Creating an Apache Kafka® Streaming Data Pipeline

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Danica Fine
https://linktr.ee/thedanicafine

Lucia Cerchie
https://luciacerchie.dev/

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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 \ ˈdā-tə ˈpīp-ˌlīn \

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

What's a *streaming* data pipeline?

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

streaming data pipeline

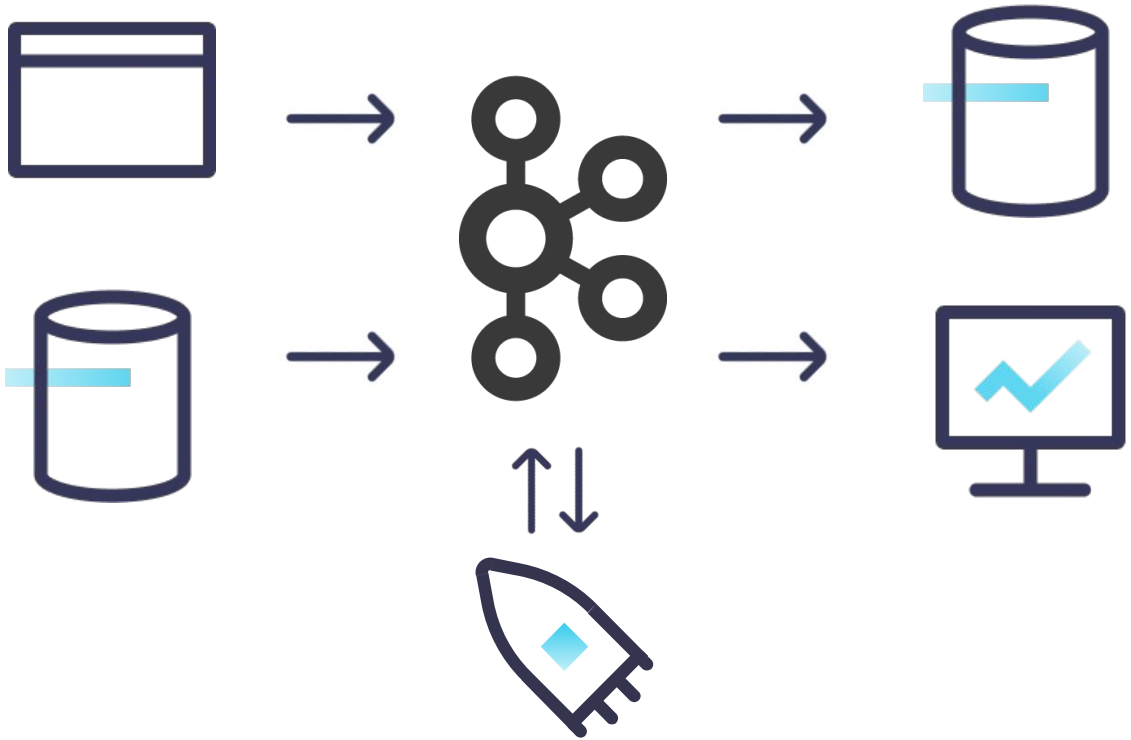
noun \ ˈstrē-miŋ ˈdā-tə ˈpīp-ˌlīn \

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Streaming Data Pipeline



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Streaming Data Pipelines

Who cares?

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Benefits of Stream Processing

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Stream Processing

What are your use cases?

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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.

dfine@confluent.io
@TheDanicaFine

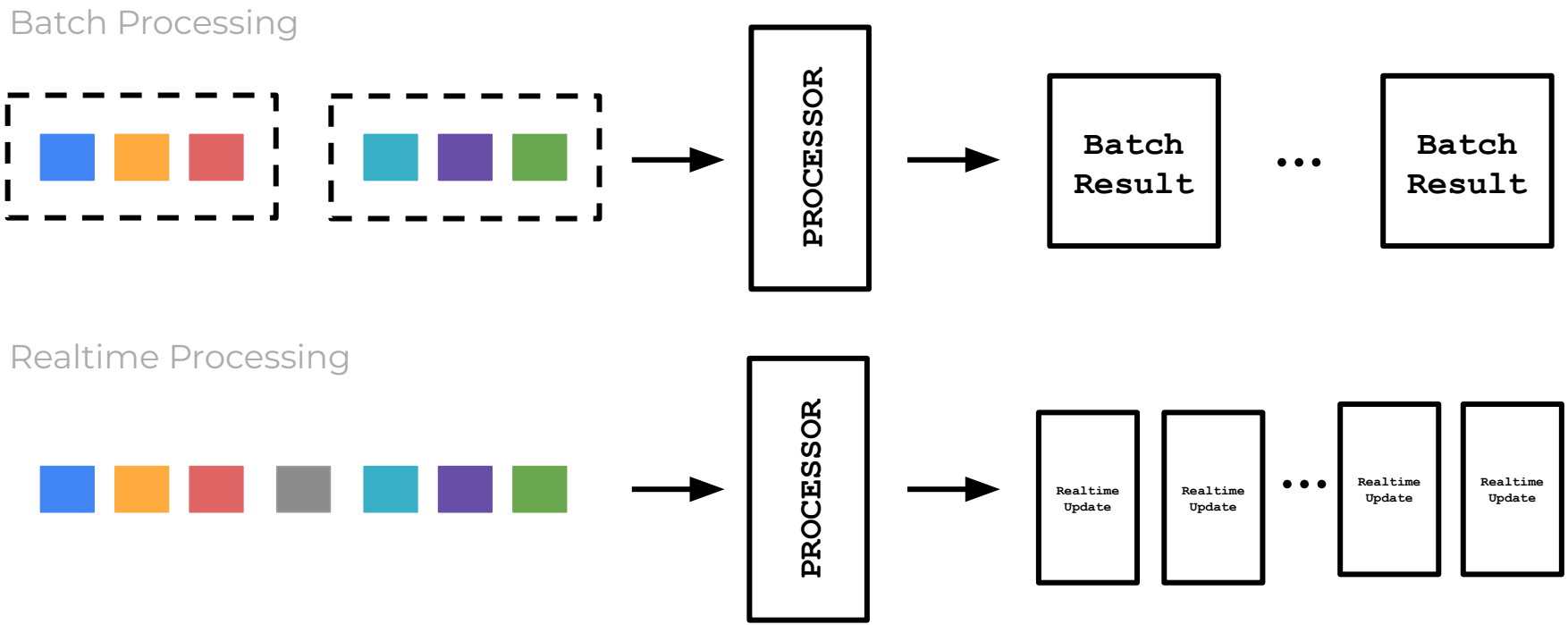
lcerchie@confluent.io
@CerchieLucia

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Paradigm Shift



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Some key differences

	Batch Data Processing	Real-Time Data Processing	Streaming Data
Hardware	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
Performance	Latency could be minutes, hours, or days	Latency needs to be in seconds or milliseconds	Latency must be guaranteed in milliseconds
Data set	Large batches of data	Current data packet or a few of them	Continuous streams of data
Analysis	Complex computation and analysis of a larger time frame	Simple reporting or computation	Simple reporting or computation

