

СИСТЕМА ДЛЯ ЗБОРУ, ОБРОБКИ ТА АНАЛІЗУ ДАНИХ ІЗ СОЦМЕРЕЖІ TWITTER У РЕАЛЬНОМУ ЧАСІ.

ФЕЩЕНКО ЯНА

МІЩЕНКО ФЕДІР

ПІДЛЕТЕЙЧУК ЮРІЙ

Дані будуть збиратись через Twitter API, а саме інформація про користувача (ім'я, опис профілю, локація,) та текст самого твіту

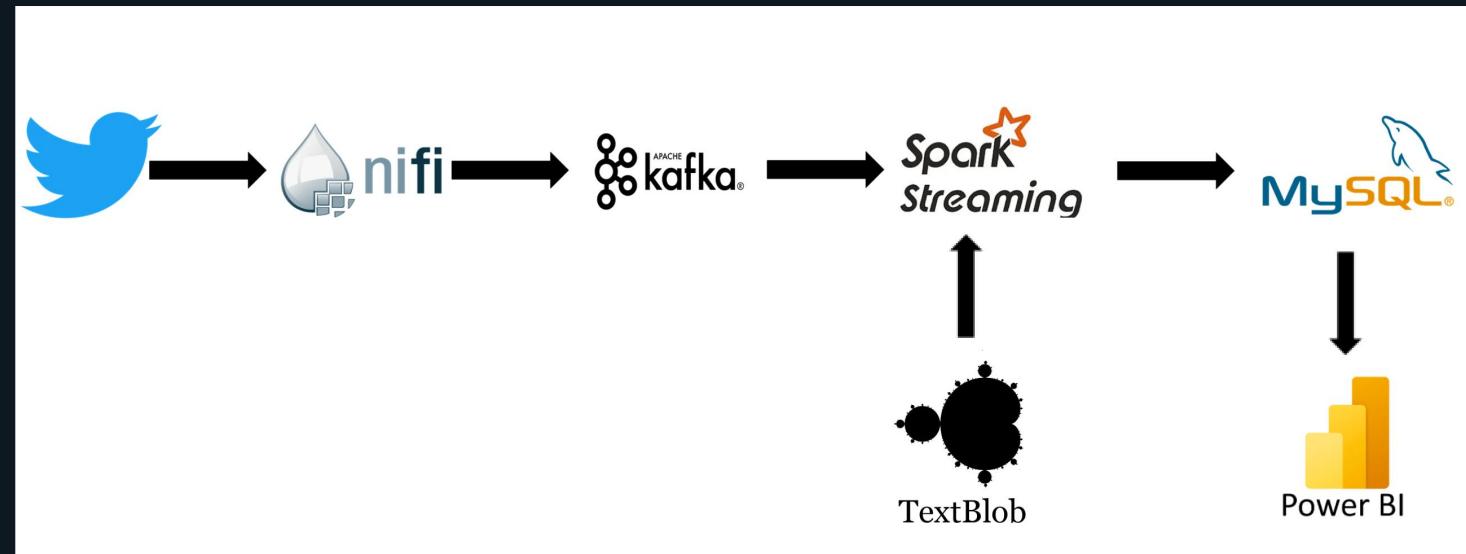
Передача даних до Apache Kafka та налаштування пайплайну у Apache NiFi

Первинний опис системи

Обробка отриманих даних,
виконання сентимент-аналізу через
Spark та збереження даних у СУБД

Побудова дашборду для
результативних даних з
різноманітними візуалізаціями

Блок-схема пайплайну. Версія №1



Проте в процесі виконання проєкту наша команда зіткнулась з різними труднощами, в результаті яких були проведені зміни щодо основного стеку технологій, які будуть використовуватись, але про це розповімо більш детально.

