

Setting Data in Motion with Confluent Cloud

Version 7.3.3-v1.0.0



CONFLUENT

Table of Contents

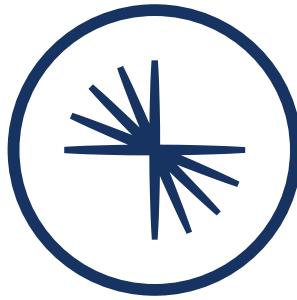
Introduction	1
Fundamentals Review	9
01: Introduction to Confluent Cloud	14
01a: Confluent Cloud - Fully Managed Service for Apache® Kafka	16
01b: Confluent Cloud Organization	20
01c: Confluent Cloud Accounts	32
Lab Module 01: Introduction to Confluent Cloud	35
02: Access Confluent Cloud	36
02a: Confluent Cloud Console - Demo	38
02b: Confluent CLI	40
02c: Confluent Cloud APIs	46
02d: Kafka Client	50
02e: Confluent Cloud Security Basics	52
Lab Module 02: Accessing Confluent Cloud	68
03: Schema Registry in Confluent Cloud	69
03a: Stream Governance	71
03b: Stream Quality: Schema Registry	75
03c: Stream Quality: Schema Validation	

03d: Stream Quality: Schema Linking	86
Lab Module 03: Working with Schema Linking	112
04: Stream Lineage & Stream Catalog	113
04a: Stream Lineage	115
04b: Stream Catalog	118
05: Move Data with Cluster Linking	125
05a: What is Cluster Linking?	127
05b: Cluster Linking Lifecycle	134
05c: Cluster Linking Security Overview	163
05d: Most Common Use Cases	169
Lab Module 05: Cluster Linking	176
06: Kafka Connect in Confluent Cloud	177
06a: What is Connect?	179
06b: Using Connectors in Confluent Cloud	188
06c: Security Considerations	204
06d: Networking with External Systems	207
06e: Self-Managed Connectors	210
Lab Module 06: Connect in Confluent Cloud	212
07: ksqlDB in Confluent Cloud	213
07a: Stream Concepts	215

07b: What is ksqlDB?	221
07c: Creating ksql Applications	224
07d: ksqlDB in Confluent Cloud	237
07e: Security Considerations	244
Lab Module 07: Using ksqlDB in Confluent Cloud for Stream Processing	247
08: Confluent Cloud Networking	248
08a: Networking Fundamentals (OPTIONAL)	250
08b: Confluent Cloud Overview	257
08c: Secure Public Endpoints	261
08d: VPC / VNet Peering	269
08e: AWS Transit Gateway	276
08f: Private Link	278
08g: Networking Summary	284
09: Automate, Deploy, and Manage Confluent Cloud	287
09a: Terraform and the Confluent Provider	289
09b: Pulumi and Confluent Cloud Provider	301
Lab Module 09: Automate, Deploy, and Manage in Confluent Cloud	310
10: Audit Logs, Metrics, and Notifications in Confluent Cloud	311
10a: Audit Logs	313
10b: Metrics	331

10c: Notifications	338
Lab Module 10: Audit Logs, Metrics, and Notifications in Confluent Cloud	342
11: Real-Life Use Case	343
Lab Module 11: Real-Life Use Case	345
Course Conclusion	346
Appendix: Additional Content	353
Appendix 1: Stream Designer	355

Introduction



CONFLUENT Global Education

Copyright & Trademarks

Copyright © Confluent, Inc. 2014-2023. [Privacy Policy](#) | [Terms & Conditions](#).

Apache, Apache Kafka, Kafka, and the Kafka logo are trademarks of the
[Apache Software Foundation](#)

All other trademarks, product names, and company names or logos cited herein are the property of their respective owners.

Prerequisites

This course requires a working knowledge of the Apache Kafka architecture.

New to Kafka? Need a refresher?

Sign up for free ***Confluent Fundamentals for Apache Kafka*** course at
<https://confluent.io/training>

Agenda



1. Introduction to Confluent Cloud
2. Access Confluent Cloud
3. Schema Registry in Confluent Cloud
4. Stream Lineage and Stream Catalog
5. Move Data with Cluster Linking
6. Connect in Confluent Cloud
7. ksqlDB in Confluent Cloud
8. Confluent Cloud Networking
9. Automate, Deploy & Manage Confluent Cloud
10. Monitor, Metrics, Audit Logs in Confluent Cloud
11. Real-Life Use Case

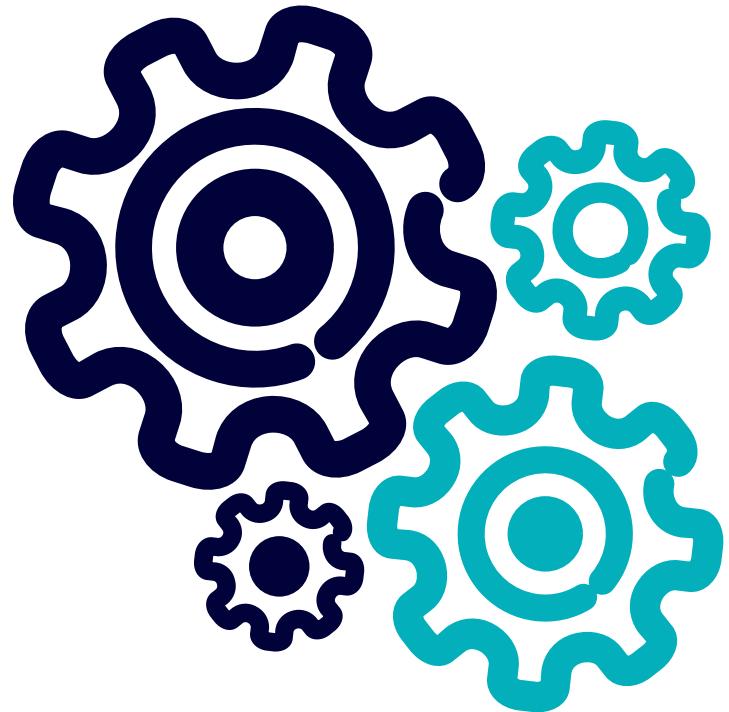
Throughout the course, Hands-On Exercises will reinforce the topics being discussed.

Course Objectives

After this course, participants will be able to:

- Apply best practices when designing your Confluent Cloud Organization
- Use Confluent CLI and APIs to perform common operations
- Integrate Schema Registry in your workloads and enable Schema Validation to prevent poisoning your data pipelines
- Mirror data between Kafka clusters using Schema Linking and Cluster Linking
- Discover your streams and their relationships using Stream Lineage and Stream Catalog
- Perform stream processing in Confluent Cloud with ksqlDB
- Integrate external data systems in Confluent Cloud using managed connectors
- Automate the infrastructure deployment in Confluent Cloud using Terraform
- Examine the Audit Logs and evaluate the Metrics API and Notifications

Class Logistics



- Timing
 - Start and end times
 - Can I come in early/stay late?
 - Breaks
 - Lunch
- Physical Class Concerns
 - Restrooms
 - Wi-Fi and other information
 - Emergency procedures
 - Don't leave belongings unattended



No recording, please!

How to get the courseware?



1. Register at **training.confluent.io**
2. Verify your email
3. Log in to **training.confluent.io** and enter your **license activation key**
4. Go to the **Classes** dashboard and select your class

Introductions



- About you:
 - What is your name, your company, and your role?
 - Where are you located (city, timezone)?
 - What is your experience with Kafka?
 - Which other Confluent courses have you attended, if any?
 - What is your language of choice?
 - Optional talking points:
 - What are some other distributed systems you like to work with?
 - What technology most excited you early in your life?
- About your instructor

Fundamentals Review

Discussion

Question Set 1 [6 mins]

Determine if each statement is true or false and why:

1. All messages in a topic are on the same broker.
2. All messages in a partition are on the same broker.
3. All messages that have the same key will be on the same broker.
4. The more partitions a topic has, the better.

Question Set 2 [3 mins]

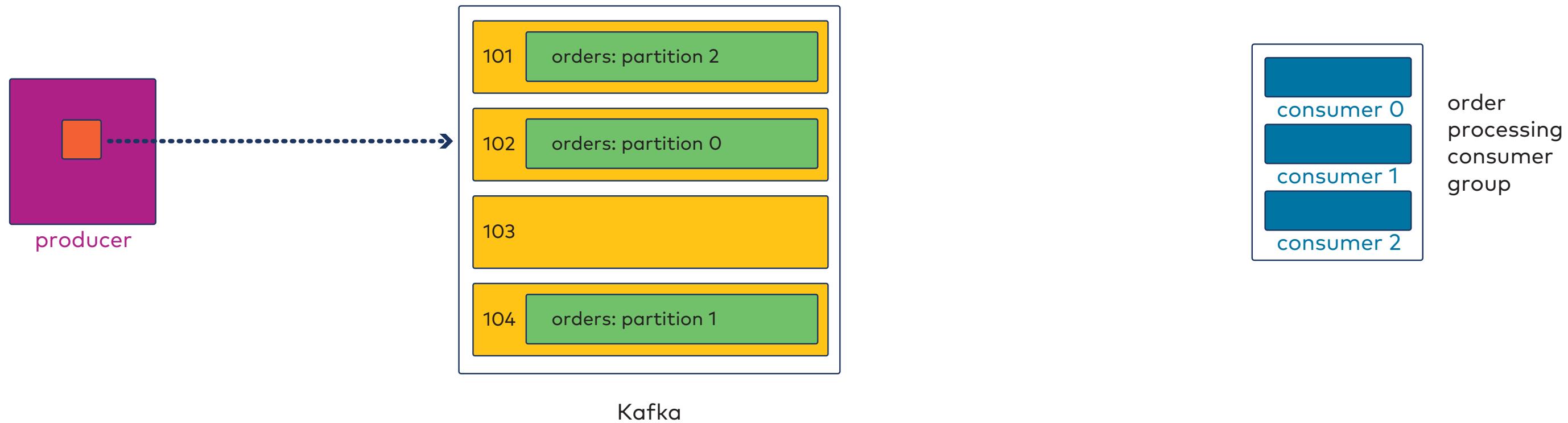
Determine the best answer to each question.

1. What are the roles of a producer and a consumer?
2. How is it decided which messages consumers read?
3. Who initiates the reading of messages: consumers or the Kafka cluster?

Instructor-Led Review

Some time is allocated here for an instructor-led review/Q&A on prerequisite concepts from Fundamentals.

Life Cycle of a Message: Producing (Optional)



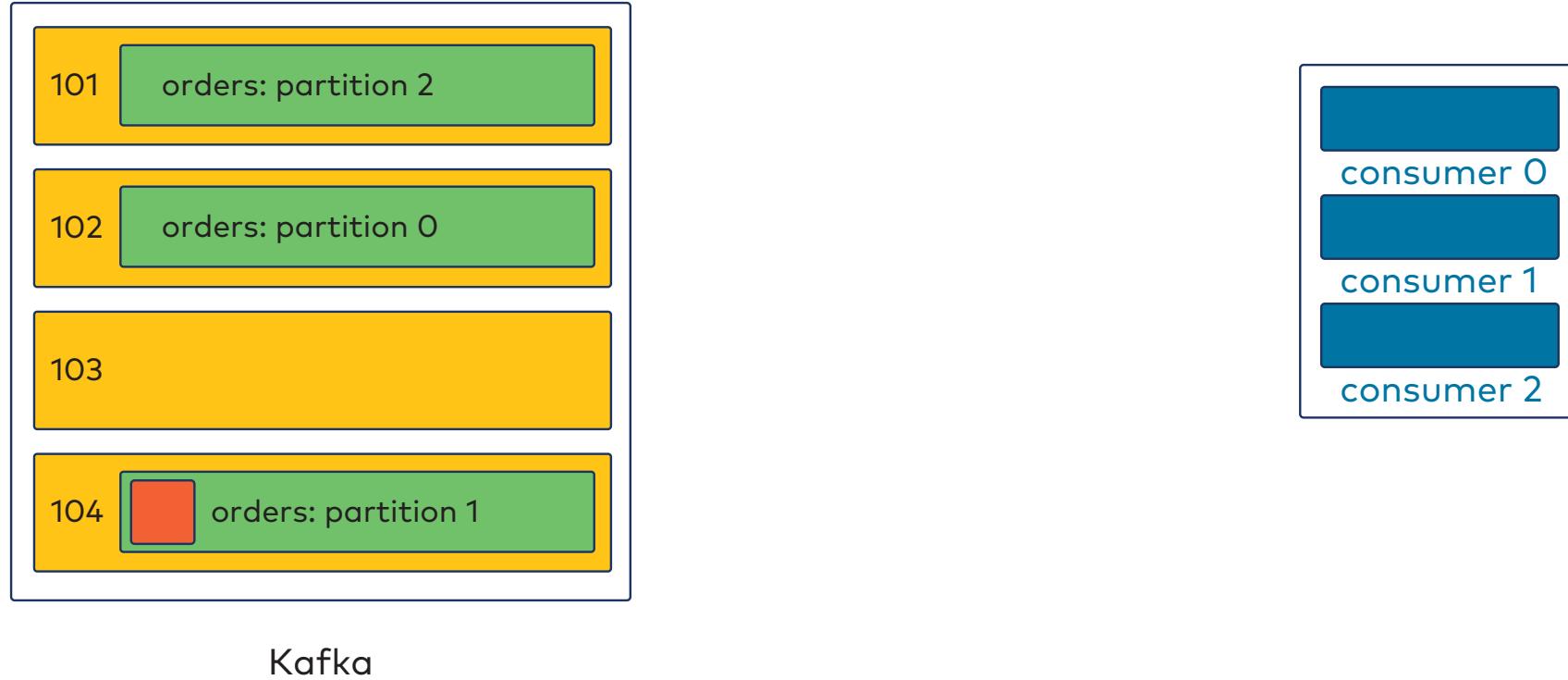
- Producers serialize and partition messages
- Producers send messages
 - ...in batches - can be configured for throughput and latency desires

Life Cycle of a Message: Kafka (Optional)

Produced messages live in Kafka, organized by topic.



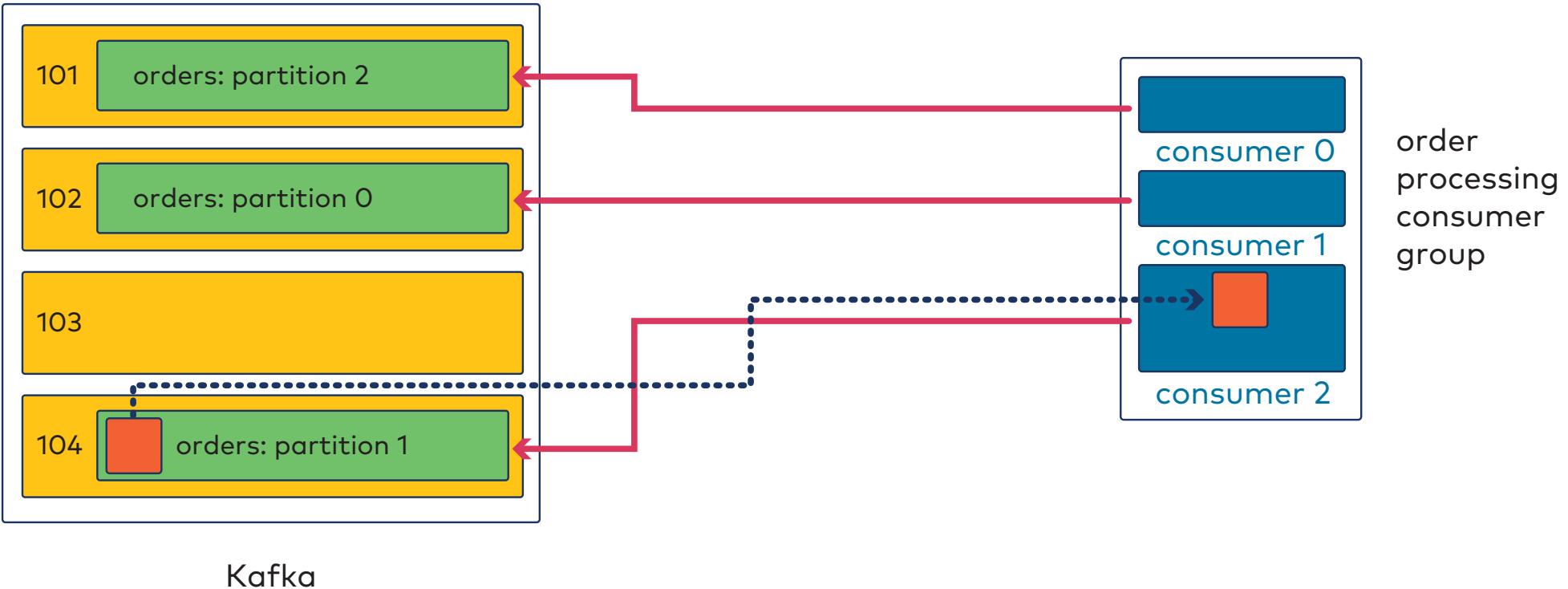
producer



- Kafka consists of brokers
- Brokers contain partitions, which contain messages
- Brokers handle retention and replication

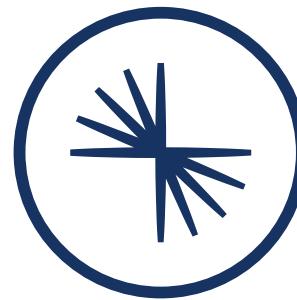
Life Cycle of a Message: Consumption (Optional)

Consumers subscribe to topics in Kafka and poll for new messages.



