

# Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI  
I TECHNIK INFORMACYJNYCH



Instytut Informatyki

Studia Podyplomowe  
Big Data - przetwarzanie i analiza dużych zbiorów danych

PRACA KOŃCOWA

Patryk Franciszek Nowicki

Monitorowanie transakcji giełdowych przy  
wykorzystaniu narzędzi Big Data

Warszawa, 2020

## Spis treści

<b>1</b>	<b><i>Wstęp.....</i></b>	<b>3</b>
<b>2</b>	<b><i>Opis zbioru danych .....</i></b>	<b>4</b>
2.1	Pochodzenie .....	4
2.2	Struktura .....	4
2.3	Migracja próbki danych na bucket S3 .....	6
<b>3</b>	<b><i>Eksploracja danych.....</i></b>	<b>7</b>
3.1	Stworzenie środowiska do analizy danych.....	7
3.2	Zawartość oraz typy danych .....	8
3.3	Wartości zagregowane oraz wyniki analizy.....	8
<b>4</b>	<b><i>Przetwarzanie strumieniowe danych w czasie rzeczywistym.....</i></b>	<b>11</b>
4.1	Opis środowiska - Docker .....	11
4.2	Zebranie i przetworzenie danych za pomocą Apache Nifi .....	12
4.3	Opis klastrów Apache Kafka oraz Apache ZooKeeper .....	15
4.4	Przetwarzanie napływających danych za pomocą Spark Streaming .....	17
<b>5</b>	<b><i>Wnioski i zakończenie.....</i></b>	<b>20</b>
<b>6</b>	<b><i>Bibliografia .....</i></b>	<b>21</b>
<b>7</b>	<b><i>Załączniki .....</i></b>	<b>22</b>
7.1	Zawartość pliku project.yml .....	22
7.2	Wydruk przetworzonych danych w czasie rzeczywistym.....	23

# 1 Wstęp

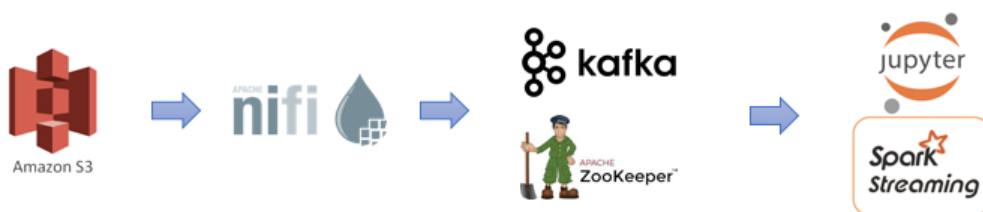
Niniejszy dokument stanowi opis projektu, którego przygotowanie i prezentacja kończy moje studia podyplomowe na Politechnice Warszawskiej na kierunku „*Big Data - przetwarzanie i analiza dużych zbiorów danych*”. Zgodnie z tematem pracy „*Monitorowanie transakcji giełdowych przy wykorzystaniu narzędzi Big Data*” w kolejnych rozdziałach opisuję wykonane w ramach projektu kroki począwszy od zebrania i analizy danych, a skończywszy na ich agregacji w czasie rzeczywistym.

Produktem pracy jest środowisko, które pozwala na zarówno gromadzenie giełdowych danych historycznych, jak i strumieniowe przetwarzanie danych. Ponadto stworzona architektura spełnia takie założenia biznesowe, jak odporność danych na awarie, łatwa skalowalność rozwiązania oraz możliwość rozszerzenia funkcjonalności o historyzację danych.

W Rozdziale 2 prezentuję pochodzenie zbioru danych, strukturę publikowania danych, przedstawiam sposób migracji danych na środowisko lokalne, a także na środowisko chmurowe AWS.

Rozdział 3 zawiera opis zebranych danych z 3 dat (25, 26, 27 maja 2020), których analiza umożliwiła odpowiednie zarządzanie strumieniami napływających danych w późniejszym etapie projektu. Dane są procesowane przy pomocy PySpark na klastrze złożonym z 3 instancji EC2. Podejście to jest wysoce skalowalne poziomo, jak i pionowo – w przypadku zwiększenia wolumetrii danych bądź zakresu analiz istnieją przeszkody (może poza finansowymi), aby zwiększyć liczbę instancji o nowe i/lub silniejsze jednostki obliczeniowe.

W Rozdziale 4 przedstawiłem opis lokalnego środowiska do przetwarzania strumieniowego stworzonego przy pomocy oprogramowania Docker. Zawarłem w nim opis procesu przetwarzania danych pochodzących z S3 i udostępnianych do klastra Apache Kafka przygotowany w Apache Nifi. Dane są „konsumowane” poprzez moduł Jupyter środowiska Python w celu wykorzystania rozszerzenia Apache Spark, jakim jest Apache SparkStreaming. Uproszczony schemat procesu został przedstawiony na poniższym rysunku.



Rysunek 1 Schemat przetwarzania danych

W Rozdziale 5 prezentuję podsumowanie wraz z wnioskami, natomiast w 6 i 7 odpowiednio bibliografię i załączniki do niniejszej pracy.

Warto dodać, iż proponowane w projekcie monitorowanie liczby transakcji nie jest jedynym przypadkiem wykorzystania przedstawionej architektury. Z kolei w rozdziale 5 zaproponowałem potencjalne rozszerzenia do handlu algorytmicznego lub platformy informacyjnej.

## 2 Opis zbioru danych

W niniejszym rozdziale omówiłem pochodzenie i strukturę danych wykorzystanych w analizie oraz przedstawiłem proces przeniesienia próbki danych na nowo utworzone wiadro (ang. bucket) S3 w środowisku chmurowym AWS (Amazon Web Services) przy wykorzystaniu AWS CLI (ang. Command Line Interface). Podczas projektu wykorzystałem konto udostępniane przez uczelnię w ramach serwisu AWS Educate. Proces nadawania uprawnień do platformy AWS<sup>1</sup> pominąłem, w utrzymaniu lepszej czytelności pracy.

### 2.1 Pochodzenie

W niniejszej pracy wykorzystałem dane pochodzące z projektu Deutsche Börse Public Dataset (PDS)<sup>2</sup>, w ramach którego upubliczniono dane z systemów transakcyjnych Deutsche Börse - spółki akcyjnej zarządzającej Giełdą Papierów Wartościowych we Frankfurcie. Jest to pierwsze przedsięwzięcie, w którym tak szczegółowe informacje na temat rynku finansowego zostały udostępnione swobodnie i nieprzerwanie od samego dostawcy źródła. Zbiór danych został stworzony z wykorzystaniem infrastruktury dostawcy chmury i jest publicznie udostępniany poprzez ich repozytorium danych - AWS Public Dataset. Głównym motywem utworzenia PDS przez grupę Deutsche Börse jest chęć wspierania innowacji i stwarzania nowych możliwości biznesowych poprzez dostępne bezpłatnie, wysokiej jakości dane. W konsekwencji dane są objęte licencją uniemożliwiającą wykorzystanie komercyjne<sup>3</sup>.

Rekordy w PDS przekazywane są z minuty na minutę<sup>4</sup> i generowane z silników Xetra i Eurex, które obejmują akcje, fundusze i pochodne papiery wartościowe. Zbiór danych zawiera szczegółowe informacje na poziomie pojedynczego papieru wartościowego, w tym najwyższe, najniższe, pierwsze i ostatnie ceny w danym okresie (minuta). Dane dostępne są od 26 czerwca 2017 r. dla Xetra i 29 maja 2017 r. dla Eurex.

### 2.2 Struktura

W ramach projektu PDS dane przesyłane są do dwóch wiader S3 (ang. bucket) w usłudze Amazon S3 w regionie środkowym UE (Frankfurt), są to:

- s3://deutsche-boerse-xetra-pds (dla danych Xetra) oraz
- s3://deutsche-boerse-eurex-pds (dla danych Eurex).

Każde powyższe wiadro zawiera katalog, w którym partycjom odpowiadają daty notowań (zgodne z formatem ISO 8601 RRRR-MM-DD) co zostało zobrazowane poniżej

---

<sup>1</sup>Wymagana odpowiednia konfiguracja, np. ~/.aws/credentials (<https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-configure.html>)

<sup>2</sup> Więcej informacji pod adresem [1]

<sup>3</sup> Non-commercial (NC) - licensees may copy, distribute, display, and perform the work and make derivative works and remixes based on it only for non-commercial purposes.

<sup>4</sup> The data is updated every minute during trading hours.

```
pnowicki ~ $aws s3 ls s3://deutsche-boerse-xetra-pds --summarize --no-sign-request
PRE 2017-06-17/
PRE 2017-06-18/
PRE 2017-06-19/
PRE 2017-06-20/
PRE 2017-06-21/
PRE 2017-06-22/
PRE 2017-06-23/
PRE 2017-06-24/
PRE 2017-06-25/
PRE 2017-06-26/
PRE 2017-06-27/
PRE 2017-06-28/
PRE 2017-06-29/
PRE 2017-06-30/
PRE 2017-07-01/
PRE 2017-07-02/
PRE 2017-07-03/
PRE 2017-07-04/
PRE 2017-07-05/
PRE 2017-07-06/
PRE 2017-07-07/
PRE 2017-07-08/
PRE 2017-07-09/
PRE 2017-07-10/
PRE 2017-07-11/
PRE 2017-07-12/
PRE 2017-07-13/
PRE 2017-07-14/
```

Rysunek 2 Struktura partycjonowania danych na wiadrze S3

Każdy plik śróddzienny zawiera dane dotyczące jednej minuty notowań. Na poniższym rysunku przedstawiłem przykładowe nazwy plików dla danych z dnia 2020-05-27 wykonane w tym samym dniu notowań tj 2020-05-27

```
pnowicki ~ $aws s3 ls s3://deutsche-boerse-xetra-pds/2020-05-27 --recursive --human-readable --no-sign-request
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0000.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0001.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0002.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0003.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0004.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0005.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0006.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0007.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0008.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0009.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0010.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0011.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0012.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0013.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0014.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0015.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0016.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0017.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0018.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0019.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0020.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0021.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0022.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0023.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0024.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0025.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0026.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0027.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0028.csv
2020-05-27 07:59:48 136 Bytes 2020-05-27/2020-05-27_BINS_XETR0029.csv
```

Rysunek 3 Nazewnictwo plików – dane z obecnego dnia notowań

Nazewnictwo plików udostępnianych w dniu notowań jest zgodne z konwencją

*RRRR-MM-DD\_BINS\_<mrkt><hh><mm>.csv*

gdzie: <mrkt> to kod identyfikacji rynku XEUR lub XETR,  
<hh> to dwucyfrowa godzina handlu w formacie 24-godzinny,  
<mm> to dwucyfrowa minuta handlu w formacie 60 minutowym, {0,...,59}.