Apache NiFi

Apache Nifi - досить потужний інструмент, який використовується для управління потоками даних між різними системами у режимі реального часу. Наша команда планувала налаштувати пайплайн у цій системі задля автоматизації процесу. Проте основні труднощі, з якими ми зіткнулися:

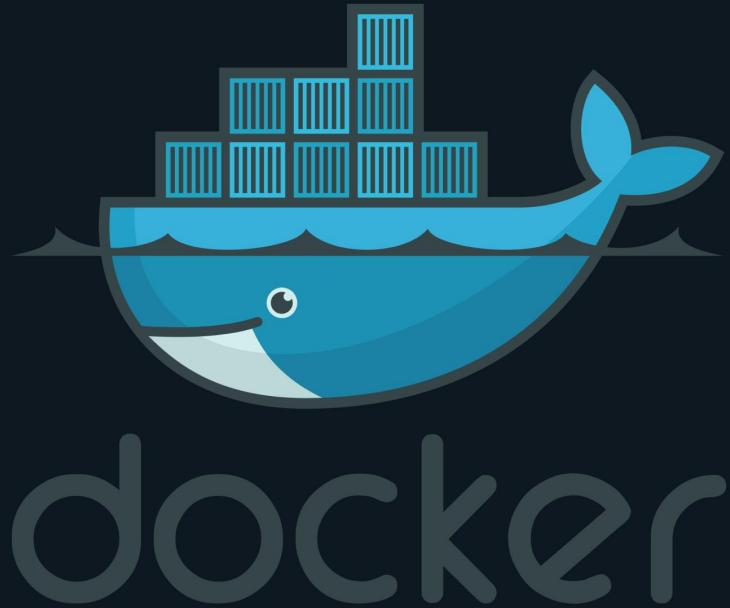
Складний інтерфейс
користувача

Складність використання
документації

Мала кількість гайдів та
відповідних проектів

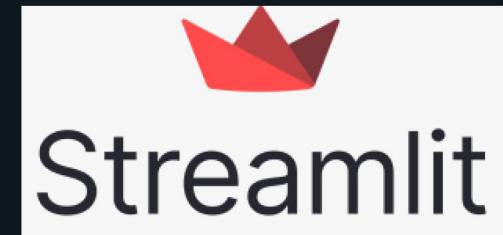
Тож наша команда прийняла рішення не використовувати даний інструмент,

Натомість, ми вирішили розібратись із системою **Docker** - інструмент для контейнеризації, що дозволяє ізолювати додатки та їх залежності, забезпечуючи легке розгортання, масштабування і управління в будь-якому середовищі.

