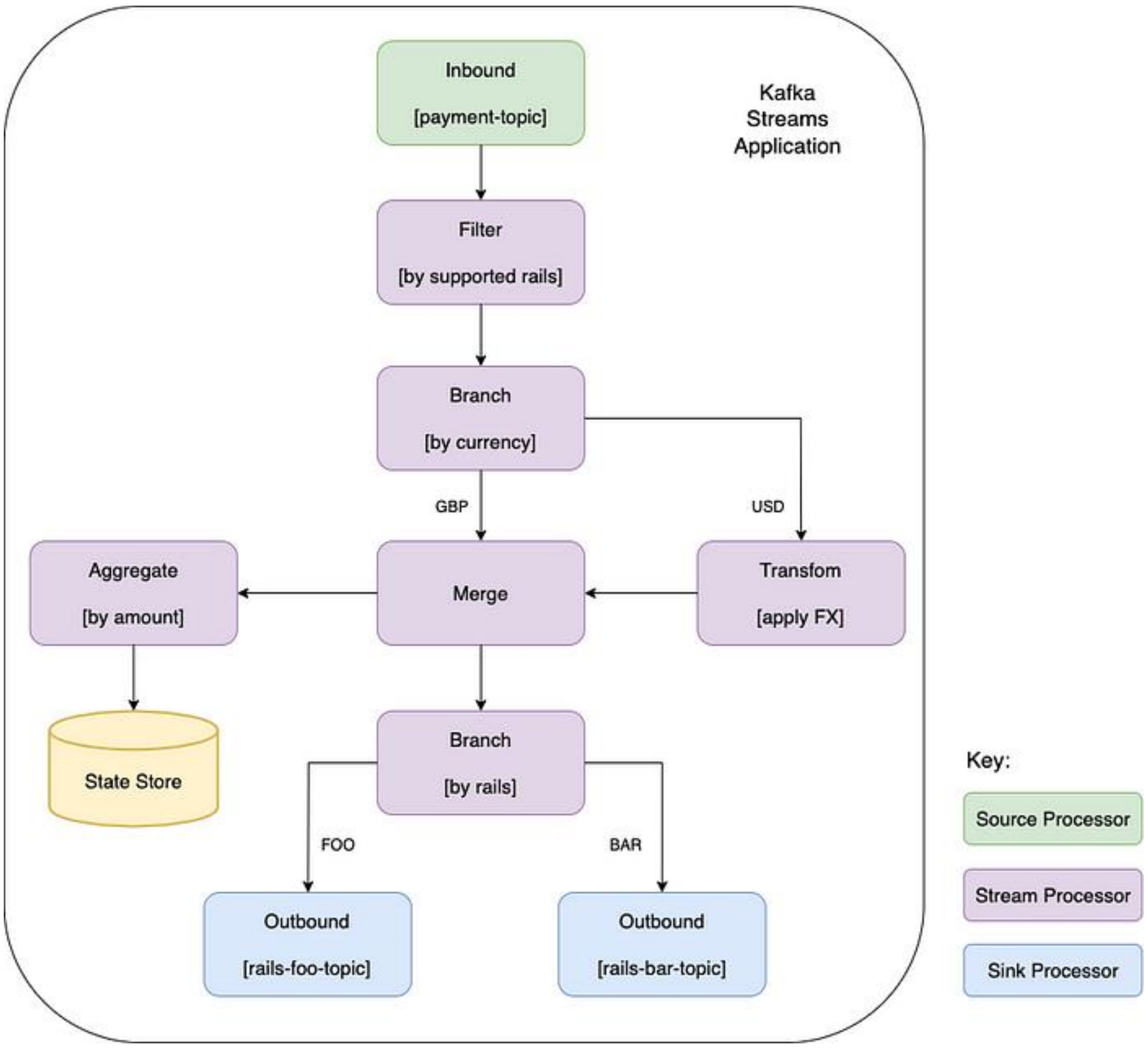
Topology

The Kafka Streams topology is as follows :



Spring Boot Application

Une application Spring Boot simule **la réception d'événements de paiement** de Kafka et traite ces paiements, y compris la **conversion de devises** à l'aide de **processeurs** en utilisant **KStream sans état (stateless KStream processors)**. L'application **suit les soldes des comptes** en agrégeant les montants des paiements à l'aide d'un **processeur KTable avec état (stateful KTable processor)**, en utilisant **RocksDB** comme **magasin d'état (State store)**. Les paiements sont ensuite émis vers le **sujet des rails sortants concerné (outbound rails topic)**. Les sujets relatifs aux rails (The rails topics) concernent les rails bancaires qui récupéreraient ensuite les paiements pour effectuer les transferts d'argent réels.

Application.yml

```
spring:
  application:
    name: kafka-streams-demo

kafka:
  bootstrap-servers: localhost:9092

server:
  port: 9001

kafkastreamsdemo:
  id: demo
  paymentInboundTopic: "payment-topic"
  railsFooOutboundTopic: "rails-foo-topic"
  railsBarOutboundTopic: "rails-bar-topic"
```

Properties

```
package demo.kafka.streams.properties;

import javax.validation.constraints.NotNull;

import lombok.Getter;
import lombok.Setter;
import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;
import org.springframework.validation.annotation.Validated;

@Configuration
@ConfigurationProperties("kafkastreamsdemo")
@Getter
@Setter
@Validated
public class KafkaStreamsDemoProperties {
  @NotNull private String id;
  @NotNull private String paymentInboundTopic;
  @NotNull private String railsFooOutboundTopic;
  @NotNull private String railsBarOutboundTopic;
}
```

Mapper

```
package demo.kafka.streams.mapper;

import com.fasterxml.jackson.databind.DeserializationFeature;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.databind.SerializationFeature;
```

```
public class JsonMapper {

  private static final ObjectMapper objectMapper = new ObjectMapper();

  static {
    objectMapper.configure(SerializationFeature.FAIL_ON_EMPTY_BEANS, false);
    objectMapper.configure(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES,
false);
    objectMapper.configure(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS, false);
    objectMapper.configure(SerializationFeature.WRITE_DURATIONS_AS_TIMESTAMPS,
false);
    objectMapper.findAndRegisterModules();
  }

  /**
   * Map the given JSON String to the required class type.
   */
  public static <T> T readFromJson(String json, Class<T> clazz) throws MappingException {
    try {
      return objectMapper.readValue(json, clazz);
    } catch (Exception e) {
      throw new MappingException(e);
    }
  }

  /**
   * Map the given Object to a JSON String.
   */
  public static String writeToJson(Object obj) throws MappingException {
    try {
      return objectMapper.writeValueAsString(obj);
    } catch (Exception e) {
      throw new MappingException(e);
    }
  }
}
package demo.kafka.streams.mapper;

public class MappingException extends RuntimeException {

   public MappingException(Throwable t) {
      super(t);
   }
}
```

Serdes

Dans Kafka, **« Serdes » est une abréviation de « Serializer » et « Deserializer »**. Les Serdes sont utilisés pour **sérialiser des objets en tableaux d'octets** afin de produire des enregistrements dans des sujets Kafka et **désérialiser des tableaux d'octets en objets** pour consommer des enregistrements de sujets Kafka.

