



Université Chouaib Doukkali
Ecole Nationale des Sciences Appliquées d'El Jadida
Département Télécommunications, Réseaux et Informatique

Filière : **2ITE**
Niveau : **3^{ème} Année**
Sujet :

**Mise en place d'une architecture Big data en
utilisant apache Kafka, Apache Nifi,
Hbase, Spark streaming et Spark ML**

Réalisé Par :

- **LAAZIZ Ahmed**
- **MHANI Mohamed Amine**

Encadré par :
Prof. KALLOUBI

Ahmed Laaziz

Mohamed Amine Mhani

Projet 8 : Création d'un système pour le traitement dynamique et la prédiction en temps réel des données extraites des principaux sites web de crypto-monnaies en utilisant Apache Kafka, Apache Kafka, Spark streaming, Hive, Streamlit et Apache Airflow.

Table des matières

I. Objectif du projet.....	4
II. Prérequis pour le projet - Installation et Configuration.....	4
1. Environnement de Développement	4
2. Configuration de l'Environnement Dockerisé	5
a. Création des Conteneurs.....	5
b. Exécution des conteneurs	5
c. Exécution d'un conteneur spécifique.....	6
III. Collecte et traitement des données	7
1. Extraction et ingestion des données	7
2. Validation des données avec Apache Nifi.....	8
3. Création des tables requises.....	17
4. Traitement et stockage des données avec Spark streaming.....	18
IV. Entraînement et évaluation de plusieurs modèles machine learning	18
a. Mise en place	18

Ahmed Laaziz	Mohamed Amine Mhani
b. Code machine learning	19
c- Evaluation des modèles	19
V. Planification avec Apache Airflow	20
VI. Visualisation avec Streamlit.....	23

Réalisé par :

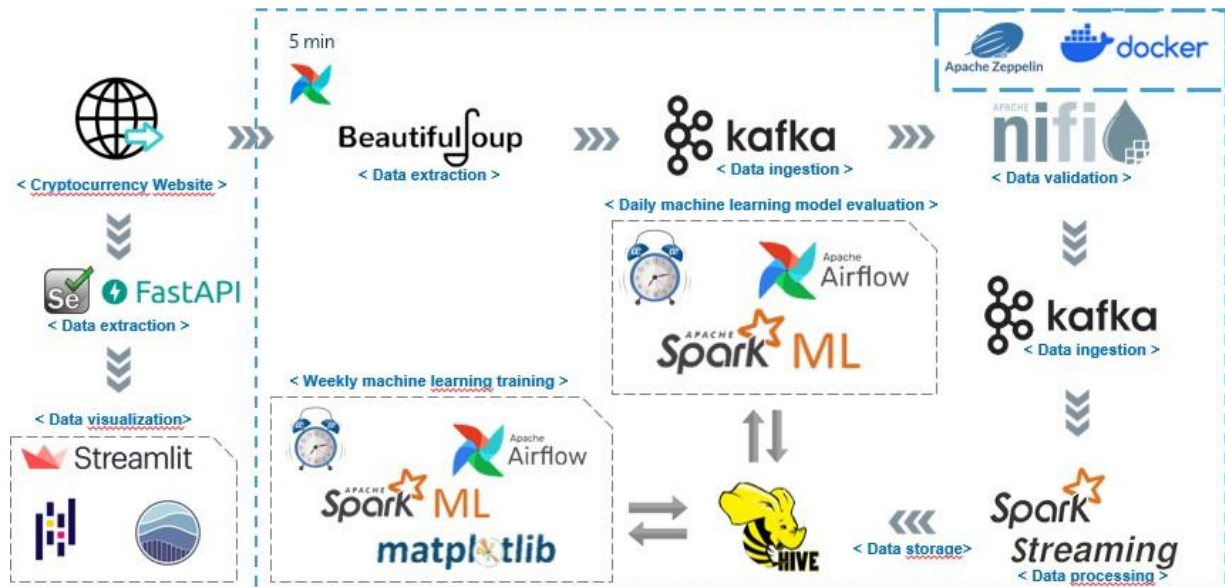
Ahmed Laaziz

Mohamed Amine Mhani

I. Objectif du projet

Notre projet vise à établir un système pour le traitement dynamique et la prédiction en temps réel des données extraites des principaux sites web de crypto-monnaies. L'objectif principal réside dans la création d'un système robuste et évolutif capable d'effectuer un traitement continu des données, de réaliser un auto-entraînement périodique (hebdomadaire) de plusieurs modèles d'apprentissage automatique, et de procéder à une auto-évaluation quotidienne pour identifier le modèle offrant les prédictions les plus précises.

Architecture



II. Prérequis pour le projet - Installation et Configuration

Ce projet nécessite l'installation et la configuration préalable de plusieurs outils et environnements. Suivez attentivement ces étapes pour garantir un déroulement sans accroc du projet.

1. Environnement de Développement

- Système d'Exploitation :** Vérifiez que vous disposez d'un système d'exploitation compatible avec les outils nécessaires (Exemple : Ici, on utilise Windows).
- RAM :** Assurez-vous d'avoir une mémoire RAM supérieure à 13 Go.
- Docker :**
 - Téléchargez et installez Docker en suivant les instructions spécifiques à votre système d'exploitation : [lien vers Docker](#)
 - Vérifiez l'installation avec la commande : `docker --version`
- Docker Compose :**

Réalisé par :

Ahmed Laaziz

Mohamed Amine Mhani

- Téléchargez et installez Docker Compose en suivant les instructions spécifiques à votre système d'exploitation : [lien vers Docker Compose](#)
- Vérifiez l'installation avec la commande : `docker-compose --version`

2. Configuration de l'Environnement Dockerisé

a. Création des Conteneurs

- Élaborez un fichier `docker-compose.yml` décrivant les services nécessaires pour cette application. Se référer au fichier `docker-compose.yml` dans le dossier du projet

b. Exécution des conteneurs

Pour démarrer les conteneurs définis dans votre fichier **docker-compose.yml** et les exécuter en arrière-plan, utilisez la commande suivante : `docker-compose up -d`

Une fois les conteneurs lancés, vous pouvez lister les identifiants des conteneurs en cours d'exécution et leurs ports associés en utilisant la commande : `docker ps`

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS
fb1a1b8f3e05	confluentinc/cp-enterprise-control-center:5.4.0	"/etc/confluent/docker..."	25 hours ago	Up 2 minutes	0.0.0.0:9021->9021/tcp
6d7e36565c82	confluentinc/cp-schema-registry:5.4.0	"/etc/confluent/docker..."	28 hours ago	Up 41 seconds	8081/tcp, 0.0.0.0:8083->8083/tcp
7e8f048174a5	confluentinc/cp-server:5.4.0	"/etc/confluent/docker..."	28 hours ago	Up About a minute	0.0.0.0:9092->9092/tcp, 0.0.0.0:29092->29092/tcp
3852bb975387	mrugankray/hive-server-sqoop:1.0	"entrypoint.sh /bin/..."	28 hours ago	Up About a minute	0.0.0.0:10000->10000/tcp, 10002/tcp
cd5662e42d2e	confluentinc/cp-zookeeper:5.4.0	"/etc/confluent/docker..."	28 hours ago	Up About a minute	2888/tcp, 0.0.0.0:2181->2181/tcp, 3888/tcp
f215981ebb1a	bde2020/hive:2.3.2-postgresql-metastore	"entrypoint.sh /opt/..."	28 hours ago	Up About a minute	10000/tcp, 0.0.0.0:9083->9083/tcp, 10002/tcp
13e26d8b50cf	mrugankray/nodemanager-python:1.0	"entrypoint.sh /run..."	28 hours ago	Up 2 minutes (healthy)	0.0.0.0:8042->8042/tcp, 0.0.0.0:19888->19888/tcp, 8088/tcp
90019b42a6ef	mrugankray/resourcemanager-python:1.0	"entrypoint.sh /run..."	28 hours ago	Up About a minute (healthy)	8042/tcp, 0.0.0.0:8088->8088/tcp
3bd1e3cd869	mrugankray/namenode-spark-airflow-flume-zeppelin:1.1	"entrypoint.sh /sta..."	28 hours ago	Up 2 minutes (healthy)	0.0.0.0:3000->3000/tcp, 0.0.0.0:4040->4040/tcp, 0.0.0.0:8080->8080/tcp
f83cec515deb	bde2020/hive-metastore-postgresql:2.3.0	"/docker-entrypoint..."	28 hours ago	Up About a minute	5432/tcp
78bc0c746935	bde2020/hadoop-historyserver:2.0.0-hadoop3.2.1-java8	"entrypoint.sh /run..."	28 hours ago	Up 2 minutes (healthy)	0.0.0.0:8188->8188/tcp
ef1e03c91d17	mrugankray/datanode-python:1.0	"entrypoint.sh /run..."	28 hours ago	Up 2 minutes (healthy)	0.0.0.0:9864->9864/tcp

Figure 1.2 Affichage de la liste des conteneurs

Pour accéder aux services dans un navigateur web, utilisez les URL suivantes :

Réalisé par :

Ahmed Laaziz

Mohamed Amine Mhani

- Zeppelin : <http://localhost:8082>
- Nifi : <http://localhost:9999>
- Confluent : <http://localhost:9021>
- Spark : <http://localhost:8080>
- Airflow : <http://localhost:3000>
- Namenode : <http://localhost:9870>

Assurez-vous que les services sont correctement démarrés et que les ports spécifiés dans votre fichier **docker-compose.yml** ne sont pas utilisés par d'autres applications sur votre système.

c. Exécution d'un conteneur spécifique

Pour accéder au shell d'un conteneur spécifique en mode bash, utilisez la commande suivante, en remplaçant **id_container** par l'identifiant du conteneur souhaité :

`docker exec -it id_container /bin/bash`

```
C:\Users\abouelkhir>docker exec -it 3852bb975387 /bin/bash
root@3852bb975387:/# hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.log4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/opt/hive/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

Figure 1.3 Exécution d'un conteneur spécifique

Ahmed Laaziz

Mohamed Amine Mhani

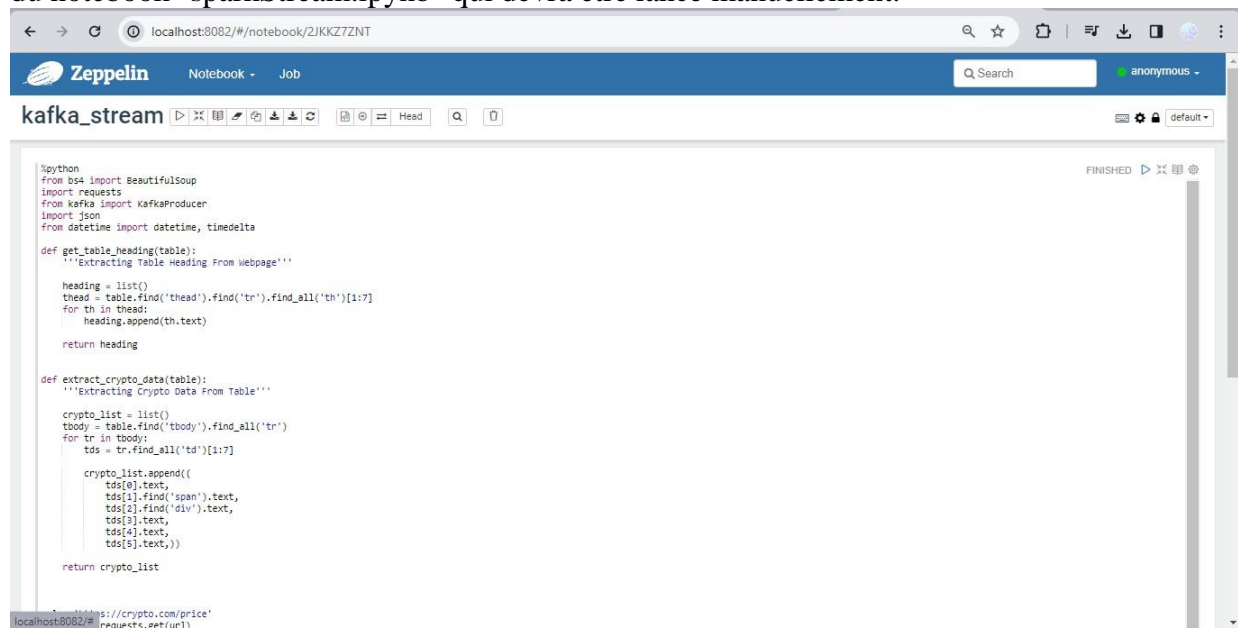
III. Collecte et traitement des données

1. Extraction et ingestion des données

Pour notre projet, on va utiliser BeautifulSoup, une bibliothèque Python, pour récupérer des données du site <https://crypto.com/price> toutes les 5 minutes. Code :

Pour accéder à l'interface graphique de Zeppelin, cliquez sur ce lien : <http://localhost:8082>. Une fois là-bas, créez un nouveau notebook. Ensuite, copiez le code du fichier nommé "kafka_stream" pour extraire les données et les ingérer avec Kafka.

Veuillez noter que l'exécution automatique des codes se fera via Apache Airflow, à l'exception du notebook "sparkStream.ipynb" qui devra être lancé manuellement.



