

Kafka

W ramach warsztatu uruchomimy klaster Hadoop dokonując w nim instalacji Kafki na platformie GCP z wykorzystaniem *Dataproc*. Zobaczmy jak działają bardzo proste wersje producentów i konsumentów "z i do konsoli", a także napiszemy naszego własnego poważnego producenta.

Uruchomienie klastra

(5 minut)

1. Wejdź na stronę GCP, otwórz *Cloud Shell*, a następnie za pomocą poniższego polecenia uruchom swój klaster instalując w nim jednocześnie brokera wiadomości Kafka

```
gcloud dataproc clusters create ${CLUSTER_NAME} \
--enable-component-gateway --region ${REGION} --subnet default \
--master-machine-type n1-standard-4 --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 ${PROJECT_ID} --max-age=2h \
--metadata "run-on-master=true" \
--initialization-actions \
gs://goog-dataproc-initialization-actions-${REGION}/kafka/kafka.sh
```

2. Podłącz się za pomocą terminala SSH do węzła master. Wykonaj poniższe polecenie sprawdzając listę dostępnych tematów.

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

kafka-topics.sh --bootstrap-server ${CLUSTER_NAME}-w-1:9092 --list
```

3. Przetestujemy działanie naszej Kafki
 - tworząc temat Kafki, a następnie

```
kafka-topics.sh --bootstrap-server ${CLUSTER_NAME}-w-1:9092 --create \
--replication-factor 1 --partitions 1 --topic test
```

- zasilając go wiadomościami

```
for i in {0..100}; do echo "message${i}"; sleep 1; done |
/usr/lib/kafka/bin/kafka-console-producer.sh \
--broker-list ${CLUSTER_NAME}-w-0:9092 --topic test &
```

- odczytując te wiadomości

```
/usr/lib/kafka/bin/kafka-console-consumer.sh \
--bootstrap-server ${CLUSTER_NAME}-w-1:9092 \
--topic test --from-beginning
```

```
message19
message20
message21
message22
message23
message24
message25
```

Jeśli wszystko działa jak należy, na ekranie powinny pojawiać się treści wysłanych komunikatów.

Po zakończeniu nadawania możesz przerwać swojego klienta (odbiorcę komunikatów) za pomocą *Ctrl+C*

4. Sprawdźmy teraz w pliku `/usr/lib/kafka/config/server.properties` parametry Kafki z jakimi została ona uruchomiona.
- Odczytaj z parametru `zookeeper.connect` port, pod którym Zookeeper jest dostępny dla Kafki.

```
cat /usr/lib/kafka/config/server.properties | grep zookeeper.connect
```

- Odczytaj także port, na którym broker wiadomości Kafki nasłuchuje (parametr `listeners`).

```
cat /usr/lib/kafka/config/server.properties | grep listeners
```

- Znajdź parametr określający liczbę godzin, przez które Kafka będzie przechowywał wiadomości w swoich logach. Ile to dni?

W przypadku gdy nie znajdziesz lub nie znasz nazw stosownych parametrów – patrz uwaga poniżej.

5. Postaraj się znaleźć jeszcze kilka ważnych parametrów konfiguracyjnych.
- Znajdź parametr konfiguracyjny `auto.create.topics.enable`. Co oznacza jego obecne ustawienie?

```
cat /usr/lib/kafka/config/server.properties | grep auto.create.topics.enable
```

- Znajdź parametr konfiguracyjny `delete.topic.enable`. Czy usuwanie tematów jest możliwe?

```
cat /usr/lib/kafka/config/server.properties | grep delete.topic.enable
```

- Znajdź parametr `num.io.threads`. Ile wątków obsługujących operacje I/O będzie używanych?

```
cat /usr/lib/kafka/config/server.properties | grep num.io.threads
```

- Znajdź parametr określający liczbę domyślnie tworzonych partycji dla każdego z nowych tematów. Ile ich będzie?

Uwaga! W przypadku gdy nie znasz lub nie znajdziesz stosownych parametrów, sprawdź jak nazywają określone parametry oraz jaka jest ich wartość domyślna

<https://kafka.apache.org/documentation/#brokerconfigs>

Weryfikuj swoje informacje w oparciu o dokumentację dla właściwej wersji Kafki. Wersję Kafki możesz sprawdzić za pomocą poniższego polecenia

```
kafka-topics.sh --version
```

CLI

(5 minut)

Postaramy się teraz samodzielnie przetestować działanie naszego brokera wiadomości Kafka, wykorzystując do tego celu interfejs dostępny z linii poleceń.

- Korzystając z terminala SSH wyświetl dostępne obecnie tematy Kafki. Ile ich jest? Jeden z nich jest oczywisty. Do czego służy dodatkowo utworzony temat?

