



# Why should Database people care about Kafka and Debezium?

Dirk Krautschick

[PGConf.EU](#) Riga 23.10.2025



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

Dirk Krautschick

[PGConf.EU](#) Riga 23.10.2025

# #whoami

**Dirk Krautschick**  
**Senior Solution Architect**

with Aiven since Nov 2023



19 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

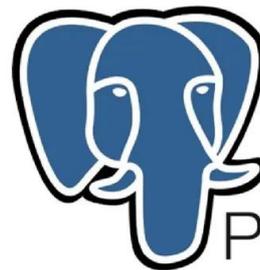
Feb 2024 Cologne, May 2024 Aachen, Oct 2024 Essen,

Feb 2025 Paderborn, Jun 2025 M'Gladbach,...

**Oct 2025 Dortmund (Apps Associates)**

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!!! ;-)***



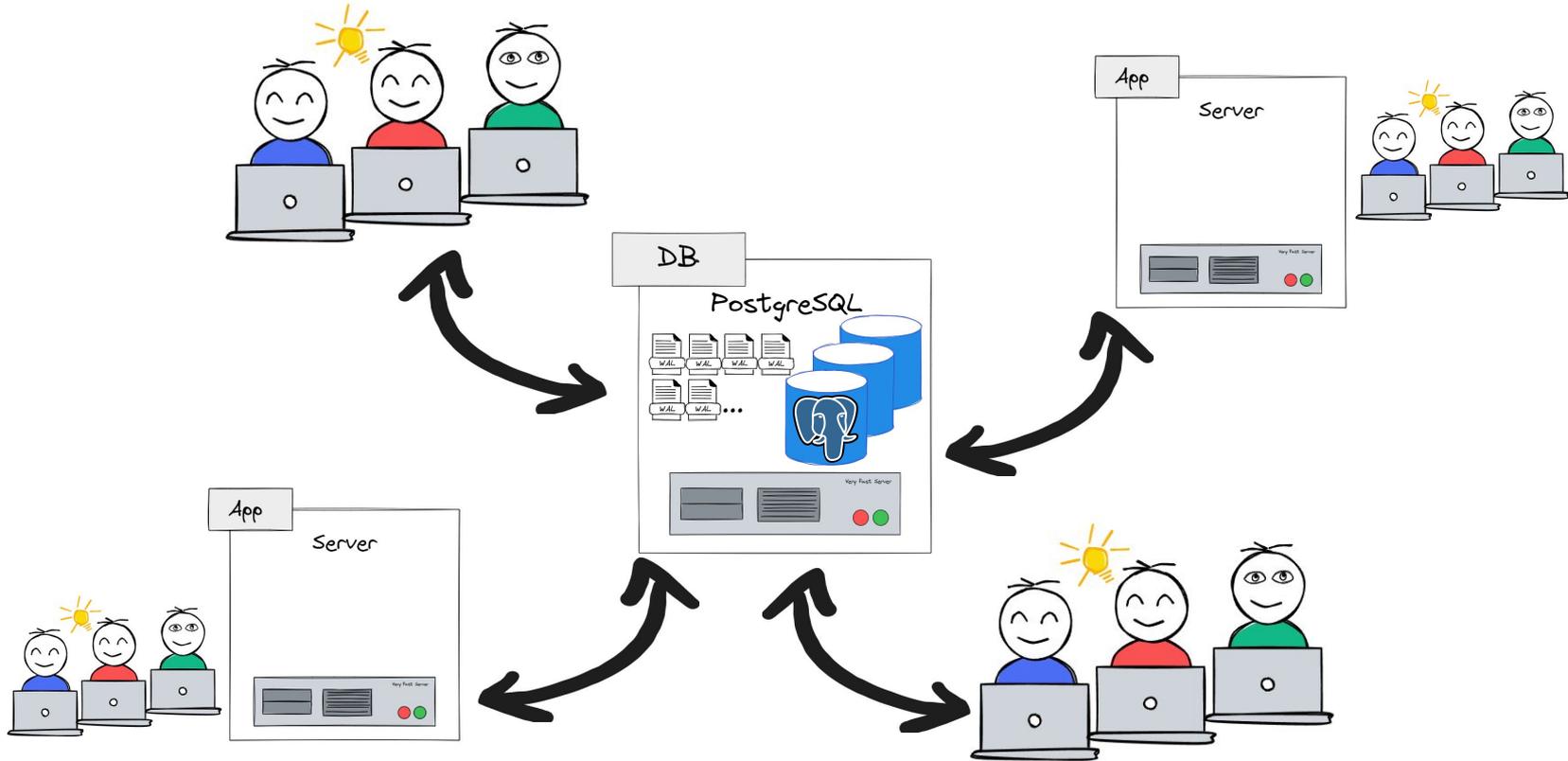
# Motivation.. once upon a time..

