Lund&Bendsen
developing developers

Kafka connect

What is Kafka Connect

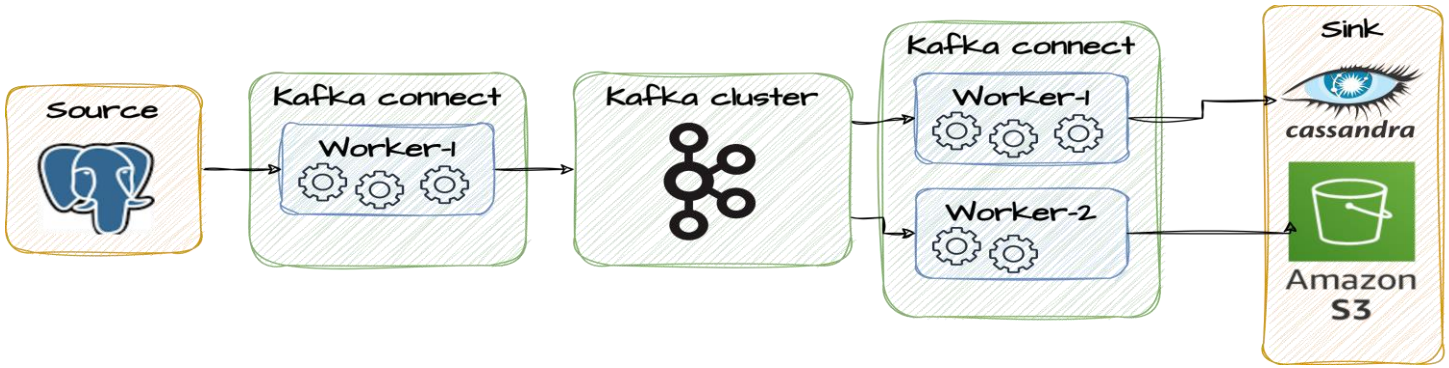
Lund&Bendsen
developing developers

Kafka Connect is for building scalable and reliable data integration pipelines between Apache Kafka and external data systems. It's part of the Apache Kafka project and is designed to simplify the process of ingesting data into Kafka and exporting data from Kafka to various data sources.

Connectors + Tasks: Kafka Connect uses connectors and tasks to interact with various data sources and sinks. Connectors are available for a wide range of databases, storage systems, and other data platforms. These connectors make it easy to configure and manage the data flow.

Configurability: Kafka Connect is highly configurable. You can define and customize the data pipeline through configuration files without the need for writing code.

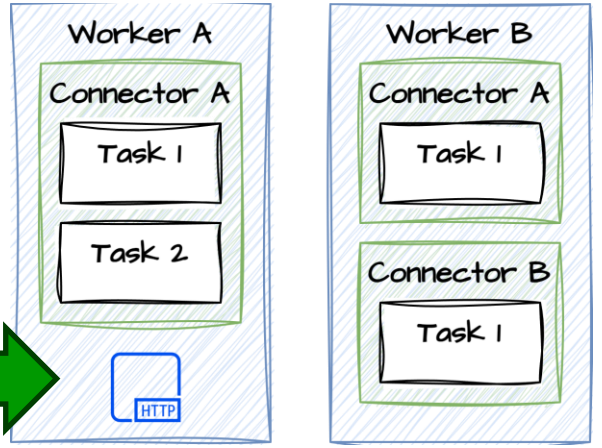
Built-in Converters: Kafka Connect provides built-in converters for data serialization and deserialization. It supports various data formats like Avro, JSON, and more.



Connectors & Tasks

Lund&Bendsen
developing developers

Connectors are responsible for creating configurations for a set of Tasks that split up the data processing. Second, they are responsible for monitoring inputs for changes that require reconfiguration and notifying the Kafka Connect runtime via the ConnectorContext. Kafka Connect will then request new configurations and update the running Tasks. Connectors can be configured during startup of Connect or at runtime by uploading a config file.