```
kafka-topics.sh --bootstrap-server ${CLUSTER_NAME}-w-1:9092 --list
```

- Utwórz nowy temat Kafki o nazwie kafka-tt

```
kafka-topics.sh --create \
  --bootstrap-server ${CLUSTER_NAME}-w-1:9092 \
  --replication-factor 2 --partitions 3 --topic kafka-tt
```

- Pobierz szczegóły dotyczące utworzonego tematu

```
kafka-topics.sh --describe \
  --bootstrap-server ${CLUSTER_NAME}-w-1:9092 --topic kafka-tt
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ kafka-topics.sh --describe \
  --bootstrap-server ${CLUSTER_NAME}-w-1:9092 --topic kafka-tt
Topic: kafka-tt TopicId: BfCjVOE3RRKtEBOqCeGHUw PartitionCount: 3 ReplicationFactor: 2
Topic: kafka-tt Partition: 0 Leader: 1 Replicas: 1,10000 Isr: 1,10000
Topic: kafka-tt Partition: 1 Leader: 10000 Replicas: 10000,0 Isr: 10000,0
Topic: kafka-tt Partition: 2 Leader: 0 Replicas: 0,1 Isr: 0,1
```

Wg powyższej odpowiedzi trzy serwery klastra są liderami poszczególnych partycji naszego tematu.

- Korzystając z interfejsu dostarczanego przez zookeeper-shell dowiedz się ile serwerów funkcjonuje w ramach klastra Kafki

```
/usr/lib/kafka/bin/zookeeper-shell.sh localhost:2181 ls /brokers/ids
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ /usr/lib/kafka/bin/zookeeper-shell.sh localhost:2181 ls /brokers/ids
Connecting to localhost:2181

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
[0, 1, 10000]
```

Wg powyższej odpowiedzi mamy trzy serwery w klastrze o identyfikatorach 0, 10000 i 1.

10. Dowiedz się czegoś więcej na temat serwera zarządzającego (lidera) partycją numer 0 utworzonego przez Ciebie tematu. Zwróć uwagę, że w Twoim przypadku nie musi to być serwer z identyfikatorem 10000

```
/usr/lib/kafka/bin/zookeeper-shell.sh localhost:2181 get /brokers/ids/10000
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ /usr/lib/kafka/bin/zookeeper-shell.sh localhost:2181 get /brokers/ids/10000
Connecting to localhost:2181

WATCHER::

WatchedEvent state:SyncConnected type:None path:null
{"listener_security_protocol_map":{"PLAINTEXT":"PLAINTEXT"},"endpoints":["PLAINTEXT://bigdata-bs-m.europe-west4-c.c.bigdata-course-lectures-187012.internal:9092"],"jmx_port":-1,"features":{"host":"bigdata-bs-m.europe-west4-c.c.bigdata-course-lectures-187012.internal","timestamp":"1668937315317","port":9092,"version":5}}
```

11. Dowiedz się także gdzie znajdują się i na jakich portach nasłuchują pozostałe serwery wchodzące w skład klastra Kafki

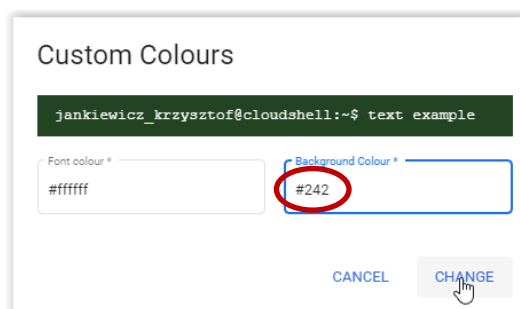
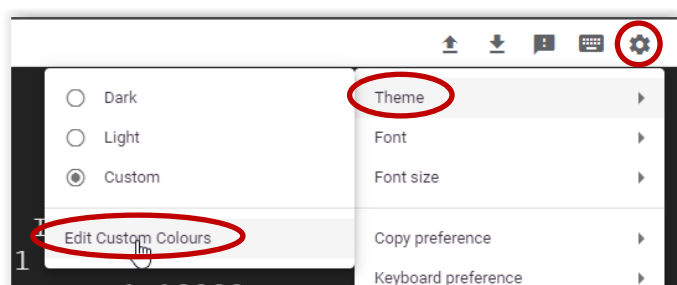
- Czy wszystkie trzy serwery są ulokowane na różnych maszynach wirtualnych?
- Czy serwer master naszego klastra został także wykorzystany przez serwer Kafki?
- Na jakich portach nasłuchują serwery Kafki?

Producenci i Konsumenci

(10 minut)

Do powyższych poleceń jeszcze będziemy wracać przy okazji mechanizmów przetwarzania strumieni danych. Teraz skorzystamy z konsumentów i producentów dostępnych za pomocą skryptów dostarczanych w ramach instalacji Kafki.

12. Otwórz kolejny terminal SSH do węzła master naszego klastra. Możesz zmienić tło na zielony. To będzie nasz **producent**.



13. Za pomocą poniższego polecenia uruchom **producenta** „z konsoli”

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

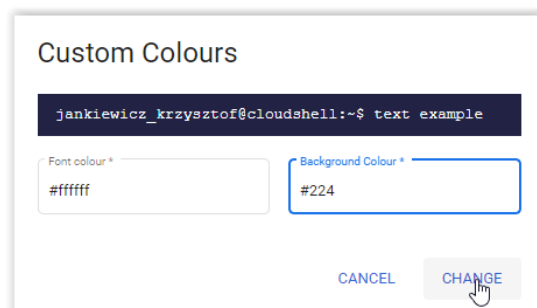
kafka-console-producer.sh \
  --broker-list ${CLUSTER_NAME}-w-0:9092 \
  --topic kafka-tt
```

