

Mode d'Emploi

Data pipeline pour le traitement de données médicales avec Jenkins

Objectif d'exercice :

A partir du repository Github présent, configurer CI/CD pour établir un data pipeline pour les tâches de data processing

Etapes de data processing

Etape 1 : Restitution de données pubmed dans un seul fichier .csv

- exécutables : pubmed_restitution.py
- inputs data:
 - data/pubmed.json
 - data/pubmed.csv
- outputs data:
 - data/pubmed_bis.json (fichier intermédiaire)
 - data/pubmed_restit.csv (fichier final)

Etape 2 : Génération d'une représentation de graphes des 3 tables drugs, clinical trials et pubmed dans un seul fichier .json

- exécutables :
 - main.py
 - components.py (appelé par main.py)
 - processing_fct.py (appelé par main.py)
- inputs data:
 - data/drugs.csv
 - data/clinical_trails.csv
 - data/pubmed_restit.csv (créé par Etape 1)
- outputs data: data/complete_data.json

Orchestration de jobs avec Jenkins

Contexte :

Pour intégrer de différentes étapes de notre data pipeline dans un orchestrateur de jobs, nous faisons appel à Jenkins pour tester et déployer notre code.

Pour cela, ce chapitre présentera les étapes suivantes sur Jenkins :

- Associer un repository Github à des jobs Jenkins
- Utiliser des fichiers .sh pour la création de Dockerfile afin de créer et lancer un container contenant toutes les dépendances nécessaires pour l'exécution de notre code
- Builder des jobs sur Jenkins and tester ces jobs
- Connecter les jobs ensemble à travers un pipeline

Pas à pas :

Utiliser la line de commande suivante pour lancer le jenkins server :

```
docker run --rm -u root -p 8080:8080 -v jenkins-data:/var/jenkins_home -v $(which
docker):/usr/bin/docker -v /var/run/docker.sock:/var/run/docker.sock -v "$HOME":/home --name
jenkins_server jenkins/jenkins:lts
```

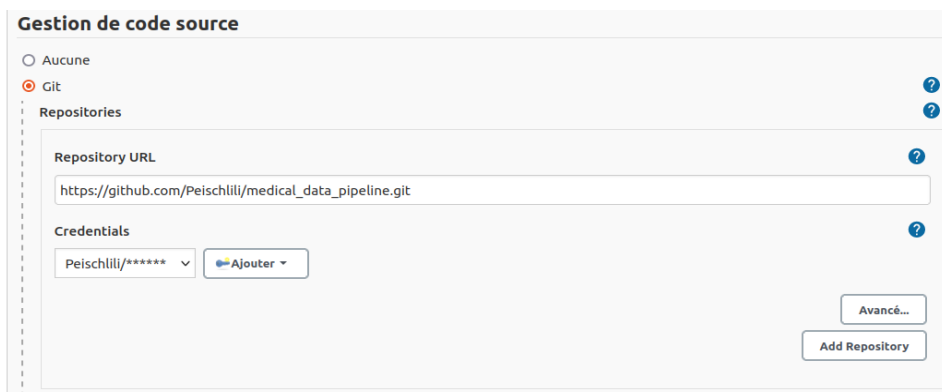
Se rendre sur le port défini sur la commande puis se connecter pour accéder à Jenkins :
<http://localhost:8080/>

Pour chaque étape de data processing mentionné, nous créerons 1 job de build et 1 job de test afin de sécuriser le bon déroulement de la tâche.

Dans ce but, pour Etape 1, nous allons créer job pubmed_restit et job pubmed_restit_test.

Création job pubmed_restit

Dans l'onglet Gestion de code source, associer le lien du repository Github en question, puis ajouter authentications Github :



Utiliser fichier pubmed_restit.sh pour la préparation d'environnement, la création de Dockerfile et la lancement du container à partir du Dockerfile :

