Creating an Apache Kafka® Streaming Data Pipeline

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Danica Fine
https://linktr.ee/thedanicafine

dfine@confluent.io
@TheDanicaFine

Amanda Gilbert
https://github.com/amanda010792

agilbert@confluent.io
@aelizabeth92

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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

What’s a data pipeline?

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

What's a *streaming* data pipeline?

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

streaming data pipeline

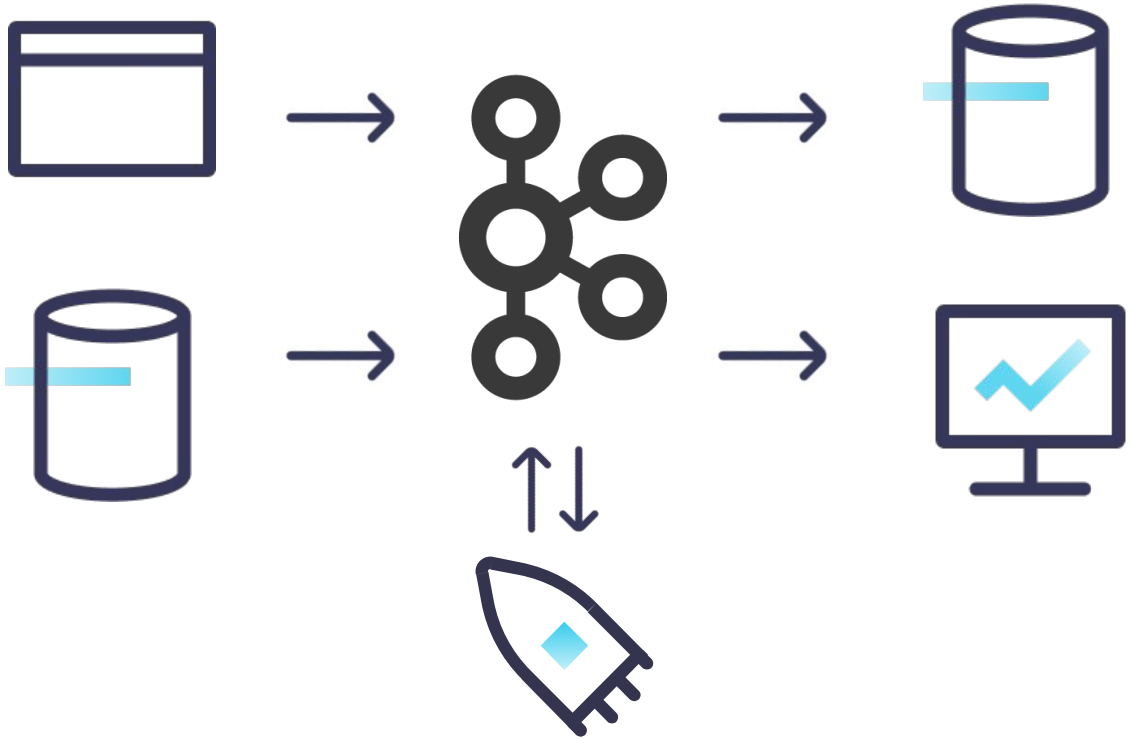
noun \ ˈstrē-miŋ ˈdā-tə ˈpīp-ˌlīn \

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

agilbert@confluent.io
@aelizabeth92

Streaming Data Pipeline



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Streaming Data Pipelines

Who cares?

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Benefits of Stream Processing

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Stream Processing

What are your use cases?

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Kafka? What's *that*?

A distributed event streaming platform.

dfine@confluent.io
@TheDanicaFine

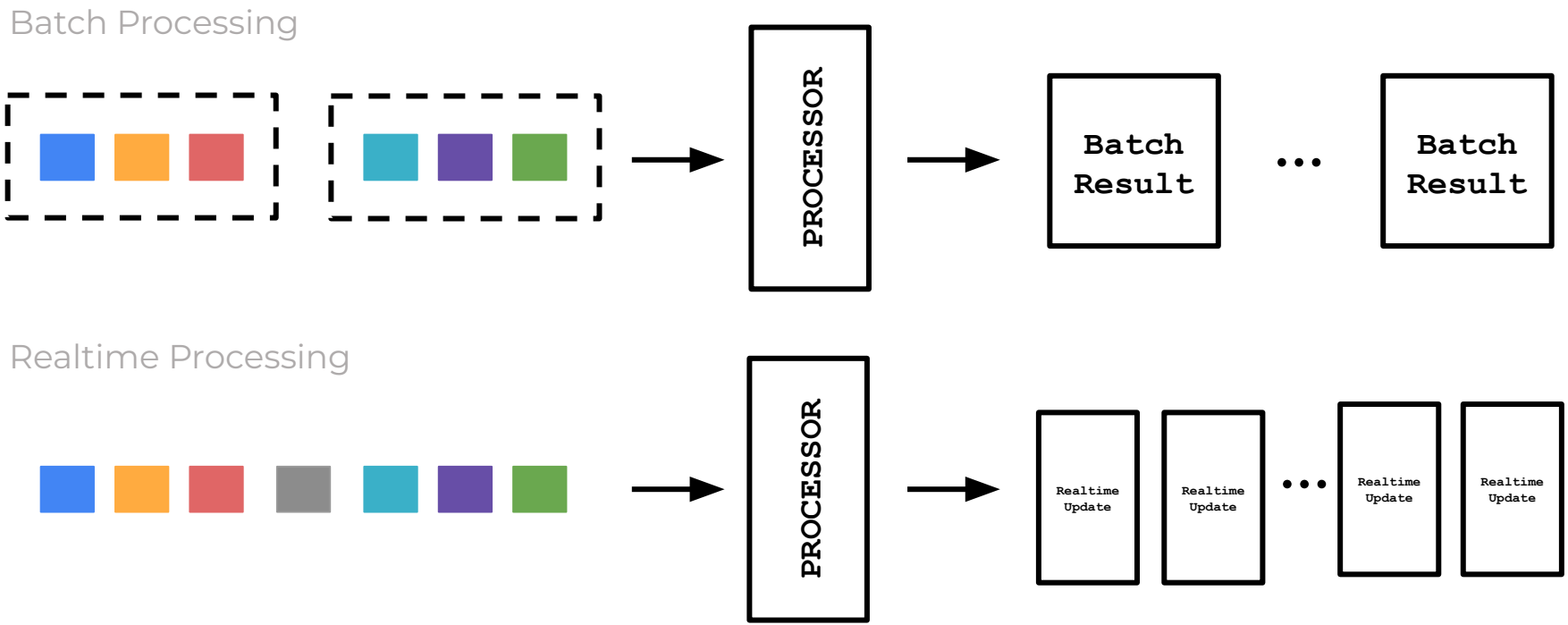
agilbert@confluent.io
@aelizabeth92

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Paradigm Shift



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

.

