

# Creating an Apache Kafka® Streaming Data Pipeline



Danica Fine  
<https://linktr.ee/thedanicafine>



Amanda Gilbert  
<https://github.com/amanda010792>

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00	LUNCH	

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00	LUNCH	

# What's a data pipeline?

# data pipeline

**noun** \ 'dā-tə 'pīp-līn \

A process (or series of processes) that moves data from a source to a target.

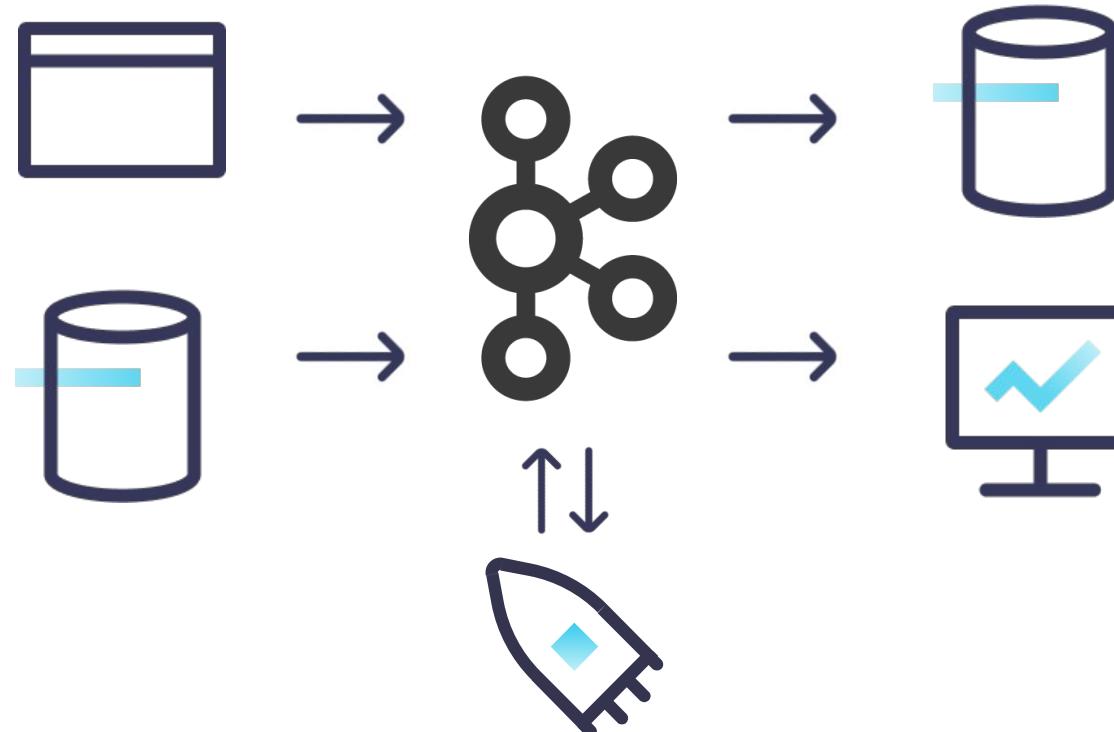
# What's a *streaming* data pipeline?

# streaming data pipeline

**noun** \ 'strē-mīn 'dā-tə 'pīp-,līn \

A process (or series of processes) that moves data from a source to a target as the data happens, *in a stream*.

# Streaming Data Pipeline



# Streaming Data Pipelines

Who cares?

# Benefits of Stream Processing

- Event-driven
- Decouples system
  - Agile development
  - Faster time-to-market
- Increase resilience



# Stream Processing

What are your use cases?

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00	LUNCH	



# Kafka? What's *that*?

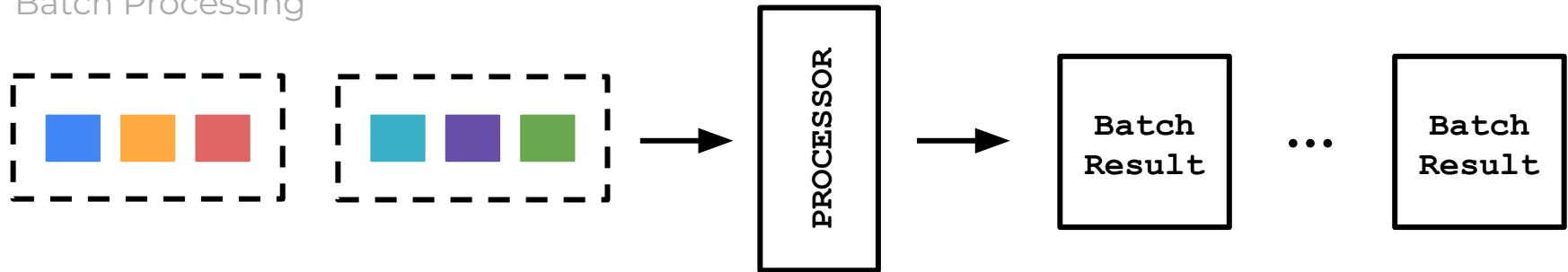
A distributed event streaming platform.



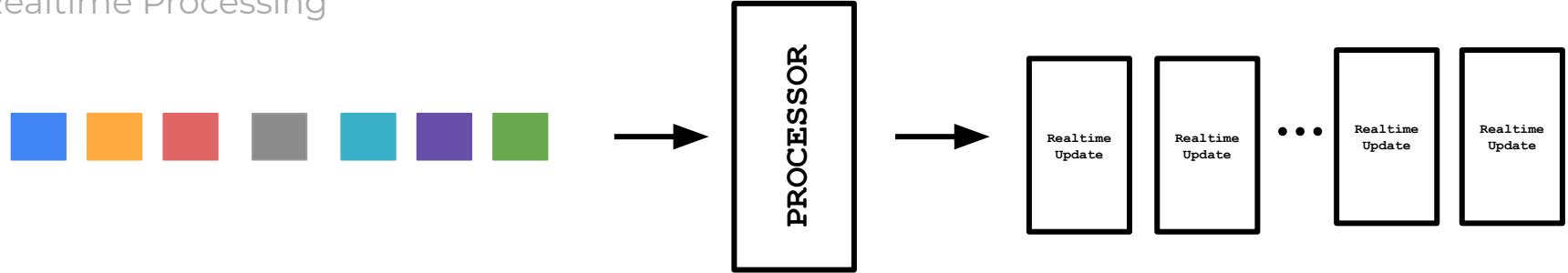
A distributed event ***streaming*** platform.

# Paradigm Shift

Batch Processing



Realtime Processing



# Some key differences

	<b>Batch Data Processing</b>	<b>Real-Time Data Processing</b>	<b>Streaming Data</b>
<b>Hardware</b>	Most storage and processing resources requirement to process large batches of data.	Less storage required to process the current or recent set of data packets. Less computational requirements.	Less storage required to process current data packets. More processing resources required to "stay awake" in order to meet real-time processing guarantees
<b>Performance</b>	Latency could be minutes, hours, or days	Latency needs to be in seconds or milliseconds	Latency must be guaranteed in milliseconds
<b>Data set</b>	Large batches of data	Current data packet or a few of them	Continuous streams of data
<b>Analysis</b>	Complex computation and analysis of a larger time frame	Simple reporting or computation	Simple reporting or computation



A distributed **event** streaming platform.

# Thinking in Events

- Natural way to reason about things
- Indicate that something *has happened*
  - When
  - What/Who
- Immutable pieces of information

# Events

^ Value Header Key

```
1  {
2      "viewtime": 111,
3      "userid": "User_8",
4      "pageid": "Page_54"
5 }
```

# Events

^ Value Header Key

```
1  [
2    {
3      "key": "task.generation",
4      "stringValue": "0"
5    },
6    {
7      "key": "task.id",
8      "stringValue": "0"
9    },
10   {
11     "key": "current.iteration",
12     "stringValue": "11"
13   }
14 ]
```

# Events



# Events

▼ {"viewtime":1191,"userid":"User\_8","pageid":"Page\_56"}

Partition: 0      Offset: 119      Timestamp: 1666370871970

▼ {"viewtime":1181,"userid":"User\_6","pageid":"Page\_28"}

Partition: 0      Offset: 118      Timestamp: 1666370871421

▼ {"viewtime":1171,"userid":"User\_1","pageid":"Page\_15"}

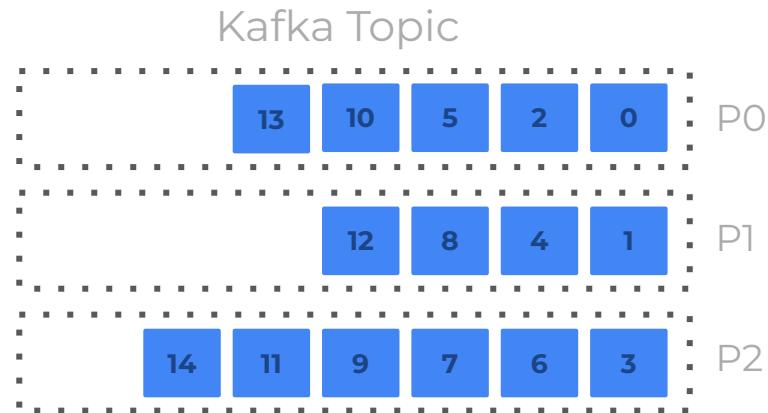
Partition: 0      Offset: 117      Timestamp: 1666370870791



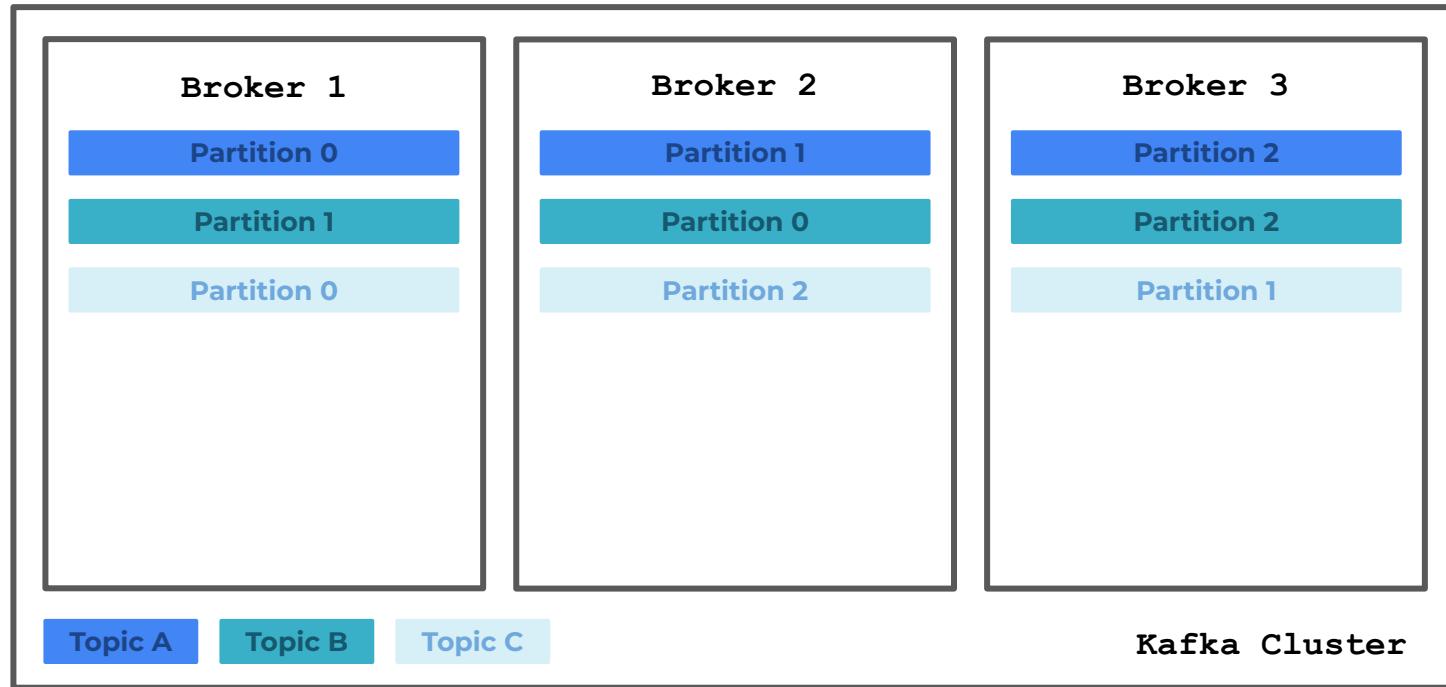
A ***distributed*** event streaming platform.