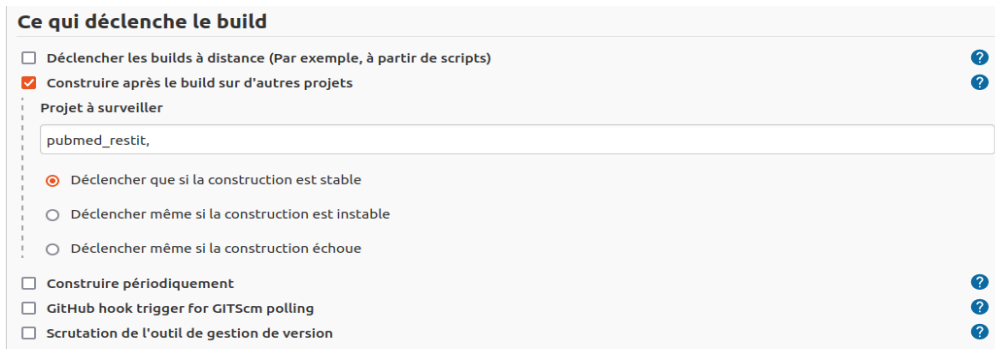


Le Dockerfile généré est visualisable dans le Répertoire de Travail du job :

```
FROM python
RUN pip3 install pandas
RUN pip3 install json
RUN pip3 install re
RUN pip3 install sys
COPY ./data /home/med_project/data/
COPY pubmed_restitution.py /home/med_project/
COPY components.py /home/med_project/
COPY processing_fct.py /home/med_project/
COPY main.py /home/med_project/
EXPOSE 5050
CMD python3 /home/med_project/pubmed_restitution.py
```

Création job pubmed_restit_test

Ce job est un test pour vérifier le bon fonctionnement du job pubmed_restit. Ainsi, nous devons configurer le déclenchement du build dans l'onglet suivant :



Ce qui déclenche le build

- ☐ Déclencher les builds à distance (Par exemple, à partir de scripts)
- ☒ Construire après le build sur d'autres projets

Projet à surveiller

pubmed_restit,

- ☒ Déclencher que si la construction est stable
- ☐ Déclencher même si la construction est instable
- ☐ Déclencher même si la construction échoue

- ☐ Construire périodiquement
- ☐ GitHub hook trigger for GITScm polling
- ☐ Scrutation de l'outil de gestion de version

Dans Build, ajouter un script shell pour vérifier si le fichier d'output a bien été généré :



Build

Exécuter un script shell

Commande

```
FILE=/home/med_project/data/pubmed_restit.csv
if [ -f "$FILE" ]; then
  echo "$FILE successfully created."
else
  echo "$FILE does not exist."
fi
```

Voir [la liste des variables d'environnement disponibles](#)

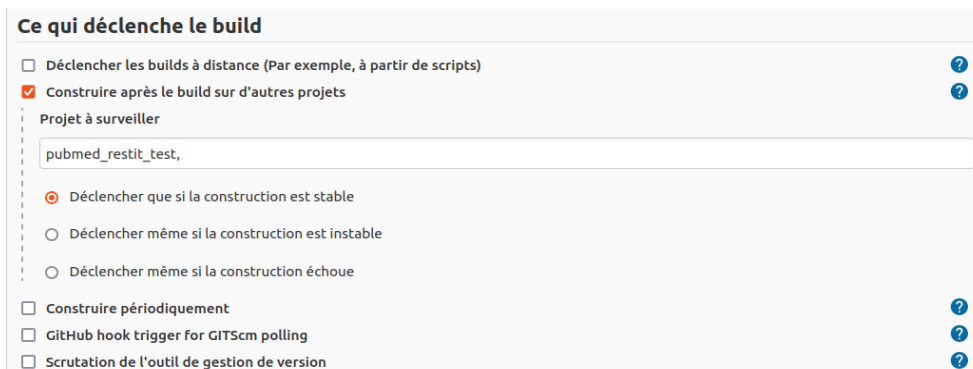
Avancé...

Ajouter une étape au build

Création job build_links

Dans l'onglet Gestion de code source, associer le lien du repository Github en question, puis ajouter authentifications Github (même procédure pour la création pubmed_restit).