```
python
from bs4 import BeautifulSoup
import requests
from kafka import KafkaProducer
import json
from datetime import datetime, timedelta

def get_table_heading(table):
    """Extracting Table Heading From Webpage"""
    heading = list()
    thead = table.find('thead').find('tr').find_all('th')[1:7]
    for th in thead:
        heading.append(th.text)
    return heading

def extract_crypto_data(table):
    """Extracting Crypto Data From Table"""
    crypto_list = list()
    tbody = table.find('tbody').find_all('tr')
    for tr in tbody:
        tds = tr.find_all('td')[1:7]
        crypto_list.append((
            tds[0].text,
            tds[1].find('span').text,
            tds[2].find('div').text,
            tds[3].text,
            tds[4].text,
            tds[5].text,))
    return crypto_list

# Example usage
url = 'https://crypto.com/price'
response = requests.get(url)
```

Figure 3.1 Code pour l'extraction et l'ingestion des données

Réalisé par :

Ahmed Laaziz

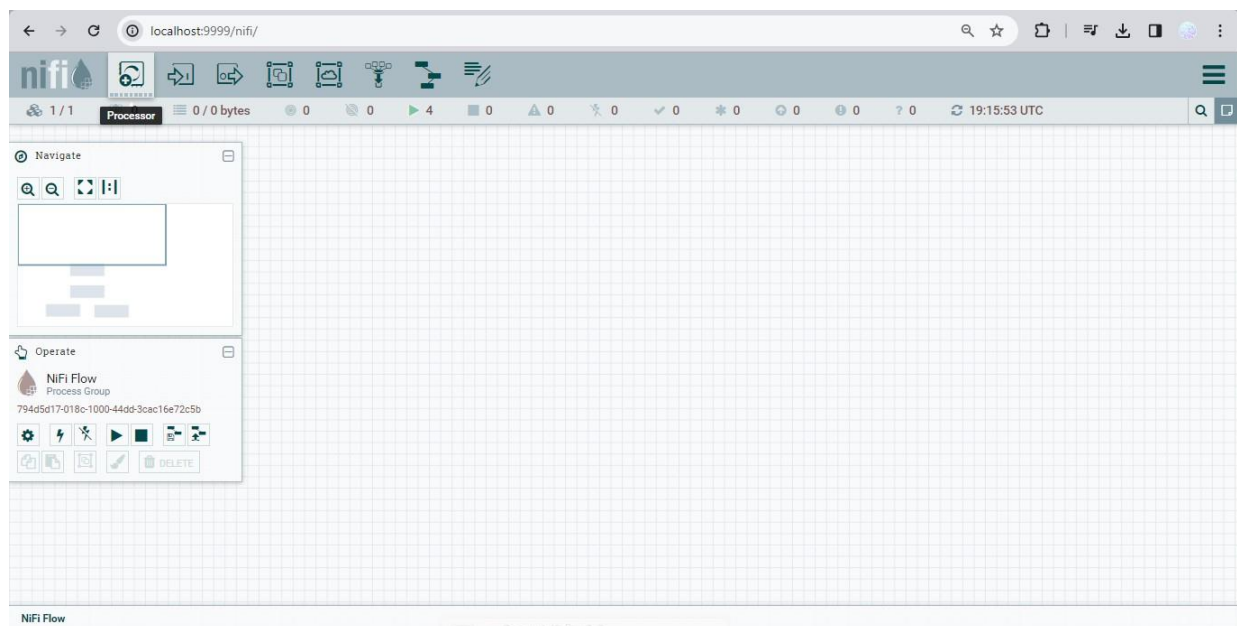
Mohamed Amine Mhani

2. Validation des données avec Apache Nifi

Apache NiFi assure la validation des données en temps réel en appliquant des filtres, des règles et des contrôles de qualité pour garantir l'exactitude, la sécurité et la cohérence des données transitant à travers les flux, permettant ainsi un traitement fiable et conforme aux exigences définies.