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

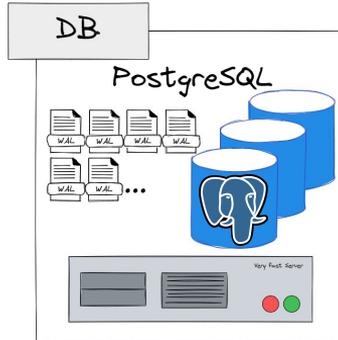


my server,  
my databases,  
my responsibilities!!!

# Motivation.. once upon a time..

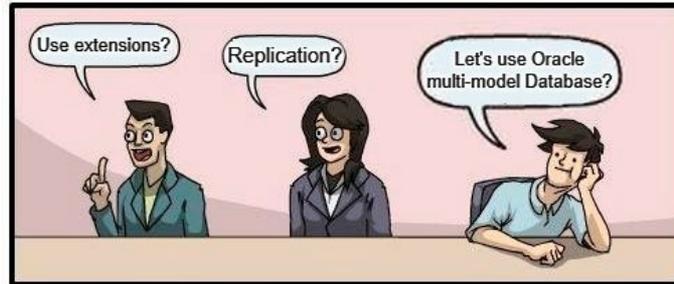


# Motivation.. but today..



DATA MIGRATION

# No worries, we'll stick with Databases



# Motivation... but today...

Not just only ordinary clients/applications anymore

Role of DBAs changing

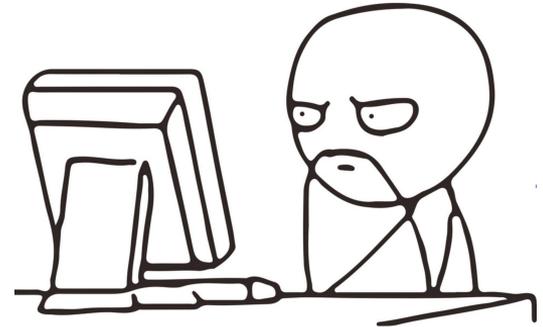
- Cloud/DBaaS vs. on premises

- Architectural and scaling challenges

- Different database types for different use cases

Real time data processing and heterogeneity

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



# First idea... 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 !!!11eleven"* ...of course :-)

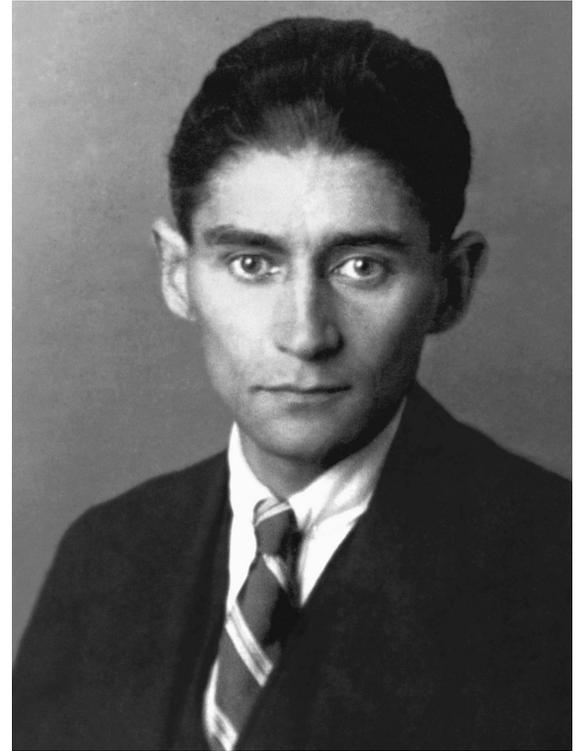


