



Why should Database people care about Kafka and Debezium?

Dirk Krautschick
PGDay Paris 20.03.2025



Why should **PostgreSQL** people care about Kafka and Debezium?

Dirk Krautschick
PGDay Paris 20.03.2025

#whoami

Dirk Krautschick
Senior Solution Architect

with Aiven since Nov 2023



18 years

DBA, Trainer, Consulting, Sales Engineering

PostgreSQL, Oracle, Kafka, Clickhouse, OpenSearch,...

Married, 2 Junior DBAs

Mountainbike, swimming, movies, music,
hifi/home cinema, 8 bit computing 

PostgreSQL User Group NRW

North Rhine-Westphalia, Germany

Founded Dec 2023

Feb 2024 Cologne, May 2024 Aachen, Oct 2024 Essen,

Feb 2025 Paderborn

Upcoming events in the pipeline, stay tuned!

Target is at least 3 meetups a year

All around NRW



PostgreSQL



Disclaimer

Inspiration and motivation for trying

I am NOT an deep dive streaming guru ...

... just a desperate database guy ...like you!? :-)

Unexceptional Open Source!!!

Everything what looks like advertising won't be meant like such...

Disclaimer

Inspiration and motivation for trying

I am NOT an deep dive streaming guru ...

... just a desperate database guy ...like you!? :-)

Unexceptional Open Source!!!

Everything what looks like advertising won't be meant like such...



...but still it's highly recommended!!! ;-)



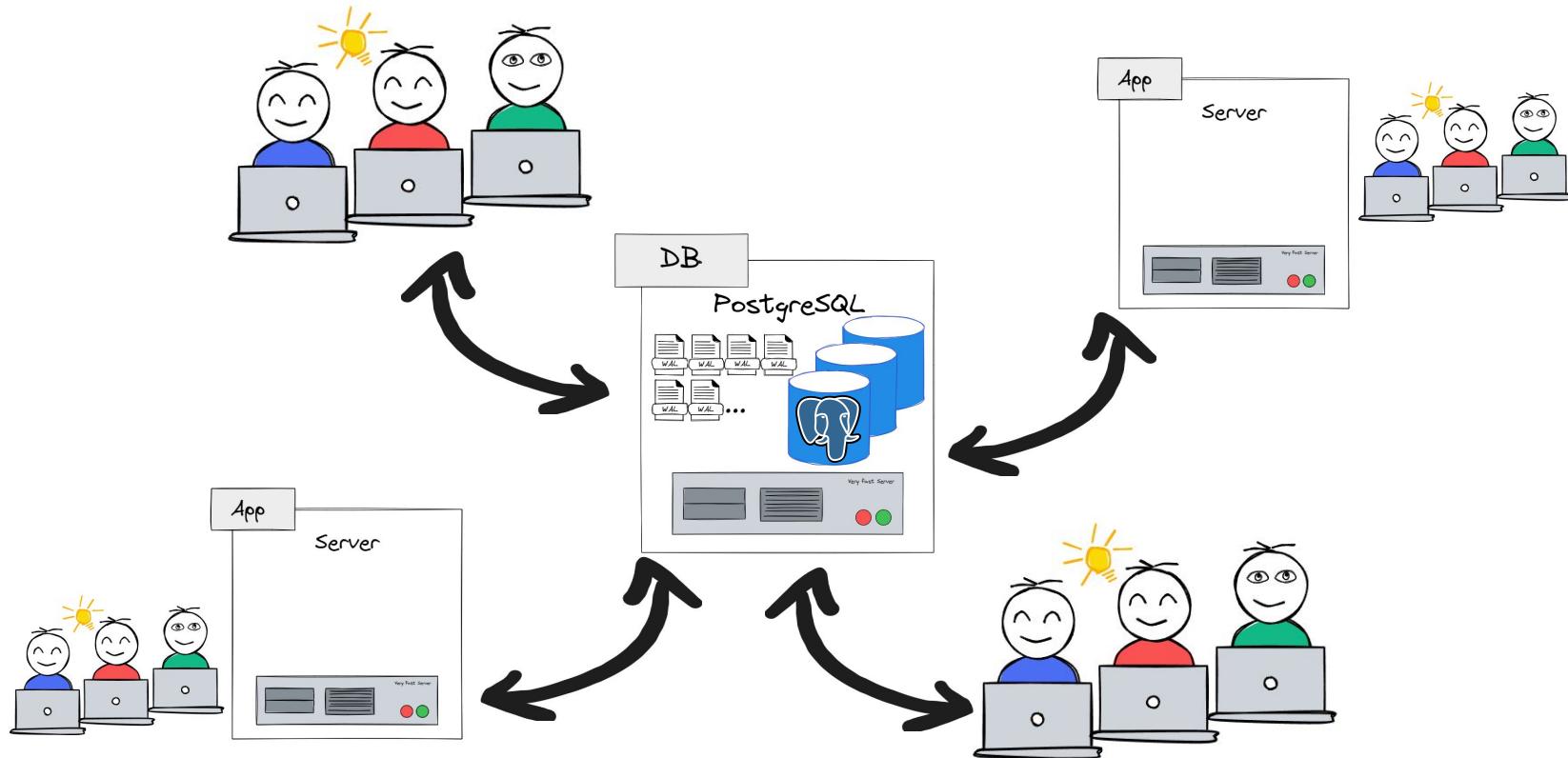
Motivation... once upon a time...

...being a DBA was like....



my server,
my databases,
my responsibilities!!!

Motivation... once upon a time...



Motivation... but today...

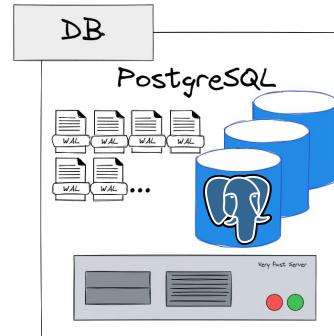
 OpenSearch



 snowflake



DATA MIGRATION



 influxdb



Motivation... but today...

