Sprawozdanie

Uwagi

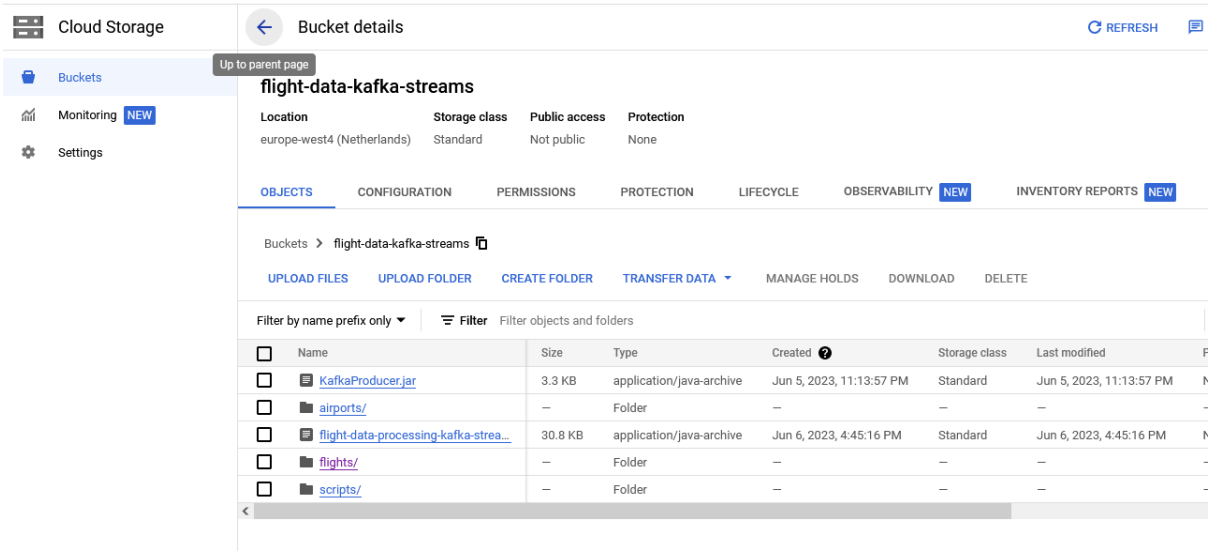
- Jeśli jakiś skrypt nie zadziałał poprawnie zalecam sprawdzić co się w nim znajduje i włączać komendy jedna po drugiej, żeby znaleźć źródło problemu.
- Moje zmienne środowiskowe:

```
export REGION=europe-west4
export ZONE=${REGION}-c
export CLUSTER_NAME=bigdata-intro
export PROJECT_ID=$(gcloud config get-value project)
```

- mój bucket nazwałem flight-data-kafka-streams
- dane o lotniskach:
  www.cs.put.poznan.pl/kjankiewicz/bigdata/stream_project/airports.csv
- dane o lotach:
  www.cs.put.poznan.pl/kjankiewicz/bigdata/stream_project/flights-2015.zip

Jak uruchomić?

1. Skopiuj do swojego bucketa (zapamiętaj nazwę swojego bucketa, przyda się przy uruchamianiu jednego ze skryptów).
   - dane o lotniskach do folderu “airports”
   - spakowane w zip dane o lotach do folderu “flights” (to może chwile zająć)
   - producenta flight-data-processing-kafka-streams.jar
   - aplikację do przetwarzania danych KafkaProducer.jar
   - folder “scripts”



2. Uruchom klaster
   ***gcloud dataproc clusters create ${CLUSTER_NAME} ***
   ***--enable-component-gateway --region ${REGION} --subnet default ***
   ***--master-machine-type n1-standard-2 --master-boot-disk-size 50 ***

***--num-workers 2 --worker-machine-type n1-standard-2 --worker-boot-disk-size 50 ***
***--image-version 2.1-debian11 --optional-components DOCKER,ZOOKEEPER ***
***--project ${PROJECT_ID} --max-age=3h ***
***--metadata "run-on-master=true" ***
***--initialization-actions ***
gs://goog-dataproc-initialization-actions-${REGION}/kafka/kafka.sh

3. Uruchom 4 terminale
    - serwerowy - do uruchamiania skryptów i uruchomienia producenta
    - odbiorczy_1 - do odbierania danych etl
    - odbiorczy_2 - do odbierania danych o anomaliach
    - strumieniowy - do uruchomienia aplikacji do przetwarzania danych

4. Przygotuj skrypty
   hadoop fs -copyToLocal gs://flight-data-kafka-streams/scripts

   chmod +x scripts/*

5. Skopiuj potrzebne dane z bucketa
   ./scripts/01_download_data.sh NAZWA_TWOJEGO_BUCKETA

   Jako argument przekaż nazwę swojego bucketa. Skrypt powinien skopiować dane o lotniskach i lotach, producenta kafki, aplikację do przetwarzania danych. Następnie powinien stworzyć odpowiednie foldery, przenieść do nich dane, odpakować dane związane z lotami oraz usunąć plik .zip.