Afin de valider les données en provenance du topic kafka, vous devez :

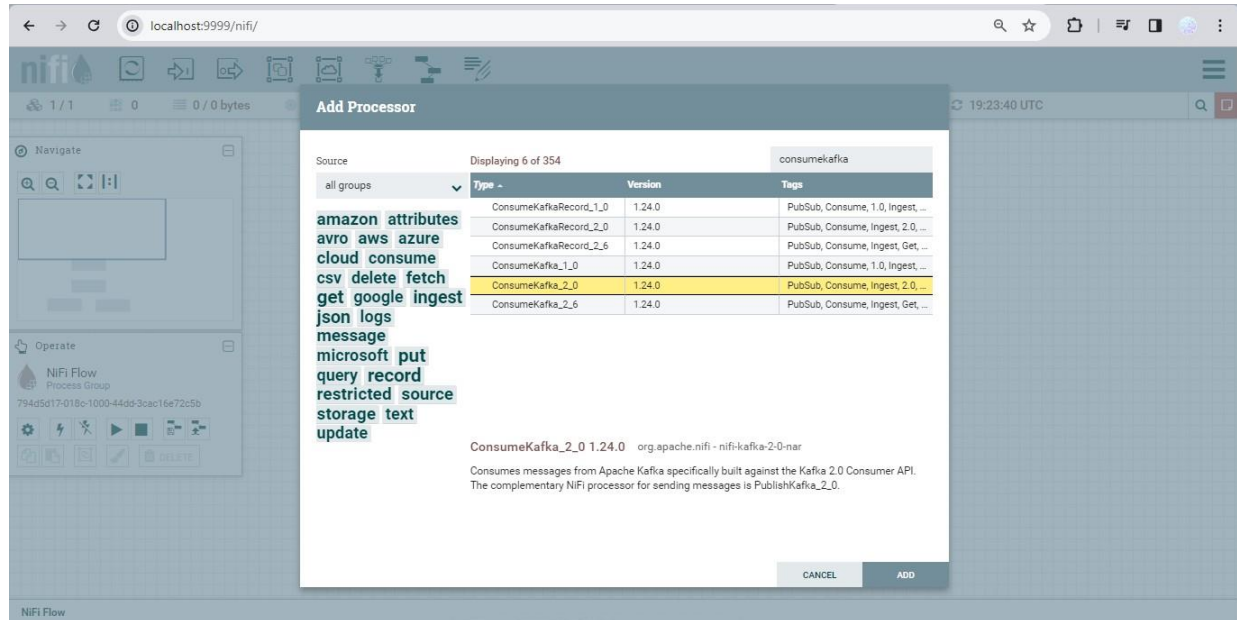
- Accéder à l'interface graphique d'apache Nifi sur le lien suivant :
[http://localhost: 9999](http://localhost:9999)



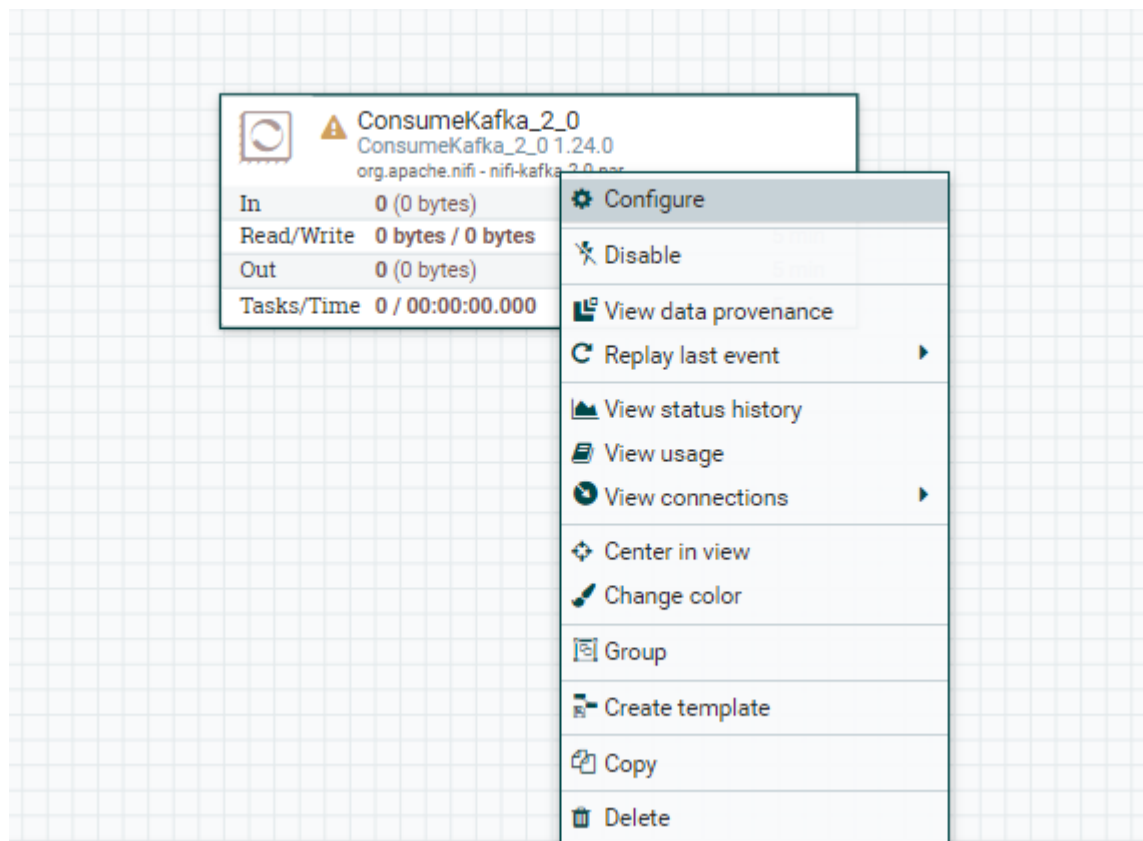
- Glissez un processeur de type consumeKafka_2_0.

Ahmed Laaziz

Mohamed Amine Mhani



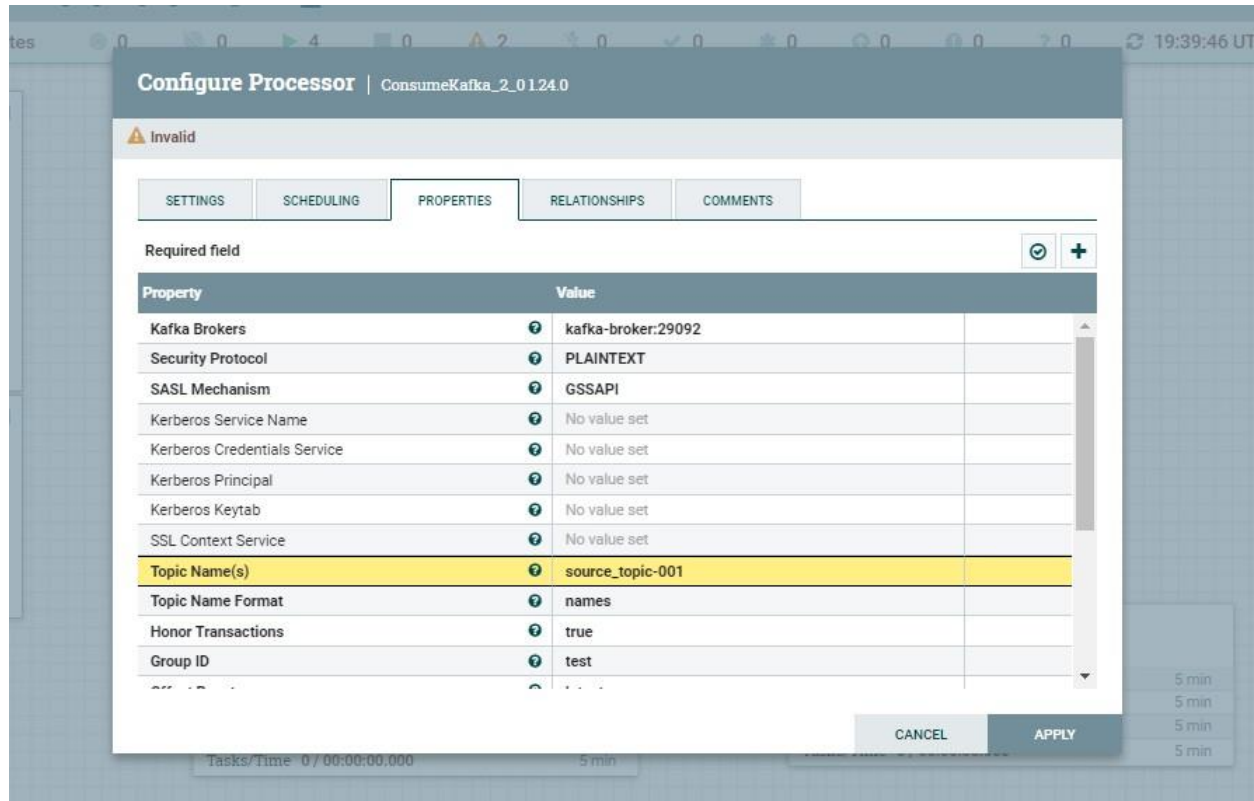
- Cliquez droit → configure pour configurer votre processeur.



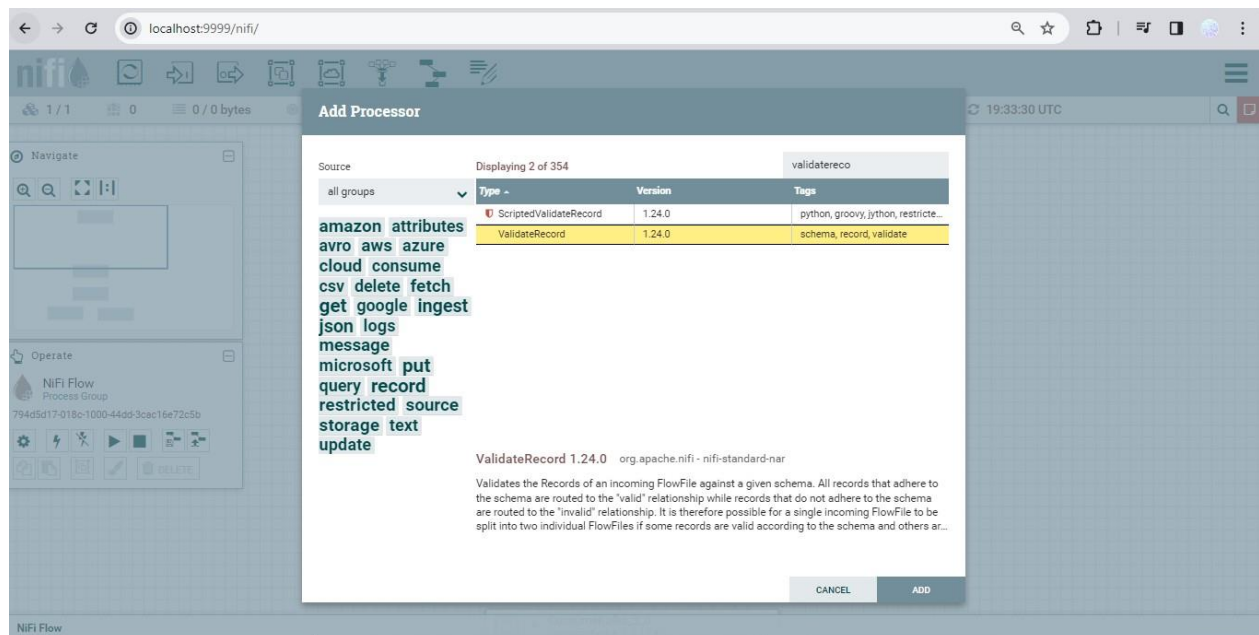
- Veuillez paramétrer votre processeur de la manière suivante.

Ahmed Laaziz

Mohamed Amine Mhani



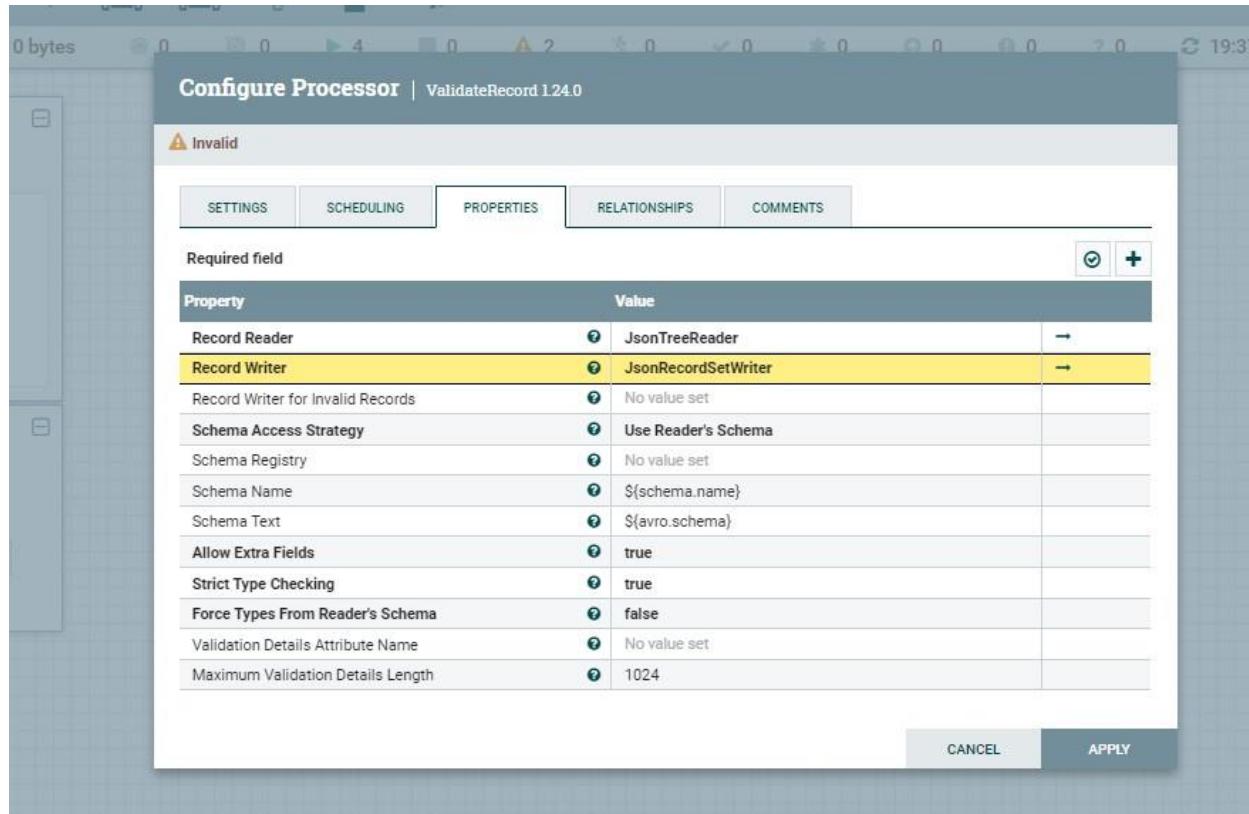
- Une fois que vous avez appuyé sur "APPLY", procédez à faire glisser un autre processeur de type **ValidateRecord**.



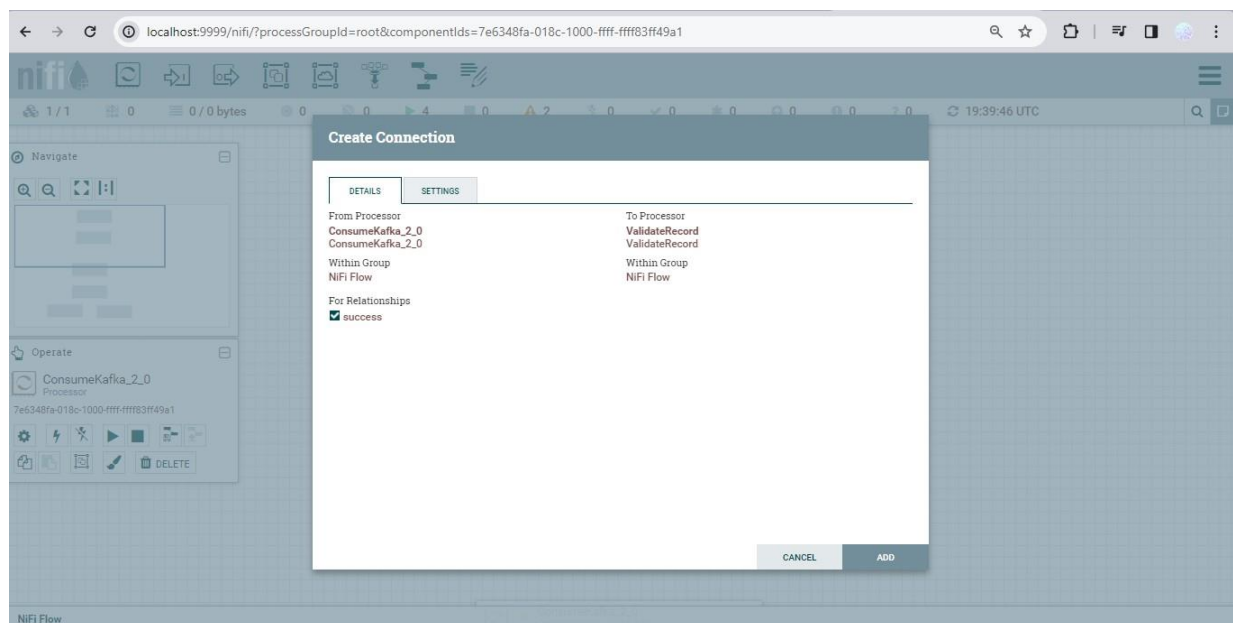
- Cliquez sur **ADD**, ensuite configurez le processeur.

Ahmed Laaziz

Mohamed Amine Mhani



- Cliquez sur **APPLY**, par la suite créez une connexion entre les deux processeurs.

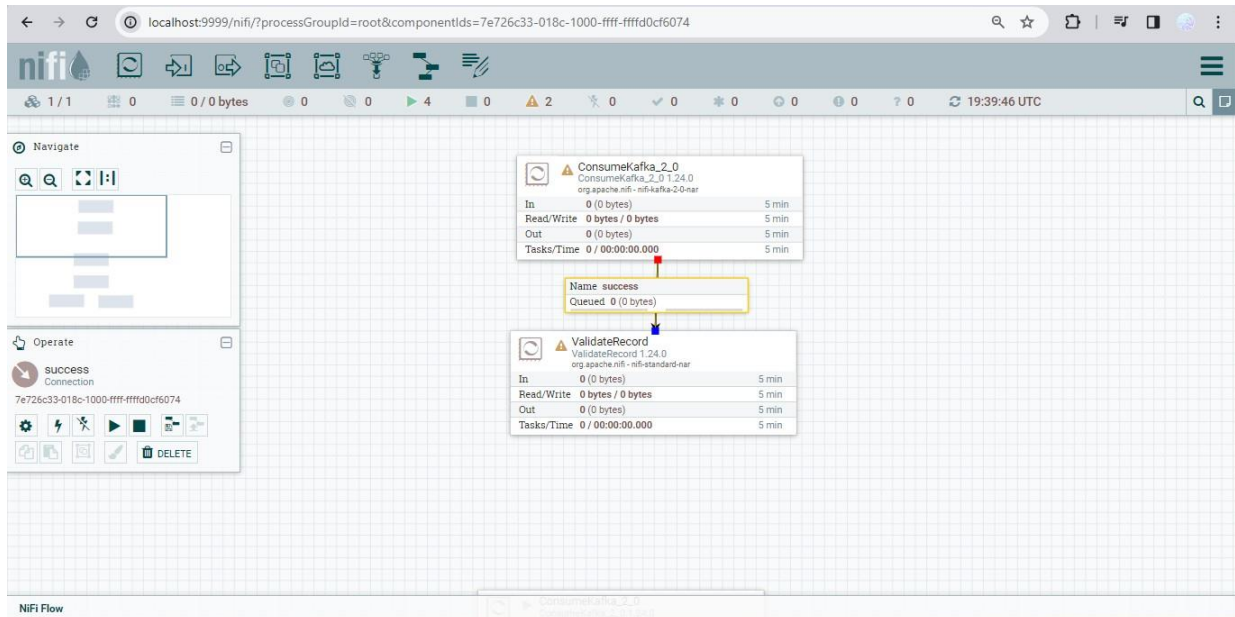


Réalisé par :

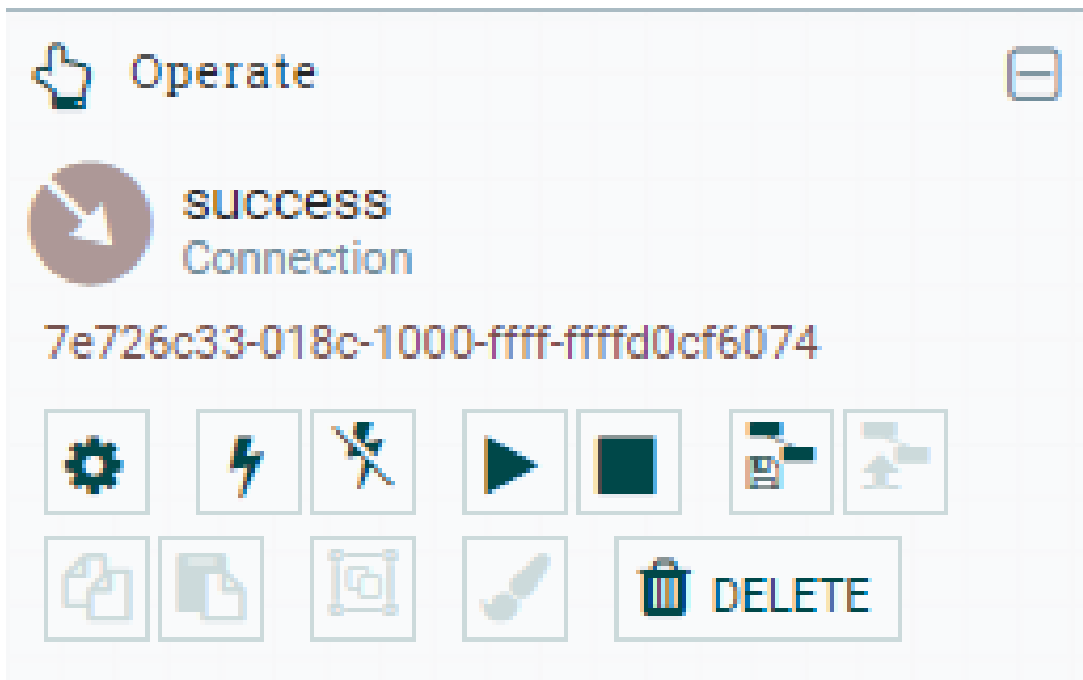
Ahmed Laaziz

Mohamed Amine Mhani

- Cliquez sur **ADD**.



- Si une erreur est survenue lors de l'établissement de la connexion, veuillez-vous diriger vers le bouton de configuration.