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

Events

Value Header Key

```
{
  "viewtime": 111,
  "userid": "User_8",
  "pageid": "Page_54"
}
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Events

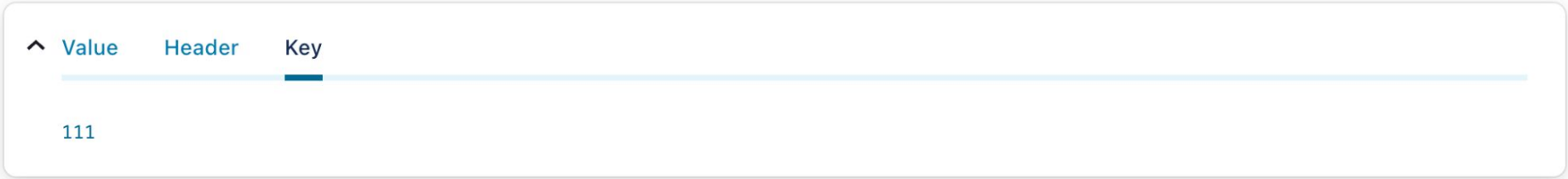
Value | Header | Key


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


dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

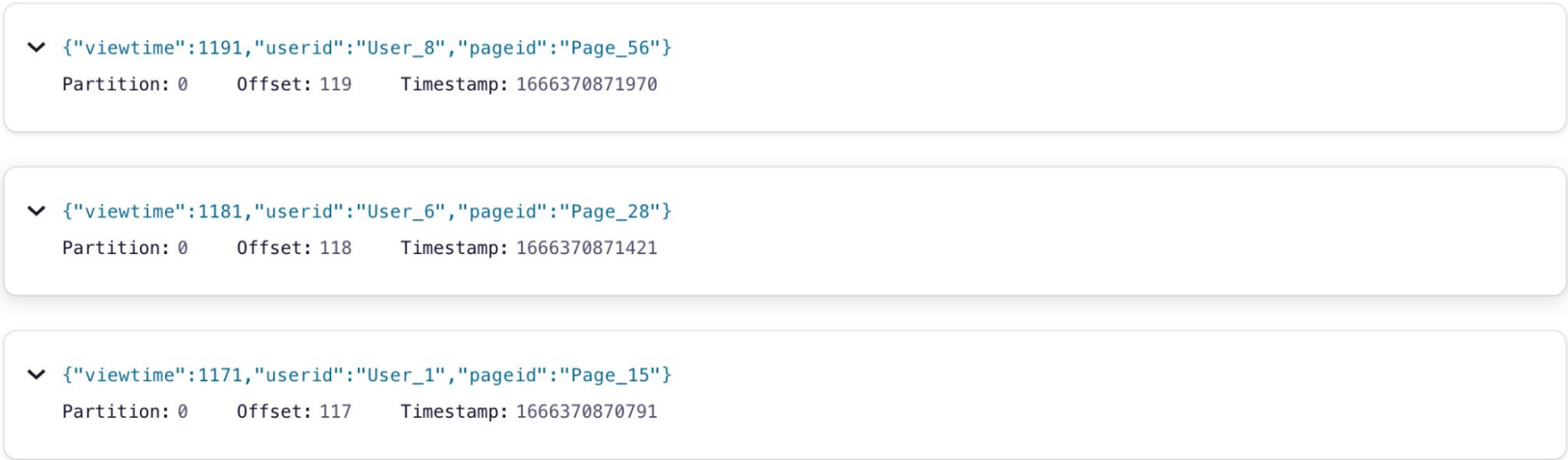
Events



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Events



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

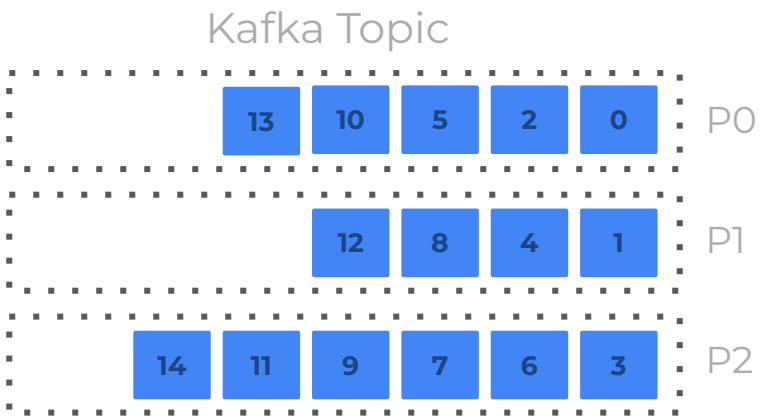
A ***distributed*** event streaming platform.

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Kafka Storage

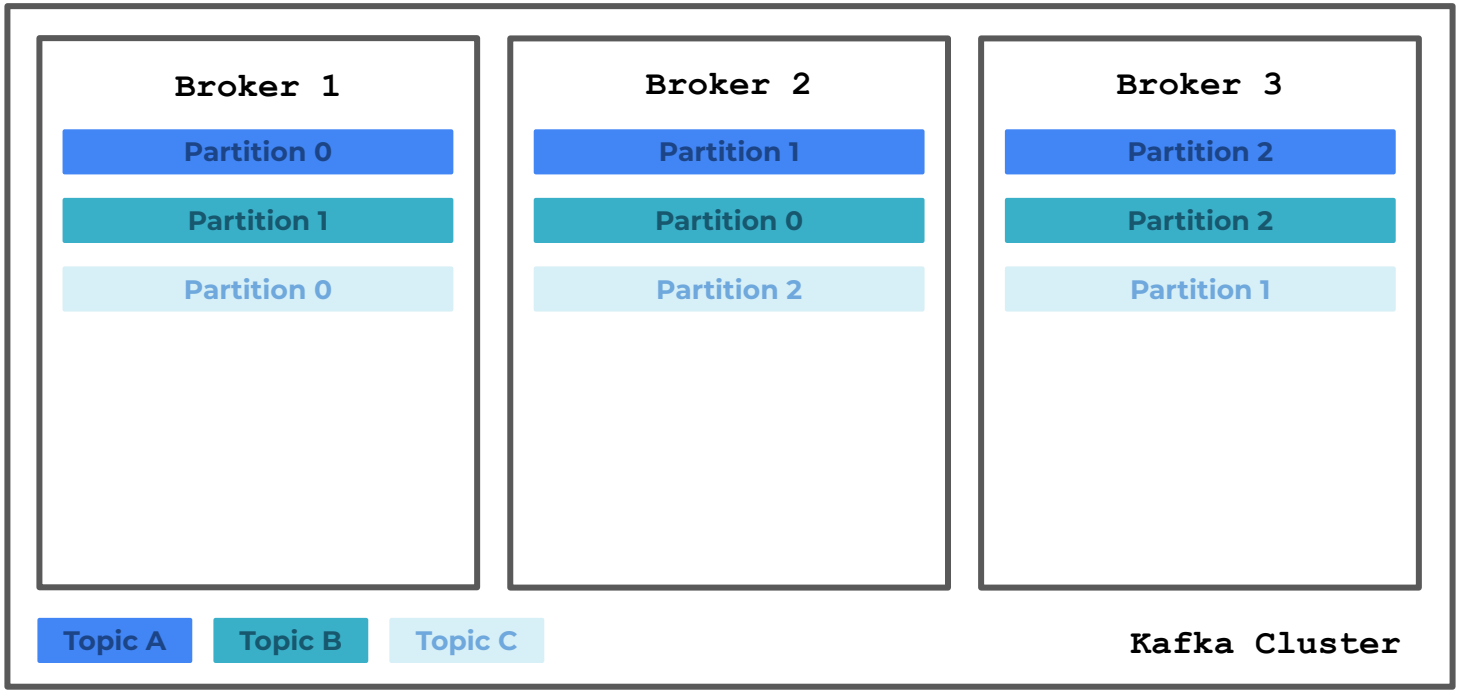
- Topics
  - Basic storage unit
  - Read/Write:
    - Producer and consumer clients
    - Completely decoupled
- Partitions
  - Immutable, append-only logs
  - Data is replicated at this level



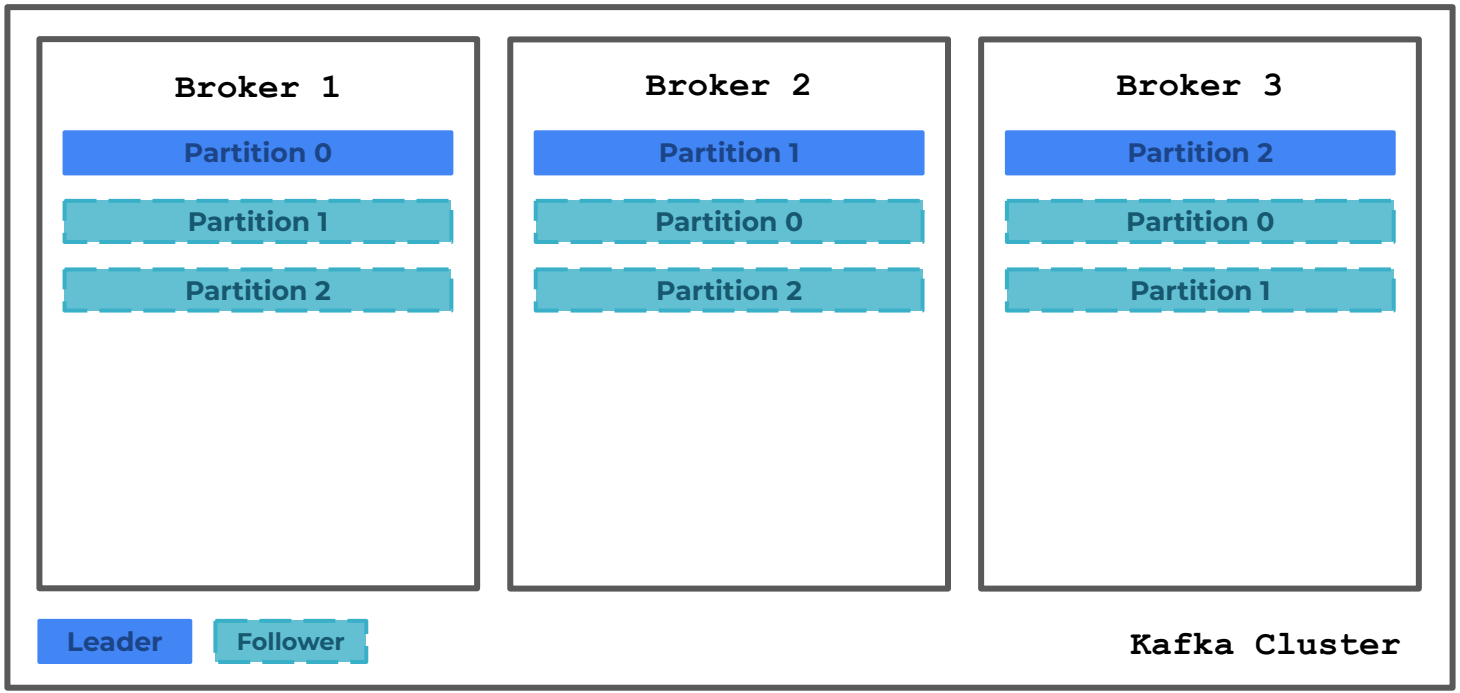
dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Kafka Cluster



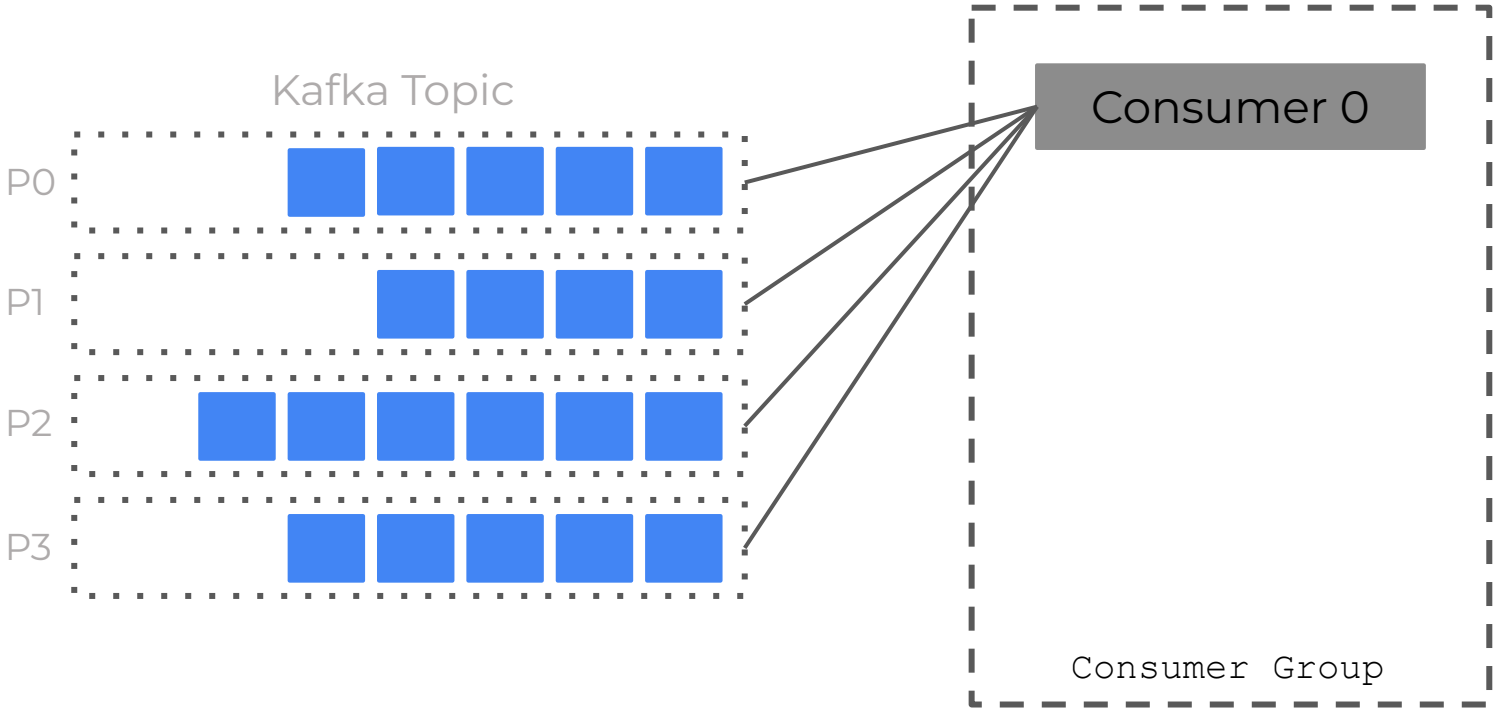
Kafka Cluster



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