Interact with Worker

https://docs.confluent.io/platform/current/connect/kafka_connectors.html

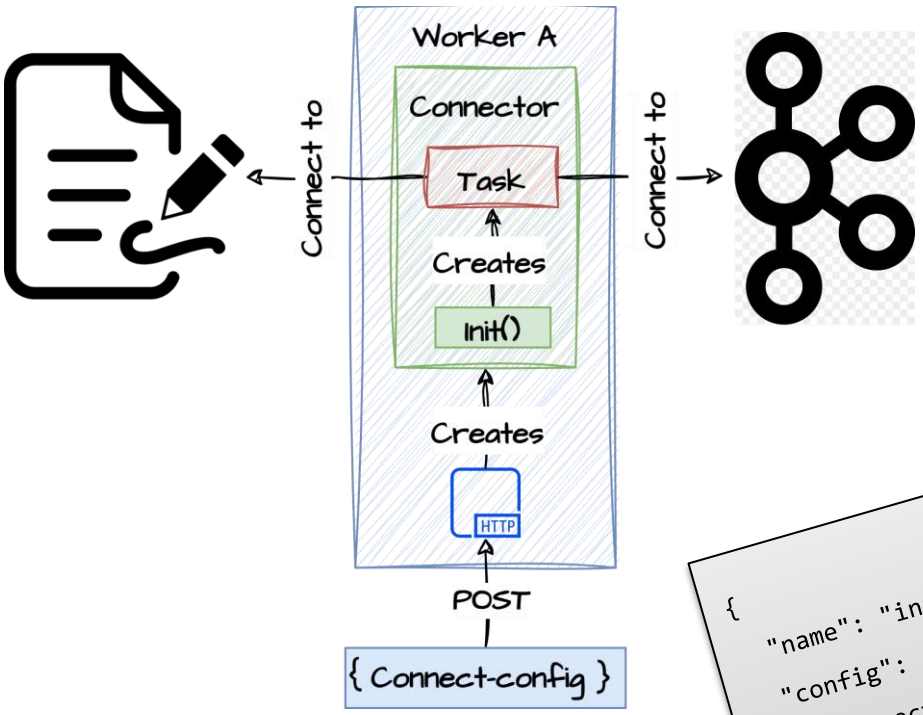
```
public abstract class Connector implements Versioned {
    public abstract void start(Map<String, String> props);
    public abstract Class<? extends Task> taskClass();
    public abstract List<Map<String, String>> taskConfigs(int maxTasks);
    public abstract void stop();
    public abstract ConfigDef config();
}
```

Task Impl of the connector

```
public interface Task {
    String version();
    void start(Map<String, String> props);
    void stop();
}
```

Specified in properties

How to configure connect




```
{
  "name": "input-file-source",
  "config": {
    "connector.class":"org.apache.kafka.connect.file.FileStreamSourceConnector",
    "tasks.max":"1",
    "topic":"input-file-data",
    "file": "/tmp/input.txt"
  }
}
```