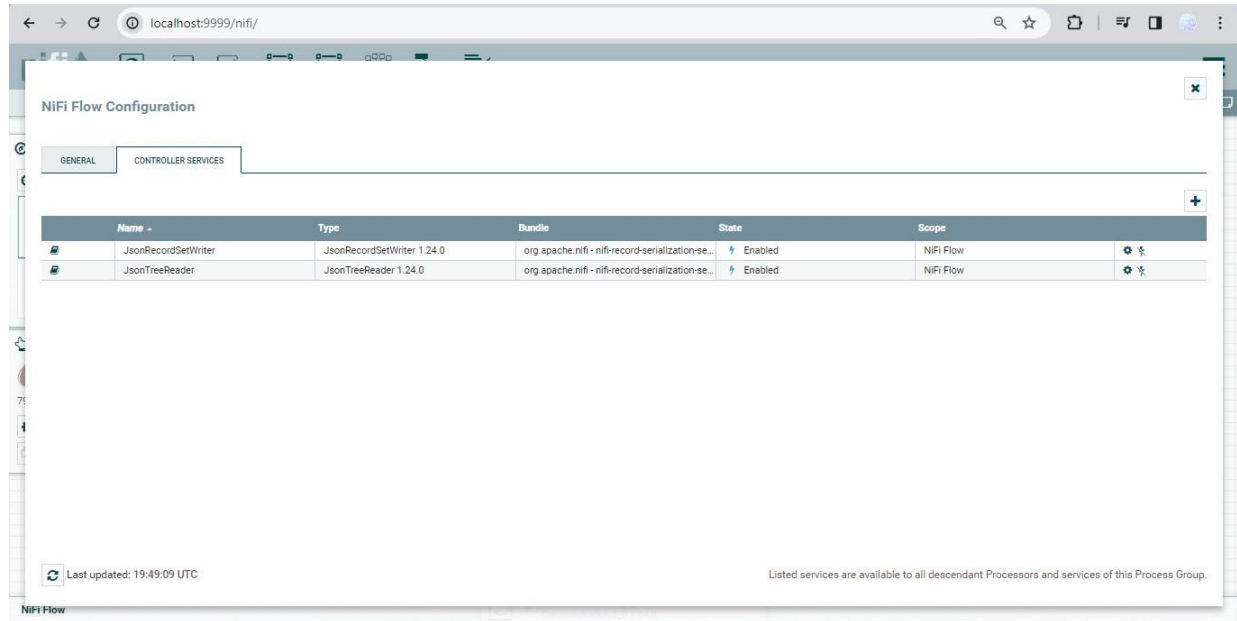


Réalisé par :

Ahmed Laaziz

Mohamed Amine Mhani

- Activez le **JsonRecordSetWriter** et le **JsonTreeReader**.

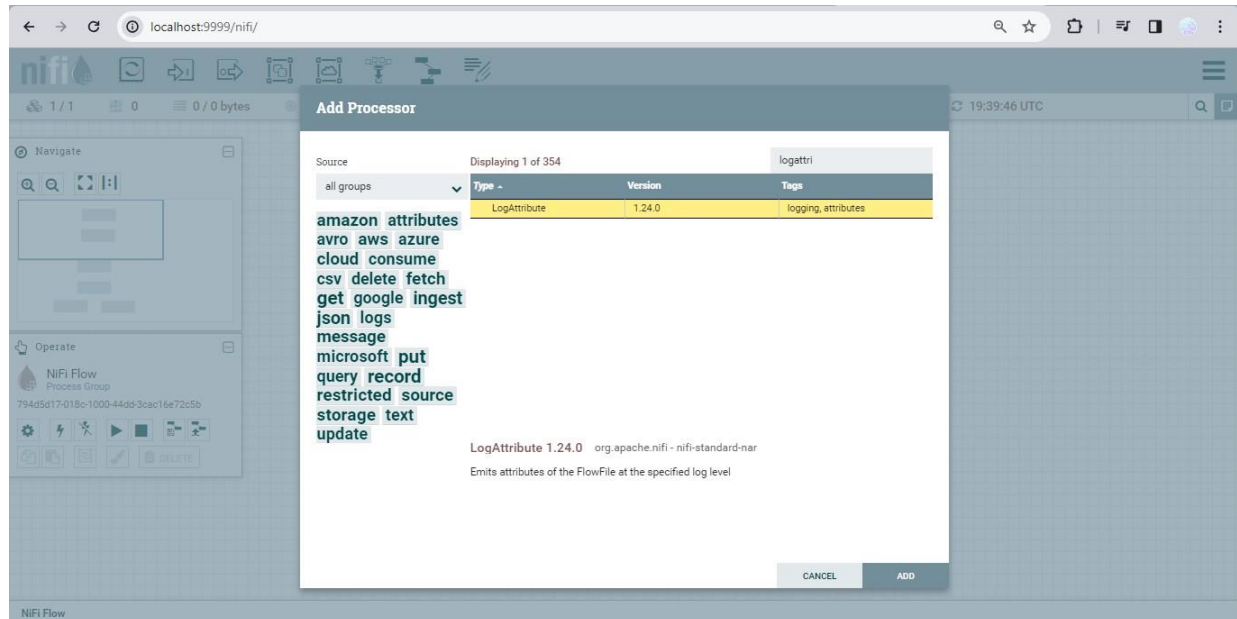


- Une fois que la connexion a été établie avec succès, faites glisser un processeur de type **LogAttribute**.

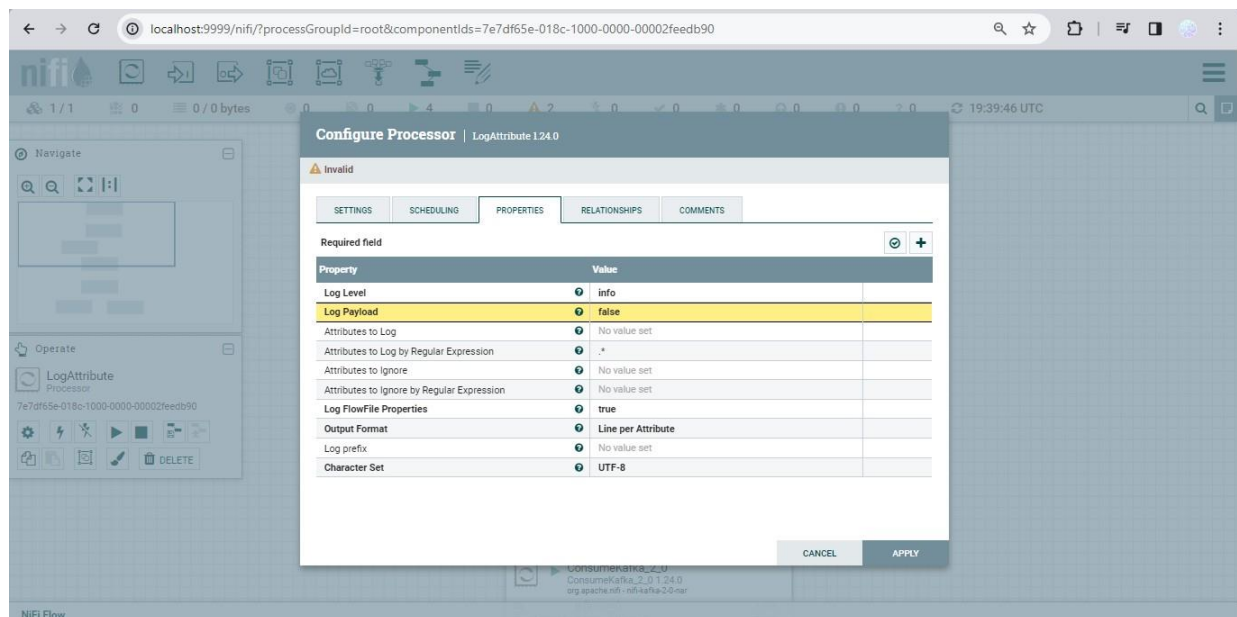
Réalisé par :

Ahmed Laaziz

Mohamed Amine Mhani



- Configurez-le comme suite.

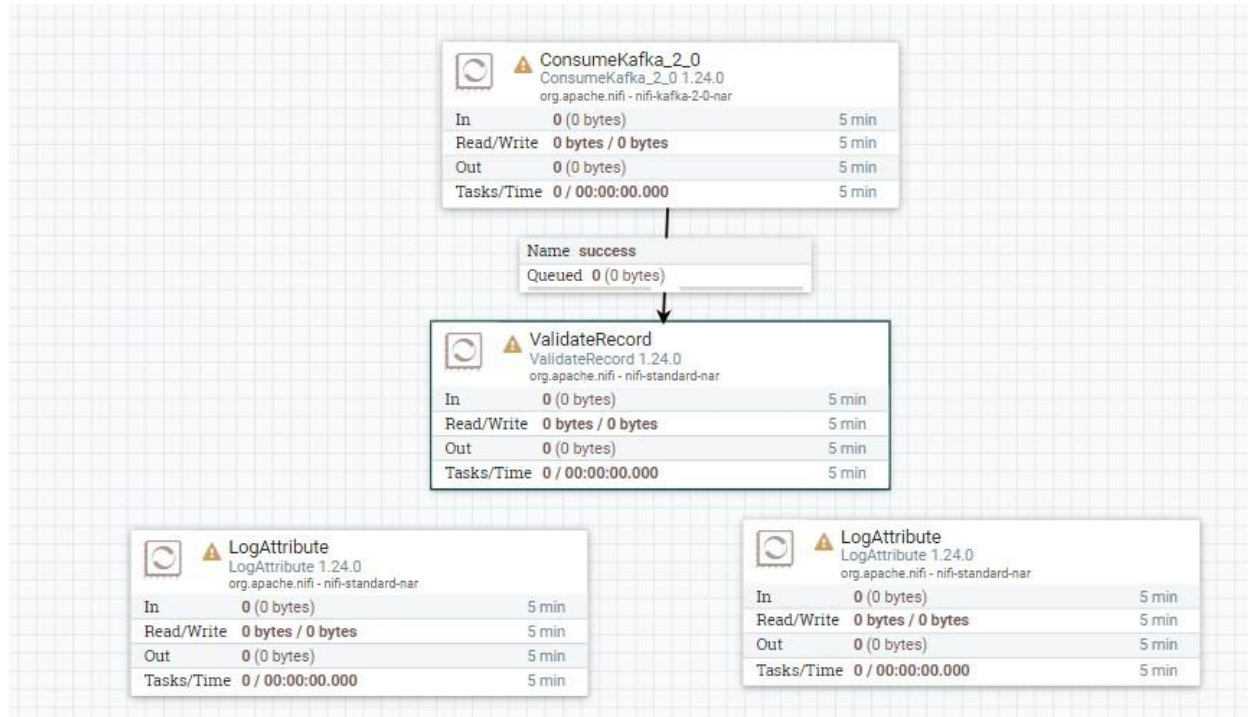


- Cliquez sur APPLY et dupliquez-le.

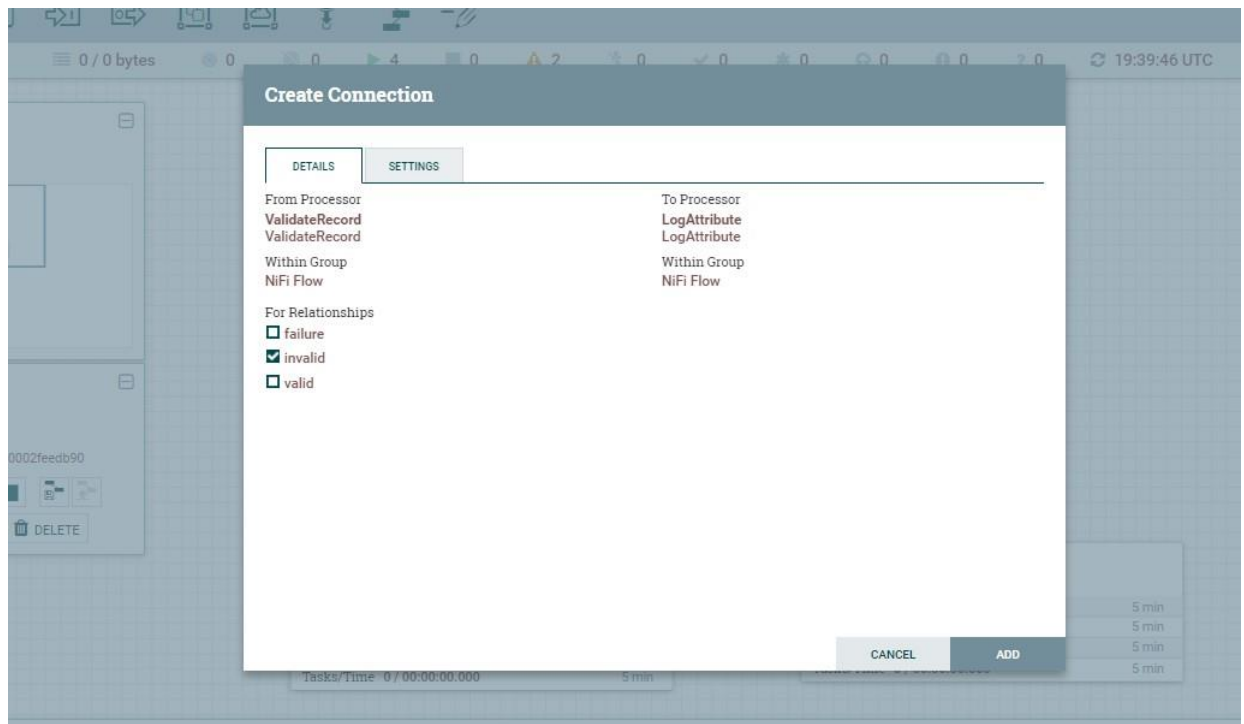
Réalisé par :

Ahmed Laaziz

Mohamed Amine Mhani



- Ensuite connectez les deux processeurs avec le processeur précédent.

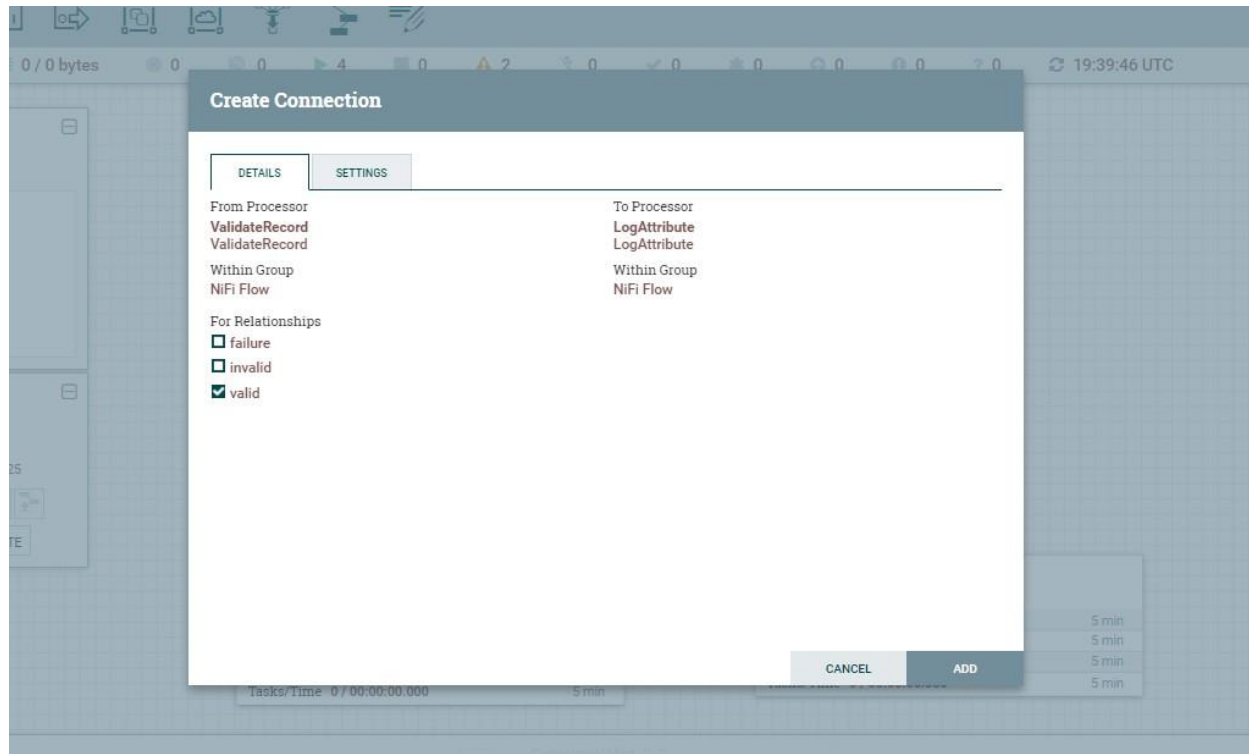


γ Même chose pour le deuxième processeur.

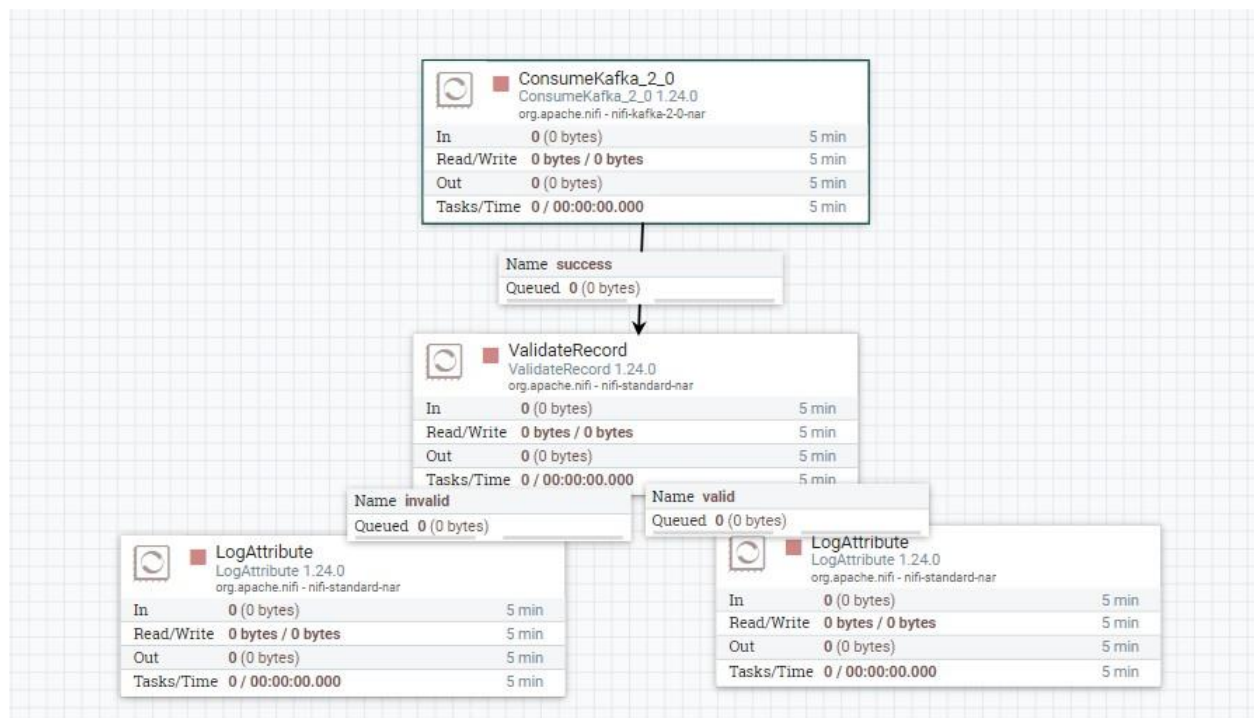
Réalisé par :

Ahmed Laaziz

Mohamed Amine Mhani



- Votre schéma de traitement devrait se présenter de la manière suivante.



- Lancez votre pipeline de validation.

Ahmed Laaziz

Mohamed Amine Mhani

3. Création des tables requises

Pour accéder au shell d'un conteneur spécifique en mode bash, utilisez la commande suivante, en remplaçant **id_container** par l'identifiant du conteneur souhaité :

`docker exec -it id_container /bin/bash`