Kafka utilise des tableaux d'octets comme format de stockage et de transmission des données. Lorsque vous produisez un message dans un sujet Kafka (topic), vous devez **convertir vos données en octets**, et lorsque vous consommez un message, vous devez **reconvertir ces octets dans le format de données souhaité**. Serdes facilite ce processus de conversion.

Les Serdes sont des composants essentiels des applications Kafka Streams, qui traitent les données en temps réel. Ils sont utilisés pour **sérialiser et désérialiser les données lors de la lecture et de l'écriture dans des sujets Kafka**. Kafka Streams fournit des Serdes intégrés pour les types de données courants tels que les chaînes, les entiers et JSON, mais vous pouvez également définir des **Serdes personnalisés** pour vos types de données spécifiques.

Dans cet exemple, Serdes.String() est utilisé pour spécifier que la clé et la valeur du flux Kafka sont des chaînes. Kafka Streams gère automatiquement la sérialisation et la désérialisation en fonction des Serdes spécifiés lors de la lecture et de l'écriture dans des sujets Kafka.

```
import org.apache.kafka.common.serialization.Serdes;

import org.apache.kafka.streams.StreamsBuilder;

import org.apache.kafka.streams.kstream.KStream;

import org.apache.kafka.streams.kstream.Produced;


public class MyKafkaStreamsApp {

  public static void main(String[] args) {

    StreamsBuilder builder = new StreamsBuilder();


    // Define a stream processing topology

    KStream<String, String> inputStream = builder.stream("input-topic");

    KStream<String, String> processedStream = inputStream.mapValues(value ->
value.toUpperCase());


    // Serialize and deserialize using Serdes
```

```java
        processedStream.to("output-topic", Produced.with(Serdes.String(), Serdes.String()));


        // Build and start the Kafka Streams application

        // Kafka Streams API handles serialization and deserialization using specified Serdes

    }

}
```

```java
package demo.kafka.streams.serdes;

import java.nio.charset.StandardCharsets;
import java.util.Map;

import demo.kafka.streams.mapper.JsonMapper;
import org.apache.kafka.common.errors.SerializationException;
import org.apache.kafka.common.serialization.Serializer;

public class JsonSerializer<T> implements Serializer<T> {

    public JsonSerializer() {
    }

    @Override
    public void configure(Map<String, ?> props, boolean isKey) {
    }

    @Override
    public byte[] serialize(String topic, T data) {
        if (data == null)
            return null;

        try {
            return JsonMapper.writeToJson(data).getBytes(StandardCharsets.UTF_8);
        } catch (Exception e) {
            throw new SerializationException("Error serializing JSON message", e);
        }
    }

    @Override
    public void close() {
    }
}
```

```java
package demo.kafka.streams.serdes;
```

```
import java.nio.charset.StandardCharsets;
import java.util.Map;

import demo.kafka.streams.mapper.JsonMapper;
import org.apache.kafka.common.errors.SerializationException;
import org.apache.kafka.common.serialization.Deserializer;

public class JsonDeserializer<T> implements Deserializer<T> {

    private Class<T> destinationClass;

    public JsonDeserializer(Class<T> destinationClass) {
        this.destinationClass = destinationClass;
    }

    @Override
    public void configure(Map<String, ?> props, boolean isKey) {
    }

    @Override
    public T deserialize(String topic, byte[] bytes) {
        if (bytes == null)
            return null;

        try {
            return JsonMapper.readFromJson(new String(bytes, StandardCharsets.UTF_8),
destinationClass);
        } catch (Exception e) {
            throw new SerializationException("Error deserializing message", e);
        }
    }

    @Override
    public void close() {
    }
}
package demo.kafka.streams.serdes;

import demo.kafka.streams.event.PaymentEvent;
import org.apache.kafka.common.serialization.Serde;
import org.apache.kafka.common.serialization.Serdes;

/**
 * Requires the WrapperSerdes to allow this to be added as the default serdes config in the
KafkaStreams configuration.
 */
public final class PaymentSerdes extends Serdes.WrapperSerde<PaymentEvent> {
   public PaymentSerdes() {
      super(new JsonSerializer<>(), new JsonDeserializer<>(PaymentEvent.class));
   }

   public static Serde<PaymentEvent> serdes() {
      JsonSerializer<PaymentEvent> serializer = new JsonSerializer<>();
      JsonDeserializer<PaymentEvent> deserializer = new
JsonDeserializer<>(PaymentEvent.class);
      return Serdes.serdeFrom(serializer, deserializer);
   }
}
```

Processor : PaymentEvent

```
package demo.kafka.streams.event;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;


@Builder
@Data // for getters and setters
@NoArgsConstructor // for default constructor
@AllArgsConstructor  // for constructor with arguments
public class PaymentEvent {

   private String paymentId; // The payment unique id

   private Long amount; // The amount to send

   private String currency; // The devise

   private String toAccount; // The name of the account sending the payment

   private String fromAccount;  // The name of the account receiving the payment

   private String rails; // A payment rail is a payment platform or payment network that
transfers money from a payer to a payee.
}


package demo.kafka.streams.processor;
```

```
public enum Currency {
   GBP, // GBP is a code which means: pound sterling, the currency of the United Kingdom
(Great Britain Pound
   USD // le dollar américain. Il est composé du code de pays (US) suivi de la lettre D pour «
dollar ».
}

// The outbound bank rails
public enum Rails {
   BANK_RAILS_FOO,
   BANK_RAILS_BAR,
   BANK_RAILS_XXX;
}
```

- Dans le contexte d'Apache Kafka, une « topologie » fait généralement référence à la **structure ou à la disposition des nœuds de traitement Kafka Streams** au sein d'une application Kafka. **Kafka Streams est une bibliothèque Java permettant de créer des applications et des microservices en temps réel** qui traitent et analysent les données stockées dans les sujets Kafka.