- Consumers operate in groups
- Consumers subscribe to topics, are assigned partitions of those topics
- Consumers poll for messages in partitions at consumer offsets
 - ...and fetch in batches - can be configured for throughput and latency desires

01: Introduction to Confluent Cloud



CONFLUENT
Global Education

Module Overview



This module contains three lessons:

- Confluent Cloud Fully Managed Service for Apache Kafka
- Confluent Cloud Organization
- Confluent Cloud Accounts

After this module you will be able to:

- Explain what “fully managed” means in the context of Confluent Cloud.
- Describe how Confluent Cloud is organized (environments, clusters, schema registry, etc.)
- Explain the differences between basic, standard and dedicated clusters.
- Describe the two types of accounts in Confluent Cloud.

01a: Confluent Cloud - Fully Managed Service for Apache® Kafka

Description

Confluent Cloud is a fully managed service from Confluent designed to set data in motion.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Define the two Confluent deployments.
- Explain what a fully managed Kafka cluster is.
- Compare between Cloud-Native and Cloud-Hosted.

Choose Your Deployment

Self-Managed Software

Confluent Platform

The Enterprise Distribution
of Apache Kafka

Deploy on any platform, on-prem or cloud



Fully-Managed Service

Confluent Cloud

Apache Kafka
Re-engineered for the Cloud

Available on the leading public clouds



Google Cloud

What is Confluent Cloud?

Fully Managed Streaming Platform Built on Apache Kafka

- **ELASTIC** - Massive scale without the ops overhead
- **GLOBAL** - Build for hybrid and multi-cloud
- **INFINITE** - Simplify planning with no-limits storage
- **COMPLETE** - Do more with data in motion (connectors, ksqlDB, Schema Registry...)
- **HIGHLY AVAILABLE** - Reliably scale mission-critical apps
- **SECURE** - Run with enterprise-grade security & compliance

Cloud-Native vs. Cloud-Hosted

	Confluent Cloud (Cloud-Native)	Cloud-Hosted (Self-Managed)
Sizing	Throughput-based	Broker-based
Infra Monitoring	Confluent proactive monitoring	Manual monitoring
Topic Monitoring	Pre-aggregated and free metrics	Per-topic per-broker metrics cost extra
Upgrades	Always on latest version	Limited version support
Vulnerability Patches	Proactive fixes	Not available
Cluster Expansions	Elastic scalability	Add brokers without data balancing
Connectors	Pre-built and fully managed	Self develop & manage
Support	Committer-driven expertise	Limited expertise
Environments	Freedom of choice	Limited
Ecosystem	Complete	Limited

01b: Confluent Cloud Organization

Description

An Organization is a structural object within Confluent Cloud.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

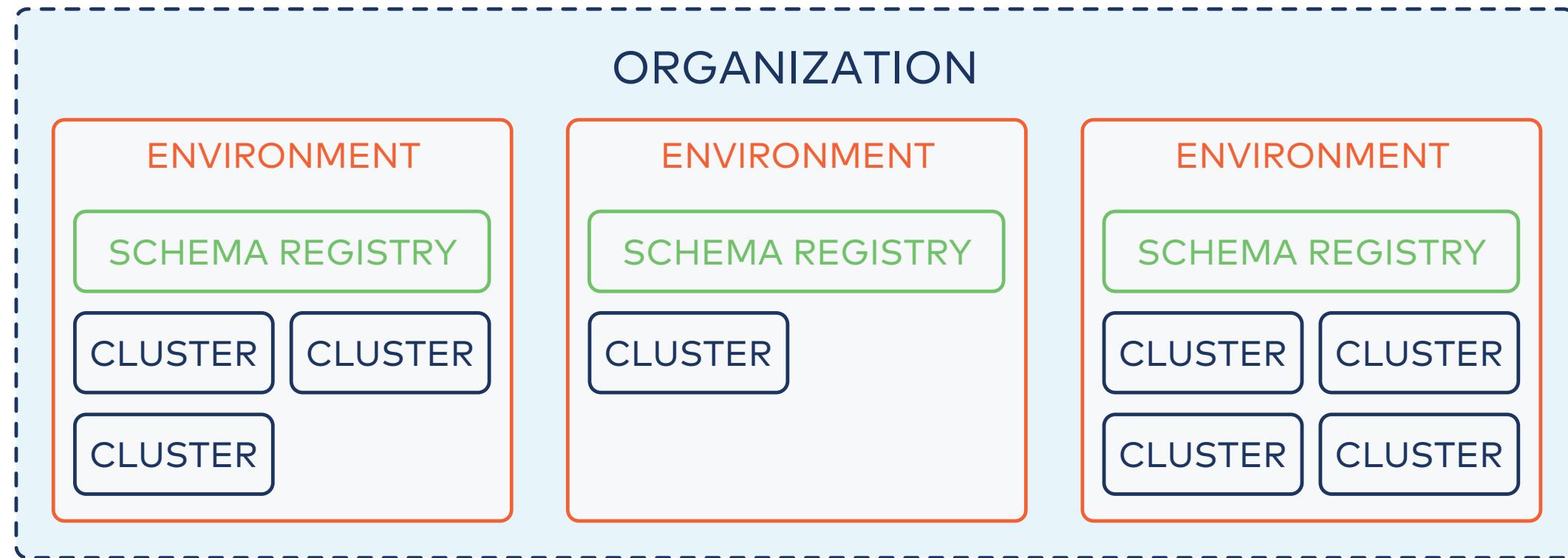
- Definition of how Confluent Cloud is structured.
- Comparison between the three types of clusters and discussion about pricing.
- Description of each component in Confluent Cloud.

How is Confluent Cloud Organized?

Confluent Cloud structure

- Organization:
 - Environment:
 - Schema Registry
 - Cluster (Kafka):
 - ksqlDB
 - Connect

Confluent Cloud Structure



Confluent Cloud Structure: Organization Level

These are the features that can be configured at the organization level:

- Manage billing (billing is at the Org level)
- Create user and service accounts
- Configure single sign-on (SSO)
- Manage support subscription level (support is at the Org level)

Confluent Cloud Structure: Environment Level

Consists of a logical number of clusters and a single Schema Registry.

You may want to create different environments based on:

- Application life cycle (development environment, staging environment, production environment)
- Regional-based deployments
- Cloud providers
- Lines of business
 - Lines of business means different teams in the company - sales, HR, Dev, QA, etc.

Confluent Cloud Structure: Cluster Level

In a Kafka cluster in Confluent Cloud you can:

- Create/update/delete topics
- View metrics of producers/consumers connected to the cluster
- Manage AuthN and AuthZ to the cluster via API keys and ACLs
- Create fully-managed connectors associated to this Kafka cluster
- Create ksqlDB clusters associated to this Kafka cluster

Types of Clusters

- BASIC (testing & development)
- STANDARD (small/medium production environments, Public Networking)
- ENTERPRISE (small/medium production environments, Private Networking)
- DEDICATED (small/medium/large production environments or special requirements, i.e., Cluster Linking, BYOK...)

Types of Clusters

Class	Basic	Standard / Enterprise	Dedicated
Use	Testing & Dev	Production	Production
Availability	Single-zone (SZ)	Single-zone (SZ) Multi-zone (MZ)	Single-zone (SZ) Multi-zone (MZ)
Tenancy	Multi-tenanted	Multi-tenanted	Single-tenanted
Uptime SLA	99.5%	99.95% - SZ 99.99% - MZ	99.95% - SZ 99.99% - MZ
Limits	<ul style="list-style-type: none">Throughput: 250 / 750 MBpsStorage: 5 TBPartitions: 4096	<ul style="list-style-type: none">Throughput: 250 / 750 MBpsInfinite StoragePartitions: ~5000	Depends on number of CKUs
Max Msg Size	8 MB	8 MB	20 MB

Pricing

Basic	Standard	Enterprise	Dedicated
Base price: \$0	Base price: \$/hour	Base price: \$/E-CKU/hour Minimum: 2 E-CKUs Fixed Ceiling: 5 E-CKUs	Base price: \$/CKU/hour Minimum: 1 CKU Maximum: 100 CKUs
Based on consumption:		Based on consumption:	
<ul style="list-style-type: none">• Ingress (\$/GB)• Egress (\$/GB)• Partitions (\$/partition/hour)• Storage (\$/GB-month)		<ul style="list-style-type: none">• Ingress (\$/GB)• Egress (\$/GB)• Storage (\$/GB-month)	
Fully-managed connectors, ksqlDB, cluster linking, audit logs, etc. are billed separately			

Fully Managed Schema Registry

Centralized repository for managing and validating schemas or topic message data

- Zero or one Schema Registry per Confluent Cloud environment
- Choose between different cloud providers and regions
- Specific API Key needed to access Schema Registry
- In multi-tenant deployments, one physical Schema Registry per cloud and geographic region hosts many logical schema registries

Fully Managed ksqlDB

ksqlDB is a database purpose-built to help developers create stream processing applications on top of Kafka Web UI for managing your ksqlDB cloud environment, including:

- SQL Editor with auto-completion
- Available in AWS, GCP, and Azure in all regions
- Integrates with Confluent Cloud Schema Registry
- Confluent charges you in Confluent Streaming Units (CSUs) per hour
- ksqlDB clusters can have 1, 2, 4, 8, 12, 16, 20, 24 and 28 CSUs

Fully Managed Connectors

Pre-built & fully managed connectors to instantly connect to popular data sources and sinks:

- Provides an HTTP API allowing you to interact with your connector
- Automatic Dead Letter Queue Topic creation when launching a sink connector
- Support for Single Message Transforms (SMTs)
- Connector Data Previews
- Connector log events can be viewed in the Confluent Cloud Console (Standard & Dedicated only)
- Connectors require credentials to be able to operate and access Kafka

01c: Confluent Cloud Accounts

Description

There are two types of accounts in Confluent Cloud: User Accounts and Service Accounts.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Define User Accounts and Service Accounts in Confluent Cloud.

User Account

- It represents one user.
- An Organization may contain one to many User Accounts.
- A given user (corresponding to a specific email address) can be a member of multiple organizations at the same time.

Service Account

- It represents an application programmatically accessing Confluent Cloud.
- An Organization may contain zero to many Service Accounts.

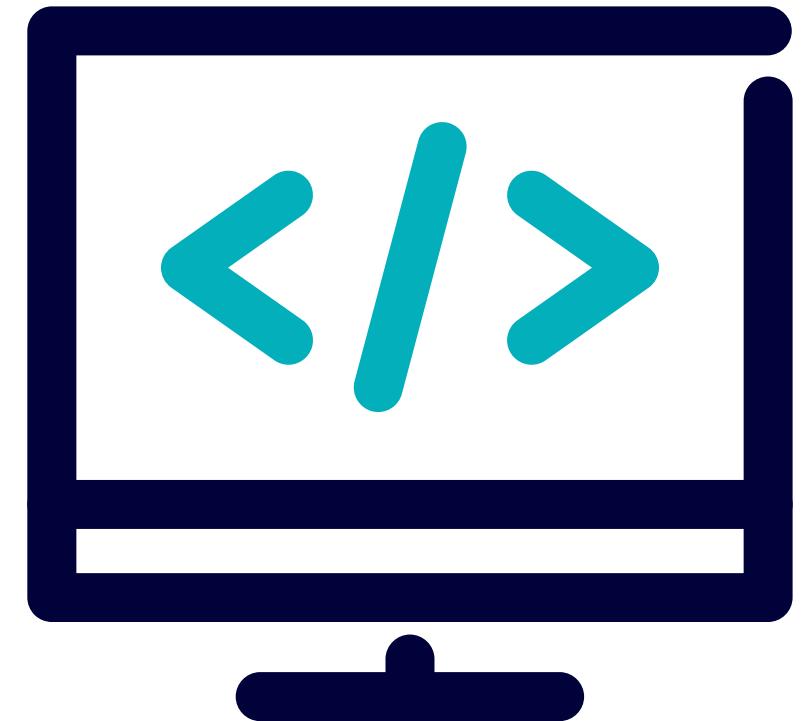


Confluent strongly recommends that you use service accounts for **all production-critical access**.

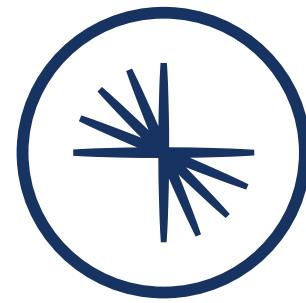
Lab Module 01: Introduction to Confluent Cloud

Please work on **Lab Module 01: Introduction to Confluent Cloud**.

Refer to the Exercise Guide.



02: Access Confluent Cloud



CONFLUENT
Global Education

Module Overview



This module contains five lessons:

- Confluent Cloud Console
- Confluent CLI
- Confluent Cloud APIs
- Kafka Client
- Confluent Cloud Security Basics

After this module you will be able to:

- Navigate in the Confluent Cloud UI to view topics, client metrics, data flows, schemas, connectors, etc.
- Use the Confluent Cloud CLI and REST API to perform operations
- Provide AuthN and AuthZ to Users and Services accounts

02a: Confluent Cloud Console - Demo

Description

Your instructor will provide a demonstration of the Confluent Cloud console.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Demo Confluent Cloud Console including, where to find the different components, how to configure them, etc.

Confluent Cloud Console - Demo



Welcome to Confluent Cloud

Log in with your email

 Sign in with Google

 Sign in with GitHub

Or

Email*

Next

[Forgot password](#)

Don't have an account? [Sign up and try it for free](#)

Copyright © Confluent, Inc. 2014-2022. [Privacy Policy](#) | [Terms & Conditions](#)
Apache, Apache Kafka, Kafka, and associated open source project names
are trademarks of the Apache Software Foundation



O2b: Confluent CLI

Description

Use the Confluent Cloud command line interface (CLI) to develop and manage a Confluent Cloud environment.

Learning Objectives



- Upon completion of this lesson and associated lab exercises, you will be able to:
- Define the Confluent CLI.
 - Explain the installation options for Confluent Cloud.
 - Identify important commands for Confluent Cloud users and administrators.

Confluent CLI Overview

- The Confluent CLI enables developers to create, manage, and deploy their Confluent components.
- Confluent CLI traffic uses the following ports and endpoints:
 - Cloud operational requests are over HTTPS / port **443** and go to <https://confluent.cloud>.
 - Kafka protocol requests, for example, for produce/consume, go over port **9092** to Kafka brokers.
- Supported operating systems: macOS, Windows, and Linux.

Confluent CLI Installation

- For **macOS** and **Linux**, run the following command:

```
brew install confluentinc/tap/cli
```

- For **Windows**:

1. Download the latest Windows ZIP file from:

```
https://github.com/confluentinc/cli/releases/latest
```

2. Unzip the following file:

```
confluent_X.X.X_windows_amd64.zip
```

3. Run `confluent.exe`

Tarball or Zip Installation

1. Download and install the most recently released CLI binaries by platform:
2. Set the `PATH` environment to include the directory that you downloaded the CLI binaries in the previous step

Important Commands for All Users

Command	Options	Description
<code>confluent help</code>	-	Get information about all available commands
<code>confluent login</code>	-	Login to your account
<code>confluent environment</code>	<code>list</code> <code>use</code>	List all environments you have access to Set your current environment
<code>confluent kafka cluster</code>	<code>list</code> <code>use</code>	List all clusters you have access to in the current environment Set your current cluster in the current environment

Important Commands for Cloud Administrators

Command	Options	Description
<code>confluent iam user</code>	<code>delete, describe, invite, list</code>	Manage all the users within your organization
<code>confluent iam service-account</code>	<code>create, delete, list, update</code>	Manage CCloud Service Accounts
<code>confluent iam rbac role-binding</code>	<code>create, delete, list</code>	Manage role bindings for principals
<code>confluent api-key</code>	<code>create, delete, list, store, update, use</code>	Manage the API keys of all resources (cluster, SR, ksqlDB, Cloud metrics)
<code>confluent environment</code>	<code>create, delete, update</code>	Manage environments in your org.
<code>confluent kafka cluster</code>	<code>create, delete, describe, update</code>	Manage clusters in your current environment
<code>confluent kafka acl</code>	<code>create, delete, list</code>	Manage ACLs for your Service Accounts

02c: Confluent Cloud APIs

Description

Use Confluent Cloud application programming interfaces (APIs) to programmatically extend and integrate.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Manage your own Confluent Cloud account or to integrate Confluent into your product:
 - Create a topic.
 - Update the topic configuration.
 - List cluster configuration.
 - List consumer groups and consumer lags.

API Requests

- API requests **must** be made over HTTPS
- API requests without authentication will fail
 - Authentication is included in Headers as an `Authorization: Basic {key}`
 - `{key}` requires you to base64 encode `<api-key>:<api-secret>:`

```
$ echo -n "<api-key>:<api-secret>" | base64
```

Create a Topic

Create a new topic `testTopic1`:

```
curl -H "Authorization: Basic ABC123ABC" -H 'Content-Type: application/json' \
--request POST \
--url 'https://pkc-lzvrd.us-west4.gcp.confluent.cloud:443/kafka/v3/clusters/lkc-vo9pz/topics' \
-d '{"topic_name": "testTopic1", "partitions_count": 5, "replication_factor": 3}'
```

List All Topics

```
curl -H "Authorization: Basic ABC123ABC" --request GET \  
--url 'https://pkc-lzvrd.us-west4.gcp.confluent.cloud:443/kafka/v3/clusters/lkc-vo9pz/topics'
```

O2d: Kafka Client

Description

Use the Confluent Cloud client to programmatically develop and manage Confluent Cloud environments and clusters.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Define the required properties to connect a Kafka client to Confluent Cloud.