# Kafka Storage

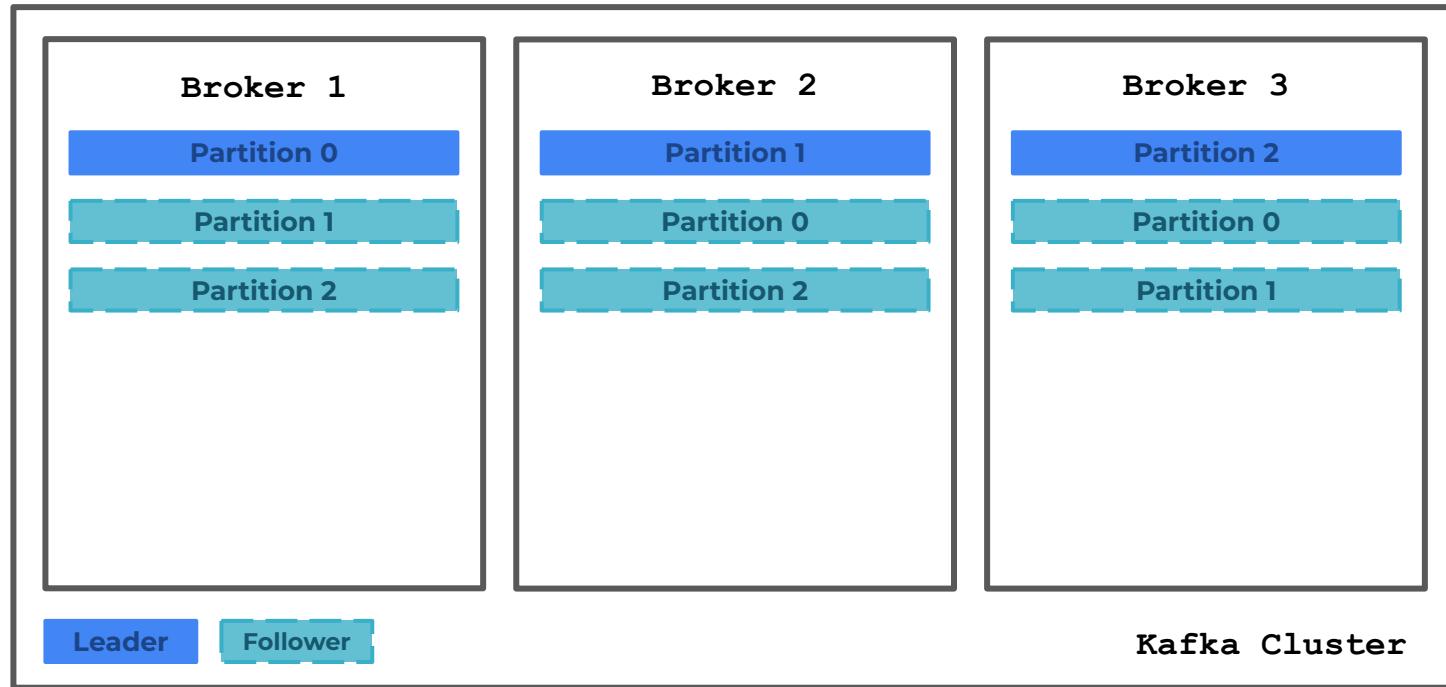
- Topics
  - Basic storage unit
  - Read/WRITE:
    - Producer and consumer clients
    - Completely decoupled
- Partitions
  - Immutable, append-only logs
  - Data is replicated at this level