Ce job est destiné à être construit dans un data pipeline une fois que le job pubmed_restit_test a été buildé. Par conséquent, nous devons configurer le Build du job comme suit :



Ce qui déclenche le build

- ☐ Déclencher les builds à distance (Par exemple, à partir de scripts)
- ☒ Construire après le build sur d'autres projets

Projet à surveiller

pubmed_restit_test,

- ☒ Déclencher que si la construction est stable
- ☐ Déclencher même si la construction est instable
- ☐ Déclencher même si la construction échoue

- ☐ Construire périodiquement
- ☐ GitHub hook trigger for GITScm polling
- ☐ Scrutation de l'outil de gestion de version

Utiliser fichier main.sh pour exécuter fichier main.py dans le container préalablement créé par le job pubmed_restit :

The screenshot shows the 'Build' configuration page in Jenkins. The title is 'Build'. Below it, there's a section 'Exécuter un script shell' with a red 'x' icon and a help icon. Under 'Commande', the text 'bash ./main.sh' is entered. Below the command field, there's a link 'Voir la liste des variables d'environnement disponibles'. At the bottom right, there's an 'Avancé...' button. At the bottom left, there's a button 'Ajouter une étape au build'.

Création job test_links

Ce job est un test pour vérifier le bon fonctionnement du job build_links dans un data pipeline. Suivant la même logique, nous devons configurer le déclenchement du build dans l'onglet suivant :

The screenshot shows the 'Ce qui déclenche le build' configuration page in Jenkins. It has several options: 'Déclencher les builds à distance (Par exemple, à partir de scripts)' is unchecked; 'Construire après le build sur d'autres projets' is checked. Below this, there's a 'Projet à surveiller' field with 'build_links,' entered. There are three radio button options: 'Déclencher que si la construction est stable' (selected), 'Déclencher même si la construction est instable', and 'Déclencher même si la construction échoue'. At the bottom, there are three unchecked checkboxes: 'Construire périodiquement', 'GitHub hook trigger for GITScm polling', and 'Scrutation de l'outil de gestion de version'. Each of these bottom checkboxes has a help icon to its right.

Similaire au job pubmed_restit_test, nous devons ajouter un script shell afin de s'assurer que le fichier final .json a été correctement généré :

The screenshot shows the 'Build' configuration page in Jenkins. The title is 'Build'. Below it, there's a section 'Exécuter un script shell' with a red 'x' icon and a help icon. Under 'Commande', a shell script is entered:

```
FILE=/home/med_project/data/complete_data.json
if [ -f "$FILE" ]; then
  echo "$FILE successfully created."
else
  echo "$FILE does not exist."
fi
```

 Below the command field, there's a link 'Voir la liste des variables d'environnement disponibles'. At the bottom right, there's an 'Avancé...' button. At the bottom left, there's a button 'Ajouter une étape au build'.

Création de pipeline

Une fois que les 4 jobs Jenkins sont créés (sauvé), nous allons créer un Jenkins pipeline pour connecter ces jobs les uns avec les autres à l'ordre prédéfinie dans chacun des jobs. Pour ce faire, il nous faudra définir dans l'onglet Pipeline les étapes de construction dans un script :

Pipeline

Definition

Pipeline script

Script

```

1 node {
2   stage('Preparation container') {
3     catchError(buildResult: 'SUCCESS') {
4       sh 'docker stop ctn_restit'
5       sh 'docker rm ctn_restit'
6     }
7   }
8   stage('Build pubmed restitution') {
9     build 'pubmed_restit'
10  }
11  stage('Result pubmed restitution') {
12    build 'pubmed_restit_test'
13  }
14  stage('Build final json') {
15    build 'build_links'
16  }
17  stage('Result final json') {
18    build 'test_links'
19  }

```

☒ Use Groovy Sandbox

[Pipeline Syntax](#)

Une fois le pipeline créé, nous pouvons lancer un build pour ce pipeline et visualiser son état d'exécution et d'avancement.

Pipeline pipeline_pubmed_restit

Pipeline following build and test jobs of following tasks:

- pubmed files restitution
- json file creation



Stage View

	Preparation container	Build pubmed restitution	Result pubmed restitution	Build final json	Result final json
Average stage times: (Average full run time: ~36s)	357ms	10s	6s	9s	9s
#19 Jan 31 15:49 No Changes	347ms	9s	6s		
#18 Jan 31 15:47 No Changes	352ms	9s	6s	10s	9s
#17 Jan 31 13:35 No Changes	376ms	9s	6s	10s	9s