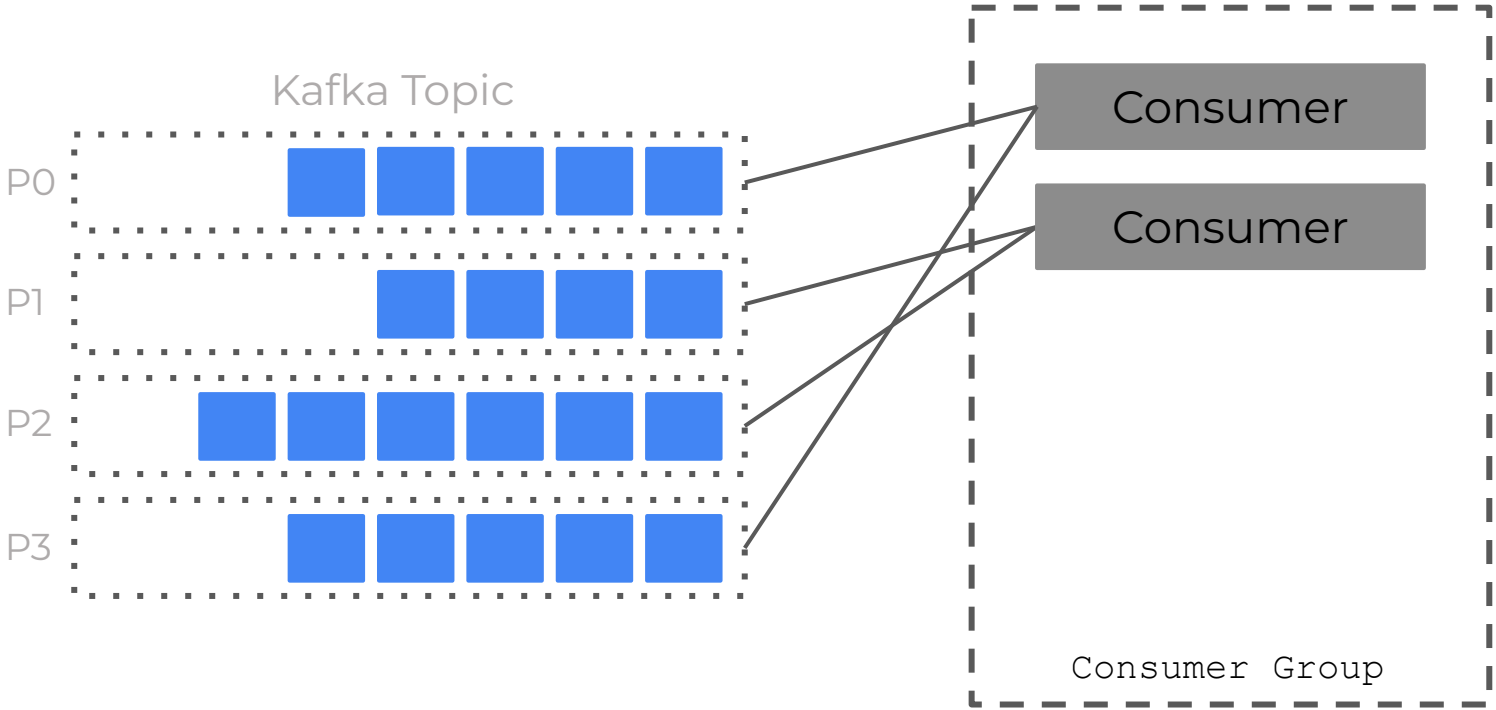
Consumer Groups



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

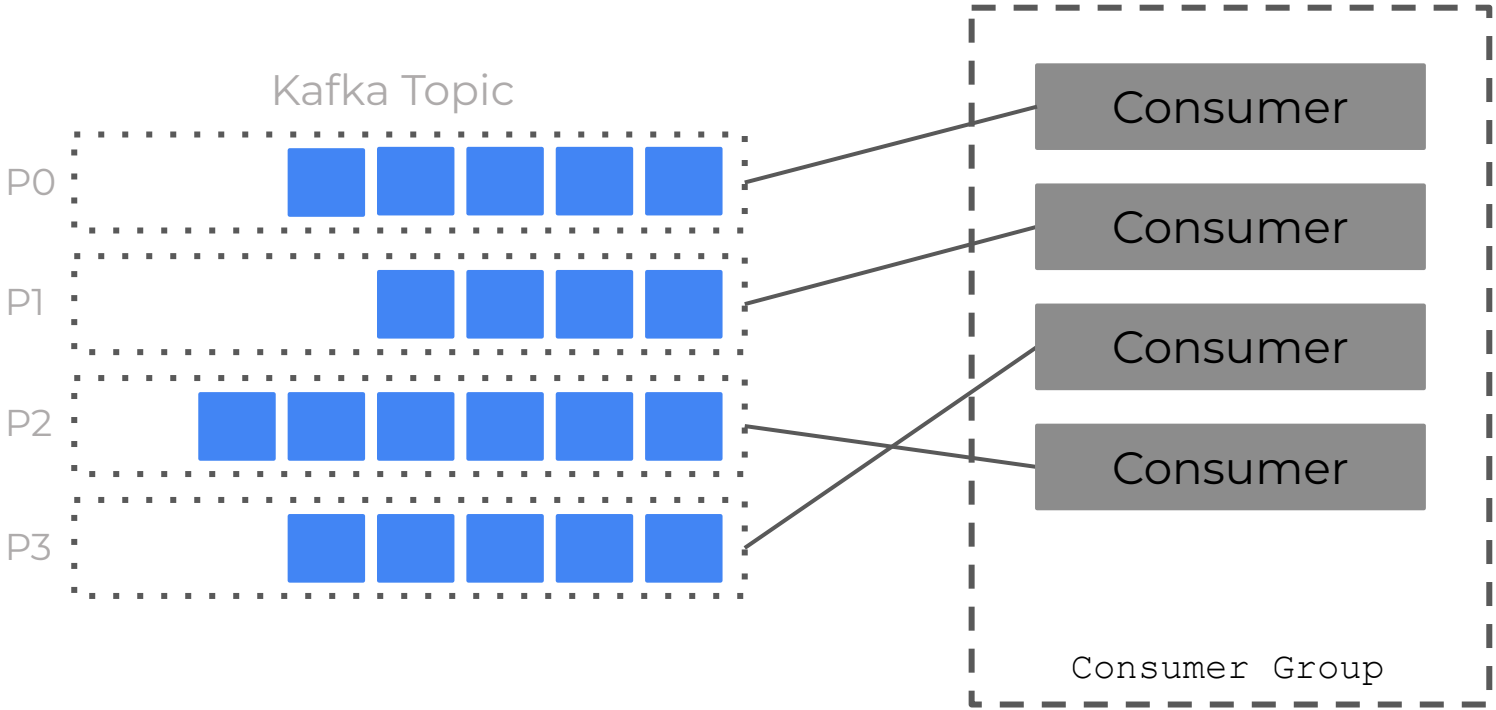
Consumer Groups



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

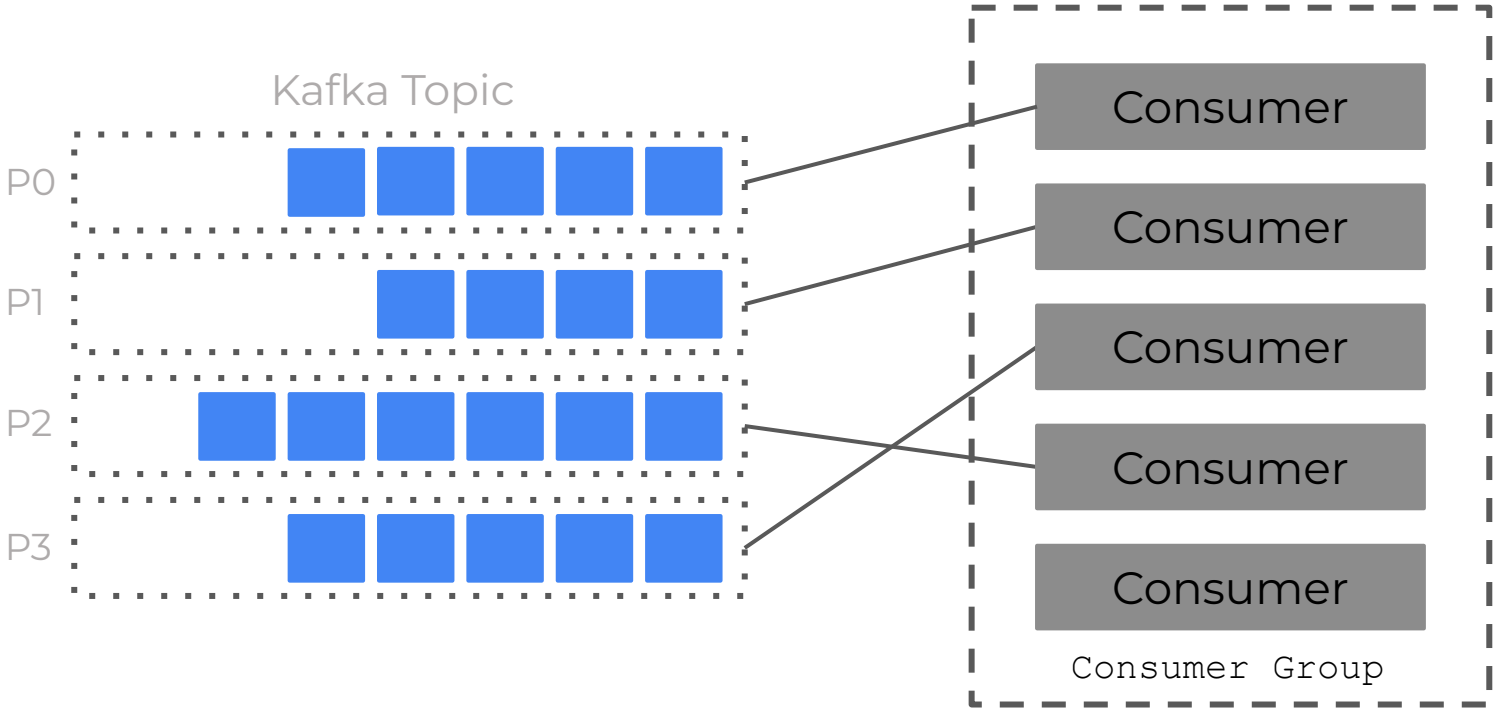
Consumer Groups



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

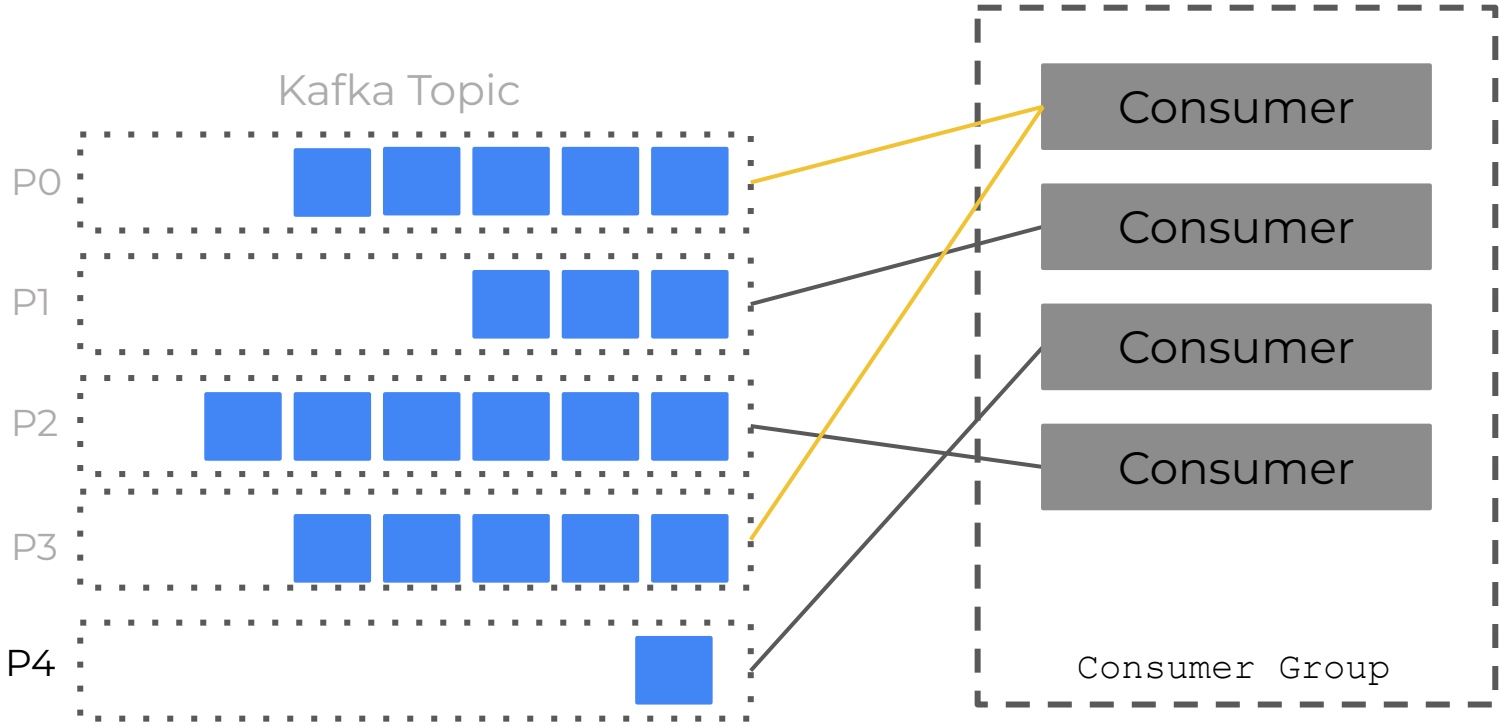
Consumer Groups



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

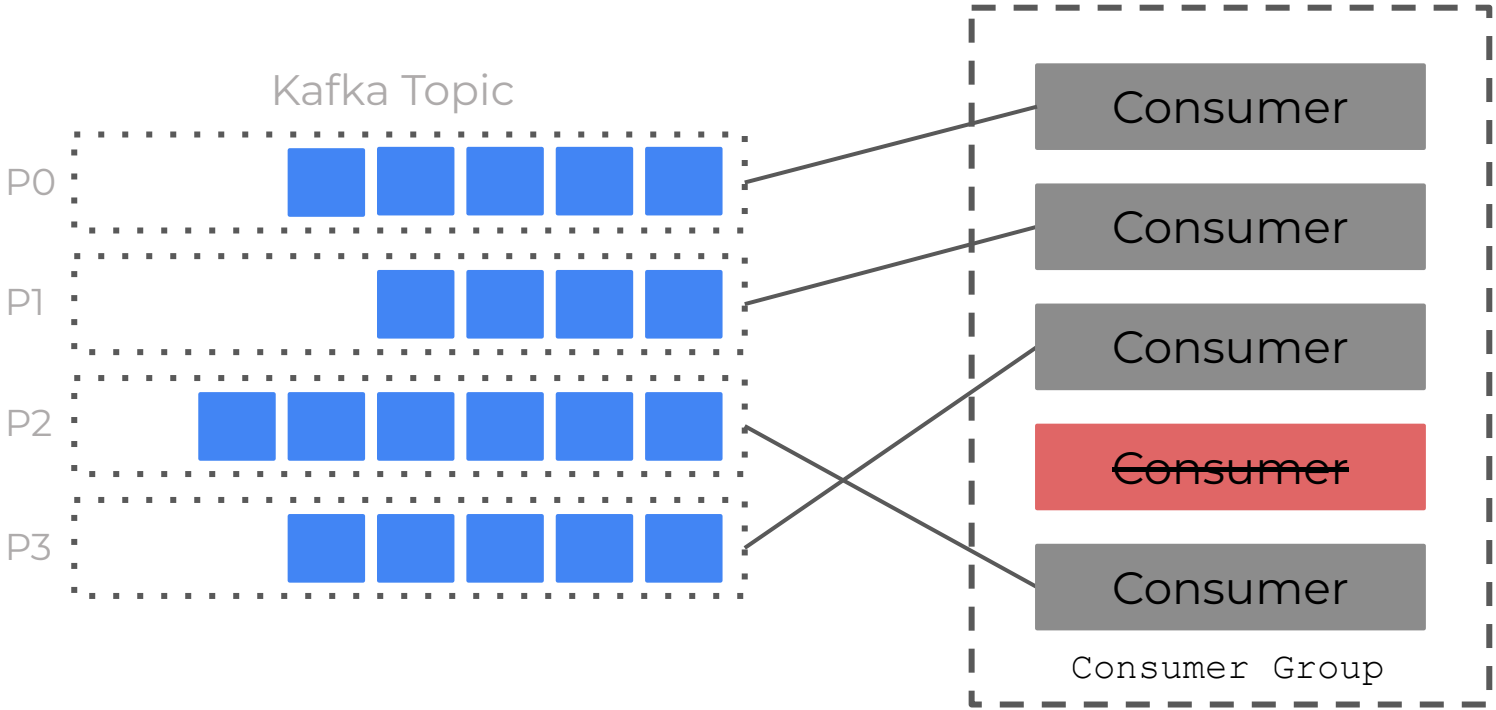
Consumer Groups



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

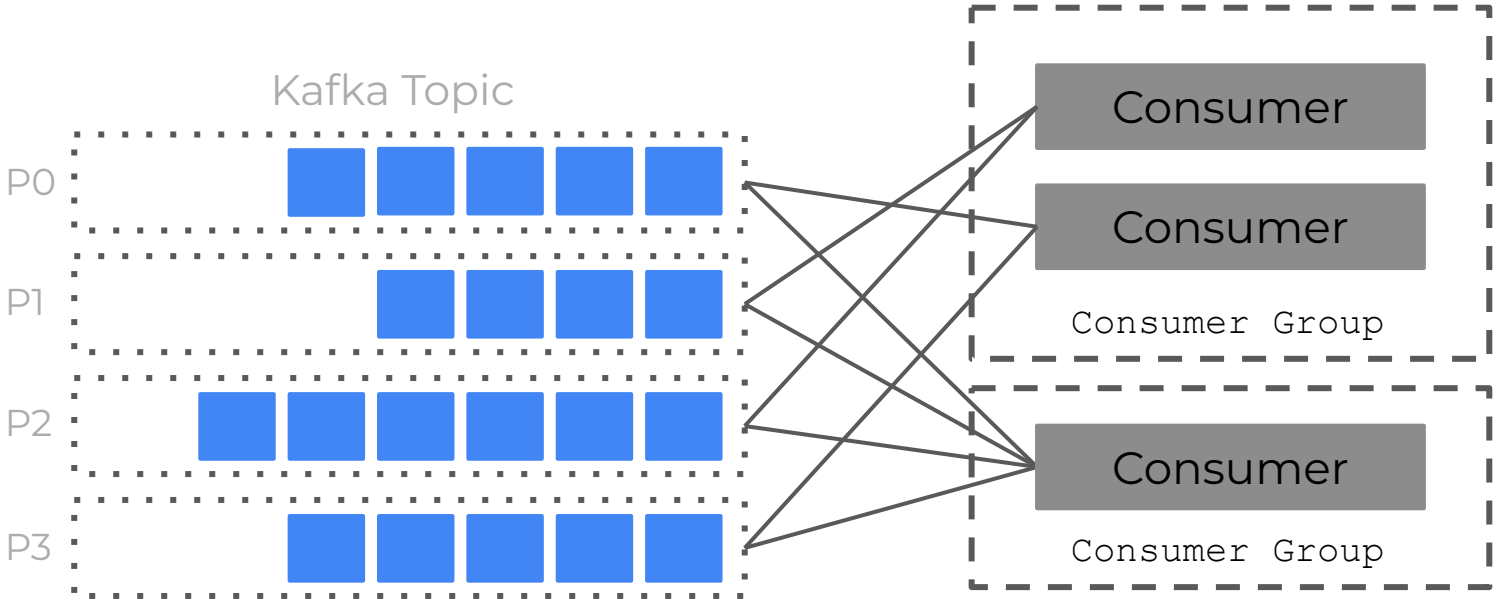
Consumer Groups



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

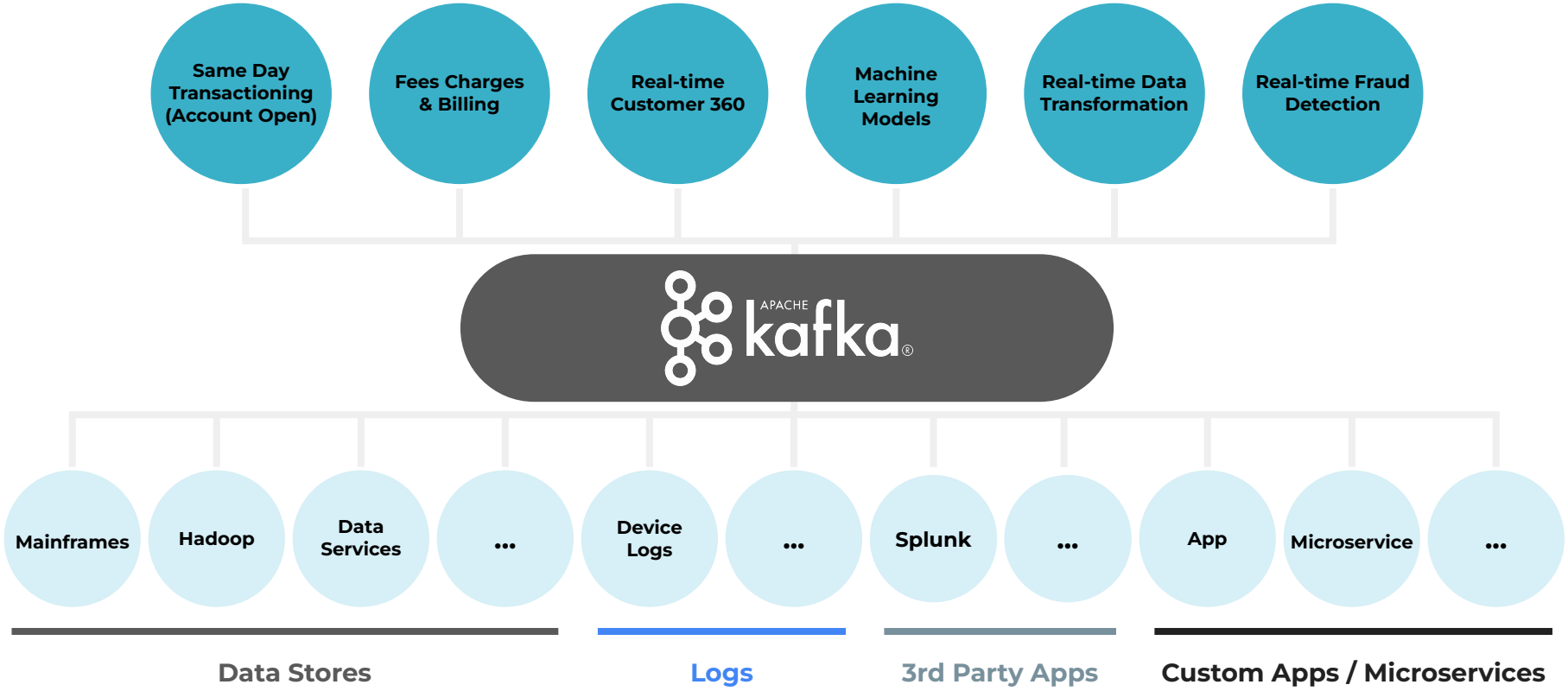
Consumer Groups



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Unified Fabric for App & Data Modernization



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Confluent? What's *that*?

Kafka, but make it easy.

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Confluent

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

CONFLUENT

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Hands On

Confluent Cloud and Apache Kafka Installation