Not just only ordinary clients/applications anymore

Role of DBAs changing

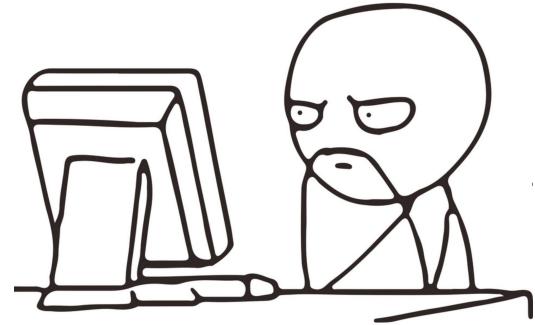
- Cloud/DBaaS vs. on premises

- Architectural and scaling challenges

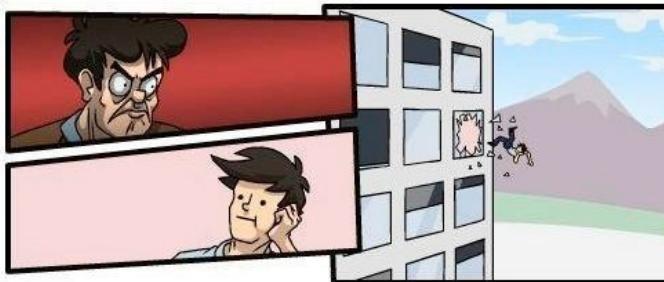
- Different database types for different use cases

Real time data processing and heterogeneity

Rethink your capabilities!? ... like I had to?!



No worries, we'll stick with Databases



Motivation... PostgreSQL rules for all!!!

Extensibility is powerful, that's for sure!

Foreign Data Wrappers

Special data types

...

Some smart interfaces

Just direct connections



...but then...



Idea of a better solution

Decoupled communication/interfaces

Real time processing

Prevent dedicated connections for each use case

Queuing

Change data capture capabilities



Idea of a better solution

Decoupled communication/interfaces

Real time processing

Prevent dedicated connections for each use case

Queuing

Change data capture capabilities

"AT SCALE !!!111eleven" ...of course :-)



→ “Distributes event streaming”

Let's introduce... Kafka

Franz Kafka (3 July 1883 – 3 June 1924) was a German-speaking Bohemian Jewish novelist and writer from Prague.....



....just kidding!

Let's introduce... Apache Kafka



Apache Kafka

Initiated by LinkedIn (Nov 2010)

Apache since 2012

Apache License 2.0

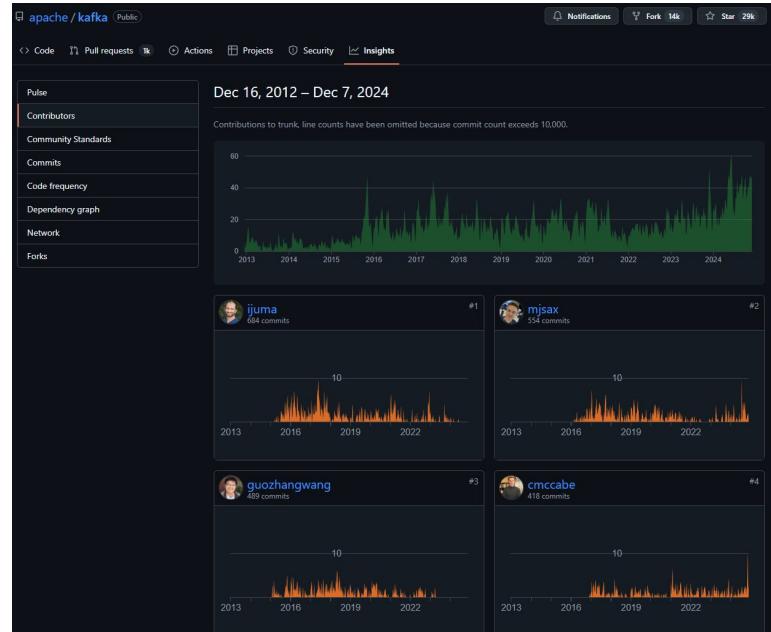
<https://github.com/apache/kafka>

Release v1.0.0 (Nov 2017)

Actual Release v3.9.0 (Nov 2024)

v4 almost there....

Java- and Scala-based



Let's introduce... Apache Kafka



Apache Kafka

Initiated by LinkedIn (**Nov 2010**)

Apache **since 2012**

Apache License 2.0

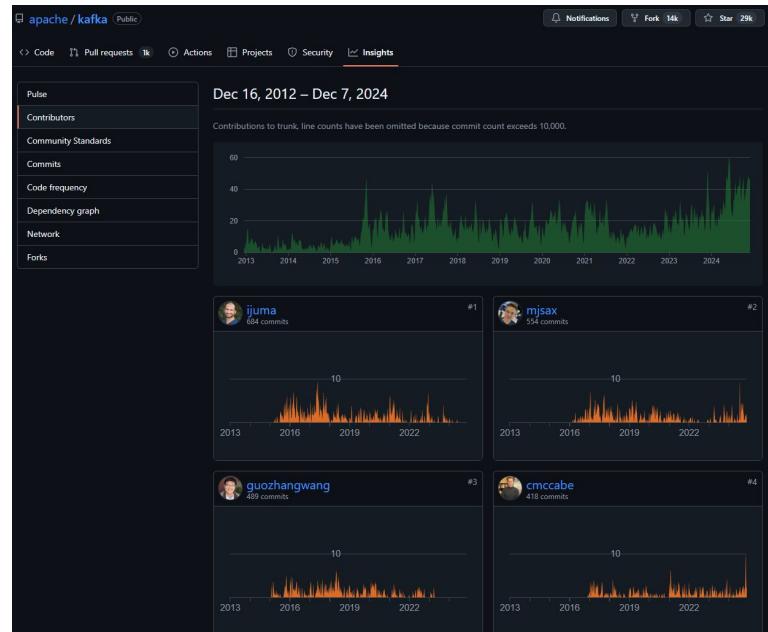
<https://github.com/apache/kafka>

Release v1.0.0 (Nov 2017)

Actual Release v3.9.0 (Nov 2024)

v4 almost there....

Java- and Scala-based



Kafka basics

Events vs. Messages vs. Queues

Streaming vs. Oracle Streams vs. Postgres Streaming Replication vs.

Be careful with the terms!