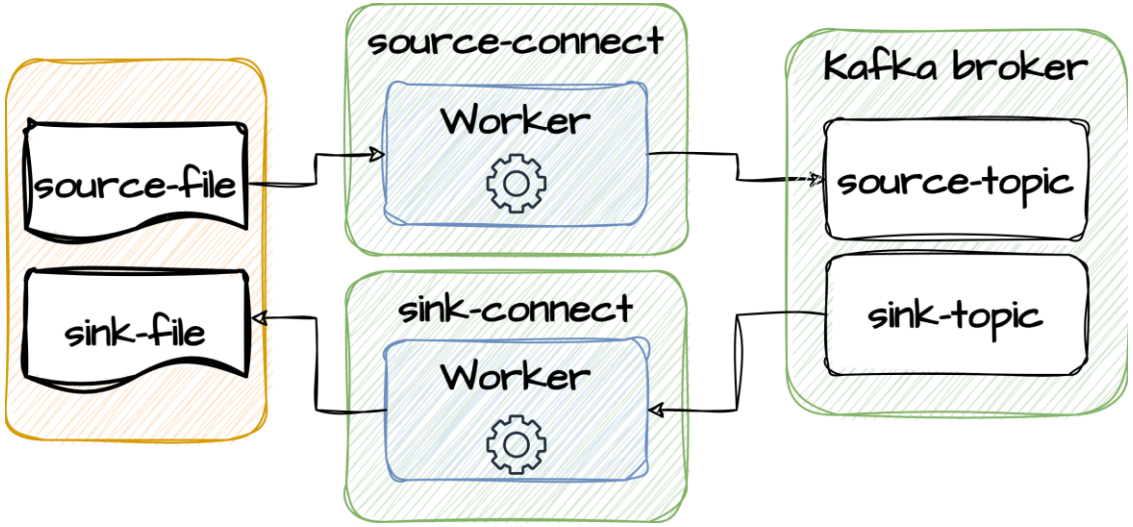
Exercise

Lund&Bendsen
developing developers

Walk through of how the file connectors are configured

- Connect docker container, boot properties
- Mapped volumne
- file-source-config.json
- file-sink-config.json

9.1



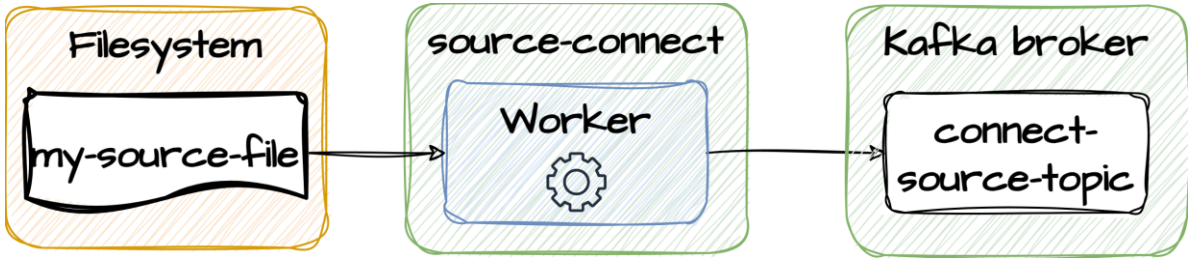
Exercise

Lund&Bendsen
developing developers

9.2

Run a Kafka connector that reads from a file and writes to a topic

- In powershell navigate to <project-folder>/docker/kafka-connect.
- Type **Invoke-WebRequest -Uri http://localhost:8083/connectors/ -Method POST -ContentType application/json -InFile .\file-source-config.json**
- Start a consumer that consumes from the topic specified for the connector properties file
- Open the file that Kafka connect if configured to read from
- Type some lines of text and save the file
- See that the lines you just typed are reflected in the topic

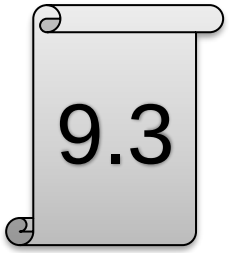


Exercise

Lund&Bendsen
developing developers

Inspect the file connectors using the build in API

- Open a browser and type **http:///localhost:8083/connectors**
- You should see the two connectors that have been configured
- Open a new tab and type **http:///localhost:8083/connectors/*{connector-name}***
- You should see how the connector is configured
- Open a new tab and type http:///localhost:8083/connectors/file-source-connector/status
- You should see runtime information about the connector