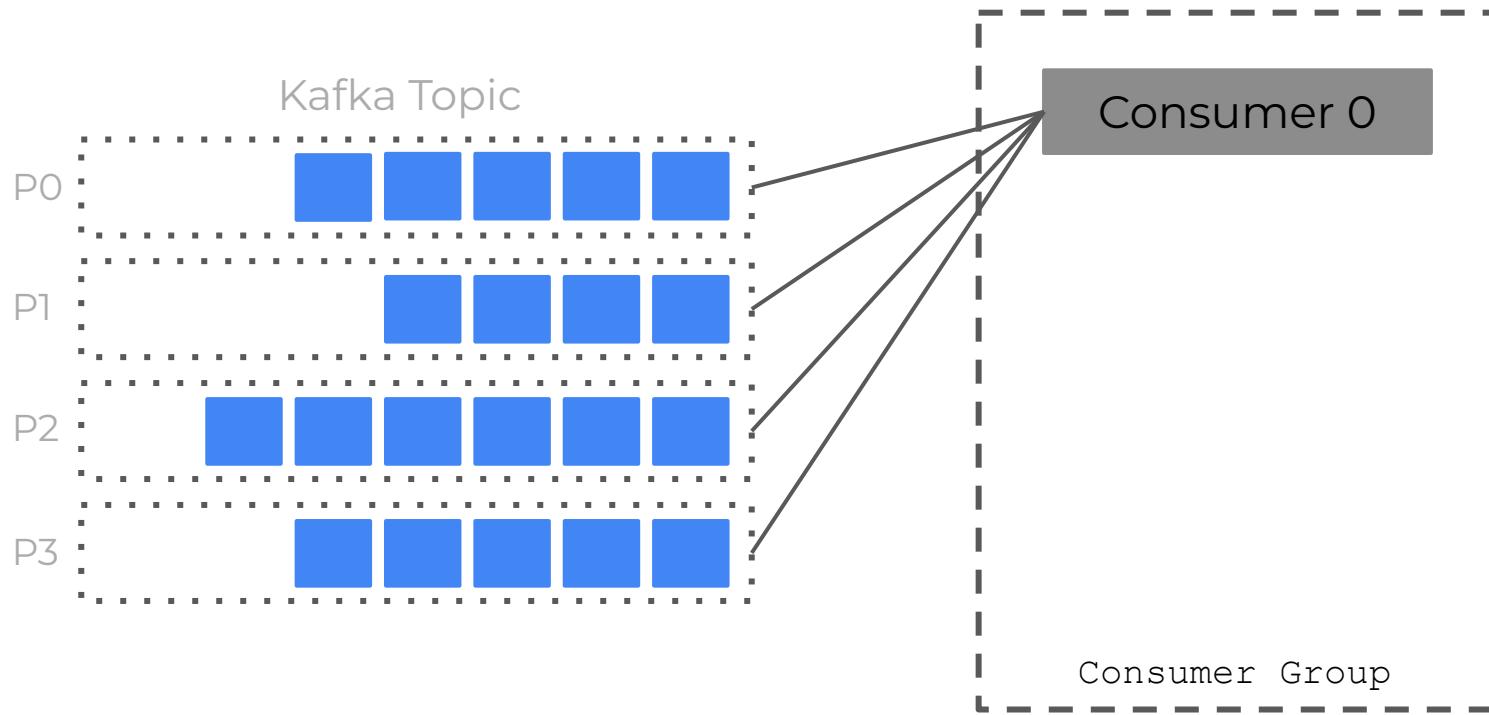
# Kafka Cluster



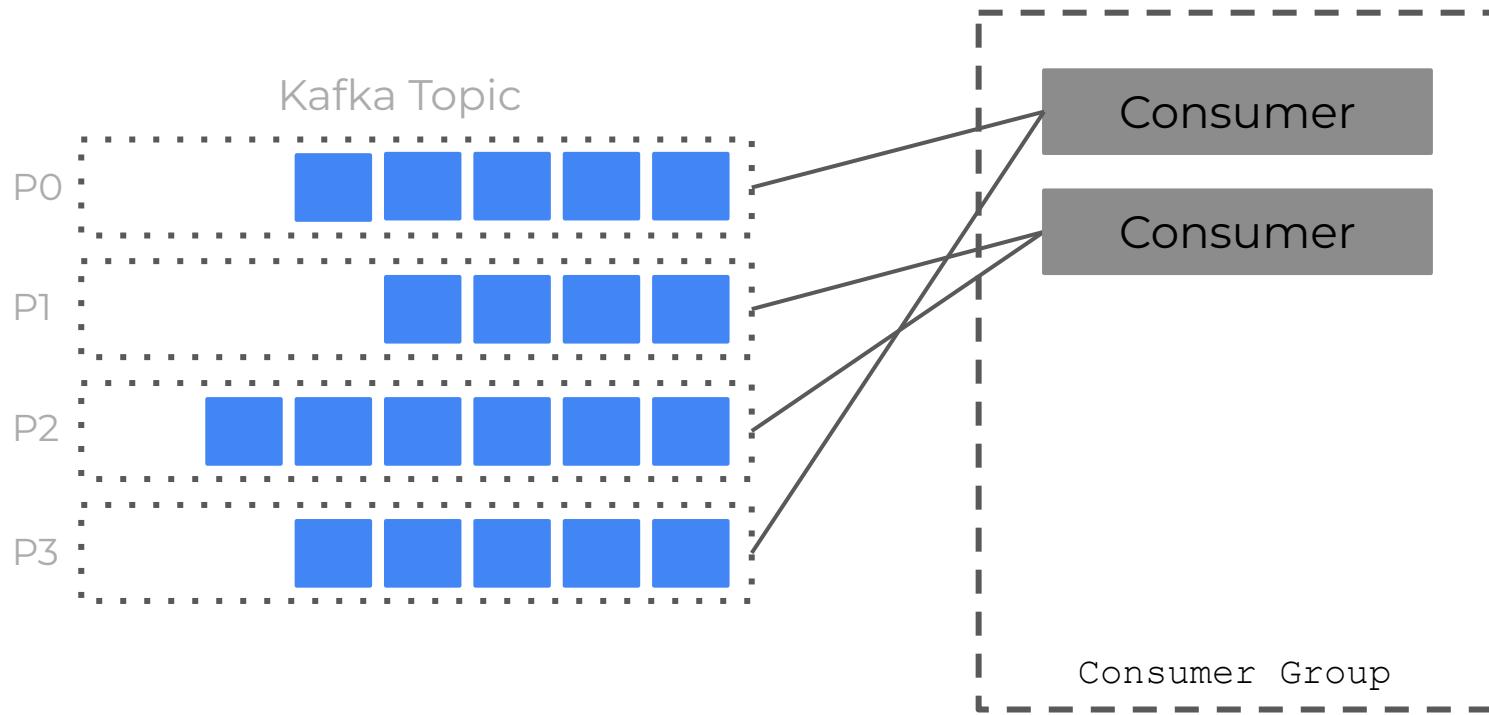
# Kafka Cluster



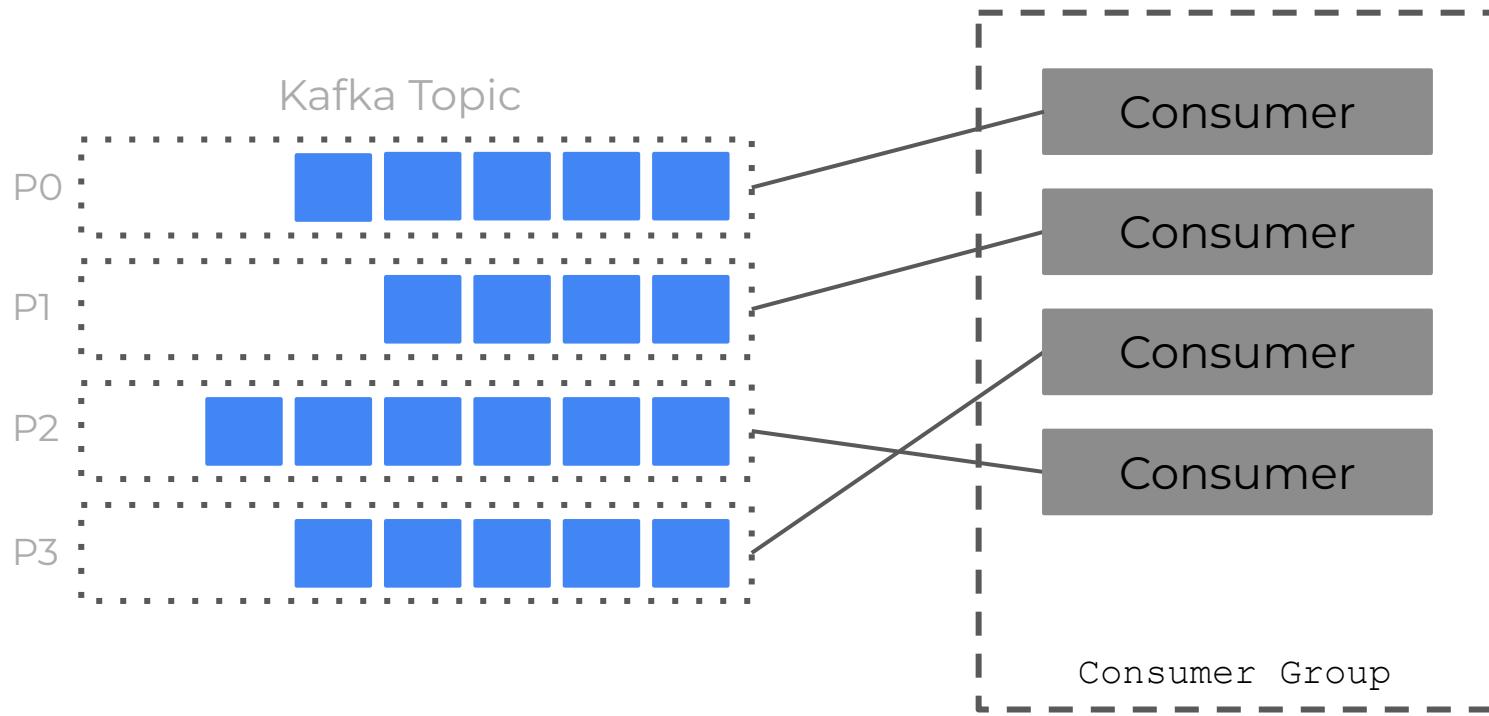
# Consumer Groups



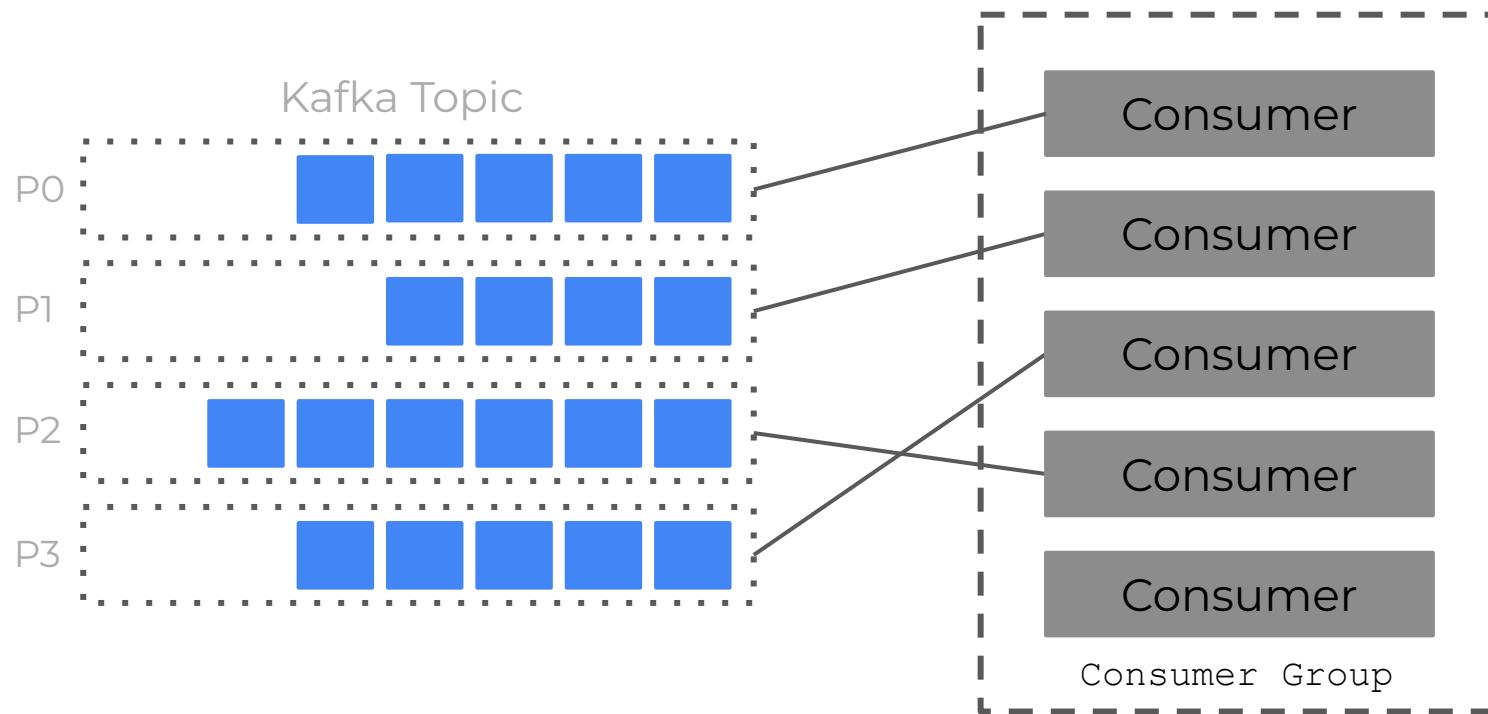
# Consumer Groups



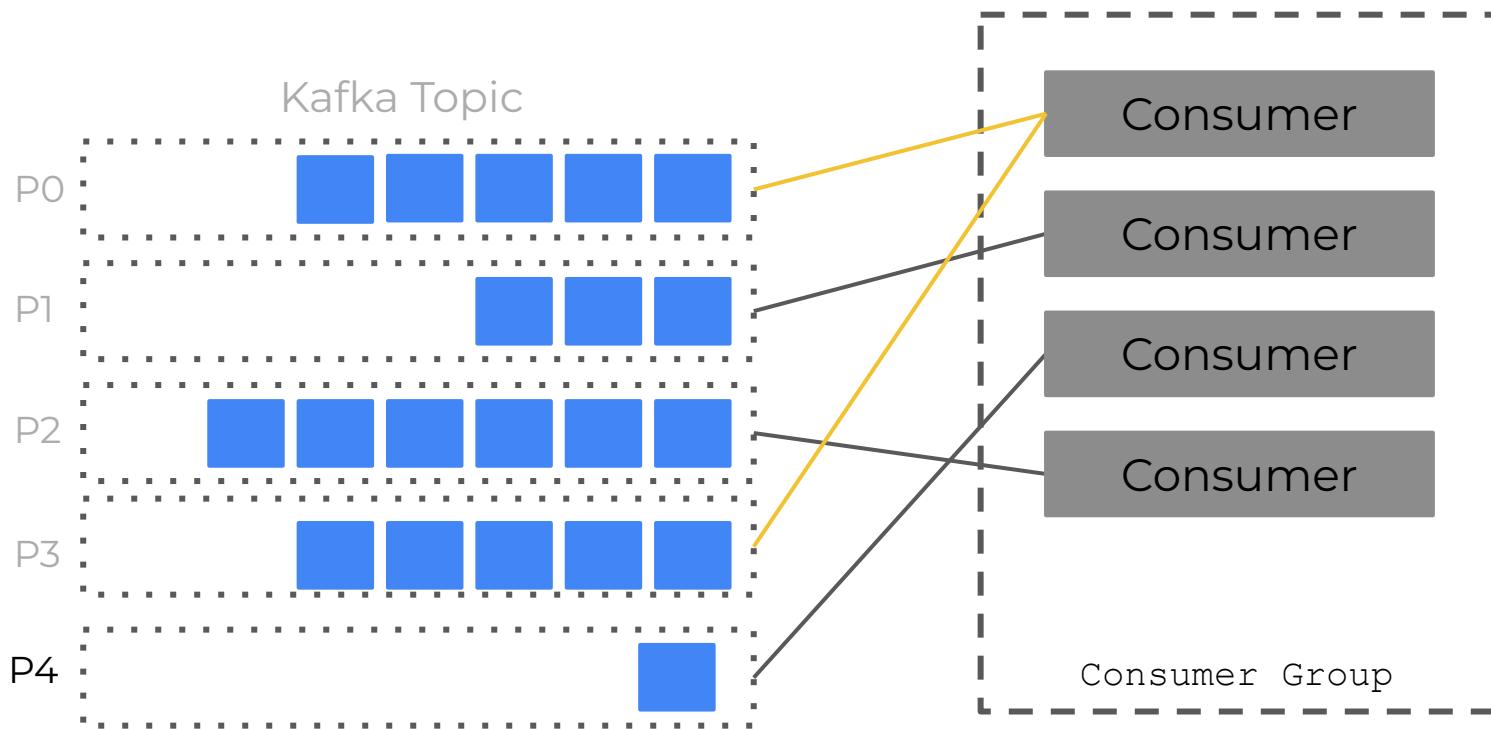
# Consumer Groups



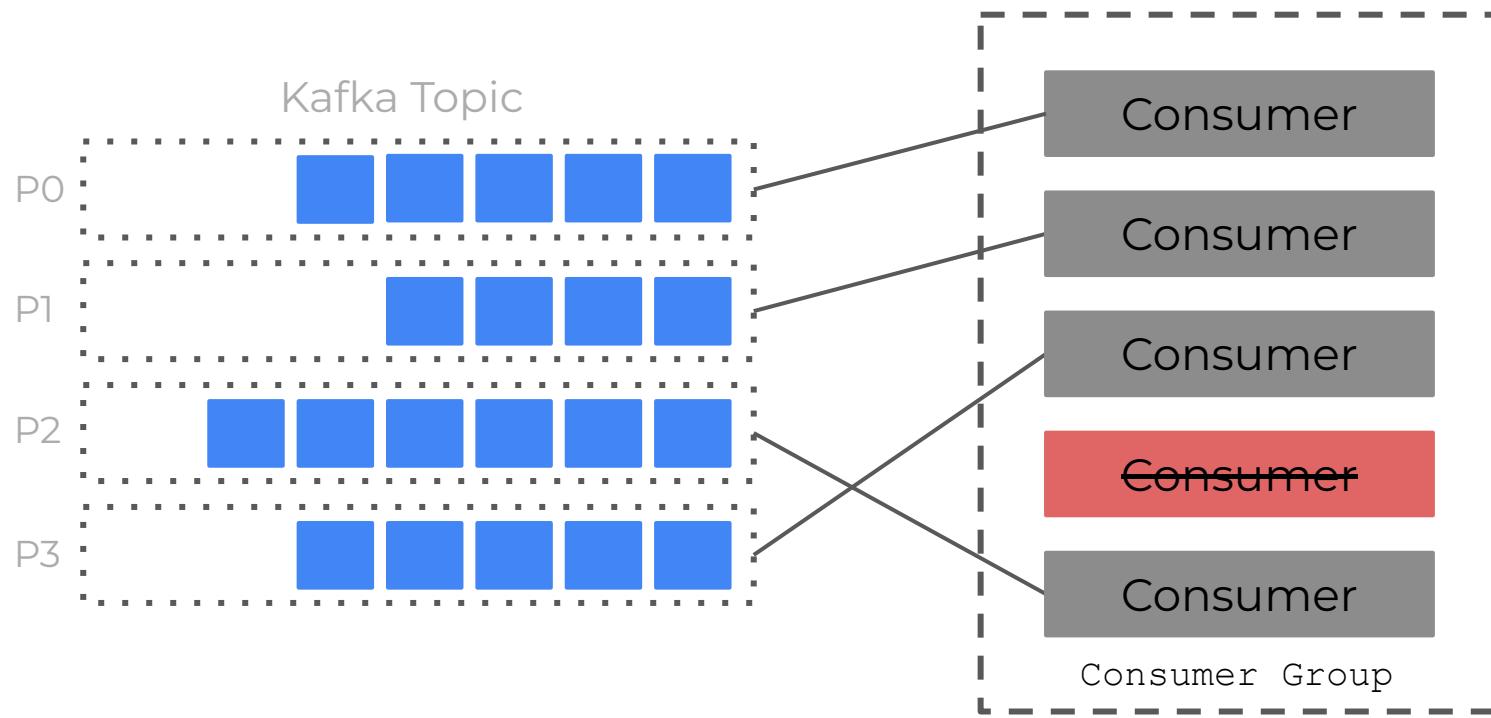
