Semantix

Kafka - Básico

Aula 4

Quem sou eu?

Eu sou Rodrigo Augusto Rebouças.

Engenheiro de dados da Semantix
Instrutor do Semantix Mentoring Academy

Você pode me encontrar em:
rodrigo.augusto@semantix.com.br

KSQL Datagen

Conceitos

Semantix

KSQL Datagen

- Ferramenta de CLI
- Gerar dados de teste
  - Para testar ambientes de desenvolvimento
- Testes
  - Diferentes tópicos
    - orders, users, pageviews
  - Diferentes formatos
    - avro, json, delimited
  - Controlar a produção de mensagens
    - Quantidade
    - Taxa/s
- Acessar por Cont

  $ docker exec --it ksql-datagen bash

  $ ksql-datagen <argumentos>
- Ex.:

  $ ksql-datagen help

KSQL Datagen - Argumentos

- Argumentos - Default

```
root@ksql-datagen:/# ksql-datagen help
usage: DataGen
[help]
[bootstrap-server=<kafka bootstrap server(s)> (defaults to localhost:9092)]
[quickstart=<quickstart preset> (case-insensitive; one of 'orders', 'users', or 'pageviews')]
schema=<avro schema file>
[schemaRegistryUrl=<url for Confluent Schema Registry> (defaults to http://localhost:8081)]
key-format=<message key format> (case-insensitive; one of 'avro', 'json', 'kafka' or 'delimited')
value-format=<message value format> (case-insensitive; one of 'avro', 'json' or 'delimited')
topic=<kafka topic name>
key=<name of key column>
[iterations=<number of rows> (if no value is specified, datagen will produce indefinitely)]
[propertiesFile=<file specifying Kafka client properties>]
[nThreads=<number of producer threads to start>]
[msgRate=<rate to produce in msgs/second>]
[printRows=<true|false>]
```