→ **"Distributes event streaming"**

# 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

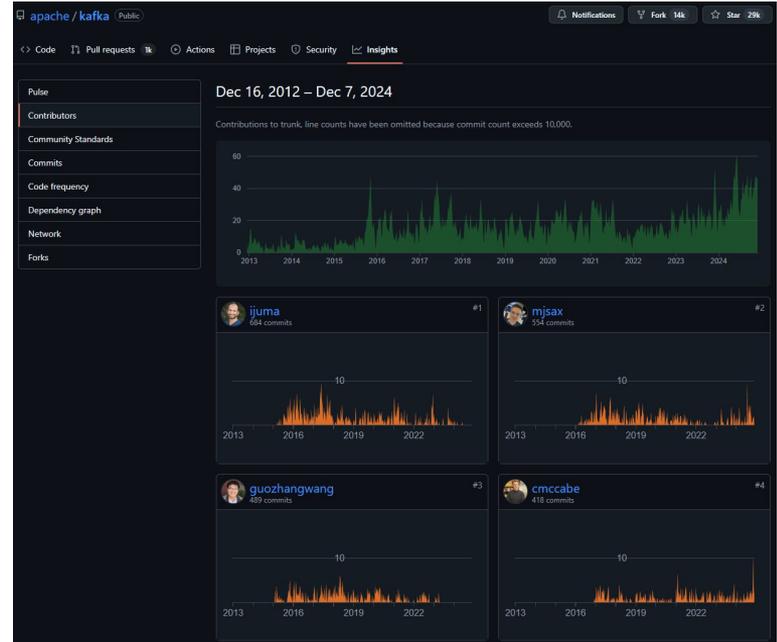
Apache License 2.0

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

Release v1.0.0 (Nov 2017)

Actual Release v4.1.0 (Sep 2025)

Java- and Scala-based



# Let's introduce... Apache Kafka



## Apache Kafka

Initiated by LinkedIn (**Nov 2010**)

Apache **since 2012**

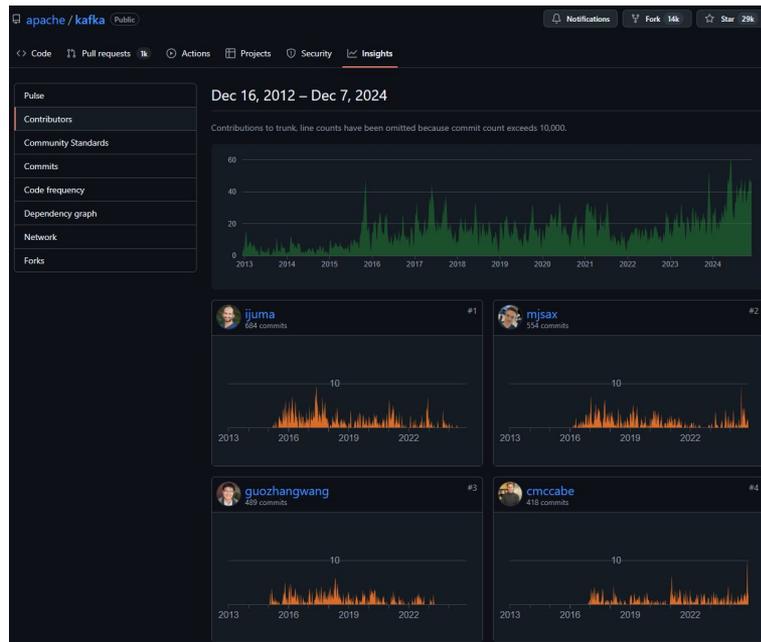
Apache License 2.0

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

Release v1.0.0 (Nov 2017)

Actual Release v4.1.0 (Sep 2025)

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

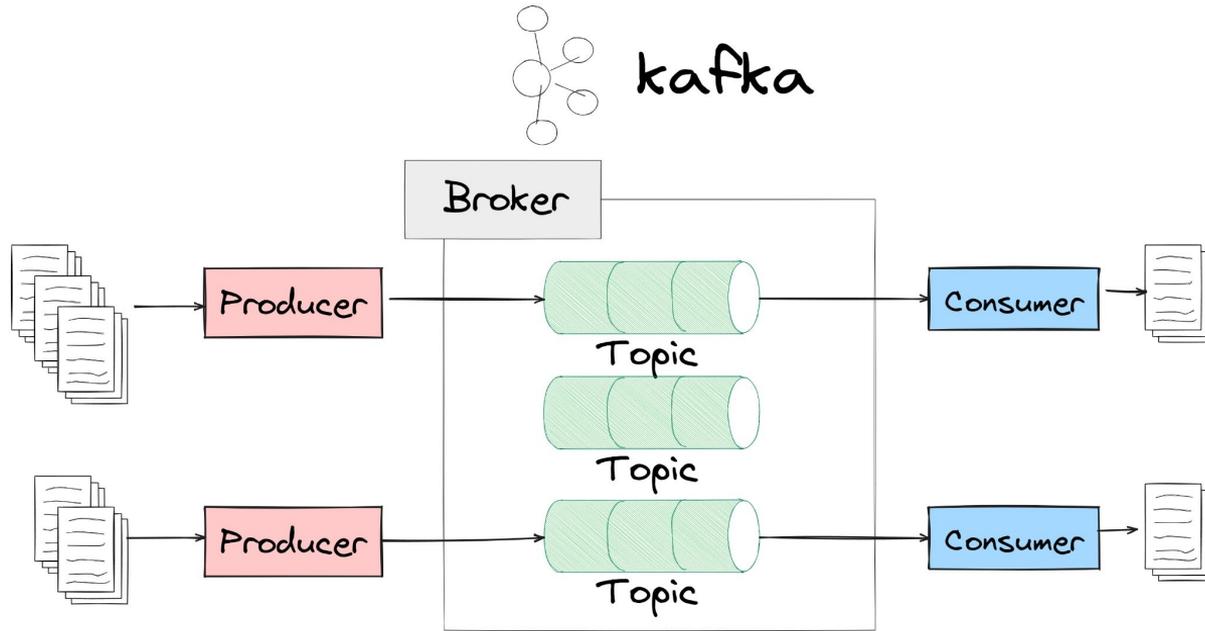
Zookeeper deprecated soon / KRaft default with 4.0

