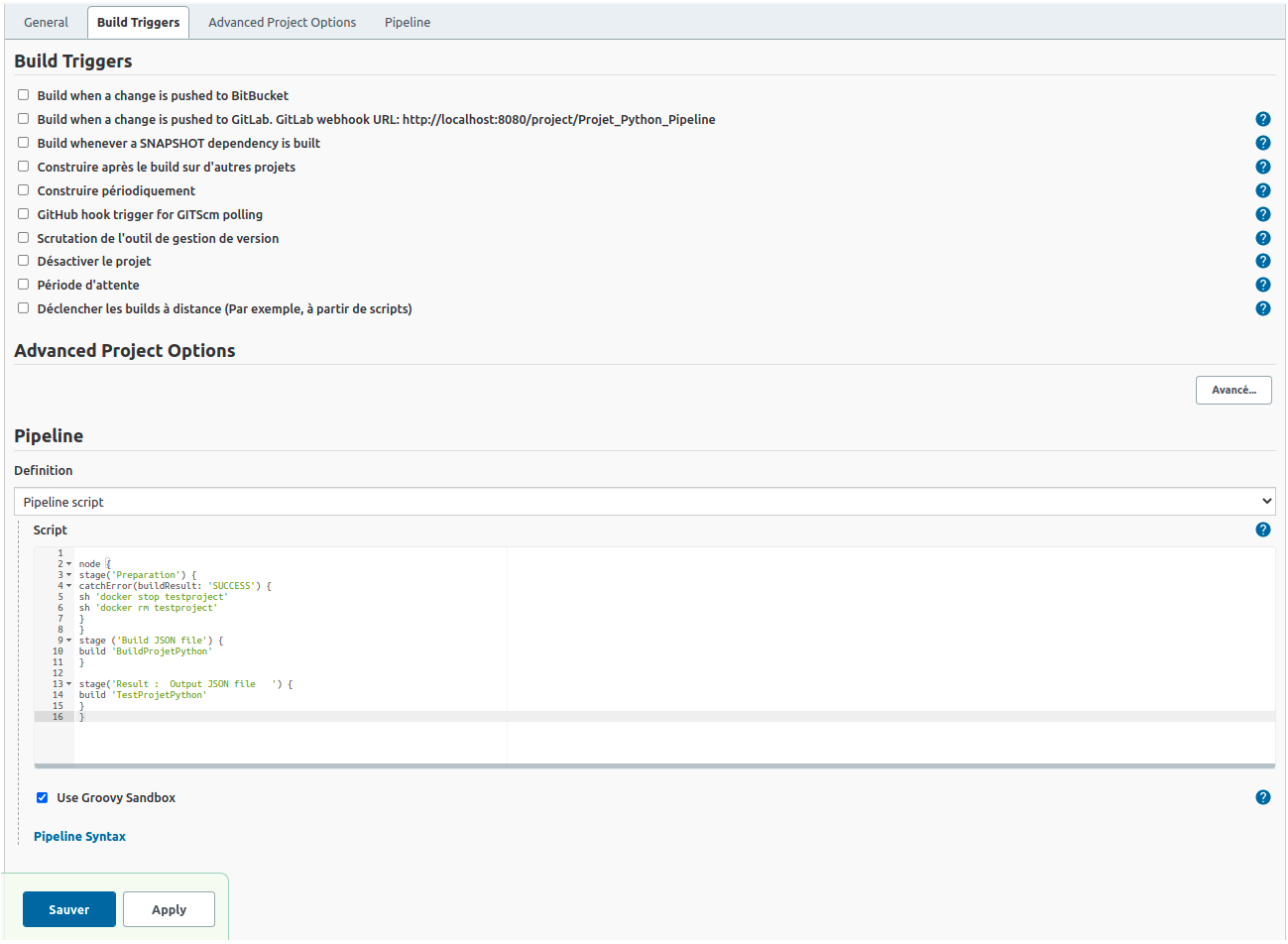


Étape 6 : Créer et exécuter un pipeline de Build et de Test pour notre projet python dans Jenkins

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

- Configurez le travail **Projet_Python_Pipeline**





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 'BuildProjetPython' }
stage('Result : Output JSON file') { build 'TestProjectPython }
}
```

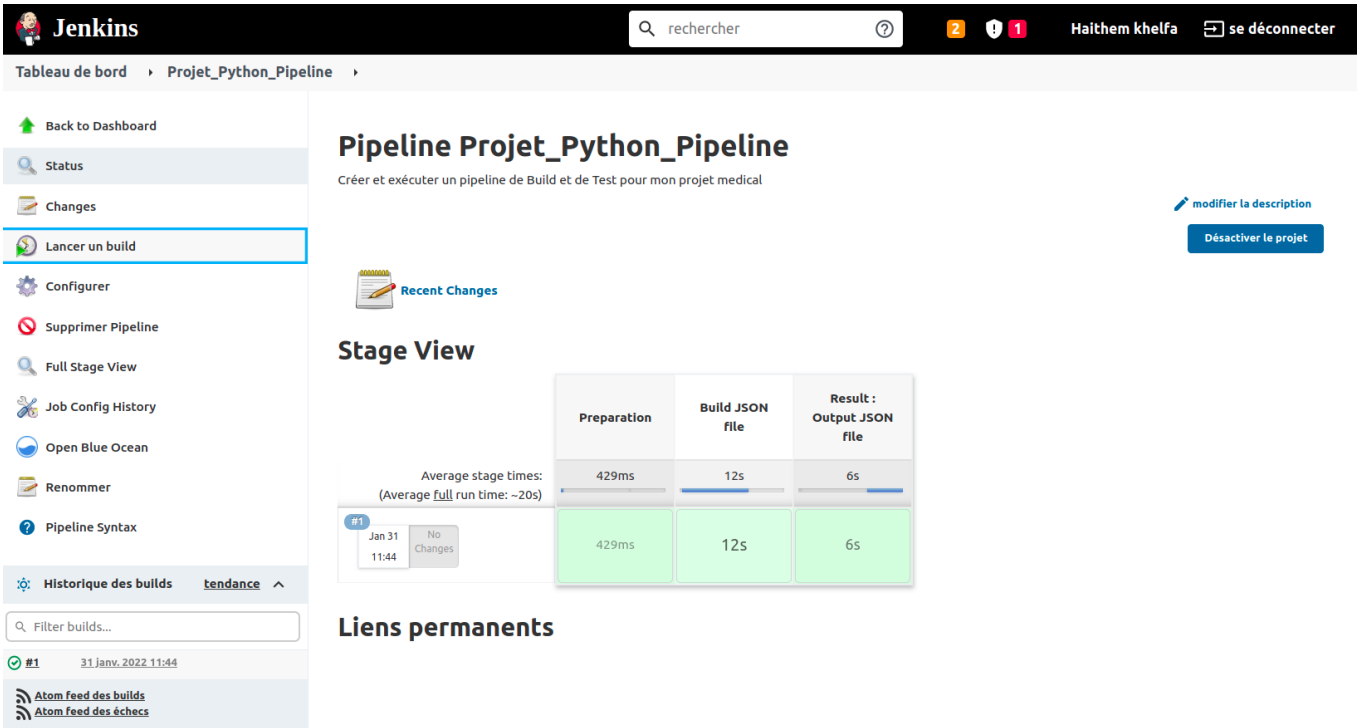
Ce script effectue les opérations suivantes :

- 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.



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