6. Przygotuj tematy kafki
   ./scripts/02_prepare_kafka.sh

   Skrypt ma za zadanie usunąć tematy kafki (nie przejmuj się errorami jeśli tematy nie istnieją), stworzyć tematy kafki oraz przesłać statyczne dane do jednego z tematów.

7. Włącz konsumera dla etl (terminal odbiorczy_1)
   ./scripts/03_start_etl_consumer.sh

   Skrypt ma za zadanie włączyć konsumera dla danych etl. Po kliknięciu enter możesz przejść do kolejnego kroku, konsument zaczął działać.

8. Włącz konsumera dla anomalii (terminal odbiorczy_2)
   ./scripts/04_start_anomaly_consumer.sh

   Skrypt ma za zadanie włączyć consumera dla anomalii. Po kliknięciu enter możesz przejść do kolejnego kroku, konsument zaczął działać.

9. Włącz producenta, który będzie generował dane dotyczące lotów (terminal serwerowy)
   ./scripts/05_run_kafka_producer.sh

Producent powinien generować dane do tematu *flights-input* korzystając ze 100 plików, w których są przechowywane. Po kliknięciu enter możesz przejść do kolejnego kroku, producent zaczął wysyłać dane..

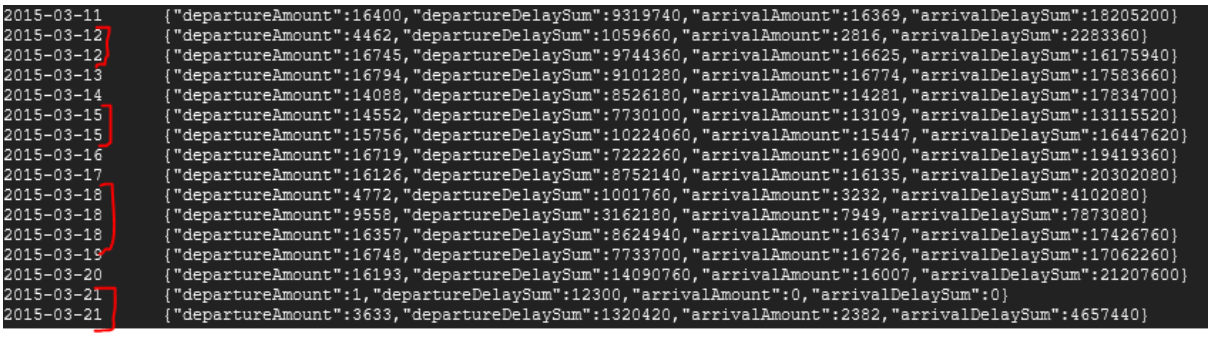
10. Włącz aplikację do przetwarzania danych (terminal strumieniowy)

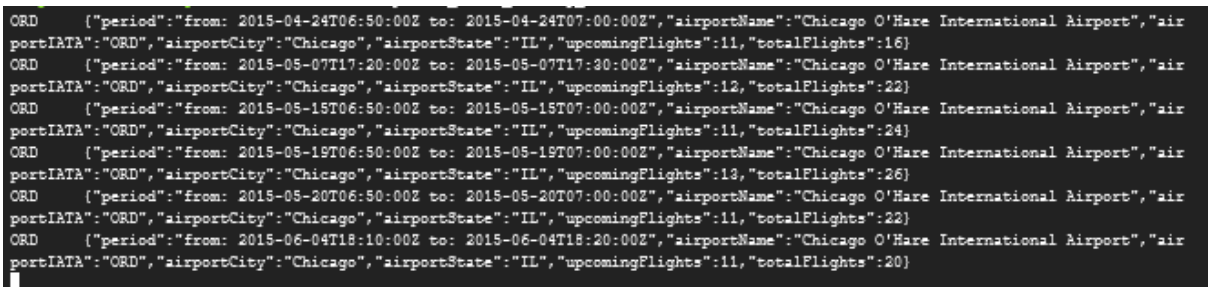
    Przed włączeniem aplikacji ustaw **CLUSTER_NAME**, aby kafka poprawnie zadziałała:

    CLUSTER_NAME=$(/usr/share/google/get_metadata_value attributes/dataproc-cluster-name)

    a) Działanie w trybie A + mało anomalii

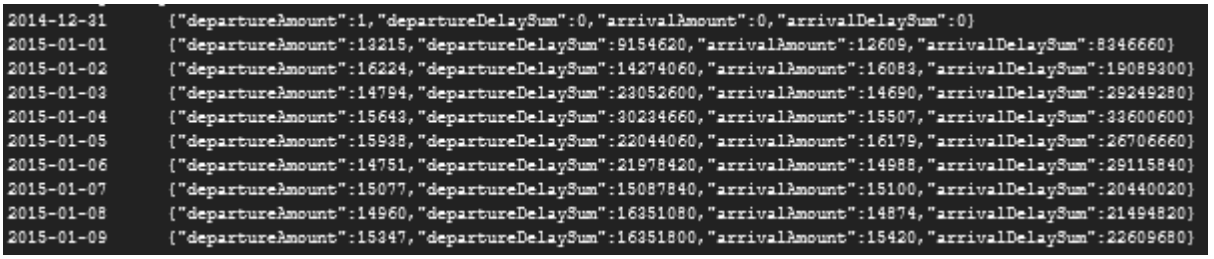
java -cp /usr/lib/kafka/libs/*:flight-data-processing-kafka-streams.jar org.example.ProcessFlightApplication 45 10 A ${CLUSTER_NAME}-w-0:9092

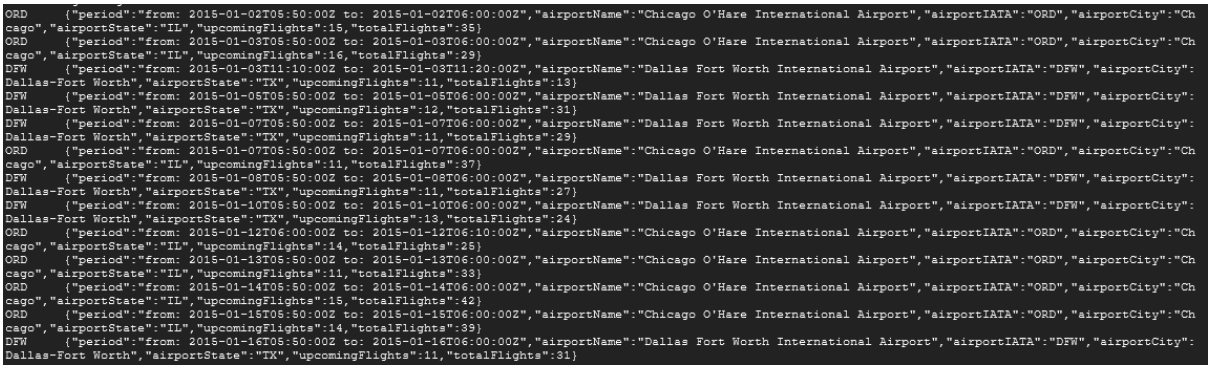




    b) Działanie w trybie C + dużo anomalii

java -cp /usr/lib/kafka/libs/*:flight-data-processing-kafka-streams.jar \ org.example.ProcessFlightApplication 60 5 C ${CLUSTER_NAME}-w-0:9092





10. Obserwuj pojawiające się dane w konsumerach etl i anomalii.
    - dane prawdopodobnie nie pojawią się od razu, trzeba chwilę poczekać, na pojawienie się pierwszych wyników.
    - jeśli chcesz zmienić parametry przy uruchamianiu aplikacji do przetwarzania danych to po prostu przerwij działanie programu (ctrl + c) i włącz z nowymi parametrami

Opis

Producent; skrypty inicjujące i zasilający

- skrypty należy skopiować do bucketa
- skrypty należy skopiować z bucketa ***hadoop fs -copyToLocal gs://flight-data-kafka-streams/scripts***
- należy dodać możliwość wykonywania się skryptom ***chmod +x scripts/****
- skrypt tworzący i usuwający tematy kafki: ***./scripts/02_prepare_kafka.sh***
- skrypty zasilające tematy kafki: ***./scripts/02_prepare_kafka.sh*** (airports) i ***./scripts/05_run_kafka_producer.sh*** (flights)
- skrypty uruchamiające konsumerów: ***./scripts/03_start_etl_consumer.sh*** (etl) i ***./scripts/04_start_anomaly_consumer.sh*** (anomalie)