Connect Clients to Confluent Cloud

- Set the configuration in the client code:

```
bootstrap.servers=pkc-ep9mm.us-east-2.aws.confluent.cloud:9092
security.protocol=SASL_SSL
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule      required username='{{ CLUSTER_API_KEY }}' password='{{ CLUSTER_API_SECRET }}';
sasl.mechanism=PLAIN
```

- If using Schema Registry in Confluent Cloud:

```
schema.registry.url=https://psrc-4nrnd.us-central1.gcp.confluent.cloud
basic.auth.credentials.source=USER_INFO
basic.auth.user.info={{ SR_API_KEY }}:{{ SR_API_SECRET }}
```

- Confluent Cloud Console provides the required configuration for your programming language:
 - In the Cluster → [Clients](#) → [New client](#)

02e: Confluent Cloud Security Basics

Description

Understand the basic fundamentals of securing a Confluent Cloud environment and users.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Explain how to control access to Confluent Cloud.
- Define API keys with concrete examples.
- Explain role-based access control (RBAC) and access control lists (ACLs).
- Describe best practices for setting access control.

API Keys

API keys control access to Confluent Cloud components and resources.

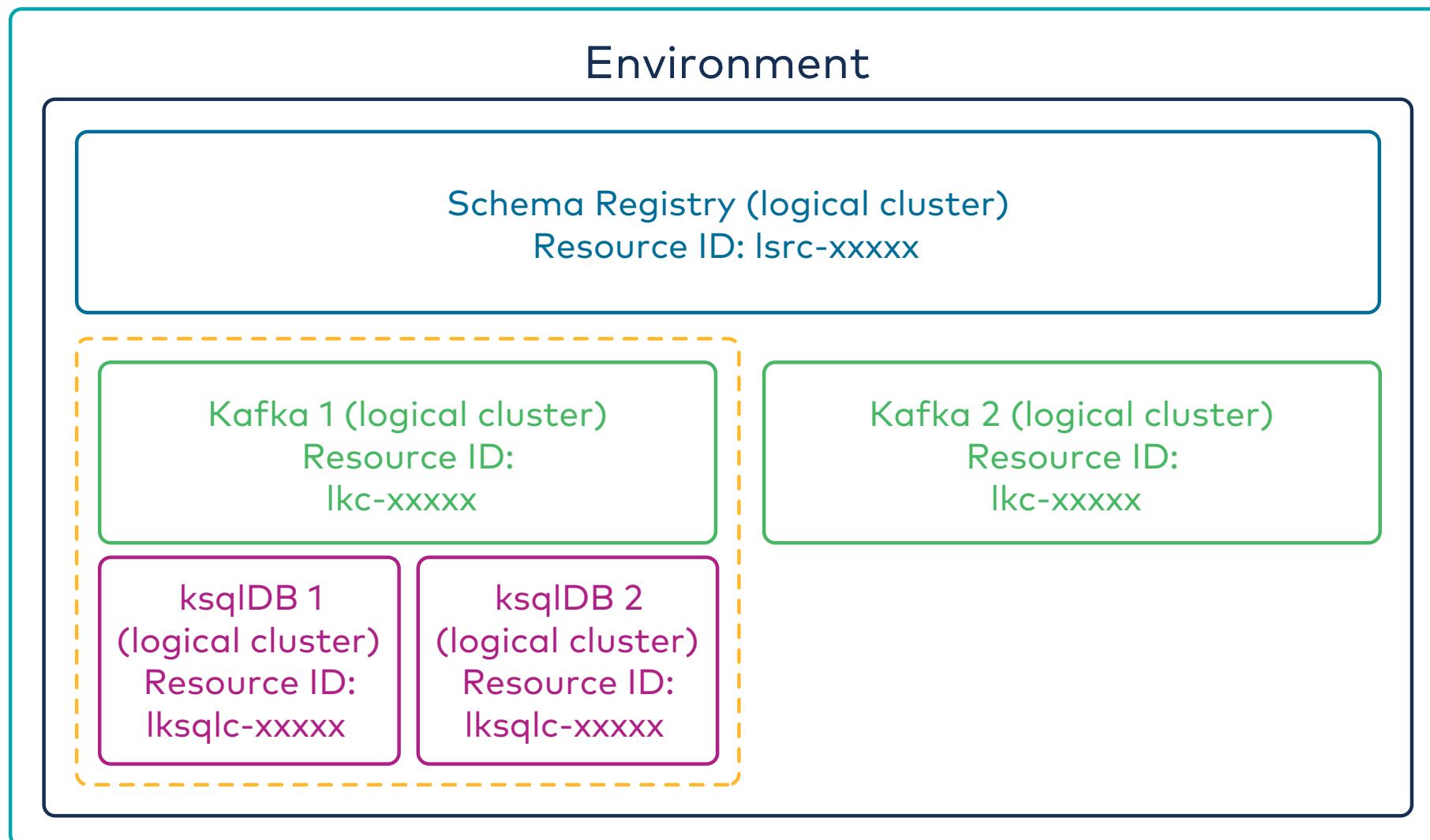
Each API key consists of a key and a secret (username and password).

There are four types of API keys:

- Kafka API keys
- Schema Registry API keys
- ksqlDB API keys
- Cloud API keys

API Keys - Cluster Overview

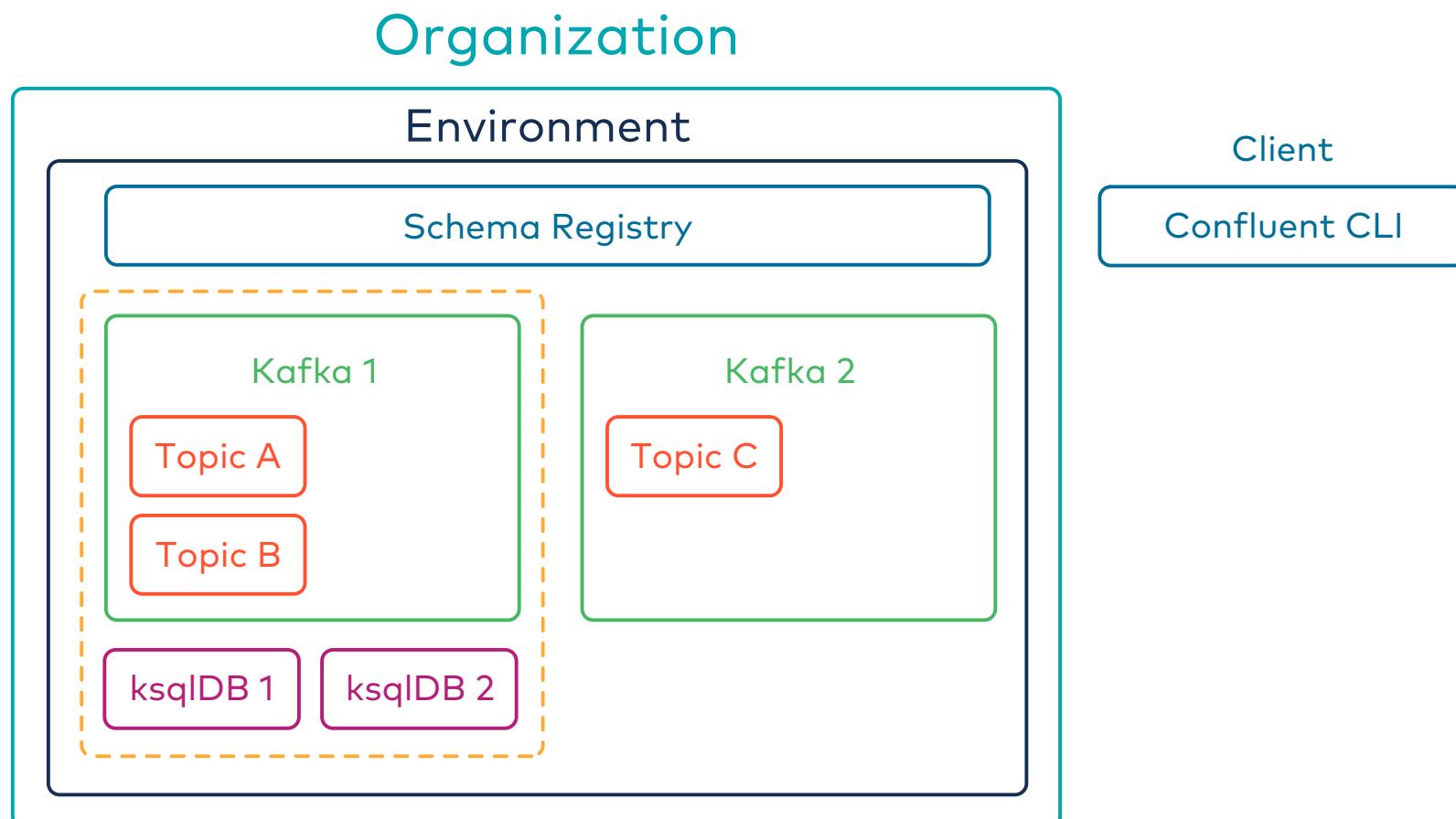
Organization



API Keys - Confluent CLI Client (1)

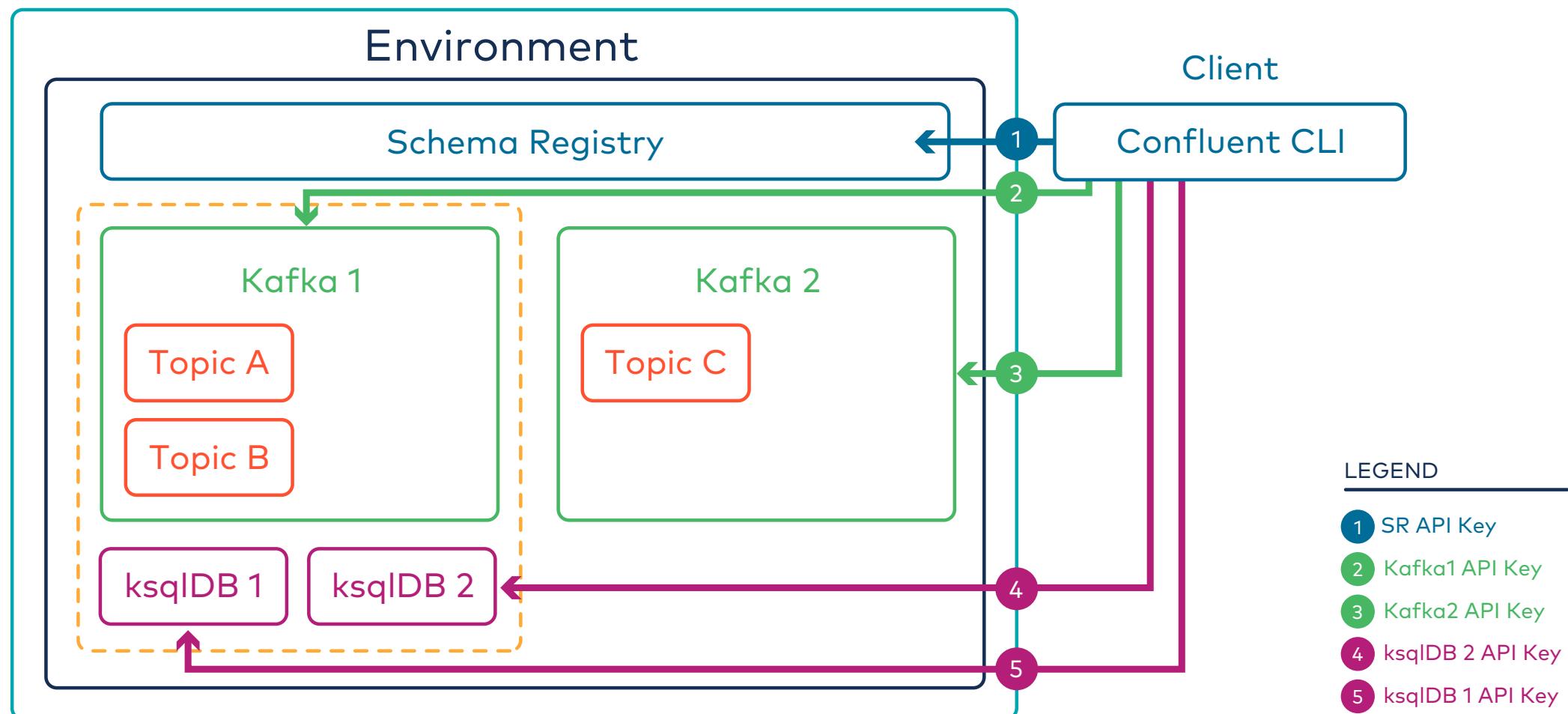
Question:

What API keys will be required if the Confluent CLI needs to register a schema to Schema Registry, produce to **Topic A** and **Topic C** and send queries to the two ksqlDB clusters?



API Keys - Confluent CLI Client (2)