Here we are talking

“streaming” of “events”, which is usually

Set of key/value pairs

Timestamp, and maybe optional metadata

Kafka basics

Topics

Organized collection of stored events

Basically folders/files on storage

Partitions

Distribution of topic parts on several brokers/nodes

Kafka components

Java JRE/JDK

KRaft, Zookeeper

Controller for observation, cluster control

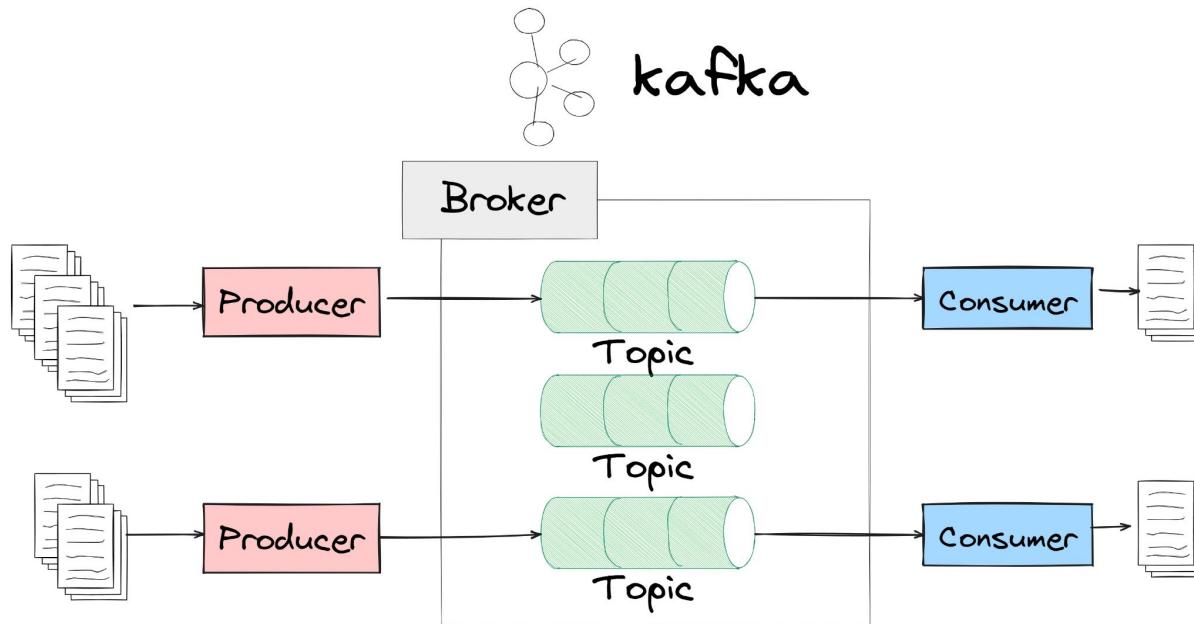
Zookeeper deprecated soon / KRaft default with 4.0

Kafka Broker

Represents an instance/server

Data layer

Kafka components

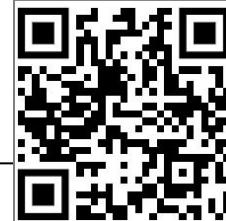


Kafka examples - Usage

```
...  
from kafka import KafkaProducer  
  
producer = KafkaProducer(  
    bootstrap_servers='kafka-dirk.a.aivencloud.com:13138',  
    security_protocol="SSL",  
    ssl_cafile='./ca.pem',  
    ssl_certfile='./service.cert',  
    ssl_keyfile='./service.key',  
    value_serializer=lambda v: json.dumps(v).encode('ascii')  
)  
  
...  
  
producer.send(  
    'some name',  
    value=  
    {  
        /*...any data or messages */  
    }  
)  
producer.flush()  
...
```



```
...  
from kafka import KafkaConsumer  
import json  
  
consumer = KafkaConsumer(  
    bootstrap_servers='kafka-dirk.a.aivencloud.com:13138',  
    security_protocol="SSL",  
    ssl_cafile='./ca.pem',  
    ssl_certfile='./service.cert',  
    ssl_keyfile='./service.key',  
    value_deserializer = lambda v: json.loads(v.decode('ascii')),  
    auto_offset_reset='earliest'  
)  
  
consumer.subscribe(topics='dirk')  
for message in consumer:  
    print ("%d:%d: v=%s" % (message.partition,  
                           message.offset,  
                           message.value))  
...  
...
```



Kafka examples - Usage

The screenshot shows the Aiven Kafka console interface. On the left, a sidebar lists various project and organization management options. The main area displays a Kafka cluster named "kafka-dirk" with Apache Kafka 3.5.1, EOL: 2024-07-31: OK, and 3 nodes. The "Messages" tab is selected, showing messages from partition 0 at offset 24. The message content is decoded from base64 and displayed as JSON:

```
{"key": "89c630d6-c8c5-43d4-80b0-0c7f0b93ae5f", "sensorDataX": 47, "sensorDataY": 54, "timestamp": "2023-10-20"}  
key: "89c630d6-c8c5-43d4-80b0-0c7f0b93ae5f",  
sensorDataX: 47,  
sensorDataY: 54,  
timestamp: "2023-10-20"
```

Below this, another message is partially visible:

```
{"key": "17f0fdff-4af3-4430-b236-c4ccdb0e8904", "sensorDataX": 85, "sensorDataY": 0, "timestamp": "2023-10-20"}
```

Buttons for "Fetch messages" and "Produce message" are also present.

Kafka Advanced – Good 2 Know!

Message formats

AVRO

recommended, less overhead

JSON

Protobuf

Schema Registry

Metadata for topic content

Kafka - An easy quick start...

Installation

<https://kafka.apache.org/downloads>

```
# wget https://downloads.apache.org/kafka/3.9.0/kafka\_2.13-3.9.0.tgz
# tar -xzf kafka_2.13-3.9.0.tgz
```

Yeah....thats it!



Kafka - An easy quick start...

Generate cluster and start Kafka with KRaft

```
# KAFKA_CLUSTER_ID=$(bin/kafka-storage.sh random-uuid)"  
# bin/kafka-storage.sh format -t $KAFKA_CLUSTER_ID -c config/kraft/server.properties  
# bin/kafka-server-start.sh config/kraft/server.properties
```

Or the good old way.... start zookeeper and broker

```
# bin/zookeeper-server-start.sh config/zookeeper.properties  
# bin/kafka-server-start.sh config/server.properties
```

Kafka - An easy quick start...