Utrzymanie obrazu czasu rzeczywistego – transformacje

```
KTable<Windowed<String>, FlightDataRecord> flightDataRecords = flightRecords
        .filter((key, value) -> isCorrectInfoType(value)) KStream<String, FlightRecord>
        .selectKey((key, value) -> FlightRecord.parseOrderColumns(value.getOrderColumn()))
        .groupByKey(Grouped.with(Serdes.String(), flightSerde)) KGroupedStream<String, FlightRecord>
        .windowedBy(TimeWindows.ofSizeWithNoGrace(Duration.ofDays(1))) TimeWindowedKStream<String, FlightRecord>
        .aggregate(
                FlightDataRecord::new,
                (key, value, aggregate) -> {
                    if (value.getInfoType().equals("D")) {
                        aggregate.setDepartureAmount(aggregate.getDepartureAmount() + 1);
                        Long departureDelaySum = FlightRecord.subtractTwoDepartureDates(value.getScheduledDepartureTime(), value.getDepartureTime());
                        aggregate.setDepartureDelaySum(aggregate.getDepartureDelaySum() + departureDelaySum);

                        aggregate.setArrivalAmount(aggregate.getArrivalAmount());
                        aggregate.setArrivalDelaySum(aggregate.getArrivalDelaySum());
                    } else if (value.getInfoType().equals("A")) {
                        aggregate.setArrivalAmount(aggregate.getArrivalAmount() + 1);
                        Long arrivalDelaySum = FlightRecord.subtractTwoArrivalDates(value.getScheduledArrivalTime(), value.getArrivalTime());
                        aggregate.setArrivalDelaySum(aggregate.getArrivalDelaySum() + arrivalDelaySum);

                        aggregate.setDepartureAmount(aggregate.getDepartureAmount());
                        aggregate.setDepartureDelaySum(aggregate.getDepartureDelaySum());
                    } else {
                        aggregate.setDepartureDelaySum(aggregate.getDepartureDelaySum());
                        aggregate.setDepartureAmount(aggregate.getDepartureAmount());
                        aggregate.setArrivalAmount(aggregate.getArrivalAmount());
                        aggregate.setArrivalDelaySum(aggregate.getArrivalDelaySum());
                    }

                    return aggregate;
                },
                Materialized.with(Serdes.String(), flightDataSerde)
        );
```

- filter -> sprawdza czy to poprawny event
- selectKey -> wybiera datę jako klucz
- groupByKey -> grupuje dane po kluczu
- windowedBy -> tworzy okno o długości 1 dzień
- aggregate ->
    - jeśli event dotyczy wylotu (D), to inkrementuje liczbę wylotów oraz dodaję opóźnienie w wylocie do sumy opóźnień
    - jeśli event dotyczy przylotu (A), to inkrementuje liczbę przylotów oraz dodaję opóźnienie w przylocie do sumy opóźnień
    - w innym wypadku ustawia poprzednie dane

Utrzymanie obrazu czasu rzeczywistego – obsługa trybu A

W Kafka Streams tryb A działa automatycznie, nie potrzeba żadnych ingerencji w kodzie.

Utrzymanie obrazu czasu rzeczywistego – obsługa trybu C

```
if (delay.equals("C")) {
    return flightDataRecords.suppress(Suppressed.untilWindowCloses(Suppressed.BufferConfig.unbounded()));
}
```

Jeśli program został uruchomiony w obsłudze trybu C, to stosowany jest suppress, który pozwala wysłać dane do tematu kafki dopiero, kiedy okno się zamknie.