# Consumer Groups



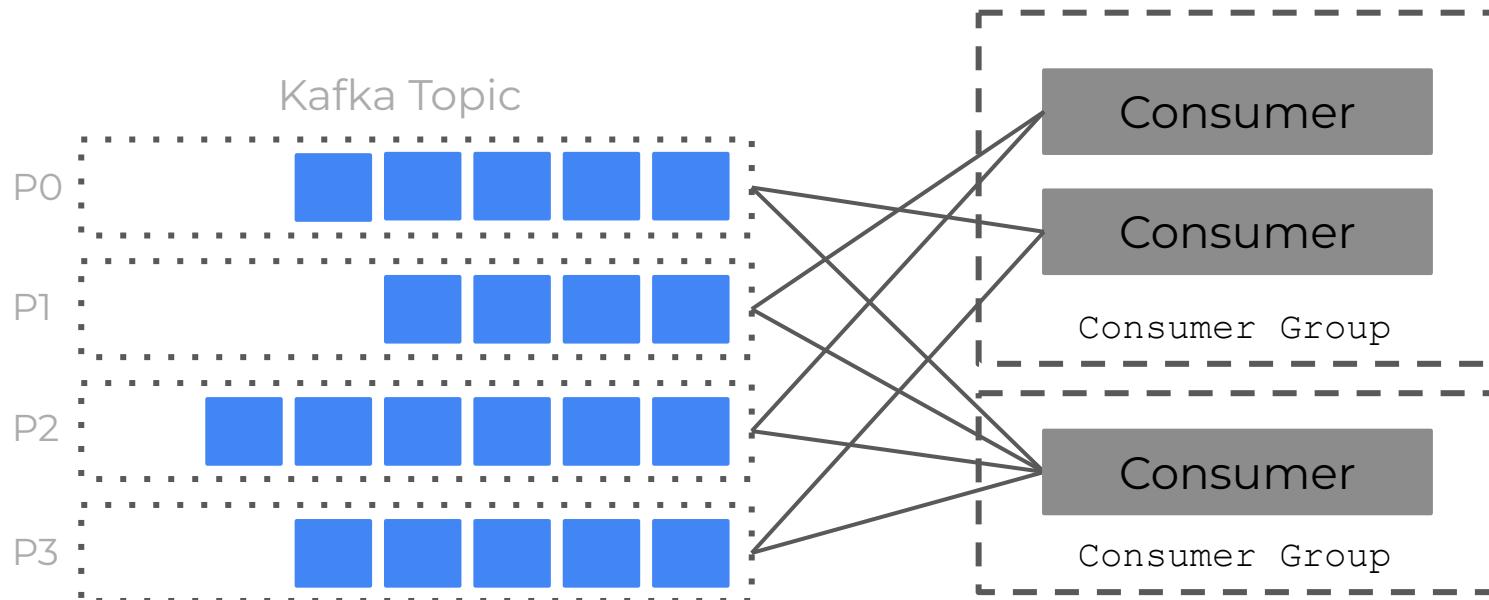
# Consumer Groups



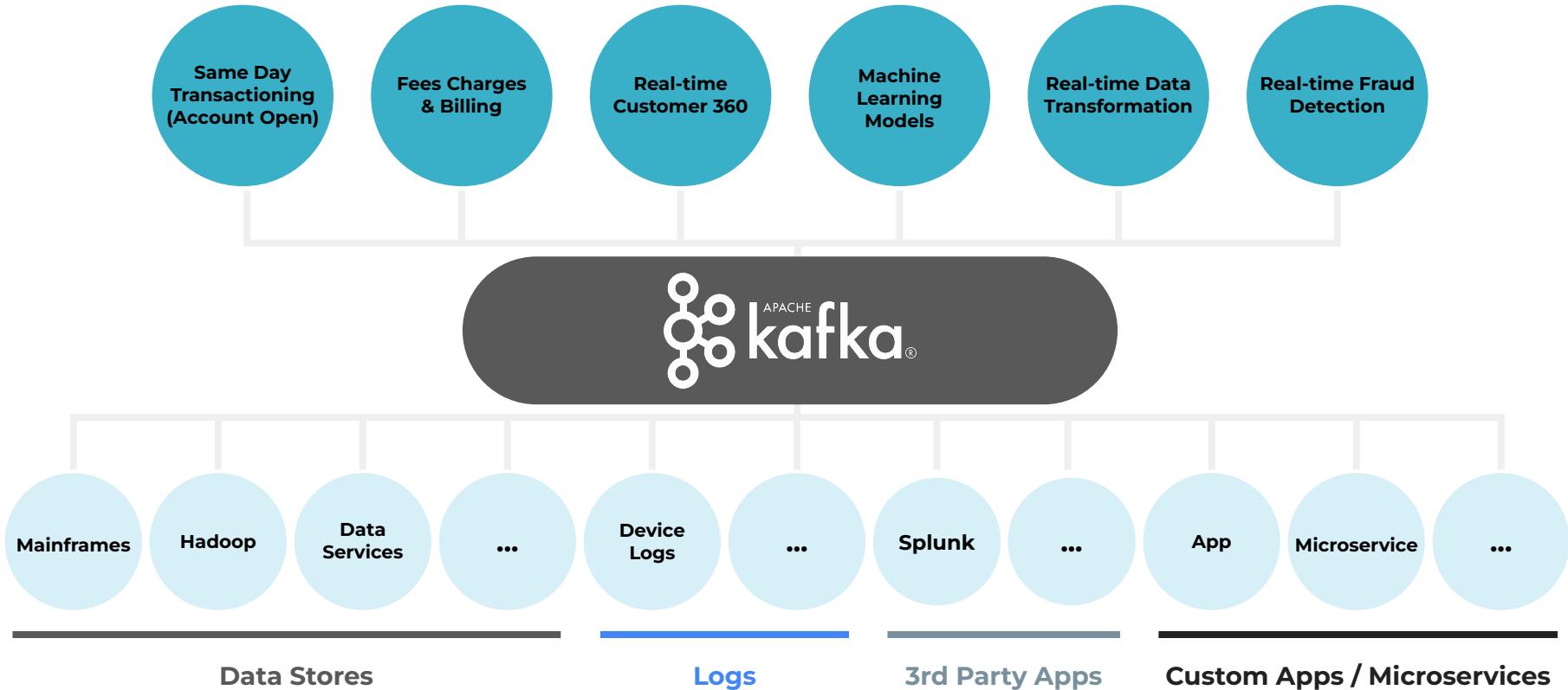
# Consumer Groups



# Consumer Groups



# Unified Fabric for App & Data Modernization



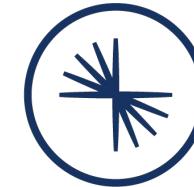


# Confluent? What's *that*?

Kafka, but make it easy.