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 displays the Aiven console interface for a Kafka cluster. The top navigation bar includes 'Home', 'Projects', 'Billing', 'Support', and 'Admin'. The user is logged in as 'My Organization'. The left sidebar shows navigation options: Overview, Integrations, Network, Metrics, Logs, Users, ACL, Topics, Backups, Connectors, Schemas, and Quotas.

The main content area shows the 'kafka-dirk' project details, including 'Apache Kafka 3.5.1', 'EOL : 2024-07-31 : OK', 'Running' status, and 'Nodes 3'. The breadcrumb path is 'My Organization / dirk-aven / kafka-dirk / Topics / dirk Messages'. There are 'Fetch messages' and 'Produce message' buttons.

Filters for the message list are: PARTITION: Show all, OFFSET: 0, TIMEOUT (S): 3, MAX BYTES: 1048576, and FORMAT: binary. A 'Decode from base64' toggle is checked.

The message list shows 'Messages 1-10 of 25 · Page 1'. The first message is expanded to show its details:

Meta	Key	Value
OFFSET: +24		<pre>{ "key": "89c630d6-c8c5-43d4-80b0-0c7f0b93ae5f", "sensorDataX": 47, "sensorDataY": 54, "timestamp": "2023-10-20" }</pre>
PARTITION: 0		<pre>{   key: "89c630d6-c8c5-43d4-80b0-0c7f0b93ae5f",   sensorDataX: 47,   sensorDataY: 54,   timestamp: "2023-10-20" }</pre>
OFFSET: +23		<pre>{ "key": "17f0dfdf-4af3-4430-b236-c4ccdb0e8904", "sensorDataX": 85, "sensorDataY": 0, "timestamp": "2023-10-20" }</pre>

# 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/4.1.0/kafka\_2.13-4.1.0.tgz
```

```
# tar -xzf kafka_2.13-4.1.0.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/server.properties --standalone  
# bin/kafka-server-start.sh config/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
```

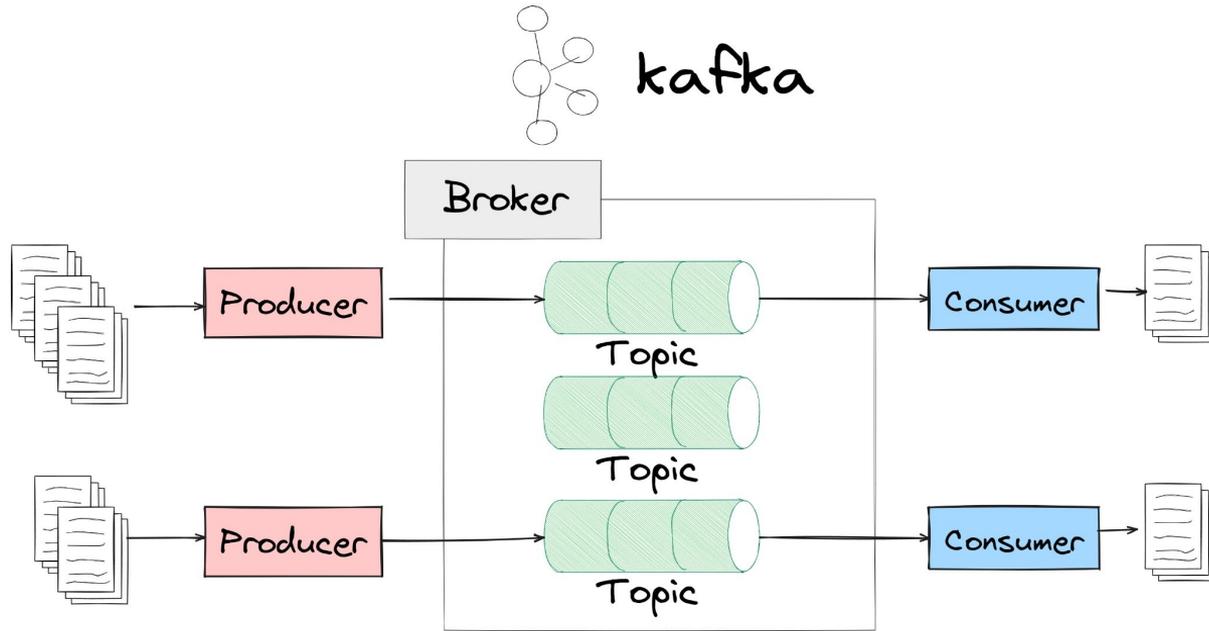
## Check, produce and consume events