- Une application Kafka Streams peut être constituée de plusieurs nœuds de traitement connectés dans diverses configurations pour effectuer des tâches de traitement de données telles **que le filtrage, l'agrégation, la transformation et la jointure de flux de données**. La disposition de ces nœuds de traitement et les flux de données entre eux constituent la topologie de l'application.

- Une topologie dans Kafka Streams peut être visualisée sous la forme d'un **graphe acyclique dirigé (DAG)** où les **nœuds représentent les opérations de traitement (par exemple, mapper, filtrer, joindre)** et **les bords représentent le flux de données entre ces opérations**. Chaque nœud correspond généralement à une étape de traitement dans l'application, et les données sont traitées en continu au fur et à mesure qu'elles circulent dans la topologie.

- Kafka Streams fournit une API pour définir et exécuter de telles topologies, facilitant ainsi la création de pipelines de traitement de données complexes en utilisant Kafka comme système de stockage de données et de messagerie sous-jacent. Les topologies peuvent être simples ou complexes, selon les exigences de l'application et les types d'opérations de traitement de données impliquées.

PaymentTopology

```
package demo.kafka.streams.processor;

import java.util.Arrays;
import java.util.List;

import demo.kafka.streams.event.PaymentEvent;
import demo.kafka.streams.properties.KafkaStreamsDemoProperties;
import demo.kafka.streams.serdes.PaymentSerdes;
```

```java
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.apache.kafka.common.serialization.Serde;
import org.apache.kafka.common.serialization.Serdes;
import org.apache.kafka.streams.KeyValue;
import org.apache.kafka.streams.StreamsBuilder;
import org.apache.kafka.streams.kstream.Aggregator;
import org.apache.kafka.streams.kstream.Consumed;
import org.apache.kafka.streams.kstream.Grouped;
import org.apache.kafka.streams.kstream.Initializer;
import org.apache.kafka.streams.kstream.KStream;
import org.apache.kafka.streams.kstream.Materialized;
import org.apache.kafka.streams.kstream.Produced;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component
@Slf4j
@RequiredArgsConstructor
public class PaymentTopology {

   @Autowired
   private final KafkaStreamsDemoProperties properties;

   private static List SUPPORTED_RAILS = Arrays.asList(Rails.BANK_RAILS_FOO.name(),
Rails.BANK_RAILS_BAR.name());

   private static final Serde<String> STRING_SERDE = Serdes.String();
   private static final Serde<Long> LONG_SERDE = Serdes.Long();

   @Autowired
   public void buildPipeline(StreamsBuilder streamsBuilder) {

      KStream<String, PaymentEvent> messageStream = streamsBuilder
         .stream(properties.getPaymentInboundTopic(), Consumed.with(STRING_SERDE,
PaymentSerdes.serdes()))
         .peek((key, payment) -> log.info("Payment event received with key=" + key + ",
payment=" + payment))

         // Filter out unsupported bank rails.
         .filter((key, payment) -> SUPPORTED_RAILS.contains(payment.getRails()))
         .peek((key, value) -> log.info("Filtered payment event received with key=" + key + ",
value=" + value));

      // Branch based on currency in order to perform any FX.
      KStream<String, PaymentEvent>[] currenciesBranches = messageStream.branch(
         (key, payment) -> payment.getCurrency().equals(Currency.GBP.name()),
            (key, payment) -> payment.getCurrency().equals(Currency.USD.name())
        );
        KStream<String, PaymentEvent> fxStream = currenciesBranches[1].mapValues(
            // Use mapValues() as we are transforming the payment, but not changing the key.
            (payment) -> {
                // Perform FX conversion.
                double usdToGbpRate = 0.8;
                PaymentEvent transformedPayment = PaymentEvent.builder()
                        .paymentId(payment.getPaymentId())
                        .amount(Math.round(payment.getAmount() * usdToGbpRate))
                        .currency(Currency.GBP.name())
                        .fromAccount(payment.getFromAccount())
                        .toAccount(payment.getToAccount())
                        .rails(payment.getRails())
                        .build();
                return transformedPayment;
            });

        // Merge the payment streams back together.
        KStream<String, PaymentEvent> mergedStreams =
currenciesBranches[0].merge(fxStream)
            .peek((key, value) -> log.info("Merged payment event received with key=" + key + ",
value=" + value));

        // Create the KTable stateful store to track account balances.
        mergedStreams
            .map((key, payment) -> new KeyValue<>(payment.getFromAccount(),
payment.getAmount()))
            .groupByKey(Grouped.with(STRING_SERDE, LONG_SERDE))
            .aggregate(new Initializer<Long>() {
                @Override
                public Long apply() {
                    return 0L;
                }
            }, new Aggregator<String, Long, Long>() {
                @Override
                public Long apply(final String key, final Long value, final Long aggregate) {
                    return aggregate + value;
                }
            }, Materialized.with(STRING_SERDE, LONG_SERDE).as("balance"));

        // Branch based on bank rails for outbound publish.
        KStream<String, PaymentEvent>[] railsBranches = mergedStreams.branch(
            (key, payment) -> payment.getRails().equals(Rails.BANK_RAILS_FOO.name()),
            (key, payment) -> payment.getRails().equals(Rails.BANK_RAILS_BAR.name()));

        // Publish outbound events.
        railsBranches[0].to(properties.getRailsFooOutboundTopic(),
Produced.with(STRING_SERDE, PaymentSerdes.serdes()));
        railsBranches[1].to(properties.getRailsBarOutboundTopic(),
Produced.with(STRING_SERDE, PaymentSerdes.serdes()));
    }
}
```

Controller

```
package demo.kafka.streams.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.kafka.config.StreamsBuilderFactoryBean;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/v1/kafka-streams")
public class TopologyController {

    @Autowired
    private StreamsBuilderFactoryBean factoryBean;

    /**
     * Endpoint providing a description of the topology.
     */
    @GetMapping("/topology")
    public ResponseEntity<String> getTopology() {
        return ResponseEntity.ok(factoryBean.getTopology().describe().toString());
    }
}
```