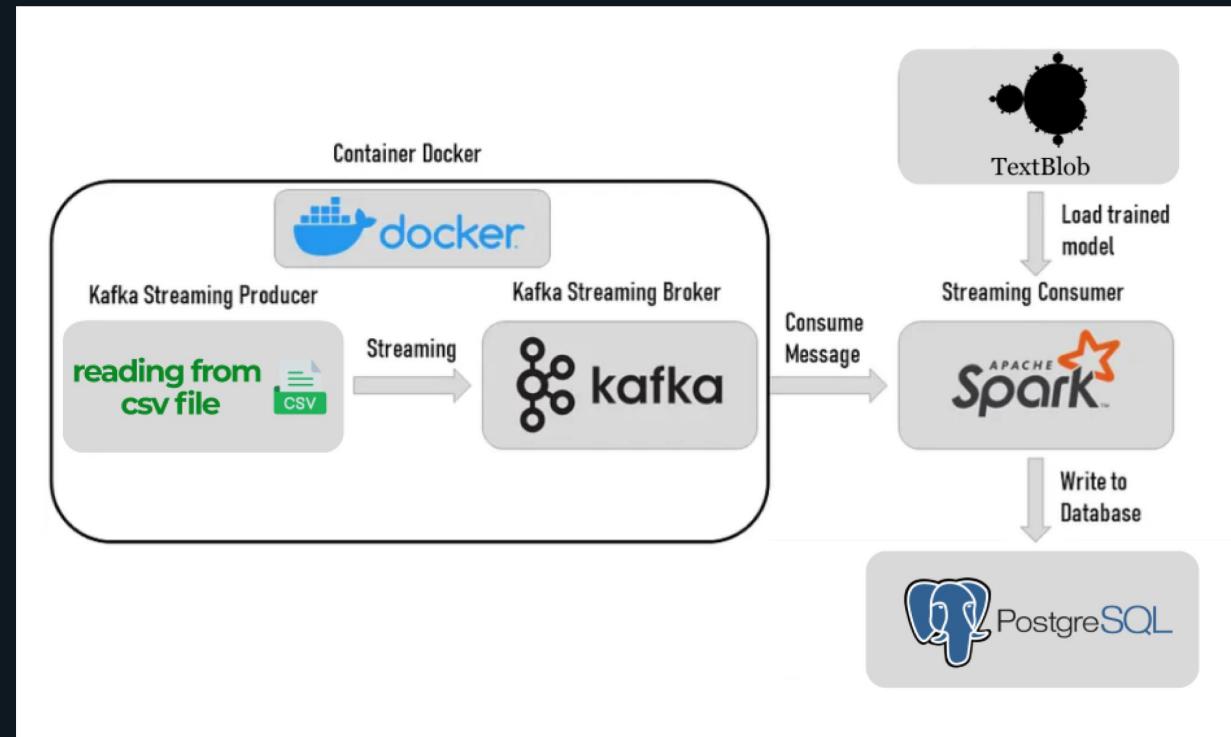


Інші зміни у стеку технологій

- 1. Використання PostgreSQL замість MySQL.** Обидві СУБД досить поширені, але для роботи з більш складними даними чи їхньою великою кількістю, краще обрати PostgreSQL, яка добре справляється з великими обсягами даних та складними запитами.
- 2. Використання Streamlit замість PowerBI.** Streamlit - фреймворк Python з відкритим вихідним кодом, який дозволяє створювати інтерактивні додатки для роботи з даними, особливо стрімінговими. Використання PowerBI не потребує коду (що є перевагою), але для використання саме аналітики реального часу потрібно оформлювати підписку.



Блок-схема пайплайну. Фінальна версія



1. Формування файлу docker-compose.yml

Оскільки у своїй роботі ми будемо використовувати сервери Kafka та Zookeeper, то створимо для цього окремий контейнер.

Для створення та запуску почати роботи серверів вводиться наступна команда в терміналі PyCharm:

```
PS D:\PROJECTS\TwitterRealTimeAnalytics> docker-compose up -d
[+] Running 3/3
✓ Network twitterrealtimeanalytics_default  Created          0.3s
✓ Container zoo1                            Started         9.4s
✓ Container kafka1                          Started         8.2s
```

2. Формування файлу kafka-producer.py

Kafka Producer — це компонент або клієнт у системі Apache Kafka, який відповідає за надсилання (продукування) даних у Kafka, записуючи їх у певну тему (topic).

Оскільки X API (former Twitter API) дозволяє отримати дані 500 постів, ми вирішили знайти альтернативне рішення: на платформі Kaggle ми завантажили датасет твітів про iPhone 14, який завантажимо у Apache Kafka, симулюючи реальний час (за допомогою функції `time.sleep()`). З частиною коду можна ознайомитись нижче:

```
# Function to send data to Kafka topic
def send_to_kafka(data, topic_name, kafka_server):
    producer = KafkaProducer(bootstrap_servers=kafka_server,
                            value_serializer=lambda v: json.dumps(v).encode('utf-8'))
    for record in data:
        producer.send(topic_name, record)
        producer.flush() # Ensures each message is sent
        sleep(0.02)
    producer.close()
```

3. Формування файлу kafka-consumer.py

Kafka Consumer — клієнт у системі Apache Kafka, який відповідає за отримання повідомлень з певної теми (topic). Основне завдання Consumer — забирати повідомлення з Kafka для подальшої обробки, збереження або аналітики.