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Kafka Installation Process

- Python Client Installation

  `pip install confluent-kafka`

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Confluent Cloud Account

Use Promo Code

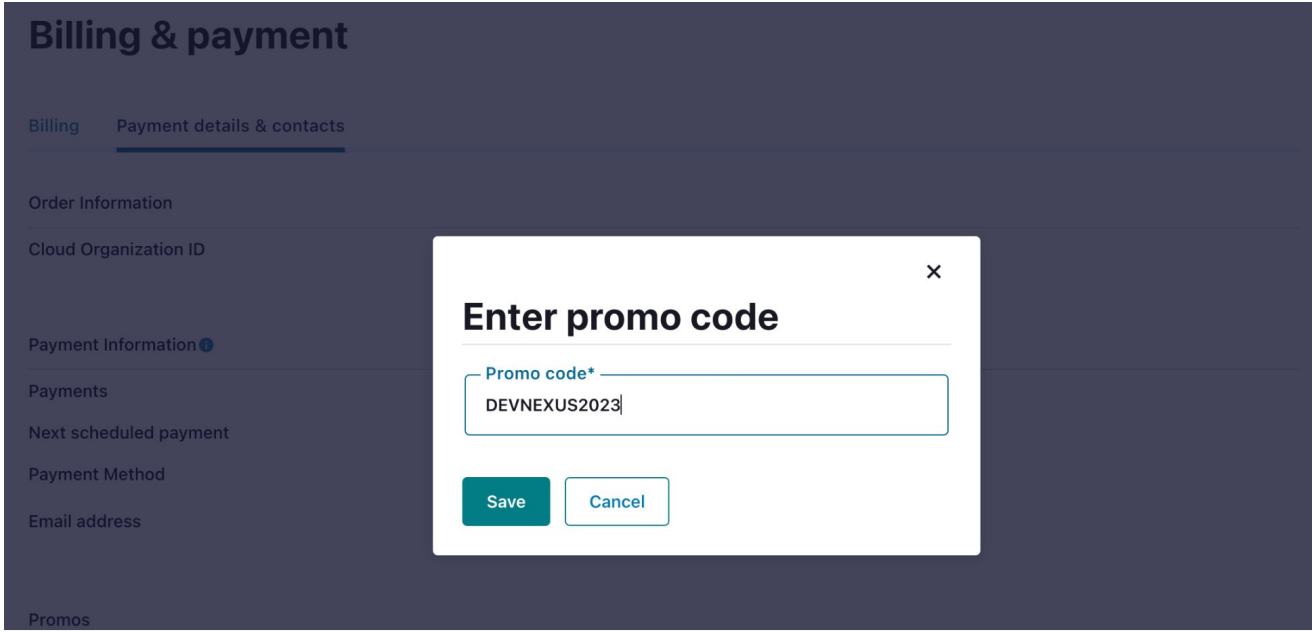
DEVNEXUS2023

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

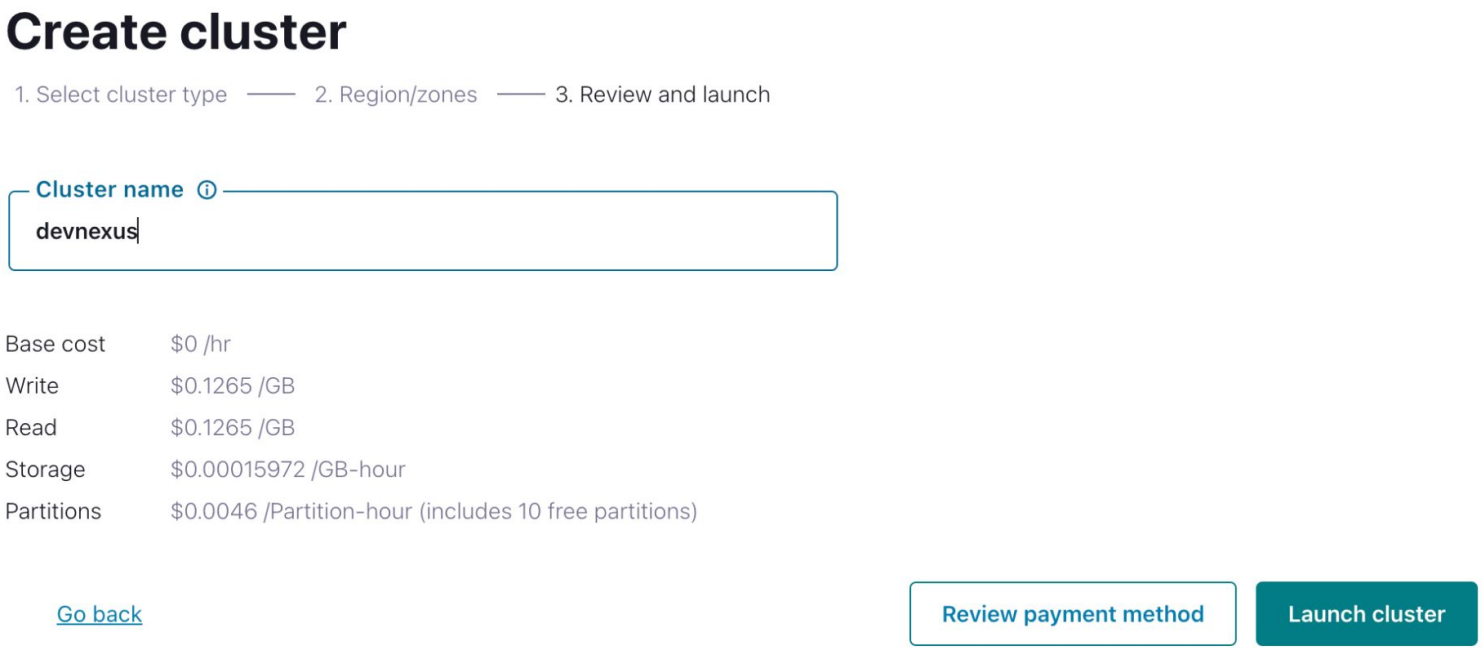
Cloud Account Set Up - Apply Promo Code

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



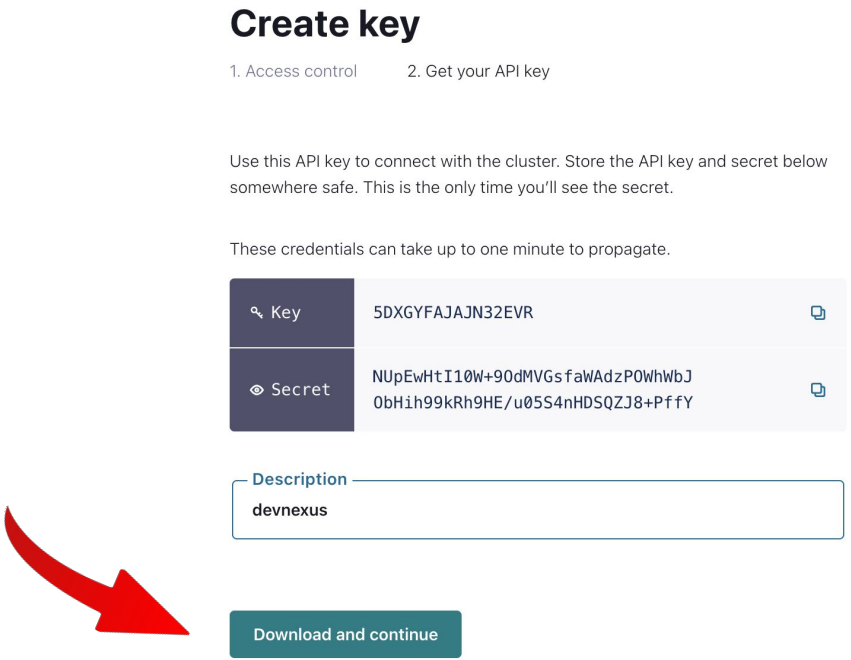
Cloud Account Set Up - Create Cluster

Default Env → + Add cluster → Basic cluster

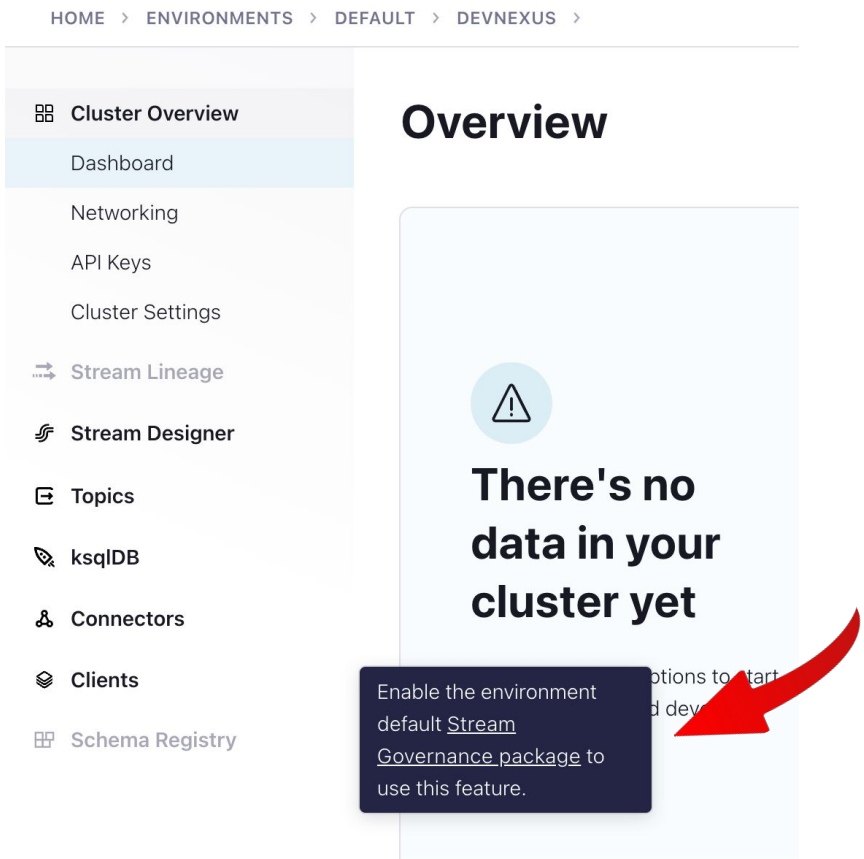


Cloud Account Set Up - Create API Key

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

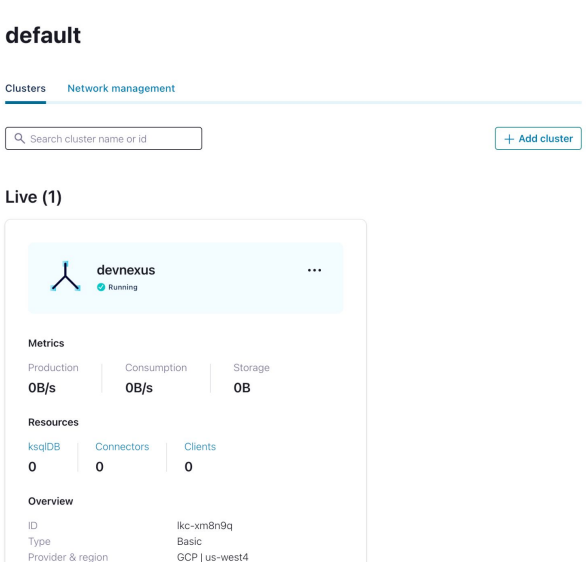


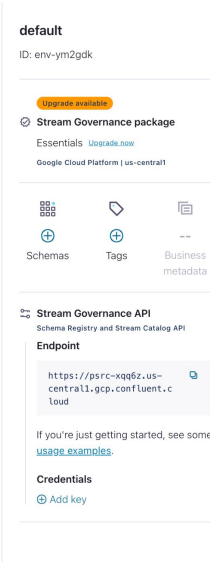
Confluent Cloud - Enable Governance

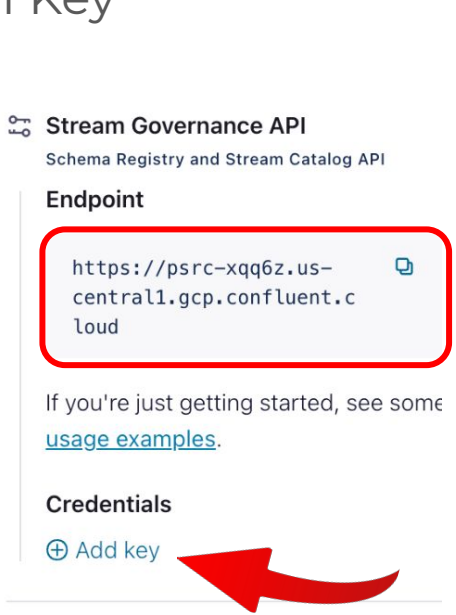


Confluent Cloud - Create SR API Key

Default Environment → <<Right sidebar>> → + 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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Coffee Break

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

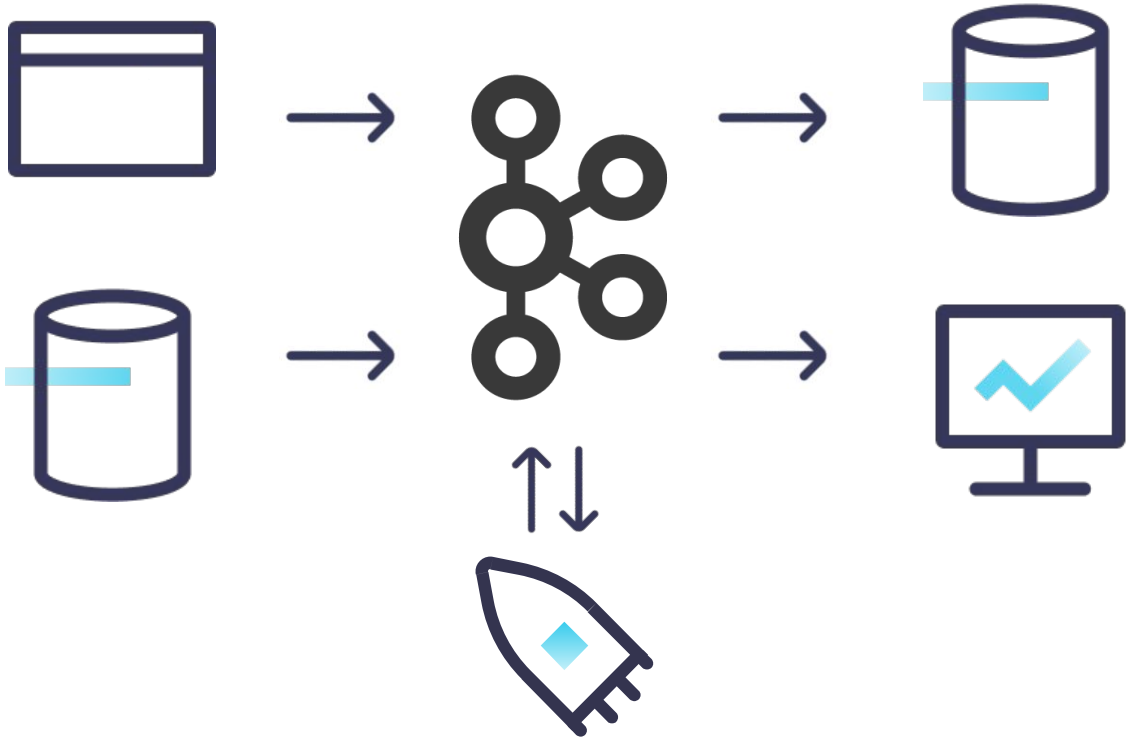
Morning Agenda

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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Streaming Data Pipeline



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Producing Data Into Kafka

Producer API vs Kafka Connect

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

But first, a use case.

Credit Card Fraud

What would a stream processing application for fraud look like?

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Typical Fraud Workflow

Purchase Event

Purchase Event

Pending Verification Event

- International Purchase Logic
- Did the customer purchase plane tickets?

Approved Event

Pending Verification Event

Fraud Analytic Platform

Approved Event

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Use Case: Credit Card Fraud

- Workshop Goal:

  Build a stream processing application to identify fraudulent credit card purchases.

- Datasets
  - Customers
  - Transactions

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Transactions

Key: 41


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


dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Customers (based on the `users` dataset)