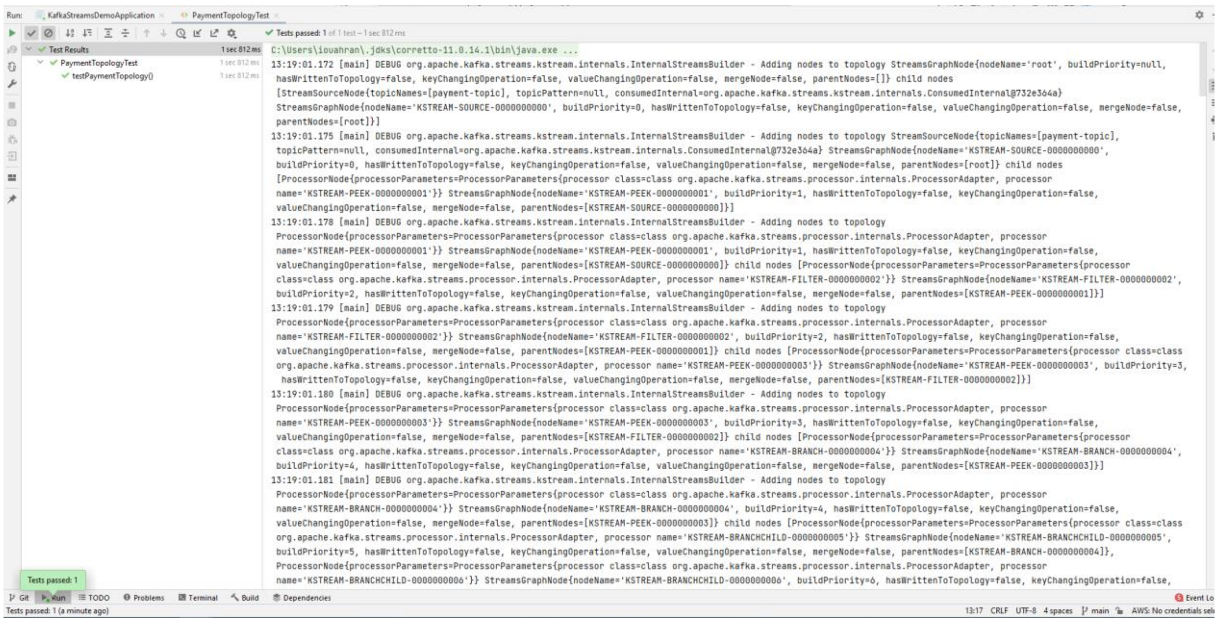
http://localhost:9001/v1/kafka-streams/topology