14. Wpisz dwie linie “ważnych” wiadomości, które będą wysłane do brokera Kafka

- 1, Lorem Ipsum jest tekstem stosowanym jako
- 2, przykładowy wypełniacz w przemyśle poligraficznym.

```
jankiewicz_krzysztof@bigdata-bs-m:~$ kafka-console-producer.sh \
--broker-list ${CLUSTER_NAME}-w-0:9092 \
--topic kafka-tt
>1, Lorem Ipsum jest tekstem stosowanym jako
2, przykładowy wypełniacz w przemyśle poligraficznym.
```

15. Czas na konsumenta. Otwórz nowy terminal. Możesz zmienić tło na terminala na niebieskie. To będzie nasz **konsument**.



16. Uruchom naszego **konsumenta**. Oba komunikaty powinny zostać przez Ciebie odczytane.

```
CLUSTER_NAME=$(/usr/share/google/get_metadata_value attributes/dataproc-cluster-name)
kafka-console-consumer.sh --group my-consumer-group \
--bootstrap-server ${CLUSTER_NAME}-w-0:9092 \
--topic kafka-tt --from-beginning
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ kafka-console-consumer.sh --group my-consumer-group \
--bootstrap-server ${CLUSTER_NAME}-w-0:9092 \
--topic kafka-tt --from-beginning
1, Lorem Ipsum jest tekstem stosowanym jako
2, przykładowy wypełniacz w przemyśle poligraficznym.
```

17. Wprowadź kolejne trzy linie do **producenta**

3. Został po raz pierwszy użyty w XV w.
4. przez nieznanego drukarza do wypełnienia
5. tekstem próbnej książki.

18. Twój konsument powinien je już mieć.

```
2, przykładowy wypełniacz w przemyśle poligraficznym.
1, Lorem Ipsum jest tekstem stosowanym jako
4. przez nieznanego drukarza do wypełnienia
5. tekstem próbnej książki.
3. Został po raz pierwszy użyty w XV w.
```

- Czy nasz producent i konsument działają poprawnie?
- Co się dzieje z kolejnością wiadomości?
- Czy ona musi odpowiadać kolejności nadania wiadomości? Czy takie działanie jest przez Ciebie oczekiwane?
- Kiedy kolejność odbieranych wiadomości przez konsumenta byłaby zawsze taka sama jak kolejność ich nadawania przez producenta?

19. A teraz za pomocą kombinacji klawiszy Ctrl+C przerwij pracę **konsumenta**.

20. Wprowadź ostatnie cztery linie do **producenta**

```
6) Pięć wieków później zaczął być używany
7) przemysłu elektronicznym,
8) pozostając praktycznie niezmiennym.
9) Za: https://pl.lipsum.com/
```

Poznamy teraz kilka przydatnych poleceń, które nie były „sprzedane” na części wykładowej.

Wykorzystaj pierwszy, początkowy (**techniczny**) terminal

21. Skorzystaj z funkcjonalności narzędzia GetOffsetShell i pobierz offsety (pierwszy dostępny i ostatni) dla każdej partycji utworzonego tematu

```
kafka-run-class.sh kafka.tools.GetOffsetShell \
  --broker-list ${CLUSTER_NAME}-w-0:9092 \
  --topic kafka-tt --time -1
```

```
jankiewicz.krzysztof@bigdata-bs-m:~$ kafka-run-class.sh kafka.tools.GetOffsetShell \
  --broker-list ${CLUSTER_NAME}-w-0:9092 \
  --topic kafka-tt --time -1
kafka-tt:0:3
kafka-tt:1:6
kafka-tt:2:0
```

Z czego wynika dystrybucja wiadomości pomiędzy tymi partycjami? Czy obecna dystrybucja jest deterministyczna? Z czego ona wynika? Kiedy można byłoby się spodziewać deterministycznej dystrybucji wiadomości wysyłanych do poszczególnych partycji?

22. Poznaj także polecenie do uzyskiwania listy grup konsumentów zarejestrowanych w brokerze Kafki

```
/usr/lib/kafka/bin/kafka-consumer-groups.sh \
  --bootstrap-server ${CLUSTER_NAME}-w-0:9092 \
  --list
```

23. Uruchom w terminalu **konsumenta** poniższe polecenia, aby dowiedzieć się jak możesz pobrać tylko ostatnie wiadomości (tylko te, które jeszcze nie zostały przez Ciebie pobrane)

```
kafka-console-consumer.sh /?
```

24. Uzupełnij poniższe polecenie o parametry, które pozwolą Ci wydobyć wiadomości, które do tej pory nie były odebrane.

```
kafka-console-consumer.sh \
  --bootstrap-server ${CLUSTER_NAME}-w-0:9092 \
  --topic kafka-tt --. . .
```

25. (opcjonalnie) Przerwij za pomocą kombinacji klawiszy Ctrl+C pracę konsumenta. A teraz spróbuj pobrać ostatnie 2 wiadomości z partycji 0.