У цьому файлі варто:

- прочитати дані з Kafka,
- обробка текстових даних;
- використати модель TextBlob для сентимент-аналізу,
- оброблені дані внести до таблиці БД PostgreSQL.

```
df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "tweets-iphone14") \
    .option("startingOffsets", "earliest") \
    .option("header", "true") \
    .option("failOnDataLoss", "false") \
    .load() \
    .selectExpr("CAST(value AS STRING)")
```

Читання даних з Kafka у датафрейм
для подальшої обробки

4. Формування файлу streamlit_dashboard.py

На дашборді ми планували розмістити такі візуалізації:

- коли було здійснено останнє оновлення;
- кількість проаналізованих твітів;
- топ-користувач, твіт якого зібрав максимальну кількість лайків;
- розподіл настрою твітів у вигляді pie chart;
- побудова «хмари слів», аби визначити найбільш повторювані слова у твітах;
- виведення найбільш популярних твітів (показник – максимальна кількість лайків);
- географічний розподіл користувачів.

```
left_column, middle_column, right_column = st.columns(3)
df = pd.DataFrame(data)
df['tweet_like_count'] = pd.to_numeric(df['tweet_like_count'])
avg_likes = round(df['tweet_like_count'].mean(), 1)
top_user = df.loc[df['tweet_like_count'].idxmax()]['username']
with left_column:
    st.subheader("Tweets Processed:")
    st.subheader(f"{total_count}")
with middle_column:
    st.subheader("Average Tweet Likes:")
    st.subheader(f"{avg_likes}")
with right_column:
    st.subheader("Top User:")
    st.subheader(f"{top_user}")
```

Фрагмент коду, що відповідає за розміщення числових метрик

Основні команди, що були використані для запуску проекту

1

```
docker-compose up -d
```

2

```
python kafka-producer.py
```

3

```
kafka-console-consumer --topic tweets_iphone14 --bootstrap-server  
localhost:9092 --from-beginning
```