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Thinking in Events

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Events: what they look like

Value Header Key

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Events: what they look like

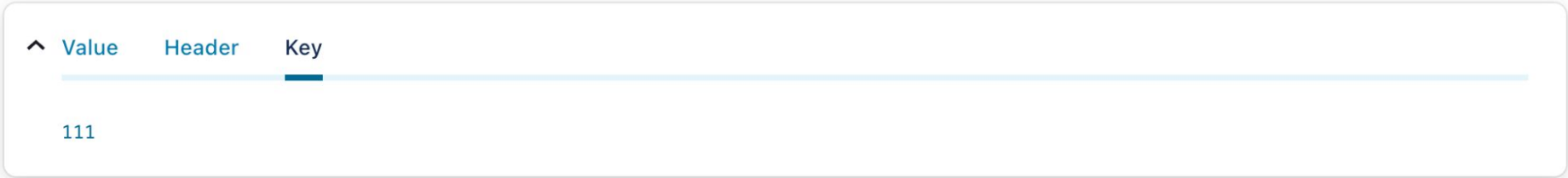
Value Header Key

```
[
  {
    "key": "task.generation",
    "stringValue": "0"
  },
  {
    "key": "task.id",
    "stringValue": "0"
  },
  {
    "key": "current.iteration",
    "stringValue": "11"
  }
]
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

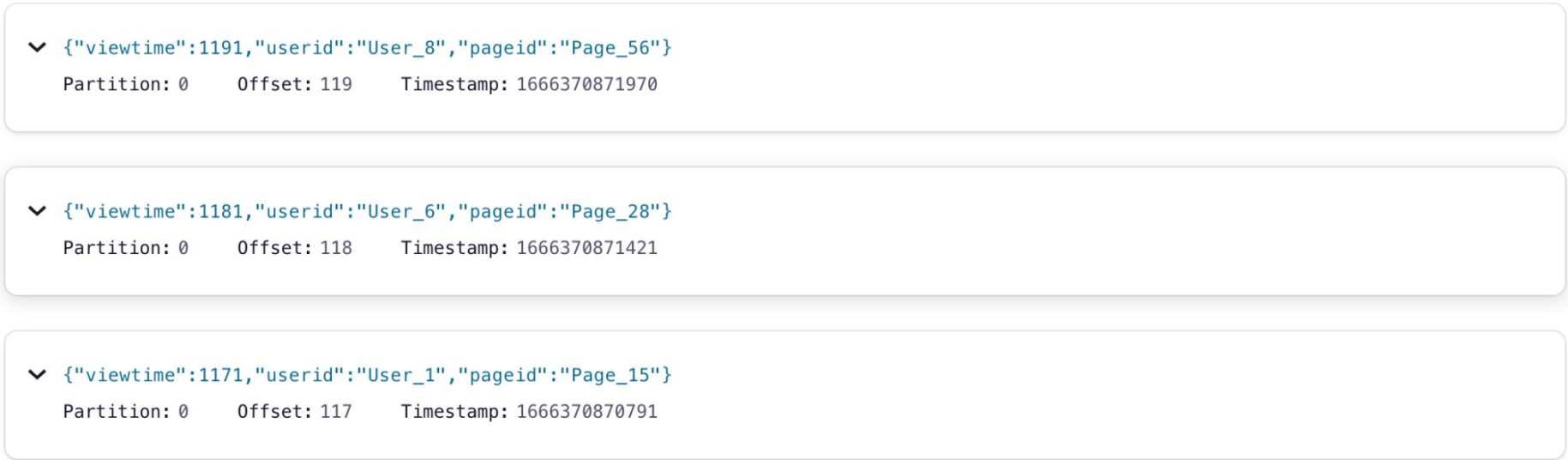
Events: what they look like



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Events: what they look like



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

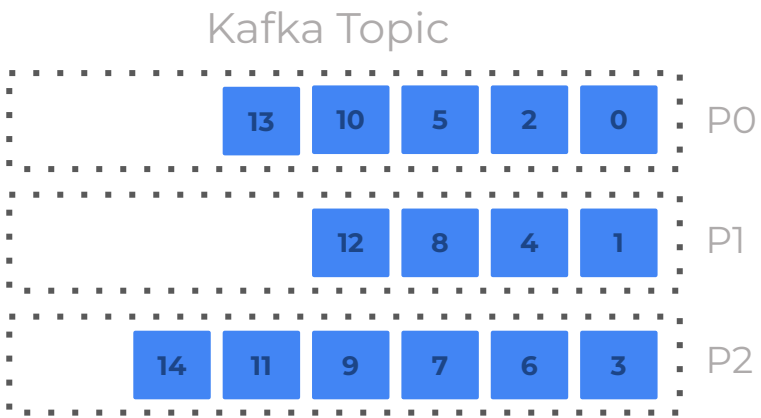
A ***distributed*** event streaming platform.

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Kafka Storage

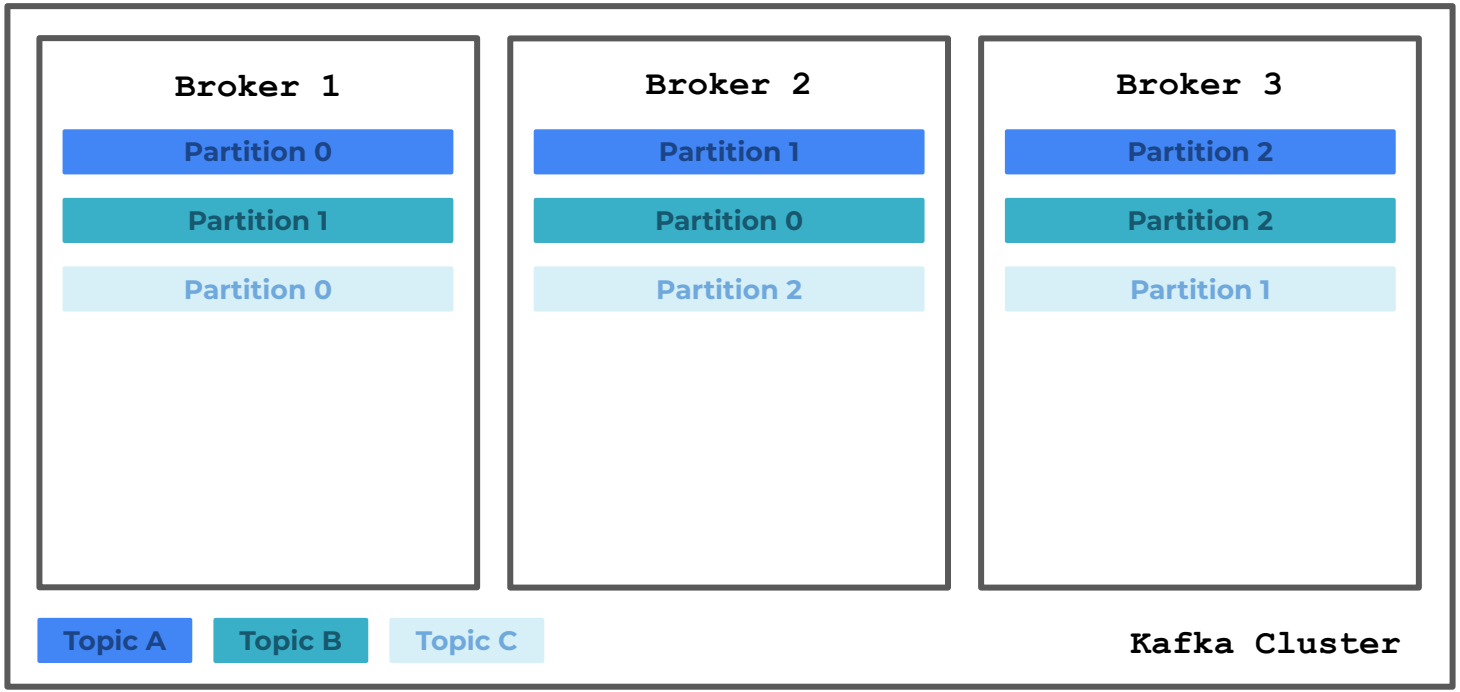
- Topics
  - Basic storage unit
  - Read/Write:
    - Producer and consumer clients
    - Completely decoupled
- Partitions
  - Immutable, append-only logs
  - Data is replicated at this level



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

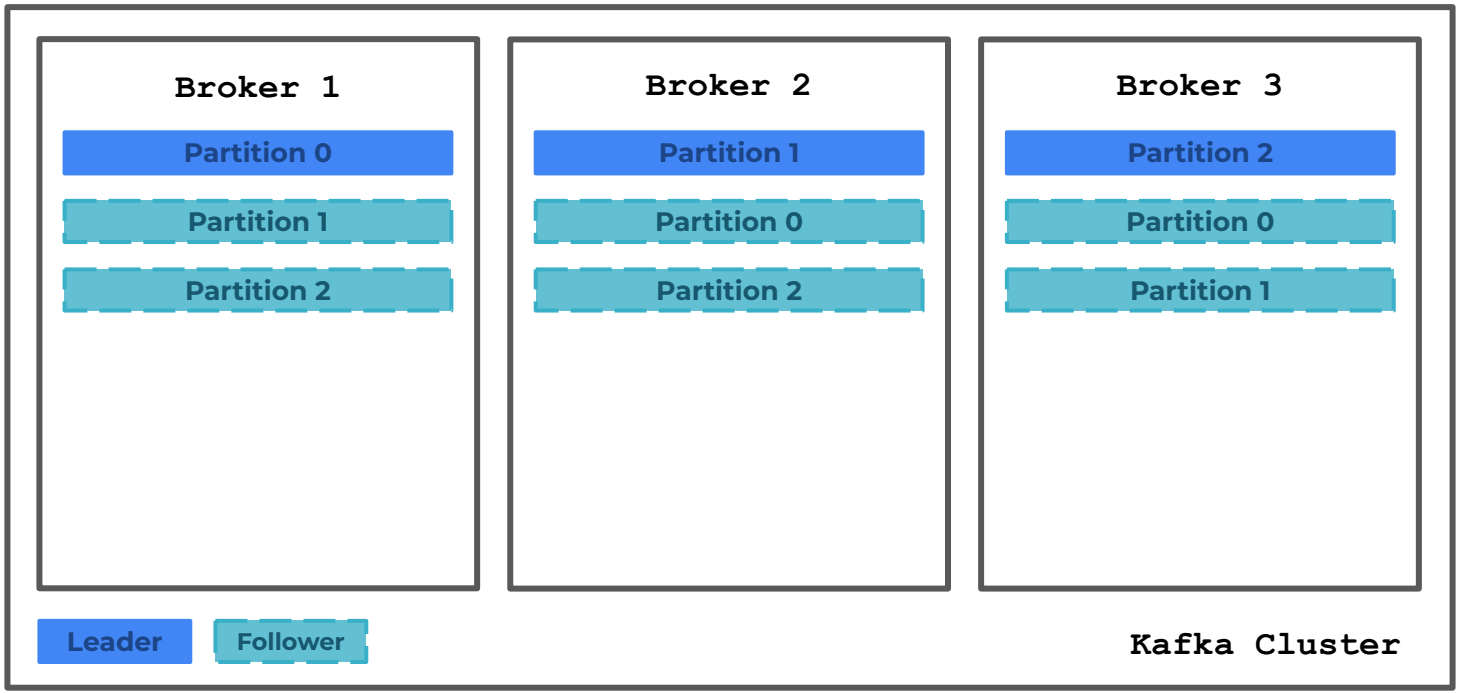
Kafka Cluster



dfine@confluent.io
@TheDanicaFine

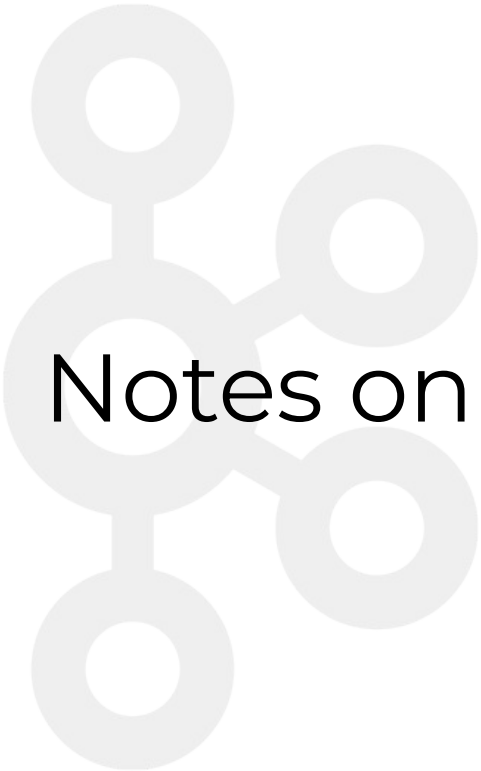
lcerchie@confluent.io
@CerchieLucia

Kafka Cluster