26. Przerwij za pomocą kombinacji klawiszy Ctrl+C pracę konsumenta i producenta.

Interfejsy programistyczne

(25 minut)

Czas na coś bardziej poważnego. Napiszemy program, który będzie działał jako producent Kafki. Jego zadaniem będzie odczytywanie zawartości plików w podanym katalogu i wprowadzanie kolejnych linii z odczytywanych plików jako wiadomości do wskazanego tematu Kafki

Program ma przyjmować następujące parametry:

- Katalog z plikami
- Czas przerwy w sekundach pomiędzy obsługą poszczególnych plików
- Nazwa tematu Kafki
- Liczba linii nagłówka w każdym z plików, które należy pominąć.
- Adres serwera Kafki

27. Biblioteki Kafki są bibliotekami Scali, dlatego chcąc programować w Scali musimy poznać wykorzystywaną w ramach naszej instalacji wersję Scali oraz wersję samej Kafki. Najprościej to zrobić podglądając biblioteki jakie są załączone w naszej instalacji. Przejdź w jednym z terminali do katalogu Kafki

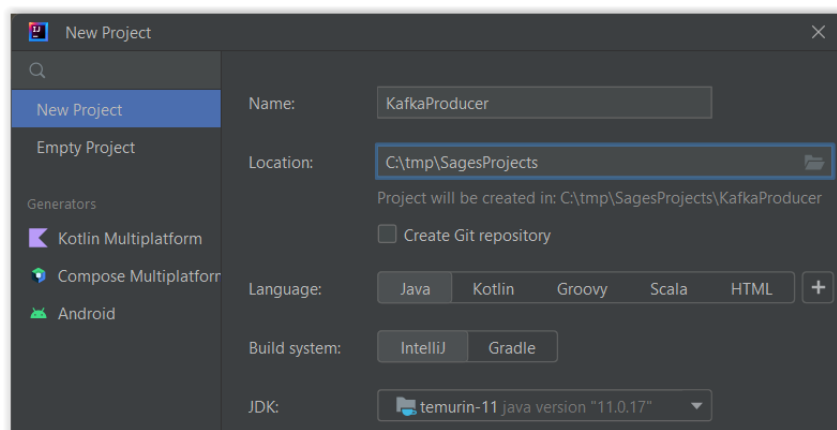
```
cd /usr/lib/kafka
```

28. Następnie wylistuj interesujące nas biblioteki. Zaprezentowany poniżej wynik świadczy o tym, że mamy biblioteki dla Scali w wersji 2.12 oraz wersję Kafki 3.1.0

```
ls libs/kafka*.jar
```

```
jankiewicz_krzysztof@bigdata-bs-m:/usr/lib/kafka$ ls libs/kafka*.jar
libs/kafka-clients-3.1.0.jar      libs/kafka-storage-api-3.1.0.jar
libs/kafka-log4j-appender-3.1.0.jar  libs/kafka-streams-3.1.0.jar
libs/kafka-metadata-3.1.0.jar    libs/kafka-streams-examples-3.1.0.jar
libs/kafka-raft-3.1.0.jar        libs/kafka-streams-scala_2.12-3.1.0.jar
libs/kafka-server-common-3.1.0.jar  libs/kafka-streams-test-utils-3.1.0.jar
libs/kafka-shell-3.1.0.jar       libs/kafka-tools-3.1.0.jar
libs/kafka-storage-3.1.0.jar     libs/kafka_2.12-3.1.0.jar
```

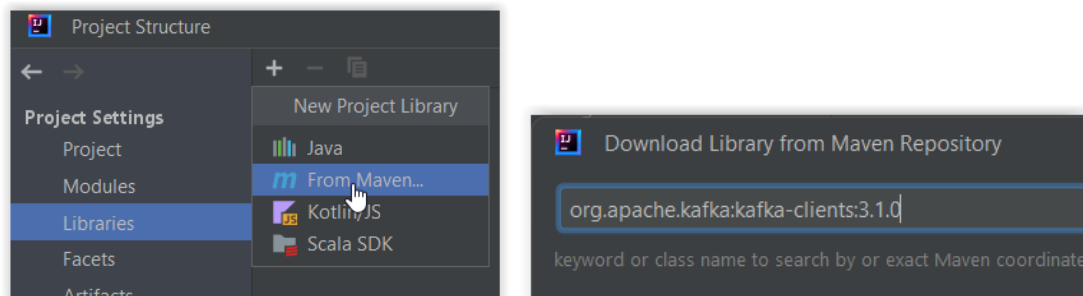
29. Programować będziemy w lokalnym środowisku *IntelliJ IDEA*. Ponadto, w związku z tym, że wsparcie społeczności (a także samych twórców Kafki) dla języka Java, stworzymy jednak projekt w Javie. Otwórz *IntelliJ IDEA*. Utwórz nowy projekt Javy o nazwie *KafkaProducer*. Zwróć uwagę na wersję JDK.