Wykrywanie anomalii

```
KTable<Windowed<String>, CountFlightsRecord> countFlightsRecords = flightRecordsIata
        .filter((key, value) -> value.getInfoType().equals("D")) KStream<String, FlightRecord>
        .groupByKey(Grouped.with(Serdes.String(), flightSerde)) KGroupedStream<String, FlightRecord>
        .windowedBy(TimeWindows.ofSizeAndGrace(Duration.ofMinutes(10), Duration.ofMinutes(5))) TimeWindo
        .aggregate(
                CountFlightsRecord::new,
                (key, value, aggregate) -> {
                    SimpleDateFormat dateFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd HH:mm:ss");
                    try {
                        aggregate.setTotalFlights(aggregate.getTotalFlights() + 1);

                        Date timeNow = dateFormat.parse(value.getOrderColumn());
                        Date timeArrival = dateFormat.parse(value.getScheduledArrivalTime());
                        long duration = timeArrival.getTime() - timeNow.getTime();
                        long diffInMinutes = TimeUnit.MILLISECONDS.toMinutes(duration);

                        if (diffInMinutes > 30 && diffInMinutes <= (30 + D)) {
                            aggregate.setUpcomingFlights(aggregate.getUpcomingFlights() + 1);
                        } else {
                            aggregate.setUpcomingFlights(aggregate.getUpcomingFlights());
                        }

                    } catch (ParseException e) {
                        throw new RuntimeException(e);
                    }

                    return aggregate;
                }, Materialized.with(Serdes.String(), countFlightsSerde)
        );
```

- filter -> sprawdzenie czy event to wylot (D)
- groupByKey -> grupuje dane po kluczu (airportIATA)
- windowedBy -> tworzy okno o długości 10 minut, z możliwością opóźnień o 5 minut
- aggregate ->
    - dodaje 1 do sumy lotów
    - sprawdza czy samolot doleci do lotniska między 30 a 30 + D minut
        - jeśli tak, to dodaje 1 do sumy zbliżających się lotów
        - jesli nie, to zostawia poprzednią sumę

```
return countFlights
        .map(ProcessFlightApplication::extractWindowTimestamps) KStream<String, CountFlightsPeriodRecord>
        .join(airportsTable, flightAirportJoiner, Joined.with(Serdes.String(), flightsPeriodRecordSerde, airportSerde))
        .toTable(Materialized.with(Serdes.String(), enrichedFlightSerde)) KTable<String, EnrichedFlightRecord>
        .mapValues((key, value) -> getAnomalyRecord(value), Materialized.with(Serdes.String(), anomalySerde));
```

- map -> wyciąga dane dotyczące początku i końca okna
- join -> łączy ze sobą dane o liczbie lotów oraz dane o lotniskach

- mapValues ->

```
private static AnomalyRecord getAnomalyRecord(EnrichedFlightRecord value) {
    AirportRecord airportRecord = value.getAirportRecord();
    CountFlightsPeriodRecord countFlightsPeriodRecord = value.getCountFlightsRecord();

    return new AnomalyRecord(
            period: "from: " + countFlightsPeriodRecord.getStartTs() + " to: " + countFlightsPeriodRecord.getEndTs(),
            airportRecord.getName(),
            airportRecord.getIata(),
            airportRecord.getCity(),
            airportRecord.getState(),
            countFlightsPeriodRecord.getCountFlightsRecord().getUpcomingFlights(),
            countFlightsPeriodRecord.getCountFlightsRecord().getTotalFlights()
    );
}
```

zwraca obiekt AnomalyRecord, który posiada wszystkie potrzebne informacje odnośnie anomalii.