dfine@confluent.io
@TheDanicaFine

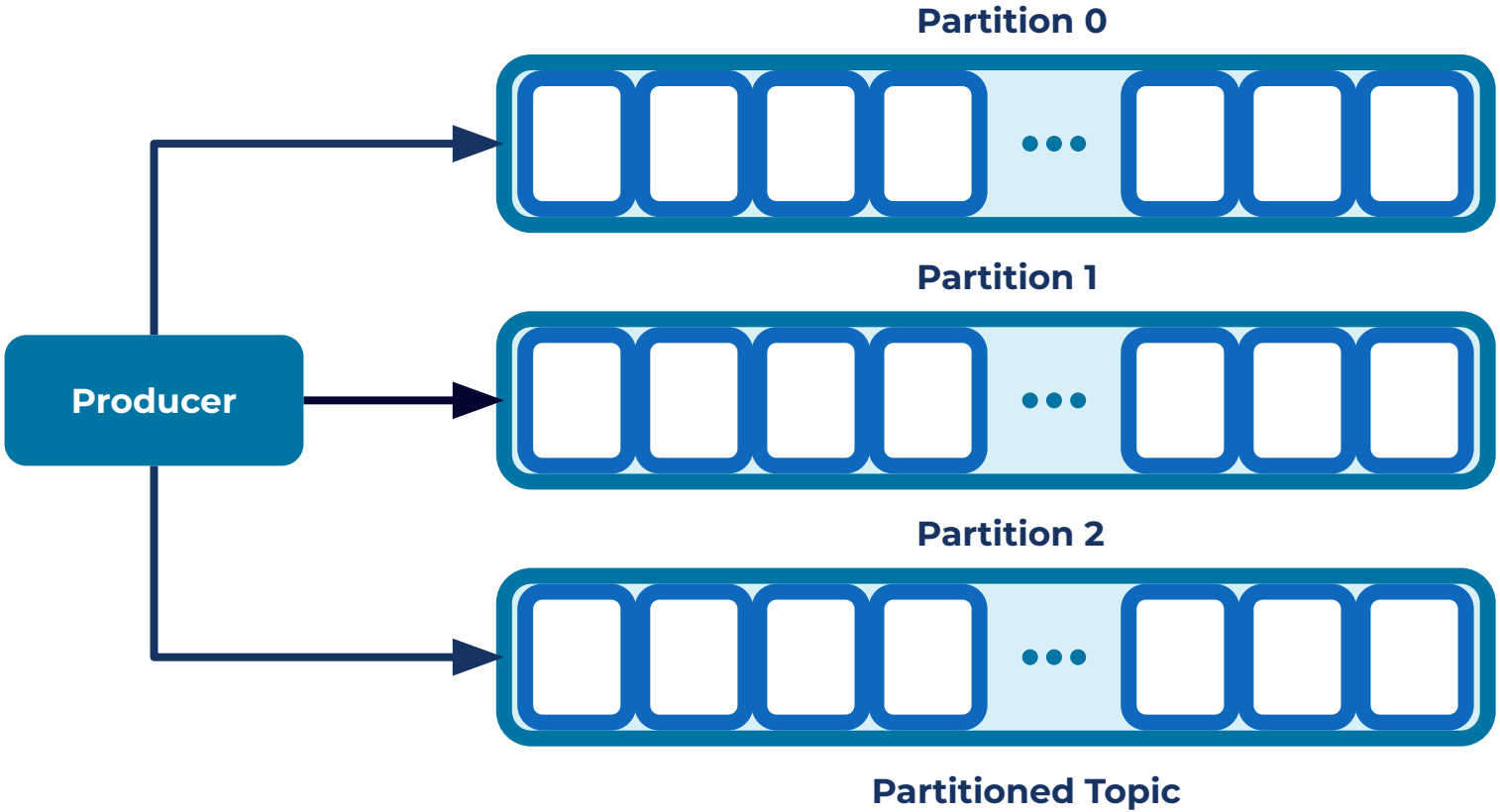
lcerchie@confluent.io
@CerchieLucia



Notes on producers and consumers.

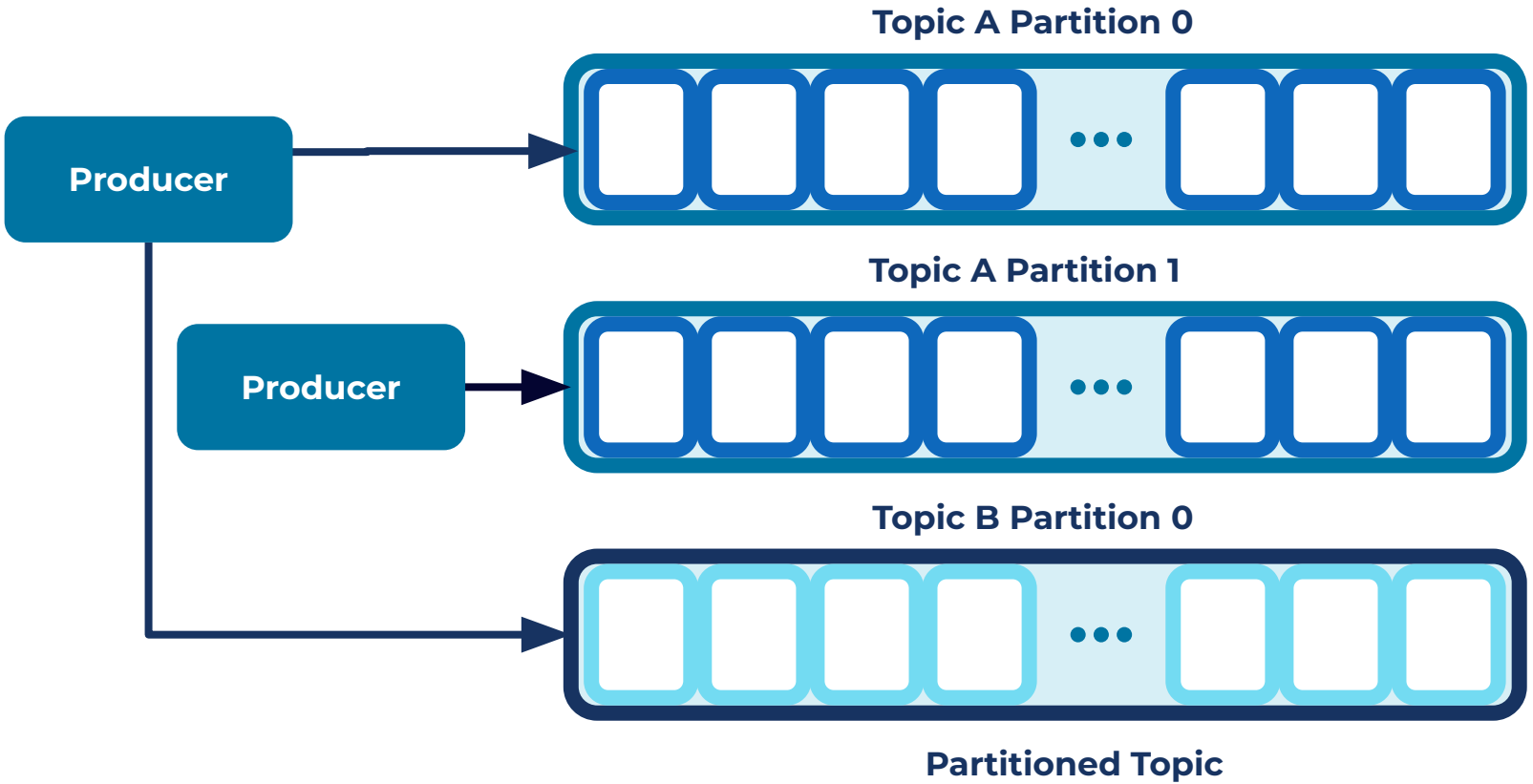
dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia



dfine@confluent.io
@TheDanicaFine

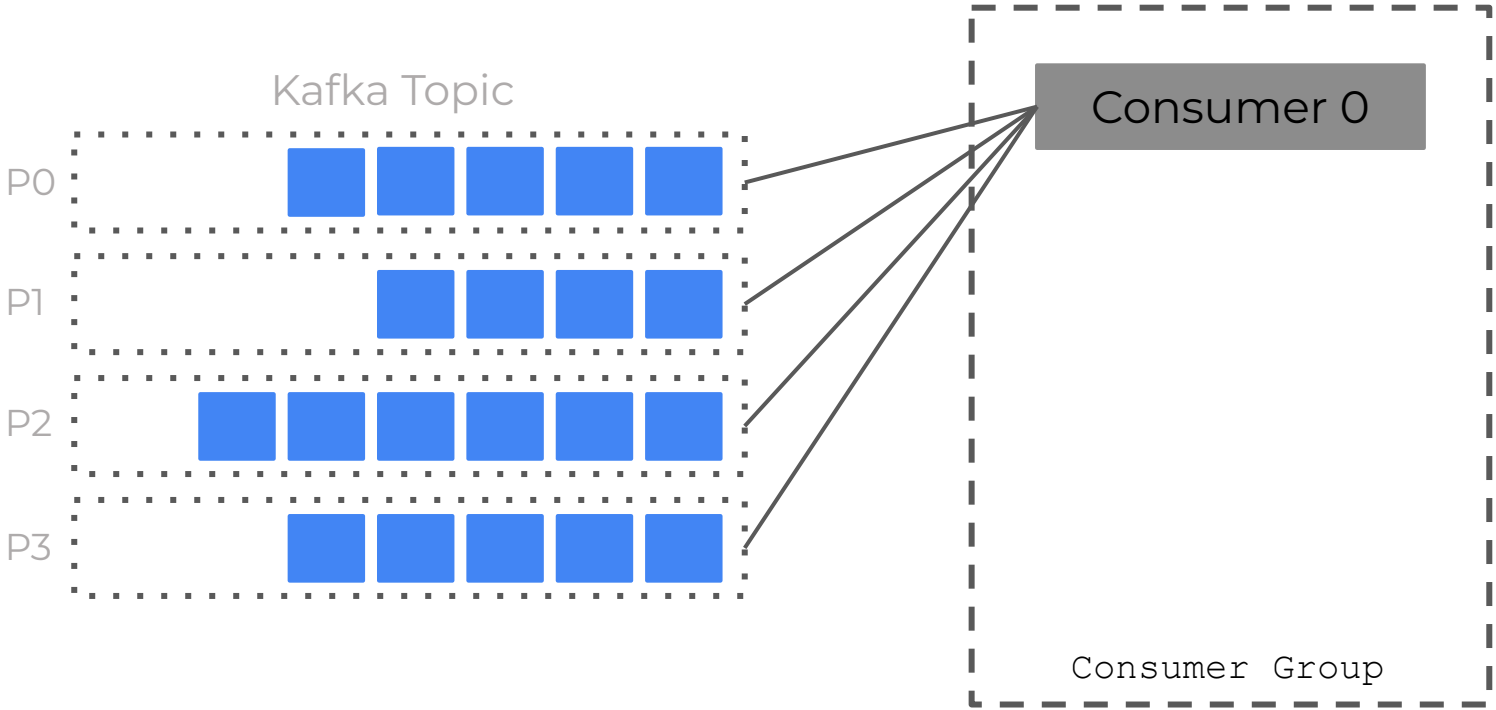
lcerchie@confluent.io
@CerchieLucia



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

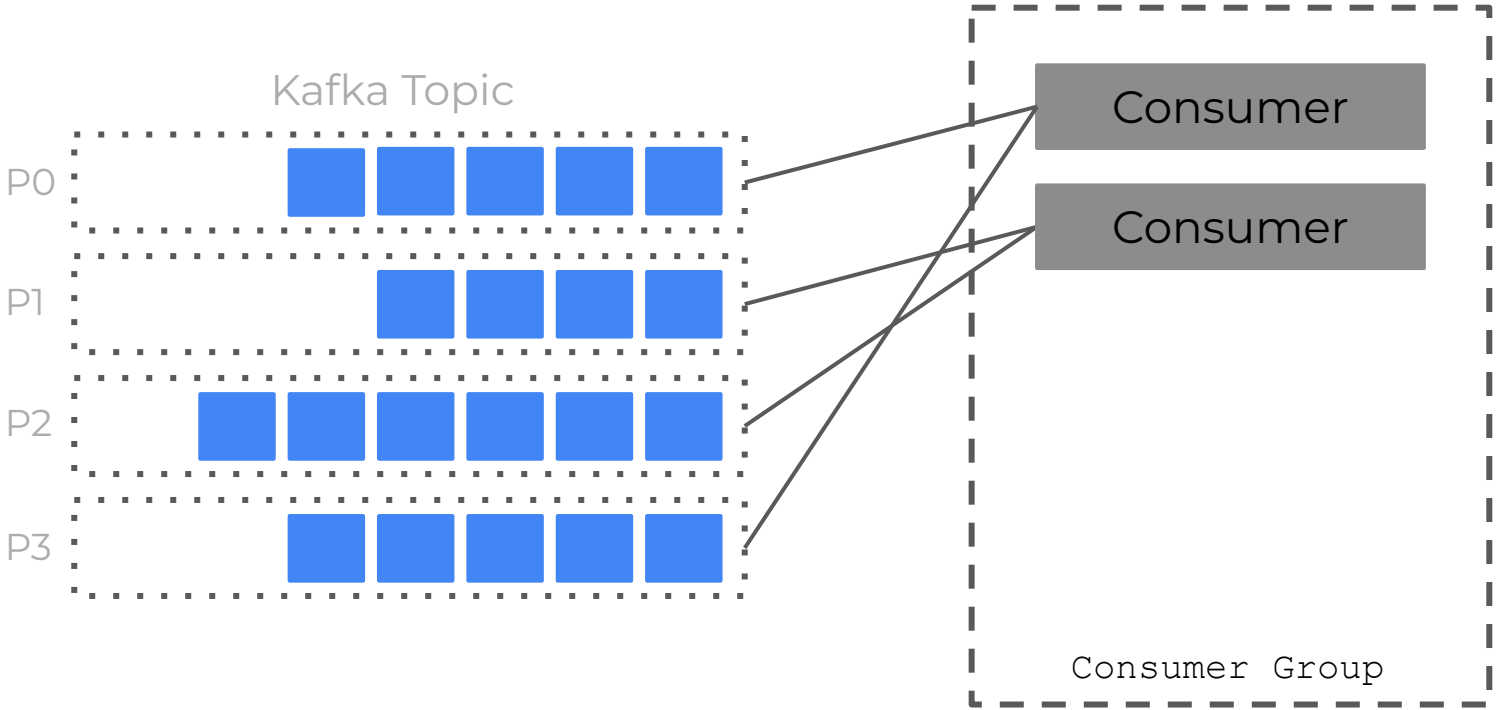
Consumer Groups



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

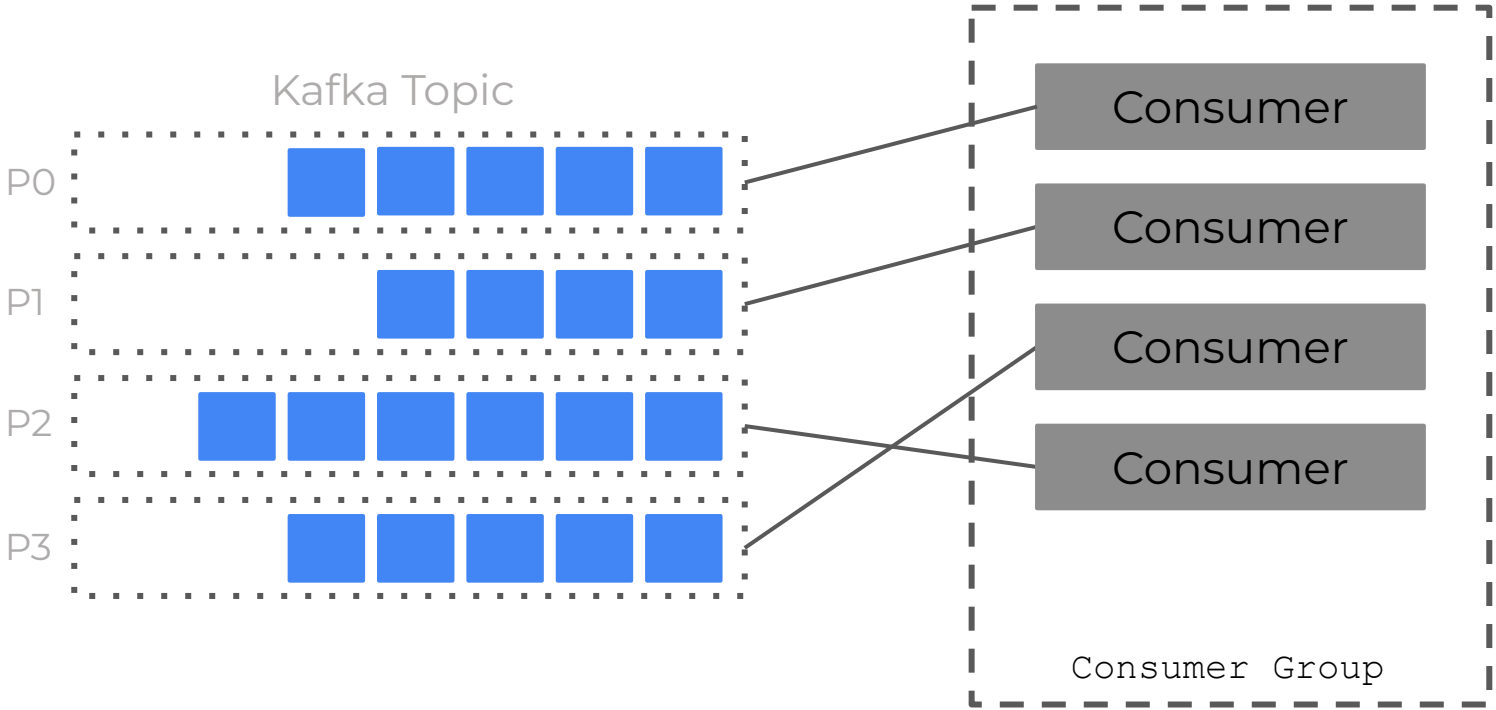
Consumer Groups



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

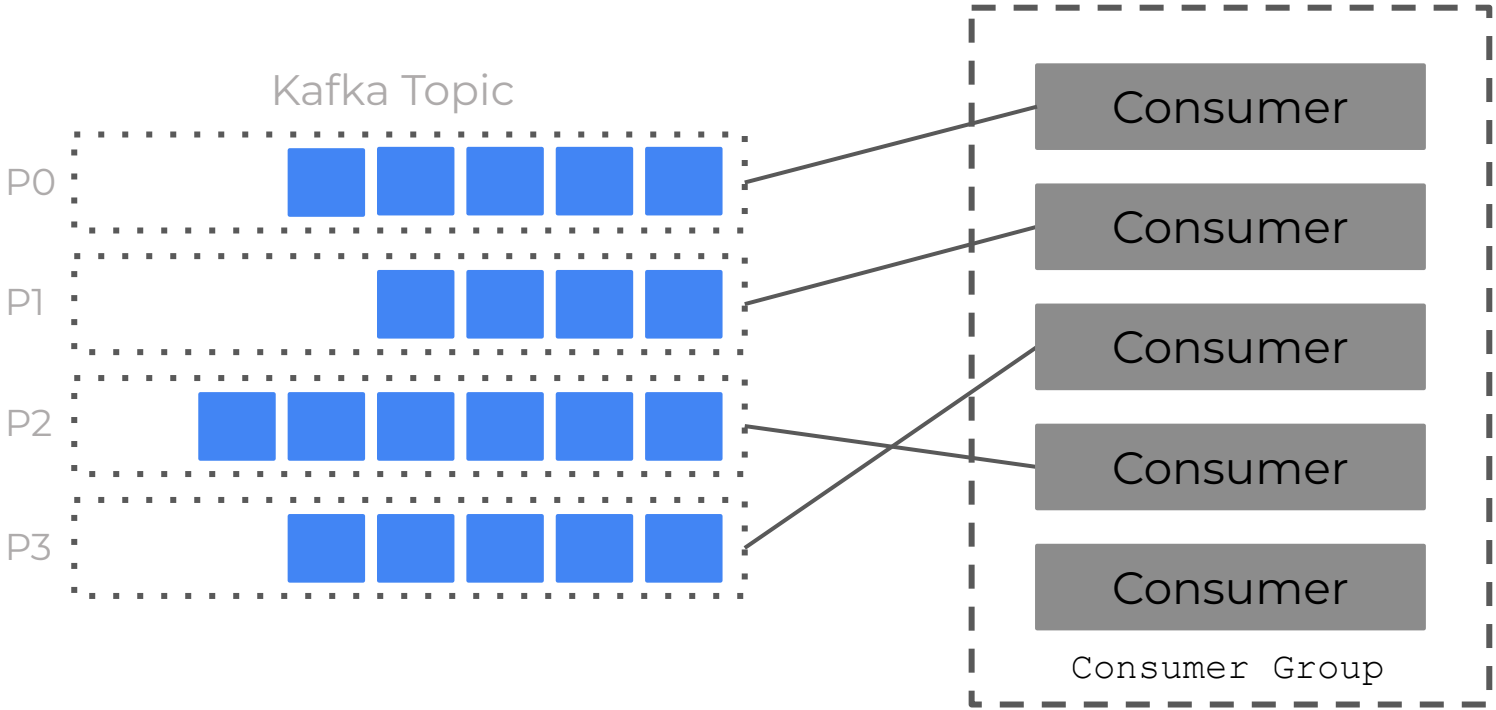
Consumer Groups



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

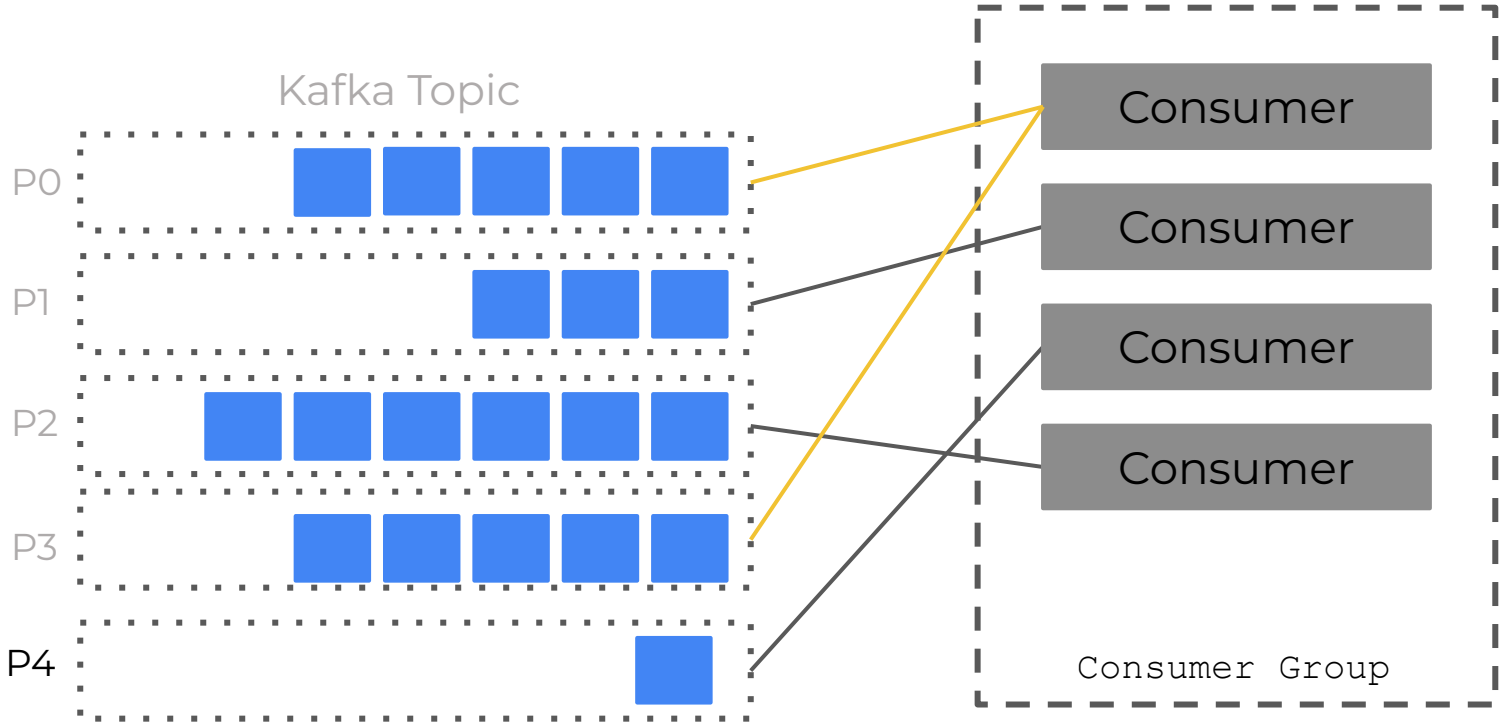
Consumer Groups



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

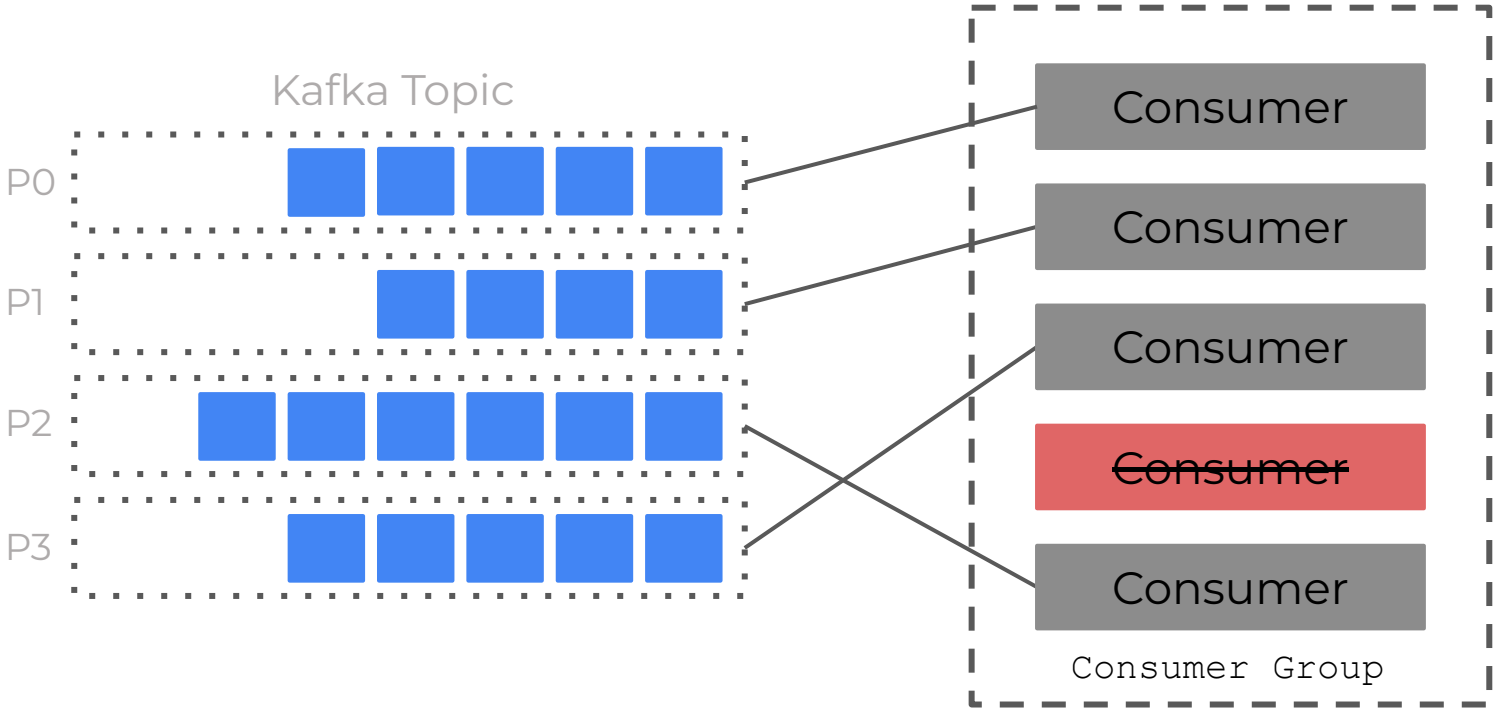
Consumer Groups



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

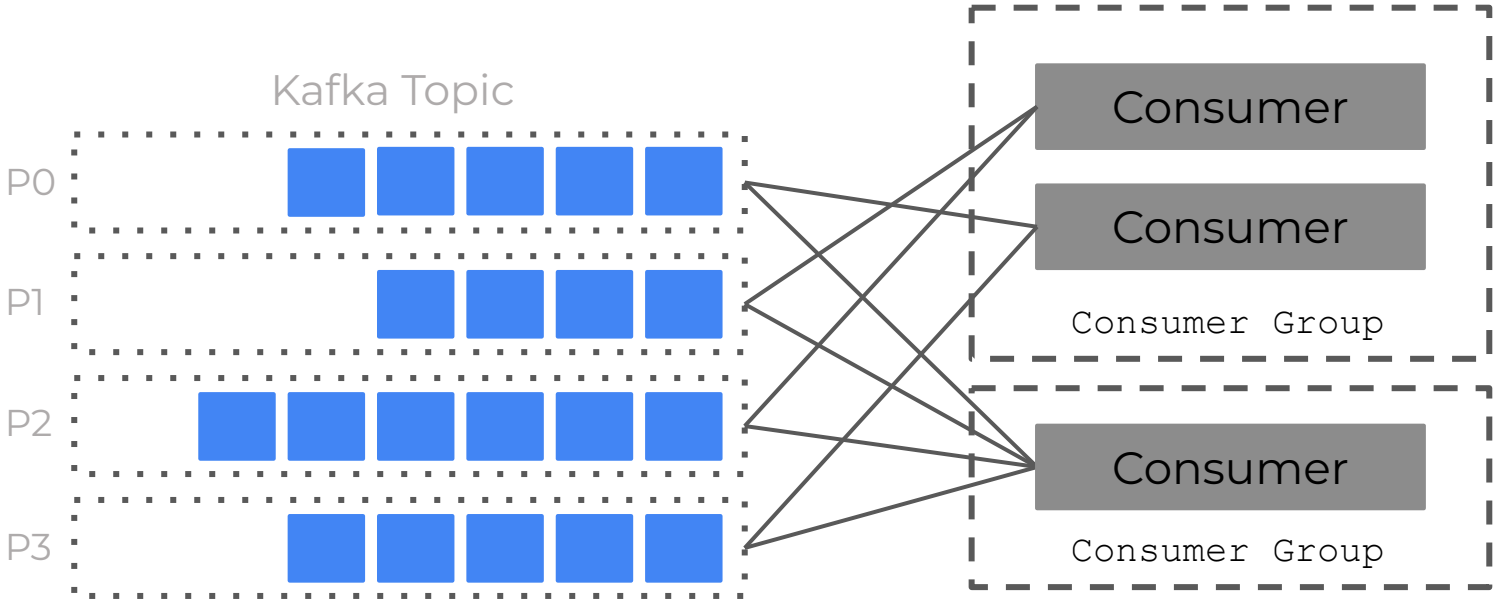
Consumer Groups



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

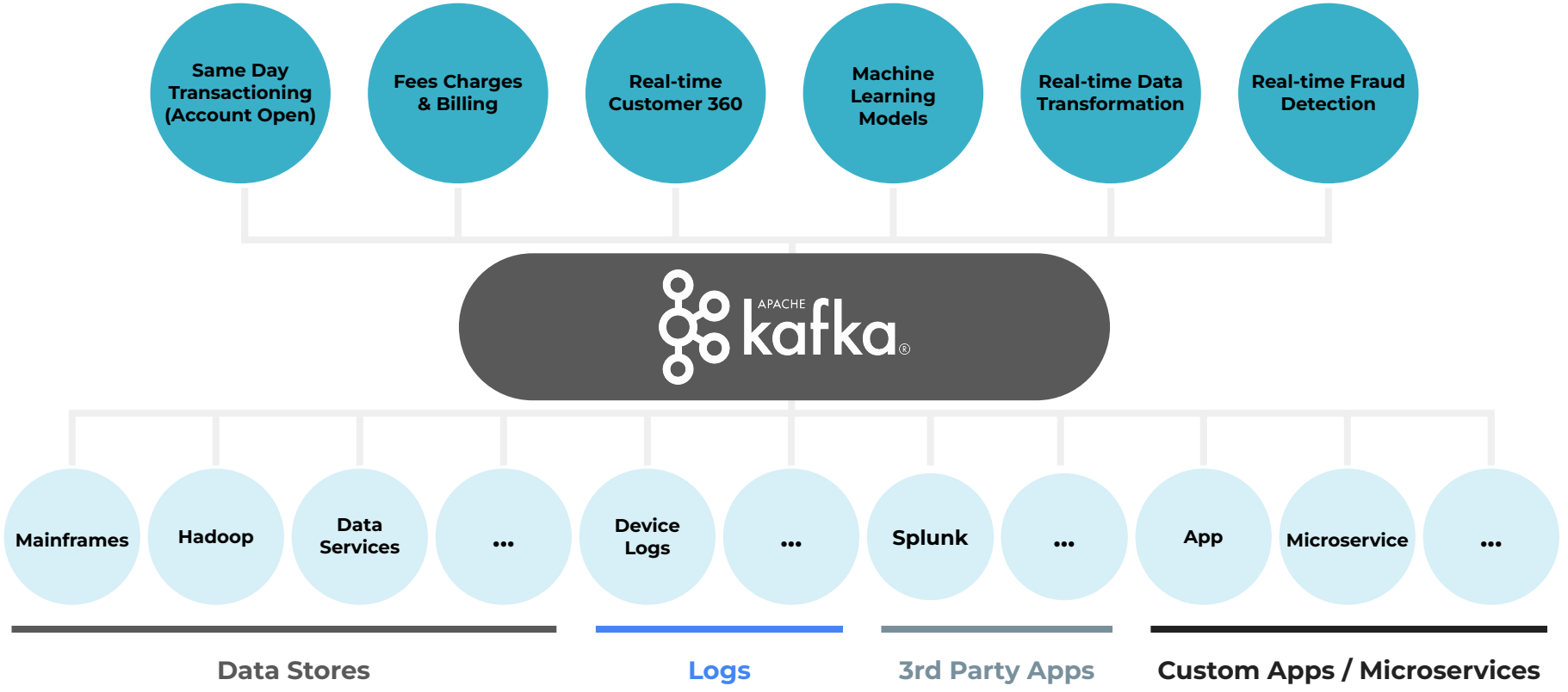
Consumer Groups



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Unified Fabric for App & Data Modernization



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Confluent? What's *that*?

Kafka, but make it easy.

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Confluent