Po zakończeniu dnia w miejsce plików minutowych umieszczane są pliki zagregowane do poziomu godziny<sup>5</sup>. Wówczas konwencja pomija czynnik <mm> i jest następująca

<sup>5</sup> Dane te są rozumiane jako dane historyczne. Nie biorą udziału przetwarzaniu w czasie rzeczywistym, niemniej dane z 3 dat poddałem analizie eksploracyjnej w celu zobrazowania niesionej za nimi informacji.

```
pnowicki ~ $  
pnowicki ~ $aws s3 ls s3://deutsche-boerse-xetra-pds/2020-05-26 --summarize --recursive --human-readable --no-si  
gn-request  
2020-05-27 02:29:49 136 Bytes 2020-05-26/2020-05-26_BINS_XETR00.csv  
2020-05-27 02:29:50 136 Bytes 2020-05-26/2020-05-26_BINS_XETR01.csv  
2020-05-27 02:29:50 136 Bytes 2020-05-26/2020-05-26_BINS_XETR02.csv  
2020-05-27 02:29:49 136 Bytes 2020-05-26/2020-05-26_BINS_XETR03.csv  
2020-05-27 02:29:50 136 Bytes 2020-05-26/2020-05-26_BINS_XETR04.csv  
2020-05-27 02:29:50 136 Bytes 2020-05-26/2020-05-26_BINS_XETR05.csv  
2020-05-27 02:29:50 136 Bytes 2020-05-26/2020-05-26_BINS_XETR06.csv  
2020-05-27 02:29:50 1.4 MiB 2020-05-26/2020-05-26_BINS_XETR07.csv  
2020-05-27 02:29:51 1.1 MiB 2020-05-26/2020-05-26_BINS_XETR08.csv  
2020-05-27 02:29:51 1.0 MiB 2020-05-26/2020-05-26_BINS_XETR09.csv  
2020-05-27 02:29:51 941.5 KiB 2020-05-26/2020-05-26_BINS_XETR10.csv  
2020-05-27 02:29:51 1.0 MiB 2020-05-26/2020-05-26_BINS_XETR11.csv  
2020-05-27 02:29:52 906.3 KiB 2020-05-26/2020-05-26_BINS_XETR12.csv  
2020-05-27 02:29:52 1.1 MiB 2020-05-26/2020-05-26_BINS_XETR13.csv  
2020-05-27 02:29:52 1.2 MiB 2020-05-26/2020-05-26_BINS_XETR14.csv  
2020-05-27 02:29:52 1.1 MiB 2020-05-26/2020-05-26_BINS_XETR15.csv  
2020-05-27 02:29:53 136 Bytes 2020-05-26/2020-05-26_BINS_XETR16.csv  
2020-05-27 02:29:53 136 Bytes 2020-05-26/2020-05-26_BINS_XETR17.csv  
2020-05-27 02:29:53 136 Bytes 2020-05-26/2020-05-26_BINS_XETR18.csv  
2020-05-27 02:29:53 136 Bytes 2020-05-26/2020-05-26_BINS_XETR19.csv  
2020-05-27 02:29:54 136 Bytes 2020-05-26/2020-05-26_BINS_XETR20.csv  
2020-05-27 02:29:54 136 Bytes 2020-05-26/2020-05-26_BINS_XETR21.csv  
2020-05-27 02:29:54 136 Bytes 2020-05-26/2020-05-26_BINS_XETR22.csv  
2020-05-27 02:29:54 136 Bytes 2020-05-26/2020-05-26_BINS_XETR23.csv  
  
Total Objects: 24  
Total Size: 9.8 MiB  
pnowicki ~ $
```

Rysunek 4 Nazewnictwo plików – dane historyczne

## 2.3 Migracja próbki danych na bucket S3

W celu wykonania analizy eksploracyjnej dane zostały pobrane z opisanego wcześniej wiadra S3 za pomocą AWS CLI z użyciem komendy:

```
aws s3 sync s3://deutsche-boerse-xetra-pds/<data> raw_data/<data>.
```

Komenda pobiera folder z danymi dla ustalonej daty <data> i umieszcza w folderze raw\_data. Zaimportowana próbka danych zawierała dane z silnika Xetra z dni 25, 26, 27 maja 2020, gdzie ostatnia z dat składała się z danych w agregacji minutowej.

W dalszej kolejności dokonałem utworzenia wiadra S3 na koncie AWS, co przedstawia zrzut na poniższym rysunku.

```
pnowicki ~ $aws s3 ls  
pnowicki ~ $aws s3api create-bucket --bucket pnowicki-xetra-data --region us-east-1  
{  
  "Location": "/pnowicki-xetra-data"  
}  
pnowicki ~ $aws s3 ls  
2020-05-27 11:39:34 pnowicki-xetra-data  
pnowicki ~ $
```

Rysunek 5 Zrzut ekranu z konsoli przedstawiający utworzenie wiadra S3

Następnie dane z lokalnego środowiska zostały przeniesione na stworzone wiadro S3 w celu ich dalszej analizy

```
pnowicki ~ $  
aws s3 sync raw_data/2020-05-27 s3://pnowicki-xetra-data/2020-05-27  
aws s3 sync raw_data/2020-05-26 s3://pnowicki-xetra-data/2020-05-26  
aws s3 sync raw_data/2020-05-25 s3://pnowicki-xetra-data/2020-05-25
```

Rysunek 6 Eksport danych na wiadro S3

Warto odnotować, że Xetra nie jest 24-godzinną platformą handlową, więc istnieją pliki, które nie zawierają żadnych transakcji (widać to na rysunkach Rysunek 2 oraz Rysunek 3). W przypadku silnika Xetra pliki takie mają rozmiar 136 bajtów<sup>6</sup>.

<sup>6</sup> Pliki są także filtrowane w rozdziale 4.2 poświęconym przetwarzania danych za pomocą Apache Nifi

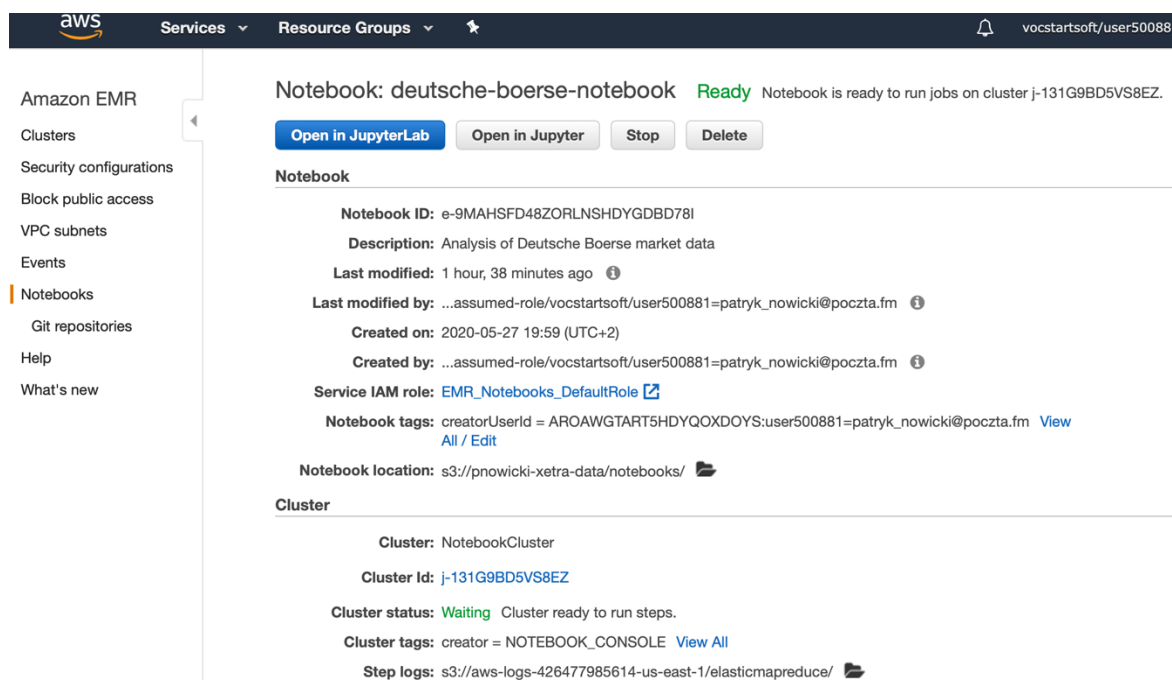
## 3 Eksploracja danych

W niniejszym rozdziale opisałem środowisko do analizy danych oraz charakterystyk pozyskanych danych, omawiając ich strukturę oraz podstawowe statystyki. Celem procesu eksploracji danych jest lepsze zrozumienia informacji niesionej w danych oraz wykrycie potencjalnych możliwości obciążeń systemu przetwarzania w czasie rzeczywistym.

### 3.1 Stworzenie środowiska do analizy danych

Celem przeprowadzenia analizy danych stworzyłem klaster z wykorzystaniem serwisu Amazon EMR<sup>7</sup>. Klaster znajduje się w strefie (ang. Availability Zone) us-east-1c (wschodnio-amerykańskiej), ze względu na edukacyjną wersję konta. Do tego zadania wybrałem jedną instancję EC2<sup>8</sup> m5.xlarge typu *master* o następujących charakterystykach: 4 vCPU, pamięć 16 GiB oraz dwie instancje typu *slave* o parametrach m5.4xlarge z 16 vCPU, pamięcią 64 GiB. Zdecydowałem się na taką konfigurację ze względu na fakt, że obliczenia dokonywane są głównie w warstwie *slave*, podczas gdy warstwa *master* zarządza i zleca zadania, więc te jednostki powinny być z reguły silniejsze.

Eksploracja i analiza danych została dokonana z wykorzystaniem EMR Notebooks<sup>9</sup> z jądrem PySpark w środowisku Jupyter. Parametry Notebooka zostały przedstawione na poniższym rysunku.



Rysunek 7 Parametry EMR Notebook utworzonego na klastrze z 3 instancjami EC2

<sup>7</sup> Więcej informacji na temat EMR dostępne w [2]

<sup>8</sup> Więcej informacji na temat oferowanych instancji EC2 dostępne w [3]

<sup>9</sup> Więcej informacji na temat EMR Notebooks dostępne w [4]

### 3.2 Zawartość oraz typy danych

Po zaimportowaniu danych do środowiska obliczeniowego wykonałem metodę printSchema zwracającą schemat danych przedstawiony poniżej.

```
root
|-- ISIN: string (nullable = true)
|-- Mnemonic: string (nullable = true)
|-- SecurityDesc: string (nullable = true)
|-- SecurityType: string (nullable = true)
|-- Currency: string (nullable = true)
|-- SecurityID: integer (nullable = true)
|-- Date: timestamp (nullable = true)
|-- Time: string (nullable = true)
|-- StartPrice: double (nullable = true)
|-- MaxPrice: double (nullable = true)
|-- MinPrice: double (nullable = true)
|-- EndPrice: double (nullable = true)
|-- TradedVolume: integer (nullable = true)
|-- NumberOfTrades: integer (nullable = true)
```

Jest on zgodny z tabelą prezentowaną poniżej zawierającą listę zmiennych opis oraz ich typ.

Zmienna	Opis	Typ
ISIN	Kod identyfikacyjny zdefiniowany w standardzie ISO 6166	Zmienna tekstowa
Mnemonic	Symbol giełdowy papieru wartościowego	Zmienna tekstowa
SecurityDesc	Opis papieru wartościowego	Zmienna tekstowa
SecurityType	Rodzaj papieru wartościowego	Zmienna tekstowa
Currency	Waluta, w której produkt jest przedmiotem obrotu	Zmienna tekstowa
SecurityID	ID kontraktu	Liczba całkowita
Date	Data	Data
Time	Godzina i minuta	Zmienna tekstowa, czas
StartPrice	Cena na początku danej minuty	Zmienna przecinkowa
MaxPrice	Maksymalna cena w ciągu danej minuty	Zmienna przecinkowa
MinPrice	Minimalna cena w ciągu danej minuty	Zmienna przecinkowa
EndPrice	Cena na końcu danej minuty	Zmienna przecinkowa
TradedVolume	Wolumen transakcji w ciągu jednej minuty	Zmienna przecinkowa
NumberOfTrades	Liczba transakcji w ciągu danej minuty	Liczba całkowita

Tabela 1 Opis zestawu zmiennych z analizowanego zbioru

### 3.3 Wartości zagregowane oraz wyniki analizy

Niniejszy rozdział przedstawia wyniki eksploracyjnej analizy próbki danych. W poniższej tabeli umieściłem licznosc poszczególnych grup papierów wartościowych. Można wywnioskować, że większość w zbiorze (57%) stanowią fundusze notowane na giełdzie oparte o instrumenty dłużne, 36% papierów wartościowych to akcje, 6% to instrumenty finansowe oparte o towary, a za jedynie 1% odpowiadają papiery dłużne.