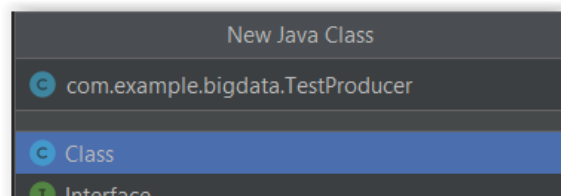
30. Korzystając z menu kontekstowego naszego projektu otwórz ustawienia modułu (F4), a następnie dodaj właściwe biblioteki klienta Kafki korzystając z repozytorium Mavena.

<https://mvnrepository.com/artifact/org.apache.kafka/kafka-clients>

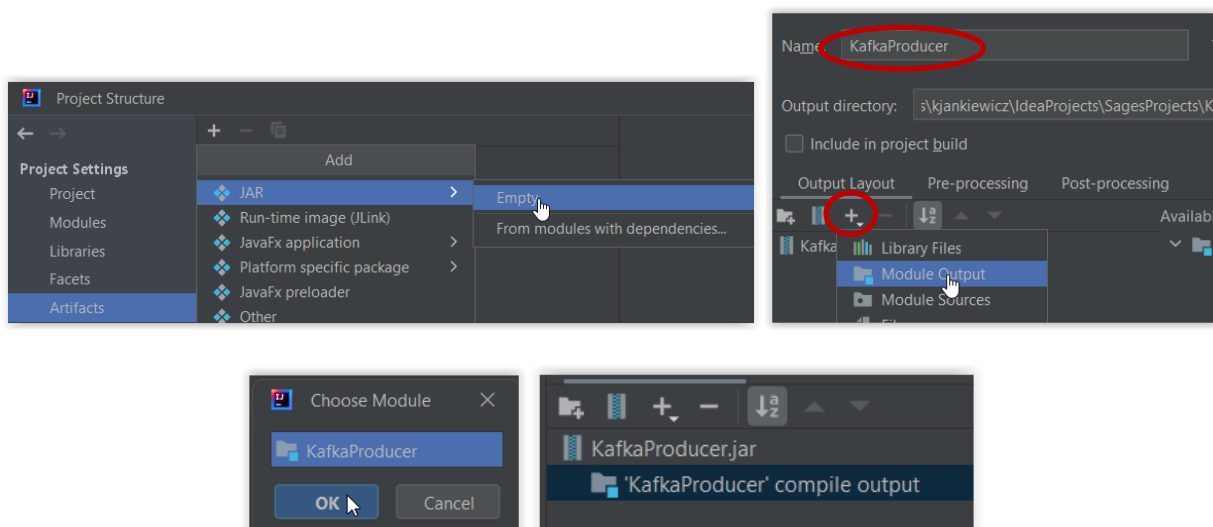
Dla naszego przykładu będą to: `org.apache.kafka:kafka-clients:3.1.0`