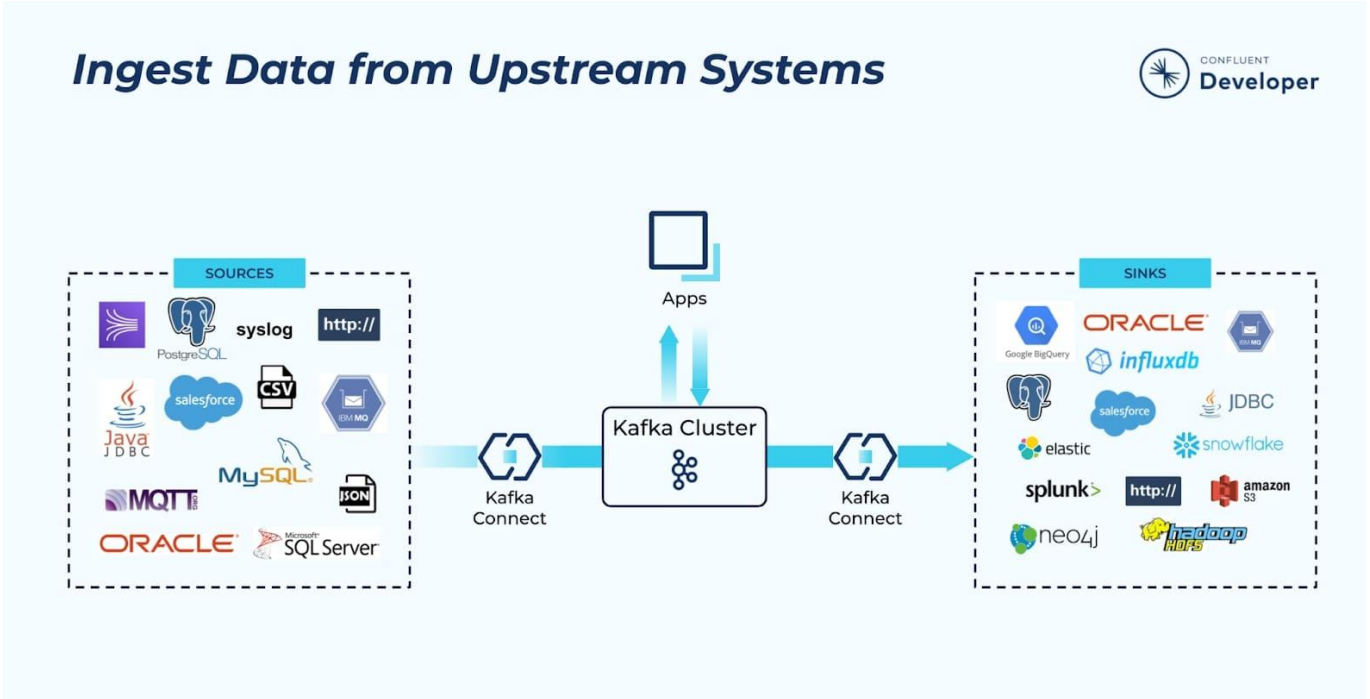
- Fully-managed, cloud-based Kafka
- Auxiliary tools:
  - Kafka Connect
  - ksqlDB
  - Schema management

CONFLUENT

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

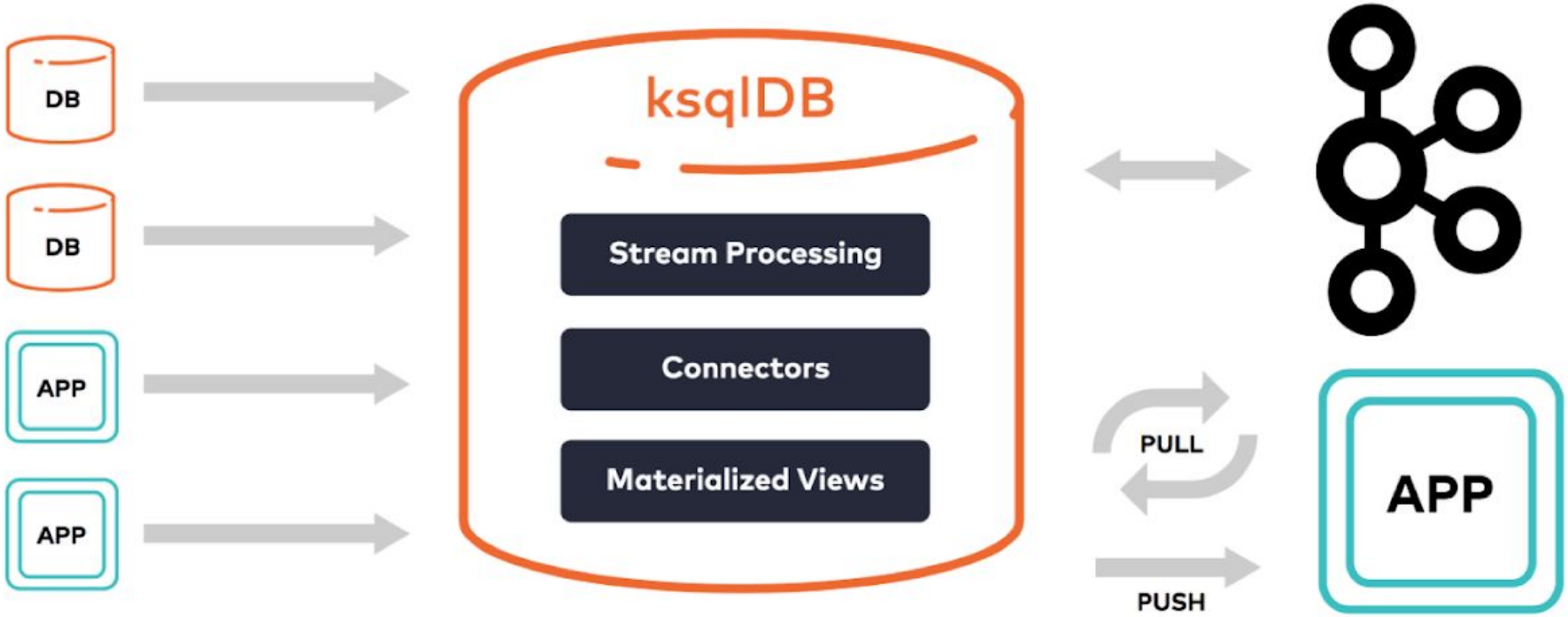
Connectors



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

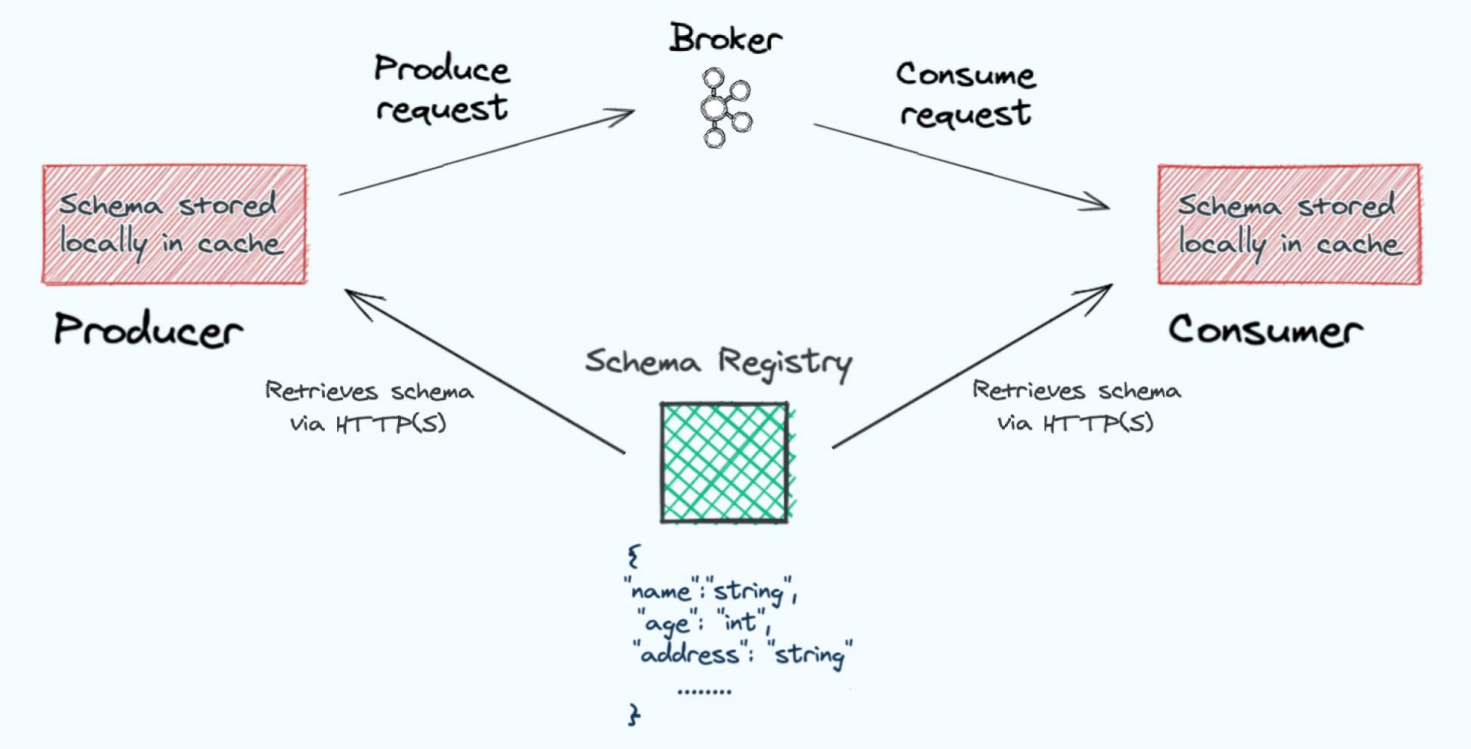
https://developer.confluent.io/learn-kafka/ksqldb/intro/



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Decoupling with producers and consumers



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Confluent

Developer

Unrestricted Developer Productivity

Multi-language Development
non-Java clients • | REST Proxy •

Rich Pre-built Ecosystem
Connectors | Hub • | Schema Registry •

SQL-based Stream Processing
KSQL (ksqlDB) •

Operator

Efficient Operations At Scale

GUI-driven Mgmt & Monitoring
Control Center

Flexible DevOps Automation
Operator | Ansible •

Dynamic Performance & Elasticity
Auto Data Balancer | Tiered Storage

Architect

Production-Stage Prerequisites

Enterprise-grade Security
RBAC | Secrets | Audit logs

Data Compatibility
Schema Registry • | Schema Validation

Global Resilience
Multi-Region Clusters | Replicator

Apache Kafka

• Open Source / Community licensed

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Kafka Installation Process

- Python Client Installation

