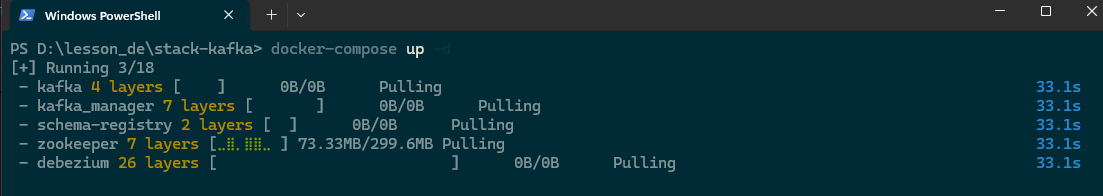
- Path to configure postgres wal

| path | /var/lib/postgresql/data/postgresql.conf |
| --- | --- |

- How on **WINDOWS** looks when we download for the first time



| **file** | docker-compose.yaml |
| --- | --- |
| version: "3.7"  services:  zookeeper:  image: confluentinc/cp-zookeeper:5.5.3  environment:  ZOOKEEPER\_CLIENT\_PORT: 2181  networks:  - belajar\_ruby  kafka:  image: confluentinc/cp-enterprise-kafka:5.5.3  depends\_on:  - zookeeper  ports:  - 29092:29092  environment:  KAFKA\_BROKER\_ID: 1  KAFKA\_OFFSETS\_TOPIC\_REPLICATION\_FACTOR: 1  KAFKA\_ZOOKEEPER\_CONNECT: zookeeper:2181  KAFKA\_ADVERTISED\_LISTENERS: PLAINTEXT://kafka:9092,PLAINTEXT\_HOST://localhost:29092  KAFKA\_LISTENER\_SECURITY\_PROTOCOL\_MAP: PLAINTEXT:PLAINTEXT,PLAINTEXT\_HOST:PLAINTEXT  networks:  - belajar\_ruby  debezium:  image: debezium/connect:2.7  environment:  BOOTSTRAP\_SERVERS: kafka:9092  GROUP\_ID: 1  CONFIG\_STORAGE\_TOPIC: connect\_configs  OFFSET\_STORAGE\_TOPIC: connect\_offsets  # KEY\_CONVERTER: io.confluent.connect.avro.AvroConverter  # VALUE\_CONVERTER: io.confluent.connect.avro.AvroConverter  CONNECT\_KEY\_CONVERTER\_SCHEMA\_REGISTRY\_URL: http://schema-registry:8081  CONNECT\_VALUE\_CONVERTER\_SCHEMA\_REGISTRY\_URL: http://schema-registry:8081  depends\_on: [kafka]  ports:  - 8083:8083  networks:  - belajar\_ruby  schema-registry:  image: confluentinc/cp-schema-registry:5.5.3  environment:  - SCHEMA\_REGISTRY\_KAFKASTORE\_CONNECTION\_URL=zookeeper:2181  - SCHEMA\_REGISTRY\_HOST\_NAME=schema-registry  - SCHEMA\_REGISTRY\_LISTENERS=http://schema-registry:8081,http://localhost:8081  ports:  - 8081:8081  depends\_on: [zookeeper, kafka]  networks:  - belajar\_ruby    kafka\_manager:  image: hlebalbau/kafka-manager:stable  restart: always  ports:  - "9000:9000"  depends\_on:  - zookeeper  - kafka  environment:  ZK\_HOSTS: "zookeeper:2181"  APPLICATION\_SECRET: "random-secret"  command: -Dpidfile.path=/dev/null  networks:  - belajar\_ruby  networks:  belajar\_ruby:  driver: bridge | |