PaymentTopologyTest

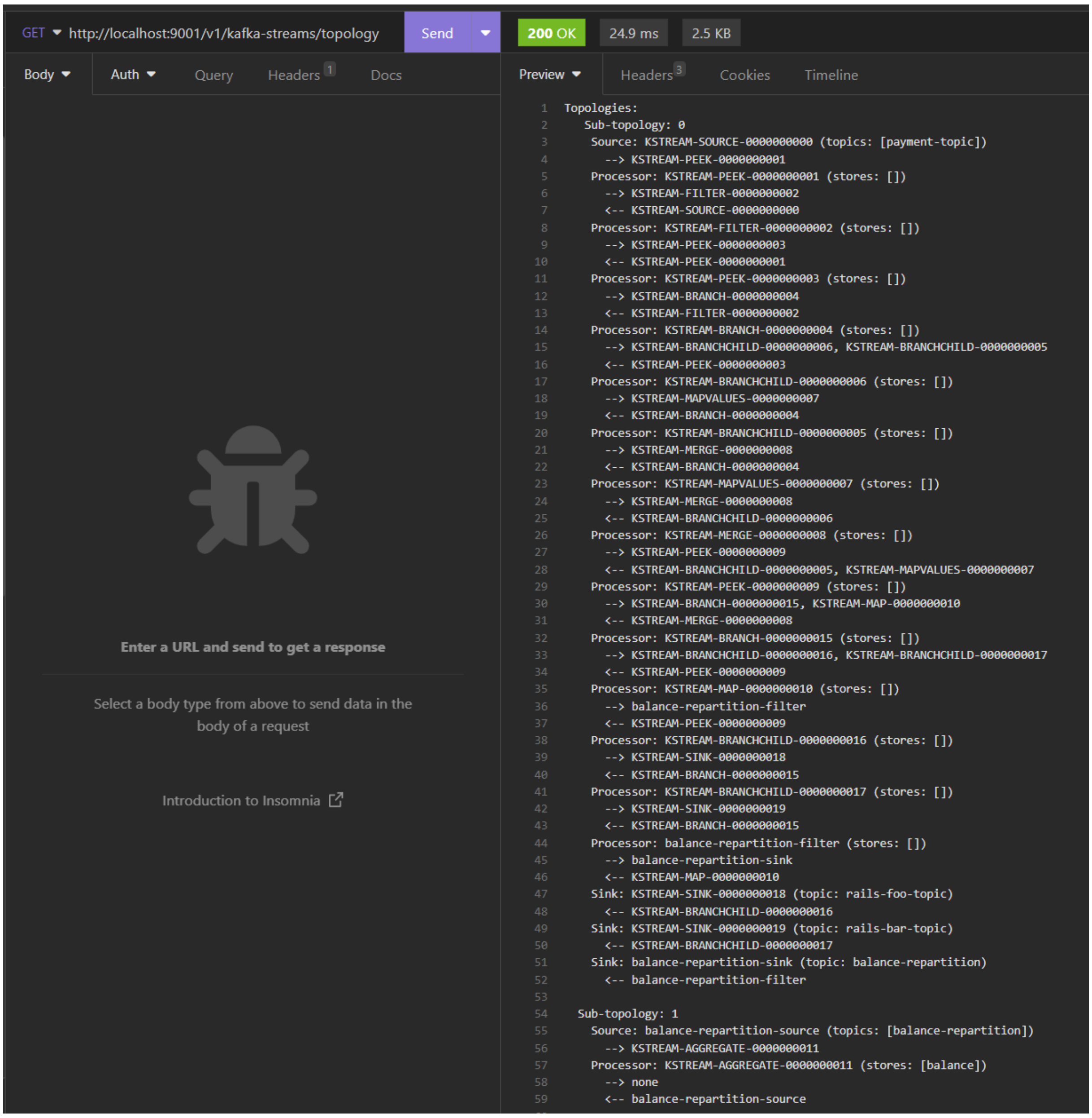
```
package demo.kafka.streams.processor;

import java.util.Properties;
import java.util.UUID;

import demo.kafka.streams.event.PaymentEvent;
import demo.kafka.streams.properties.KafkaStreamsDemoProperties;
import demo.kafka.streams.serdes.PaymentSerdes;
import org.apache.kafka.common.serialization.Serdes;
import org.apache.kafka.common.serialization.StringDeserializer;
import org.apache.kafka.common.serialization.StringSerializer;
import org.apache.kafka.streams.KeyValue;
import org.apache.kafka.streams.StreamsBuilder;
import org.apache.kafka.streams.TestInputTopic;
import org.apache.kafka.streams.TestOutputTopic;
import org.apache.kafka.streams.Topology;
import org.apache.kafka.streams.TopologyTestDriver;
import org.apache.kafka.streams.state.KeyValueStore;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

import static demo.kafka.streams.processor.Rails.BANK_RAILS_BAR;
import static demo.kafka.streams.processor.Rails.BANK_RAILS_FOO;
import static demo.kafka.streams.processor.Rails.BANK_RAILS_XXX;
import static demo.kafka.streams.util.TestEventData.buildPaymentEvent;
import static
org.apache.kafka.streams.StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG;
import static
org.apache.kafka.streams.StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.equalTo;
import static org.hamcrest.Matchers.hasItems;
import static org.hamcrest.Matchers.nullValue;
import static org.mockito.Mockito.mock;
import static org.mockito.Mockito.when;

class PaymentTopologyTest {

  private PaymentTopology paymentTopology;
  private KafkaStreamsDemoProperties properties;

  private static final String PAYMENT_INBOUND_TOPIC = "payment-topic";
  private static final String RAILS_FOO_OUTBOUND_TOPIC = "rails-foo-topic";
  private static final String RAILS_BAR_OUTBOUND_TOPIC = "rails-BAR-topic";

  // GBP Accounts.
  private static final String ACCOUNT_GBP_ABC = "ABC-"+UUID.randomUUID();
  private static final String ACCOUNT_GBP_DEF = "DEF-"+UUID.randomUUID();

  // USD Accounts.
  private static final String ACCOUNT_USD_XYZ = "XYZ-"+UUID.randomUUID();

  @BeforeEach
  void setUp() {
    properties = mock(KafkaStreamsDemoProperties.class);
    when(properties.getPaymentInboundTopic()).thenReturn(PAYMENT_INBOUND_TOPIC);
when(properties.getRailsFooOutboundTopic()).thenReturn(RAILS_FOO_OUTBOUND_TOPIC);

when(properties.getRailsBarOutboundTopic()).thenReturn(RAILS_BAR_OUTBOUND_TOPIC);

        paymentTopology = new PaymentTopology(properties);
    }

    @Test
    void testPaymentTopology() {
        StreamsBuilder streamsBuilder = new StreamsBuilder();

        paymentTopology.buildPipeline(streamsBuilder);
        Topology topology = streamsBuilder.build();

        Properties streamsConfiguration = new Properties();

        streamsConfiguration.put(DEFAULT_KEY_SERDE_CLASS_CONFIG,
Serdes.String().getClass().getName());
        streamsConfiguration.put(DEFAULT_VALUE_SERDE_CLASS_CONFIG,
Serdes.Long().getClass().getName());

        TopologyTestDriver topologyTestDriver = new TopologyTestDriver(topology,
streamsConfiguration);
// Tests sur input et output topic
        TestInputTopic<String, PaymentEvent> inputTopic = topologyTestDriver
                .createInputTopic(PAYMENT_INBOUND_TOPIC, new StringSerializer(),
PaymentSerdes.serdes().serializer());

        TestOutputTopic<String, PaymentEvent> railsFooOutputTopic = topologyTestDriver
                .createOutputTopic(RAILS_FOO_OUTBOUND_TOPIC, new StringDeserializer(),
PaymentSerdes.serdes().deserializer());

        TestOutputTopic<String, PaymentEvent> railsBarOutputTopic = topologyTestDriver
                .createOutputTopic(RAILS_BAR_OUTBOUND_TOPIC, new StringDeserializer(),
PaymentSerdes.serdes().deserializer());

        // Three payments via FOO rails from ABC to DEF, total 210 GBP.
        PaymentEvent payment1 = buildPaymentEvent(UUID.randomUUID().toString(),
                100L,
                "GBP",
                ACCOUNT_GBP_ABC,
                ACCOUNT_GBP_DEF,
                BANK_RAILS_FOO.name());
        inputTopic.pipeInput(payment1.getPaymentId(), payment1);
        PaymentEvent payment2 = buildPaymentEvent(UUID.randomUUID().toString(),
                50L,
            "GBP",
            ACCOUNT_GBP_ABC,
            ACCOUNT_GBP_DEF,
            BANK_RAILS_FOO.name());
12:19:20.610 INFO  d.k.s.p.PaymentTopology - Payment event received with key=3670b4f6-
28ad-41c8-ae5a-944d2d8c9cdd, payment=PaymentEvent(paymentId=3670b4f6-28ad-41c8-
ae5a-944d2d8c9cdd, amount=50, currency=GBP, toAccount=DEF-c96ded21-16f6-42be-a692-
d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)       inputTopic.pipeInput(payment2.getPaymentId(), payment2);
        PaymentEvent payment3 = buildPaymentEvent(UUID.randomUUID().toString(),
            60L,
            "GBP",
            ACCOUNT_GBP_ABC,
            ACCOUNT_GBP_DEF,
            BANK_RAILS_FOO.name());
        inputTopic.pipeInput(payment3.getPaymentId(), payment3);

        // Payment on an unsupported rails should be filtered out.
        PaymentEvent payment4 = buildPaymentEvent(UUID.randomUUID().toString(),
            1200L,
            "GBP",
            ACCOUNT_GBP_ABC,
            ACCOUNT_GBP_DEF,
            BANK_RAILS_XXX.name());
        inputTopic.pipeInput(payment4.getPaymentId(), payment4);

        // Payment from a USD account will require FX.
        PaymentEvent payment5 = buildPaymentEvent(UUID.randomUUID().toString(),
            1000L,  // Converts to 800 GBP.
            "USD",
            ACCOUNT_USD_XYZ,