  ```
  pip install confluent-kafka
  ```

- Bundled with `librdkafka`
- Note: M1 MacBooks ***must*** install from source.

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Confluent Cloud Account

Use Promo Code

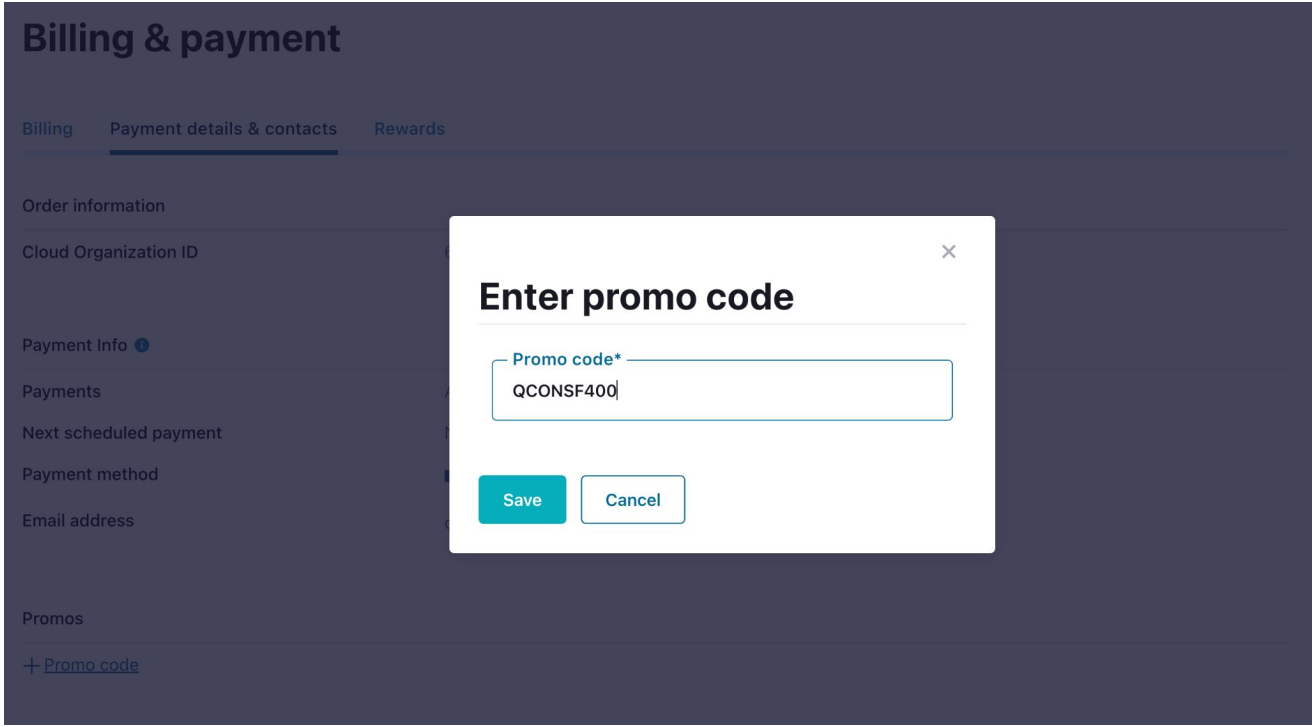
QCONSF400

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

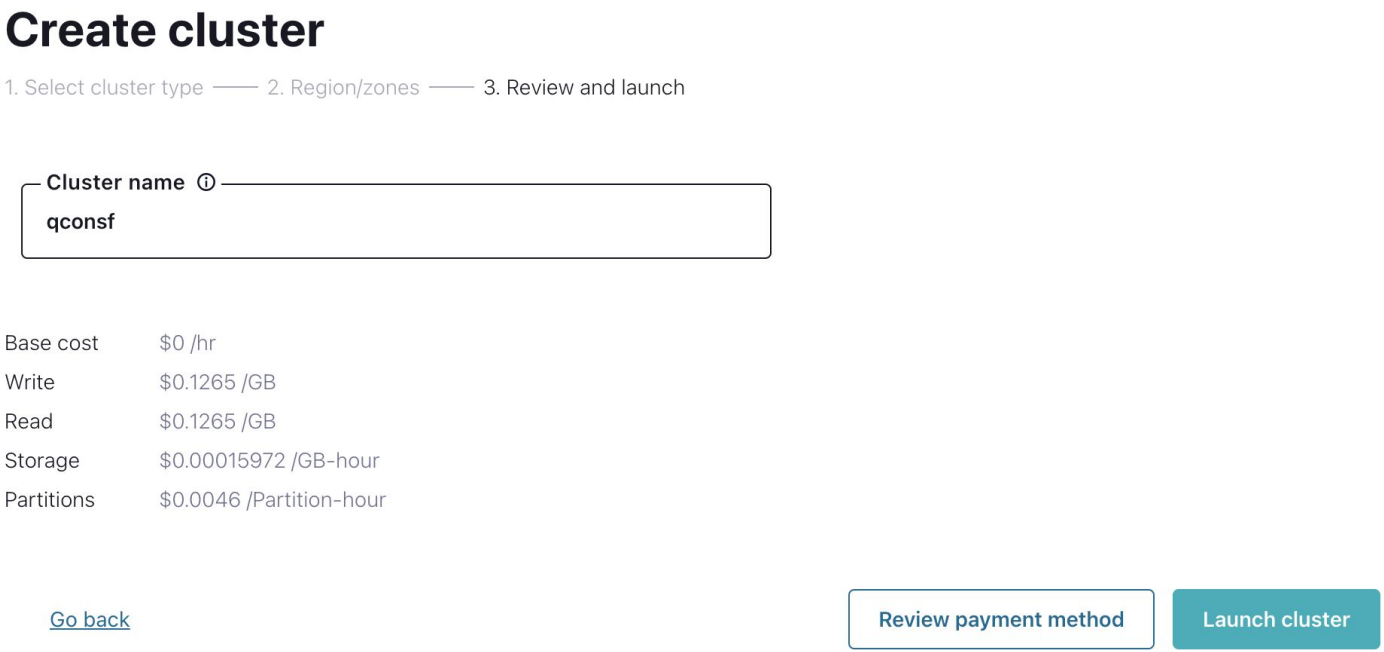
Cloud Account Set Up - Apply Promo Code

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



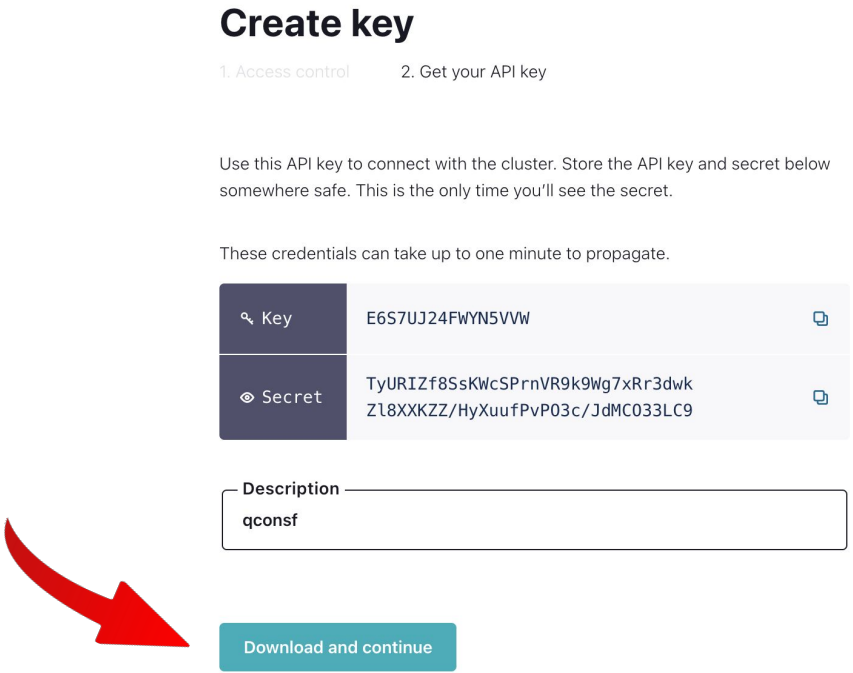
Cloud Account Set Up - Create Cluster

Default Env → + Add cluster → Basic cluster



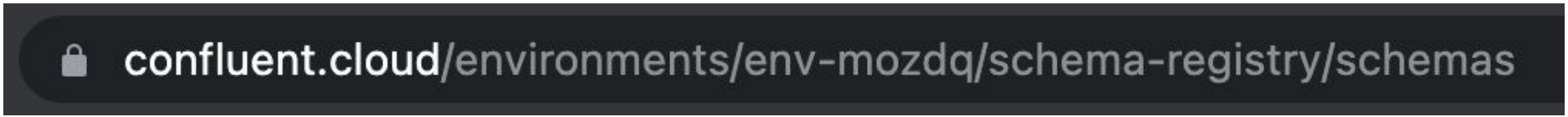
Cloud Account Set Up - Create API Key

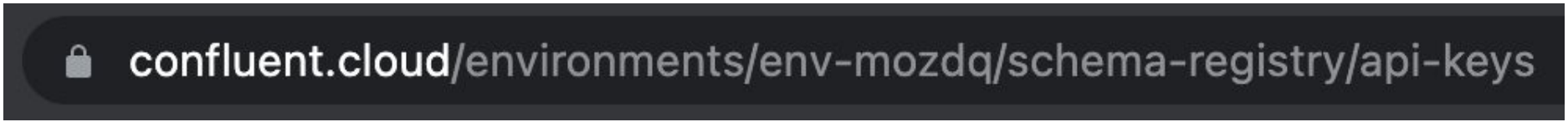
Cluster Overview → + API Keys → + Add key → Global Access



Confluent Cloud - Create SR API Key

Cluster Overview → Schema Registry → Change URL → + API 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

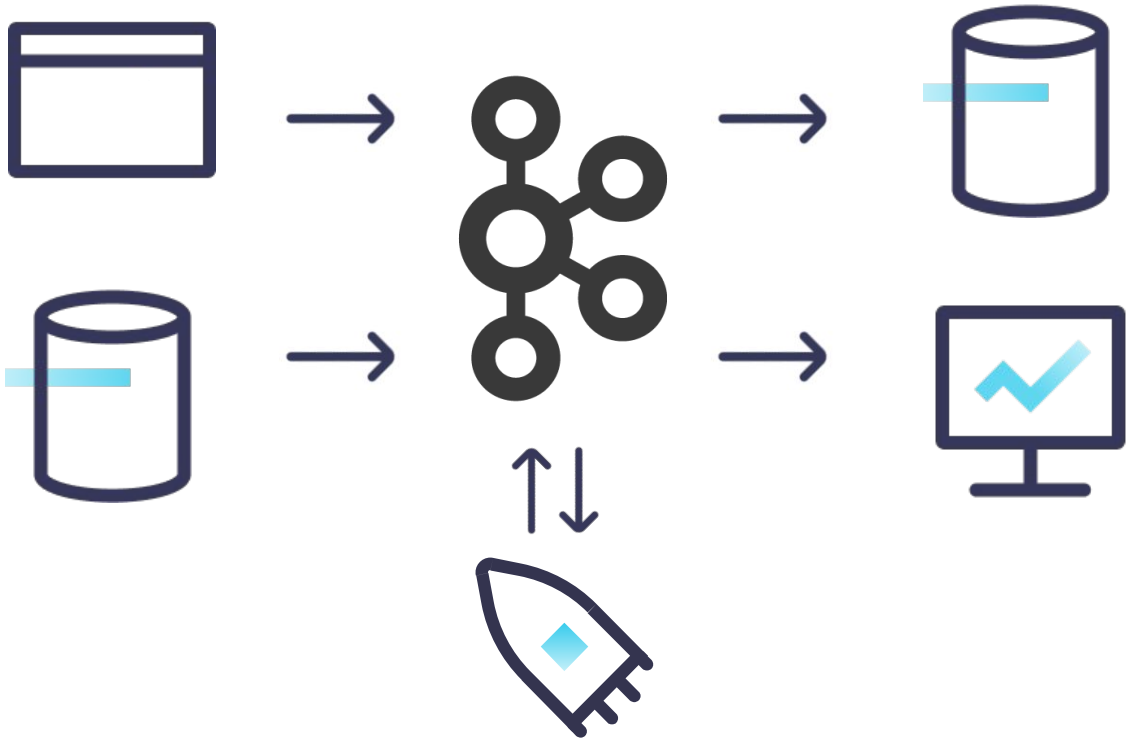
dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Producing Data Into Kafka

Producer API vs Kafka Connect

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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?

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Use Case: Website Analytics

- E-commerce website
- Goal: Assess popularity of products
    - How long do users spend on a product page?
    - Are users from certain regions spending more time on product pages?

Users

Key: User_9