Create a topic

```
# bin/kafka-topics.sh --create --topic mytopic --bootstrap-server localhost:9092
```

List topics

```
# bin/kafka-topics.sh --list --bootstrap-server localhost:9092
```

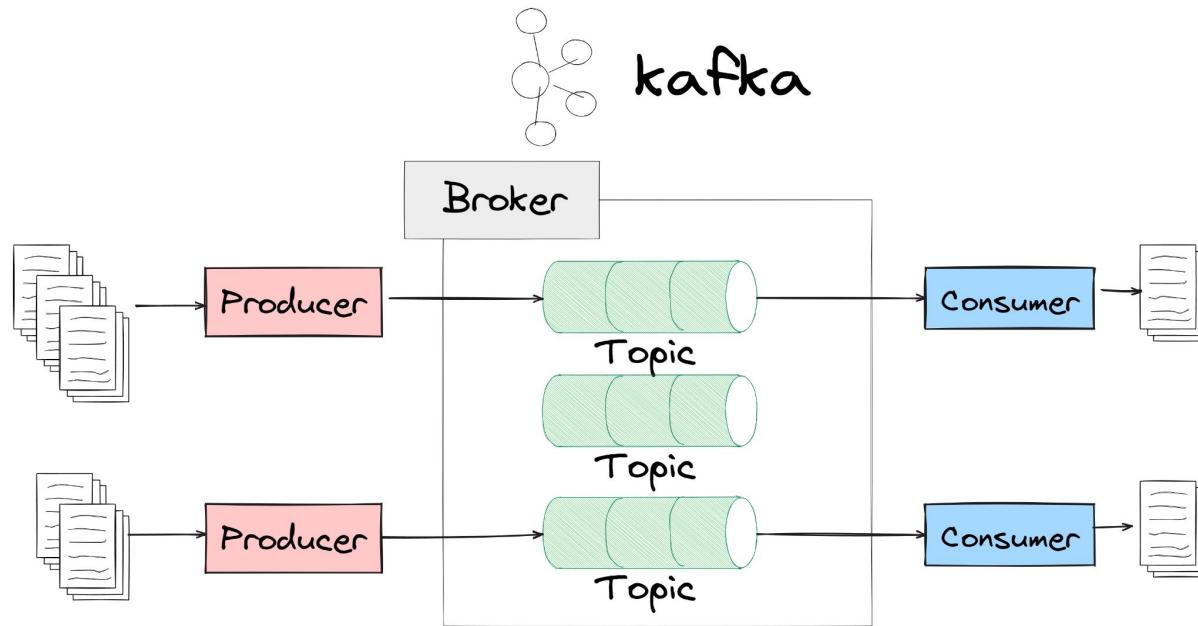
Check, produce and consume events

```
# bin/kafka-topics.sh --describe --topic mytopic --bootstrap-server localhost:9092
# bin/kafka-console-producer.sh --topic mytopic --bootstrap-server localhost:9092
# bin/kafka-console-consumer.sh --topic mytopic --from-beginning --bootstrap-server localhost:9092
```

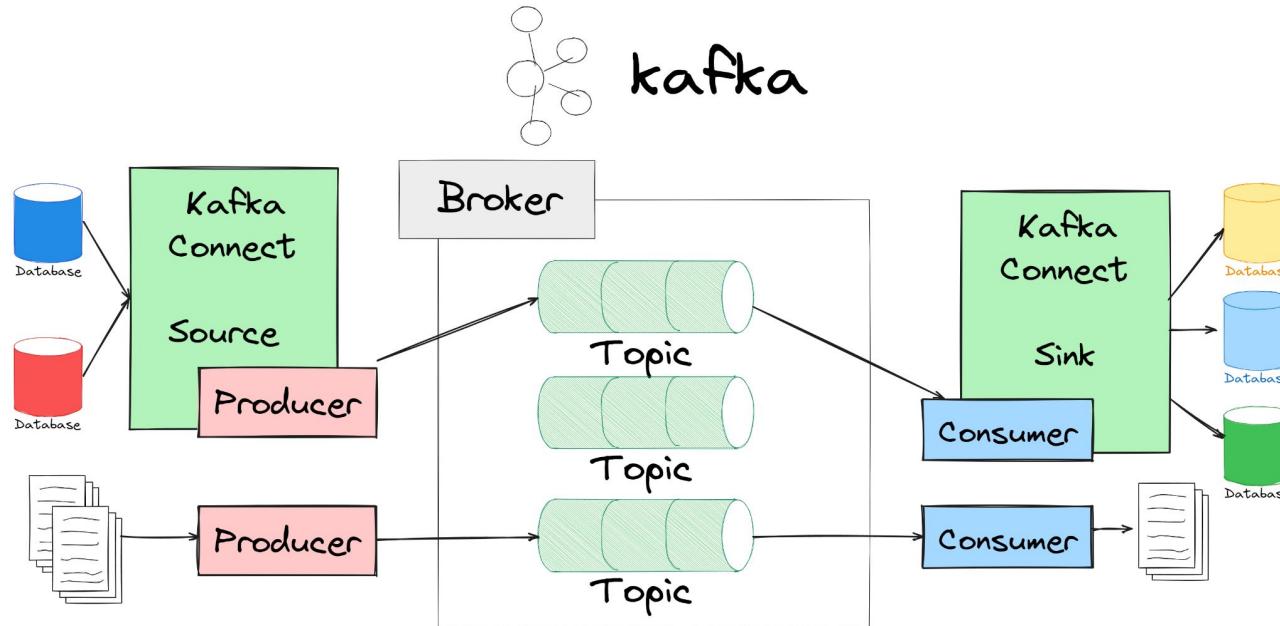
Kafka - An easy quick start...

Demo

Nice, but what about the databases?



Kafka Connect - Overview



Kafka Connect – Overview

Kafka Connect API

Framework for continually pull/push

Stand alone or distributed

Source- and Sink-Connection

Query based approach, but ...

... lots of load, polling time, ordering limitations like updates/deletes

Way better... log based approach (out of the Write Ahead Log)

Let's introduce... Debezium



Debezium

Initiated March 2016

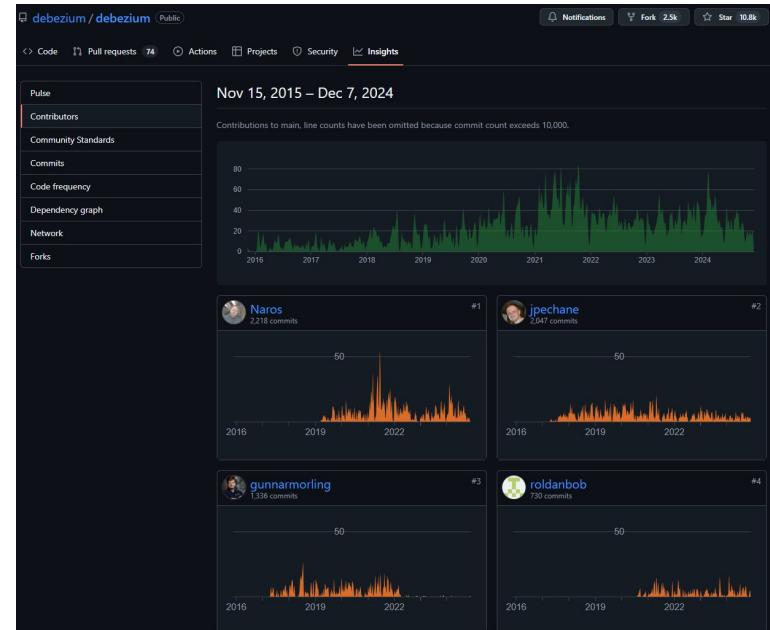
Apache License 2.0

<https://github.com/debezium>

Release v1.0.0 (Dec 2019)