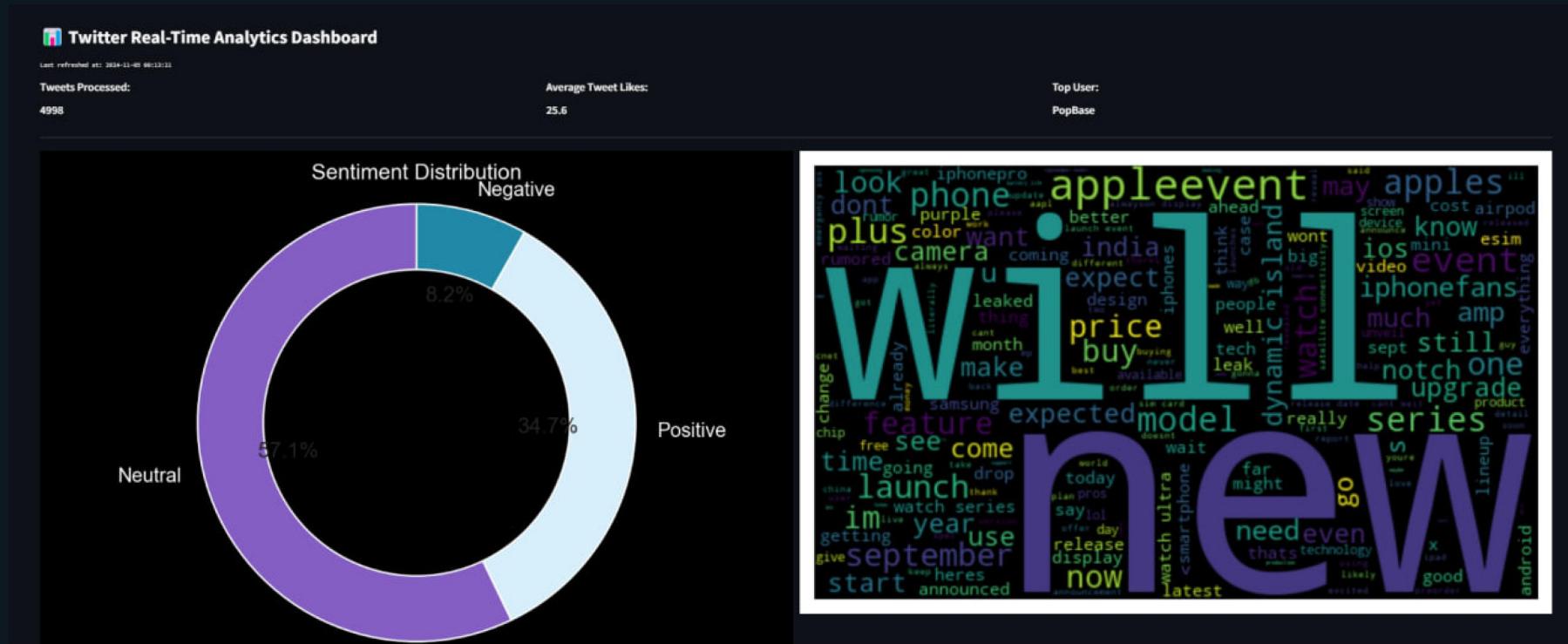
4

```
python kafka-consumer.py
```

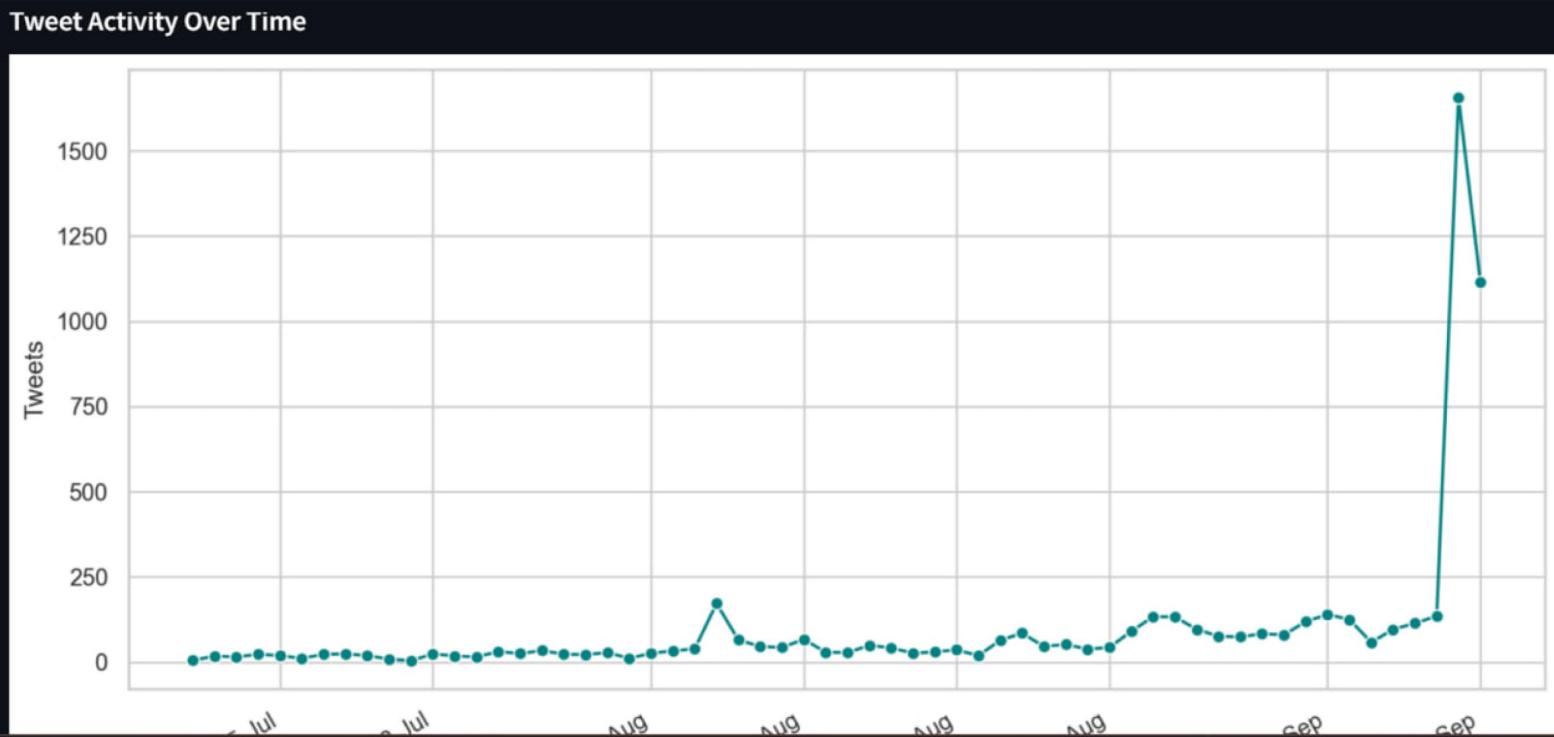
5

```
streamlit run dashboard.py
```

Фінальний вигляд дашборду



Фінальний вигляд дашборду



ПОБУДОВА BAR CHART'У ДЛЯ ЗАДАЧІ TOPIC
MODELING З ВИКОРИСТАННЯМ
ВІДПОВІДНОЇ МОДЕЛІ;



ЗОСЕРЕДЖЕННЯ ТА РОБОТА З ПОЛЕМ ГЕОЛОКАЦІЇ
КОРИСТУВАЧІВ ДЛЯ ПОБУДОВИ ГЕОГРАФІЧНОЇ МАПИ



МОЖЛИВІ ПОКРАЩЕННЯ/ДОПРАЦЮВАННЯ

ПОКРАЩЕННЯ ФУНКЦІЇ АКТИВНОСТІ ТВІТІВ
З МЕТОЮ ДОСЯГНЕННЯ ПЕВНОЇ