31. Utwórz klasę Javy o nazwie `TestProducer` w ramach pakietu `com.example.bigdata`.



32. Wprowadź do utworzonego pliku zawierającego definicję klasy `TestProducer` zawartość umieszczoną na następnej stronie.
33. Uzupełnij fragmenty kodu zgodnie z poleceniami w komentarzach. Każdy brakujący fragment został zastąpiony znakami `???`. Korzystaj w tym celu z materiałów wykładowych. Znajdziesz tam wiele przydatnych fragmentów kodu.
34. Gdy Twój program będzie już gotowy, w celu wygenerowania pliku jar (`KafkaProducer.jar`) utwórz stosowną definicję artefaktu




```

package com.example.bigdata;

import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;

import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Arrays;
import java.util.Properties;
import java.util.concurrent.TimeUnit;
import java.util.stream.Stream;

public class TestProducer {
    public static void main(String[] args) {
        if (args.length < 5) {
            System.out.println("Należy podać pięć parametrów: " +
                               "inputDir sleepTime topicName headerLength bootstrapServers");
            System.exit(0);
        }
        String inputDir = args[0];
        String sleepTime = args[1];
        String topicName = args[2];
        String headerLength = args[3];
        String bootstrapServers = args[4];

        Properties props = new Properties();
        props.put("bootstrap.servers", bootstrapServers);
        // wprowadź poniżej pozostałe parametry producenta Kafki
        // patrz materiały wykładowe lub dokumentacja
        ???

        // uzupełnij polecenie tworzące producenta Kafki
        KafkaProducer<String, String> producer = ???;

        // przeanalizuj poniższy kod aby dowiedzieć się jak on działa
        final File folder = new File(inputDir);
        File[] listOfFiles = folder.listFiles();
        String[] listOfPaths = Arrays.stream(listOfFiles).
            map(file -> file.getAbsolutePath()).toArray(String[]::new);
        Arrays.sort(listOfPaths);

        for (final String fileName : listOfPaths) {
            try (Stream<String> stream = Files.lines(Paths.get(fileName)).
                skip(Integer.parseInt(headerLength))) {

                // uzupełnij polecenie wysyłające komunikat do odpowiedniego
                // tematu Kafki. Do wskazania tematu użyj zmiennej topicName
                // Kluczem niech będzie wyrażenie String.valueOf(line.hashCode())

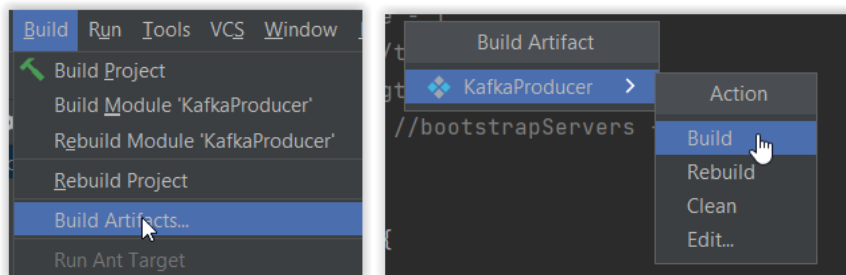
                stream.forEach(line -> ???);

                TimeUnit.SECONDS.sleep(Integer.parseInt(sleepTime));

            } catch (IOException | InterruptedException e) {
                e.printStackTrace();
            }
        }
        producer.close();
    }
}

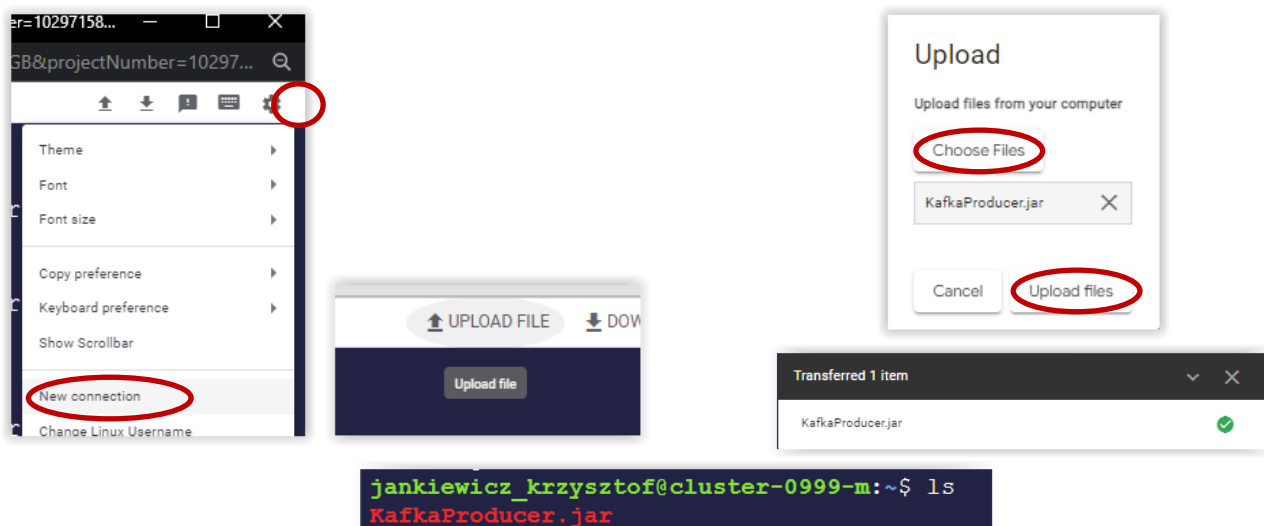
```

35. Korzystając ze stworzonej definicji artefaktu wygeneruj plik jar.



36. Tak utworzony plik wyślij do Twojego katalogu domowego w systemie plików na serwerze master.

Możesz skorzystać z funkcjonalności domyślnego terminala SSH (patrz poniżej). Otwórz w tym celu nowy terminal SSH.



37. Czas na sprawdzenie działania naszego producenta. W terminalu **producenta** przygotujemy się do nadawania. W katalogu domowym Twojego użytkownika utwórz w katalog kafka-input

```
cd ~
mkdir kafka-input
```

38. Utwórz plik, który będzie zawierał nadawaną zawartość.

```
nano kafka-input/input.txt
```

39. Wprowadź do pliku poniższą zawartość:

```
New 1, Lorem Ipsum jest tekstem stosowanym jako
New 2, przykładowy wypełniacz w przemyśle poligraficznym.
New 3. Został po raz pierwszy użyty w XV w.
New 4. przez nieznanego drukarza do wypełnienia
New 5. tekstem próbnej książki.
New 6) Pięć wieków później zaczął być używany
New 7) przemyśle elektronicznym,
New 8) pozostając praktycznie niezmienionym.
New 9) Za: https://pl.lipsum.com/
```

40. Zapisz zawartość pliku, a następnie wyjdź z edytora.

41. Uruchom napisanego przez Ciebie **producenta** z następującymi parametrami

```
CLUSTER_NAME=$(/usr/share/google/get_metadata_value attributes/dataproc-cluster-name)
java -cp /usr/lib/kafka/libs/*:KafkaProducer.jar \
    com.example.bigdata.TestProducer kafka-input 15 kafka-tt \
    0 ${CLUSTER_NAME}-w-0:9092
```