Actual Release v3.0.4 (Dec 2024)

Java-based



Kafka Connect – ...with Debezium

Logical Decoding API for PostgreSQL

Sink Connector for JDBC targets

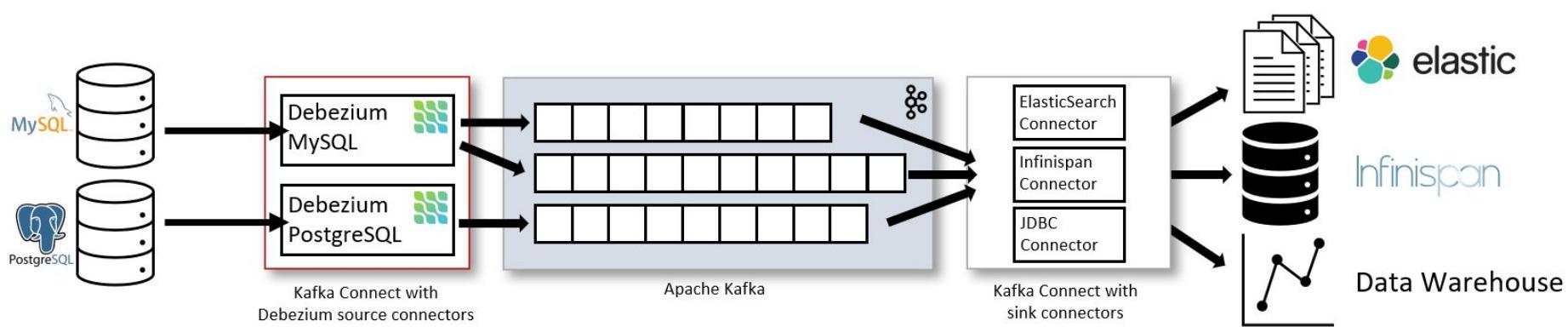
REST API

Connectors for

PostgreSQL

Oracle

...



Kafka Connect – ...with Debezium

Logical replication slots, Publication/Subscription based

Replica Identity if no primary keys available

Single Message Transformation (SMT)

REST API

Decoding plugins

pgoutput

decoderbufs (additional)

Kafka Connect – A quick start...

Download and extract an connector of your choice

<https://debezium.io/releases/>

```
# wget https://repo1.maven.org/...../debezium-connector-postgres-3.0.4.Final-plugin.tar.gz
# mv ./debezium-connector-postgres-3.0.4.Final-plugin.tar.gz <e.g. YOUR KAFKA CLUSTER>
# cd <e.g. YOUR KAFKA CLUSTER>
# tar -xzf debezium-connector-postgres-3.0.4.Final-plugin.tar.gz
```



Kafka Connect – A quick start...

Add the extract path to e.g.

```
# vi connect-distributed.properties OR # vi connect-standalone.properties  
...  
plugin.path=/home/postgres/kafka_2.13-3.9.0/  
...
```

Start Kafka Connect (as standalone on same server)

```
# bin/connect-standalone.sh config/connect-standalone.properties
```

Don't forget to prepare your database

```
wal_level = logical
```

Kafka Connect – A quick start...

Demo

Kafka Connect – A quick start...

Configure Debezium connector

```
curl --location 'http://localhost:8083/connectors' \
--header 'Accept: application/json' \
--header 'Content-Type: application/json' \
--data '{
  "name": "debezium-connector",
  "config": {
    "connector.class": "io.debezium.connector.postgresql.PostgresConnector",
    "database.hostname": "localhost",
    "database.port": "5432",
    "database.user": "postgres",
    "database.password": "postgres",
    "database dbname": "postgres",
    "table.include.list": "public.*",
    "topic.prefix": "cdc",
    "plugin.name": "pgoutput"
  }
}'
```

Kafka Connect - Example test

```
CREATE TABLE beer (
    name varchar(255) primary key
);
ALTER TABLE beer replica identity FULL;

insert into beer values ( 'Chimay' );
insert into beer values ( 'Delirium Tremens' );
insert into beer values ( 'Leffe' );
insert into beer values ( 'Straffe Hendrik' );
insert into beer values ( 'St. Bernadus Abt 12' );

update beer set name = 'Leffe bruin' where name = 'Leffe';
update beer set name = 'Leffe royal' where name = 'Leffe bruin';
update beer set name = 'Chimay Rouge' where name = 'Chimay';

delete from beer;
```