```
C:\Users\labouelkhir>docker exec -it 3852bb975387 /bin/bash
root@3852bb975387:/# hive
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop-2.7.4/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in file:/opt/hive/conf/hive-log4j2.properties Async: true
Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
hive>
```

Figure 1.3 Exécution d'un conteneur spécifique

Après cela, exécutez la commande suivante pour créer la table "crypto_data1", qui va stocker les données provenant du topic Kafka :

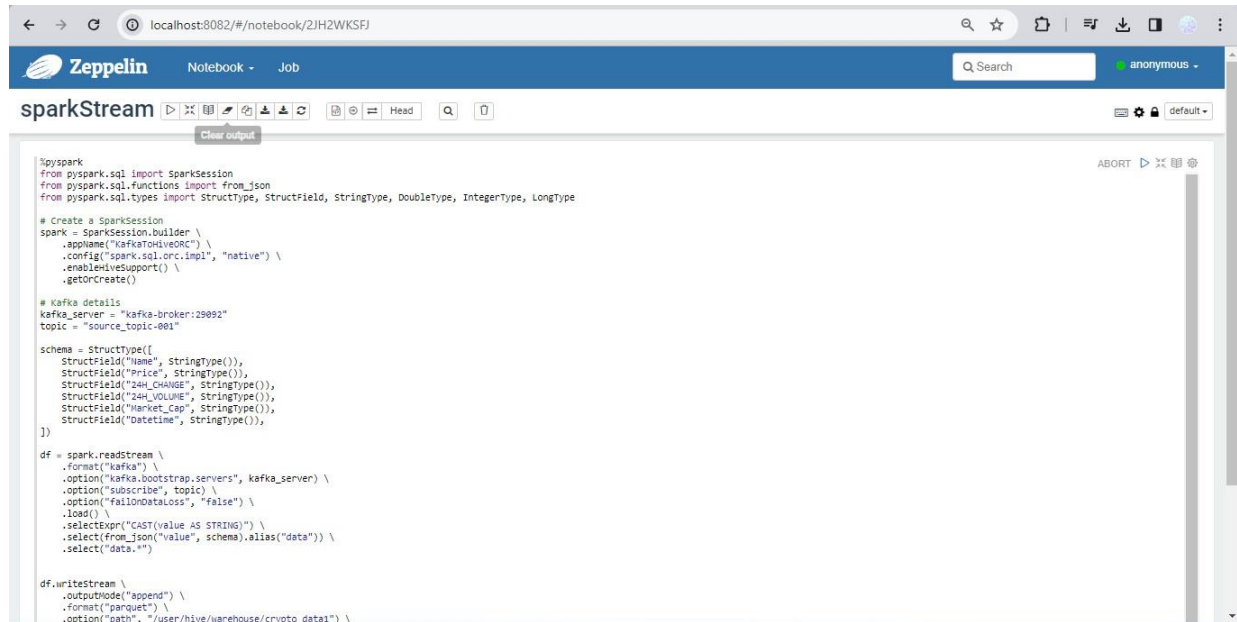
```
CREATE TABLE IF NOT EXISTS crypto_data1 (
  Name STRING,
  Price STRING,
  24H_CHANGE STRING,
  24H_VOLUME STRING,
  Market_Cap STRING,
  Datetime STRING
)
STORED AS Parquet;
```

Ahmed Laaziz

Mohamed Amine Mhani

4. Traitement et stockage des données avec Spark streaming

Pour effectuer le traitement, importez le notebook appelé "sparkStream.ipynb".