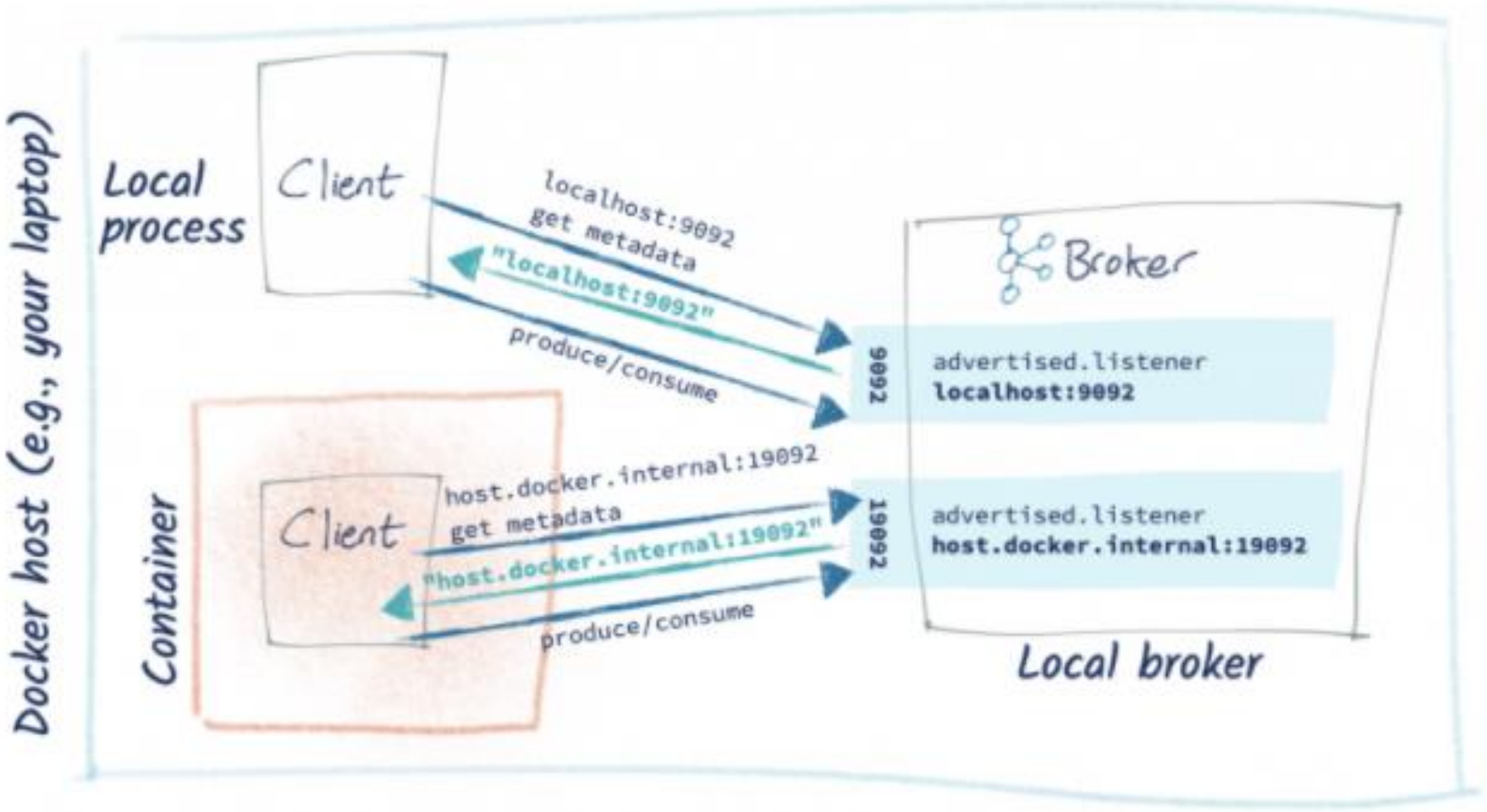
KSQL Datagen

Propriedades com containers

Semantix

Conexão KSQL Datagen

- https://www.confluent.io/blog/kafka-client-cannot-connect-to-broker-on-aws-on-docker-etc/



Semantix

KSQL Datagen - Argumentos

- Argumentos – Cluster em Docker
  - $ ksql-datagen \
    bootstrap-server=broker:29092 \
    quickstart= <orders, users, pageviews> \
    schema=<ArquivoAvro> \
    schemaRegistryUrl=schema-registry:8081 \
    key-format=<avro, json, Kafka ou delimited> \
    value-format=<avro, json ou delimited> \
    topic=<nomeTopico> \
    key=<campoChave> \
    iterations=<númeroLinhas> \
    msgRate=<TaxaMsg/segundo>

- cat docker-compose.yml

```
...
broker:
  ...
  environment:
    kafka_listener_security_protocol_map:
        plaintext:plaintext,
        plaintext_host:plaintext
    kafka_advertised_listeners:
        plaintext://broker:29092,
        plaintext_host://localhost:9092
    ...
```

KSQL Datagen para Stream

Semantix

Exemplo Datagen

- Criação de dados no tópico orders_topic

$ ksql-datagen bootstrap-server=broker:29092 schemaRegistryUrl=schema-registry:8081 quickstart=orders topic=orders_topic

INFO AvroDataConfig values:

schemas.cache.config = 1

enhanced.avro.schema.support = false

connect.meta.data = true

0 --> ([1515722137194 | 0 | 'Item_975' | 3.492080368535175 | Struct{city=City_53,state=State_56,zipcode=20946}]) ts:1566309102570

1 --> ([1504020399494 | 1 | 'Item_696' | 1.3494102846621505 | Struct{city=City_13,state=State_42,zipcode=55583}]) ts:1566309102926

2 --> ([1490789383235 | 2 | 'Item_211' | 0.6351559561132624 | Struct{city=City_29,state=State_53,zipcode=37622}]) ts:1566309103186

3 ...

Exemplo Datagen

- Visualizar dados no tópico orders_topic

ksql> print “orders_topic”

{"ROWTIME":1566309935674,"ROWKEY":"3274","ordertime":1512663430008,"orderid":3274,"itemid":"Item_264","orderunits":4.014151453785212,"address":{"city":"City_95","state":"State_49","zipcode":61704}}

{"ROWTIME":1566309935974,"ROWKEY":"3275","ordertime":1516186024588,"orderid":3275,"itemid":"Item_715","orderunits":7.7582401985317375,"address":{"city":"City_95","state":"State_74","zipcode":43747}}

{"ROWTIME":1566309936214,"ROWKEY":"3276","ordertime":1491636506736,"orderid":3276,"itemid":"Item_871","orderunits":8.818365185144284,"address":{"city":"City_98","state":"State_52","zipcode":67297}}

...

Exemplo Datagen

- Criação de Stream

CREATE STREAM orders_filtrada (orderid INT, orderunits DOUBLE, address STRUCT<city VARCHAR, zipcode INT>, ordertime VARCHAR) WITH (KAFKA_TOPIC='orders_topic', VALUE_FORMAT='JSON’);

- Visualização de dados Steam

ksql> select * from orders_filtrada;

156631046760 4 | 5434 | 5434 | 1.895561873349924 4 | {CITY=City_83, ZIPCODE=61609} | 1509277499165

156631046793 7 | 5435 | 5435 | 0.5909432198422736 | {CITY=City_66, ZIPCODE=11024} | 1501173885894

156631046835 9 | 5436 | 5436 | 1.748748185280725 8 | {CITY=City_13, ZIPCODE=41065} | 1491958709551

...