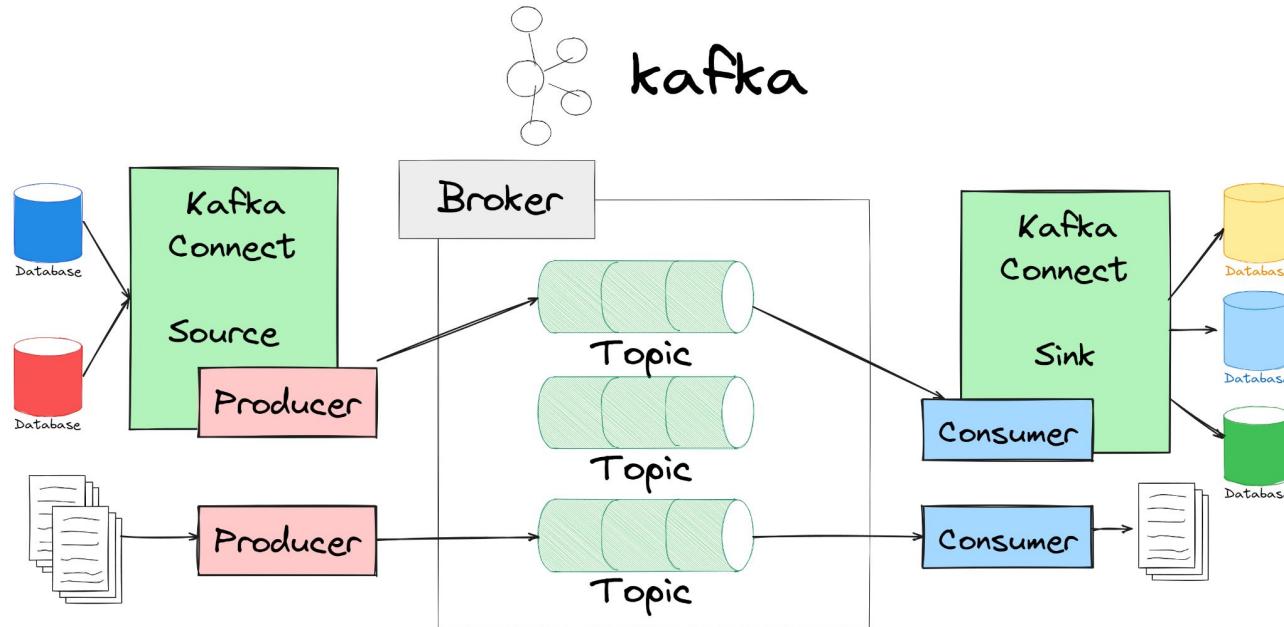
Kafka Connect - Example test

```
# bin/kafka-topics.sh --list --bootstrap-server localhost:9092
    cdc.public.artist

# bin/kafka-console-consumer.sh --topic cdc.public.artist --from-beginning --bootstrap-server
localhost:9092

{"schema": {"type": "struct", "fields": [{"type": "struct", "fields": [{"type": "string", "optional": false, "field": "name"}], "optional": true, "name": "cdc.public.artist.Value", "field": "before"}, {"type": "struct", "fields": [{"type": "string", "optional": false, "field": "name"}], "optional": true, "name": "cdc.public.artist.Value", "field": "after"}, {"type": "struct", "fields": [{"type": "string", "optional": false, "field": "connector"}, {"type": "string", "optional": false, "field": "name"}, {"type": "int64", "optional": false, "field": "ts_ms"}, {"type": "string", "optional": true, "name": "io.debezium.data.Enum", "version": 1, "parameters": {"allowed": "true,last,..."}]}]}
```

Back to our nice setup....



Kafka scales pretty well, several brokers on several nodes...

But still...what if...?