```
# 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
```

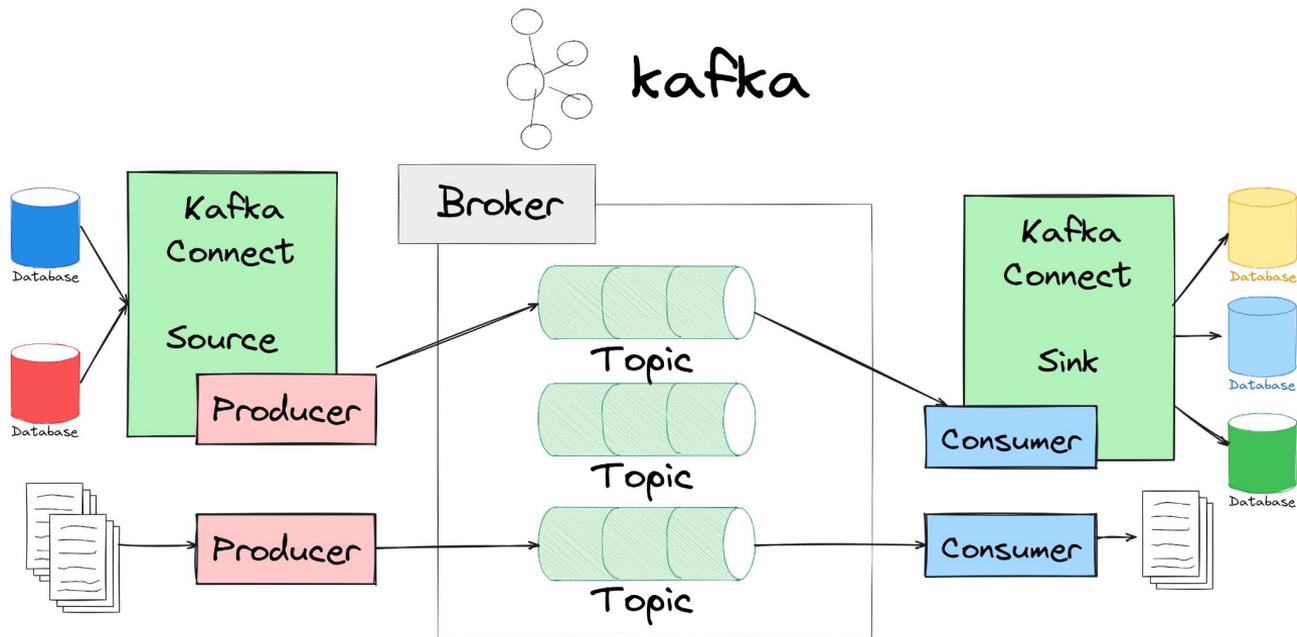
# Kafka – An easy quick start..

Demo

# Nice, but what about the databases?



# Nice, but what about the databases?



# 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

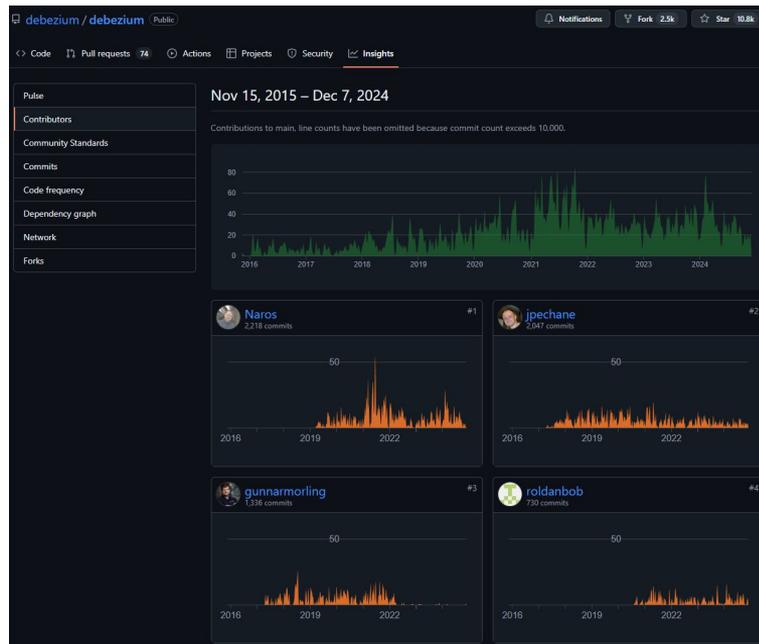
Apache License 2.0

<https://github.com/debezium>

Release v1.0.0 (Dec 2019)

Actual Release v3.3.1 (Oct 2025)

Java-based