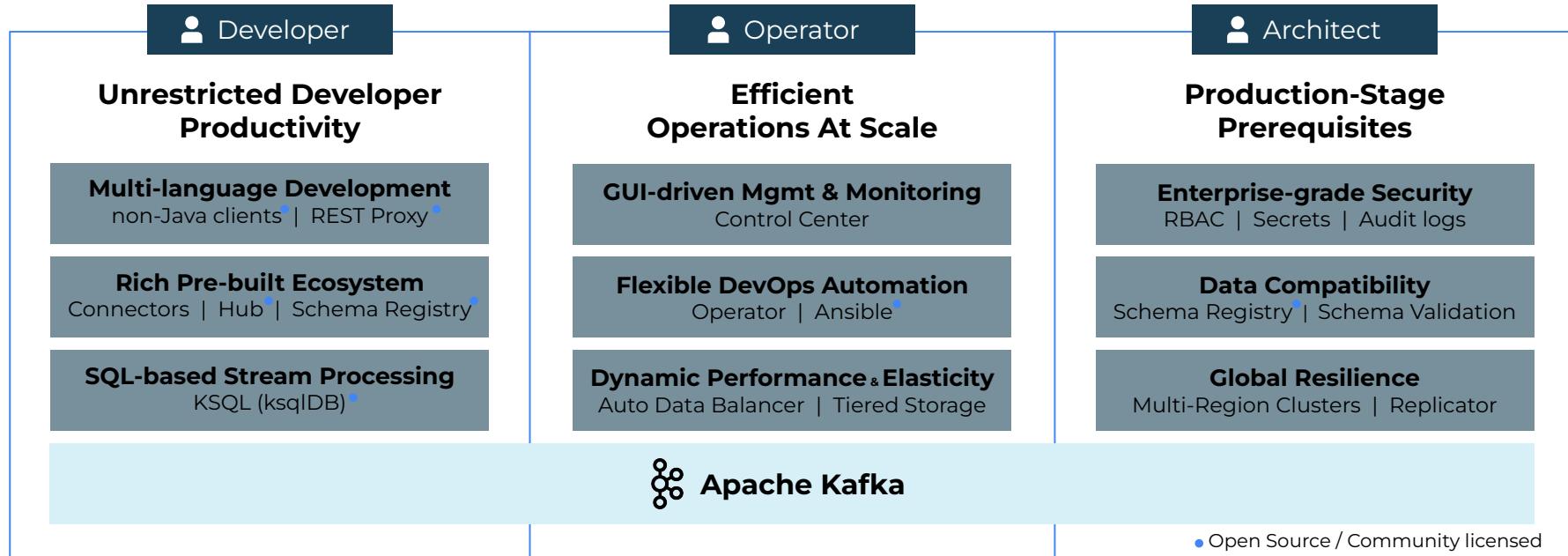
# Confluent

- Fully-managed, cloud-based Kafka
- Auxiliary tools:
  - Kafka Connect
  - ksqlDB
  - Schema management



CONFLUENT

# Confluent



Self Managed Software

**Freedom Of Choice**



Fully Managed Cloud Service



Enterprise Support



Professional Services

**Committer-Led Expertise**



Training



Partners

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30		COFFEE BREAK
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00		LUNCH

# Hands On

Confluent Cloud and Apache Kafka Installation

# Kafka Installation Process



- Python Client Installation  
`pip install confluent-kafka`
- Bundled with librdkafka
- Note: M1 MacBooks ***must*** install from source.

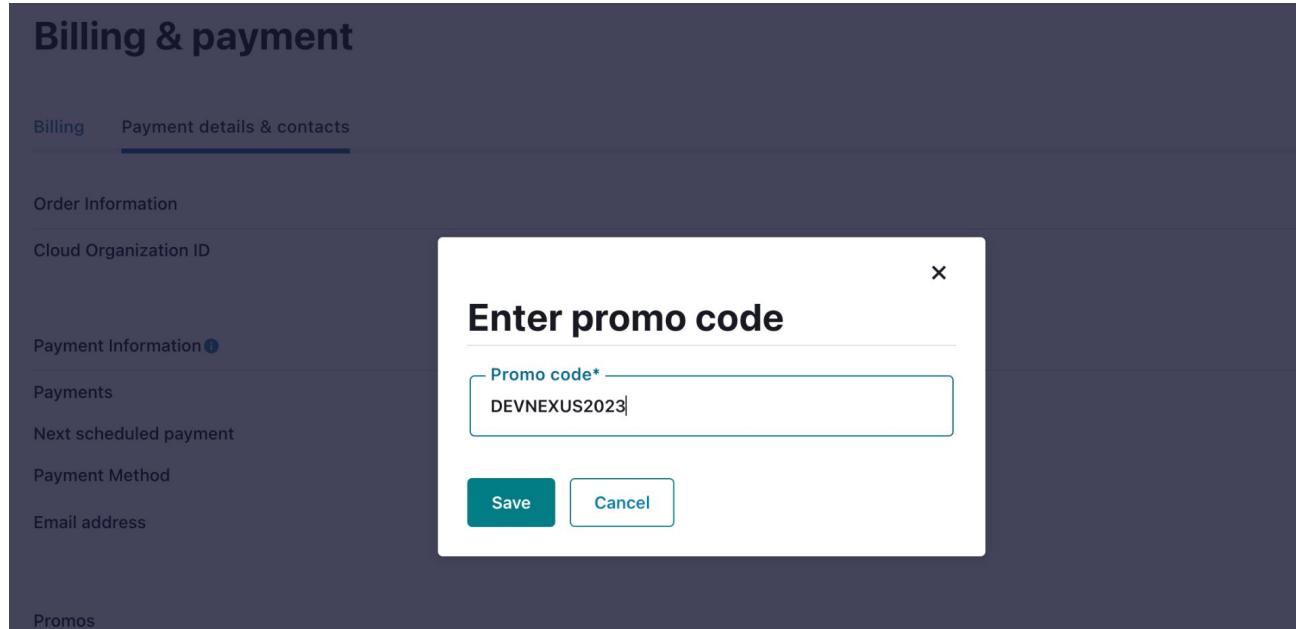
# Confluent Cloud Account



Use Promo Code  
**DEVNEXUS2023**

# Cloud Account Set Up - Apply Promo Code

Menu → Billing & payment → Payment details & contacts → + Promo code



# Cloud Account Set Up - Create Cluster

Default Env → + Add cluster → Basic cluster

## Create cluster

1. Select cluster type
2. Region/zones
3. Review and launch

Cluster name ⓘ

Base cost	\$0 /hr
Write	\$0.1265 /GB
Read	\$0.1265 /GB
Storage	\$0.00015972 /GB-hour
Partitions	\$0.0046 /Partition-hour (includes 10 free partitions)

[Go back](#)

[Review payment method](#)

[Launch cluster](#)

# Cloud Account Set Up - Create API Key

Cluster Overview → + API Keys → + Add key → Global Access

## Create key

1. Access control
2. Get your API key

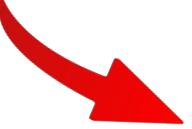
Use this API key to connect with the cluster. Store the API key and secret below somewhere safe. This is the only time you'll see the secret.

These credentials can take up to one minute to propagate.

	Key	5DXGYFAJAJN32EVR	
	Secret	NUpEwHtI10W+90dMVGsfawAdzPOWhWbJ ObHih99kRh9HE/u05S4nHDSQZJ8+PffY	

### Description

devnexus

Download and continue

# Confluent Cloud - Enable Governance

The screenshot shows the Confluent Cloud interface. At the top, there's a navigation bar with links: HOME > ENVIRONMENTS > DEFAULT > DEVNEXUS >. Below this is a sidebar with various options: Cluster Overview (selected), Dashboard, Networking, API Keys, Cluster Settings, Stream Lineage, Stream Designer, Topics, ksqlDB, Connectors, Clients, and Schema Registry. The main content area has a title 'Overview' and a large message: 'There's no data in your cluster yet'. A red arrow points from the bottom right towards a callout box at the bottom right of the message area. The callout box contains the text: 'Enable the environment default Stream Governance package to use this feature.'

HOME > ENVIRONMENTS > DEFAULT > DEVNEXUS >

Cluster Overview

Dashboard

Networking

API Keys

Cluster Settings

Stream Lineage

Stream Designer

Topics

ksqlDB

Connectors

Clients

Schema Registry

## Overview

⚠ There's no data in your cluster yet

Enable the environment default Stream Governance package to use this feature.

# Confluent Cloud - Create SR API Key

Default Environment → <<Right sidebar>> → + API Key

**default**

Clusters Network management

Search cluster name or id

+ Add cluster

Live (1)

devnexus (Running)

Metrics

Production 0B/s	Consumption 0B/s	Storage 0B
-----------------	------------------	------------

Resources

ksqldb 0	Connectors 0	Clients 0
----------	--------------	-----------

Overview

ID lkcx-xm8n9q	Type Basic	Provider & region GCP   us-west4
----------------	------------	----------------------------------

default  
ID: env-ym2gdk

Stream Governance package  
Essentials [Upgrade now](#)  
Google Cloud Platform | us-central1

Schemas Tags Business metadata

Stream Governance API  
Schema Registry and Stream Catalog API

Endpoint  
<https://psrc-xqq6z.us-central1.gcp.confluent.cloud>



Stream Governance API  
Schema Registry and Stream Catalog API

Endpoint

<https://psrc-xqq6z.us-central1.gcp.confluent.cloud>

If you're just getting started, see some [usage examples](#).

## Credentials

+ Add key

# Morning Agenda

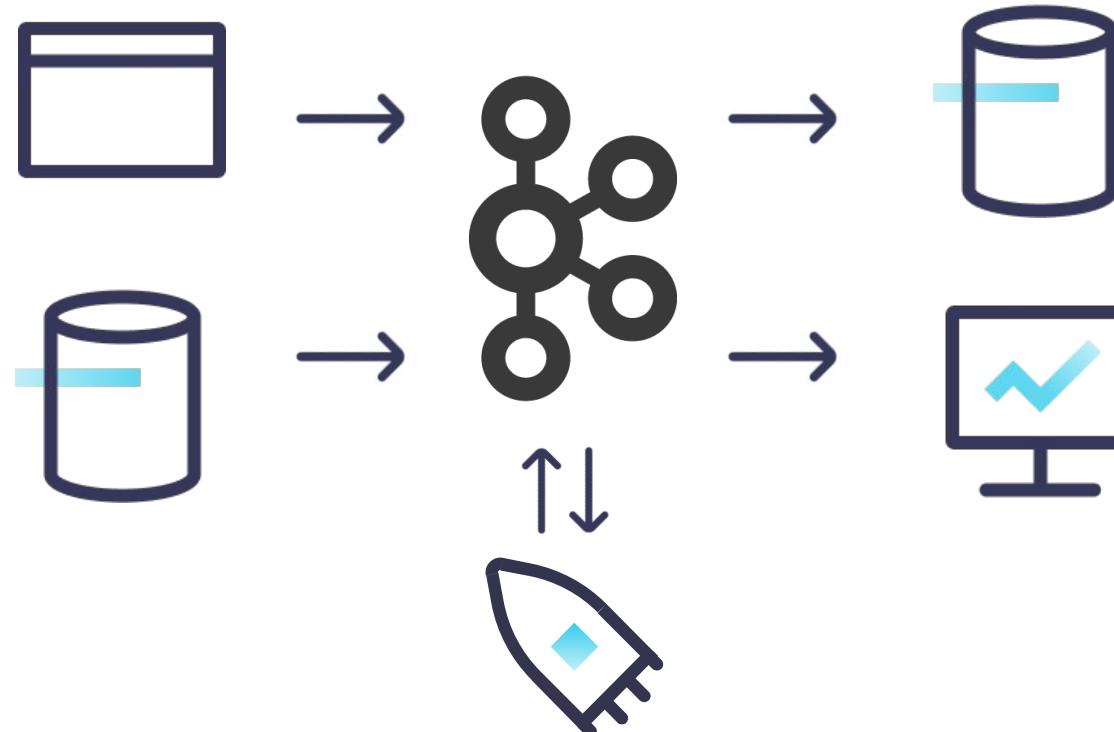
Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00	LUNCH	

# Coffee Break

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30		COFFEE BREAK
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00		LUNCH

# Streaming Data Pipeline



# Producing Data Into Kafka

Producer API vs Kafka Connect

# Writing Data to Kafka

- Kafka Producer API
  - All of your favorite languages
  - Great when you own the application producing the data
- Kafka Connect
  - Great for integrating with data *at rest*
  - Low- to no-code option
  - Hundreds of data sources (and sinks)

# Kafka Connect

- Handles ingest and egress
- Many data sources/sinks:
  - Cloud Warehouses
  - Relational Databases
  - NoSQL Stores
  - SaaS Platforms
- Configuration-driven (no-code, ftw!)
- In the cloud or self-managed



# Checkpoint

Think of a dataset you use regularly; would you use the Producer API or Kafka Connect to capture and produce that data into Kafka?