Scaling vs. High Availability

Regional Outages

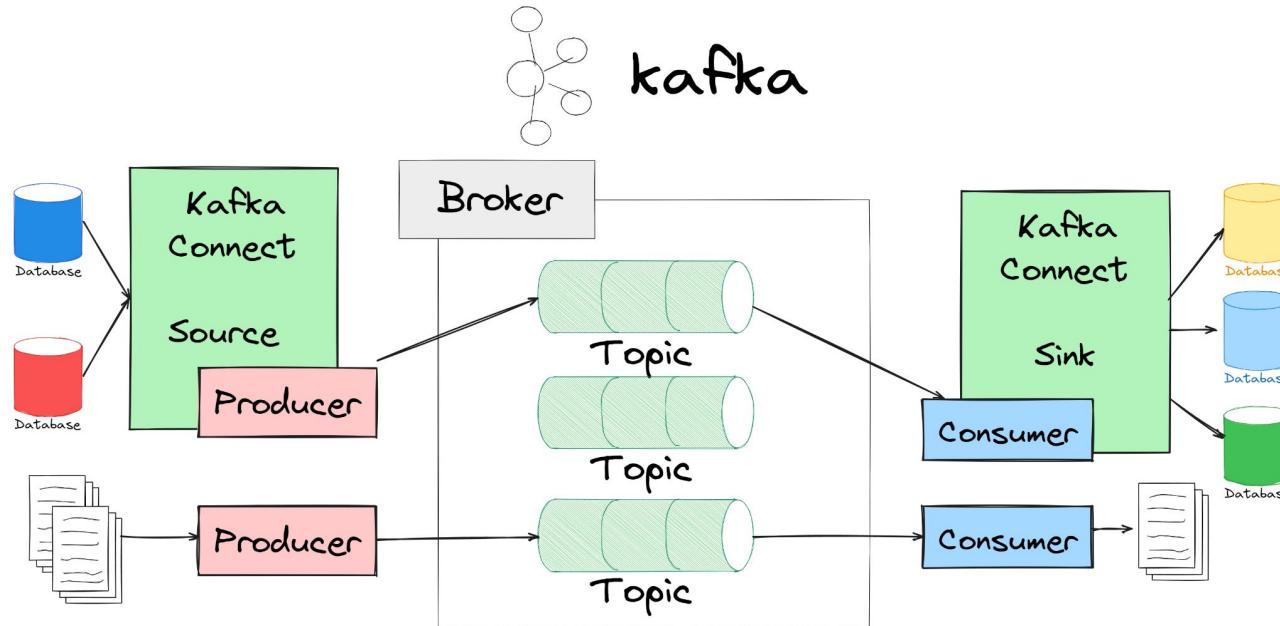
Disaster Recovery

Logical distribution

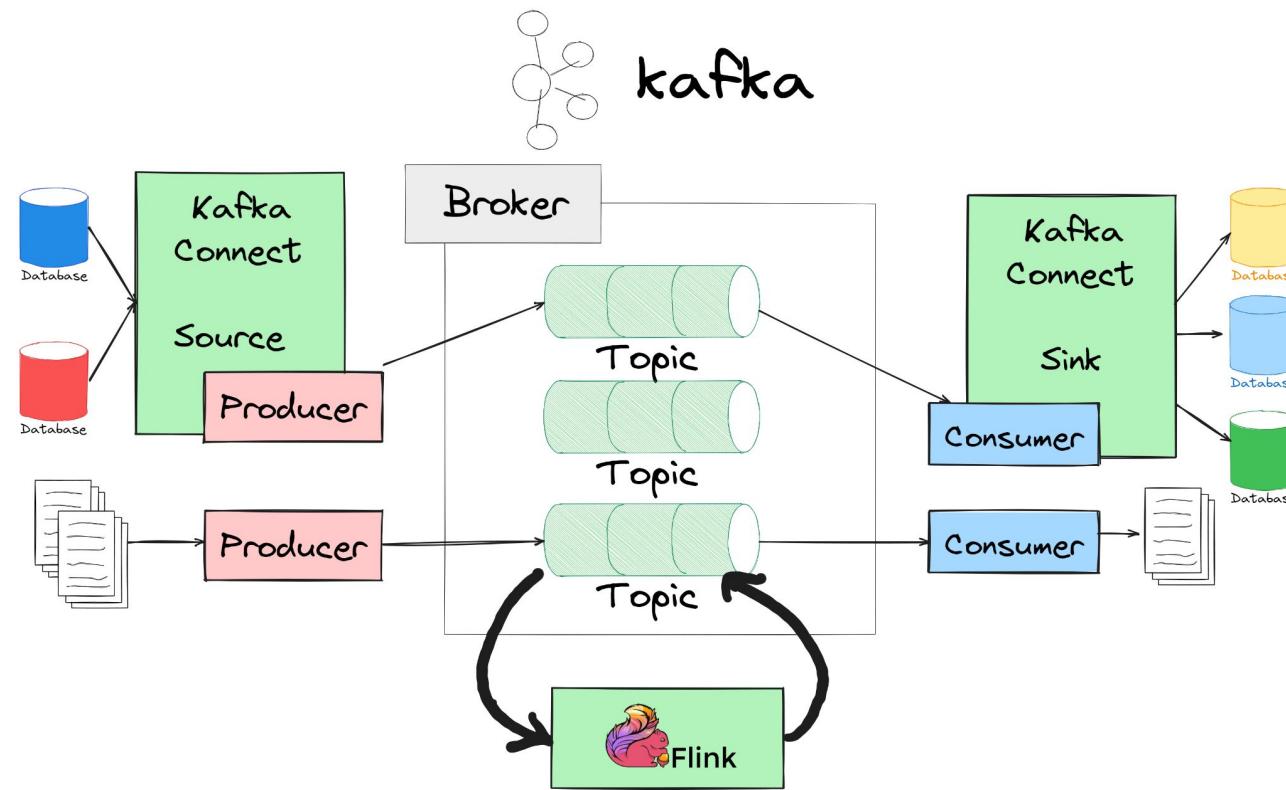
MirrorMaker 2 as a included replication tool

Built on top of Kafka Connect framework

There is even more ...



There is even more ... processing!



There is even more ...

Let's go data pipelining like a boss....!!!!

Apache Flink

Stream processing

Realtime

Transforming data on the fly

"Low latency, high throughput, high fault tolerance"



...or continue using common ETL batch ways like a caveman!

Let's introduce... Apache Flink



Initiated Jan 2015

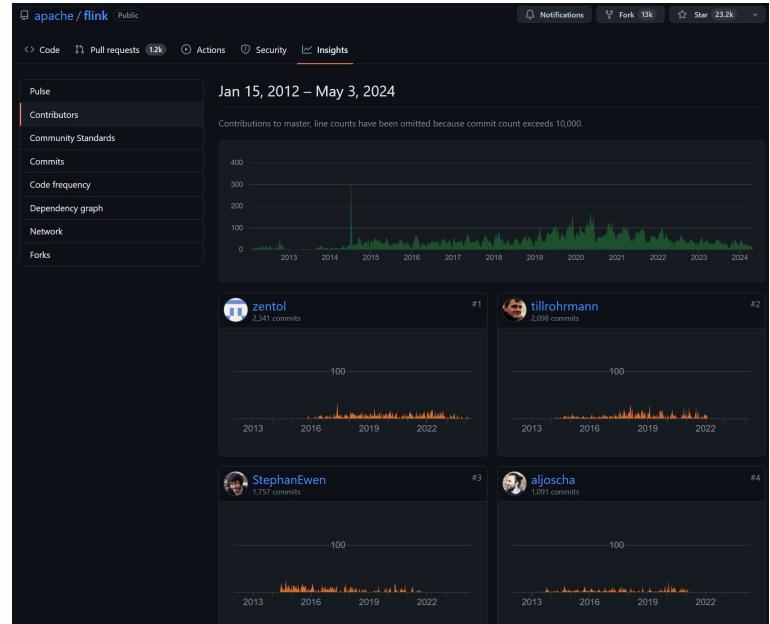
Apache License 2.0

<https://github.com/apache/flink>

Release v1.0.0 (March 2016)

Actual Release v1.19.0 (March 2024)

Java-, Scale-based



Apache Flink

“Application” as a definition of a pipeline

Creating Source- and Sink Table Definitions

Data processing using SQL

Integration of custom code with JAR file

Add new source table

Integrated service: [Alven for Apache Kafka - kafka](#)

Table SQL [?](#)

```
4   name VARCHAR,
5   phoneNumber VARCHAR,
6   address VARCHAR,
7   pizzas ARRAY<ROW(pizzaName VARCHAR, additionalToppings ARRAY <VARCHAR>>)
8 ) WITH (
9   'connector' = 'kafka',
10  'properties.bootstrap.servers' = '',
11  'scan.startup.mode' = 'earliest-offset',
12  'topic' = 'pizza_orders',
13  'value.format' = 'json'
```

▶ Run a sample query to see how the data is being pulled from the data source

	Output	Exceptions				
	id	shop	name	phoneNumber	address	pizzas
0	"Luigis Pizza"	"Jason Brown"	"(510)290-7469"	"2701 Samuel Summit Suite 938 Ryanbury, MH 78954"	▶ [...]	5 items
1	"Ill Make You a Pizza You Cant Refuse"	"Edward Liu"	"(661)154-9335x314"	"561 Lester Points Apt. 140 East Katelynland, NC 75965"	▶ [...]	9 items

[Cancel](#) [Add table](#)