```
Key: User_5

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

`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"
          }
    ]
}
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

`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"]
            ]
        }
    }
}
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

`schema.string` Example – Int Range

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

To Confluent Cloud!

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Morning Agenda

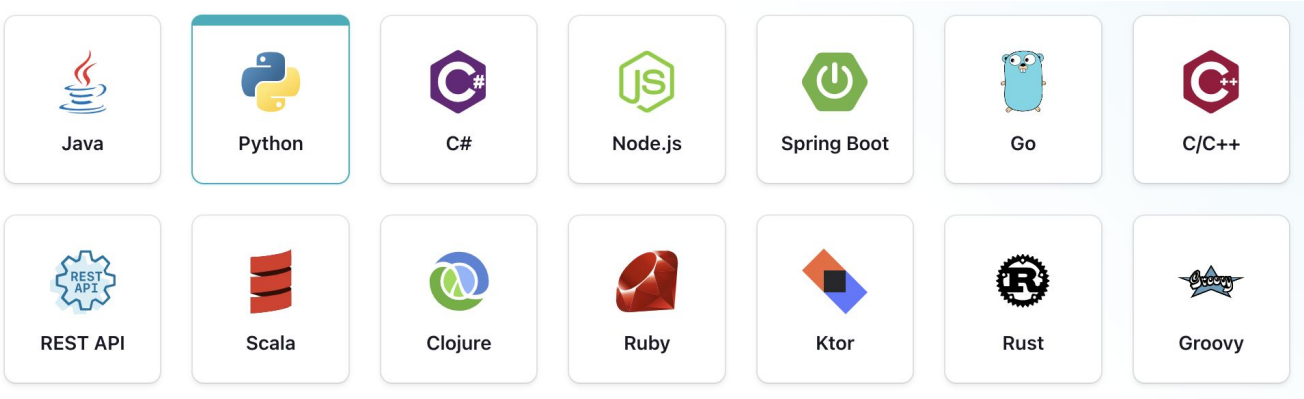
Time	Module Type	Description
09:00	Instruction	Introductions Streaming Data Pipelines – Benefits and Use Cases
09:15	Instruction	Apache Kafka – Architecture, Ecosystem, and Confluent
10:00	Hands On	Apache Kafka Installation Confluent Cloud Account Set Up
10:30	COFFEE BREAK	
10:45	Instruction	Introduction 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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

Checkpoint Quiz

Which configuration options apply to the following scenarios?