```

%pySpark
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json
from pyspark.sql.types import StructType, StructField, StringType, DoubleType, IntegerType, LongType

# Create a SparkSession
spark = SparkSession.builder \
    .appName("sparkStreaming") \
    .config("spark.sql.orc.impl", "native") \
    .enableHiveSupport() \
    .getOrCreate()

# Kafka details
kafka_server = "kafka-broker:29092"
topic = "source_topic-001"

schema = StructType([
    StructField("name", StringType()),
    StructField("price", StringType()),
    StructField("24H_CHANGE", StringType()),
    StructField("24H_VOLUME", StringType()),
    StructField("Market_Cap", StringType()),
    StructField("datetime", StringType())
])

df = spark.readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", kafka_server) \
    .option("subscribe", topic) \
    .option("failOnDataLoss", "false") \
    .load() \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json("value", schema).alias("data")) \
    .select("data.*")

df.writeStream \
    .outputMode("append") \
    .format("parquet") \
    .option("path", "/user/hive/warehouse/crypto_data1") \
    .start()
    
```

IV. Entraînement et évaluation de plusieurs modèles machine learning

a. Mise en place

Avant de procéder à l'entraînement et à l'évaluation des modèles, veuillez créer les tables suivantes en utilisant les commandes suivantes :

```

CREATE TABLE models_infos_table (
    name STRING,
    training_date STRING
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe';
    
```

```

CREATE TABLE models_testing_infos_table (
    rmse STRING,
    testing_date STRING
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe';
    
```

Ahmed Laaziz

Mohamed Amine Mhani

La première table, **Table1**, stockera des détails sur les meilleurs modèles sauvegardés chaque semaine dans le système de fichiers HDFS.

La seconde table, **Table2**, enregistrera les performances des meilleurs modèles entraînés chaque semaine.

b. Code machine learning

Vous trouverez le code du traitement machine learning dans le dossier partagé sous le nom de machine_learning.ipynp. Veuillez l'importer dans Zeppelin.

```

%spark
from pyspark.ml.feature import VectorAssembler, StringIndexer, MinMaxScaler
from pyspark.ml.regression import RandomForestRegressor
from pyspark.ml import Pipeline
from pyspark.sql.functions import col, sum
from pyspark.sql.functions import regexp_replace, col, hour, to_timestamp, udf
from pyspark.sql.functions import year, month, daymonth, hour

%spark
parquet_df = spark.read.parquet("/user/hive/warehouse/crypto_data/part-*.parquet")

# Perform operations on the read Dataframe parquet_df
# For example, you can show the content of the Dataframe
parquet_df.show()
  
```

Name	Price	24H_CHANGE	24H_VOLUME	Market_Cap	Datetime
XMR	\$168.49	-0.87%	\$77.7 M	\$9.1 B	2023-12-18T00:01:...
BTC	\$20.07	-3.21%	\$144.7 M	\$2.9 B	2023-12-18T00:01:...
FIL	\$5.43	-5.05%	\$485.95 M	\$2.61 B	2023-12-18T00:01:...
HBAR	\$0.07065	-2.07%	\$59.69 M	\$2.64 B	2023-12-18T00:01:...
CRO	\$0.05047	-2.74%	\$8.99 M	\$2.6 B	2023-12-18T00:01:...
INO	\$20.64	+2.82%	\$189.29 M	\$2.57 B	2023-12-18T00:01:...
INX	\$1.99	-5.65%	\$129.83 M	\$2.56 B	2023-12-18T00:01:...
APT	\$0.11	-1.83%	\$146.75 M	\$2.47 B	2023-12-18T00:01:...
TUSD	\$0.9971	-0.00%	\$110.45 M	\$2.44 B	2023-12-18T00:01:...
KAS	\$0.1083	-8.13%	\$67.72 M	\$2.38 B	2023-12-18T00:01:...
NEAR	\$2.29	-4.02%	\$179.64 M	\$2.3 B	2023-12-18T00:01:...
VET	\$0.02985	-1.71%	\$42.93 M	\$2.16 B	2023-12-18T00:01:...
TIA	\$13.00	-1.94%	\$310.32 M	\$1.97 B	2023-12-18T00:01:...
OP	\$2.09	-5.07%	\$196.95 M	\$1.9 B	2023-12-18T00:01:...

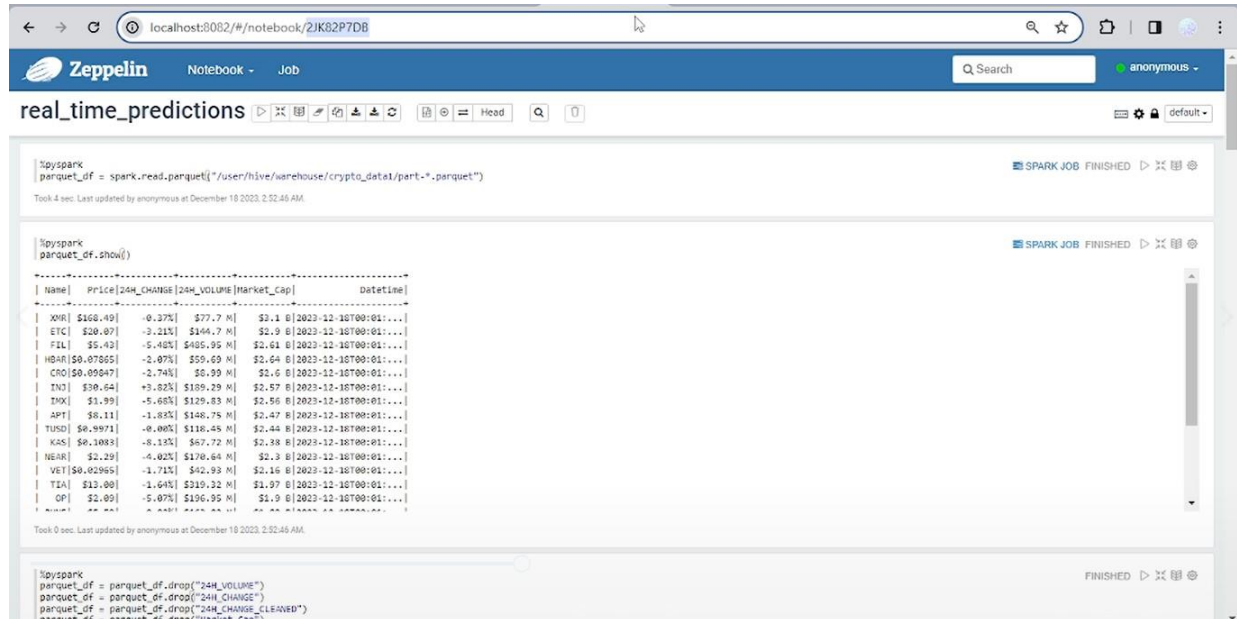
c- Evaluation des modèles

Évaluer quotidiennement un modèle de machine learning permet de garantir sa fiabilité et son adaptation continue. Cette pratique permet de surveiller la performance du modèle, de détecter rapidement tout déclin dans ses prédictions, et d'ajuster promptement les paramètres en cas de changements dans les données ou d'anomalies. Cela favorise une amélioration continue, assurant ainsi que le modèle reste précis et efficace dans son application quotidienne.

Vous allez trouver le code de l'évaluation continue des modèles dans le dossier partagé sous le nom real_time_predictions.ipynp importez le sur zeppelin.

Ahmed Laaziz

Mohamed Amine Mhani



V. Planification avec Apache Airflow

Airflow est un outil puissant pour la planification et l'automatisation des flux de travail. Il permet de définir, planifier et gérer des pipelines de données complexes, en orchestrant les tâches et en gérant leurs dépendances. En utilisant des DAGs (Directed Acyclic Graphs), Airflow permet de spécifier l'ordre et les relations entre différentes tâches, ce qui facilite l'exécution séquentielle ou parallèle de ces tâches. Il offre également des fonctionnalités pour surveiller l'avancement des tâches, gérer les erreurs et les reprises, tout en offrant une vue d'ensemble complète sur l'état et la performance des flux de travail.

Airflow est livré avec une interface utilisateur qui vous permet de voir ce que font les DAG et leurs tâches, de déclencher des exécutions de DAG, d'afficher les journaux et d'effectuer un débogage et une résolution limités des problèmes avec vos DAG.

Pour se connecter à l'interface utilisez les identifiants suivants :

- Nom d'utilisateur : admin
- Mot de passe : admin

Ahmed Laaziz

Mohamed Amine Mhani

DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
Extract_data_DAG	airflow	1/1	@daily	2023-12-18, 00:55:00	2023-12-18, 01:00:00	1/1
Weekly_machine_learning_training_DAG	airflow	1/1	@weekly	2023-12-18, 01:23:44	2023-12-17, 00:00:00	1/1
daily_machine_learning_evaluation_DAG	airflow	1/1	@daily	2023-12-18, 01:32:18	2023-12-18, 00:00:00	1/1
example_bash_operator	airflow	1/1	@daily	2023-12-17, 00:00:00		1/1

Afin d'automatisez votre pipeline :

- Utilisez votre éditeur de code préféré (comme VSCode, PyCharm, etc.).
- Créez un nouveau dossier appelé "dags" à l'emplacement où vous souhaitez stocker vos DAGs.
- À l'intérieur du dossier "dags", créez trois fichiers Python pour vos DAG.
- Dans ces fichiers, écrivez le code décrivant les différentes étapes du processus.

Vous trouverez les codes dans les fichiers (dag1.py, dag2.py, dag3.py) dans le dossier du projet, copier et coller les codes dans vos fichiers et faites les installations nécessaires.

- Faites toutes les installations de packages nécessaires avec pip ou si vous utiliser PyCharm à partir de l'interpréteur.
- Dans le code du dag **modifier** les adresses des notebooks selon les id.

Réalisé par :

Ahmed Laaziz

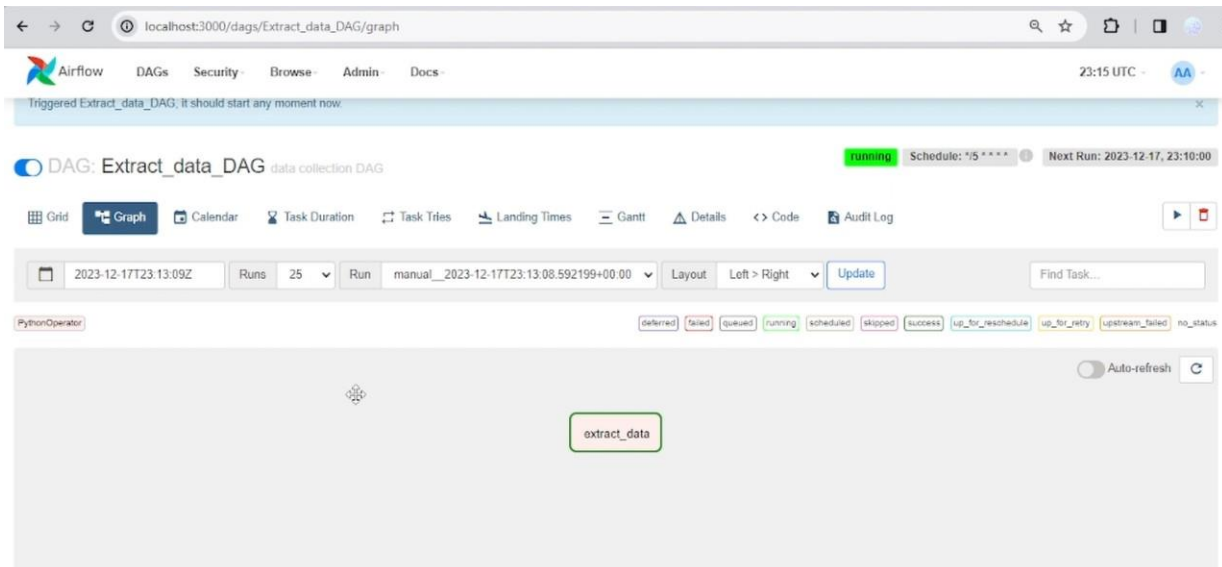
Mohamed Amine Mhani