One cloud data platform

STREAM

STORE

SERVE

Event streaming



Aiven for Apache Kafka® and Kafka® Connect

Event stream processing



Aiven for Apache Flink®

Relational databases



Aiven for AlloyDB Omni



Aiven for PostgreSQL®



Aiven for MySQL

Key-value database



Aiven for Valkey



Aiven for Dragonfly®

Data warehouse



Aiven for ClickHouse®

Time series database



Aiven for Metrics

Search engine



Aiven for OpenSearch®

Data visualization



Aiven for Grafana®

UNIFIED PLATFORM

Host



Bring your own cloud

Deploy



Terraform



Kubernetes



REST API



Aiven CLI



Aiven Console

Integrate



Datadog



Prometheus



AWS CloudWatch



GCP Monitoring



MongoDB



AWS S3



GCP BigQuery



Couchbase



Snowflake



Splunk



Sumologic



Debezium



GCP Pub/Sub

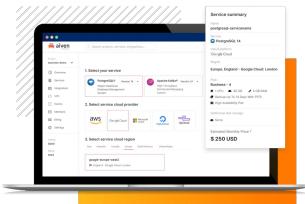
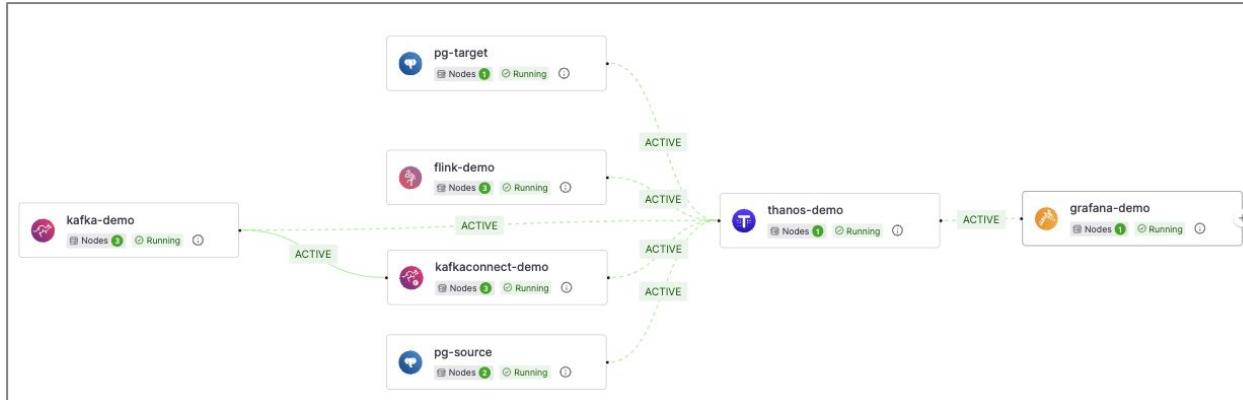


GCP Storage

Fast, easy, reliable... → PaaS

Easy start, challenges in real environments

Managed service recommended (no ad, but still... ;-))



Example – Be lazy using Terraform!

Create/prepare/clone from Github

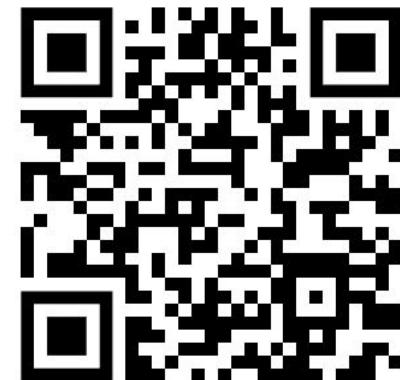
https://github.com/dkrautschick/pg_kafka_cdc

Adjust files (at least the var* files)

provider.tf
var-values.tfvars
services.tf
variables.tf

Run

```
% terraform init  
% terraform plan -var-file=var-values.tfvars  
% terraform apply -var-file=var-values.tfvars
```

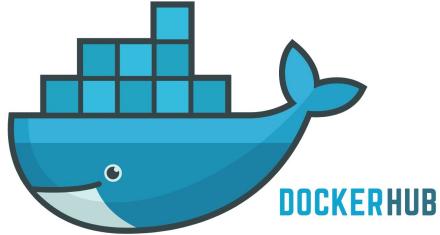


About Containers...

Recommended Docker Repositories

<https://hub.docker.com/u/debezium>

<https://hub.docker.com/r/apache/kafka>



Use Cases and Conclusion

Endless migration scenarios

Very powerful data pipeline stack

Enhanced replication scenarios

Flexibility, Scalability

Materialized views at source

Database migration capabilities are endless

Denormalize at target

Steep learning curve

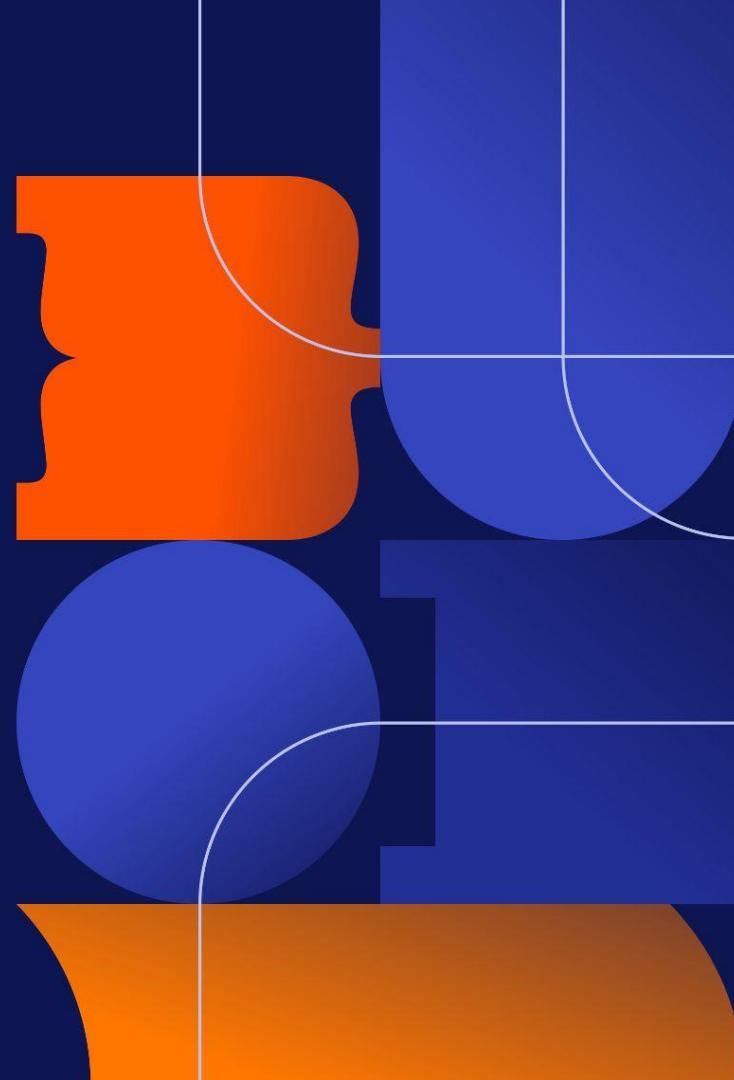
Open Source

Get rid of old habits, test it yourself, use the cloud offerings!



Questions?

Please do ask or
catch me outside!





Your Trusted Data & AI Platform

Streaming | Database Optimization |
Analytics | Search | Data Warehousing |
In-Memory Caching



Customers

okta



snyk

DOORDASH

priceline®

fiverr.

Norauto

DECATHLON

GTL

ACTIVISION | BLIZZARD®

MIRAKL

GOV.UK

goto financial

spare

Schibsted

TOYOTA

paf

CONRAD

adeo

ometria

WÄRTSILÄ