Program przetwarzający strumienie danych; skrypt uruchamiający

a) Działanie w trybie A + mało anomalii

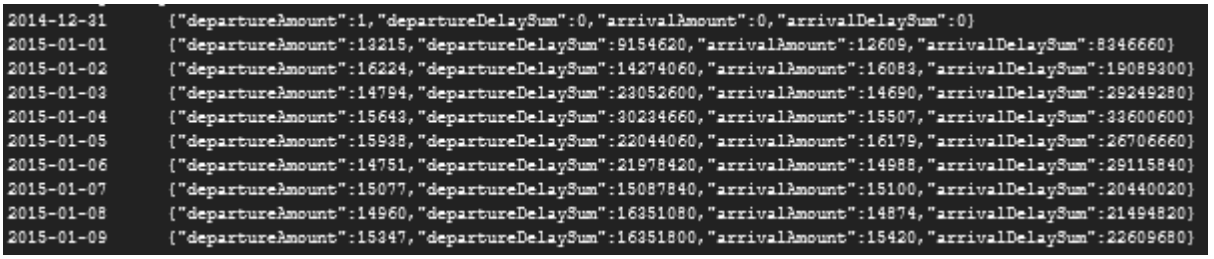
java -cp /usr/lib/kafka/libs/*:flight-data-processing-kafka-streams.jar org.example.ProcessFlightApplication 45 10 A ${CLUSTER_NAME}-w-0:9092

```
2015-03-11      {"departureAmount":16400,"departureDelaySum":9319740,"arrivalAmount":16369,"arrivalDelaySum":18205200}
2015-03-12      {"departureAmount":4462,"departureDelaySum":1059660,"arrivalAmount":2816,"arrivalDelaySum":2283360}
2015-03-12      {"departureAmount":16745,"departureDelaySum":9744360,"arrivalAmount":16625,"arrivalDelaySum":16175940}
2015-03-13      {"departureAmount":16794,"departureDelaySum":9101280,"arrivalAmount":16774,"arrivalDelaySum":17583660}
2015-03-14      {"departureAmount":14088,"departureDelaySum":8526180,"arrivalAmount":14281,"arrivalDelaySum":17834700}
2015-03-15      {"departureAmount":14552,"departureDelaySum":7730100,"arrivalAmount":13109,"arrivalDelaySum":13115520}
2015-03-15      {"departureAmount":15756,"departureDelaySum":10224060,"arrivalAmount":15447,"arrivalDelaySum":16447620}
2015-03-16      {"departureAmount":16719,"departureDelaySum":7222260,"arrivalAmount":16900,"arrivalDelaySum":19419360}
2015-03-17      {"departureAmount":16126,"departureDelaySum":8752140,"arrivalAmount":16135,"arrivalDelaySum":20302080}
2015-03-18      {"departureAmount":4772,"departureDelaySum":1001760,"arrivalAmount":3232,"arrivalDelaySum":4102080}
2015-03-18      {"departureAmount":9558,"departureDelaySum":3162180,"arrivalAmount":7949,"arrivalDelaySum":7873080}
2015-03-18      {"departureAmount":16357,"departureDelaySum":8624940,"arrivalAmount":16347,"arrivalDelaySum":17426760}
2015-03-19      {"departureAmount":16748,"departureDelaySum":7733700,"arrivalAmount":16726,"arrivalDelaySum":17062260}
2015-03-20      {"departureAmount":16193,"departureDelaySum":14090760,"arrivalAmount":16007,"arrivalDelaySum":21207600}
2015-03-21      {"departureAmount":1,"departureDelaySum":12300,"arrivalAmount":0,"arrivalDelaySum":0}
2015-03-21      {"departureAmount":3633,"departureDelaySum":1320420,"arrivalAmount":2382,"arrivalDelaySum":4657440}
```

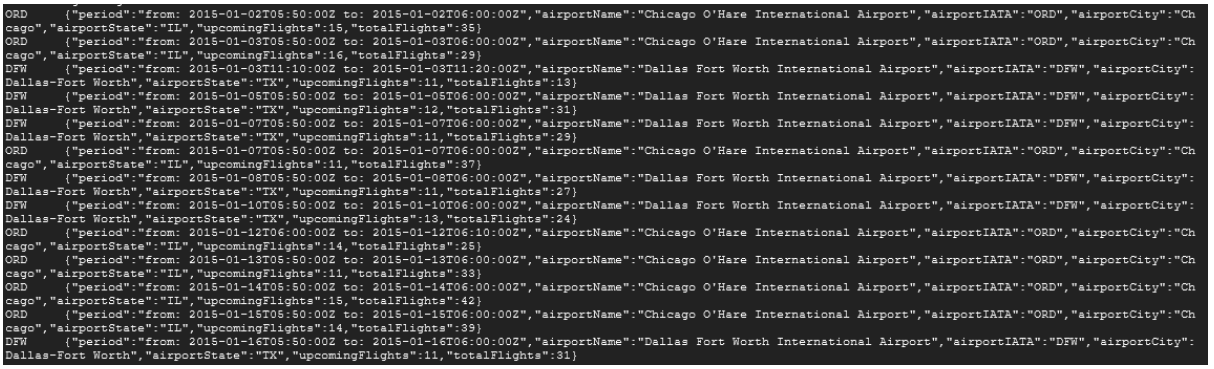
```
ORD     {"period":"from: 2015-04-24T06:50:00Z to: 2015-04-24T07:00:00Z","airportName":"Chicago O'Hare International Airport","air
portIATA":"ORD","airportCity":"Chicago","airportState":"IL","upcomingFlights":11,"totalFlights":16}
ORD     {"period":"from: 2015-05-07T17:20:00Z to: 2015-05-07T17:30:00Z","airportName":"Chicago O'Hare International Airport","air
portIATA":"ORD","airportCity":"Chicago","airportState":"IL","upcomingFlights":12,"totalFlights":22}
ORD     {"period":"from: 2015-05-15T06:50:00Z to: 2015-05-15T07:00:00Z","airportName":"Chicago O'Hare International Airport","air
portIATA":"ORD","airportCity":"Chicago","airportState":"IL","upcomingFlights":11,"totalFlights":24}
ORD     {"period":"from: 2015-05-19T06:50:00Z to: 2015-05-19T07:00:00Z","airportName":"Chicago O'Hare International Airport","air
portIATA":"ORD","airportCity":"Chicago","airportState":"IL","upcomingFlights":13,"totalFlights":26}
ORD     {"period":"from: 2015-05-20T06:50:00Z to: 2015-05-20T07:00:00Z","airportName":"Chicago O'Hare International Airport","air
portIATA":"ORD","airportCity":"Chicago","airportState":"IL","upcomingFlights":11,"totalFlights":22}
ORD     {"period":"from: 2015-06-04T18:10:00Z to: 2015-06-04T18:20:00Z","airportName":"Chicago O'Hare International Airport","air
portIATA":"ORD","airportCity":"Chicago","airportState":"IL","upcomingFlights":11,"totalFlights":20}
```