# Kafka Connect – ...with Debezium

Logical Decoding API for PostgreSQL

Sink Connector for JDBC targets

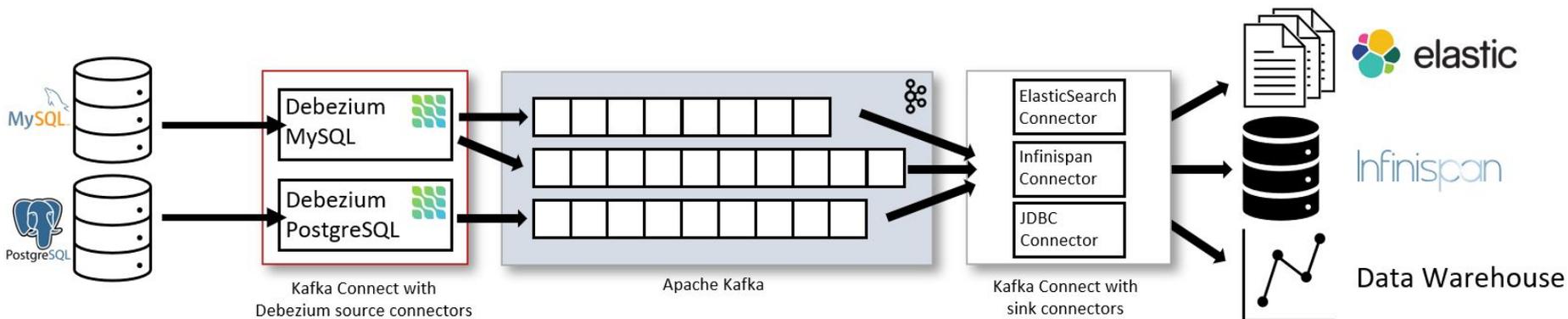
REST API

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/maven2/debezium/debezium-connector-postgres-3.3.1.Final-plugin.tar.gz  
  
# mv ./debezium-connector-postgres-3.3.1.Final-plugin.tar.gz <e.g. YOUR KAFKA CLUSTER>  
  
# cd <e.g. YOUR KAFKA CLUSTER>  
  
# tar -xzf debezium-connector-postgres-3.3.1.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;
```