```

1 from datetime import timedelta
2 from airflow import DAG
3 from airflow.utils.dates import days_ago
4 from airflow.operators.python import PythonOperator
5 import requests
6
7 # Default DAG arguments
8 default_args = {
9     'owner': 'airflow',
10    'depends_on_past': False,
11    'start_date': days_ago(1),
12    'retries': 1,
13    'retry_delay': timedelta(minutes=1),
14 }
15
16
17 def execute_zeppelin_notebook_1(**kwargs):
18     # Zeppelin API endpoint to execute a notebook
19     zeppelin_api_url = 'http://localhost:8082/api/notebook/job/23KKZ7ZNT'
20
21     # Make a POST request to execute the Zeppelin notebook
22     response = requests.post(zeppelin_api_url)
23
24     # Check if the request was successful (HTTP status code 200)
25     if response.status_code == 200:
26         print("Zeppelin notebook executed successfully 1.")
    
```

Activez vos dags dans l'ordre suivant :

1- « Extract data dag »

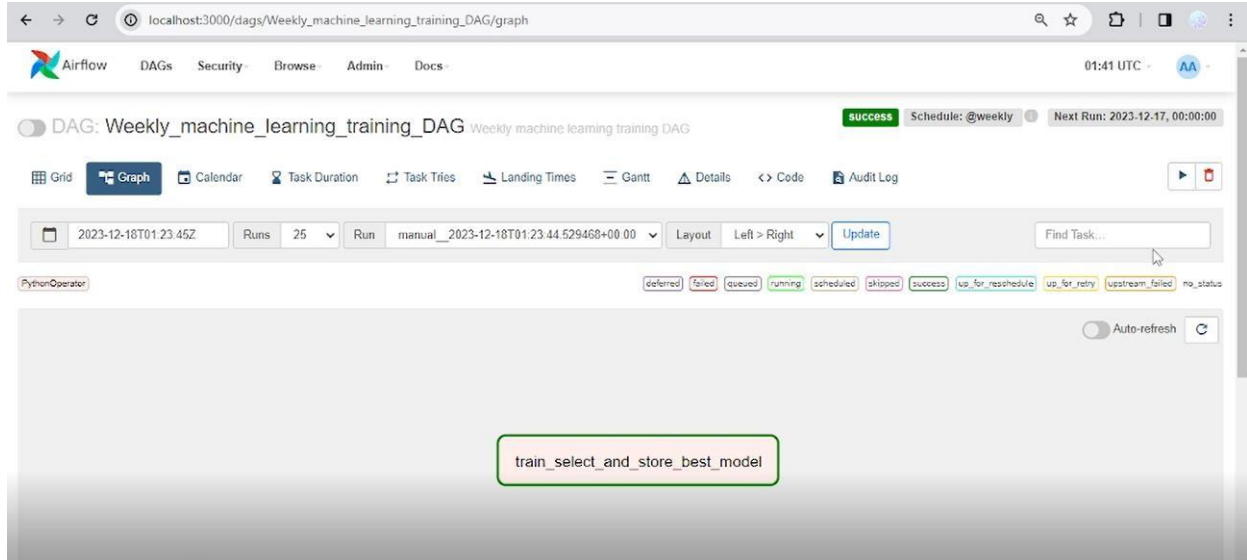


2- « Weekly machine learning training dag »

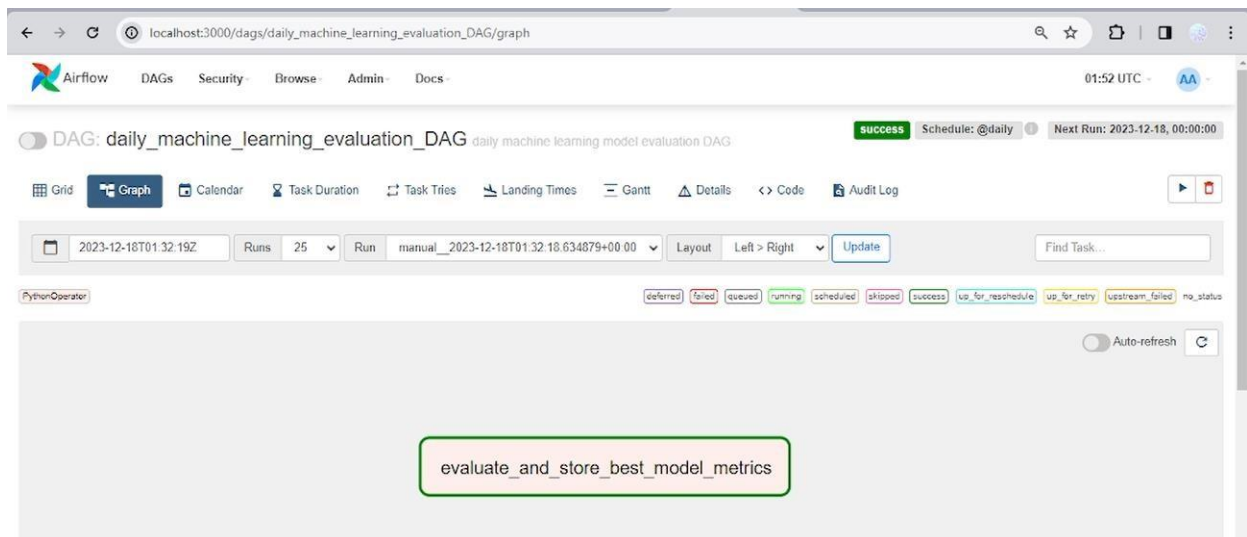
Réalisé par :

Ahmed Laaziz

Mohamed Amine Mhani



3- « daily machine learning evaluation dag »



Pour vérifier le bon fonctionnement du code, utilisez des commandes HQL :

(Exemple) select * from <table name>

VI. Visualisation avec Streamlit :

Streamlit est une puissante bibliothèque Python qui simplifie considérablement le processus de création d'applications web interactives pour la visualisation de données. Conçu pour être convivial et efficace, Streamlit permet aux développeurs de générer rapidement des tableaux de bord dynamiques sans avoir à se plonger dans des détails complexes de codage.

Dans cette section, nous explorerons comment intégrer Streamlit dans vos pipelines de données et tirer

Ahmed Laaziz

Mohamed Amine Mhani

parti de ses fonctionnalités pour créer des interfaces utilisateur interactives. Nous examinerons comment présenter vos données de manière attractive, tout en offrant aux utilisateurs la possibilité d'explorer et d'interagir avec les résultats.

Afin de visualiser les données en temps réel de streaming :

- Utilisez votre éditeur de code préféré (comme VSCode, PyCharm, etc.).
 - Ouvrir le projet **Visual_Stream**.
 - Exécutez les fichiers **.py** dans l'ordre suivant :
1. Le premier fichier **scrapper.py** qui est le responsable d'extraction des données depuis la table des crypto-currencies à travers **BeautifulSoup** et **Selenium** :

```

1  from selenium import webdriver
2  from selenium.webdriver.chrome.service import Service as ChromeService
3  from selenium.webdriver.chrome.options import Options
4  from bs4 import BeautifulSoup
5  from pprint import pprint
6
7  import time
8  import json
9  import datetime
10
11  usage
12  def get_table_heading(table):
13      """Extracting Table Heading From Webpage"""
14      heading = list()
15      thead = table.find('thead').find('tr').find_all('th')[1:7]
16      for th in thead:
17          heading.append(th.text)
18      return heading
19
20  usage
21  def extract_crypto_data(table):
22      """Extracting Crypto Data From Table"""
23      crypto_list = list()
24      tbody = table.find('tbody').find_all('tr')
25      for tr in tbody:
26          tds = tr.find_all('td')[1:7]
27
28          crypto_list.append((
29              tds[0].text,
30              tds[1].find('span').text,
31              tds[2].find('div').text,
32              tds[3].text,
33              tds[4].text,
34              tds[5].text,))
35
36  return {"heading": heading, "crypto_data": crypto_list}

```

2. Le deuxième fichier **server.py** va nous permettre de rendre les données extraites sous forme d'une **API** en utilisant **FastAPI** :

Ahmed Laaziz

Mohamed Amine Mhani