            ACCOUNT_GBP_DEF,
            BANK_RAILS_BAR.name());
        inputTopic.pipeInput(payment5.getPaymentId(), payment5);

        // Assert the outbound rails topics have the expected events.
        assertThat(railsFooOutputTopic.readKeyValuesToList(),
            hasItems(
                KeyValue.pair(payment1.getPaymentId(), payment1),
                KeyValue.pair(payment2.getPaymentId(), payment2),
                KeyValue.pair(payment3.getPaymentId(), payment3)
            ));

        // Expected event after FX transform.
        PaymentEvent payment5fx = buildPaymentEvent(payment5.getPaymentId(),
            800L,
            "GBP",  // Converted from 1000 USD.
```

```
                payment5.getFromAccount(),
                payment5.getToAccount(),
                payment5.getRails());
        assertThat(railsBarOutputTopic.readKeyValuesToList(),
                hasItems(
                        KeyValue.pair(payment5.getPaymentId(), payment5fx)
                ));

        // Expect the balances are correctly aggregated in the state store.
        KeyValueStore<String, Long> balanceStore =
topologyTestDriver.getKeyValueStore("balance");
        assertThat(balanceStore.get(ACCOUNT_GBP_ABC), equalTo(210L)); // Payments: 100 +
60 + 50.
        assertThat(balanceStore.get(ACCOUNT_GBP_DEF), nullValue()); // No payments from
this account.
        assertThat(balanceStore.get(ACCOUNT_USD_XYZ), equalTo(800L)); // 1000 USD * 0.8
FX.
    }
}
```

```
12:19:20.531 INFO  d.k.s.p.PaymentTopology - Payment event received with key=14706efd-
8e6b-49fe-a565-47b9f9a4f17c, payment=PaymentEvent(paymentId=14706efd-8e6b-49fe-
a565-47b9f9a4f17c, amount=100, currency=GBP, toAccount=DEF-c96ded21-16f6-42be-
a692-d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.534 INFO  d.k.s.p.PaymentTopology - Filtered payment event received with
key=14706efd-8e6b-49fe-a565-47b9f9a4f17c, value=PaymentEvent(paymentId=14706efd-
8e6b-49fe-a565-47b9f9a4f17c, amount=100, currency=GBP, toAccount=DEF-c96ded21-16f6-
42be-a692-d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.539 INFO  d.k.s.p.PaymentTopology - Merged payment event received with
key=14706efd-8e6b-49fe-a565-47b9f9a4f17c, value=PaymentEvent(paymentId=14706efd-
8e6b-49fe-a565-47b9f9a4f17c, amount=100, currency=GBP, toAccount=DEF-c96ded21-16f6-
42be-a692-d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.610 INFO  d.k.s.p.PaymentTopology - Payment event received with key=3670b4f6-
28ad-41c8-ae5a-944d2d8c9cdd, payment=PaymentEvent(paymentId=3670b4f6-28ad-41c8-
ae5a-944d2d8c9cdd, amount=50, currency=GBP, toAccount=DEF-c96ded21-16f6-42be-a692-
d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.611 INFO  d.k.s.p.PaymentTopology - Filtered payment event received with
key=3670b4f6-28ad-41c8-ae5a-944d2d8c9cdd, value=PaymentEvent(paymentId=3670b4f6-
28ad-41c8-ae5a-944d2d8c9cdd, amount=50, currency=GBP, toAccount=DEF-c96ded21-16f6-
42be-a692-d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.611 INFO  d.k.s.p.PaymentTopology - Merged payment event received with
key=3670b4f6-28ad-41c8-ae5a-944d2d8c9cdd, value=PaymentEvent(paymentId=3670b4f6-
28ad-41c8-ae5a-944d2d8c9cdd, amount=50, currency=GBP, toAccount=DEF-c96ded21-16f6-
42be-a692-d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.642 INFO  d.k.s.p.PaymentTopology - Payment event received with key=1a9839c4-
f868-4d06-a3ec-f3845bc6951d, payment=PaymentEvent(paymentId=1a9839c4-f868-4d06-
a3ec-f3845bc6951d, amount=60, currency=GBP, toAccount=DEF-c96ded21-16f6-42be-a692-
d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.643 INFO  d.k.s.p.PaymentTopology - Filtered payment event received with
key=1a9839c4-f868-4d06-a3ec-f3845bc6951d, value=PaymentEvent(paymentId=1a9839c4-
f868-4d06-a3ec-f3845bc6951d, amount=60, currency=GBP, toAccount=DEF-c96ded21-16f6-
42be-a692-d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.643 INFO  d.k.s.p.PaymentTopology - Merged payment event received with
key=1a9839c4-f868-4d06-a3ec-f3845bc6951d, value=PaymentEvent(paymentId=1a9839c4-
f868-4d06-a3ec-f3845bc6951d, amount=60, currency=GBP, toAccount=DEF-c96ded21-16f6-
42be-a692-d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_FOO)
12:19:20.673 INFO  d.k.s.p.PaymentTopology - Payment event received with key=354022e6-
52b5-4f15-8b12-d41bcfb9356e, payment=PaymentEvent(paymentId=354022e6-52b5-4f15-
8b12-d41bcfb9356e, amount=1200, currency=GBP, toAccount=DEF-c96ded21-16f6-42be-
a692-d76512227475, fromAccount=ABC-7afe3e36-366e-453e-9de2-55895bfd2598,
rails=BANK_RAILS_XXX)
12:19:20.685 INFO  d.k.s.p.PaymentTopology - Payment event received with key=e862b176-
e81e-478a-8a19-5676a5c088b9, payment=PaymentEvent(paymentId=e862b176-e81e-478a-
8a19-5676a5c088b9, amount=1000, currency=USD, toAccount=DEF-c96ded21-16f6-42be-
a692-d76512227475, fromAccount=XYZ-bd919f74-4887-4c6f-a162-95f7c2fadbc0,
rails=BANK_RAILS_BAR)
12:19:20.685 INFO  d.k.s.p.PaymentTopology - Filtered payment event received with
key=e862b176-e81e-478a-8a19-5676a5c088b9,
value=PaymentEvent(paymentId=e862b176-e81e-478a-8a19-5676a5c088b9, amount=1000,
currency=USD, toAccount=DEF-c96ded21-16f6-42be-a692-d76512227475,
fromAccount=XYZ-bd919f74-4887-4c6f-a162-95f7c2fadbc0, rails=BANK_RAILS_BAR)
12:19:20.685 INFO  d.k.s.p.PaymentTopology - Merged payment event received with
key=e862b176-e81e-478a-8a19-5676a5c088b9,
value=PaymentEvent(paymentId=e862b176-e81e-478a-8a19-5676a5c088b9, amount=800,
currency=GBP, toAccount=DEF-c96ded21-16f6-42be-a692-d76512227475,
fromAccount=XYZ-bd919f74-4887-4c6f-a162-95f7c2fadbc0, rails=BANK_RAILS_BAR)
```