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00	LUNCH	

# Hands On

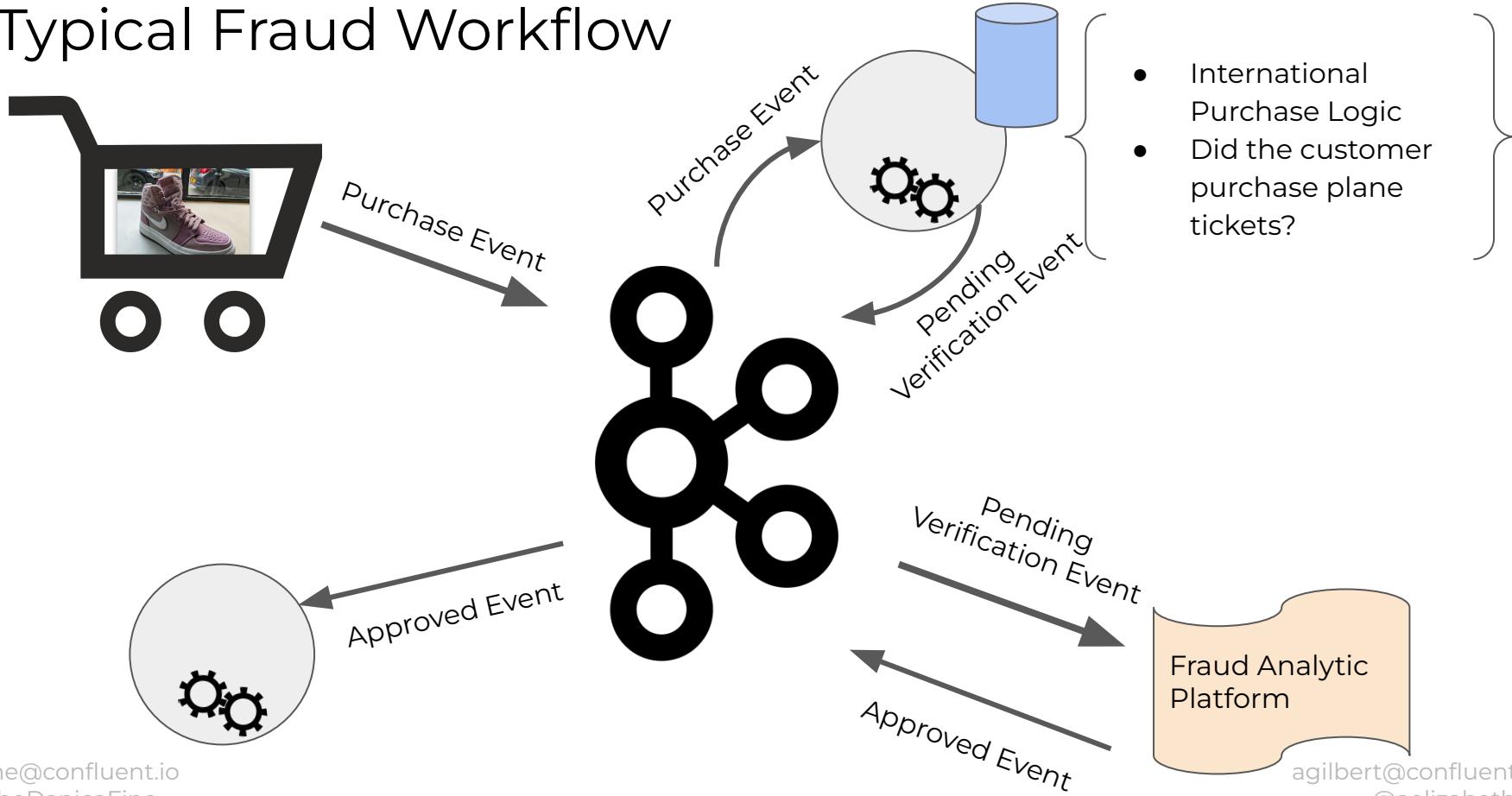
## DataGen Source Connector

# But first, a use case.

# Credit Card Fraud

What would a stream processing application for fraud look like?

# Typical Fraud Workflow



# Use Case: Credit Card Fraud

- Workshop Goal:  
Build a stream processing application to identify fraudulent credit card purchases.
- Datasets
  - Customers
  - Transactions

# Transactions

Key: 41

```
{  
  "transaction_id": 41,  
  "card_id": 47,  
  "user_id": "User_4",  
  "purchase_id": 40,  
  "store_id": 1  
}
```

# Customers (based on the users dataset)

Key: User\_5

```
{  
  "registertime": 1507482707295,  
  "userid": "User_5",  
  "regionid": "Region_7",  
  "gender": "Female"  
}
```

# DataGen Datasets

- Quickstart options – explore them!
- Coming soon to Confluent Cloud: `schema.string`
  - Optional configuration parameter
  - Provide any JSON string outlining an AVRO schema
  - Must adhere to avro-random-generator standards

# schema.string Example – Base Schema

```
{  
  "type": "record",  
  "namespace": "survey.bot",  
  "name": "survey",  
  "doc": "Single-question survey.",  
  "fields": [  
    {  
      "doc": "Unique survey ID.",  
      "name": "survey_id",  
      "type": "int"  
    },  
    {  
      "doc": "Survey question.",  
      "name": "question",  
      "type": "string"  
    },  
    {  
      "doc": "Survey summary.",  
      "name": "summary",  
      "type": "string"  
    },  
    {  
      "doc": "Response options.",  
      "name": "options",  
      "type": {  
        "type": "array",  
        "items": "string"  
      }  
    },  
    {  
      "doc": "Survey enabled flag.",  
      "name": "enabled",  
      "type": "boolean"  
    }  
  ]  
}
```

# schema.string Example – String Options

```
{  
    "doc": "Response options.",  
    "name": "options",  
    "type": {  
        "type": "array",  
        "items": "string"  
        "arg.properties": {  
            "options": [  
                ["Always", "Sometimes", "Never"],  
                ["18-30", "31-50", "50+"],  
                ["Red", "Blue", "Yellow", "Purple"]  
            ]  
        }  
    }  
}
```

# schema.string Example – Int Range

```
{  
    "doc": "Unique survey ID.",  
    "name": "survey_id",  
    "type": {  
        "type": int,  
        "arg.properties": {  
            "range": {  
                "min": 100,  
                "max": 1000  
            }  
        }  
    }  
}
```

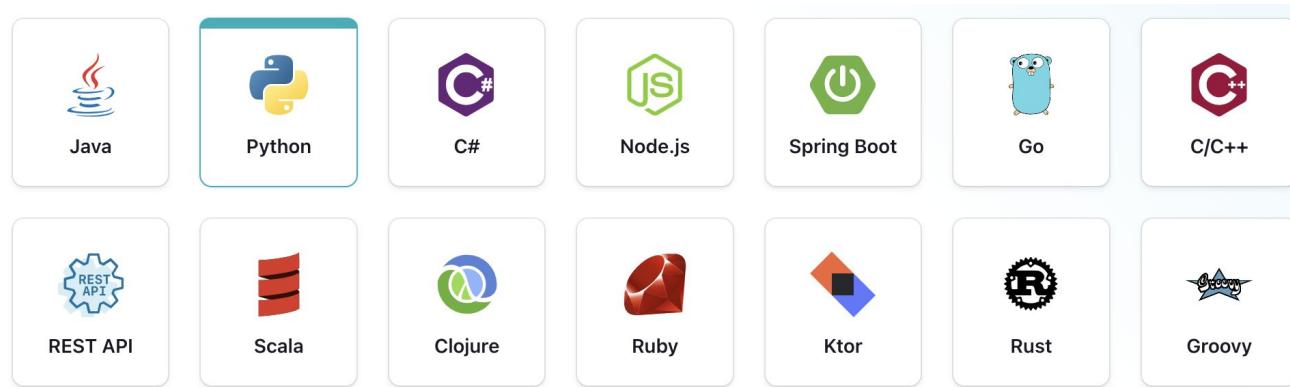
# To Confluent Cloud!

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00	LUNCH	

# Kafka Clients

- Low-level access to Kafka clusters
- High-level stream processing
- Available in most programming languages



# Confluent Cloud - Python Client Configuration

```
# Required connection configs for Kafka producer, consumer, and admin
bootstrap.servers=pkc-lzvrd.us-west4.gcp.confluent.cloud:9092
security.protocol=SASL_SSL
sasl.mechanisms=PLAIN
sasl.username={{ CLUSTER_API_KEY }}
sasl.password={{ CLUSTER_API_SECRET }}

# Best practice for higher availability in librdkafka clients prior to 1.7
session.timeout.ms=45000

# Required connection configs for Confluent Cloud Schema Registry
schema.registry.url=https://psrc-22250.us-central1.gcp.confluent.cloud
basic.auth.credentials.source=USER_INFO
basic.auth.user.info={{ SR_API_KEY }}:{{ SR_API_SECRET }}
```

# Producer Configurations

- key.serializer and value.serializer
- acks
- partitioner.class
- batch.size
- retries
- delivery.timeout.ms  
request.timeout.ms, and linger.ms



Confluent Producer  
Documentation

# Consumer Configurations

- key.deserializer and value.deserializer
- group.id and partition.assignment.strategy
- session.timeout.ms
- auto.offset.reset and enable.auto.commit



Confluent Consumer  
Documentation

# Checkpoint Quiz

Which configuration options apply to the following scenarios?