```

1  from fastapi import FastAPI
2  from fastapi.responses import JSONResponse
3  import uvicorn
4  import time
5  import json
6
7  app = FastAPI()
8
9  #usage
10 def get_latest_scraped_data():
11     while True:
12         try:
13             with open("scraped_data.json", "r") as f:
14                 data = json.load(f)
15                 yield data
16
17         except Exception as e:
18             yield {"error": f"Error reading scraped data: {e}"}
19
20         time.sleep(10) # Adjust the sleep time based on the update frequency
21
22 @app.get("/get_latest_scraped_data")
23 def get_latest_scraped_data_endpoint():
24     return JSONResponse(content=next(get_latest_scraped_data()))
25
26 if __name__ == "__main__":
27     uvicorn.run(app)
    
```

3. Les fichiers **writer.py** et **overwriter.py** vont servir à stocker les données dans des fichiers csv qu'on va utiliser dans la visualisation :

- **Writer.py :**

```

1  import csv
2  import requests
3  import time
4  import threading
5
6  # CSV file name
7  csv_file_name = 'crypto_data.csv'
8
9  #usage
10 def fetch_open_meteo_data():
11     api_url = "http://127.0.0.1:8000/get_latest_scraped_data"
12     while True:
13         try:
14             response = requests.get(api_url)
15
16             if response.status_code == 200:
17                 data = response.json()
18                 write_to_csv(data)
19             else:
20                 print(f"Error fetching data from API. Status code: {response.status_code}")
21
22         except Exception as e:
23             print(f"Error fetching data from API: {e}")
24
25         time.sleep(2)
26
27 #usage
28 def write_to_csv(data):
29     with open(csv_file_name, mode='a', newline='', encoding='utf-8') as file:
30         writer = csv.writer(file)
31
32         # Check if the file is empty, if so, write the header
33         if file.tell() == 0:
34             writer.writerow(data['heading'] + ['timestamp'])
    
```

- **Overwriter.py :**

Ahmed Laaziz

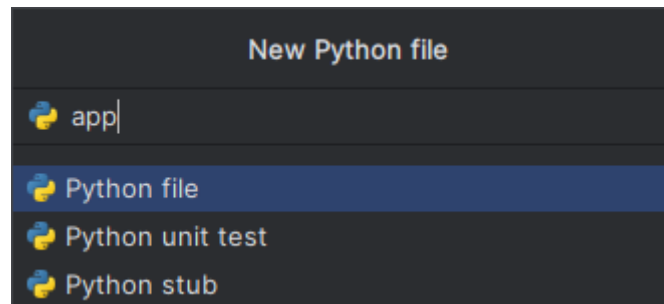
Mohamed Amine Mhani

```

1 import csv
2 import requests
3 import time
4 import threading
5
6 # CSV file name
7 csv_file_name = 'crypto_cont.csv'
8
9 1 usage
10 def fetch_open_meteo_data():
11     api_url = "http://127.0.0.1:8000/get_latest_scraped_data"
12     while True:
13         try:
14             response = requests.get(api_url)
15
16             if response.status_code == 200:
17                 data = response.json()
18                 write_to_csv(data)
19             else:
20                 print(f"Error fetching data from API. Status code: {response.status_code}")
21
22         except Exception as e:
23             print(f"Error fetching data from API: {e}")
24
25     time.sleep(2)
26
27 1 usage
28 def write_to_csv(data):
29     with open(csv_file_name, mode='w', newline='', encoding='utf-8') as file:
30         writer = csv.writer(file)
31
32         # Write the header to the CSV file
33         writer.writerow(data['heading'] + ['timestamp'])
34
35         # Write the crypto data to the CSV file

```

- Créez un fichier **app.py** pour l'interface **streamlit** :



- Dans le fichier **app.py** on doit définir les composant suivants :
 - La session de note streamlit application on va stocker des informations de session :

```

1 usage
19 class SessionState:
20     def __init__(self):
21         self.real = True
22         self.currency = ""
23
24 1 usage
25 def get_session_state():
26     if "session_state" not in st.session_state:
27         st.session_state.session_state = SessionState()
28     return st.session_state.session_state
29
30 session_state = get_session_state()

```

Ahmed Laaziz

Mohamed Amine Mhani

- Une fonction **read_crypto_data** pour lire les données depuis les fichiers csv :

```
32 3 usages
33 def read_crypto_data(file):
34     try:
35         crypto_data = pd.read_csv(file)
36         return crypto_data
37     except FileNotFoundError:
38         st.error("CSV file not found.")
39         return pd.DataFrame()
40     except pd.errors.EmptyDataError:
41         st.error("CSV file is empty or has no valid data.")
42         return pd.DataFrame()
43     except Exception as e:
44         st.error(f"Error reading CSV file: {e}")
45         return pd.DataFrame()
46
47 # Defining positions
48 crypto_currencies = read_crypto_data('crypto_cont.csv')['Name'].unique()
49 crypto_currencies = np.insert(crypto_currencies, obj: 0, values: "-- Currencies Ranking")
```

- Le composant **streamlit sidebar** :

```
50 with st.sidebar:
51     st.header("Crypto Currency")
52     selected_currency = st.selectbox("Choose the crypto currency :", crypto_currencies)
53
54     if selected_currency == "-- Currencies Ranking":
55         session_state.real = True
56         session_state.currency = ""
57     else:
58         session_state.real = False
59         session_state.currency = selected_currency
60
```

- Le composant **streamlit container** :

Ahmed Laaziz

Mohamed Amine Mhani

```

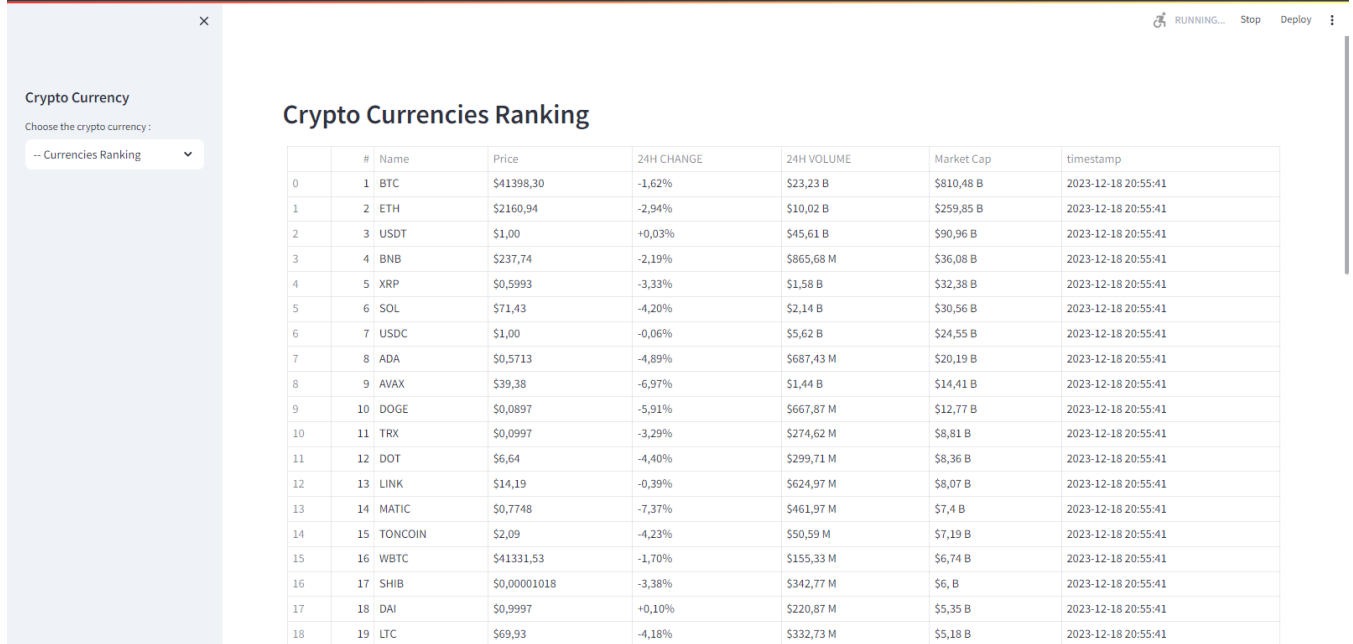
61 with st.container():
62     if session_state.real == True:
63         st.header("Crypto Currencies Ranking")
64
65         # Initialize an empty container for the table
66         crypto_table_container = st.empty()
67
68         while True:
69             # Read data from the CSV file
70             crypto_data = read_crypto_data('crypto_cont.csv')
71
72             # Display the table with dynamically updating data
73             crypto_table_container.table(crypto_data)
74
75             # Allow Streamlit to update every 2 seconds
76             time.sleep(2)
77     else:
78         st.header(f"{session_state.currency} Evolution")
79         crypto_data = read_crypto_data('crypto_data.csv')
80
81         # Filter data for the selected cryptocurrency
82         selected_crypto_data = crypto_data[crypto_data['Name'] == session_state.currency]
83
84         # Reverse the order of y-axis values
85         reversed_prices = selected_crypto_data.set_index('timestamp')['Price'][::-1]
86
87         # Plot the evolution of the cryptocurrency's price over time with height set to 500
88         st.line_chart(reversed_prices, height=500)
89
    
```

- Maintenant qu'on a créé notre application **streamlit** on peut lancer cette application à travers la commande :
 - **streamlit run app.py**
 - L'application sera accessible sur l'adresse : <http://localhost:8501/>

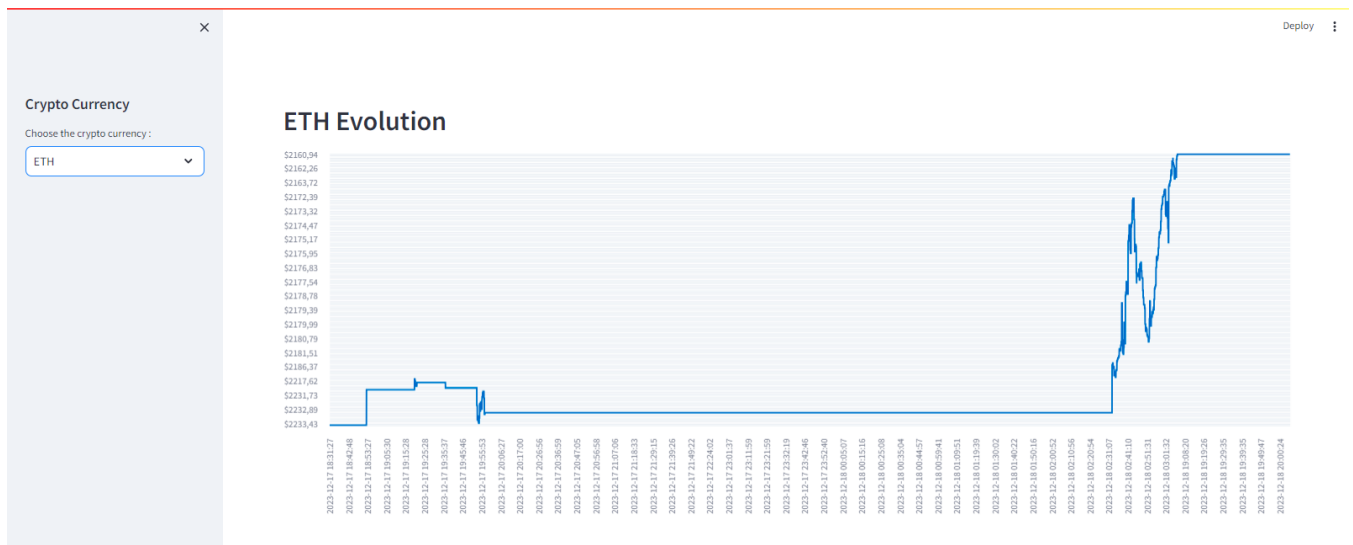
Réalisé par :

Ahmed Laaziz

Mohamed Amine Mhani



- La première page est dédiée à la présentation du **ranking** des crypto-monnaies en temps réel.



- La deuxième page est pour le tracking de l'évolution des prix des différentes crypto-monnaies.