b) Działanie w trybie C + dużo anomalii

java -cp /usr/lib/kafka/libs/*:flight-data-processing-kafka-streams.jar \ org.example.ProcessFlightApplication 60 10 C ${CLUSTER_NAME}-w-0:9092



```
2014-12-31      {"departureAmount":1,"departureDelaySum":0,"arrivalAmount":0,"arrivalDelaySum":0}
2015-01-01      {"departureAmount":13215,"departureDelaySum":9154620,"arrivalAmount":12609,"arrivalDelaySum":8346660}
2015-01-02      {"departureAmount":16224,"departureDelaySum":14274060,"arrivalAmount":16083,"arrivalDelaySum":19089300}
2015-01-03      {"departureAmount":14794,"departureDelaySum":23052600,"arrivalAmount":14690,"arrivalDelaySum":29249280}
2015-01-04      {"departureAmount":15643,"departureDelaySum":30234660,"arrivalAmount":15507,"arrivalDelaySum":33600600}
2015-01-05      {"departureAmount":15938,"departureDelaySum":22044060,"arrivalAmount":16179,"arrivalDelaySum":26706660}
2015-01-06      {"departureAmount":14751,"departureDelaySum":21978420,"arrivalAmount":14988,"arrivalDelaySum":29115840}
2015-01-07      {"departureAmount":15077,"departureDelaySum":15087840,"arrivalAmount":15100,"arrivalDelaySum":20440020}
2015-01-08      {"departureAmount":14960,"departureDelaySum":16351080,"arrivalAmount":14874,"arrivalDelaySum":21494820}
2015-01-09      {"departureAmount":15347,"departureDelaySum":16351800,"arrivalAmount":15420,"arrivalDelaySum":22609680}
```



```
ORD     {"period":"from: 2015-01-02T05:50:00Z to: 2015-01-02T06:00:00Z","airportName":"Chicago O'Hare International Airport","airportIATA":"ORD","airportCity":"Ch
cago","airportState":"IL","upcomingFlights":15,"totalFlights":35}
ORD     {"period":"from: 2015-01-03T05:50:00Z to: 2015-01-03T06:00:00Z","airportName":"Chicago O'Hare International Airport","airportIATA":"ORD","airportCity":"Ch
cago","airportState":"IL","upcomingFlights":16,"totalFlights":29}
DFW     {"period":"from: 2015-01-03T11:10:00Z to: 2015-01-03T11:20:00Z","airportName":"Dallas Fort Worth International Airport","airportIATA":"DFW","airportCity":
Dallas-Fort Worth","airportState":"TX","upcomingFlights":11,"totalFlights":13}
DFW     {"period":"from: 2015-01-05T05:50:00Z to: 2015-01-05T06:00:00Z","airportName":"Dallas Fort Worth International Airport","airportIATA":"DFW","airportCity":
Dallas-Fort Worth","airportState":"TX","upcomingFlights":12,"totalFlights":31}
DFW     {"period":"from: 2015-01-07T05:50:00Z to: 2015-01-07T06:00:00Z","airportName":"Dallas Fort Worth International Airport","airportIATA":"DFW","airportCity":
Dallas-Fort Worth","airportState":"TX","upcomingFlights":11,"totalFlights":29}
ORD     {"period":"from: 2015-01-07T05:50:00Z to: 2015-01-07T06:00:00Z","airportName":"Chicago O'Hare International Airport","airportIATA":"ORD","airportCity":"Ch
cago","airportState":"IL","upcomingFlights":11,"totalFlights":37}
DFW     {"period":"from: 2015-01-08T05:50:00Z to: 2015-01-08T06:00:00Z","airportName":"Dallas Fort Worth International Airport","airportIATA":"DFW","airportCity":
Dallas-Fort Worth","airportState":"TX","upcomingFlights":11,"totalFlights":27}
DFW     {"period":"from: 2015-01-10T05:50:00Z to: 2015-01-10T06:00:00Z","airportName":"Dallas Fort Worth International Airport","airportIATA":"DFW","airportCity":
Dallas-Fort Worth","airportState":"TX","upcomingFlights":13,"totalFlights":24}
ORD     {"period":"from: 2015-01-12T06:00:00Z to: 2015-01-12T06:10:00Z","airportName":"Chicago O'Hare International Airport","airportIATA":"ORD","airportCity":"Ch
cago","airportState":"IL","upcomingFlights":14,"totalFlights":25}
ORD     {"period":"from: 2015-01-13T05:50:00Z to: 2015-01-13T06:00:00Z","airportName":"Chicago O'Hare International Airport","airportIATA":"ORD","airportCity":"Ch
cago","airportState":"IL","upcomingFlights":11,"totalFlights":33}
ORD     {"period":"from: 2015-01-14T05:50:00Z to: 2015-01-14T06:00:00Z","airportName":"Chicago O'Hare International Airport","airportIATA":"ORD","airportCity":"Ch
cago","airportState":"IL","upcomingFlights":15,"totalFlights":42}
ORD     {"period":"from: 2015-01-15T05:50:00Z to: 2015-01-15T06:00:00Z","airportName":"Chicago O'Hare International Airport","airportIATA":"ORD","airportCity":"Ch
cago","airportState":"IL","upcomingFlights":14,"totalFlights":39}
DFW     {"period":"from: 2015-01-16T05:50:00Z to: 2015-01-16T06:00:00Z","airportName":"Dallas Fort Worth International Airport","airportIATA":"DFW","airportCity":
Dallas-Fort Worth","airportState":"TX","upcomingFlights":11,"totalFlights":31}
```