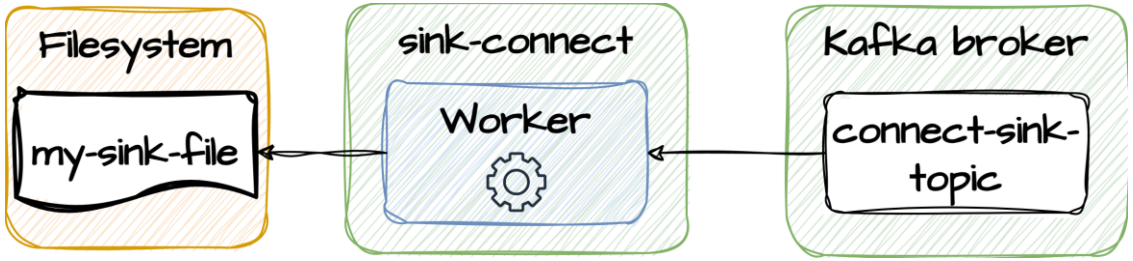
Exercise

Lund&Bendsen
developing developers

9.4

Run a Kafka connector that reads from a topic and writes to a file

- In powershell navigate to <project-folder>/docker/kafka-connect.
- Type **Invoke-WebRequest -Uri http://localhost:8083/connectors/ -Method POST -ContentType application/json -InFile .\file-sink-config.json**
- Start a consumer that consumer from the topic specified for the connector properties file
- Write some messages to the topic
- Open the file that Kafka connect if configured to write to as see that the messages are there