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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`

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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`

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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`

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Hands On

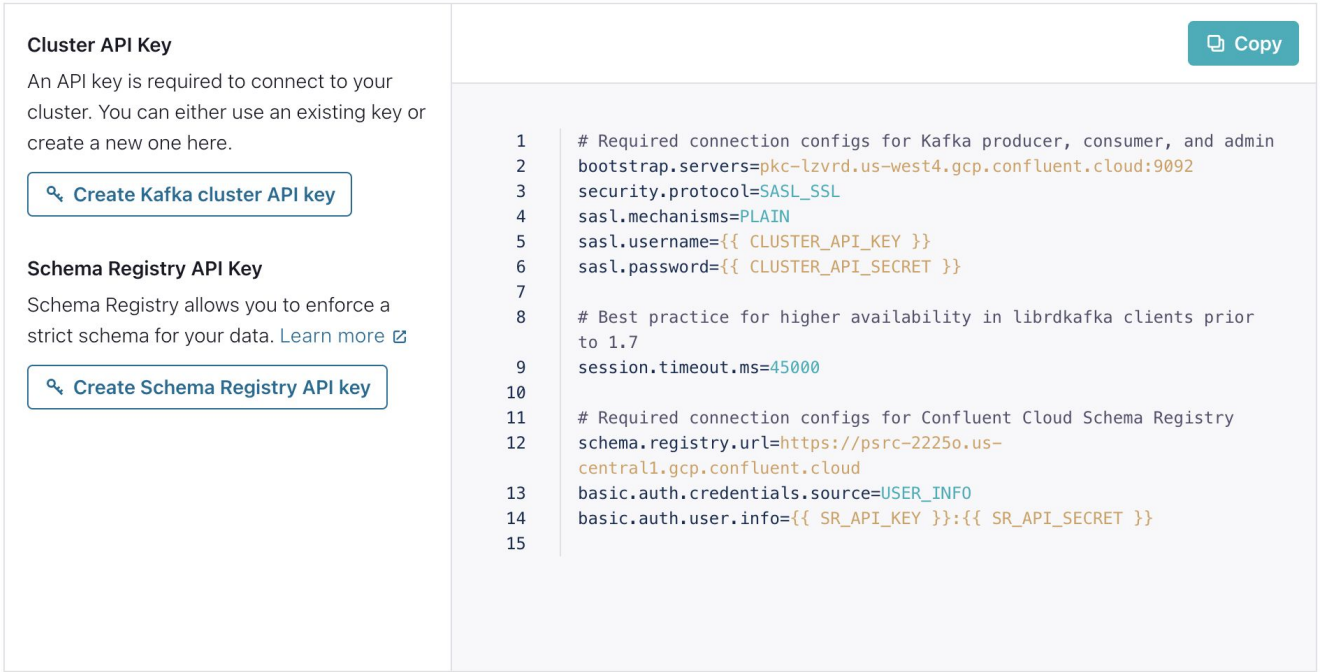
Create a Python Client

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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 `customers` and `transactions` datasets more interesting

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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	

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Lunch

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

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

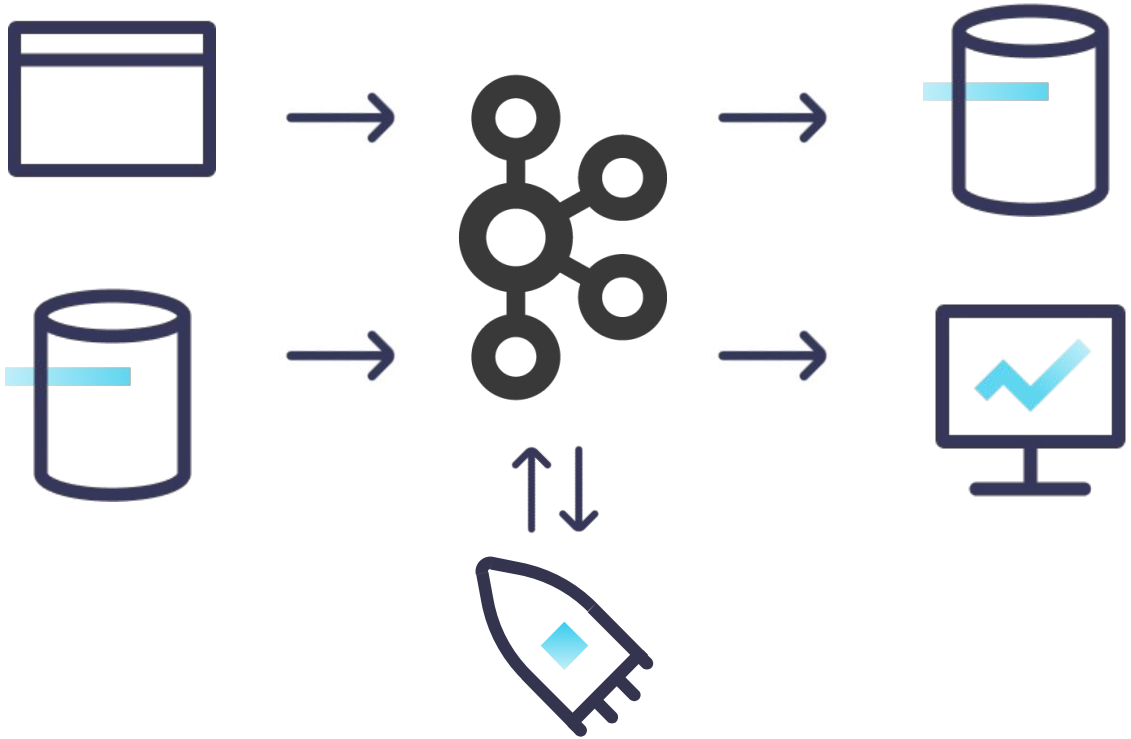
agilbert@confluent.io
@aelizabeth92

Stream Processing Techniques

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Streaming Data Pipeline



dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Use Case: Credit Card Fraud

- Workshop Goal:

  Build a stream processing application to identify fraudulent credit card purchases.

- Datasets
  - Customers
  - Transactions

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Consumer/Producer API

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

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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(), …
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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();
}
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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, …
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Example: ksqlDB

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

Hands On

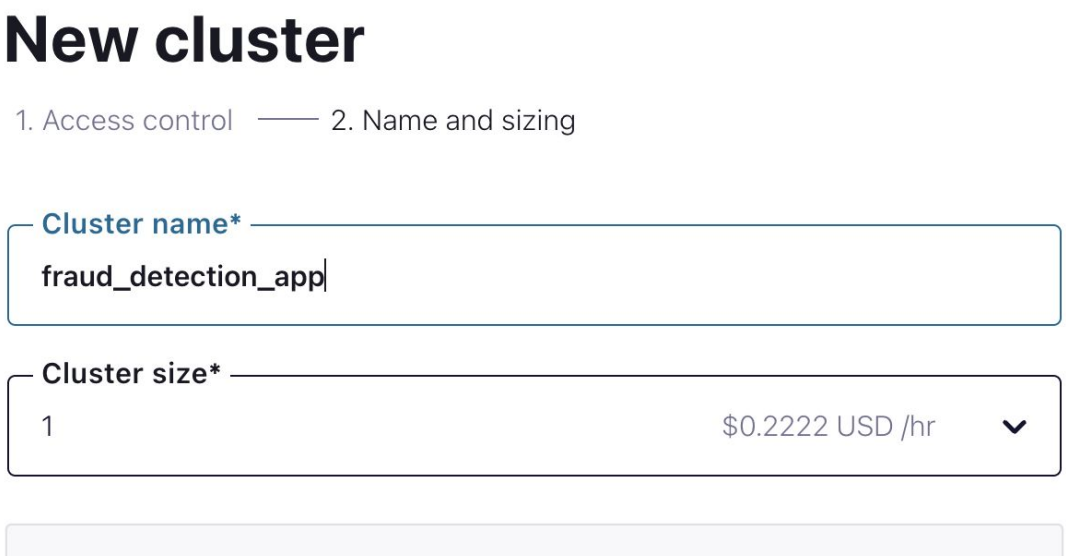
Stream Processing With ksqlDB

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

ksqlDB - Create a Cluster

Cluster Overview → ksqlDB → Create with Tutorial → Follow Prompts



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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Creating a Table

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Editor | Flow | Streams | Tables | Persistent queries | Performance | Settings | CLI instructions

```
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'
);
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Editor | Flow | Streams | Tables | Persistent queries | Performance | Settings | CLI instructions

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

Add query properties

auto.offset.reset = Earliest

+Add another field

Stop | Run query

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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;
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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;
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Pull Query