Run:
KafkaStreamsDemoApplication
PaymentTopologyTest
Tests passed: 1 of 1 test – 1 sec 812 ms
Test Results 1 sec 812 ms
PaymentTopologyTest 1 sec 812 ms
testPaymentTopology() 1 sec 812 ms
Tests passed: 1
Git
Run
TODO
Problems
Terminal
Build
Dependencies
Event Log
Tests passed: 1 (a minute ago)
13:17 CRLF UTF-8 4 spaces main AWS: No credentials sele

```
C:\Users\iouahran\.jdks\corretto-11.0.14.1\bin\java.exe ...
13:19:01.172 [main] DEBUG org.apache.kafka.streams.kstream.internals.InternalStreamsBuilder - Adding nodes to topology StreamsGraphNode{nodeName='root', buildPriority=null,
 hasWrittenToTopology=false, keyChangingOperation=false, valueChangingOperation=false, mergeNode=false, parentNodes=[]} child nodes
 [StreamSourceNode{topicNames=[payment-topic], topicPattern=null, consumedInternal=org.apache.kafka.streams.kstream.internals.ConsumedInternal@732e364a}
 StreamsGraphNode{nodeName='KSTREAM-SOURCE-0000000000', buildPriority=0, hasWrittenToTopology=false, keyChangingOperation=false, valueChangingOperation=false, mergeNode=false,
 parentNodes=[root]}]
13:19:01.175 [main] DEBUG org.apache.kafka.streams.kstream.internals.InternalStreamsBuilder - Adding nodes to topology StreamSourceNode{topicNames=[payment-topic],
 topicPattern=null, consumedInternal=org.apache.kafka.streams.kstream.internals.ConsumedInternal@732e364a} StreamsGraphNode{nodeName='KSTREAM-SOURCE-0000000000',
 buildPriority=0, hasWrittenToTopology=false, keyChangingOperation=false, valueChangingOperation=false, mergeNode=false, parentNodes=[root]} child nodes
 [ProcessorNode{processorParameters=ProcessorParameters{processor class=class org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor
 name='KSTREAM-PEEK-0000000001'}} StreamsGraphNode{nodeName='KSTREAM-PEEK-0000000001', buildPriority=1, hasWrittenToTopology=false, keyChangingOperation=false,
 valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-SOURCE-0000000000]}]
13:19:01.178 [main] DEBUG org.apache.kafka.streams.kstream.internals.InternalStreamsBuilder - Adding nodes to topology
 ProcessorNode{processorParameters=ProcessorParameters{processor class=class org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor
 name='KSTREAM-PEEK-0000000001'}} StreamsGraphNode{nodeName='KSTREAM-PEEK-0000000001', buildPriority=1, hasWrittenToTopology=false, keyChangingOperation=false,
 valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-SOURCE-0000000000]} child nodes [ProcessorNode{processorParameters=ProcessorParameters{processor
 class=class org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor name='KSTREAM-FILTER-0000000002'}} StreamsGraphNode{nodeName='KSTREAM-FILTER-0000000002',
 buildPriority=2, hasWrittenToTopology=false, keyChangingOperation=false, valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-PEEK-0000000001]}]
13:19:01.179 [main] DEBUG org.apache.kafka.streams.kstream.internals.InternalStreamsBuilder - Adding nodes to topology
 ProcessorNode{processorParameters=ProcessorParameters{processor class=class org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor
 name='KSTREAM-FILTER-0000000002'}} StreamsGraphNode{nodeName='KSTREAM-FILTER-0000000002', buildPriority=2, hasWrittenToTopology=false, keyChangingOperation=false,
 valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-PEEK-0000000001]} child nodes [ProcessorNode{processorParameters=ProcessorParameters{processor class=class
 org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor name='KSTREAM-PEEK-0000000003'}} StreamsGraphNode{nodeName='KSTREAM-PEEK-0000000003', buildPriority=3,
  hasWrittenToTopology=false, keyChangingOperation=false, valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-FILTER-0000000002]}]
13:19:01.180 [main] DEBUG org.apache.kafka.streams.kstream.internals.InternalStreamsBuilder - Adding nodes to topology
 ProcessorNode{processorParameters=ProcessorParameters{processor class=class org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor
 name='KSTREAM-PEEK-0000000003'}} StreamsGraphNode{nodeName='KSTREAM-PEEK-0000000003', buildPriority=3, hasWrittenToTopology=false, keyChangingOperation=false,
 valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-FILTER-0000000002]} child nodes [ProcessorNode{processorParameters=ProcessorParameters{processor
 class=class org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor name='KSTREAM-BRANCH-0000000004'}} StreamsGraphNode{nodeName='KSTREAM-BRANCH-0000000004',
 buildPriority=4, hasWrittenToTopology=false, keyChangingOperation=false, valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-PEEK-0000000003]}]
13:19:01.181 [main] DEBUG org.apache.kafka.streams.kstream.internals.InternalStreamsBuilder - Adding nodes to topology
 ProcessorNode{processorParameters=ProcessorParameters{processor class=class org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor
 name='KSTREAM-BRANCH-0000000004'}} StreamsGraphNode{nodeName='KSTREAM-BRANCH-0000000004', buildPriority=4, hasWrittenToTopology=false, keyChangingOperation=false,
 valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-PEEK-0000000003]} child nodes [ProcessorNode{processorParameters=ProcessorParameters{processor class=class
 org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor name='KSTREAM-BRANCHCHILD-0000000005'}} StreamsGraphNode{nodeName='KSTREAM-BRANCHCHILD-0000000005',
 buildPriority=5, hasWrittenToTopology=false, keyChangingOperation=false, valueChangingOperation=false, mergeNode=false, parentNodes=[KSTREAM-BRANCH-0000000004]},
 ProcessorNode{processorParameters=ProcessorParameters{processor class=class org.apache.kafka.streams.processor.internals.ProcessorAdapter, processor
 name='KSTREAM-BRANCHCHILD-0000000006'}} StreamsGraphNode{nodeName='KSTREAM-BRANCHCHILD-0000000006', buildPriority=6, hasWrittenToTopology=false, keyChangingOperation=false,
```