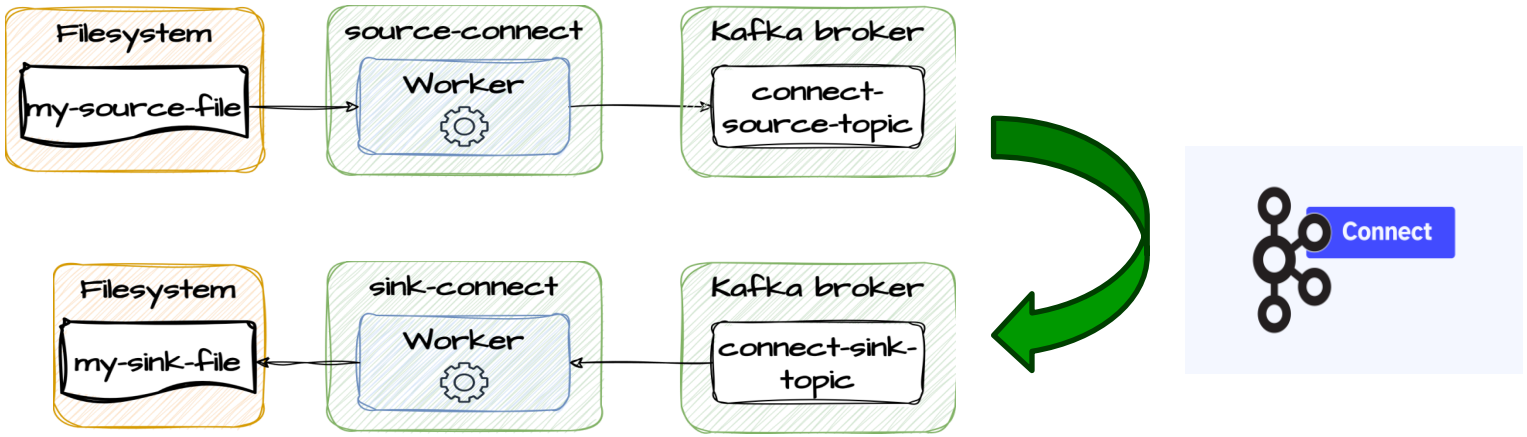
Exercise

Lund&Bendsen
developing developers

Build a Kafka Streams application that reads from one topic to the other

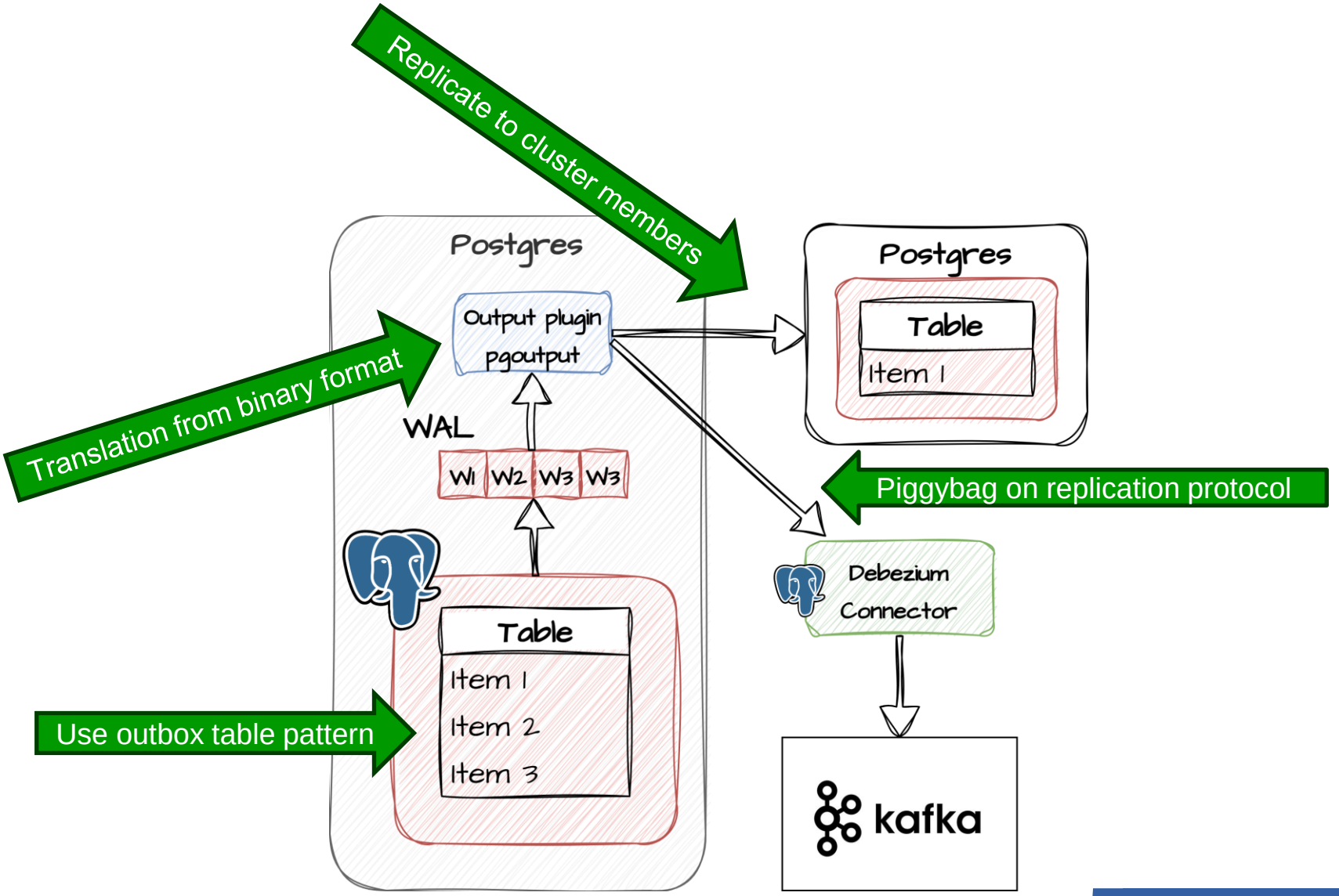
- Create a Kafka Streams application that reads from the input-file-data topic, does some String transformation (eg reverse or upperCase) to the message and writes it to the output-file-data topic
- Start the application
- Open the **input.txt** and type some lines of text. Save file and see if the transformed lines show up in **output.txt**