Rodzaj papieru wartościowego	Liczebność
fundusze notowane na giełdzie oparte o instrumenty dłużne (ang. ETF)	1550
akcje	958
instrumenty finansowe oparte o towary (ang. ETC)	158
papiery dłużne (ang. ETN)	29

Tabela 2 Liczebność poszczególnych rodzajów papierów wartościowych

Poniższa tabela przedstawia liczebność obserwacji dla poszczególnych rodzajów papierów wartościowych. Najwięcej rekordów odnotowano dla akcji (około 71%), około 26% stanowiły fundusze notowane na giełdzie oparte o instrumenty dłużne, za około 2% odpowiadały instrumenty finansowe oparte o towary, a jedynie około 0,1% stanowiły papiery dłużne.



Rodzaj papieru wartościowego	Liczebność
akcje	195 896
fundusze notowane na giełdzie oparte o instrumenty dłużne (ang. ETF)	72 038
instrumenty finansowe oparte o towary (ang. ETC)	6 232
papiery dłużne (ang. ETN)	499

Tabela 3 Liczebność obserwacji dla poszczególnych grup papierów wartościowych

Jeśli chodzi o waluty, w których notowane były papiery wartościowe zdecydowaną większość stanowiło euro – aż około 99%, dla około 1% papierów walutą rozliczeniową był dolar amerykański, a znikoma część papierów wartościowych notowana była we funcie brytyjski, koronie szwedzkiej oraz chińskim juanie. Dane te zestawilem w poniższej tabeli.

Waluta	Liczebność
EUR	2657
USD	33
GBP	3
SEK	1
CNY	1

Tabela 4 Liczebność papierów wartościowych notowanych w poszczególnych walutach

Poniższa tabela przedstawia natomiast liczebność obserwacji dla papierów wartościowych notowanych w poszczególnych walutach. W tym przypadku również euro stanowi zdecydowaną większość – niemal 100%. Za mniej niż 1% odpowiadają natomiast papiery wartościowe notowane w dolarze amerykańskim, funcie brytyjskim, koronie szwedzkiej oraz chińskim juanie.

Waluta	Liczebność
EUR	274249
USD	366
GBP	26
SEK	15
CNY	9

Tabela 5 Liczebność obserwacji dla papierów wartościowych notowanych w poszczególnych walutach

Poniższa tabela przedstawia podstawowe statystyki dla akcji, z czego wszystkie notowane były w euro. Można zaobserwować, że ceny oscylowały wokół 78 euro, całkowity wolumen transakcji w analizowanym oknie wyniósł około 596 milionów euro, a całkowita liczba transakcji 1,2 miliona.

Agregat	Wartość
liczba unikalnych akcji	168 815
data początkowa	25-05-2020 08:00:00
data końcowa	27-05-2020 15:38:00:00
średnia cena na początku minuty	78,09
średnia maksymalna cena w ciągu minuty	78,11
średnia minimalna cena w ciągu minuty	78,06
średnia cena na końcu minuty	78,08
całkowity wolumen transakcji	596 542 867
całkowita liczba transakcji	1 256 976

Tabela 6 Podstawowe statystyki dla akcji

Największa liczba unikalnych akcji wystąpiła dla akcji Allianz SE NA O.N., których średnie ceny wynosiły około 163 euro, a na drugim miejscu znalazły się akcje firmy Bayer oraz SAP z cenami odpowiednio około 61 euro oraz 111 euro. Agregaty te przedstawiłem dla 5 grup papierów wartościowych z największą liczbą akcji w poniższej tabeli.

Kod identyfikacyjny	Symbol giełdowy	Opis papieru wartościowego	Średnia cena na początku minuty	Średnia maksymalna cena w ciągu minuty	Średnia minimalna cena w ciągu minuty	Średnia cena na końcu minuty	Całkowity wolumen transakcji	Całkowita liczba transakcji
DE0008404005	ALV	ALLIANZ SE NA O.N.	163,19	163,07	163,07	163,13	4819436	37 915
DE000BAY0017	BAYN	BAYER AG NA O.N.	61,7	61,72	61,68	61,7	10542 555	32 112
DE0007164600	SAP	SAP SE O.N.	111,51	111,55	111,48	111,51	7695 384	28 339
DE0007472060	WDI	WIRECARD AG	87,0	87,06	86,95	87,0	4864 838	30 033
DE0007664039	VOW3	VOLKSWAGEN AG VZO O.N.	137,09	137,18	137,01	137,1	4766 414	45 521

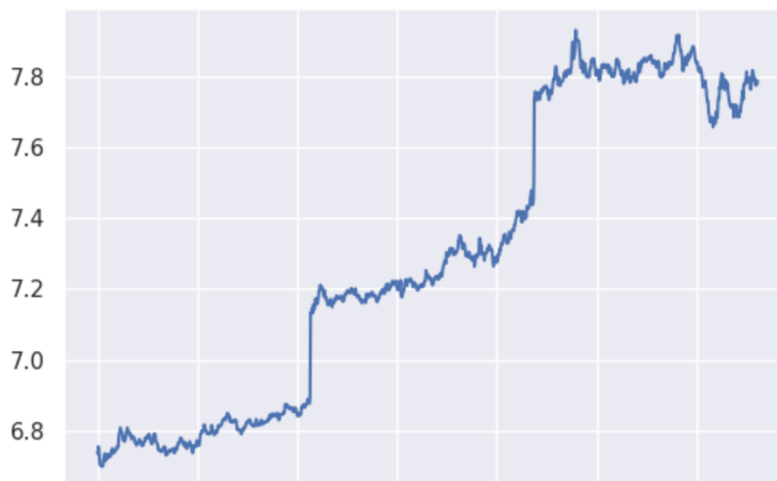
Tabela 7 Podstawowe statystyki dla pięciu najbardziej licznych grup akcji

W poniższej tabeli zestawilem pięć momentów w czasie, kiedy suma transakcji była najwyższa. Okazuje się, że najwięcej handlu odbywa się niedługo po otwarciu sesji i tuż przed jej zamknięciem.

Stempel czasowy	Całkowity wolumen transakcji w danej minucie	Całkowita liczba transakcji w danej minucie
2020-05-26 15:29:00	954 370	3 809
2020-05-27 15:29:00	878 641	3 581
2020-05-26 15:26:00	1 025 959	2 972
2020-05-27 08:36:00	1 016 192	2 964
2020-05-25 08:47:00	717 452	2 877

Tabela 8 Agregaty dla czasów, kiedy doszło do największej liczby transakcji

Dodatkowo na poniższym wykresie przedstawiłem zmianę ceny akcji na początku każdej minuty dla akcji DBK (Deutsche Bank) w dniach 25 maja do 27 maja 2020. Można zaobserwować, że z dnia na dzień ceny akcji wzrastały, a niewielki spadek odnotowano ostatniego dnia przed zamknięciem sesji.



Rysunek 8 Wykres cen akcji w dniach 25/05 - 27/05/2020

Podsumowując:

- rolę klucza instrumentu może pełnić ISIN Mnemonic, SecurityDesc, podczas gdy do strumieniowego przetwarzania danych (rozdziale 4 podpunkt 1.2.9.1) użyłem ISIN,
- dane są podzielone ze względu na typ (Common Stock, ETF, ETC oraz ETN) – podział ten zostanie wykorzystany do tworzenia tematów w Apache Kafka,
- najwięcej instrumentów jest w grupie ETF (1550),
- najwięcej danych minutowych jest w grupie Common Stocks (195896),
- grupy ETC oraz ETN są z reguły mało istotne,
- handel głównie odbywa się na początku i zakończeniu sesji,
- notowania akcji odbywają się jedynie w walucie Euro (EUR).

## 4 Przetwarzanie strumieniowe danych w czasie rzeczywistym

Rozdział ten prezentuje proces zbierania i obróbki surowych danych giełdowych z publicznie udostępnionego wiadra S3 przy wykorzystaniu Apache Nifi, przekazanie danych do poszczególnych tematów na klastrze Apache Kafka, a także proces wykorzystania danych do agregacji wartości transakcji poszczególnych instrumentów w czasie rzeczywistym w celu monitoringu popytu na giełdzie.

W pierwszym podrozdziale opisałem etap przygotowania środowiska obliczeniowego poprzez wykorzystanie oprogramowania Docker oraz narzędzia docker-compose. Wykorzystałem infrastrukturę kontenerów analogiczną do omawianej podczas zajęć dydaktycznych zawierającą: kontener Apache Nifi, 3 kontenery Apache ZooKeeper oraz Apache Kafka opracowane przez ConfluentInc połączone w komunikujące się ze sobą klastry, kontener z PySpark w celu użycia rozszerzenia Apache Spark – Apache SparkStreaming. W drugim podrozdziale opisałem grupy procesowe (ang. process groups) stworzone w Apache Nifi:

- 1) grupa wykorzystująca rzeczywiste dane w celu analizy w czasie rzeczywistym, pozwalająca na odpowiednie przeprosowanie publikowanych danych źródłowych
- 2) grupa generująca przykładowe dane poza godzinami pracy giełdy.

W podrozdziale 3 zawarłem opis struktury klastra Apache Kafka oraz klastra Apache ZooKeeper, a także schemat ich komunikacji. W podrozdziale 4 opisałem agregację danych przesyłanych przez Apache Kafkę, wykorzystując do tego rozszerzenie Apache Spark – SparkStreaming<sup>10</sup>

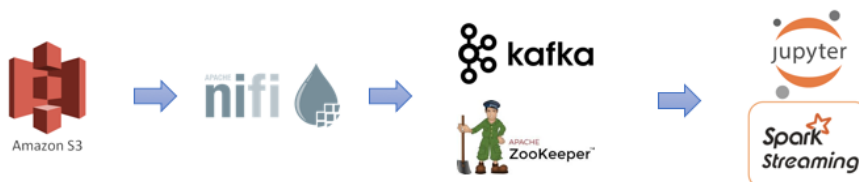
### 4.1 Opis środowiska - Docker

Do zbudowania środowiska obliczeniowego wykorzystałem narzędzie Docker-compose udostępniane przez oprogramowanie Docker. Narzędzie to pozwala na jednoczesne uruchomienie wielokontenerowej aplikacji za pomocą predefiniowanego pliku YAML – project.yml – wykorzystany plik został załączony w rozdziale 7.1. Informacje na temat Docker-compose można odnaleźć w dokumentacji dostępnej pod [5]. Plik konfiguracyjny project.yml zawiera następującą konfigurację:

- kontener z Apache Nifi powstały z obrazu *apache/nifi*, [7],
- 3 kontenery Apache Kafka powstałe z obrazów *confluentinc/cp-zookeeper*, [8],
- 3 kontenery Apache ZooKeeper powstałe z obrazów *confluentinc/cp-kafka*, [9],
- kontener z notatnikiem Jupyter powstały z obrazu *jupyter/pyspark-notebook*, [10].