42. Uruchom w terminalu **konsumenta**

```
CLUSTER_NAME=$(/usr/share/google/get_metadata_value attributes/dataproc-cluster-name)
kafka-console-consumer.sh \
    --bootstrap-server ${CLUSTER_NAME}-w-0:9092 \
    --topic kafka-tt --from-beginning
```

```
jankiewicz_krzysztof@bigdata-bs-m:~$ kafka-console-consumer.sh \
    --bootstrap-server ${CLUSTER_NAME}-w-0:9092 \
    --topic kafka-tt --from-beginning
1, Lorem Ipsum jest tekstem stosowanym jako
2, przykładowy wypełniacz w przemyśle poligraficznym.
6) Pięć wieków później zaczął być używany
7) przemyśle elektronicznym,
8) pozostając praktycznie niezmiennym.
9) Za: https://pl.lipsum.com/
3. Został po raz pierwszy użyty w XV w.
4. przez nieznanego drukarza do wypełnienia
5. tekstem próbnej książki.
New 3. Został po raz pierwszy użyty w XV w.
New 4. przez nieznanego drukarza do wypełnienia
New 5. tekstem próbnej książki.
New 6) Pięć wieków później zaczął być używany
New 8) pozostając praktycznie niezmiennym.
New 1, Lorem Ipsum jest tekstem stosowanym jako
New 2, przykładowy wypełniacz w przemyśle poligraficznym.
New 7) przemyśle elektronicznym,
New 9) Za: https://pl.lipsum.com/
```

Nieźły bałagan. Ale czy naprawdę. Na jakim poziomie porządek został zachowany?

43. Zatrzymaj pracę konsumenta oraz producenta.

44. Zachowaj napisany przez Ciebie program (plik jar). Przyda nam się on w przyszłości.

45. Jeśli chcesz poznać coś jeszcze, potrafisz zdefiniować tunel do naszego środowiska i masz czas, zaglądaj do dodatku na następnej stronie.

Jeśli nie usuń klaster. Przez chwilę nie będzie on nam potrzebny.

Dodatek

Apache Kafka nie posiada graficznego interfejsu sieciowego, który pozwala monitorować jej funkcjonowanie. Istnieje jednak wiele narzędzi zewnętrznych, które można w tym celu wykorzystać.

Chyba najbardziej popularnymi są:

- UI for Apache Kafka - <https://github.com/provectus/kafka-ui>
- Confluent CC - <https://github.com/confluentinc>
- Konduktor - <https://github.com/konduktor>

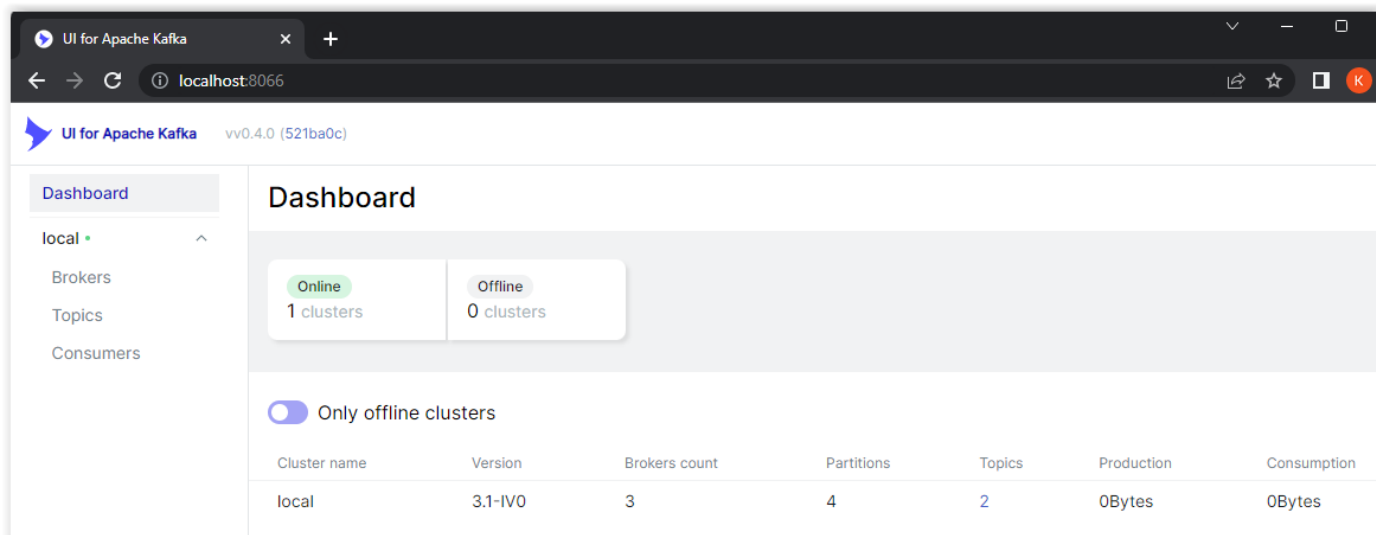
Wśród powyższych pierwsze narzędzie jest open-source i bezpłatne, dlatego z niego skorzystamy.

