# Projet programation spécialisée : *Datastream avec python, MySQL et RabbitMQ*

## **Objectif**

L'objectif du projet est de créer un système d'analyse des logs d'un serveur web en temps réel en utilisant Python, MySQL et RabbitMQ . Le système comprend un Producer appelé logs-producer qui lit ligne par ligne un fichier de logs web-server-nginx.log, les publie dans un échange de type topic, et les envoie à deux fichiers d'attente, l'une appelée queue-data-lake et l'autre queue-data-clean. Le système comprend également deux consommateurs, le data-lake-consumer et le data-clean-consumer, qui nécessitent chaque fichier d'attente différemment en temps réel.

## **Prerequis**

* Python
* Docker
* Docker compse

## **Installation**

Exécuteur :

git clone <https://github.com/MichDeRoanne/Project_RabbitMQ>

python -m venv venv

pip install -r requirements.txt

## **Configuration**

Une configuration du variables d'environnement dans le fichier .env, est nécessaire avant l'exécution de l'application, . Voici les variables à configurer :

# Utilisation du conteneur rabbit MQ

RABBIT\_USER=""

RABBIT\_PASSWORD=""

# Utilisation du conteneur MySQL

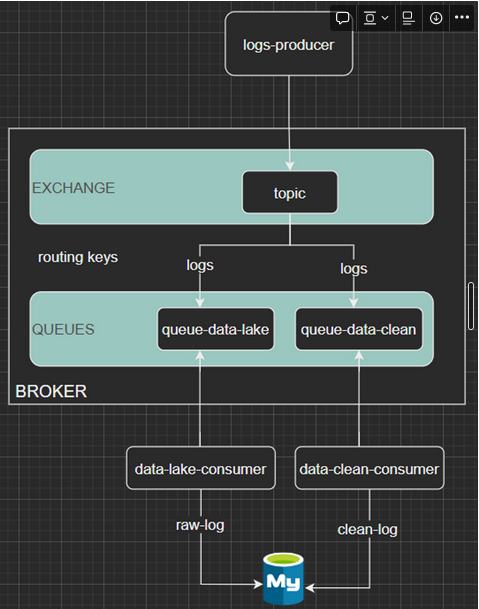
DB\_NAME=""

DB\_USER=""

DB\_PASSWORD=""

DB\_ROOT\_PASSWORD=""

## **Architecture**

[](https://user-images.githubusercontent.com/115103788/231180603-0db8c6c0-f124-4dae-98ee-7090916e9b6a.png)

### **L’application**

#### Etape 1 : RabbitMR

Exécuter le fichier python server.pyafin de créer une connexion au serveur via un channelafin qu'il soit utilisable par les producerset consummer.

python server.py

#### Etape 2 : le serveur web

Exécuter le fichier python logs-producer.pypour créer un exchangeet des queue. Cela permettra ensuite de publier les logs dans les queueet de les aiguiller correctement à l'aide de routing\_key.

python logs-producer.py

#### Etape 3 : la base SQL

Exécuter le fichier python consumer.pyafin de consommer les logs en attente dans nos deux queue. Les logs seront alors traités en FIFO (First In First Out) jusqu'à ce que les 2 queuesoient vides.

python consumer.py

Pour chaque file d'attente, nous appliquons des traitements particuliers avant de pousser un log dans la base de données :

* process\_msg\_data\_clean: permet de 'parser' notre journal à l'aide de fonctions et d'expressions régulières (RegEx) pour le stockage de manière déformée en base. Il pourra ainsi être facilement utilisé à des fins d'analyse, de tableau de bord ou de ML.
* process\_raw\_message: permet de stoker notre log brut afin de ne pas le perdre.

Toutes les fonctions de traitement sont dans le fichier\functions\Myfunctions.py

#### Etape 4 : Chargement des logs dans la base SQL

Pour interagir avec la base MySQL, nous utilisons SQLAlchemy, un ORM. Ce genre d'outil permet de simplifier grandement l'interface entre le code et une base de données.

La connexion à la base de données en python est effectuée dans le fichier\functions\orm\_conn.py

Les 2 tables de la base sont représentées dans le fichier\functions\models.py

Il s'agit de la dernière étape de notre flux de données : les données consommées sont ainsi poussées dans notre base MySQL.