Wersje obrazów są zgodne z datą 20-05-2020 natomiast szczegółowy opis można znaleźć na platformie DockerHub, [10].

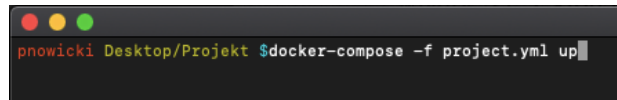
Na poniższym schemacie umieściłem uproszczony proces przetwarzania danych poczynając od wiadra S3, z którego pozyskiwane są dane poprzez transformacje w Apache Nifi, klaster Apache Kafka, a skończywszy na SparkStreaming i notatniku Jupyter



Rysunek 9 Schemat przetwarzania danych

Poprzez wykonanie poniższej komendy docker-compose

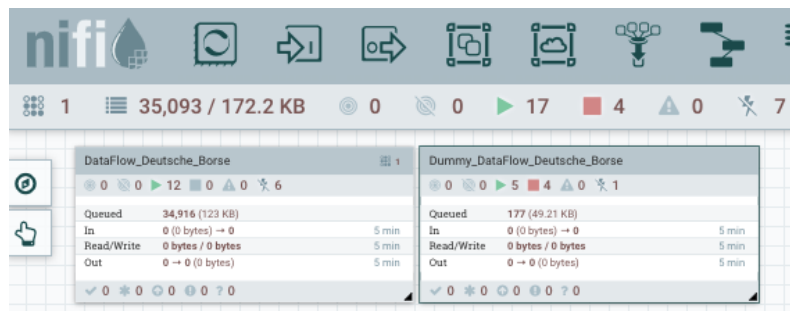
<sup>10</sup> Więcej informacji na temat Spark Streaming pod adresem [5]



w środowisku Docker powstała następująca struktura kontenerów:

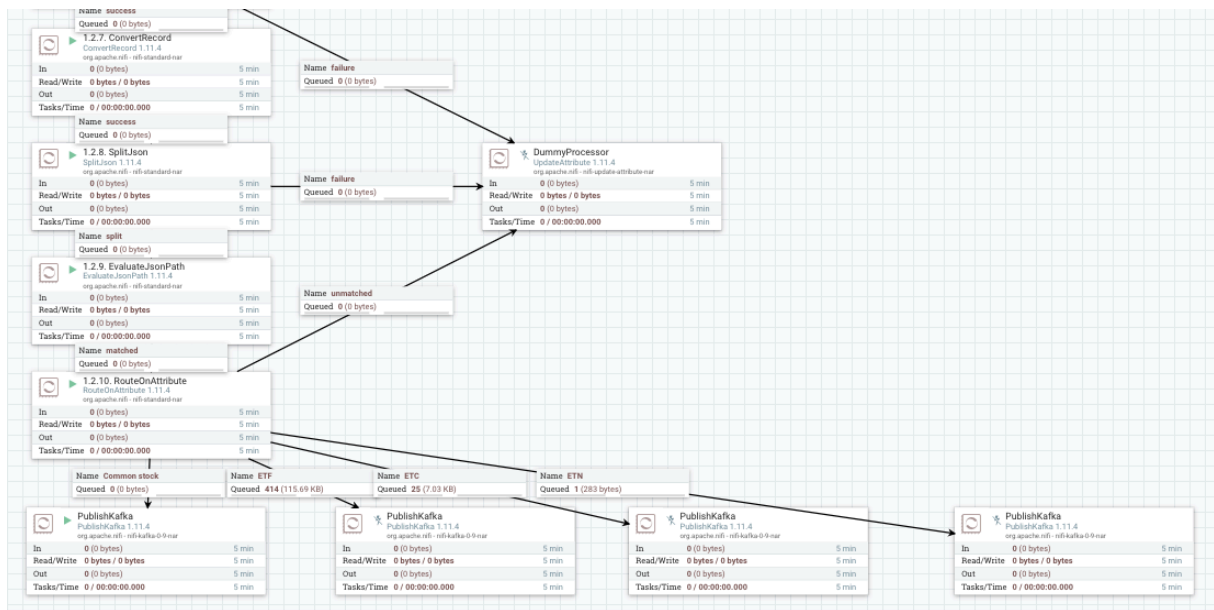
### Rysunek 11 Lista kontenerów w środowisku obliczeniowym

## 4.2 Zebranie i przetworzenie danych za pomocą Apache Nifi



Proces grupa *DataFlow\_Deutsche\_Borse* pozwala na transport i przetworzenie danych z wiadra S3 *deutsche-boerse-xetra-pds* do klastra Apache Kafka (omawianego w kolejnym podrozdziale. Schemat został zobrazowany na następujących rysunkach.

**Rysunek 13 Schemat przetwarzania danych w proces grupie DataFlow Deutsche Borse – część I**



Rysunek 14 Schemat przetwarzania danych w proces grupie DataFlow\_Deutsche\_Borse – część II

Grupa obejmuje poniższe etapy, numeracja odpowiada procesorom<sup>11</sup> na schematach wyżej:

- 1.1. Pobranie listy dostępnych obiektów w wiadrze S3 (w chwili tworzenia dokumentu źródło danych zawiera 34 916 plików).
- 1.2. Wydzielenie z obecnego dnia i obiektów historycznych (po dacie katalogu), następnie:

- dla **obiektów historycznych** rozważyłem następujące etapy:

- 1.2.1. wyłączenie obiektów, które nie posiadają danych – 136 bajtów,
- 1.2.2. wybranie obiektów ze względu na typ agregacji danych - dane godzinowe,
- 1.2.3. pobranie obiektów z dat historycznych (liczba obiektów dostępnych do pobrania wynosi 15 274 - są to tylko metadane obiektów przechowywanych na s3<sup>12</sup> ich pobranie oraz umieszczenie w dedykowanym systemie plików (np. hdfs) lub bazie danych (np. Oracle, Cassandra) wykracza poza zakres tego projektu.

- dla **obiektów z obecnego dnia notowań**:

- 1.2.4. wyłączenie obiektów, które nie posiadają danych – 136 bajtów,
- 1.2.5. wybranie obiektów ze względu na typ agregacji danych - dane minutowe,
- 1.2.6. pobranie obiektów,
- 1.2.7. konwersja danych z formy płaskiej (.csv) na ustrukturyzowaną (.json),
- 1.2.8. podział danych w postaci .json na pliki zawierające pojedyncze instrumenty,
- 1.2.9. Utworzenie atrybutów *key*, *time*, *type*.
  - 1.2.9.1. *key* - klucz przekazywanej informacji do Apache Kafka;
  - 1.2.9.2. *time* - oznaczający godzinę (hh:mm) danych;
  - 1.2.9.3. *type* - wskazujący na typ instrumentu finansowego,
- 1.2.10. podział danych po atrybucie *type* (na Common Stocks, ETF, ETN, ETC),
- 1.2.11. publikacja danych do Apache Kafka (opisanych w kolejnym podrozdziale) dla odpowiednich tematów<sup>13</sup>.

<sup>11</sup> opis wykorzystanych procesorów znajduje się w dokumentacji Apache Nifi w [12]

<sup>12</sup> pobranie plików do dedykowanego systemu plików (np. hdfs) lub bazy danych (np. Oracle, Cassandra) wykracza poza zakres projektu

<sup>13</sup> w celach prezentacyjnych w dokumencie wykonałem publikację danych do tematu CommonStocks

Na szczególną uwagę zasługuje punkt 1.2.7 czyli konwersja z postaci płaskiej pliku (.csv) przedstawionej poniżej:



The screenshot shows a flat CSV file view with a 'View as: original' dropdown. The data is a single line of comma-separated values representing stock market data for various companies.

1	ISIN,Mnemonic,SecurityDesc,SecurityType,Currency,SecurityID,Date,Time,StartPrice,MaxPrice,MinPrice,EndPrice,TradedVolume,NumberOfTrades
2	DE0006231004,IFX,INFINEON TECH.AG NA O.N.,Common stock,EUR,2505024,2020-06-05,07:00,20.54,20.805,20.5,20.77,388187,161
3	DE0005190003,BMW,BAY.MOTOREN WERKE AG ST,Common stock,EUR,2504900,2020-06-05,07:00,58.85,58.9,58.68,58.9,54763,48
4	DE0005190037,BMW3,BAY.MOTOREN WERKE VZO,Common stock,EUR,2504901,2020-06-05,07:00,45.02,45.28,45.02,45.28,2120,2
5	DE0004161830,GLJ,GERKE AG NA O.N.,Common stock,EUR,2504606,2020-06-05,07:00,80.80,80.80,723,1
6	DE0005552004,DPW,DEUTSCHE POST AG NA O.N.,Common stock,EUR,2504953,2020-06-05,07:00,30.84,30.94,30.71,30.86,89111,33
7	DE0007480204,DEQ,DEUTSCHE EUROSCHOP NA O.N.,Common stock,EUR,2505102,2020-06-05,07:00,14.84,15.14,14.84,14.91,6376,9
8	DE0006452907,NEM,NEMETSCHKE SE O.N.,Common stock,EUR,2505040,2020-06-05,07:00,67.8,67.8,67.45,67.45,1887,6
9	NL0012044747,SAB,SHOP APOTHERE EUROPE INH.,Common stock,EUR,2506271,2020-06-05,07:00,91.91,90.4,90.4,610,4
10	DE0008303504,TEG,TAG IMMOBILIEN AG,Common stock,EUR,2505131,2020-06-05,07:00,22.32,22.32,22.32,22.32,458,1
11	DE0008404005,ALV,ALLIANZ SE NA O.N.,Common stock,EUR,2505133,2020-06-05,07:00,188.5,188.74,188.2,188.46,36508,93
12	DE0005785802,FME,FRESHEN.MED.CARE KGAA O.N.,Common stock,EUR,2504974,2020-06-05,07:00,76.8,76.8,76.5,76.5,26022,48
13	DE0006062144,ICOV,COVESTRO AG O.N.,Common stock,EUR,2505008,2020-06-05,07:00,36.97,37.36,36.93,36.97,19116,24
14	DE0006062144,ICOV,COVESTRO AG O.N.,Common stock,EUR,2504380,2020-06-05,07:00,80.05,80.05,80.05,80.05,69,1

Rysunek 15 Postać płaska pliku

na postać ustrukturyzowaną (.json) zilustrowaną na poniższym rysunku:

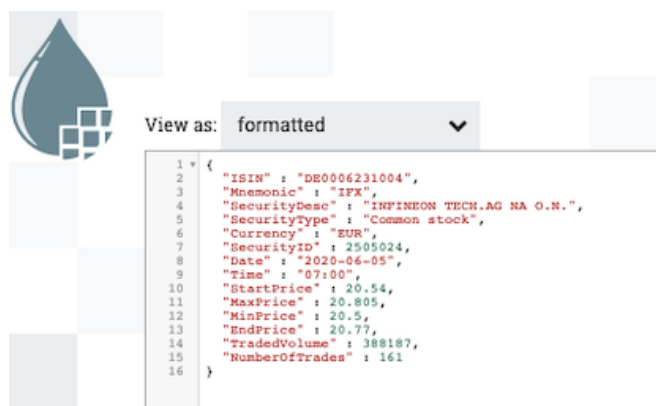


The screenshot shows a structured JSON file view with a 'View as: formatted' dropdown. The data is organized into a JSON array of objects, each representing a stock market entry.

```
1 {
2   "ISIN": "DE0006231004",
3   "Mnemonic": "IFX",
4   "SecurityDesc": "INFINEON TECH.AG NA O.N.",
5   "SecurityType": "Common stock",
6   "Currency": "EUR",
7   "SecurityID": "2505024",
8   "Date": "2020-06-05",
9   "Time": "07:00",
10  "StartPrice": "20.54",
11  "MaxPrice": "20.805",
12  "MinPrice": "20.5",
13  "EndPrice": "20.77",
14  "TradedVolume": "388187",
15  "NumberOfTrades": "161"
16 },
17 {
18   "ISIN": "DE0005190003",
19   "Mnemonic": "BMW",
20   "SecurityDesc": "BAY.MOTOREN WERKE AG ST",
21   "SecurityType": "Common stock",
22   "Currency": "EUR",
23   "SecurityID": "2504900",
24   "Date": "2020-06-05",
25   "Time": "07:00",
26   "StartPrice": "58.85",
27   "MaxPrice": "58.9",
28   "MinPrice": "58.68",
29   "EndPrice": "58.9",
30   "TradedVolume": "54763",
31   "NumberOfTrades": "48"
32 },
33 {
34   "ISIN": "DE0005190037",
35   "Mnemonic": "BMW3",
36   "SecurityDesc": "BAY.MOTOREN WERKE VZO",
37   "SecurityType": "Common stock",
38   "Currency": "EUR",
39   "SecurityID": "2504901",
40   "Date": "2020-06-05",
41   "Time": "07:00",
42   "StartPrice": "45.02",
43   "MaxPrice": "45.28",
44   "MinPrice": "45.02",
45   "EndPrice": "45.28",
46   "TradedVolume": "2120",
47   "NumberOfTrades": "2"
48 },
49 {
50   "ISIN": "DE0004161830",
51   "Mnemonic": "GLJ",
52   "SecurityDesc": "GERKE AG NA O.N.",
53   "SecurityType": "Common stock",
54   "Currency": "EUR",
55   "SecurityID": "2504606",
56   "Date": "2020-06-05",
57   "Time": "07:00",
58   "StartPrice": "80.80",
59   "MaxPrice": "80.80",
60   "MinPrice": "80.80",
61   "EndPrice": "80.80",
62   "TradedVolume": "723",
63   "NumberOfTrades": "1"
64 },
65 {
66   "ISIN": "DE0005552004",
67   "Mnemonic": "DPW",
68   "SecurityDesc": "DEUTSCHE POST AG NA O.N.",
69   "SecurityType": "Common stock",
70   "Currency": "EUR",
71   "SecurityID": "2504953",
72   "Date": "2020-06-05",
73   "Time": "07:00",
74   "StartPrice": "30.84",
75   "MaxPrice": "30.94",
76   "MinPrice": "30.71",
77   "EndPrice": "30.86",
78   "TradedVolume": "89111",
79   "NumberOfTrades": "33"
80 },
81 {
82   "ISIN": "DE0007480204",
83   "Mnemonic": "DEQ",
84   "SecurityDesc": "DEUTSCHE EUROSCHOP NA O.N.",
85   "SecurityType": "Common stock",
86   "Currency": "EUR",
87   "SecurityID": "2505102",
88   "Date": "2020-06-05",
89   "Time": "07:00",
90   "StartPrice": "14.84",
91   "MaxPrice": "15.14",
92   "MinPrice": "14.84",
93   "EndPrice": "14.91",
94   "TradedVolume": "6376",
95   "NumberOfTrades": "9"
96 },
97 {
98   "ISIN": "DE0006452907",
99   "Mnemonic": "NEM",
100  "SecurityDesc": "NEMETSCHKE SE O.N.",
101  "SecurityType": "Common stock",
102  "Currency": "EUR",
103  "SecurityID": "2505040",
104  "Date": "2020-06-05",
105  "Time": "07:00",
106  "StartPrice": "67.8",
107  "MaxPrice": "67.8",
108  "MinPrice": "67.45",
109  "EndPrice": "67.45",
110  "TradedVolume": "1887",
111  "NumberOfTrades": "6"
112 },
113 {
114   "ISIN": "NL0012044747",
115   "Mnemonic": "SAB",
116   "SecurityDesc": "SHOP APOTHERE EUROPE INH.",
117   "SecurityType": "Common stock",
118   "Currency": "EUR",
119   "SecurityID": "2506271",
120   "Date": "2020-06-05",
121   "Time": "07:00",
122   "StartPrice": "91.91",
123   "MaxPrice": "91.91",
124   "MinPrice": "90.4",
125   "EndPrice": "90.4",
126   "TradedVolume": "610",
127   "NumberOfTrades": "4"
128 },
129 {
130   "ISIN": "DE0008303504",
131   "Mnemonic": "TEG",
132   "SecurityDesc": "TAG IMMOBILIEN AG",
133   "SecurityType": "Common stock",
134   "Currency": "EUR",
135   "SecurityID": "2505131",
136   "Date": "2020-06-05",
137   "Time": "07:00",
138   "StartPrice": "22.32",
139   "MaxPrice": "22.32",
140   "MinPrice": "22.32",
141   "EndPrice": "22.32",
142   "TradedVolume": "458",
143   "NumberOfTrades": "1"
144 },
145 {
146   "ISIN": "DE0008404005",
147   "Mnemonic": "ALV",
148   "SecurityDesc": "ALLIANZ SE NA O.N.",
149   "SecurityType": "Common stock",
150   "Currency": "EUR",
151   "SecurityID": "2505133",
152   "Date": "2020-06-05",
153   "Time": "07:00",
154   "StartPrice": "188.5",
155   "MaxPrice": "188.74",
156   "MinPrice": "188.2",
157   "EndPrice": "188.46",
158   "TradedVolume": "36508",
159   "NumberOfTrades": "93"
160 },
161 {
162   "ISIN": "DE0005785802",
163   "Mnemonic": "FME",
164   "SecurityDesc": "FRESHEN.MED.CARE KGAA O.N.",
165   "SecurityType": "Common stock",
166   "Currency": "EUR",
167   "SecurityID": "2504974",
168   "Date": "2020-06-05",
169   "Time": "07:00",
170   "StartPrice": "76.8",
171   "MaxPrice": "76.8",
172   "MinPrice": "76.5",
173   "EndPrice": "76.5",
174   "TradedVolume": "26022",
175   "NumberOfTrades": "48"
176 },
177 {
178   "ISIN": "DE0006062144",
179   "Mnemonic": "ICOV",
180   "SecurityDesc": "COVESTRO AG O.N.",
181   "SecurityType": "Common stock",
182   "Currency": "EUR",
183   "SecurityID": "2505008",
184   "Date": "2020-06-05",
185   "Time": "07:00",
186   "StartPrice": "36.97",
187   "MaxPrice": "37.36",
188   "MinPrice": "36.93",
189   "EndPrice": "36.97",
190   "TradedVolume": "19116",
191   "NumberOfTrades": "24"
192 },
193 {
194   "ISIN": "DE0006062144",
195   "Mnemonic": "ICOV",
196   "SecurityDesc": "COVESTRO AG O.N.",
197   "SecurityType": "Common stock",
198   "Currency": "EUR",
199   "SecurityID": "2504380",
200   "Date": "2020-06-05",
201   "Time": "07:00",
202   "StartPrice": "80.05",
203   "MaxPrice": "80.05",
204   "MinPrice": "80.05",
205   "EndPrice": "80.05",
206   "TradedVolume": "69",
207   "NumberOfTrades": "1"
208 }
```

Rysunek 16 Postać ustrukturyzowana pliku

Finalnie pliki przekazywane do klastra Apache Kafka reprezentują dane z jednej minuty dla danego instrumentu finansowego notowanego na giełdzie. Dla przykładu powyższy plik jest rozdzielany per instrument finansowy, to znaczy na plik postaci przedstawionej poniżej.



The screenshot shows the final structured JSON file view with a 'View as: formatted' dropdown. The data is organized into a JSON array of objects, each representing a stock market entry.

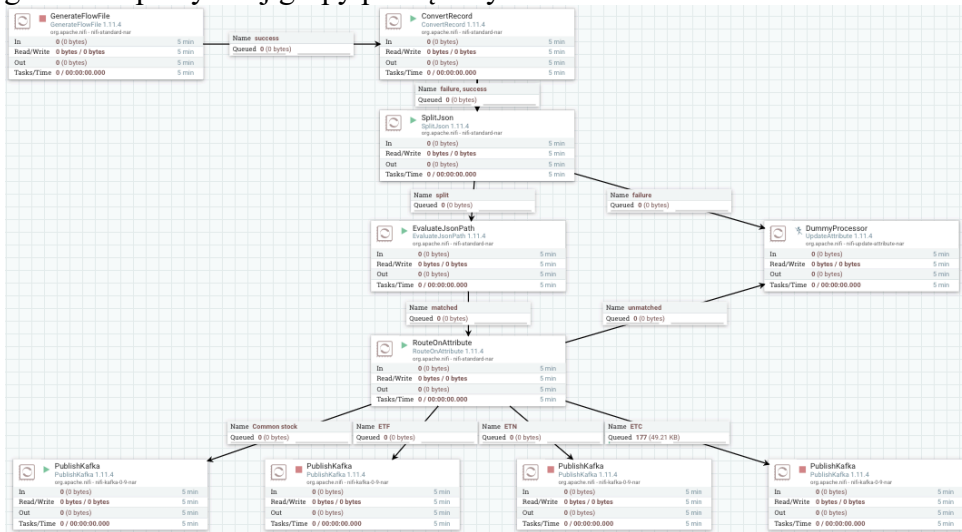
```
1 {
2   "ISIN": "DE0006231004",
3   "Mnemonic": "IFX",
4   "SecurityDesc": "INFINEON TECH.AG NA O.N.",
5   "SecurityType": "Common stock",
6   "Currency": "EUR",
7   "SecurityID": "2505024",
8   "Date": "2020-06-05",
9   "Time": "07:00",
10  "StartPrice": "20.54",
11  "MaxPrice": "20.805",
12  "MinPrice": "20.5",
13  "EndPrice": "20.77",
14  "TradedVolume": "388187",
15  "NumberOfTrades": "161"
16 }
```

Rysunek 17 Finalna postać pliku przekazywanego do Apache Kafka

Plik ten reprezentuje zatem informacje o handlu z godziny 7:00 (UTC, +2 CEST) z dnia 2020-06-05 dla akcji o symbolu IFX zgodnie z kluczem ISIN.

Dруга процес група *Dummy\_DataFlow\_Deutsche\_Borse* pozwala na przekształcenie próbki przykładowych danych w postaci płaskiej do postaci umożliwiającej publikację do Apache Kafka. Grupa zaczyna się od wygenerowania pliku płaskiego reprezentującego dane

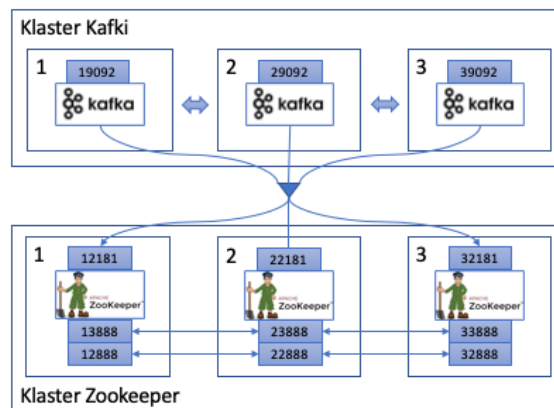
z jednej minuty, jak prezentowana na rysunku Rysunek 15 Postać płaska pliku, a następnie jest analogiczna do powyższej grupy począwszy od kroku 1.2.7.