9.5



Postgres debezium

Lund&Bendsen
developing developers



Exercise

Lund&Bendsen
developing developers

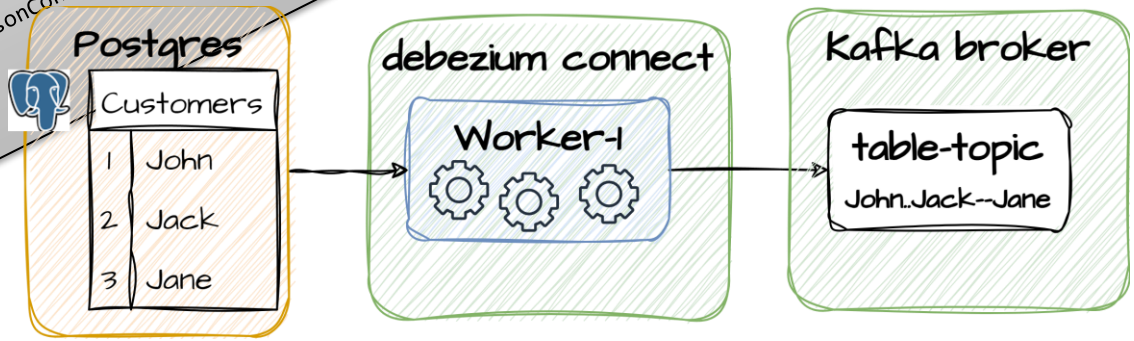
Walk through of how the CDC connector is configured

- Docker compose file: postgres and postgres-connect
- pg-source-config.json

9.6

```
{
  "name": "pg-orders-source",
  "config": {
    "connector.class":    "io.debezium.connector.postgresql.PostgresConnector",
    "database.hostname": "postgres",
    "database.port": "5432",
    "database.user": "postgres",
    "database.password": "postgres",
    "database.dbname": "postgres",
    "database.server.name": "myserver",
    "plugin.name": "pgoutput",
    "table.include.list": "public.customers",
    "value.converter": "org.apache.kafka.connect.json.JsonConverter"

  }
}
```



Exercise

Lund&Bendsen
developing developers

9.7

Configure a Postgres CDC connector

- Look at the Docker Compose file and find the two services: postgres and postgres-connector. Understand the setup!
- Connect to Postgres. Open PowerShell and type **docker exec -it postgres bash**
- Login to Postgres. Type **psql -U postgres**
- Create db-table, type: **CREATE TABLE customers (id int GENERATED ALWAYS AS IDENTITY PRIMARY KEY, name text);**
- Alter table, type: **ALTER TABLE customers REPLICA IDENTITY USING INDEX customers_pkey;**
- Insert data, type: **INSERT INTO customers (name) VALUES ('john'), ('jack'), ('jane');**
- Configure debezium-connector. Open PowerShell and navigate to <project-folder>/docker/kafka-connect.
- Open a browser and type **https://localhost:9090** to confirm the connector is running
- In powershell, type **Invoke-WebRequest -Uri http://localhost:8083/connectors/ -Method POST -ContentType application/json -InFile .\pg-source-config.json**
- Open akhq.io on http://localhost:8080/ to see that a new topic has been created and populated

Debezium transformations

Lund&Bendsen
developing developers

Transform	Description
Topic Routing	Re-routes records to different topics based on a regular expression applied to the original topic name.
Content-Based Routing	Reroute selected events to other topics, based on the event content.
New Record State Extraction	Extracts the flat structure of field names and values from Debezium change events, facilitating sink connectors which cannot process Debezium's complex event structure.
MongoDB New Document State Extraction	The MongoDB-specific counter-part to the New Record State Extraction SMT.
Outbox Event Router	Provides a way to safely and reliably exchange data between multiple (micro) services.
MongoDB Outbox Event Router	The MongoDB-specific counter-part to the Outbox Event Router SMT.
Message Filtering	Applies a filter to the change events emitted by the connectors, based on their content. This lets you propagate only those records that are relevant to you.
Compute Partition	Re-routes records to specific partition based on configured table column.
HeaderToValue	Moves or copies headers into the record value.
Partition Routing	Re-routes records to specific partitions based on configured payload fields.
Timezone Converter	Converts Debezium and Kafka Connect timestamp fields in event records te
TimescaleDB Integration	Routes and enriches messages that the Debezi

Debezium Outbox SMT

```
{
  "name": "pg-orders-sourc
  "config": {
    "connector.clas          .debezium.connector.postgresql.PostgresConnector
    "table.include           "public.customers-outbox"
    ...
    "transforms"=outbox,...
    "transforms.outbox.type": "io.debezium.transforms.outbox.EventRouter"
  }
}
```