```
{
  "registertime": 1511778995516,
  "userid": "User_9",
  "regionid": "Region_6",
  "gender": "FEMALE"
}
```


dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Pageviews

Key: 49601

```
{
  "viewtime": 49601,
  "userid": "User_6",
  "pageid": "Page_33"
}
```

DataGen Datasets

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

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"
          }
    ]
}
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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"]
            ]
        }
    }
}
```

Example – Int Range

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

dfine@confluent.io
@TheDanicaFine

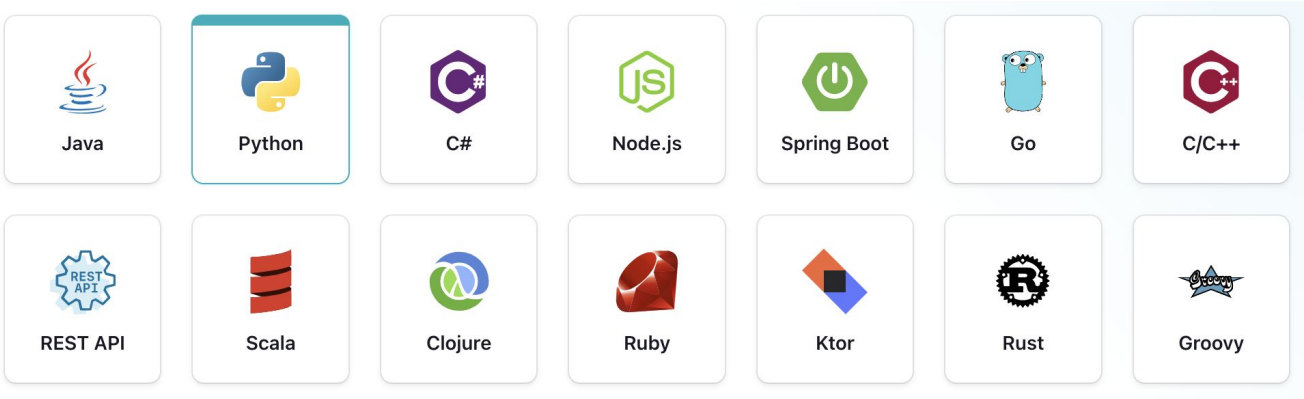
lcerchie@confluent.io
@CerchieLucia

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-2225o.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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Checkpoint Quiz

Which configuration options apply to the following scenarios?