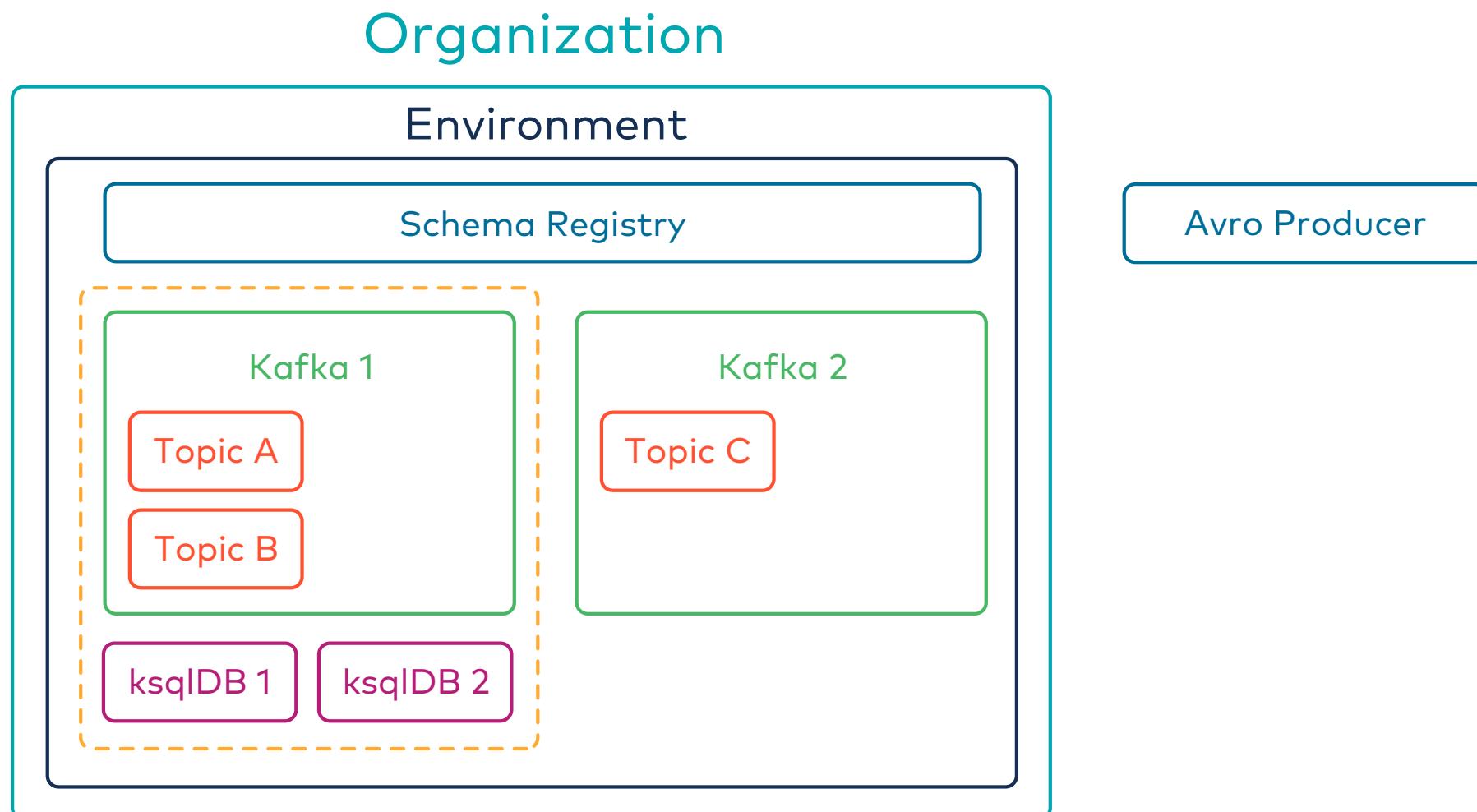
Organization



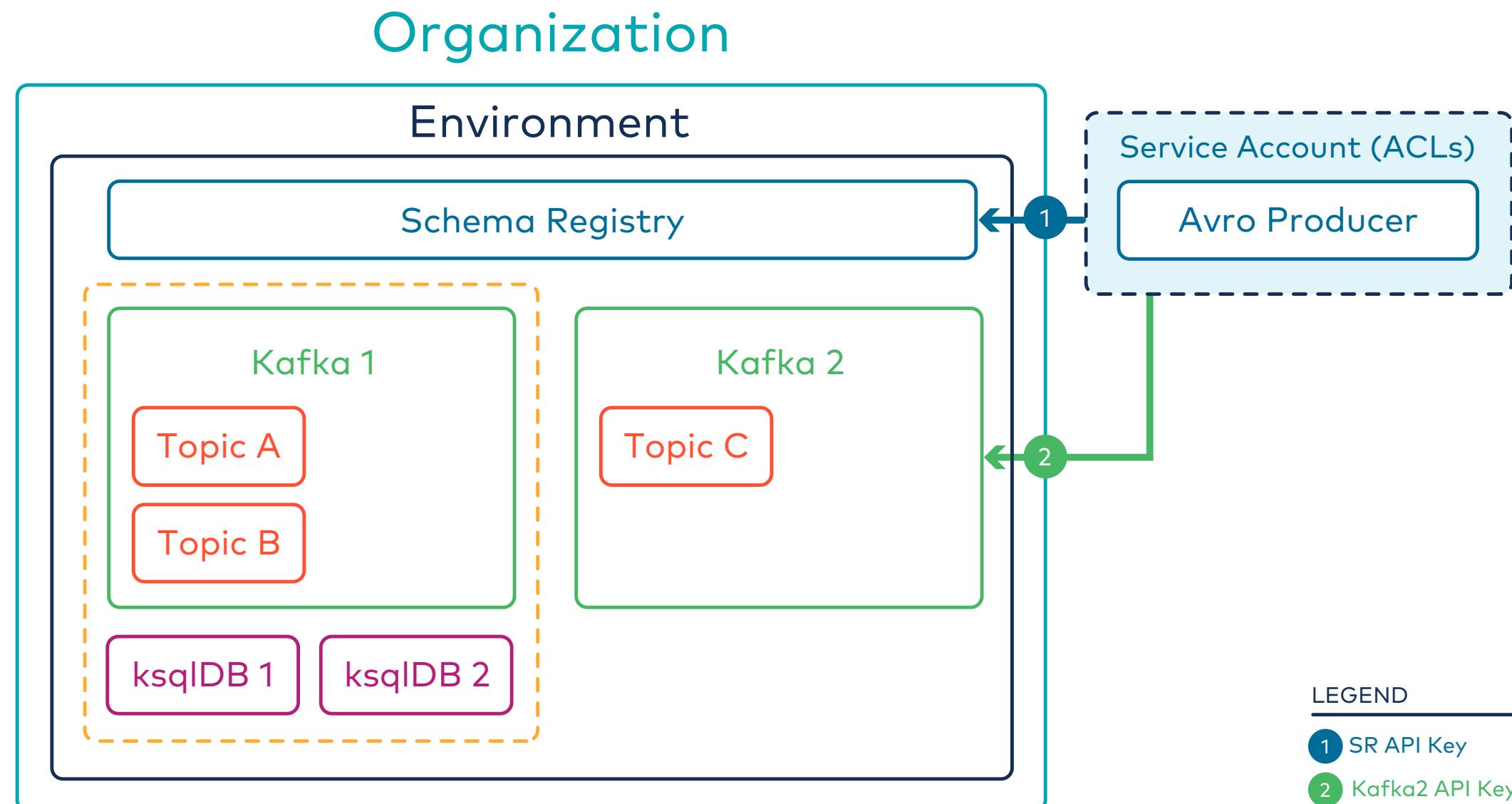
API Keys - Kafka App (1)

Question:

What API keys will be required if the Avro Producer needs to write messages to Topic C?



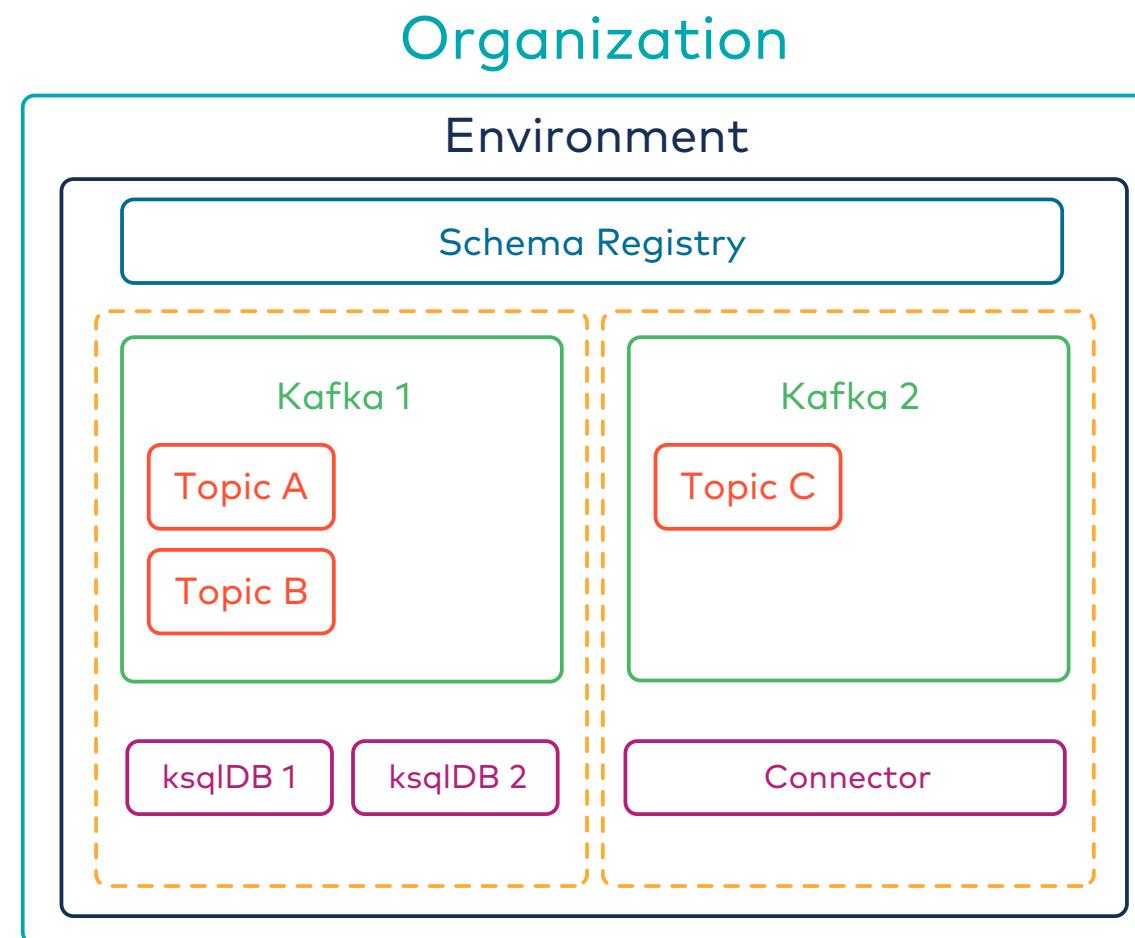
API Keys - Kafka App (2)



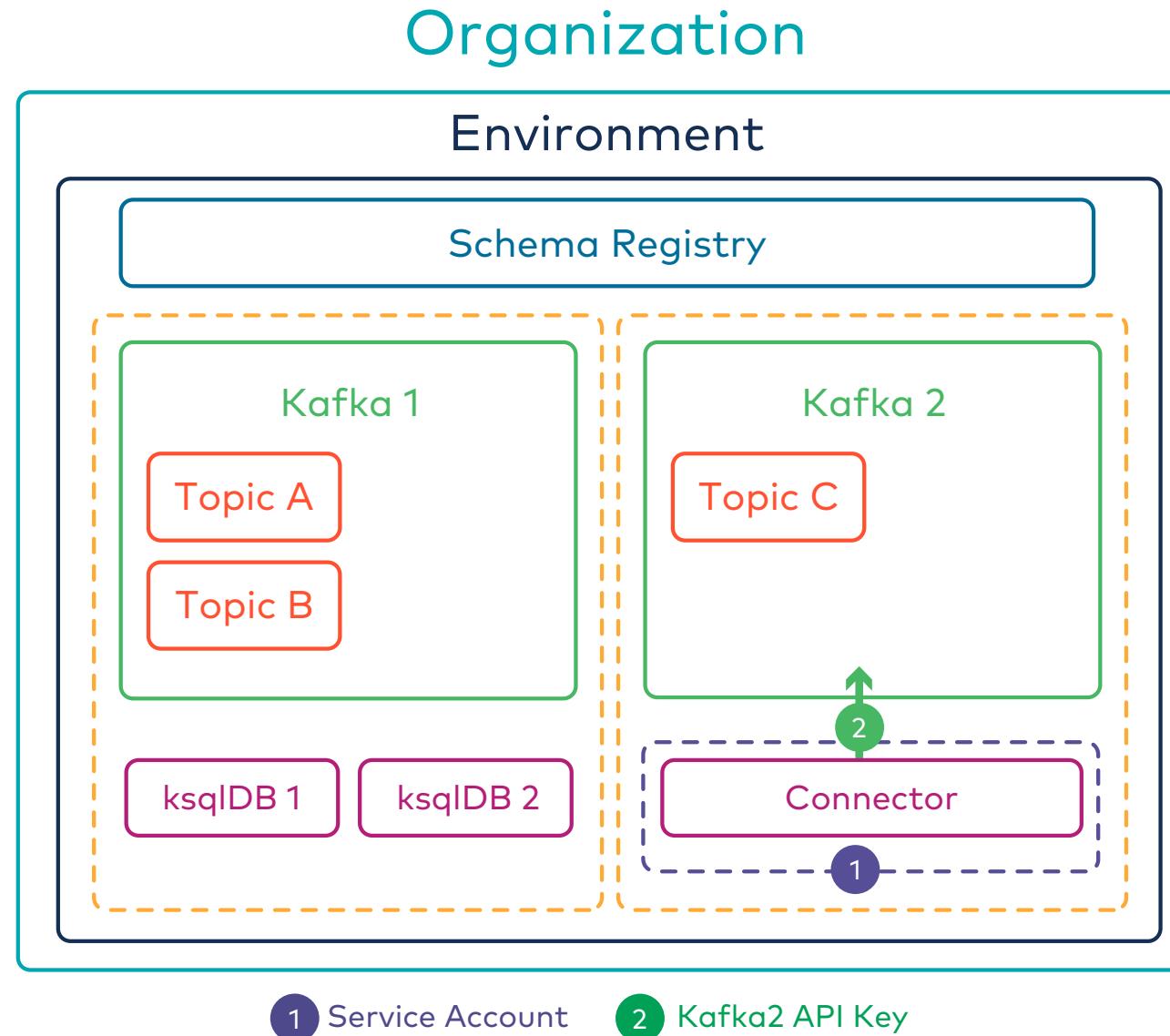
API Keys - Connector (1)

Question:

You want to create a **Connector** associated to **Kafka 2** to write to **Topic C**. What API key/s will be required?



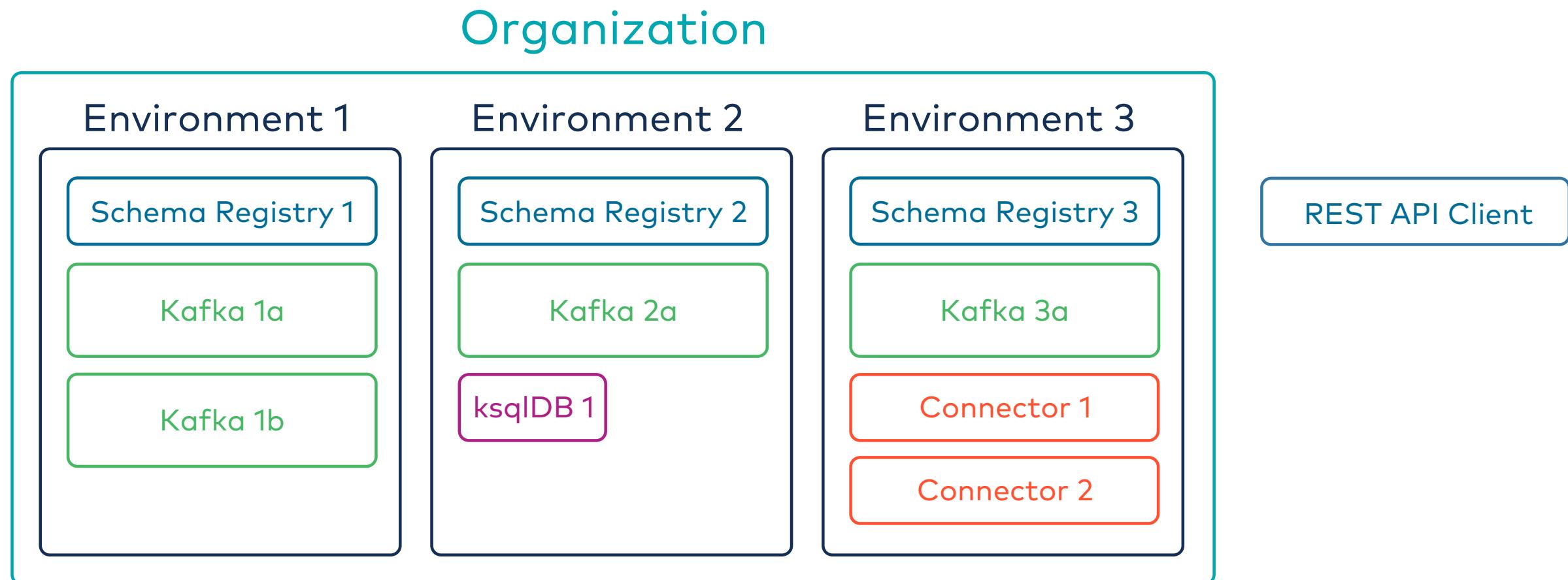
API Keys - Connector (2)



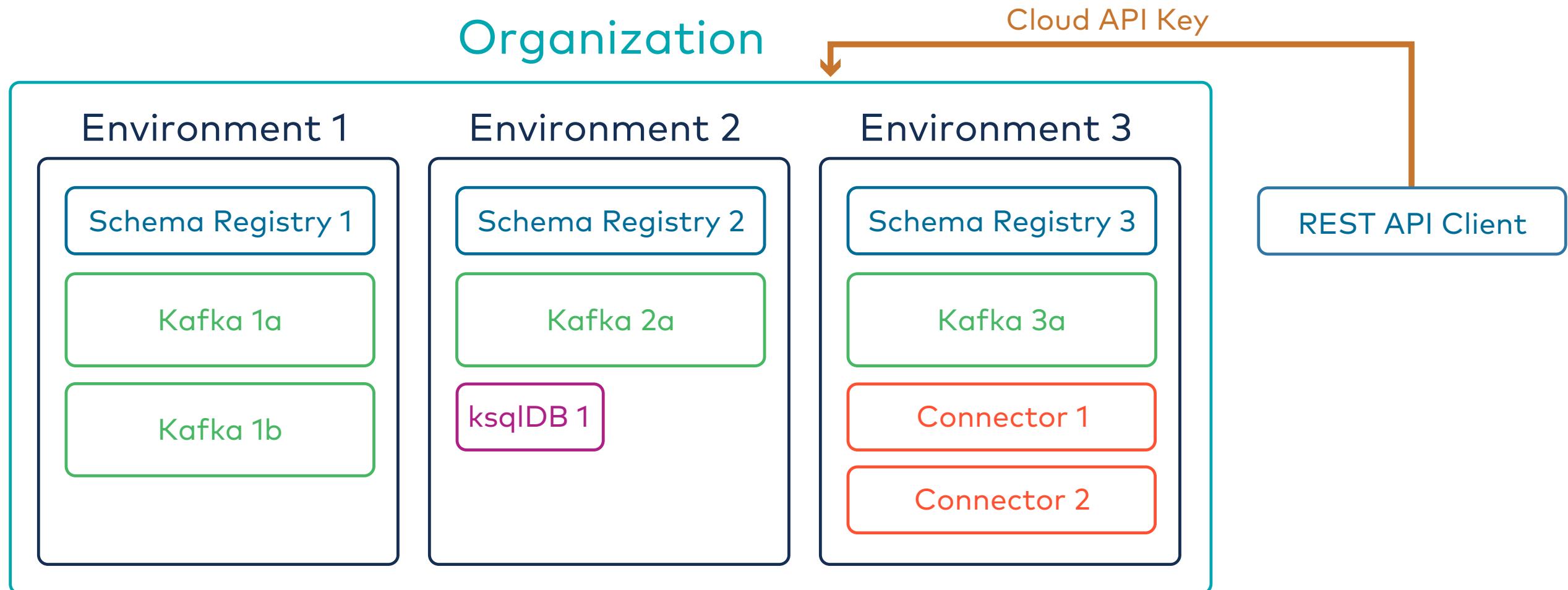
API Keys - Cloud API Key (1)

Question:

You have an Application that wants to monitor the Confluent Cloud performance using the Metrics API. What API key/s will be required?