Docker command to attach network

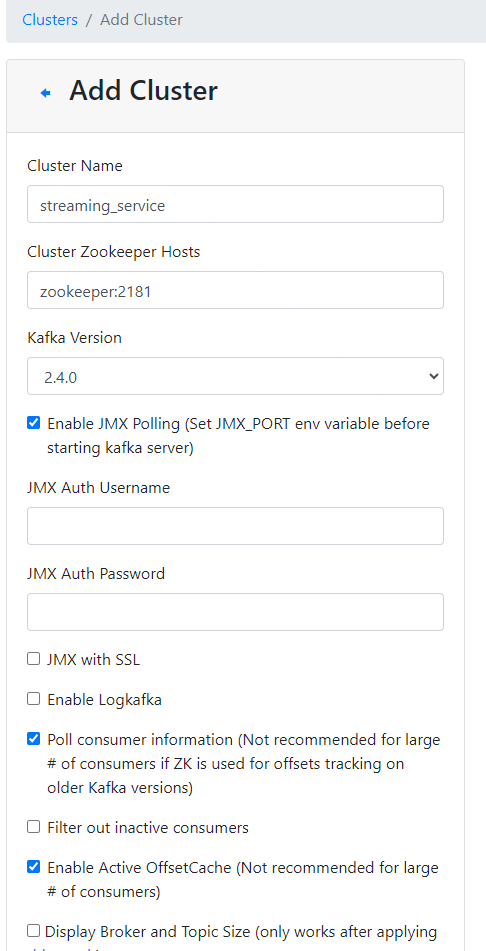
| –connect  docker network connect [network\_name] [container\_name]  –disconnect  docker network disconnect [network\_name] [container\_name] |
| --- |

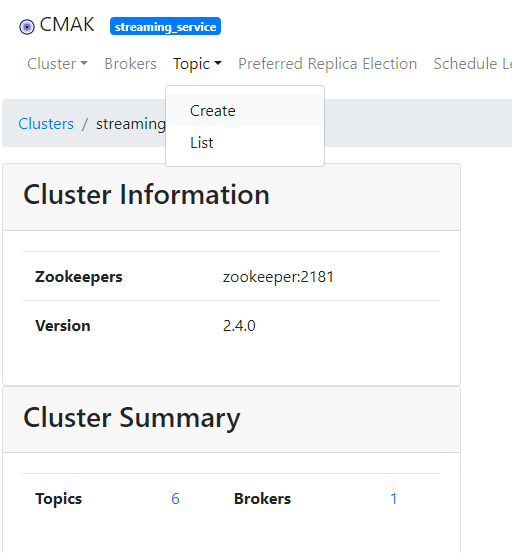
Some basic **KAFKA** useful command:

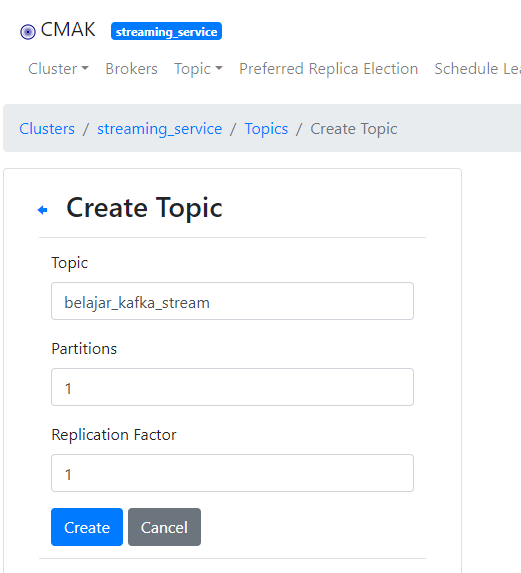
| --create topic  kafka-topics --create --topic test --bootstrap-server localhost:9092 --partitions 1 --replication-factor 1  --list kafka topics  kafka-topics --bootstrap-server=localhost:9092 --list  --start console proudcer and send messages  kafka-console-producer --topic test --bootstrap-server localhost:9092  --start console consumer and receive messages  kafka-console-consumer --topic test --from-beginning --bootstrap-server localhost:9092 |
| --- |