Rysunek 18 Schemat przetwarzania danych w proces grupie Dummy\_DataFlow\_Deutsche\_Borse

### 4.3 Opis klastrów Apache Kafka oraz Apache ZooKeeper

Kolejnym elementem środowiska są klastry Apache Kafka oraz Apache ZooKeeper<sup>14</sup>. Na poniższym rysunku przedstawiłem poglądowy schemat komunikacji pomiędzy klastrami uwzględniając porty zdefiniowane w pliku konfiguracyjnym zawartym w 7.1.



Rysunek 19 Schemat komunikacji klastra Apache Kafka oraz Apache ZooKeeper

Na klastrze Kafki stworzyłem cztery tematy odpowiadające typom instrumentów finansowych notowanych na giełdzie, zgodnie z podziałem w poprzednich rozdziałach:

- *CommonStocks* – zawierający informacje o akcjach,
- *ETF* – zawierający informacje o funduszach,
- *ETC* – zawierający informacje o instrumentach opartych na surowcach,
- *ETN* – zawierający informacje o instrumentach dłużnych.

Zgodnie z oczekiwaniami i analizą przykładowych danych w poprzednim rozdziale cele oraz obciążenie procesowanych informacji w odpowiednich tematach mogą się różnić od siebie. Na przykład z biznesowego punktu widzenia nie w każdym temacie utrata danych jest istotna. To znaczy oczekuję, że informacje przetwarzane w temacie:

- *CommonStock* będą zawsze dostępne oraz odporne na utratę jakichkolwiek danych,
- *ETF* będą dostępne oraz odporne na utratę jakichkolwiek danych,

<sup>14</sup> Więcej na temat narzędzie Apache Kafka oraz Apache ZooKeeper znajduje się w [13]

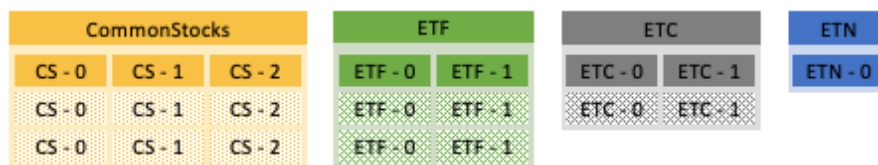


- *ETC* będą dostępne oraz dopuszcza się częściową utratę danych,
- *ETN* będą średnio dostępne oraz dopuszcza się całkowitą utratę danych.

Powyższa dostępność przekłada się na ilość brokerów/partycji (partycje są rozkładane po brokerach) obsługujących dany temat. Odporność na utratę danych przekłada się natomiast na ilość przechowywanych replik stworzonych partycji (rozłożonych po brokerach), które mogą zostać odzyskane po awarii brokerów. Wobec tego postanowiłem stworzyć tematy z następującymi założeniami:

- *CommonStocks*: liczba partycji: 3, liczba replik: 3,
- *ETF*: liczba partycji: 2, liczba replik: 3,
- *ETN*: liczba partycji: 2, liczba replik: 2,
- *ETC*: liczba partycji: 1, liczba replik: 1.

Na poniższym rysunku zobrazowałem założenia partycjonowania i replikowania partycji danych tematów



Rysunek 20 Założenia przy tworzeniu tematów w klastrze Apache Kafka

Poniższy rysunek przedstawia stworzenie tematów zgodnie z powyższymi założeniami

```
pnowicki ~ $docker exec -it projekt_kafka-1_1 bash
root@24f7668918ee:/# kafka-topics --bootstrap-server kafka-1:19091,kafka-2:29091,kafka-3:39091 --create --topic CommonStocks --partitions 3 --replication-factor 3
Created topic CommonStocks.
root@24f7668918ee:/# kafka-topics --bootstrap-server kafka-1:19091,kafka-2:29091,kafka-3:39091 --create --topic ETF --partitions 2 --replication-factor 3
Created topic ETF.
root@24f7668918ee:/# kafka-topics --bootstrap-server kafka-1:19091,kafka-2:29091,kafka-3:39091 --create --topic ETC --partitions 2 --replication-factor 2
Created topic ETC.
root@24f7668918ee:/# kafka-topics --bootstrap-server kafka-1:19091,kafka-2:29091,kafka-3:39091 --create --topic ETN
Created topic ETN.
root@24f7668918ee:/#
```

Rysunek 21 Stworzenie tematów CommonStocks, ETF, ETC, ETN w Apache Kafka

Apache Kafka przyporządkował następujące brokery (Leader: <numer brokera>) do obsługi danych partycji i replik

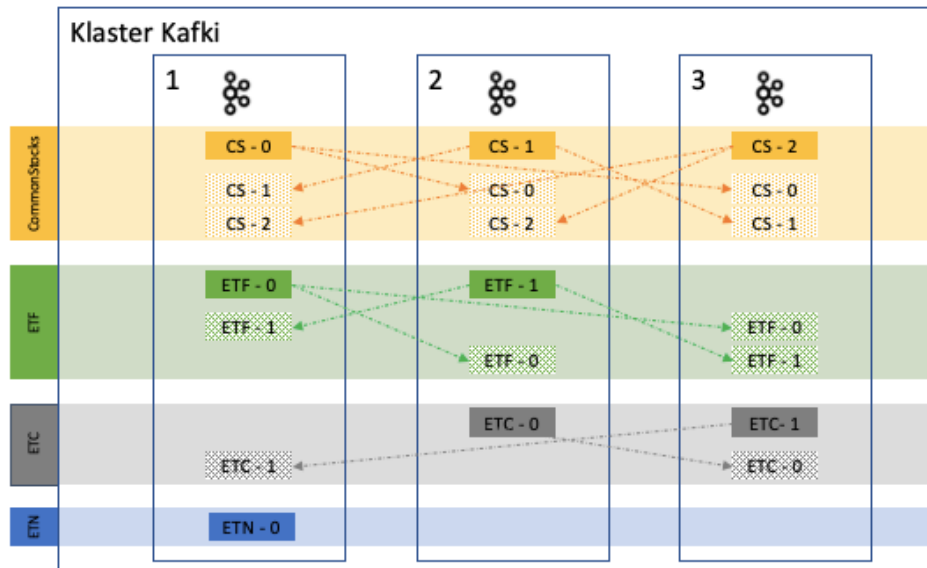
```
pnowicki ~ $docker exec -it projekt_kafka-1_1 bash
root@24f7668918ee:/# kafka-topics --bootstrap-server kafka-1:19091 --topic CommonStocks,ETF,ETN,ETC --describe
Topic: CommonStocks PartitionCount: 3 ReplicationFactor: 3 Configs:
Topic: CommonStocks Partition: 0 Leader: 1 Replicas: 1,2,3 Isr: 1,2,3
Topic: CommonStocks Partition: 1 Leader: 2 Replicas: 2,3,1 Isr: 2,3,1
Topic: CommonStocks Partition: 2 Leader: 3 Replicas: 3,1,2 Isr: 3,1,2
Topic: ETC PartitionCount: 2 ReplicationFactor: 2 Configs:
Topic: ETC Partition: 0 Leader: 2 Replicas: 2,3 Isr: 2,3
Topic: ETC Partition: 1 Leader: 3 Replicas: 3,1 Isr: 3,1
Topic: ETF PartitionCount: 2 ReplicationFactor: 3 Configs:
Topic: ETF Partition: 0 Leader: 1 Replicas: 1,3,2 Isr: 1,3,2
Topic: ETF Partition: 1 Leader: 2 Replicas: 2,1,3 Isr: 2,1,3
Topic: ETN PartitionCount: 1 ReplicationFactor: 1 Configs:
Topic: ETN Partition: 0 Leader: 1 Replicas: 1 Isr: 1
root@24f7668918ee:/#
```

Rysunek 22 Podsumowanie tematów CommonStocks, ETF, ETC, ETN

Zgodnie z powyższym na przykład temat *CommonStocks* został rozłożony na trzy partycje, które są zarządzane przez odpowiadające im brokery, powodując wysoką dostępność danych. To znaczy partycja o zerowym numerze (Partition: 0) jest zarządzana przez pierwszy broker (Leader: 1). Analogicznie dla pozostałych partycji. W przypadku tematu ETF stworzone dwie partycje są zarządzane poprzez pierwszy i drugi broker (Partition: 0, Leader: 1, Partition: 1, Leader: 2) z replikami dostępnymi na każdym brokerze (Replicas: 1,3,2 oraz 2,1,3 odpowiednio). Tematy te są odporne na utratę danych - w przypadku awarii dwóch z trzech



brokerów rolę lidera przejmie pozostały broker i dane przetwarzane zostaną odzyskane ze stworzonych repliki. Na poniższym schemacie przedstawiłem uproszczoną architekturę rozłożenia tematów w klastrze, gdzie CS-0 jest partycją 0 tematu CommonStocks oraz kropkowane wypełnienie wskazuje na replikę.



Rysunek 23 Architektura rozłożenia tematów w klastrze Apache Kafka

#### 4.4 Przetwarzanie napływających danych za pomocą Spark Streaming

Odebranie danych z tematów stworzonych na klastrze Apache Kafka w poprzednim podrozdziale oraz ich agregacja zostały wykonane z wykorzystaniem notatnika Jupyter i SparkStreaming. Po uruchomieniu kontenera notatnik Jupyter jest on dostępny pod adresem lokalnego hosta: <http://localhost:8888/> - token znajduje się bezpośrednio w logach kontenera, jak na poniższym rysunku.

```

pnowicki ~ $docker logs projekt_pyspark-notebook_1
Executing the command: jupyter notebook
[I 20:24:55.773 NotebookApp] Writing notebook server cookie secret to /home/jovyan/.local/share/jupyter/runtime/notebook_cookie_secret
[I 20:25:01.104 NotebookApp] JupyterLab extension loaded from /opt/conda/lib/python3.7/site-packages/jupyterlab
[I 20:25:01.110 NotebookApp] JupyterLab application directory is /opt/conda/share/jupyter/lab
[I 20:25:01.131 NotebookApp] Serving notebooks from local directory: /home/jovyan
[I 20:25:01.131 NotebookApp] The Jupyter Notebook is running at:
[I 20:25:01.132 NotebookApp] http://d0d8ca3744e5:8888/?token=851093412280bc0fa4439ddf33b19281d4153351cb5cf0f6
[I 20:25:01.134 NotebookApp] or http://127.0.0.1:8888/?token=851093412280bc0fa4439ddf33b19281d4153351cb5cf0f6
[I 20:25:01.135 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 20:25:01.164 NotebookApp]