API Keys - Cloud API Key (2)

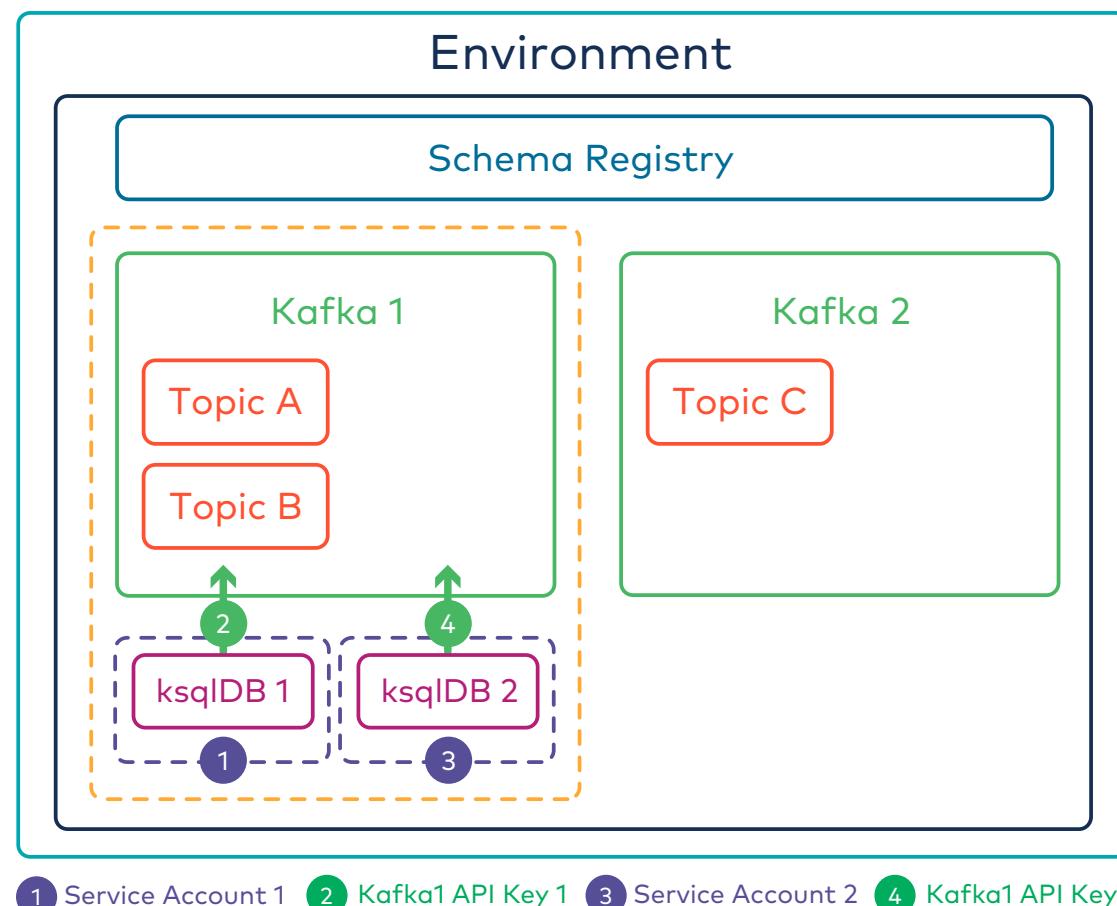


API Keys - ksqlDB clusters to Kafka cluster

Scenario:

You have two ksqlDB clusters associated with **Kafka 1**. These ksqlDB clusters need access to **Kafka 1** to consume the input data and to produce the output of the transformations.

Organization



Managing API Keys

All API keys can be managed using the Confluent CLI:

```
$ confluent api-key list  
$ confluent api-key create --resource lksqlc-xxxxx --service-account sa-xxxxx  
$ confluent api-key delete ESP40MLMFQRNEXOC  
$ confluent api-key list --service-account sa-xxxxx
```



All API keys can be managed using Confluent Cloud Console, except ksqlDB API keys.

Best Practices for Using API Keys

- Ensure only service account API keys are used in production
- User account API keys recommended only for development and testing
- Delete unneeded API keys and service accounts
- Rotate API keys regularly:
 1. Create a new API key
 2. Update the resource or application to use the new API key
 3. Delete the old API key

Role-Based Access Control (RBAC)

Access Control for **User** and **Service** accounts based on predefined roles

- Can control access to:
 - Organization
 - Environment
 - Kafka cluster
 - Schema Registry
 - ksqlDB cluster
 - Pipelines
 - Network
- There are 15 predefined roles:
 - `OrganizationAdmin`
 - `EnvironmentAdmin`
 - `CloudClusterAdmin`
 - `Operator`
 - `KsqlAdmin`
 - `NetworkAdmin`
 - `MetricsViewer`
 - and more...

Access Control Lists (ACLs)

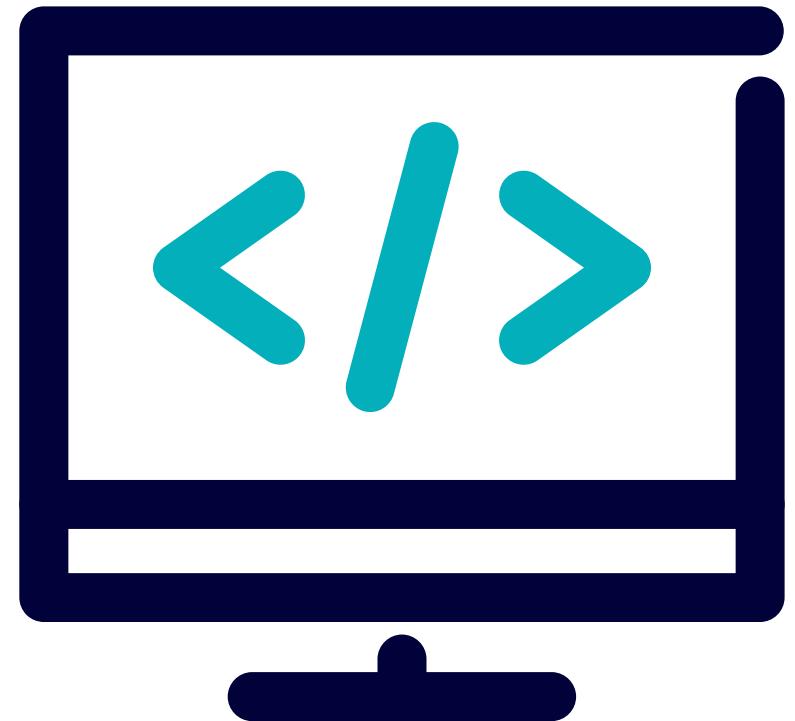
- Provide secure access to your Confluent Cloud resources and data
- Can be applied to **User** and **Service** accounts
- Prefix matching supported

Cluster	Consumer group	Topic	Transactional ID
Topic name*	vehicle-sensors		
Pattern type*	LITERAL		
Operation*	WRITE	Permission*	ALLOW
Delete ACL			

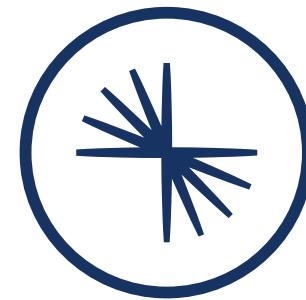
Lab Module 02: Accessing Confluent Cloud

Please work on **Lab Module 02: Accessing Confluent Cloud**.

Refer to the Exercise Guide.



03: Schema Registry in Confluent Cloud



CONFLUENT
Global Education

Module Overview



This module contains four lessons:

- Stream Governance
- Stream Quality: Schema Registry
- Stream Quality: Schema Validation
- Stream Quality: Schema Linking

After this module you will be able to:

- Create schemas and plan schema evolution for your data using Schema Registry.
- Enable Schema Validation in your topics to prevent poisoning your data pipelines.
- Run and configure Schema Linking to replicate your schemas between Confluent Cloud clusters.

03a: Stream Governance

Description

Learn why Stream Governance is important in Confluent Cloud environments.

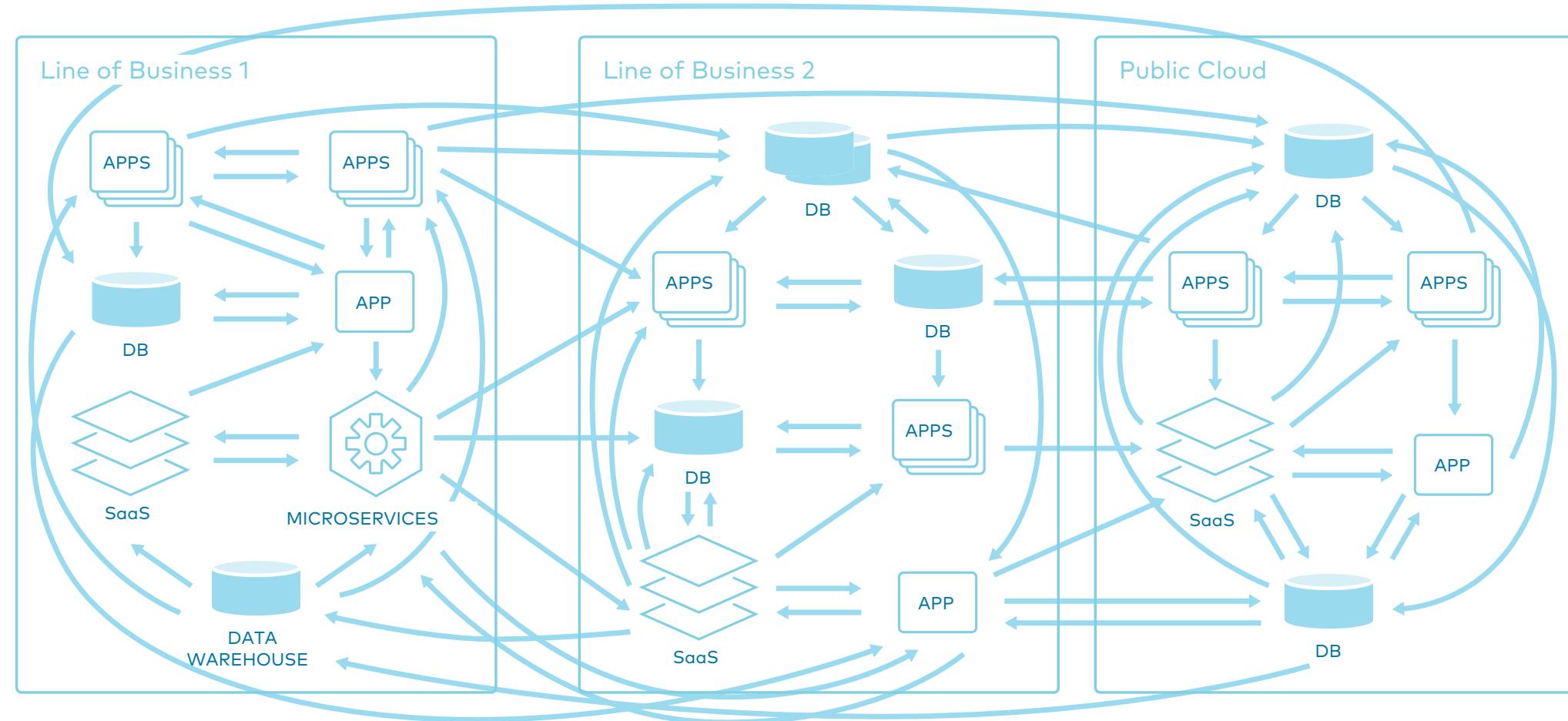
Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

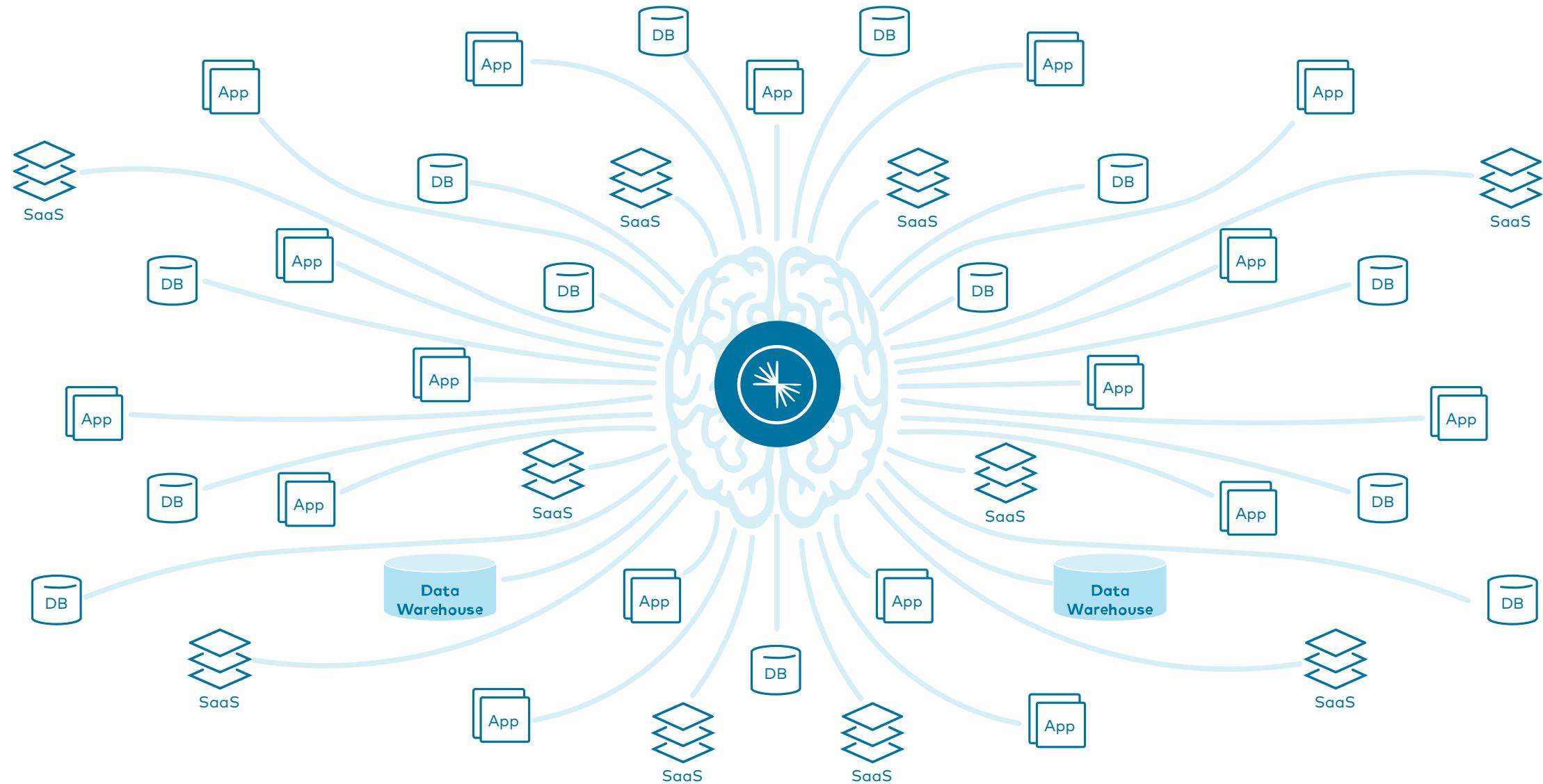
- Motivation for using Stream Governance in Confluent Cloud.

Stream Governance - Data at Rest



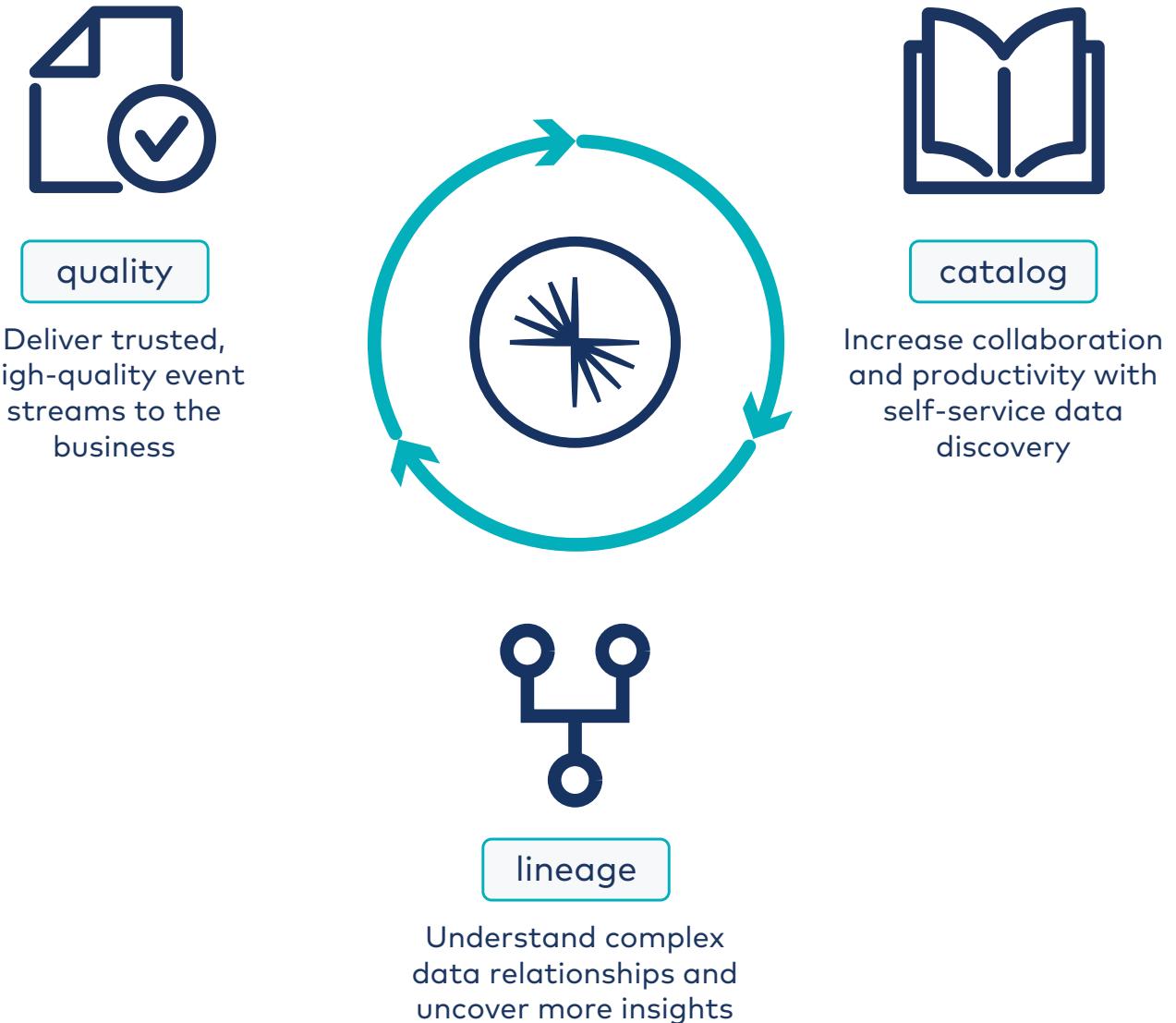
Stream Governance - Data in Motion

The Data in Motion Platform



Stream Governance - Why is it Important?