```
SELECT
   *
FROM transactions
WHERE user_id != 'User_99';
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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;
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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;
```

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

Coffee Break

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

Use Cases and Inspiration

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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)

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

Houseplant Alerting System

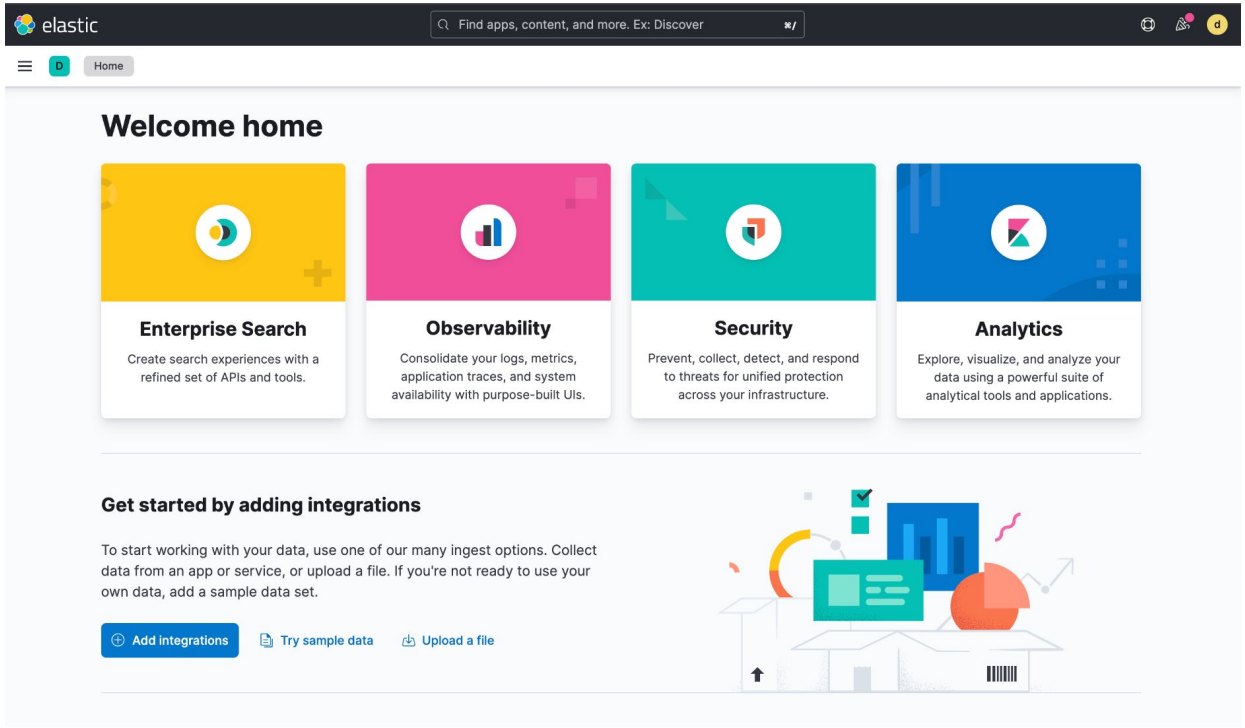
- Raspberry Pi-based, IoT Project
  - Collect moisture readings from plants using sensors
  - Use Python script to send readings to Kafka as events
- Stream Processing
  - Enrich readings with houseplant metadata
  - Determine if plants need to be watered and send a text alert

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

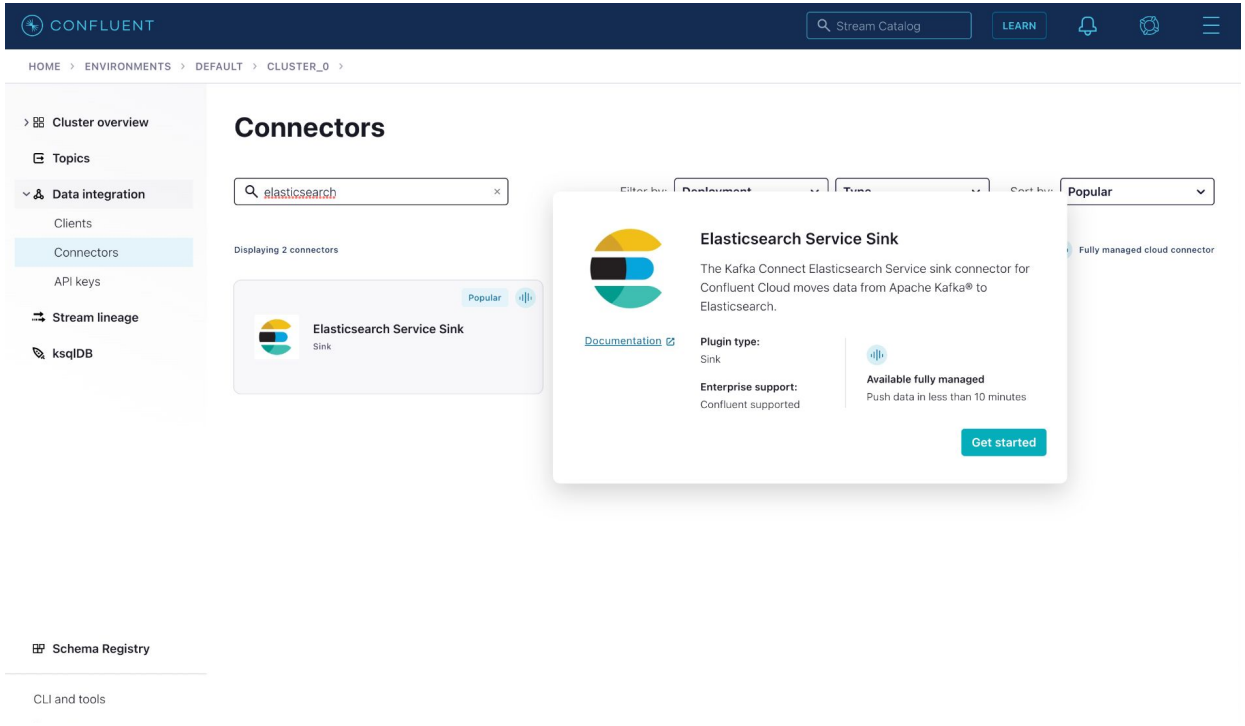
Kafka Connect - Elastic Sink Connector

Create a free account



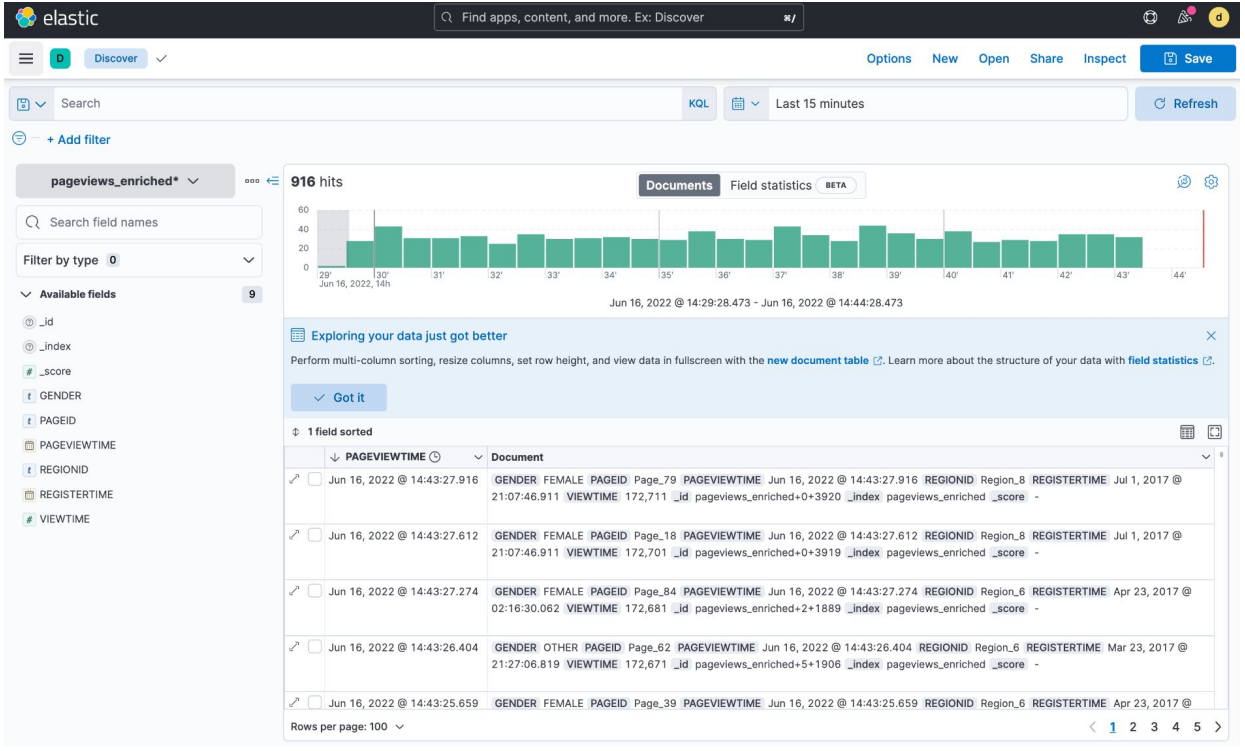
Kafka Connect - Elastic Sink Connector

Configure Elastic Sink Connector



Kafka Connect - Elastic Sink Connector

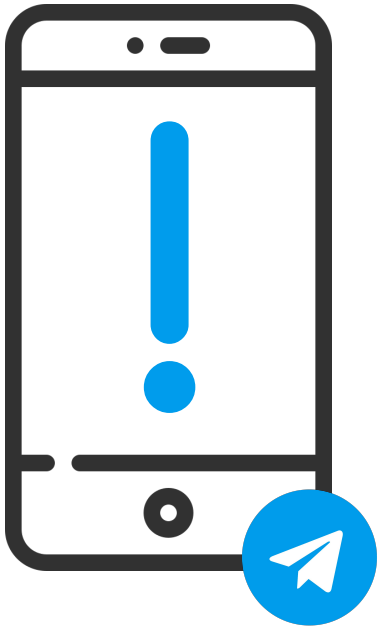
Conduct analysis



Kafka Connect - HTTP Sink Connector

Send alerts to your phone

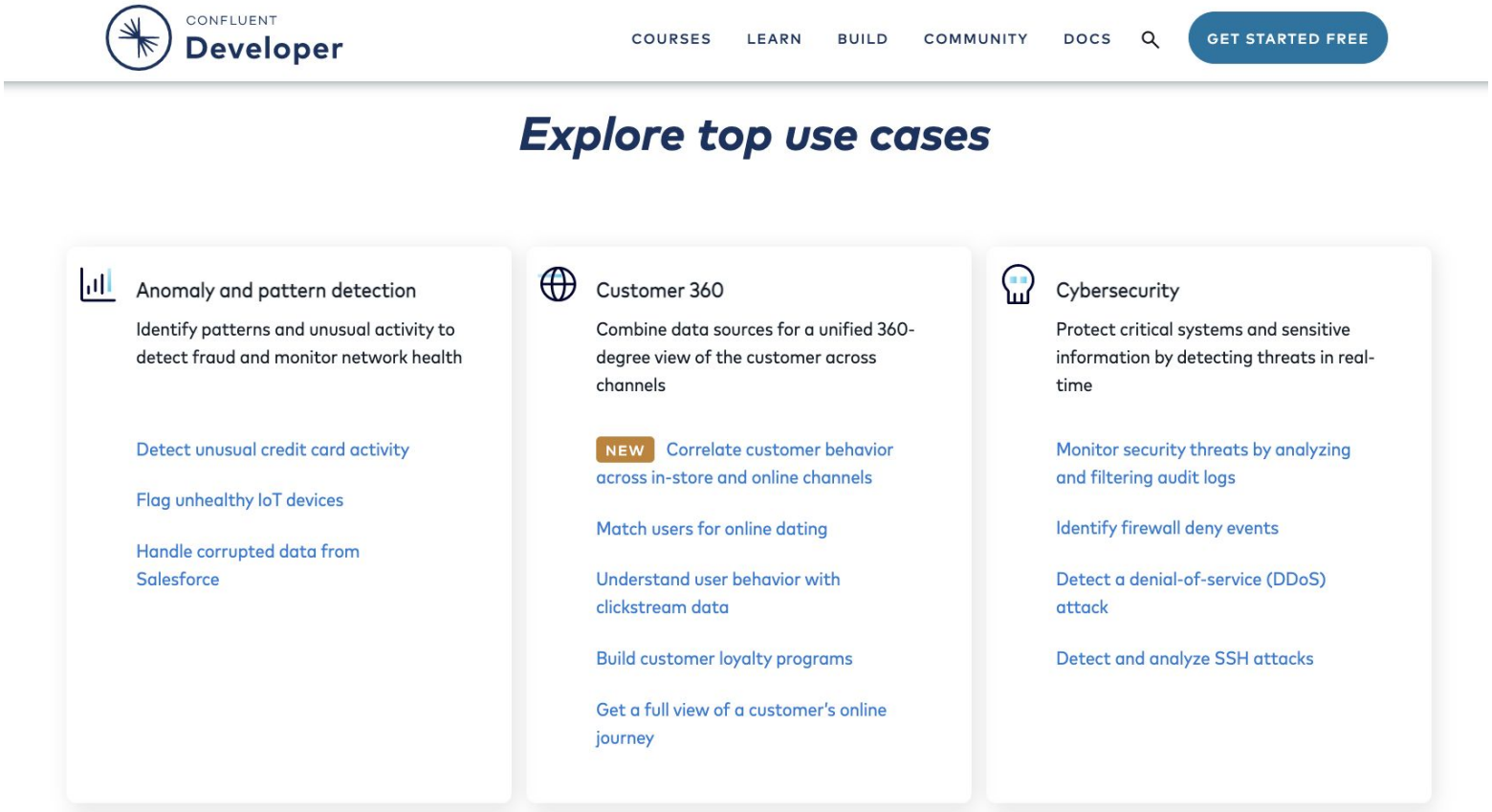
- Telegram App
- Create a bot
- Send data to bot endpoint



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

More Starting Points: ksqlDB Recipes



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?

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92

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

agilbert@confluent.io
@aelizabeth92

Hands On

End-to-End Pipelines

dfine@confluent.io
@TheDanicaFine

agilbert@confluent.io
@aelizabeth92