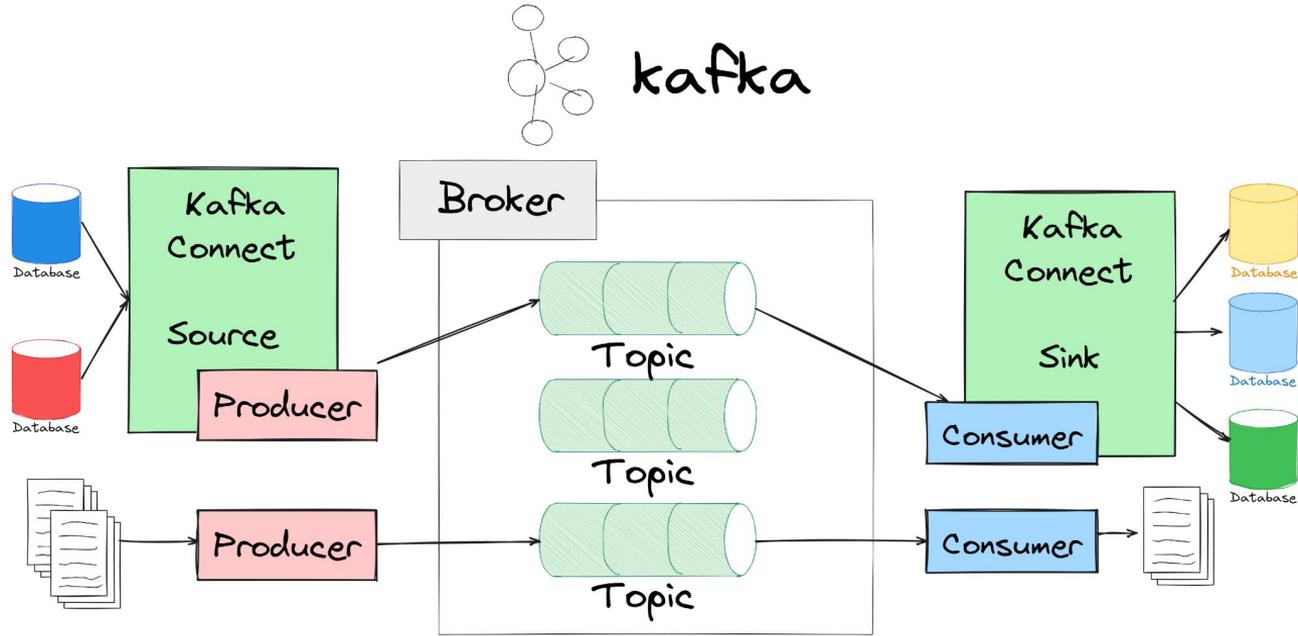
# Kafka Connect – Example test

```
# 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":"version"}, {"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 ...

**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!**



# What's about Inkless/Diskless?

**Streaming  
is not cloud  
native!**

## The Big Problem (s)



**Kafka in 2025** uses aging design which is expensive and not cloud-native.



Kafka shifts to thin clients **increasing the cost** to run as a service.



Streaming data is **expensive** - 10x more than storing data on S3 due to inter-az and local disks usage.



Business users **do not work** with streaming data, streaming is limited to Operations.

# What's about Inkless/Diskless?



# One cloud data platform

## Unified Platform

### Streaming

-  Aiven Inkless
-  Aiven for Apache Kafka®
-  Aiven for Apache Kafka® Connect
-  Aiven for Apache Kafka® MirrorMaker 2
-  Aiven for Apache Flink®
-  Karapace
-  Klaw

### Databases

-  Aiven for PostgreSQL®
-  Aiven for MySQL
-  Aiven for Valkey
-  Aiven for Dragonfly
-  Aiven for ClickHouse®
-  Aiven for OpenSearch®
-  Aiven for Metrics
-  Aiven for Grafana®

### Deploy



### Tooling



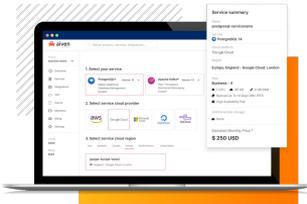
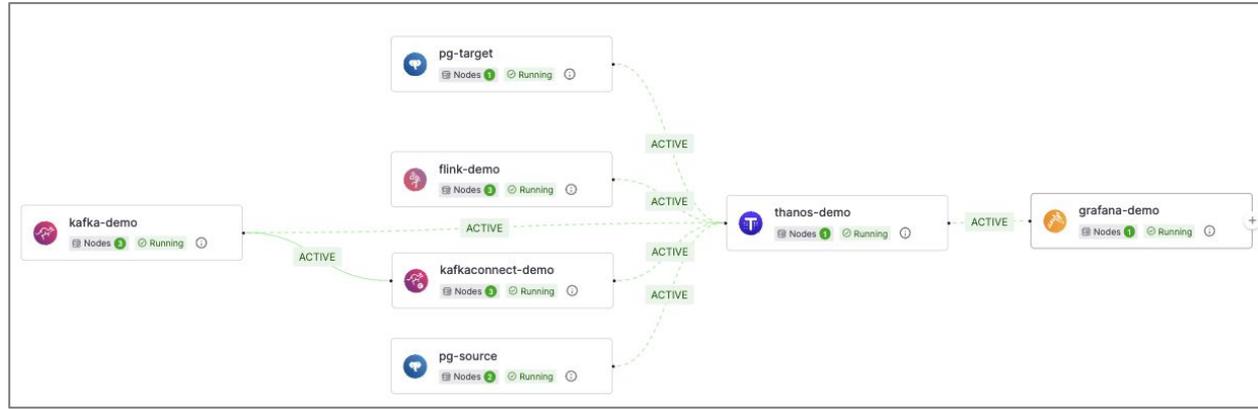
### Third-party integration



# 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](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
```