- With the explosion in volume, variety and velocity of data, companies are demanding tools to control and manage their data.
- Stream Governance aims to manage the availability, integrity, and security of data used across an organization upon three key strategic pillars.



03b: Stream Quality: Schema Registry

Description

Use Schema Registry in Confluent Cloud to ensure stream quality.

Learning Objectives



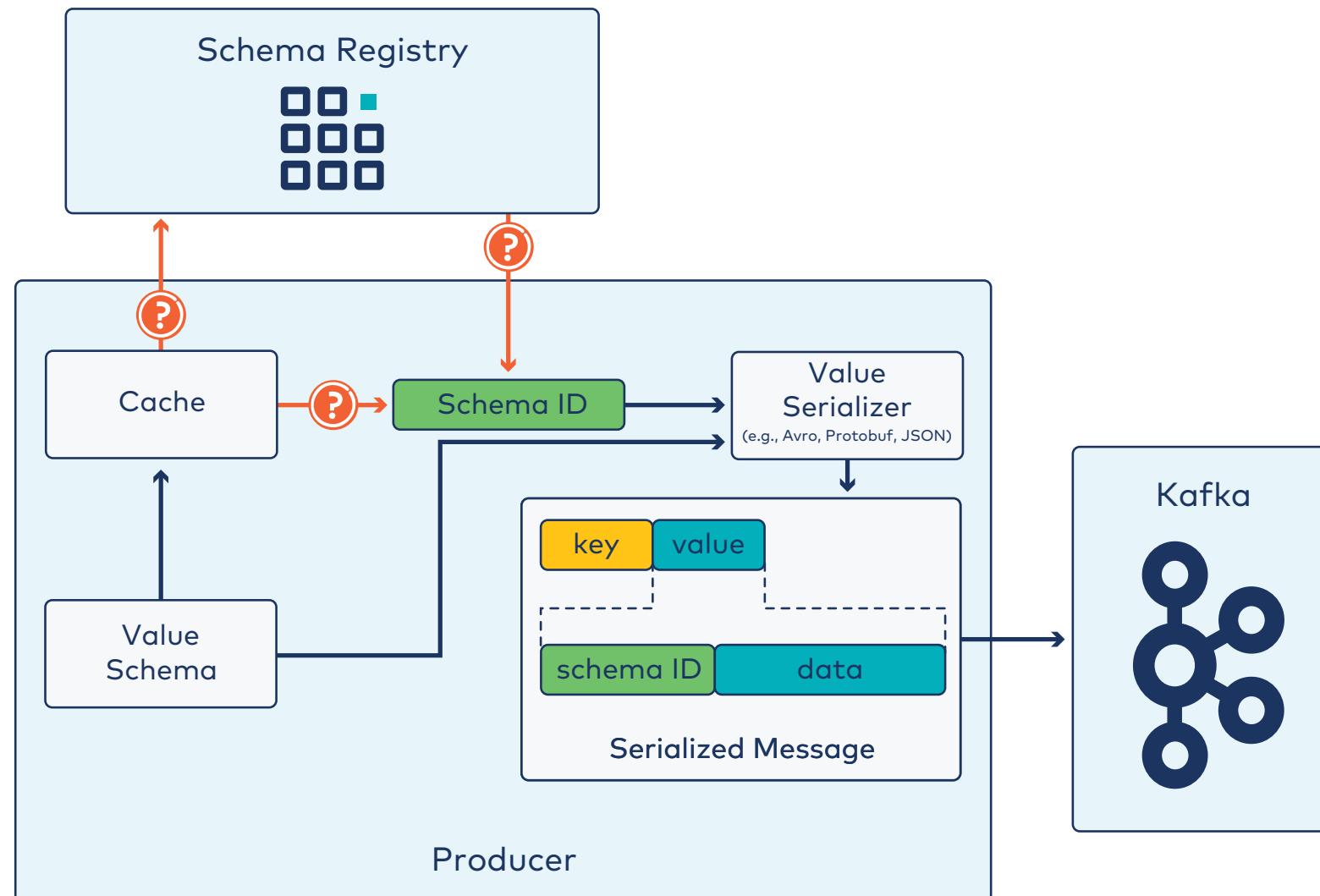
Upon completion of this lesson and associated lab exercises, you will be able to:

- Overview of how Schema Registry works.
- Considerations and best practice when defining schemas.
- Explanation about the different ways to access Schema Registry.

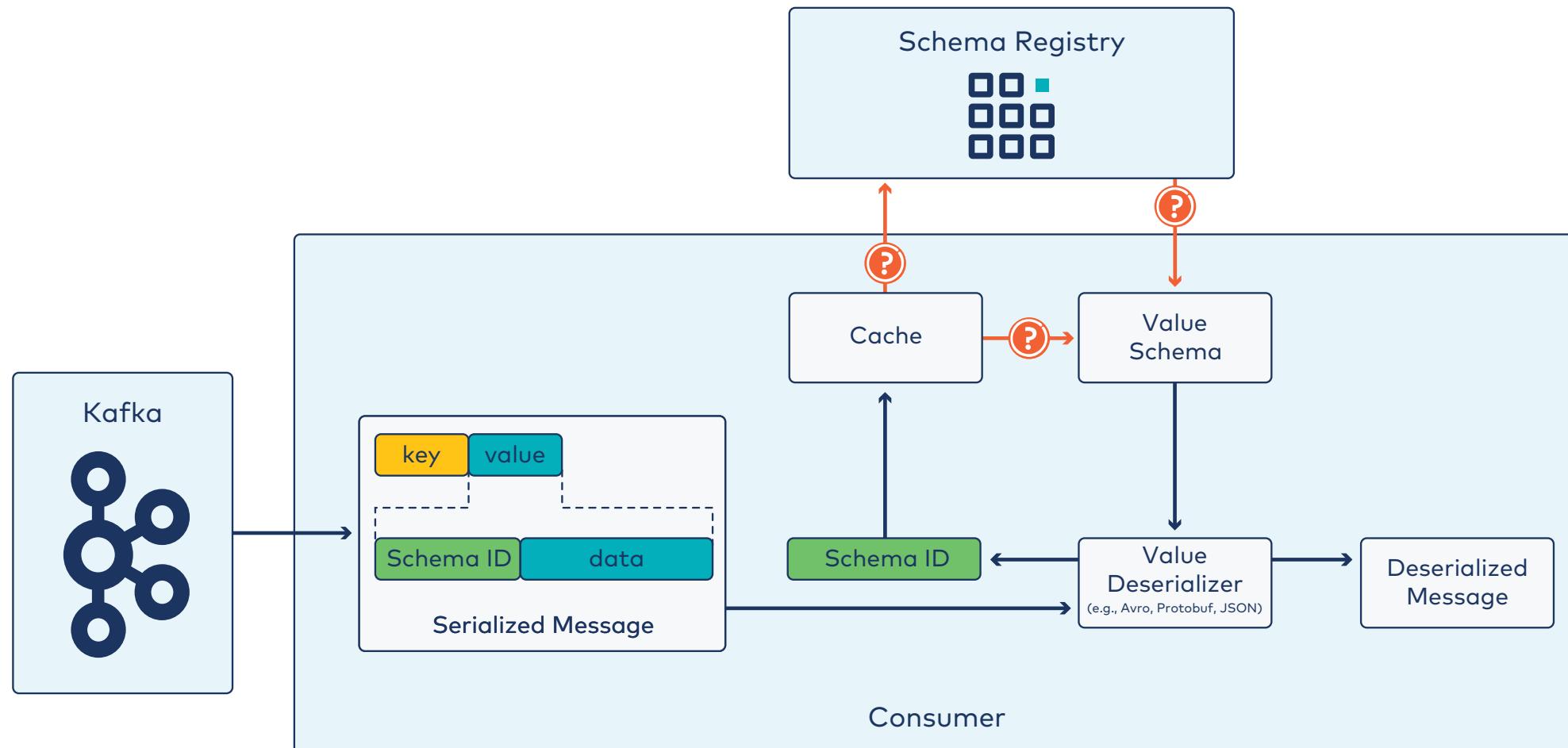
Schema Registry

- Confluent Schema Registry provides a centralized management of schemas:
 - Stores a versioned history of all schemas
 - Provides a RESTful interface for storing and retrieving schemas
 - Checks schemas and throws an exception if data does not conform to the schema
 - Allows evolution of schemas according to the configured compatibility setting
- Sending the schema with each message would be inefficient.
 - Instead, a globally unique ID representing the schema is sent with each message
- The Schema Registry is accessible via a Confluent Cloud Console, Confluent CLI and REST API.

Schema Registry Data Flow - Producer



Schema Registry Data Flow - Consumer



Integration with Schema Registry

- Java clients (producer, consumer)
- ksqlDB
- Kafka Streams
- Kafka Connect
- Confluent REST Proxy
- Non-Java clients based on [librdkafka](#)

Best Practices

Keep key schema complexity to a minimum to guarantee deterministic serialization and not break topic partitioning.

Key	Value
<ul style="list-style-type: none">• Simple non-serialized data type (string, UUID, long ID, etc.)• An Avro record that does not include maps or arrays	<ul style="list-style-type: none">• Avro• Protobuf• JSON Schema

Limits

- 1,000 free schemas per environment.
- Rate limits of API requests:
 - 25 requests per second for each API key.

Enable Schema Registry (I)

The screenshot shows the Confluent Cloud interface for a 'Staging' environment. The left sidebar has 'Environments' selected. The main area shows a cluster named 'inventory' (Running) with metrics like 0B/s for Production and Consumption, and 0B for Storage. The 'Schema Registry' tab is active in the top navigation bar. On the right, there's a sidebar with sections for 'Description', 'Tags', 'Stream Governance package', 'Schemas', 'Tags', and 'Business metadata'. A prominent red box highlights a callout for 'Stream Governance API' with the text: 'Enable Stream Governance to access Schema Registry, Stream Catalog and Stream Lineage.' An arrow points from this box to a blue 'Enable now' button. The bottom of the page has a support link.

Stream Catalog LEARN ?

HOME > ENVIRONMENTS >

Home Environments Cluster links Stream shares

Staging

Clusters Network management Schema Registry

Search cluster name or id + Add cluster

Live (1)

inventory Running

Metrics

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

Resources

ksqlDB 0	Connectors 0	Clients 0
----------	--------------	-----------

Overview

ID lkc-pk5yno	Type Basic	Provider & region AWS us-east-2
---------------	------------	-----------------------------------

Schemas Tags Business metadata

Stream Governance API Schema Registry and Stream Catalog API

ID --

Endpoint --

Credentials --

Enable Stream Governance to access Schema Registry, Stream Catalog and Stream Lineage.

Enable now

Support

Enable Schema Registry (II)

Stream Governance Packages

Confluent's Stream Governance suite establishes trust in the data streams moving throughout your cloud environments and delivers an easy, self-service experience for more teams to discover, understand, and put streaming data to work.



Essentials

The fundamentals for getting started.

Stream Quality
Schema Registry & validation

- 99.5% uptime SLA
- 1,000 schemas included*
- 9 cloud regions supported

Stream Catalog
Data organization & discoverability

- Auto-technical metadata ingestion
- Tags metadata
- Cloud UI & REST API

Stream Lineage
Data origin & tracking

- Real-time data streams lineage

*Starting at \$0.002/schema/hour after 1,000 schemas per environment

Begin configuration

Starting at
FREE

Upgrade to Advanced at any time



Advanced

Enterprise-ready controls for data in motion.

Stream Quality
Schema Registry & validation

- 99.95% uptime SLA
- 20,000 schemas included
- 31 cloud regions supported

Stream Catalog
Data organization & discoverability

- All existing Essentials features +
 - Business metadata
 - GraphQL API

Stream Lineage
Data origin & tracking

- All existing Essentials features +
 - Point-in-time lineage
 - Lineage search

Begin configuration

Starting at
\$1 /hr

The full Stream Governance feature set

Enable Stream Governance Essentials

Select the cloud provider and region where you want the environment Schema Registry and Stream Catalog to run and metadata to be stored. [Learn more](#)

 The cloud provider and region cannot be changed once you enable the environment package.



Region*

United States

Ohio (us-east-2)
+\$0 /hr

N. Virginia (us-east-1)
Upgrade required

Oregon (us-west-2)
Upgrade required

Europe

Frankfurt (eu-central-1)
+\$0 /hr

Enable Schema Registry (III)

The screenshot shows the Confluent Cloud interface for managing environments. On the left, the 'Environments' tab is selected, showing a 'Staging' environment with one 'Live' cluster named 'inventory'. The cluster details show 0B/s production, consumption, and storage. The 'Metrics' section shows 0 ksqlDB instances, 0 connectors, and 0 clients. The 'Overview' section provides the cluster ID (lkc-pk5yno), type (Basic), and provider & region (AWS | us-east-2). To the right, the 'Stream Governance package' configuration is displayed, which includes:

- Schemas**: A red arrow points to the 'Schemas' section.
- Schema Registry ID**: A red arrow points to the 'ID' field containing 'lsrc-q8nzvm'.
- Schema Registry URL**: A red arrow points to the 'Endpoint' field containing 'https://psrc-nx65v.us-east-2.aws.confluent.cloud'.
- Schema Registry API Keys**: A red arrow points to the 'Credentials' section with a '+ Add key' button.

Access to Schema Registry: Confluent CLI

Create a schema

```
confluent schema-registry schema create --subject <subject-name> --schema <path-to-file> --type  
<AVRO|PROTOBUF|JSON>
```

List all subjects

```
confluent schema-registry subject list
```

Describe a schema

```
confluent schema-registry schema describe <schemaID>
```

```
confluent schema-registry schema describe --subject <subject-name> --version <version-number|latest>
```

Access to Schema Registry: REST API

Create a schema:

```
curl -X POST -u <api-key:api-secret> -H "Content-Type: application/json" --data <json-payload> <schema-registry-url>/subjects/<subject-name>/versions
```

List all subjects:

```
curl -X GET -u <api-key:api-secret> <schema-registry-url>/subjects
```

Retrieve a schema by ID:

```
curl -X GET -u <api-key:api-secret> <schema-registry-url>/schemas/ids/<schema-id>
```

```
{  
  "schema":  
  "{  
    \"namespace\": \"clients.avro\",  
    \"type\": \"record\",  
    \"name\": \"PositionValue\",  
    \"fields\": [  
      {\"name\": \"latitude\",  
       \"type\": \"double\"},  
      {\"name\": \"longitude\",  
       \"type\": \"double\"}  
    ]  
  }",  
  "schemaType": "AVRO"  
}
```



<json-payload> must contain the fields: "schema"; "schemaType" (AVRO/PROTOBUF/JSON)

Access to Schema Registry: Confluent Cloud Console - Demo

The screenshot shows the Confluent Cloud Schema Registry interface. The top navigation bar includes the Confluent logo, a search bar labeled "Stream Catalog", a "LEARN" button, a notification bell, a help icon, and a menu icon. The breadcrumb path is "HOME > ENVIRONMENTS > STAGING > SCHEMAS >". On the left, a sidebar lists "Home", "Environments" (which is selected and highlighted in grey), "Cluster links", and "Stream shares". The main content area displays the schema for the topic "test-topic-value". The schema is defined as Avro, with compatibility mode set to "Backward" and no topics used. It is version 1 (current) with Schema ID 100001. The schema structure is as follows:

```
sampleRecord (3) +

- doc: Sample schema to help you get started.
- namespace: com.mycorp.mynamespace

fields (3)

- 0 (3) +
  - name: my_field1
  - doc: The int type is a 32-bit signed integer.
  - type: int
- 1 (3) +

```

03c: Stream Quality: Schema Validation

Description

Use Schema Validation in Confluent Cloud to ensure stream quality.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Motivation to use schema validation.
- How to configure schema validation in Confluent Cloud and examples.

Why Validate Schemas?

It allows you to quickly identify and block misconfigured producers before they poison your data pipelines.