Exercícios Ksql Datagen

1. Criar o tópico users com os dados do ksql-datagen
   - quickstart=users
   - topic=users
2. Visualizar os dados do tópico no Ksql
3. Criar o stream users_raw com os dados do tópico users com as seguintes propriedades
   - kafka_topic='users’
   - value_format='JSON’
   - key = 'userid’
   - timestamp='viewtime’
4. Visualizar a estrutura da Stream users_raw
5. Visualizar os dados da Stream users_raw
6. Repetir todo o proceso para o tópico pageviews
   - ksql-datagen quickstart=pageviews topic= pageviews

Schema Registry

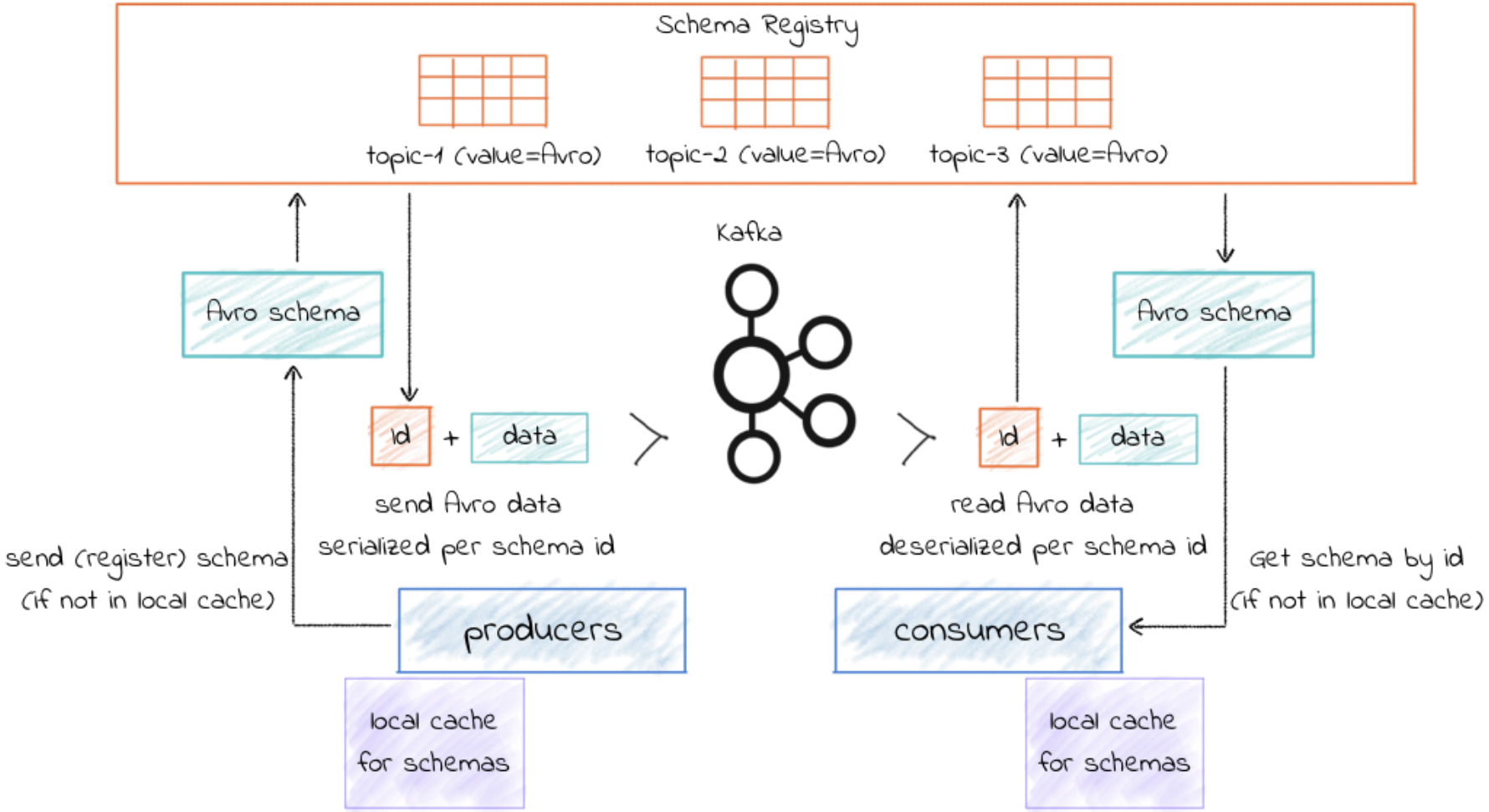
Semantix

Schema Registry

- Camada de armazenamento distribuído para esquema avro
- Mecanismo de armazenamento subjacente do Kafka
  - Atribui um id exclusivo para cada esquema registrado
- Armazenamento e recuperação do esquema para produtores e consumidores
- Diminuir a payload dos dados enviados para o Kafka