# Durable Writes

- Scenario:
  - Highly-available system
  - Minimal impact in a disaster recovery situation
- Goal: Ensure that all brokers containing an in-sync replica will receive every message.
- Producer Configuration:
  - acks=all

# Non-Committing Consumer

- Scenario:
  - Compacted topic
  - Consumer is building up state
- Goal: every time the consumer comes up, it should be able to read from the beginning of the Kafka topic to build up state.
- Consumer Configuration:
  - `auto.offset.reset=earliest`
  - `enable.auto.commit=false`

# Exactly Once Consumer

- Scenario: A mission-critical application that needs to receive every message once.
- Goal: eliminate data loss or duplicate reads from consumers
- Consumer Configuration:
  - enable.auto.commit=false
  - Manually commit after processing each message.
- Or:
  - enable.auto.commit=true
  - auto.commit.interval.ms=1000

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00	LUNCH	

# Hands On

## Create a Python Client

# Confluent Cloud - Create Python Client

Cluster Overview → Clients → + New client → Python

<p><b>Cluster API Key</b></p> <p>An API key is required to connect to your cluster. You can either use an existing key or create a new one here.</p> <p><a href="#">Create Kafka cluster API key</a></p> <p><b>Schema Registry API Key</b></p> <p>Schema Registry allows you to enforce a strict schema for your data. Learn more ↗</p> <p><a href="#">Create Schema Registry API key</a></p>	<p><span style="float: right;">Copy</span></p> <pre>1 # Required connection configs for Kafka producer, consumer, and admin 2 bootstrap.servers=pkc-lzvrd.us-west4.gcp.confluent.cloud:9092 3 security.protocol=SASL_SSL 4 sasl.mechanisms=PLAIN 5 sasl.username={{ CLUSTER_API_KEY }} 6 sasl.password={{ CLUSTER_API_SECRET }} 7 8 # Best practice for higher availability in librdkafka clients prior 9 # to 1.7 10 session.timeout.ms=45000 11 12 # Required connection configs for Confluent Cloud Schema Registry 13 schema.registry.url=https://psrc-22250.us- 14 central1.gcp.confluent.cloud 15 basic.auth.credentials.source=USER_INFO 16 basic.auth.user.info={{ SR_API_KEY }}:{{ SR_API_SECRET }}</pre>
---	--

# Python Client Sandbox

- Update client configurations to point to your cluster using appropriate API Key
- Execute producer and consumer
- Play in the sandbox:
  - Alter configuration parameters
  - Add new datasets/schemas and produce them to Confluent Cloud
  - Conduct some analysis
  - Make the customers and transactions datasets more interesting



<https://github.com/danicafine/devnexus-kafka>

# Morning Agenda

Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction to Kafka Connect
10:55	Hands On	Kafka Connect Data Generator on Confluent Cloud
11:15	Instruction	Producer and Consumer Configurations Python Code Samples and Walk Through
11:45	Hands On	Setting Up Producers and Consumers
12:00	LUNCH	

# Lunch

# Afternoon Agenda

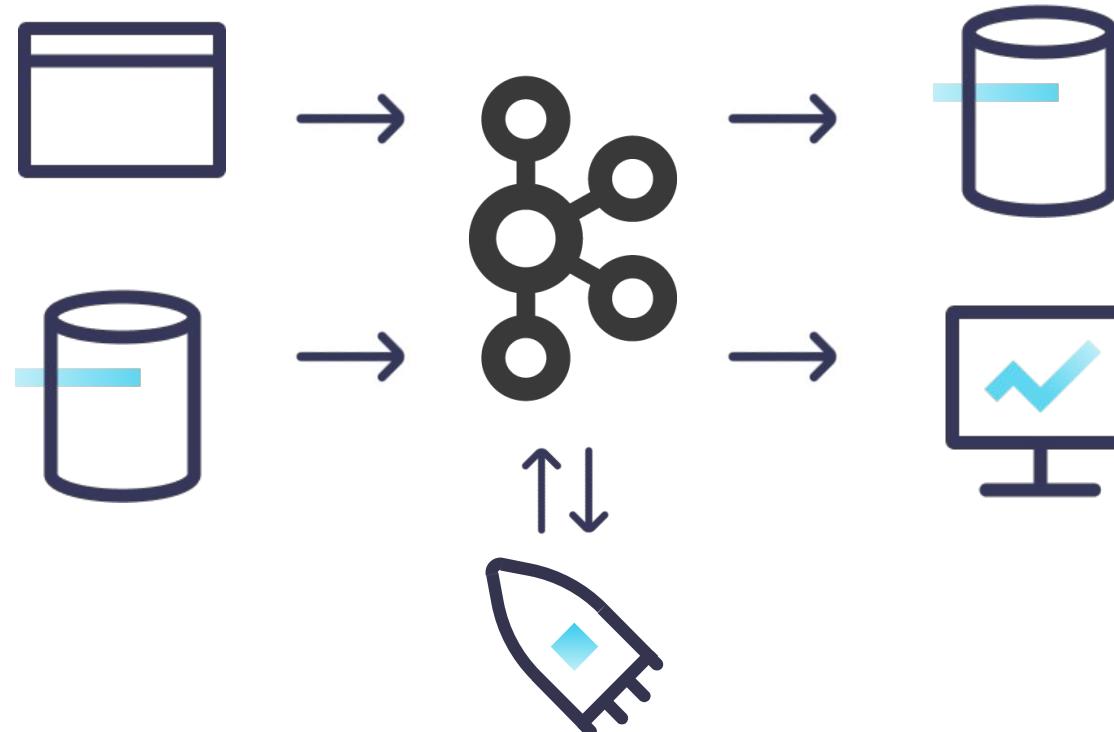
Time	Module Type	Description
12:00		LUNCH
13:00	Instruction	Stream Processing Techniques in the Apache Kafka Ecosystem ksqlDB in Confluent Cloud
13:45	Hands On	Build a Stream Processing Application with ksqlDB
14:30		COFFEE BREAK
14:45	Instruction	Advanced Stream Processing Pipelines Use Cases and Inspiration
15:00	Hands On	Outline an End-to-End Pipeline Independent Working Time to Complete Stream Processing Pipelines

# Afternoon Agenda

Time	Module Type	Description
12:00		LUNCH
13:00	Instruction	Stream Processing Techniques in the Apache Kafka Ecosystem ksqlDB in Confluent Cloud
13:45	Hands On	Build a Stream Processing Application with ksqlDB
14:30		COFFEE BREAK
14:45	Instruction	Advanced Stream Processing Pipelines Use Cases and Inspiration
15:00	Hands On	Outline an End-to-End Pipeline Independent Working Time to Complete Stream Processing Pipelines

# Stream Processing Techniques

# Streaming Data Pipeline



# Use Case: Credit Card Fraud

- Workshop Goal:  
Build a stream processing application to identify fraudulent credit card purchases.
- Datasets
  - Customers
  - Transactions

# Consumer/Producer API

- Lowest level
- Consume – Process – Produce
- Handle state and fault-tolerance

```
subscribe(), poll(), send(),
flush(), beginTransaction(), ...
```

# Example: Consumer/Producer API

```
# set up Kafka Consumer for Customers
consumer = clients.consumer(clients.customer_deserializer(),
    'consumer-group-customermask', [config['topics'][customeres]])

# set up Kafka Producer for CustomerMasks
producer = clients.producer(clients.customermask_serializer())

# start consumption loop
while True:
    msg = consumer.poll(1.0)

    if msg is not None:
        # received a message
        customer = msg.value()

        # transform the data
        customermask =
            CustomerMask(customer.registertime, customer.userid, customer.regionid)

        # send data to Kafka
        producer.produce(config['topics']['customermasks'],
                         key=str(customer.userid), value=customermask)
```

# Kafka Streams

- Java library for stream processing
- Rich API of transformations
- Built-in state handling and failover

`KStream, KTable,  
filter(), map(), flatMap(),  
join(), aggregate(), ...`

# Example: Kafka Streams

```
static Topology buildTopology(String inputTopic, String outputTopic) {  
    Serde<String> stringSerde = Serdes.String();  
    StreamsBuilder builder = new StreamsBuilder();  
    builder  
        .stream(inputCustomerTopic,  
                Consumed.with(stringSerde, stringSerde))  
        .peek((k,v) -> logger.info("Observed customer: {}", v))  
        .mapValues(c -> c.maskCustomer())  
        .peek((k,v) -> logger.info("Transformed customer: {}", v))  
        .to(outputCustomerTopic,  
             Produced.with(stringSerde, stringSerde));  
    return builder.build();  
}
```

# ksqldb

- SQL syntax
- Kafka Streams under the hood
- Available as self-managed and cloud-based offerings

**CREATE STREAM, CREATE TABLE,  
SELECT, JOIN, GROUP BY, SUM, ...**

# Example: ksqlDB

```
CREATE STREAM customermasks  
  
WITH (kafka_topic='customermasks', value_format='avro', partitions=1) AS  
  
SELECT  
  
    REGISTERTIME,  
  
    USERID,  
  
    REGIONID,  
  
    MASK(GENDER)  
  
FROM CUSTOMERS;
```

# Afternoon Agenda

Time	Module Type	Description
12:00		LUNCH
13:00	Instruction	Stream Processing Techniques in the Apache Kafka Ecosystem ksqlDB in Confluent Cloud
13:45	Hands On	Build a Stream Processing Application with ksqlDB
14:30		COFFEE BREAK
14:45	Instruction	Advanced Stream Processing Pipelines Use Cases and Inspiration
15:00	Hands On	Outline an End-to-End Pipeline Independent Working Time to Complete Stream Processing Pipelines

# Hands On

## Stream Processing With ksqlDB

# ksqlDB - Create a Cluster

Cluster Overview → ksqlDB → Create with Tutorial → Follow Prompts

## New cluster

1. Access control —— 2. Name and sizing

Cluster name\* \_\_\_\_\_

`fraud_detection_app`

Cluster size\* \_\_\_\_\_

1 \$0.2222 USD /hr ▾

# ksqldb Stream Processing

- Use Case: Fraud Detection
  - Register data: Stream or Table?
  - Pre-processing:
    - Filtering a stream
    - Join our two data sources
  - Aggregation over a window
- Other Use Cases?



ksqldb Reference Materials

# Use Case: Fraud Detection

- Register data: Stream or Table?
- Pre-processing:
  - Filtering a stream
  - Join our two data sources
- Identify suspicious activity
  - What does that mean for fraud?
  - Windowing transactions

# Creating a Table

```
CREATE TABLE customers (
    userid STRING PRIMARY KEY,
    registertime BIGINT,
    regionid STRING,
    gender STRING
) WITH (
    KAFKA_TOPIC = 'customers',
    VALUE_FORMAT = 'AVRO'
);
```

```
1 CREATE TABLE customers (
2     userid STRING PRIMARY KEY,
3     registertime BIGINT,
4     regionid STRING,
5     gender STRING
6 ) WITH (
7     KAFKA_TOPIC = 'customers',
8     VALUE_FORMAT = 'AVRO'
9 );
10
```

- [Add query properties](#)

auto.offset.reset = Earliest  

+Add another field

Stop

Run query

# Creating a Stream

```
CREATE STREAM transactions WITH (
    kafka_topic='transactions',
    value_format='AVRO'
);
```

```
1 CREATE STREAM transactions WITH (
2     kafka_topic='transactions',
3     value_format='AVRO'
4 );
5
```

- [Add query properties](#)

[+ Add another field](#)[Stop](#)[Run query](#)

# Filtering a Stream

```
CREATE TABLE customers_filtered WITH (
    kafka_topic='customers-filtered',
    value_format='AVRO'
) AS
SELECT
    *
FROM CUSTOMERS
WHERE USERID != 'User_8'
EMIT CHANGES;
```

# Stream-to-Table Join

```
CREATE STREAM transactions_enriched WITH (
    kafka_topic='transactions_enriched',
    value_format='AVRO'
) AS
SELECT
    *
FROM transactions t
LEFT JOIN customers_filtered c
ON t.user_id = c.id
EMIT CHANGES;
```



Filter by keyword



# Pull Query

SELECT

\*

FROM transactions

WHERE user\_id != 'User\_99';

▼ {"TRANSACTION\_ID":4857408,"CARD\_ID":20,"USER\_ID":"User\_","PURCHASE\_ID":...}

▼ {"TRANSACTION\_ID":4857405,"CARD\_ID":11,"USER\_ID":"User\_","PURCHASE\_ID":...}

▼ {"TRANSACTION\_ID":4857394,"CARD\_ID":23,"USER\_ID":"User\_4","PURCHASE\_ID":...}

# Windowing

```
CREATE TABLE transactions_suspicious WITH (
    kafka_topic='transactions_suspicious',
    format='AVRO'
) AS
SELECT
    t_user_id AS user_id,
    COUNT(t_transaction_id) AS transactions_count
FROM transactions_enriched
WINDOW TUMBLING (SIZE 60 SECONDS)
GROUP BY t_user_id
HAVING COUNT(t_transaction_id) > 4
EMIT CHANGES;
```