```

Rysunek 24 Logi kontenera z notatnikiem Jupyter

Po określeniu dostępu do notatnika wykorzystałem poniższy skrypt w celu agregacji wolumetrii transakcji dla akcji (odebranych z tematu CommonStocks) w liczoną w 5-minutowych odstępach czasu.

```

In [1]: import os
import json
from pyspark import SparkContext
from pyspark.streaming import StreamingContext
from pyspark.streaming.kafka import KafkaUtils

In [2]: # streaming setup
os.environ['PYSARK_SUBMIT_ARGS'] = '--packages org.apache.spark:spark-streaming-kafka-0-8_2.11:2.4.5 pyspark-shell'
brokers = "kafka-1:19091,kafka-2:29091,kafka-3:39091"
kafkaParams = {"metadata.broker.list": brokers}
topics = ["CommonStocks"]
sc = SparkContext("local[2]", "CommonStocks")
ssc = StreamingContext(sc, 300)
ssc.checkpoint('/tmp')

In [3]: # read & process
kafka_raw_stream = KafkaUtils.createDirectStream(ssc, topics, kafkaParams)
kafka_mapped = kafka_raw_stream.map(lambda x: (json.loads(x[1])["SecurityDesc"],
                                              json.loads(x[1])["TradedVolume"] )

In [4]: def updateFunction(newValues, runningCount):
    if runningCount is None:
        runningCount = 0
    return sum(newValues, runningCount)
kafka_stream_stateful = kafka_mapped.updateStateByKey(updateFunction)

In [5]: kafka_stream_stateful.repartition(1).saveAsTextFiles("work/outputs/CommonStocks/cumulated-traded-volume")
kafka_stream_stateful.pprint()

In [6]: ssc.start() # Start the computation
ssc.awaitTermination() # Wait for the computation to terminate

...

In [6]: ssc.stop(True, True)

```

Rysunek 25 Skrypt notatnika Jupyter wykorzystany do agregacji informacji o wolumetrii transakcji na giełdzie

Skrypt zakłada pobranie i aktualizację każdego z klucza instrumentu finansowego dla akcji (ISIN – ustawione jako wartość atrybutu *key* w rozdziale 4.2 podpunkt 1.2.9.1). Poprzez wybranie z napływających danych nazwy *SecurityDesc* oraz wartości *TradedVolume*. Następnie wykonywana zostaje funkcja *updateFunction* przyjmująca jako argumenty aktualne oraz zwracane zdarzenie - *runningCount* - w metodzie *updateStateByKey()*. Wielkość agregatu jest liczona od początku przetwarzania danych, aż do końca rozważanego okna czasowego. Istnieje tutaj stempel czasu + 5 minut odczytywania danych. Wydruki z notatnika - z komórki [6] – wykonanego 2020-06-05 15:35:00 przez 30 minut do 2020-06-05 16:10:00 są dostępne w załączniku 7.2 i mają następującą postać:

```

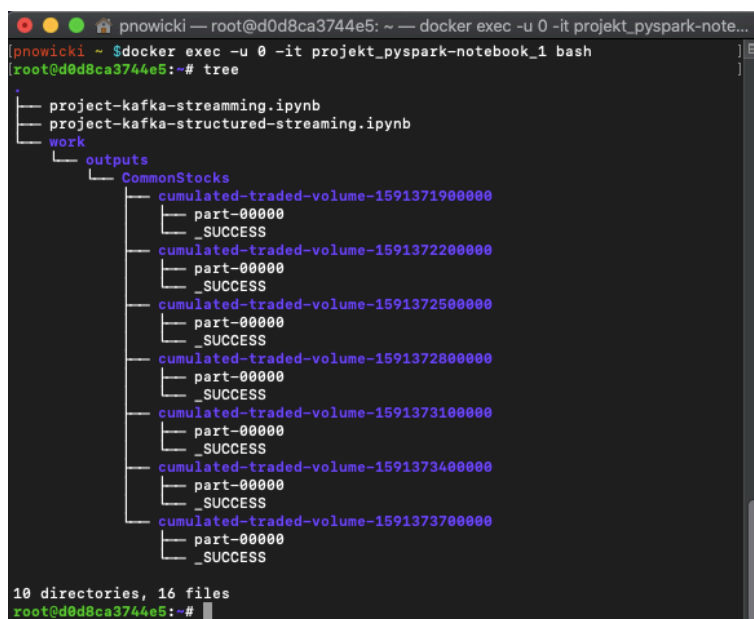
-----
Time: 2020-06-05 15:45:00
-----
('MERCK KGAA O.N.', 6094)
('AROUNDTOWN EO-,01', 76791)
('LINDE PLC EO 0,001', 11759)
('AIRBUS', 12936)
('BASF SE NA O.N.', 49170)
('RWE AG INH O.N.', 304931)
('MPC MUENCH.PET.CAP.', 11473)
('SYMRISE AG INH. O.N.', 15422)
('XPHYTO THERAPEUTICS', 16500)
('QSC AG NA O.N.', 55000)
...

```

Interpretacja wydruku jest następująca:

*od początku przetwarzania danych, tzn w okresie od 2020-06-05 15:35:00 (wykonanie skryptu) do końca okna czasowego 2020-06-05 15:45:00 zakupiono 6094 akcji firmy MERCK KGAA O.N., 12936 akcji firmy AIRBUS itd.*

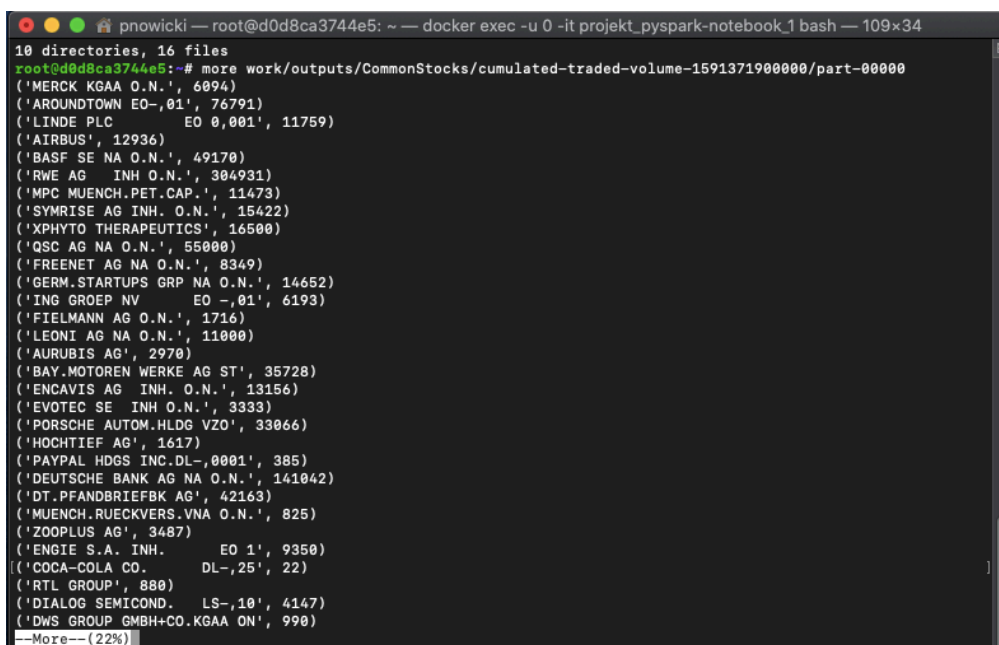
Wydruki agregatów są także zapisywane w lokalizacji *work/outputs/CommonStock/* w kontenerze<sup>15</sup> (poprzez metodę *saveAsTextFiles*). Nazewnictwo plików wynikowych zawiera przedrostek *cumulated-traded-volume-*, do którego dopisywany jest stempel czasowy. Struktura folderów została zobrazowana na poniższym rysunku.



```
pnowicki ~ $ docker exec -u 0 -it projekt_pyspark-notebook_1 bash
root@d0d8ca3744e5:~# tree
.
├── project-kafka-streaming.ipynb
├── project-kafka-structured-streaming.ipynb
├── work
│   └── outputs
│       └── CommonStocks
│           ├── cumulated-traded-volume-1591371900000
│           │   ├── part-00000
│           │   └── _SUCCESS
│           ├── cumulated-traded-volume-1591372200000
│           │   ├── part-00000
│           │   └── _SUCCESS
│           ├── cumulated-traded-volume-1591372500000
│           │   ├── part-00000
│           │   └── _SUCCESS
│           ├── cumulated-traded-volume-1591372800000
│           │   ├── part-00000
│           │   └── _SUCCESS
│           ├── cumulated-traded-volume-1591373100000
│           │   ├── part-00000
│           │   └── _SUCCESS
│           ├── cumulated-traded-volume-1591373400000
│           │   ├── part-00000
│           │   └── _SUCCESS
│           └── cumulated-traded-volume-1591373700000
│               ├── part-00000
│               └── _SUCCESS
└── 10 directories, 16 files
root@d0d8ca3744e5:~#
```

Rysunek 26 Struktura folderów zapisanego wydruku analiz

Zgodnie z powyższym plik *cumulated-traded-volume-1591371900000/part-00000* zawiera zrzut wyniku streamingu z momentu o stemple czasowym równym 1591371900000 (to znaczy 2020-06-05 15:45:00). Poniższy rysunek przedstawia odczyt tego pliku, wynik odpowiada powyższemu wydrukowi oraz umieszonemu w załączniku 7.2.



```
pnowicki ~ $ docker exec -u 0 -it projekt_pyspark-notebook_1 bash
10 directories, 16 files
root@d0d8ca3744e5:~# more work/outputs/CommonStocks/cumulated-traded-volume-1591371900000/part-00000
('MERCK KGAA O.N.', 6094)
('AROUNDTOWN EO-,01', 76791)
('LINDE PLC EO 0,001', 11759)
('AIRBUS', 12936)
('BASF SE NA O.N.', 49170)
('RWE AG INH O.N.', 304931)
('MPC MUENCH.PET.CAP.', 11473)
('SYMRISE AG INH. O.N.', 15422)
('XPHYTO THERAPEUTICS', 16500)
('QSC AG NA O.N.', 55000)
('FREENET AG NA O.N.', 8349)
('GERM.STARTUPS GRP NA O.N.', 14652)
('ING GROEP NV EO -,01', 6193)
('FIELMANN AG O.N.', 1716)
('LEONI AG NA O.N.', 11000)
('AURUBIS AG', 2970)
('BAY.MOTOREN WERKE AG ST', 35728)
('ENCAVIS AG INH. O.N.', 13156)
('EVOTEC SE INH O.N.', 3333)
('PORSCHE AUTOM.HLDG VZO', 33066)
('HOCHTIEF AG', 1617)
('PAYPAL HDGS INC.DL-,0001', 385)
('DEUTSCHE BANK AG NA O.N.', 141042)
('DT.PFANDBRIEFBK AG', 42163)
('MUENCH.RUECKVERS.VNA O.N.', 825)
('ZOOPLUS AG', 3487)
('ENGIE S.A. INH. EO 1', 9350)
('COCA-COLA CO. DL-,25', 22)
('RTL GROUP', 880)
('DIALOG SEMICONDUCTOR LS-,10', 4147)
('DWS GROUP GMBH+CO.KGAA ON', 990)
--More-- (22%)
```

Rysunek 27 Podgląd pliku zapisanego podczas streamingu

<sup>15</sup> Dane też są łatwo dostępne poprzez komendę *docker cp* pozwalającą na transfer danych pomiędzy kontenerem, a lokalną maszyną.

## 5 Wnioski i zakończenie

W ramach niniejszej pracy zaprezentowałem pełną architekturę powstałą podczas prac projektowych. W szczególności architektura pozwala na:

- analizę i eksplorację danych w postaci batch w środowisku chmurowym,
- przetwarzanie napływających danych w czasie rzeczywistym i zapis wyników prac.