**STEP ON KAFKA MANAGER**

1. Access <https://localhost:9000>
2. Following default sample, and click save.



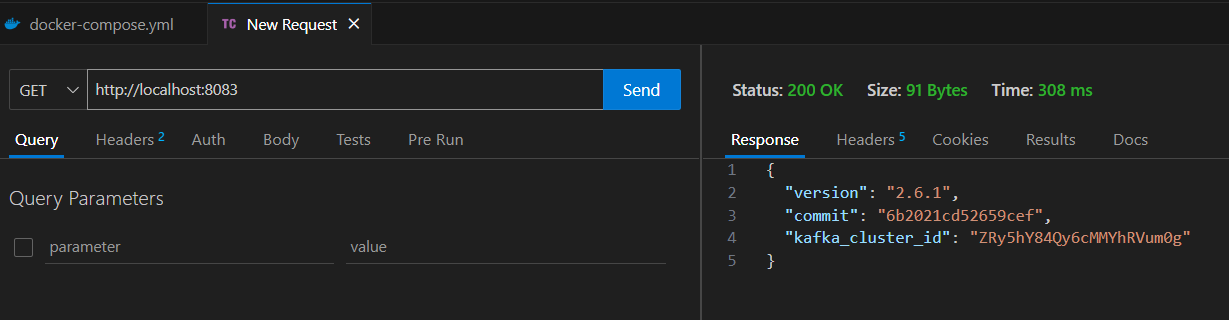
1. Create topic by the kafka cluster  
   
2. Create the topic and let it remain default, save.



1. Remain try the testing producer and consumer methods

**STEP ON DEBEZIUM CONNECTOR**

Point interact with REST Client at <http://localhost:8083>,



| **GET** LIST OF CONNECTORS | http://localhost:8083/connectors |
| --- | --- |
| **GET** DEBEZIUM INITIAL | http://localhost:8083 |
| **POST** CREATE CONNECTOR | http://localhost:8083/connectors |

| json body for CREATE CONNECTOR |
| --- |
| {  "name": "source-sales-data-connector-",  "config": {  "connector.class": "io.debezium.connector.postgresql.PostgresConnector",  "database.hostname": "belajar\_postgres",  "database.port": "5432",  "database.user": "postgres",  "database.password": "password",  "database.dbname": "postgres",  "plugin.name": "pgoutput",  "database.server.name": "source",  "key.converter.schemas.enable": "false",  "value.converter.schemas.enable": "false",  "transforms": "unwrap",  "transforms.unwrap.type": "io.debezium.transforms.ExtractNewRecordState",  "value.converter": "org.apache.kafka.connect.json.JsonConverter",  "key.converter": "org.apache.kafka.connect.json.JsonConverter",  "table.include.list": "public.sales\_data",  "slot.name" : "sales\_data\_slot",  "topic.prefix" : "sd"  }  } |

**CLICKHOUSE** Part

- deploy local

| docker run -d -p 18123:8123 -p 19000:9000 -e CLICKHOUSE\_PASSWORD=password --name belajar\_ch --ulimit nofile=262144:262144 clickhouse/clickhouse-server |
| --- |

- base table

| CREATE TABLE default.sales\_data  (  `date` DateTime,  `name` String,  `market\_area` String,  `number\_of\_sales` UInt32,  `pricing\_unit` Float64,  `insert\_time\_clickhouse` SimpleAggregateFunction(max, Datetime('Asia/Jakarta')) DEFAULT NOW()  )  ENGINE = ReplacingMergeTree  PARTITION BY name  ORDER BY (date, name, market\_area)  SETTINGS index\_granularity = 8192; |
| --- |

- kafka base table

| CREATE TABLE default.sales\_data\_queue  (  `date` DateTime,  `name` String,  `market\_area` String,  `number\_of\_sales` UInt32,  `pricing\_unit` Float64  )  ENGINE = Kafka  SETTINGS kafka\_broker\_list = 'stack-kafka-kafka-1:9092',  kafka\_topic\_list = 'sd.public.sales\_data',  kafka\_group\_name = 'test-1',  kafka\_format = 'JSONEachRow'; |
| --- |

- mv to base table

| CREATE MATERIALIZED VIEW default.sales\_data\_queue\_mv TO default.sales\_data  AS SELECT  date,  name,  market\_area,  number\_of\_sales,  pricing\_unit  FROM default.sales\_data\_queue; |
| --- |