http://localhost:9001/v1/kafka-streams/topology

200 OK 24.9 ms 2.5 KB

Enter a URL and send to get a response

Select a body type from above to send data in the body of a request

```
Topologies:
   Sub-topology: 0
    Source: KSTREAM-SOURCE-0000000000 (topics: [payment-topic])
      --> KSTREAM-PEEK-0000000001
    Processor: KSTREAM-PEEK-0000000001 (stores: [])
      --> KSTREAM-FILTER-0000000002
      <-- KSTREAM-SOURCE-0000000000
    Processor: KSTREAM-FILTER-0000000002 (stores: [])
      --> KSTREAM-PEEK-0000000003
      <-- KSTREAM-PEEK-0000000001
    Processor: KSTREAM-PEEK-0000000003 (stores: [])
      --> KSTREAM-BRANCH-0000000004
      <-- KSTREAM-FILTER-0000000002
    Processor: KSTREAM-BRANCH-0000000004 (stores: [])
      --> KSTREAM-BRANCHCHILD-0000000006, KSTREAM-BRANCHCHILD-0000000005
      <-- KSTREAM-PEEK-0000000003
    Processor: KSTREAM-BRANCHCHILD-0000000006 (stores: [])
      --> KSTREAM-MAPVALUES-0000000007
      <-- KSTREAM-BRANCH-0000000004
    Processor: KSTREAM-BRANCHCHILD-0000000005 (stores: [])
      --> KSTREAM-MERGE-0000000008
      <-- KSTREAM-BRANCH-0000000004
    Processor: KSTREAM-MAPVALUES-0000000007 (stores: [])
      --> KSTREAM-MERGE-0000000008
      <-- KSTREAM-BRANCHCHILD-0000000006
    Processor: KSTREAM-MERGE-0000000008 (stores: [])
      --> KSTREAM-PEEK-0000000009
      <-- KSTREAM-BRANCHCHILD-0000000005, KSTREAM-MAPVALUES-0000000007
    Processor: KSTREAM-PEEK-0000000009 (stores: [])
      --> KSTREAM-BRANCH-0000000015, KSTREAM-MAP-0000000010
      <-- KSTREAM-MERGE-0000000008
    Processor: KSTREAM-BRANCH-0000000015 (stores: [])
      --> KSTREAM-BRANCHCHILD-0000000016, KSTREAM-BRANCHCHILD-0000000017
      <-- KSTREAM-PEEK-0000000009
    Processor: KSTREAM-MAP-0000000010 (stores: [])
      --> balance-repartition-filter
      <-- KSTREAM-PEEK-0000000009
    Processor: KSTREAM-BRANCHCHILD-0000000016 (stores: [])
      --> KSTREAM-SINK-0000000018
      <-- KSTREAM-BRANCH-0000000015
    Processor: KSTREAM-BRANCHCHILD-0000000017 (stores: [])
      --> KSTREAM-SINK-0000000019
      <-- KSTREAM-BRANCH-0000000015
    Processor: balance-repartition-filter (stores: [])
      --> balance-repartition-sink
      <-- KSTREAM-MAP-0000000010
    Sink: KSTREAM-SINK-0000000018 (topic: rails-foo-topic)
      <-- KSTREAM-BRANCHCHILD-0000000016
    Sink: KSTREAM-SINK-0000000019 (topic: rails-bar-topic)
      <-- KSTREAM-BRANCHCHILD-0000000017
    Sink: balance-repartition-sink (topic: balance-repartition)
      <-- balance-repartition-filter

  Sub-topology: 1
    Source: balance-repartition-source (topics: [balance-repartition])
      --> KSTREAM-AGGREGATE-0000000011
    Processor: KSTREAM-AGGREGATE-0000000011 (stores: [balance])
      --> none
      <-- balance-repartition-source
```

```
Topologies:
  Sub-topology: 0
   Source: KSTREAM-SOURCE-0000000000 (topics: [payment-topic])
     --> KSTREAM-PEEK-0000000001
   Processor: KSTREAM-PEEK-0000000001 (stores: [])
     --> KSTREAM-FILTER-0000000002
     <-- KSTREAM-SOURCE-0000000000
   Processor: KSTREAM-FILTER-0000000002 (stores: [])
     --> KSTREAM-PEEK-0000000003
  <-- KSTREAM-PEEK-0000000001
Processor: KSTREAM-PEEK-0000000003 (stores: [])
  --> KSTREAM-BRANCH-0000000004
  <-- KSTREAM-FILTER-0000000002
Processor: KSTREAM-BRANCH-0000000004 (stores: [])
  --> KSTREAM-BRANCHCHILD-0000000006, KSTREAM-BRANCHCHILD-0000000005
  <-- KSTREAM-PEEK-0000000003
Processor: KSTREAM-BRANCHCHILD-0000000006 (stores: [])
  --> KSTREAM-MAPVALUES-0000000007
  <-- KSTREAM-BRANCH-0000000004
Processor: KSTREAM-BRANCHCHILD-0000000005 (stores: [])
  --> KSTREAM-MERGE-0000000008
  <-- KSTREAM-BRANCH-0000000004
Processor: KSTREAM-MAPVALUES-0000000007 (stores: [])
  --> KSTREAM-MERGE-0000000008
  <-- KSTREAM-BRANCHCHILD-0000000006
Processor: KSTREAM-MERGE-0000000008 (stores: [])
  --> KSTREAM-PEEK-0000000009
  <-- KSTREAM-BRANCHCHILD-0000000005, KSTREAM-MAPVALUES-0000000007
Processor: KSTREAM-PEEK-0000000009 (stores: [])
  --> KSTREAM-BRANCH-0000000015, KSTREAM-MAP-0000000010
  <-- KSTREAM-MERGE-0000000008
Processor: KSTREAM-BRANCH-0000000015 (stores: [])
  --> KSTREAM-BRANCHCHILD-0000000016, KSTREAM-BRANCHCHILD-0000000017
  <-- KSTREAM-PEEK-0000000009
Processor: KSTREAM-MAP-0000000010 (stores: [])
  --> balance-repartition-filter
  <-- KSTREAM-PEEK-0000000009
Processor: KSTREAM-BRANCHCHILD-0000000016 (stores: [])
```

```
  --> KSTREAM-SINK-0000000018
  <-- KSTREAM-BRANCH-0000000015
Processor: KSTREAM-BRANCHCHILD-0000000017 (stores: [])
  --> KSTREAM-SINK-0000000019
  <-- KSTREAM-BRANCH-0000000015
Processor: balance-repartition-filter (stores: [])
  --> balance-repartition-sink
  <-- KSTREAM-MAP-0000000010
Sink: KSTREAM-SINK-0000000018 (topic: rails-foo-topic)
  <-- KSTREAM-BRANCHCHILD-0000000016
Sink: KSTREAM-SINK-0000000019 (topic: rails-bar-topic)
  <-- KSTREAM-BRANCHCHILD-0000000017
Sink: balance-repartition-sink (topic: balance-repartition)
  <-- balance-repartition-filter

Sub-topology: 1
Source: balance-repartition-source (topics: [balance-repartition])
  --> KSTREAM-AGGREGATE-0000000011
Processor: KSTREAM-AGGREGATE-0000000011 (stores: [balance])
  --> none
  <-- balance-repartition-source
```