Schema Registry

- Armazenar e recuperar esquemas Avro



Schema Registry - Operações

- Operações são feitas através de API REST
- Operações com os esquemas
  - Adicionar
  - Recuperar
  - Atualizar
  - Deletar

Formato Avro

Semantix

Formato Avro

- Estrutura de serialização de dados
  - Escrito em Json
  - Formato binário compacto

- Vantagens
  - Dados
    - Digitados
    - Comprimido automaticamente
      - Usar avro tools para ler os dados
  - Documentação é embarcada no esquema
  - Suportado pelo Hadoop e diversas linguagens
  - Evolução do esquema ao longo do tempo de forma segura

Avro Estrutura

- Armazena o esquema através de JSON
- Campos padrões:
  - Name: Nome do esquema
  - Type: Tipo do esquema
  - Namespace: Equivalente ao package em java
  - Doc: Documentação do esquema
  - Aliasses: Outros nomes para o esquema
  - Fields
    - Name: Nome do campo
    - Doc: Documentação do campo
    - Type: Tipo do campo
    - Default: Valor padão para o campo

Avro Exemplo

Campos: Id, nome, status

Arquivo: nomes.avsc

```
{
        “type”: “record”,
        “Name”: “nomes”,
        “Namespace”: “example.avro”
        “Doc”: “Exemplo de um esquema”
        “Fields”: [
                {“name”: “id”, “type”: “int”},
                {“name”: “nome”, “type”: “string”},
                {“name”: “status”, “type”: “boolean”, “default”: true}
        ]
}
```

Semantix

Avro Console Consumer

Semantix

Avro Console Consumer

- Avro Console Consumer permitir o consumo de dados do Kafka
  - Não mostra todos os dados se usar kafka-console-consumer
- Acessar o container do schema-registry

  $ docker exec –it schema-registry bash
- Ex. Comando

  sudo kafka-avro-console-consumer

  --topic test-avro

  --bootstrap-server broker:29092

  --property schema.registry.url=http://localhost:8081

  --from-beginning

Avro Console Producer

Semantix

Avro Console Producer

- Avro Console Producer permitir o envio rápido de dados para o Kafka manualmente
  - Especificando o esquema no argumento
- Ex. Comando:

$ kafka-avro-console-producer

--broker-list broker:29092

--topic test-avro

--property schema.registry.url=http://localhost:8081

--property value.schema='{"type":"record","name":"myrecord","fields":[{"name":"id","type":“int"}, {"name":"nome","type":"string"}]}’

{"id":1,"nome":"Rodrigo"}

{"id":2,"nome":"Augusto"}

Evoluir Esquema

- Evoluir o esquema
  - Necessário colocar o campo default

$ sudo kafka-avro-console-producer --broker-list localhost:9092 --topic test-avro --property schema.registry.url=http://localhost:8081 --property value.schema='{"type":"record","name":"myrecord", "fields":[{"name":"id","type":"int"}, {"name":"nome","type":"string**"},{"name":"cidade","type":"string", "default":"null"}**]}'

{"id":3,"nome":"Ana","cidade":"sp"}

{"id":4,"nome":"Marcos","cidade":"sp"}

Stream de Avro

Semantix

Criar Stream para formato Avro

- O esquema já está no tópico Kafka
  - Schema Registry
  - Avro
    - Dados
    - Esquema
- Precisa apenas configurar as propriedades do Stream
- Ex. comando

  Ksql> create stream str_test-avro with (Kafka_topic='test-avro', value_format='avro’);

Exercício Schema Registry

1. Visualizar os dados do tópico users;
2. Criar o tópico users-avro
   a) Usar o kafka-avro-console-producer para enviar 1 mensagem
   b) Usar o kafka-avro-console-consumer para consumir a mensagem
   c) Visualizar o esquema no Control Center
3. Visualizar os dados do users-avro no KSQL
4. Criar um stream users-avro1 para o tópico users-avro
5. Visualizar os dados do stream users-avro1
6. Usar o kafka-avro-console-producer para adicionar um novo campo chamado “unit” com valor padrão “1”
7. Usar o kafka-avro-console-consumer para consumir as mensagens
8. Comparar os esquemas do users-avro no Control Center
9. Visualizar os dados no stream do tópico users-avro

Semantix

Obrigado!

Alguma pergunta?

Você pode me encontrar em:

rodrigo.augusto@semantix.com.br

GET SMARTER