Schema validation checks for:

- The message has an associated schema (a schema ID is prepended).
- The schema must match the topic (the schema ID is registered under the proper subject name).

Considerations

- Schema validation is configured at the topic level.
- Value schema and key schema validation are independent of each other; you can enable either or both.
- Schema validation is supported on only dedicated clusters.

Activity

In the following examples, will the record be appended or rejected?

- For topic `order` with schema validation enabled for its value field:
 - Record with a schema-ID in the value that can be found in the `order-topic-value` subject in the Schema Registry
 - Record with a schema-ID just in the key of the message that can be found in the `order-key` subject in the Schema Registry
 - Record with a schema-ID in the value that can be found under the `order-value` subject in the Schema Registry

Schema ID	Subject
Value	<code>order-topic-value</code>
Key	<code>order-key</code>
Value	<code>order-value</code>

Activity Solution

For topic `order` with schema validation enabled for its value field...

Record with a schema-ID for the value that can be found under the `order-topic-value` subject



Record rejected

Record with a schema-ID just in the key of the message that can be found in the `order-key` subject



Record rejected

Record with a schema-ID for the value that can be found under the `order-value` subject



Record appended

Configuration Options

Name	Description	Default
<code>confluent.key.schema.validation</code>	When set to <code>true</code> , enables schema ID validation on the message key	<code>false</code>
<code>confluent.value.schema.validation</code>	When set to <code>true</code> , enables schema ID validation on the message value	<code>false</code>
<code>confluent.key.subject.name.strategy</code>	The subject name strategy for the message key	<code>TopicNameStrategy</code>
<code>confluent.value.subject.name.strategy</code>	The subject name strategy for the message value.	<code>TopicNameStrategy</code>

Enabling Schema Validation

From the Confluent CLI:

```
confluent kafka topic <create|update> <topic-name>
--config confluent.<key|value>.schema.validation=true
--config confluent.<key|value>.subject.name.strategy=<strategy>
```

From the Confluent Cloud Console:

1. Navigate to a topic.
2. Click the **Configuration** tab.
3. Click **Edit Settings**.
4. Click **Switch to expert mode**.
5. In Expert mode:
 - a. Change `confluent.value.schema.validation` and/or `confluent.key.schema.validation` from `false` to `true`.
 - b. Check `confluent.value.subject.name.strategy`, `confluent.key.subject.name.strategy`.

03d: Stream Quality: Schema Linking

Description

Use Schema Linking in Confluent Cloud to ensure stream quality.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

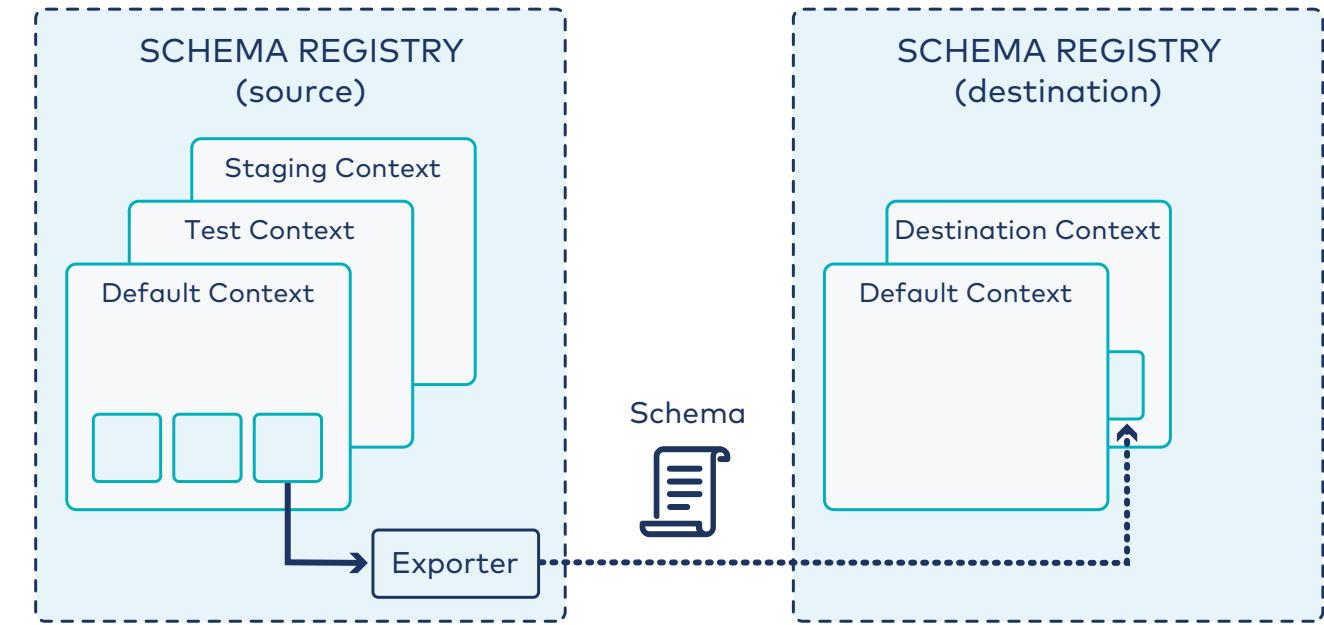
- Definition of schema linking and its components.
- Explanation about schema contexts, how to define them and how to access them using REST API and Kafka clients.
- Description of schema exporters and instructions to manage them.

Schema Linking Concepts

Schema linking keeps schemas in sync across two Schema Registry clusters.

There are two new concepts to support schema linking:

- Schema contexts
- Schema exporters



Schema Contexts (1)

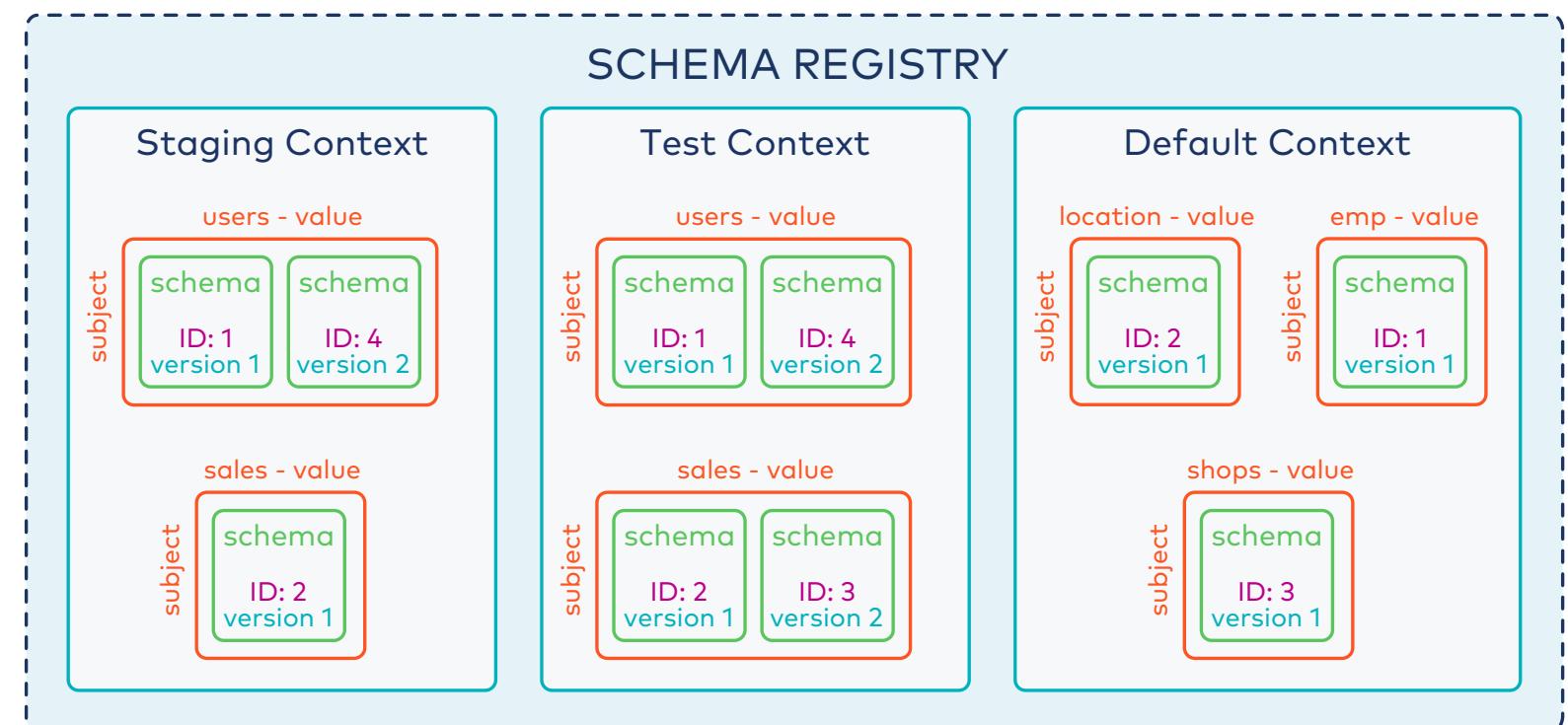
- A schema context is essentially a grouping of subject names and schema IDs.
- A single Schema Registry cluster can host any number of contexts.
- A context can be copied to another Schema Registry cluster, using a schema exporter.
- There is a default context called Default.

Schema Contexts (2)

Each context can be thought of as a separate "sub-registry."

So, it's possible to have two contexts in the same Schema Registry cluster which can each have:

- a schema with the same ID, such as `100001`
- a subject with the same name, such as `mytopic-value`



Schemas are scoped by context.

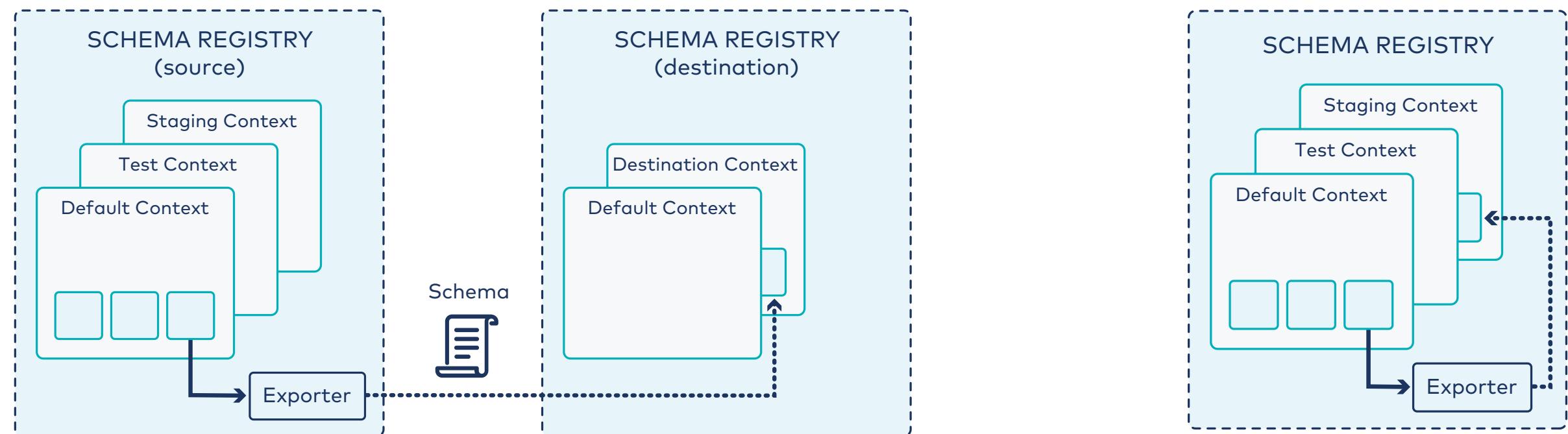
Schema Contexts - Qualified Subject Name

```
:.<context-name>:<subject-name>
```

- Default context is represented by a single dot: `.` (e.g., `:.:mysubject`).
- Any schema ID or subject name without an explicit context lives in the default context.
 - Example: If you create a subject named as `users-value`, the qualified subject name is `:.:users-value`
- Create a subject inside a context by using the qualified subject where the subject name is expected.
 - Example: By creating a new subject named as `:.test:users-value`, you are creating the subject `users-value` inside the context `test`

Schema Exporters - Overview

- A Schema Exporter can be used to copy schemas:
 - Between two Schema Registry clusters
 - From one context to another within the same Schema Registry cluster

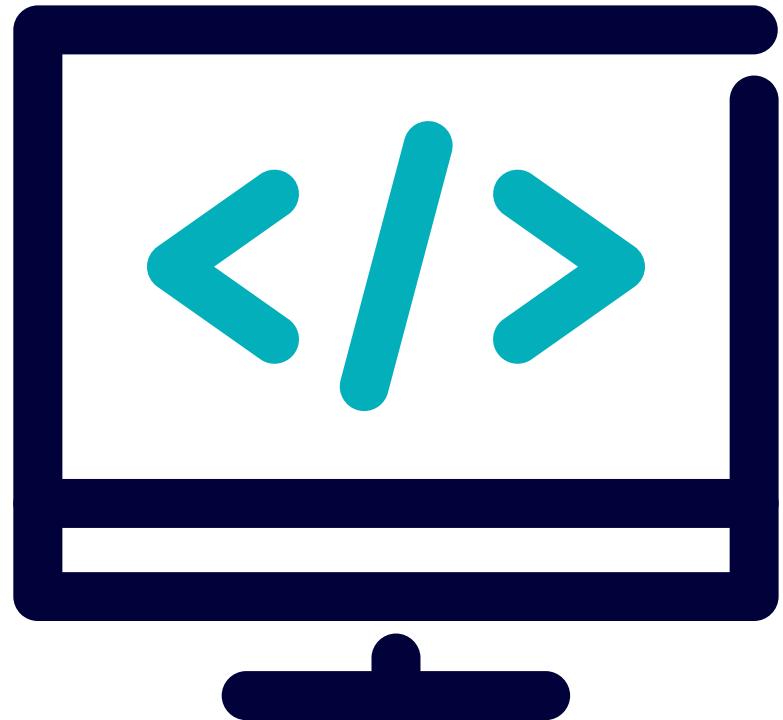


Maximum 10 exporters per Schema Registry.

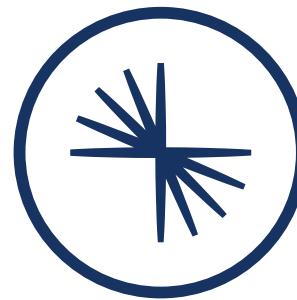
Lab Module 03: Working with Schema Linking

Please work on **Lab Module 03: Working with Schema Linking**.

Refer to the Exercise Guide.



04: Stream Lineage & Stream Catalog



CONFLUENT
Global Education

Module Overview



This module contains two lessons:

- Stream Lineage
- Stream Catalog

After this module you will be able to:

- Understand complex data relationships using Stream Lineage in Confluent Cloud.
- Perform data discovery (schemas, connectors, topics, etc.) to increase productivity and collaboration using Stream Catalog in Confluent Cloud.
- Tag your schema versions, records and fields.

04a: Stream Lineage

Description

Use Stream Lineage to understand the origins of the data.

Learning Objectives



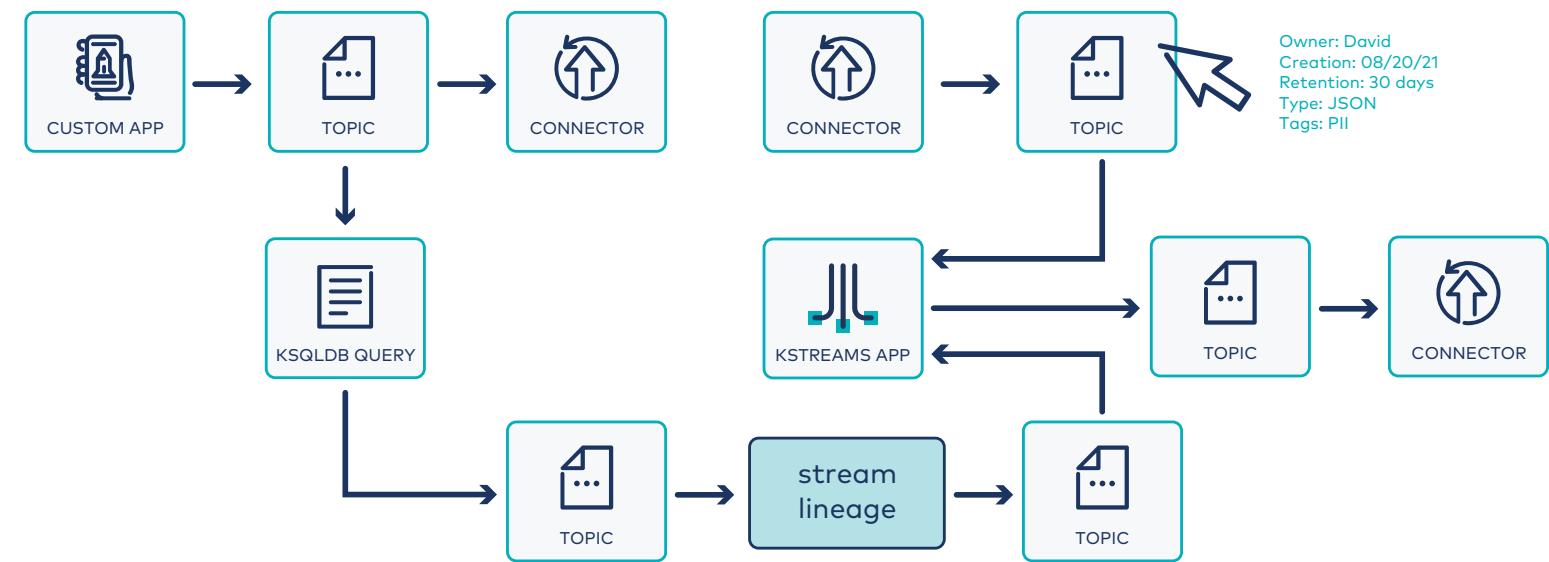
Upon completion of this lesson and associated lab exercises, you will be able to:

- What Stream lineage is and how to use it to understand your data pipelines in Confluent Cloud.
- Demo showing Stream lineage in action.