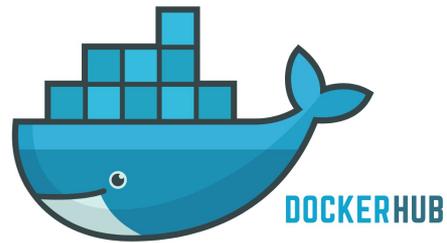


# About Containers...

Recommended Docker Repositories

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

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



# Use Cases and Conclusion

Endless migration scenarios

Enhanced replication scenarios

Materialized views at source

Denormalize at target

Very powerful data pipeline stack

Flexibility, Scalability

Database migration capabilities are endless

Steep learning curve

Open Source

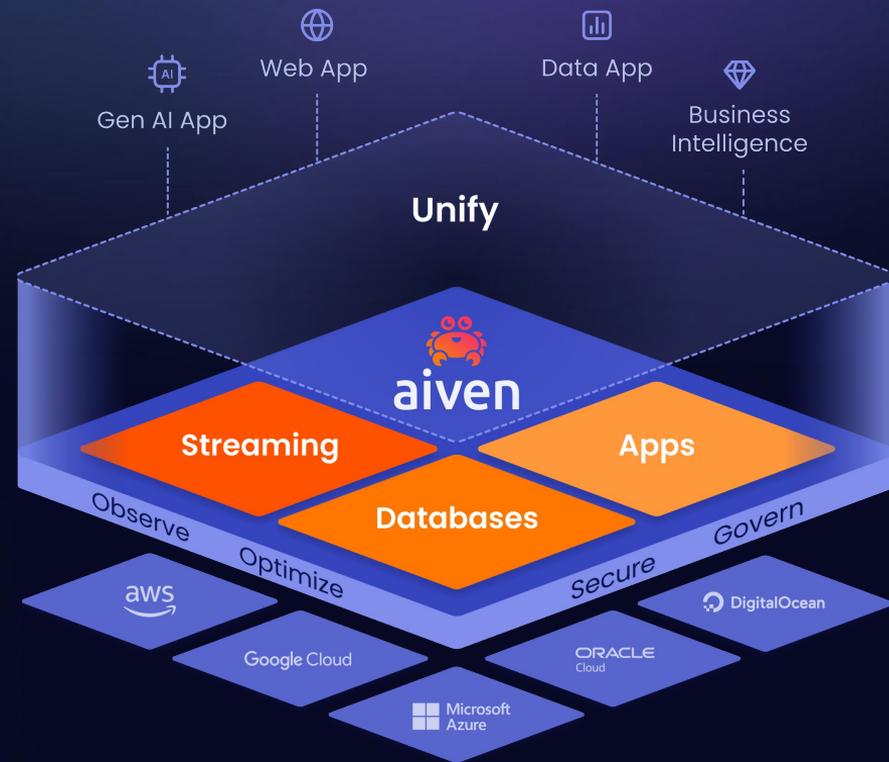
Get rid of old habits, test it yourself, use the cloud offerings!



**Thank you**

# Your AI-ready Open Source Data Platform

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





# Your AI-ready Open Source Data Platform

The screenshot displays the Aiven console interface. At the top left is the Aiven logo and navigation links for Home, Projects, Billing, and Admin. The top right shows the user's name 'Aiven', a grid icon, and notification, help, and user profile icons. The main content area is divided into two sections: 'Services' and 'Platform status'. The 'Services' section contains a table with columns for Service, Nodes, Plan, and Cloud. The 'Platform status' section shows a green checkmark and the text 'All systems operational' with a 'Subscribe' button. The 'Product updates' section has an 'RSS Feed' button and a list of update items.

Service	Nodes	Plan	Cloud
Aiven for Apache Kafka®			
Aiven for Apache Flink®			
Aiven for AlloyDB Omni			
Aiven for OpenSearch®			
Aiven for PostgreSQL®			
Aiven for MySQL			

**Platform status** Subscribe

All systems operational  
No incidents reported.

**Product updates** RSS Feed

# One cloud data platform

## Unified Platform

### Streaming

-  Aiven Inkless
-  Aiven for Apache Kafka®
-  Aiven for Apache Kafka® Connect
-  Aiven for Apache Kafka® MirrorMaker 2
-  Aiven for Apache Flink®
-  Karapace
-  Klaw

### Databases

-  Aiven for PostgreSQL®
-  Aiven for MySQL
-  Aiven for Valkey
-  Aiven for Dragonfly
-  Aiven for ClickHouse®
-  Aiven for OpenSearch®
-  Aiven for Metrics
-  Aiven for Grafana®

### Deploy



### Tooling



### Third-party integration