# Windowing Final

```
CREATE TABLE transactions_suspicious WITH (
    kafka_topic='transactions_suspicious',
    format='AVRO'
) AS
SELECT
    t_user_id AS user_id,
    COUNT(t_transaction_id) AS transactions_count
FROM transactions_enriched
WINDOW TUMBLING (SIZE 30 SECONDS)
GROUP BY t_user_id
HAVING COUNT(t_transaction_id) > 4
EMIT FINAL;
```

# Afternoon Agenda

Time	Module Type	Description
12:00		LUNCH
13:00	Instruction	Stream Processing Techniques in the Apache Kafka Ecosystem ksqlDB in Confluent Cloud
13:45	Hands On	Build a Stream Processing Application with ksqlDB
14:30		COFFEE BREAK
14:45	Instruction	Advanced Stream Processing Pipelines Use Cases and Inspiration
15:00	Hands On	Outline an End-to-End Pipeline Independent Working Time to Complete Stream Processing Pipelines

# Coffee Break

# Afternoon Agenda

Time	Module Type	Description
12:00		LUNCH
13:00	Instruction	Stream Processing Techniques in the Apache Kafka Ecosystem ksqlDB in Confluent Cloud
13:45	Hands On	Build a Stream Processing Application with ksqlDB
14:30		COFFEE BREAK
14:45	Instruction	Advanced Stream Processing Pipelines Use Cases and Inspiration
15:00	Hands On	Outline an End-to-End Pipeline Independent Working Time to Complete Stream Processing Pipelines

# Use Cases and Inspiration

# Online Multiplayer System

- Kafka forms the backbone of the system
  - All user-input is treated as an event
  - Allows for easy conflict-resolution
- Wide variety of applications
  - Gaming
  - Collaborative work tools (think Google Docs)

# Event-Driven Campaign Management - Moving Company

- Kafka integrates legacy systems with customer-centric apps
  - When a customer updates their status it gets registered in Kafka as an event
  - Real-time population of campaign management
- Wide variety of applications
  - Order tracking
  - Inventory management
  - Customer 360 and analytics

# Houseplant Alerting System

- Raspberry Pi-based, IoT Project
  - Collect moisture readings from plants using sensors
  - Use Python script to send readings to Kafka as events
- Stream Processing
  - Enrich readings with houseplant metadata
  - Determine if plants need to be watered and send a text alert

# Kafka Connect - Elastic Sink Connector

Create a free account

The screenshot shows the Elastic homepage with the following content:

- Welcome home**: A header section featuring four colored cards: yellow (Enterprise Search), pink (Observability), teal (Security), and blue (Analytics).
- Enterprise Search**: Create search experiences with a refined set of APIs and tools.
- Observability**: Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.
- Security**: Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.
- Analytics**: Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.
- Get started by adding integrations**: A section with a callout graphic showing data being processed and analyzed. It includes three buttons: [Add integrations](#), [Try sample data](#), and [Upload a file](#).
- Text below integrations**: To start working with your data, use one of our many ingest options. Collect data from an app or service, or upload a file. If you're not ready to use your own data, add a sample data set.

# Kafka Connect - Elastic Sink Connector

## Configure Elastic Sink Connector

The screenshot shows the Confluent Platform interface for managing Kafka Connectors. The top navigation bar includes links for Stream Catalog, LEARN, Notifications, and a three-dot menu. The main navigation on the left has sections for Cluster overview, Topics, Data integration (Clients, Connectors selected), API keys, Stream lineage, and ksqlDB. A search bar at the top right is set to "elasticsearch". The central area is titled "Connectors" and displays a search result for "elasticsearch" showing two connectors: "Elasticsearch Service Sink" and "Elasticsearch Service Source". The "Elasticsearch Service Sink" card is detailed below:

**Elasticsearch Service Sink**

The Kafka Connect Elasticsearch Service sink connector for Confluent Cloud moves data from Apache Kafka® to Elasticsearch.

**Documentation**

**Plugin type:** Sink

**Enterprise support:** Confluent supported

**Available fully managed**

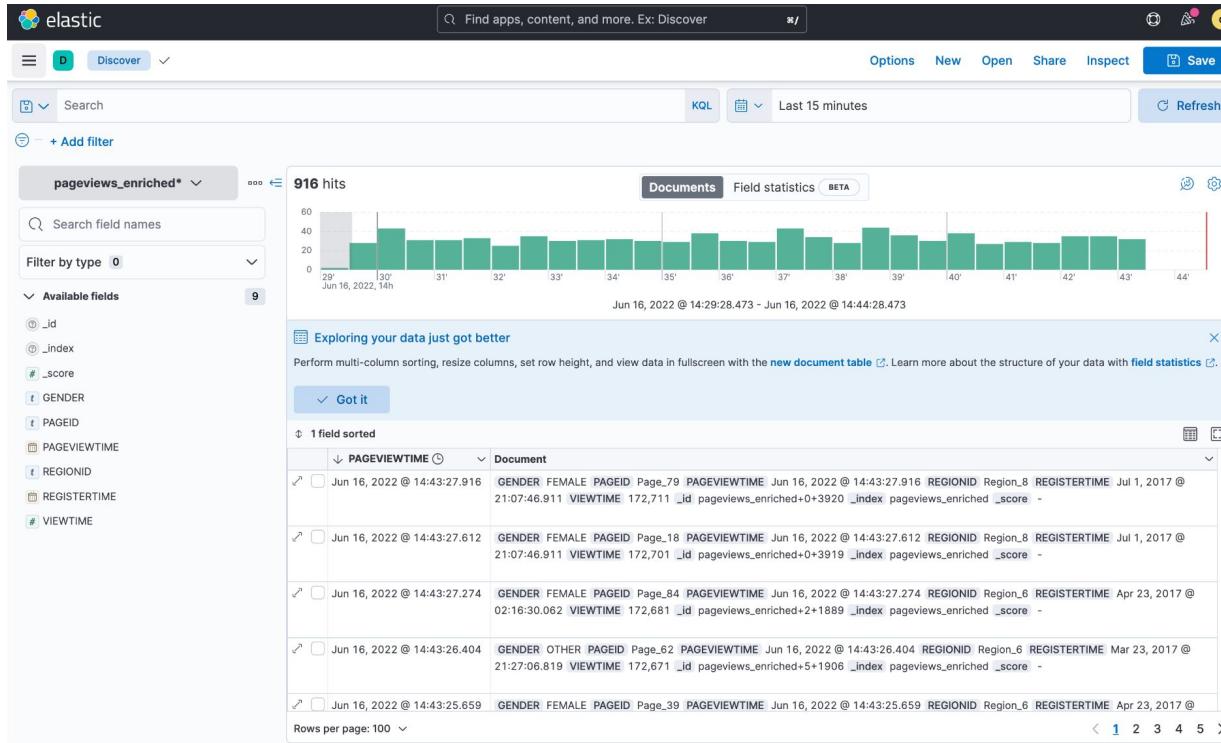
Push data in less than 10 minutes

**Get started**

At the bottom of the page, there are links for Schema Registry and CLI and tools.

# Kafka Connect - Elastic Sink Connector

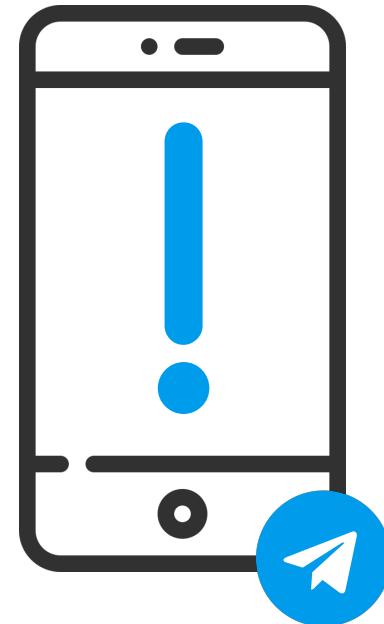
## Conduct analysis



# Kafka Connect - HTTP Sink Connector

Send alerts to your phone

- Telegram App
- Create a bot
- Send data to bot endpoint



# Sample Data Sets

- US Census Bureau -  
<https://www.census.gov/quickfacts/fact/table/US/PST045221>
- Northern Data Hub -  
[https://datahub.bradford.gov.uk/ebase/datahubext.eb?ebd=0&ebp=10&ebz=1\\_1666813862928](https://datahub.bradford.gov.uk/ebase/datahubext.eb?ebd=0&ebp=10&ebz=1_1666813862928)
- Network Rail -  
<https://datafeeds.networkrail.co.uk/ntrod/login;jsessionid=1C499ED60E92B2BCD2332267555EE21B>

# More Starting Points: ksqlDB Recipes



COURSES   LEARN   BUILD   COMMUNITY   DOCS   [GET STARTED FREE](#)

## Explore top use cases



### Anomaly and pattern detection

Identify patterns and unusual activity to detect fraud and monitor network health

[Detect unusual credit card activity](#)

[Flag unhealthy IoT devices](#)

[Handle corrupted data from Salesforce](#)



### Customer 360

Combine data sources for a unified 360-degree view of the customer across channels

**NEW** [Correlate customer behavior across in-store and online channels](#)

[Match users for online dating](#)

[Understand user behavior with clickstream data](#)

[Build customer loyalty programs](#)

[Get a full view of a customer's online journey](#)



### Cybersecurity

Protect critical systems and sensitive information by detecting threats in real-time

[Monitor security threats by analyzing and filtering audit logs](#)

[Identify firewall deny events](#)

[Detect a denial-of-service \(DDoS\) attack](#)

[Detect and analyze SSH attacks](#)

# More Starting Points: ksqlDB Recipes



## Predictive analytics

Train predictive models by using machine learning to anticipate future outcomes

Optimize omni-channel inventory

Create personalized banking promotions

Retrain a machine learning model

**NEW** Analyze political campaign fundraising performance



## Real-time analytics

Extract real-time insights by performing analytics on streaming data as it's generated

Capture and analyze survey responses

Analyze datacenter power usage

Automate instant payment verifications

Enrich orders with change data capture (CDC)

Build a dynamic pricing strategy

Assess marketing promotional campaign efficacy



## Real-time Alerts and Notifications

Build real-time alerts and notifications to power digital customer experiences



## Infrastructure Modernization

Modernize legacy technologies and rationalize infrastructure footprint with modern systems



ksqlDB Recipes Hub

# What are your use cases?

How can we implement them now?

# Afternoon Agenda

Time	Module Type	Description
12:00		LUNCH
13:00	Instruction	Stream Processing Techniques in the Apache Kafka Ecosystem ksqlDB in Confluent Cloud
13:45	Hands On	Build a Stream Processing Application with ksqlDB
14:30		COFFEE BREAK
14:45	Instruction	Advanced Stream Processing Pipelines Use Cases and Inspiration
15:00	Hands On	Outline an End-to-End Pipeline Independent Working Time to Complete Stream Processing Pipelines

# Hands On

## End-to-End Pipelines