Monitorowanie liczby transakcji nie jest jedynym przypadkiem użycia, na które pozwala prezentowane środowisko. Podejście można rozszerzyć do wielu studium przypadków wykraczającym poza zdefiniowany zakres projektu. Poniżej przedstawiłem kilka możliwych rozszerzeń przedstawionego problemu:

- Dodatkowe pobranie i umieszczenie danych historycznych w bazie danych oraz systematyczne wstrzykiwanie przychodzących danych w czasie rzeczywistym pozwoliłoby na zbudowanie w pełni skalowalnej platformy informacyjnej<sup>16</sup> o cenach, wolumenie transakcji giełdowych z dokładnością co do minuty notowań.
- Wzbogacenie analiz w środowisku chmurowym o zbudowanie modeli uczenia maszynowego, na przykład sieci neuronowej LSTM do predykcji cen akcji w celu zwrócenie informacji o przeszacowaniu lub niedoszacowaniu wartości akcji.
- Dołączenie do rozważanych źródeł danych o cenach i wolumenie akcji informacji na temat:
  - spółki,
  - rynku, na którym operuje,
  - sytuacji geopolitycznej w regionie/rynku

przykładowo takie informacje jak:

- kursy walutowe,
- dane ekonomiczne,
- dane geolokalizacyjne,
- wiadomości na temat spółki, krajów, gdzie prowadzony jest biznes<sup>17</sup>,
- wiadomości o spółce z portali społecznościowych, jak Twitter<sup>18</sup>,
- sentyment wiadomości np. poprzez AWS Comprehend<sup>19</sup>,

a następnie budowa modelu predykcyjnego czy prezentacja informacji o spółce na platformie informacyjnej.

Podsumowując prezentowane rozwiązanie z powodzeniem jest w stanie w sposób w pełni skalowany pionowo jak i poziomo analizować oraz przetwarzać zarówno dane historyczne, jak i napływające dane w czasie rzeczywistym. Dodatkowo oferuje wiele przypadków wykorzystania, co czyni je bardzo wartościowym nie tylko pod kątem technicznym, ale także biznesowym.

---

<sup>16</sup> W tym celu jako bazę danych mogłaby posłużyć Cassandra, a wizualizacja danych odbyć się w Grafanie

<sup>17</sup> Na przykład poprzez wykorzystanie modułu wikiextractor pozwalającego na pobieranie danych z Wikipedii (też wiadomości)

<sup>18</sup> Spark oferuje klasę TwitterUtils umożliwiającą bezpośrednie wstrzyknięcie danych na temat wiadomości na Twitterze

<sup>19</sup> <https://docs.aws.amazon.com/comprehend/latest/dg/how-sentiment.html>

## 6 Bibliografia

- [1] <https://registry.opendata.aws/deutsche-boerse-pds/>
- [2] <https://aws.amazon.com/emr/?whats-new-cards.sort-by=item.additionalFields.postDateTime&whats-new-cards.sort-order=desc>
- [3] <https://aws.amazon.com/ec2/instance-types/>
- [4] <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-managed-notebooks.html>
- [5] <https://spark.apache.org/docs/latest/streaming-programming-guide.html>
- [6] <https://docs.docker.com/compose/>
- [7] <https://hub.docker.com/r/apache/nifi/>
- [8] <https://hub.docker.com/r/confluentinc/cp-kafka/>
- [9] <https://hub.docker.com/r/confluentinc/cp-zookeeper>
- [10] <https://hub.docker.com/r/jupyter/pyspark-notebook/>
- [11] <https://hub.docker.com>
- [12] <https://nifi.apache.org/docs.html>
- [13] <https://kafka.apache.org/documentation/>

## 7 Załączniki

### 7.1 Zawartość pliku *project.yml*

```
---
version: '1'
services:
  zookeeper-1:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_SERVER_ID: 1
      ZOOKEEPER_CLIENT_PORT: 12181
      ZOOKEEPER_TICK_TIME: 2000
      ZOOKEEPER_INIT_LIMIT: 5
      ZOOKEEPER_SYNC_LIMIT: 2
      ZOOKEEPER_SERVERS: zookeeper-1:12888:13888;zookeeper-2:22888:23888;zookeeper-3:32888:33888

  zookeeper-2:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_SERVER_ID: 2
      ZOOKEEPER_CLIENT_PORT: 22181
      ZOOKEEPER_TICK_TIME: 2000
      ZOOKEEPER_INIT_LIMIT: 5
      ZOOKEEPER_SYNC_LIMIT: 2
      ZOOKEEPER_SERVERS: zookeeper-1:12888:13888;zookeeper-2:22888:23888;zookeeper-3:32888:33888

  zookeeper-3:
    image: confluentinc/cp-zookeeper:latest
    environment:
      ZOOKEEPER_SERVER_ID: 3
      ZOOKEEPER_CLIENT_PORT: 32181
      ZOOKEEPER_TICK_TIME: 2000
      ZOOKEEPER_INIT_LIMIT: 5
      ZOOKEEPER_SYNC_LIMIT: 2
      ZOOKEEPER_SERVERS: zookeeper-1:12888:13888;zookeeper-2:22888:23888;zookeeper-3:32888:33888

  kafka-1:
    image: confluentinc/cp-kafka:latest
    depends_on:
      - zookeeper-1
      - zookeeper-2
      - zookeeper-3
    environment:
      KAFKA_BROKER_ID: 1
      KAFKA_ZOOKEEPER_CONNECT: zookeeper-1:12181,zookeeper-2:22181,zookeeper-3:32181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka-1:19091,PLAINTEXT_HOST://localhost:19092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
    ports:
      - 19092:19092

  kafka-2:
    image: confluentinc/cp-kafka:latest
    depends_on:
      - zookeeper-1
      - zookeeper-2
      - zookeeper-3
    environment:
      KAFKA_BROKER_ID: 2
      KAFKA_ZOOKEEPER_CONNECT: zookeeper-1:12181,zookeeper-2:22181,zookeeper-3:32181
      KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka-2:29091,PLAINTEXT_HOST://localhost:29092
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
      KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
    ports:
      - 29092:29092

  kafka-3:
    image: confluentinc/cp-kafka:latest
    depends_on:
      - zookeeper-1
      - zookeeper-2
      - zookeeper-3
```

```
environment:
  KAFKA_BROKER_ID: 3
  KAFKA_ZOOKEEPER_CONNECT: zookeeper-1:12181,zookeeper-2:22181,zookeeper-3:32181
  KAFKA_ADVERTISED_LISTENERS: PLAINTEXT://kafka-3:39091,PLAINTEXT_HOST://localhost:39092
  KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT_HOST:PLAINTEXT
  KAFKA_INTER_BROKER_LISTENER_NAME: PLAINTEXT
ports:
  - 39092:39092

nifi:
  image: apache/nifi:latest
  ports:
    - 8080:8080

pyspark-notebook:
  image: jupyter/pyspark-notebook:latest
  ports:
    - 8888:8888
```

## 7.2 Wydruk przetworzonych danych w czasie rzeczywistym

```
-----
Time: 2020-06-05 15:35:00
-----
```

```
('MERCK KGAA O.N.', 554)
('AROUNDTOWN EO-,01', 6981)
('LINDE PLC          EO 0,001', 1069)
('AIRBUS', 1176)
('BASF SE NA O.N.', 4470)
('RWE AG   INH O.N.', 27721)
('MPC MUENCH.PET.CAP.', 1043)
('SYMRISE AG INH. O.N.', 1402)
('XPHYTO THERAPEUTICS', 1500)
('QSC AG NA O.N.', 5000)
...
```

```
-----
Time: 2020-06-05 15:40:00
-----
```

```
('MERCK KGAA O.N.', 3324)
('AROUNDTOWN EO-,01', 41886)
('LINDE PLC          EO 0,001', 6414)
('AIRBUS', 7056)
('BASF SE NA O.N.', 26820)
('RWE AG   INH O.N.', 166326)
('MPC MUENCH.PET.CAP.', 6258)
('SYMRISE AG INH. O.N.', 8412)
('XPHYTO THERAPEUTICS', 9000)
('QSC AG NA O.N.', 30000)
...
```

```
-----
Time: 2020-06-05 15:45:00
-----
```

```
('MERCK KGAA O.N.', 6094)
('AROUNDTOWN EO-,01', 76791)
('LINDE PLC          EO 0,001', 11759)
('AIRBUS', 12936)
('BASF SE NA O.N.', 49170)
('RWE AG   INH O.N.', 304931)
('MPC MUENCH.PET.CAP.', 11473)
('SYMRISE AG INH. O.N.', 15422)
('XPHYTO THERAPEUTICS', 16500)
('QSC AG NA O.N.', 55000)
...
```

```
-----
Time: 2020-06-05 15:50:00
-----
```

```
('MERCK KGAA O.N.', 8864)
('AROUNDTOWN EO-,01', 111696)
('LINDE PLC          EO 0,001', 17104)
('AIRBUS', 18816)
('BASF SE NA O.N.', 71520)
('RWE AG   INH O.N.', 443536)
('MPC MUENCH.PET.CAP.', 16688)
```

('SYMRISE AG INH. O.N.', 22432)  
('XPHYTO THERAPEUTICS', 24000)  
('QSC AG NA O.N.', 80000)  
...

-----  
Time: 2020-06-05 15:55:00  
-----

('MERCK KGAA O.N.', 11634)  
('AROUNDTOWN EO-,01', 146601)  
('LINDE PLC EO 0,001', 22449)  
('AIRBUS', 24696)  
('BASF SE NA O.N.', 93870)  
('RWE AG INH O.N.', 582141)  
('MPC MUENCH.PET.CAP.', 21903)  
('SYMRISE AG INH. O.N.', 29442)  
('XPHYTO THERAPEUTICS', 31500)  
('QSC AG NA O.N.', 105000)  
...

-----  
Time: 2020-06-05 16:00:00  
-----

('MERCK KGAA O.N.', 14404)  
('AROUNDTOWN EO-,01', 181506)  
('LINDE PLC EO 0,001', 27794)  
('AIRBUS', 30576)  
('BASF SE NA O.N.', 116220)  
('RWE AG INH O.N.', 720746)  
('MPC MUENCH.PET.CAP.', 27118)  
('SYMRISE AG INH. O.N.', 36452)  
('XPHYTO THERAPEUTICS', 39000)  
('QSC AG NA O.N.', 130000)  
...

-----  
Time: 2020-06-05 16:05:00  
-----

('MERCK KGAA O.N.', 17174)  
('AROUNDTOWN EO-,01', 216411)  
('LINDE PLC EO 0,001', 33139)  
('AIRBUS', 36456)  
('BASF SE NA O.N.', 138570)  
('RWE AG INH O.N.', 859351)  
('MPC MUENCH.PET.CAP.', 32333)  
('SYMRISE AG INH. O.N.', 43462)  
('XPHYTO THERAPEUTICS', 46500)  
('QSC AG NA O.N.', 155000)  
...

-----  
Time: 2020-06-05 16:10:00  
-----

('MERCK KGAA O.N.', 19944)  
('AROUNDTOWN EO-,01', 251316)  
('LINDE PLC EO 0,001', 38484)  
('AIRBUS', 42336)  
('BASF SE NA O.N.', 160920)  
('RWE AG INH O.N.', 997956)  
('MPC MUENCH.PET.CAP.', 37548)  
('SYMRISE AG INH. O.N.', 50472)  
('XPHYTO THERAPEUTICS', 54000)  
('QSC AG NA O.N.', 180000)  
...