ДИНАМІЧНОСТІ



МОЖЛИВІСТЬ ФІЛЬТРУВАННЯ ЗА ДАТОЮ,
АБИ АНАЛІЗУВАТИ КОНКРЕТНІ МОМЕНТИ



1

ПРИДЛЯТИ ДОСТАТНЮ УВАГУ ПРАВИЛЬНІЙ ІНСТАЛЯЦІЇ
ПРОГРАМ ДЛЯ УСУНЕННЯ МОЖЛИВИХ БАГІВ

2

ЗВЕРТАТИ УВАГУ НА ВИМОГИ ПЕРЕД ІНСТАЛЯЦІЮ
СОФТУ, АБІ УНИКНУТИ ПРОБЛЕМИ НЕСУМІСНОСТІ

3

ПОПРАЦЮВАЛИ З НОВИМ СТЕКОМ ТЕХНОЛОГІЙ, ЩО
БУЛО ДОСИТЬ ЦІКАВО

4

ДОСЯГНУТА ЦІЛЬ ПОПРИ ТРУДНОЩІ ТА
ЗМІНИ, ЯКІ БУЛИ ПРИЙНЯТИ ПРОТЯГОМ
ВИКОНАННЯ ПРОЄКТУ

ДЯКУЄМО
ЗА УВАГУ.

ЧАС ДЛЯ ВАШИХ ЗАПИТАНЬ.

ФЕЩЕНКО ЯНА

МІЩЕНКО ФЕДІР

ПІДЛЕТЕЙЧУК ЮРІЙ