Miejsce utrzymywania obrazów czasu rzeczywistego – skrypt tworzący

skrypt tworzący tematy kafki: ***./scripts/02_prepare_kafka.sh***

temat, do którego generowane są dane odnośnie lotów
kafka-topics.sh --create --topic flights-input --bootstrap-server ${CLUSTER_NAME}-w-0:9092 --replication-factor 1 --partitions 1

temat, w którym utrzymywane są informacje o etl
kafka-topics.sh --create --topic flights-output --bootstrap-server ${CLUSTER_NAME}-w-0:9092 --replication-factor 1 --partitions 1

temat, do którego jednokrotnie dodawane są statyczne dane odnośnie lotnisk
kafka-topics.sh --create --topic airports-input --bootstrap-server ${CLUSTER_NAME}-w-0:9092 --config cleanup.policy=compact --replication-factor 1 --partitions 1

temat, w którym utrzymywane są informacje o anomaliach
kafka-topics.sh --create --bootstrap-server ${CLUSTER_NAME}-w-0:9092 --topic airports-output --replication-factor 1 --partitions 1

Miejsce utrzymywania obrazów czasu rzeczywistego – cechy

Wybrałem takie obrazy, ponieważ są one spełniają one wszystkie potrzeby dla naszego projektu: wszystkie informacje są widoczne, można szybko usuwać/tworzyć tematy, nie ma potrzeby instalacji żadnych zewnętrznych narzędzi.

Konsument: skrypt odczytujący wyniki przetwarzania

skrypt uruchamiający konsumenta etl: ***./scripts/03_start_etl_consumer.sh***

```
kafka-console-consumer.sh \
--bootstrap-server ${CLUSTER_NAME}-w-0:9092  \
--topic flights-output \
--formatter kafka.tools.DefaultMessageFormatter \
--property print.key=true \
--property print.value=true
```

skrypt uruchamiający konsumenta anomaliil: ***.04_start_anomaly_consumer.sh***

```
kafka-console-consumer.sh \
--bootstrap-server ${CLUSTER_NAME}-w-0:9092  \
--topic airports-output \
--formatter kafka.tools.DefaultMessageFormatter \
--property print.key=true \
--property print.value=true
```