46. Przejdź do terminala SSH, a następnie uruchom kontener dockerowy z instancją tego narzędzia

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

docker run -p 8066:8080 \
-e KAFKA_CLUSTERS_0_NAME=local \
-e KAFKA_CLUSTERS_0_BOOTSTRAPSERVERS=${CLUSTER_NAME}-w-0:9092 \
-d provectuslabs/kafka-ui:latest
```

47. Utwórz tunel SSH do portu 8066, a następnie wejdź na stronę narzędzia UI for Apache Kafka

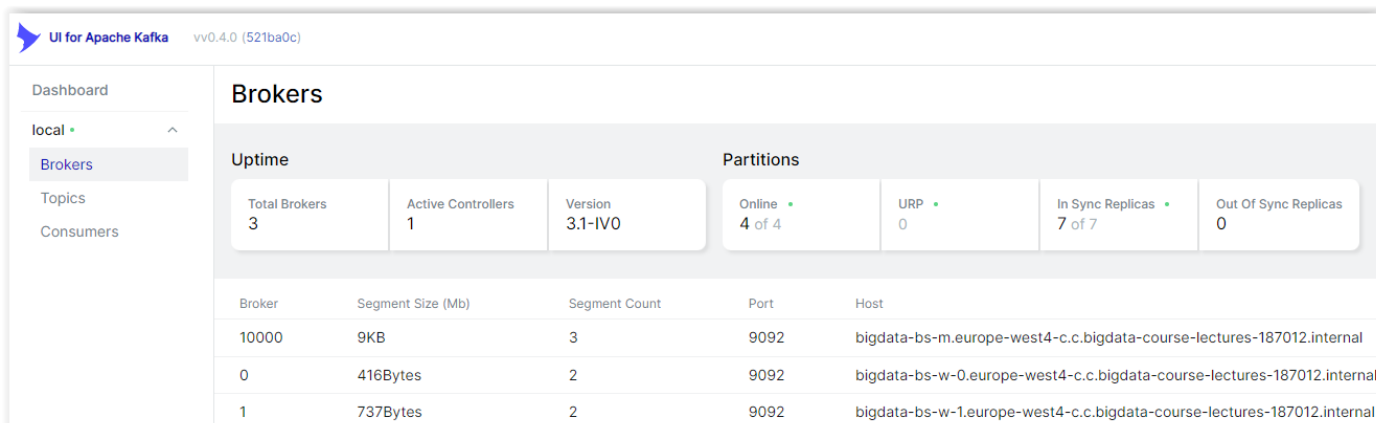


The screenshot shows the 'UI for Apache Kafka' dashboard at localhost:8066. The dashboard displays the following information:

- Online:** 1 clusters
- Offline:** 0 clusters
- Only offline clusters:** (toggle is off)
- Cluster Details Table:**

Cluster name	Version	Brokers count	Partitions	Topics	Production	Consumption
local	3.1-IV0	3	4	2	0Bytes	0Bytes

48. Zglądnij do zakładki z serwerami Kafki funkcjonującymi w ramach klastra.



The screenshot shows the 'Brokers' tab in the 'UI for Apache Kafka' dashboard. It displays the following information:

- Uptime:**
 - Total Brokers: 3
 - Active Controllers: 1
 - Version: 3.1-IV0
- Partitions:**
 - Online: 4 of 4
 - URP: 0
 - In Sync Replicas: 7 of 7
 - Out Of Sync Replicas: 0
- Broker Details Table:**

Broker	Segment Size (Mb)	Segment Count	Port	Host
10000	9KB	3	9092	bigdata-bs-m.europe-west4-c.c.bigdata-course-lectures-187012.internal
0	416Bytes	2	9092	bigdata-bs-w-0.europe-west4-c.c.bigdata-course-lectures-187012.internal
1	737Bytes	2	9092	bigdata-bs-w-1.europe-west4-c.c.bigdata-course-lectures-187012.internal

49. Przejdź do listy tematów

UI for Apache Kafka vv0.4.0 (521ba0c)

Dashboard

local

Brokers

Topics

Consumers

All Topics

+ Add a Topic

Search by Topic Name

Show Internal Topics

Topic Name	Total Partitions	Out of sync replicas	Replication Factor	Number of messages	Size
IN __consumer_offsets	50	0	1	2	480Bytes
<input type="checkbox"/> kafka-tt	3	0	2	19	2KB
<input type="checkbox"/> test	1	0	1	101	7KB

50. Przejdź do szczegółów tematu kafka-tt

UI for Apache Kafka vv0.4.0 (521ba0c)

Dashboard

local

Brokers

Topics

Consumers

Topics / kafka-tt

kafka-tt

Overview Messages Consumers Settings

Partitions	Replication Factor	URP	In Sync Replicas	Type	Segment Size	Segment Count
3	2	0	6 of 6	External	2KB	6

Clean Up Policy

DELETE

Message Count

19

Partition ID	Broker Leader	First Offset	Next Offset	Message Count
0	0	0	6	6
1	1	0	0	0
2	10000	0	13	13

51. Przejdź do wiadomości, zawartych w tym temacie

52. Wyślij do tego tematu jeszcze jedną wiadomość

53. Usuń wszystkie wiadomości z tego tematu.

54. Utwórz nowy temat

- Nazwa: koniec,
- Liczba partycji: 15
- Czas utrzymywania danych: 4 tygodnie

55. Usuń utworzony temat.

56. To kończy ostatecznie nasz warsztat. Usuń klaster. Przez chwilę nie będzie on nam potrzebny.