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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 ever 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

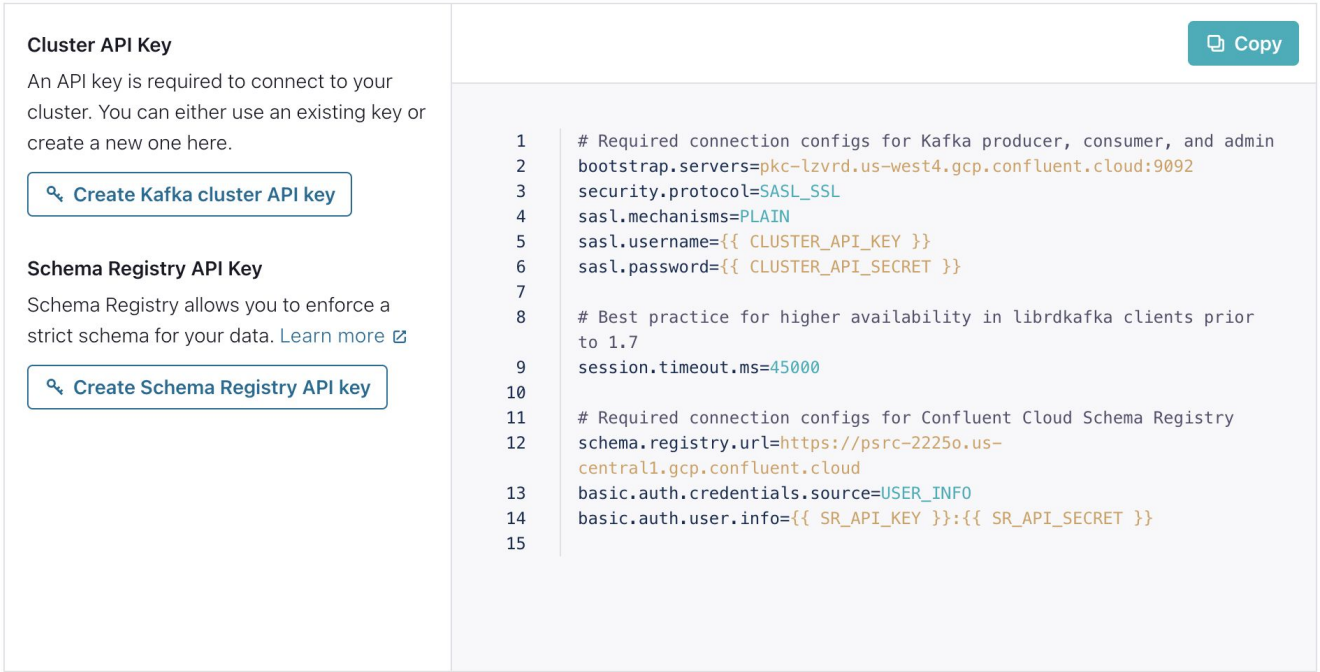
Create a Python Client

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Confluent Cloud - Create Python Client

Cluster Overview → Clients → + New client → Python



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 `users` and `pageviews` data more interesting

https://github.com/danicafine/qconsf-python

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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

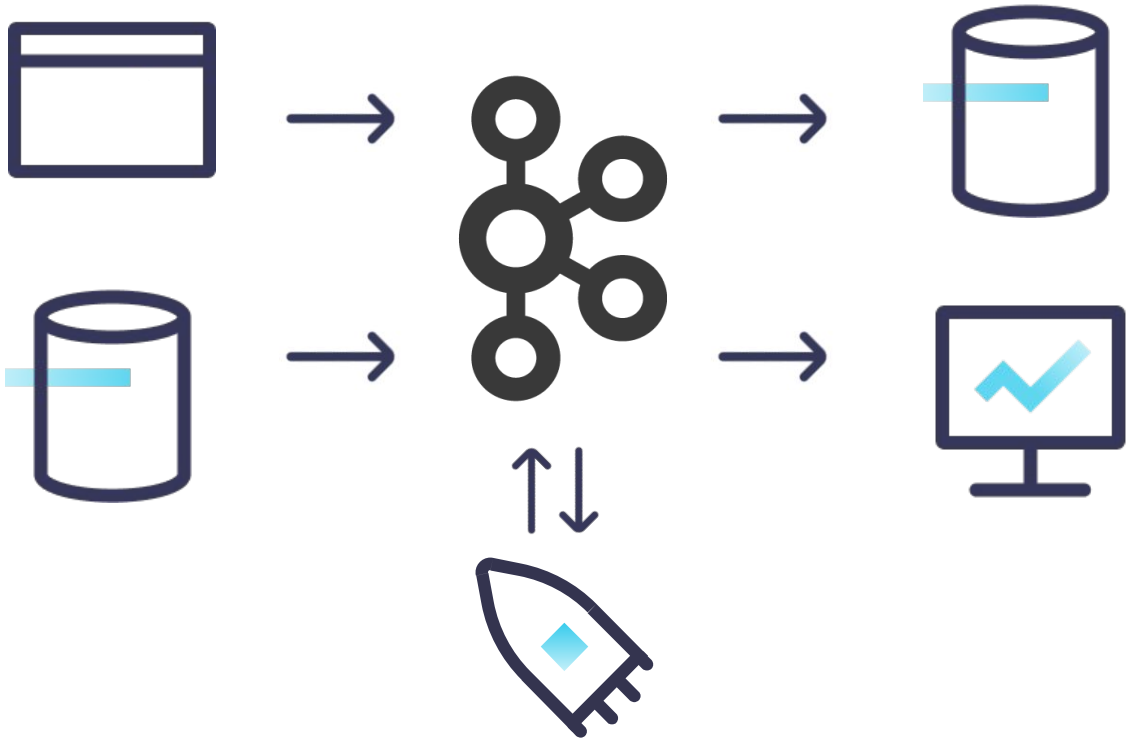
Time	Module Type	Description
12:00	LUNCH	
13:00	Instruction	Stream Processing Techniques in the Apache Kafka Ecosystem ksqlDB in Confluent Cloud
13:45	Hands On	Build a Stream Processing Application with ksqlDB
14:30	COFFEE BREAK	
14:45	Instruction	Advanced Stream Processing Pipelines Use Cases and Inspiration
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



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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 Users
consumer = clients.consumer(clients.user_deserializer(),
                'consumer-group-usermask', [config['topics']['users']])

# set up Kafka Producer for UserMask
producer = clients.producer(clients.usermask_serializer())

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

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

        # transform the data
        usermask = UserMask(user.registertime, user.userid, user.regionid)

        # send data to Kafka
        producer.produce(config['topics']['usermasks'], key=str(user.userid), value=usermask)
```

Kafka Streams

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

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Example: Kafka Streams

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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 usermasks
    WITH (kafka_topic='usermasks', value_format='avro', partitions=1) AS
    SELECT
        REGISTERTIME,
        USERID,
        REGIONID,
        MASK(GENDER)
    FROM USERS;
```

ksqlDB

Use Case: Website Analytics

- Register data: `Stream` or `Table`?
- Pre-processing:
  - Filtering out unnecessary data
  - Enrichment
- Aggregation

dfine@confluent.io
@TheDanicaFine

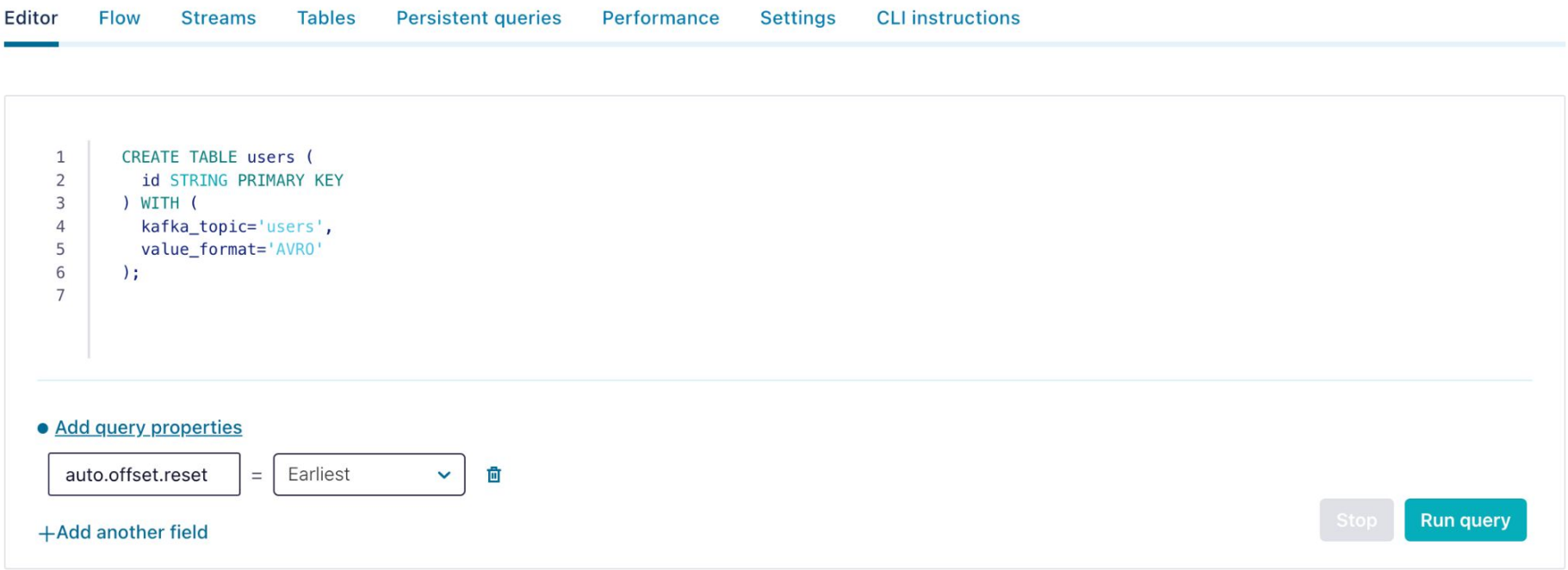
lcerchie@confluent.io
@CerchieLucia

Creating a Table

```
CREATE TABLE users (
  id STRING PRIMARY KEY
) WITH (
  kafka_topic='users',
  value_format='AVRO'
);
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Creating a Stream

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Rekeying a Stream

```
CREATE STREAM pageviews_rekeyed WITH (
  kafka_topic='pageviews-rekeyed',
  value_format='AVRO'
) AS
SELECT
  *
FROM PAGEVIEWS
PARTITION BY USERID;
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Filtering a Stream

```
CREATE STREAM pageviews_filtered WITH (
  kafka_topic='pageviews-filtered',
  value_format='AVRO'
) AS
SELECT
  ROWTIME AS pageviewtime,
  *
FROM PAGEVIEWS_REKEYED
WHERE USERID != 'User_1'
EMIT CHANGES;
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Stream-to-Table Join

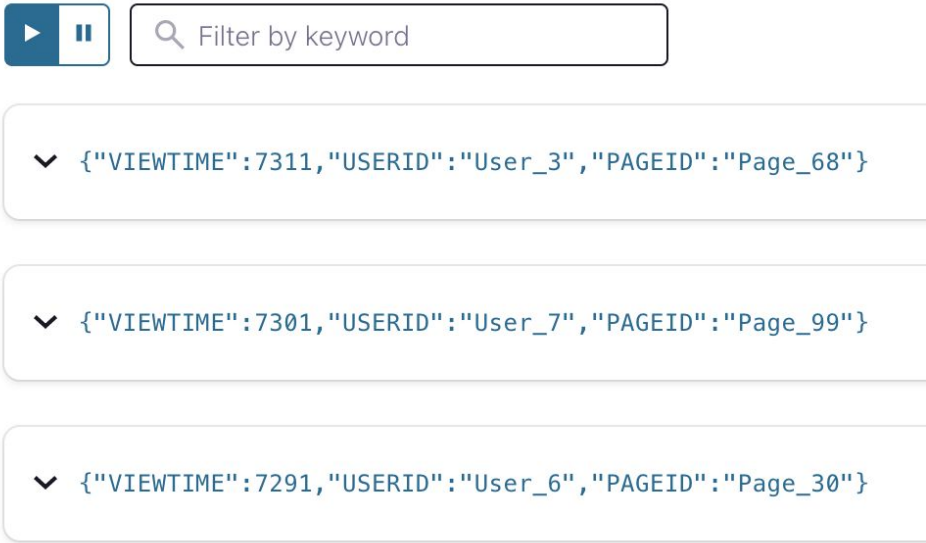
```
CREATE STREAM pageviews_enriched WITH (
  kafka_topic='pageviews-enriched',
  value_format='AVRO'
) AS
SELECT
  pageviews.viewtime                        AS viewtime,
  FROM_UNIXTIME(pageviews.pageviewtime)     AS pageviewtime,
  pageviews.pageid                          AS pageid,
  pageviews.userid                          AS userid,
  users.regionid                            AS regionid,
  users.gender                              AS gender,
  FROM_UNIXTIME(users.registertime)         AS registertime
FROM pageviews_filtered AS pageviews
INNER JOIN users
ON pageviews.userid = users.id
EMIT CHANGES;
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Pull Query

```
SELECT
   *
FROM pageviews
WHERE userid != User_0';
```



dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Windowing

```
CREATE TABLE pageview_regional_analysis WITH (
  kafka_topic='pageview-regional-analysis',
  format='AVRO'
) AS
SELECT
  regionid,
  COUNT(*) AS views,
  AVG(viewtime) AS viewtme_avg
FROM pageviews_enriched
WINDOW TUMBLING (SIZE 30 MINUTES)
GROUP BY regionid
EMIT FINAL;
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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

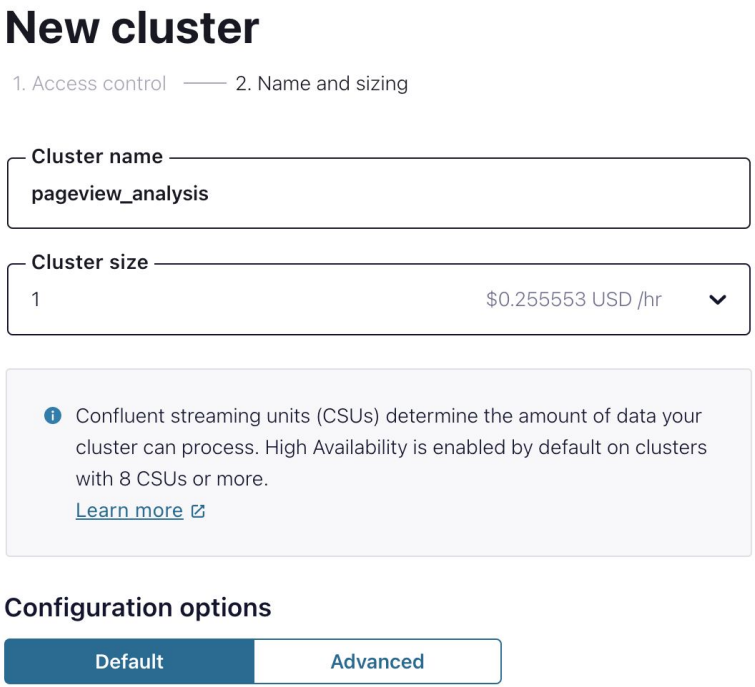
Stream Processing With ksqlDB

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

ksqlDB - Create a Cluster

Cluster Overview → ksqlDB → Create with Tutorial → Follow Prompts



ksqlDB Stream Processing

- Use Case: Website Analytics
  - Register data: `Stream` or `Table`?
  - Pre-processing:
    - Filtering out unnecessary data
    - Enrichment
  - Aggregation
- Another use case?

ksqlDB Reference Materials

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

Use Cases and Inspiration

Online Multiplayer System

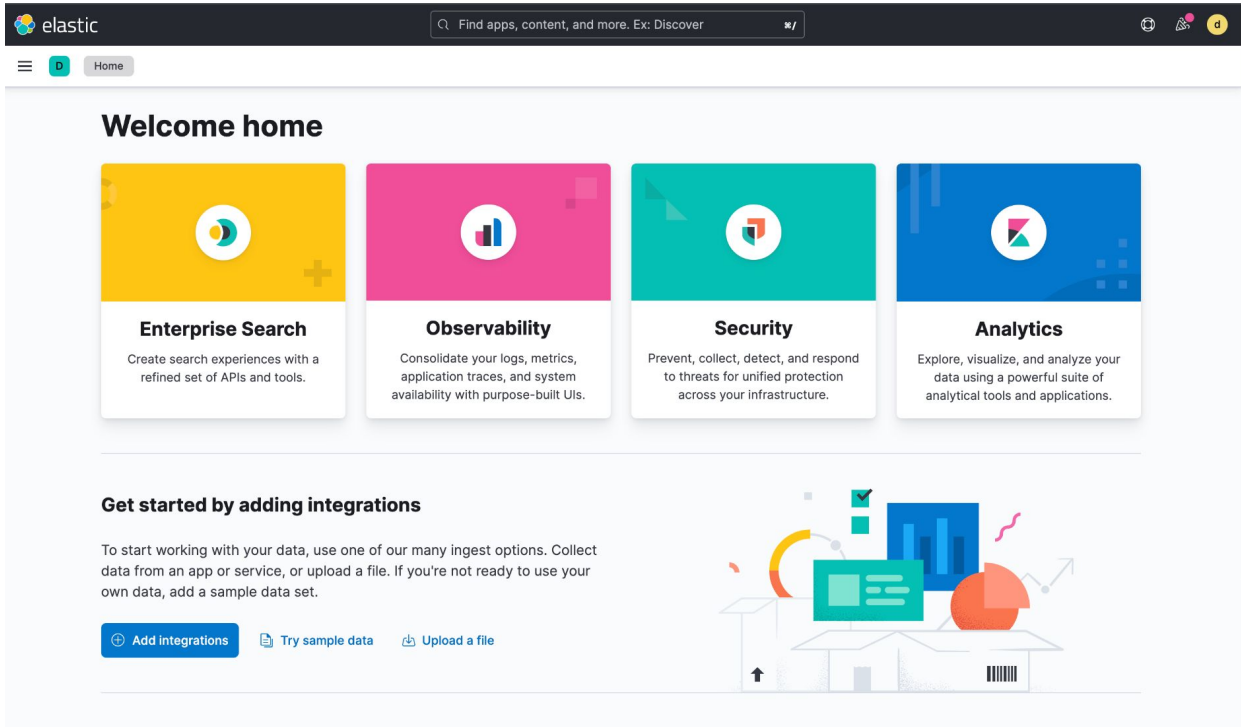
- Kafka forms the backbone of the system
  - All user-input is treated as an event
  - Allows for easy conflict-resolution
- Wide variety of applications
  - Gaming
  - Collaborative work tools (think Google Docs)

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

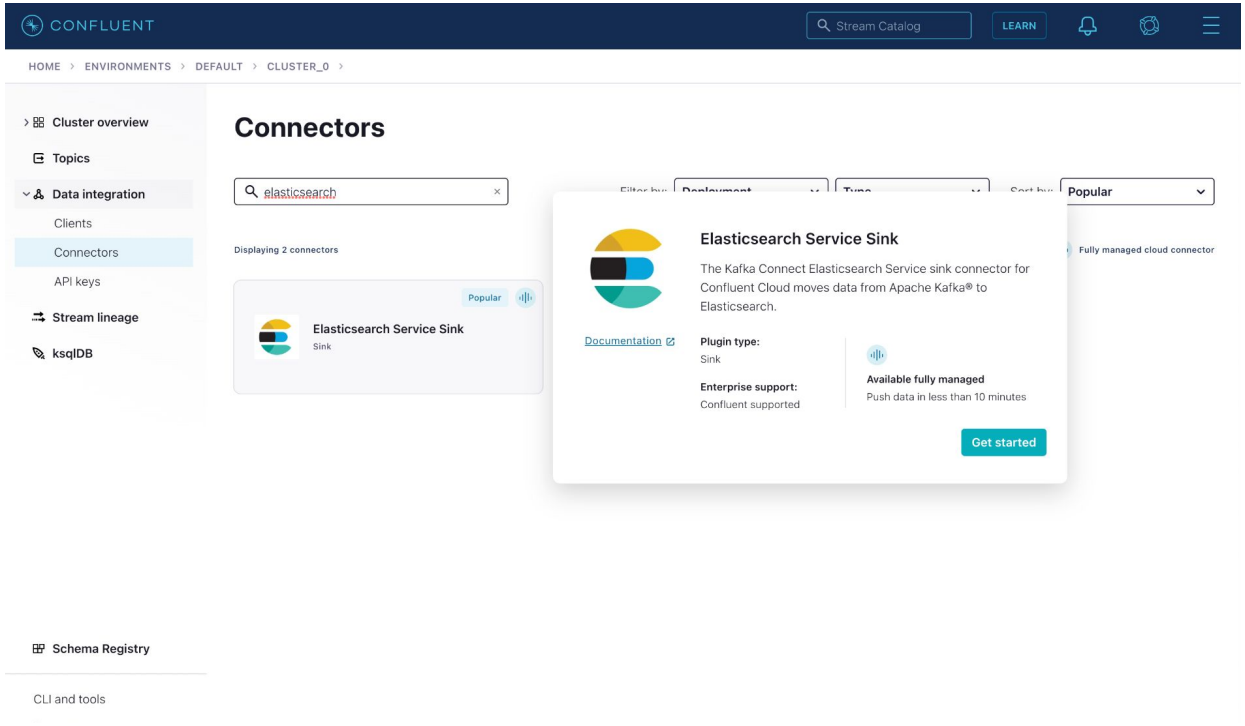
Kafka Connect - Elastic Sink Connector

Create a free account



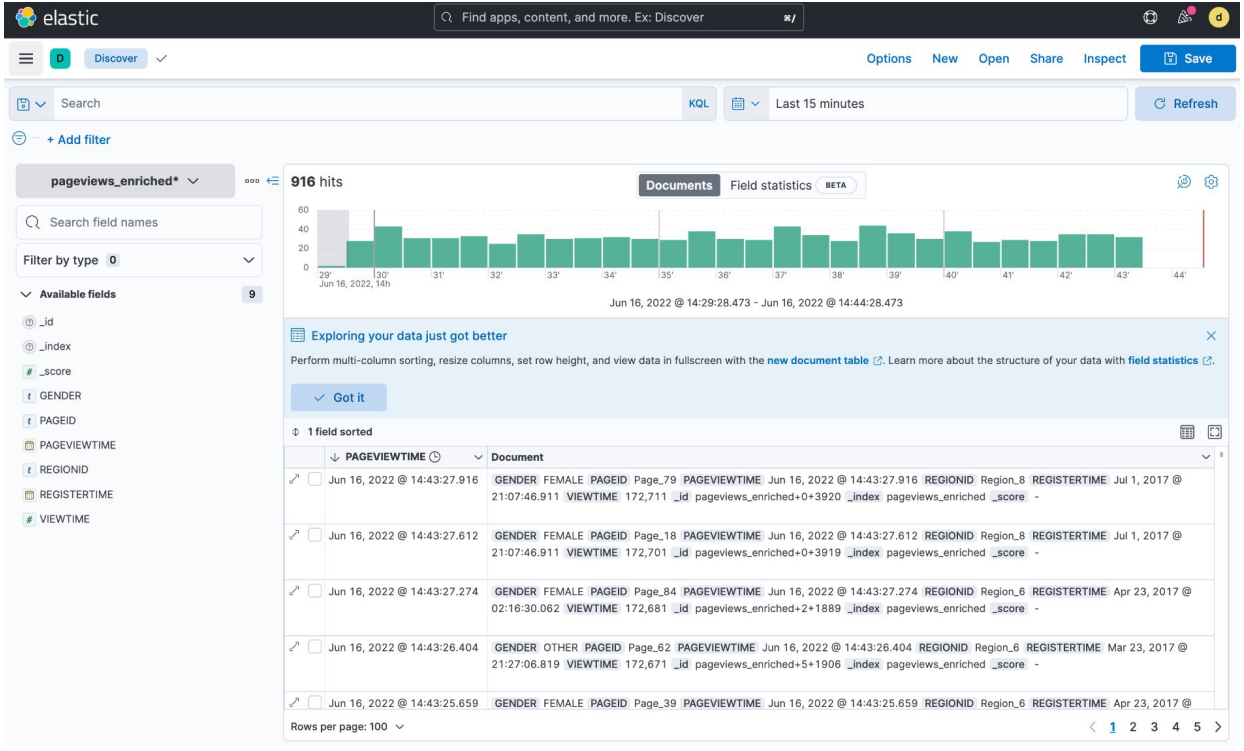
Kafka Connect - Elastic Sink Connector

Configure Elastic Sink Connector



Kafka Connect - Elastic Sink Connector

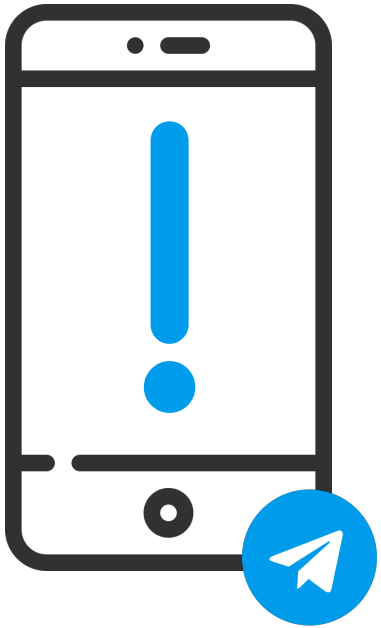
Conduct analysis



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
- Network Rail - https://datafeeds.networkrail.co.uk/ntrod/login;jsessionid=1C499ED60E92B2BCD2332267555EE21B

DataGen Schema String – 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"
          }
     ]
}
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

DataGen Schema String – 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"]
            ]
        }
    }
}
```

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

DataGen Schema String – Int Range

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

What are your use cases?

How can we implement them now?

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia

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

dfine@confluent.io
@TheDanicaFine

lcerchie@confluent.io
@CerchieLucia