Stream Governance - Lineage

Comprehending the big picture
journey of data in motion.

- Where did the data come from?
- Where is it going?
- Where, when, and how was it transformed?



Stream lineage provides a graphical UI of event streams and data relationships with both a bird's eye view and drill-down magnification.

04b: Stream Catalog

Description

Use Stream Catalog to search and find what is needed.

Learning Objectives



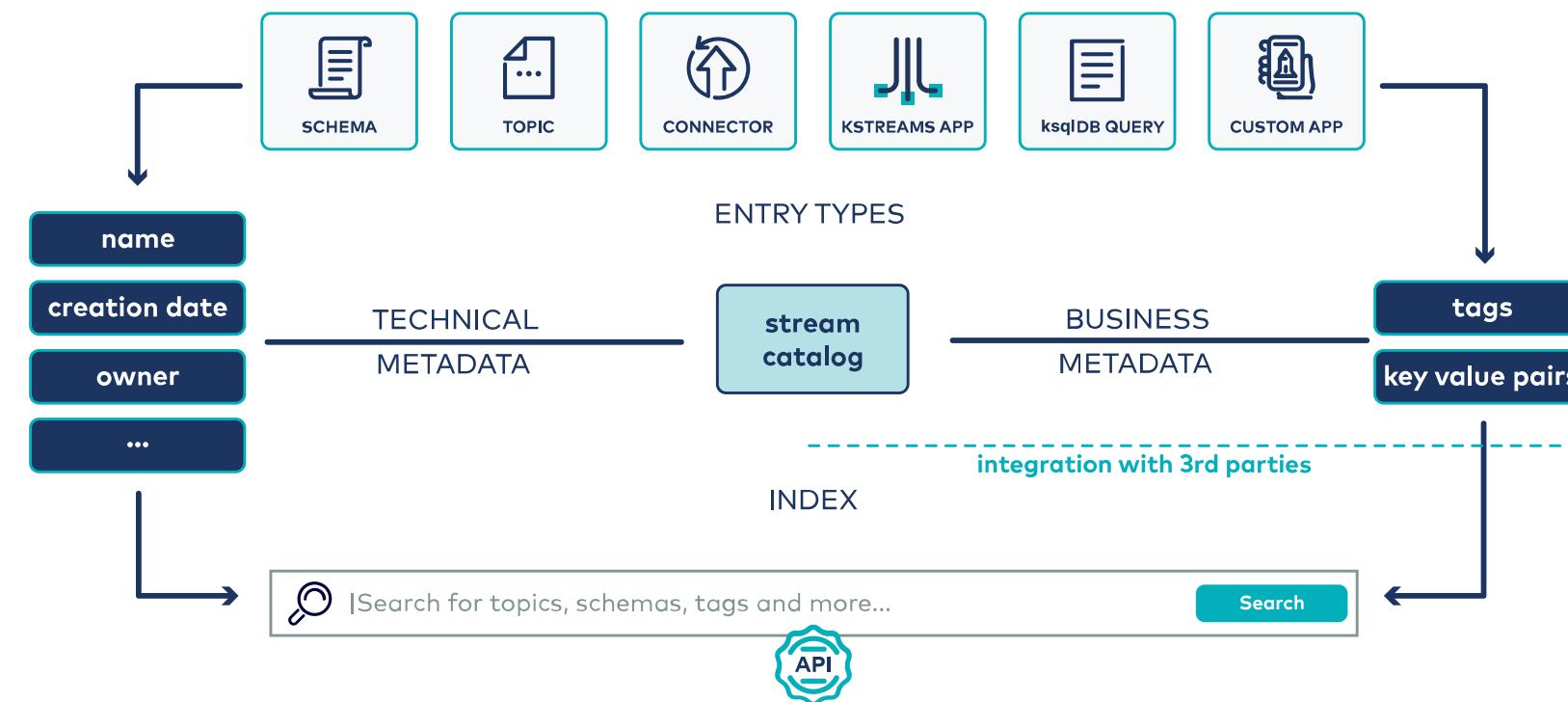
Upon completion of this lesson and associated lab exercises, you will be able to:

- What stream catalog is and how it can help to improve productivity.
- Definition of tags and demo showing how to use them.
- REST API interface to use stream catalog.

Stream Catalog for Stream Governance

The stream catalog is the digital library for data in motion in Confluent Cloud.

The stream catalog allows any user, experienced with Kafka or not, to search for what they need, find what's already been built, and put it to use right away.



Available through both the Cloud Console and API.

Tags

Organize data based on multiple concepts and categories:

- Tags can be applied to Environments, Topics, Connectors, Schema Subjects, Schema Records, Schema Fields
- Multiple tags can be associated to the same element
- Use provided tags (Public, Private, Sensitive, PII) and custom tags
 - Custom tag:
 - Tag name
 - Description

Business Metadata

Collection of attributes in the form of key-value pairs that provide more contextual information

Suppose you want to document or find out:

- Which team is responsible for a particular schema?
- Which product domain does a schema belong to?
- What is the GitHub location for a schema?

The screenshot shows a user interface for managing business metadata. At the top right, there is a 'Description' field containing the text: "All payments across our web and mobile applications. This data is sensitive so please be careful." Below it is a 'Tags' section with two tags: 'PCI' and 'Sensitive'. A link 'Add tags to this version' is also present. In the center, there is a schema definition with a red box around the 'Domain' section. The schema includes:

- Team_owner: Finance
- Slack_contact: #domain-payments
- Name: payments

A button 'Add business metadata' is located at the bottom right of the schema area.



- Business metadata is mainly used for defining extra information for entities
- Tags are used for organizing and classifying entities

Stream Governance - Catalog - Demo

In this demo we will see where to apply where to apply tags and do a quick search.

The screenshot shows the Confluent Stream Catalog interface for the schema `financial_txns-value`. The top navigation bar includes the Confluent logo, a search bar labeled "Stream Catalog", a "LEARN" button, and a three-dot menu icon. Below the navigation, the breadcrumb trail shows: HOME > ENVIRONMENTS > DEFAULT > SCHEMA REGISTRY > SCHEMAS.

The main content area displays the schema details for `financial_txns-value`:

- Type: JSON Schema
- Compatibility mode: Backward
- Used by topic: `financial_txns`
- Version 3 (current) Schema ID: 100038
- Search by keyword (highlighted with a red box)
- Compare versions
- Evolve schema
- More options (three dots)

The schema structure is shown on the left:

- `object` (6):
 - `type: object`
 - `additionalProperties: false`
 - `$schema: http://json-schema.org/draft-07/schema#`
 - `title: Transaction`
 - `definitions` (2):
 - `Company` (4)
 - `Customer` (4)
 - `properties` (5):
 - `amount` (1)
 - `company` (1)
 - `customer` (1)
 - `num_shares` (1)
 - `txn_ts` (1)

The right side of the interface contains the "Overview" section, which includes:

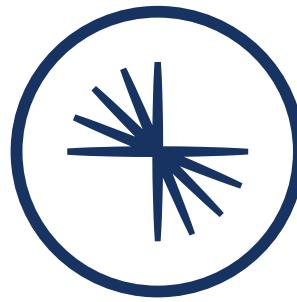
- Description (empty)
- Date created: Jan. 12 2022 4:46 PM
- Tags (highlighted with a red box):
 - No tags added
 - Add tags to this version
 - Sensitive
 - Public
 - Private
 - PII (selected)
 - my_stocks
 - PCI
 - test

Stream Governance - Catalog - REST API

Currently, catalog API provides search and tagging capabilities for schema entity types and topics:

- Search fields by name or tag
- Search schema by tag
- Search schema record by name
- Get tags from a field
- Create/List/Get tag definitions
- Tag a schema version, record or field
- Add business metadata to entities

05: Move Data with Cluster Linking



CONFLUENT
Global Education

Module Overview



This module contains four lessons:

- What is Cluster Linking?
- Most Common Use Cases
- Cluster Linking Lifecycle
- Cluster Linking Security Overview

After this module you will be able to:

- Create Cluster Links to sync topics between two clusters.
- Create mirror topics manually and with filters.
- Sync consumer group offsets and ACLs between clusters.
- Safely delete source and mirror topics.
- Apply appropriate security permissions.

05a: What is Cluster Linking?

Description

Cluster Linking is a fully managed service from Confluent designed to mirror topics from one cluster to another.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Describe how Cluster linking directly connects clusters and mirror topics from one cluster to another.

Cluster Linking Overview

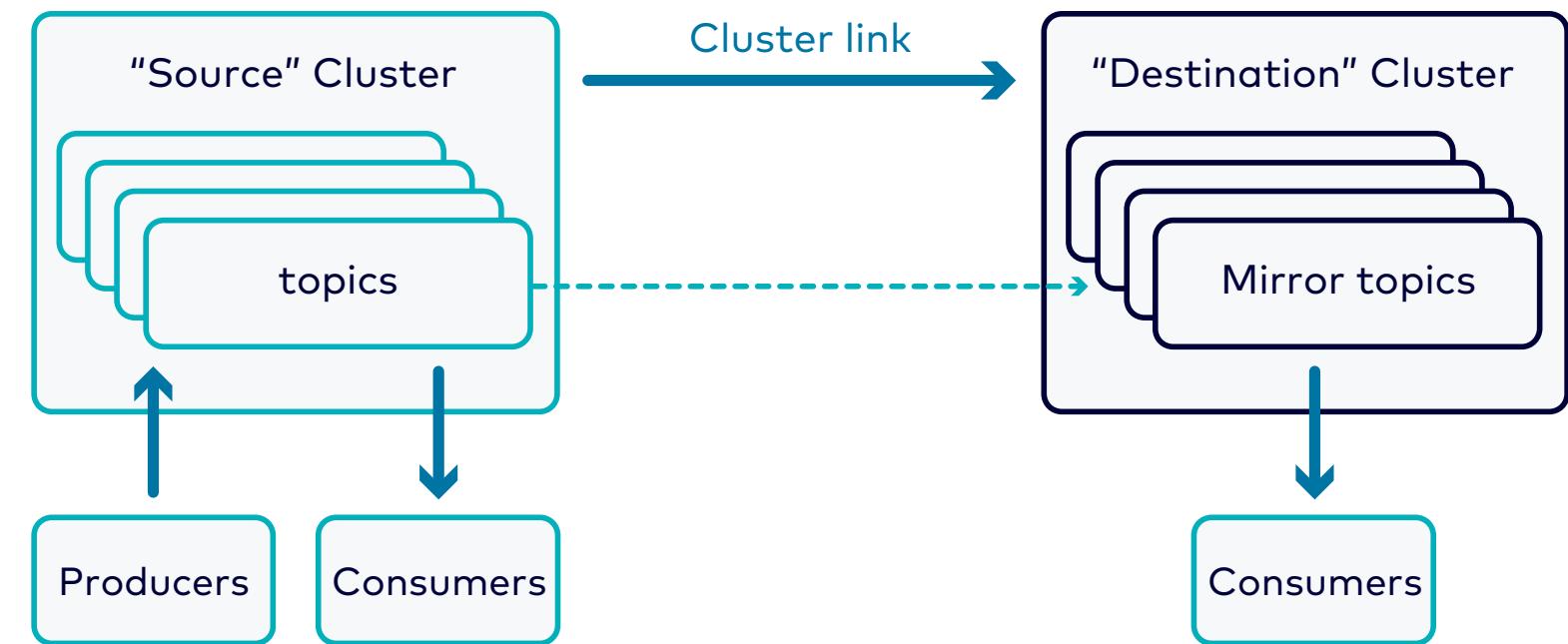
- Secure, performant and tolerant of network latency.
- Built into Confluent Cloud and Confluent Server.
- Replicate topics, consumer offsets and ACLs.
- Sync data between clusters in a different region, cloud, line of business, or organization.
- Makes it easy to build multi-datacenter, multi-region, and hybrid cloud deployments.



Cluster Linking Components

The building blocks of cluster linking are:

- A cluster link
- A mirror topic
- A source topic



Cluster Link

- A cluster link connects mirror topics to their source topics.
- The link is created and resides on the destination cluster.



This is a two step process.

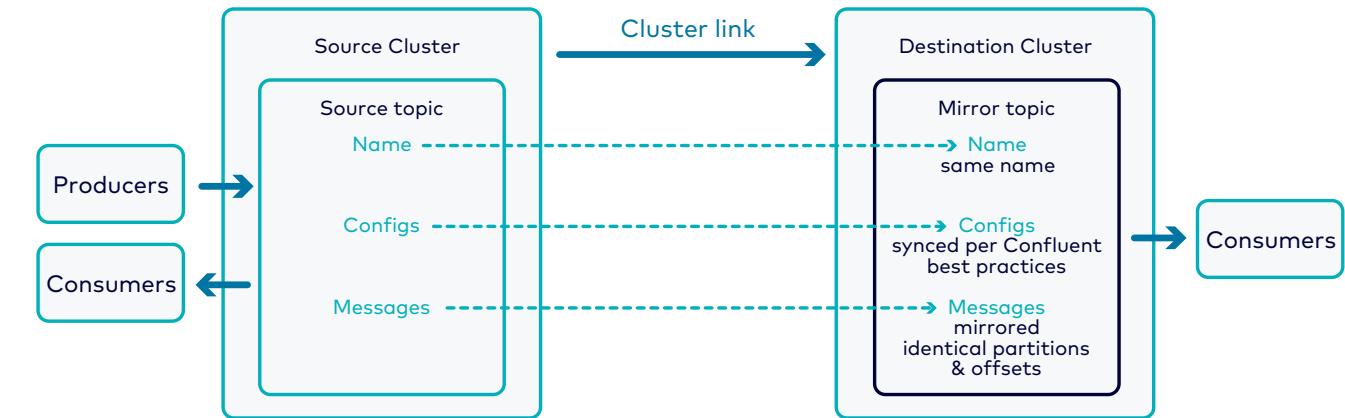
Example of the creation of a Cluster Link:

```
confluent kafka link create <link-name> \
--cluster <dst-cluster-id> \
--source-cluster-id <src-cluster-id> \
--source-bootstrap-server <src-bootstrap-url> \
--source-api-key <src-api-key> \
--source-api-secret <src-api-secret>
```

Question: What topics will this link sync?

Mirror Topic

- It's a special type of topic in the destination cluster.
- Created by and owned by a cluster link.
- Mirror topics are read-only.
- Byte-for-byte copy (offset-preservation).
- Many of the mirror topic's configurations are copied and synced from the source topic.
- Mirror topics **preserve topic names**.



This is the second step

Supported Cluster Combinations

Source Cluster	Destination Cluster
Confluent Cloud	Confluent Cloud
Confluent Platform	Confluent Platform
Apache Kafka	

Confluent Cloud Source and Destination Clusters

Source Cluster	Destination Cluster
<ul style="list-style-type: none">• Any Confluent Cloud cluster• Confluent Platform 5.4• Apache Kafka 2.4+	<ul style="list-style-type: none">• Dedicated Confluent Cloud cluster• Confluent Platform Enterprise

05b: Cluster Linking Lifecycle

Description

An overview of the lifecycle of a cluster link.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Create/delete a cluster link.
- Create/delete a mirror topic.

Creating a Cluster Link

1. Create Link

- Cluster link between source and destination clusters.

2. Create Mirror Topics

- Manually mirror individual topics.
- Auto-create mirror topics using filters.

CLI Tool to Create Cluster Linking

```
confluent kafka link create <link-name>
--cluster <dst-cluster-id>
--source-cluster-id <src-cluster-id>
--source-bootstrap-server <src-bootstrap-url>
--source-api-key <src-api-key>
--source-api-secret <src-api-secret>
```

Using API Key in Config File

```
confluent kafka link create <link-name>
--cluster <dst-cluster-id>
--source-cluster-id <src-cluster-id>
--source-bootstrap-server <src-bootstrap-url>
--config-file <file-name>
```

- Example of a config file:

```
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required username="<api-key>"
password="<api-secret>";
```

Creating a Mirror Topic

To create a mirror topic, you can use:

- Confluent CLI
- Confluent Cloud Console (UI)
- kafka-mirrors (binary from Confluent Platform)
- Confluent Cloud REST API
- Confluent AdminClient API
- Confluent for Kubernetes

Create Mirror Topic Using Confluent CLI

```
confluent kafka mirror create <topic-name> --cluster <destination-id> --link <link-name>
```

Create Mirror Topic Using kafka-mirrors (Binary from Confluent Platform)

```
kafka-mirrors --create  
--mirror-topic <topic-name>  
--link <link-name>  
--bootstrap-server <host:port>
```

05c: Cluster Linking Security Overview

Description

An overview of cluster linking security considerations.

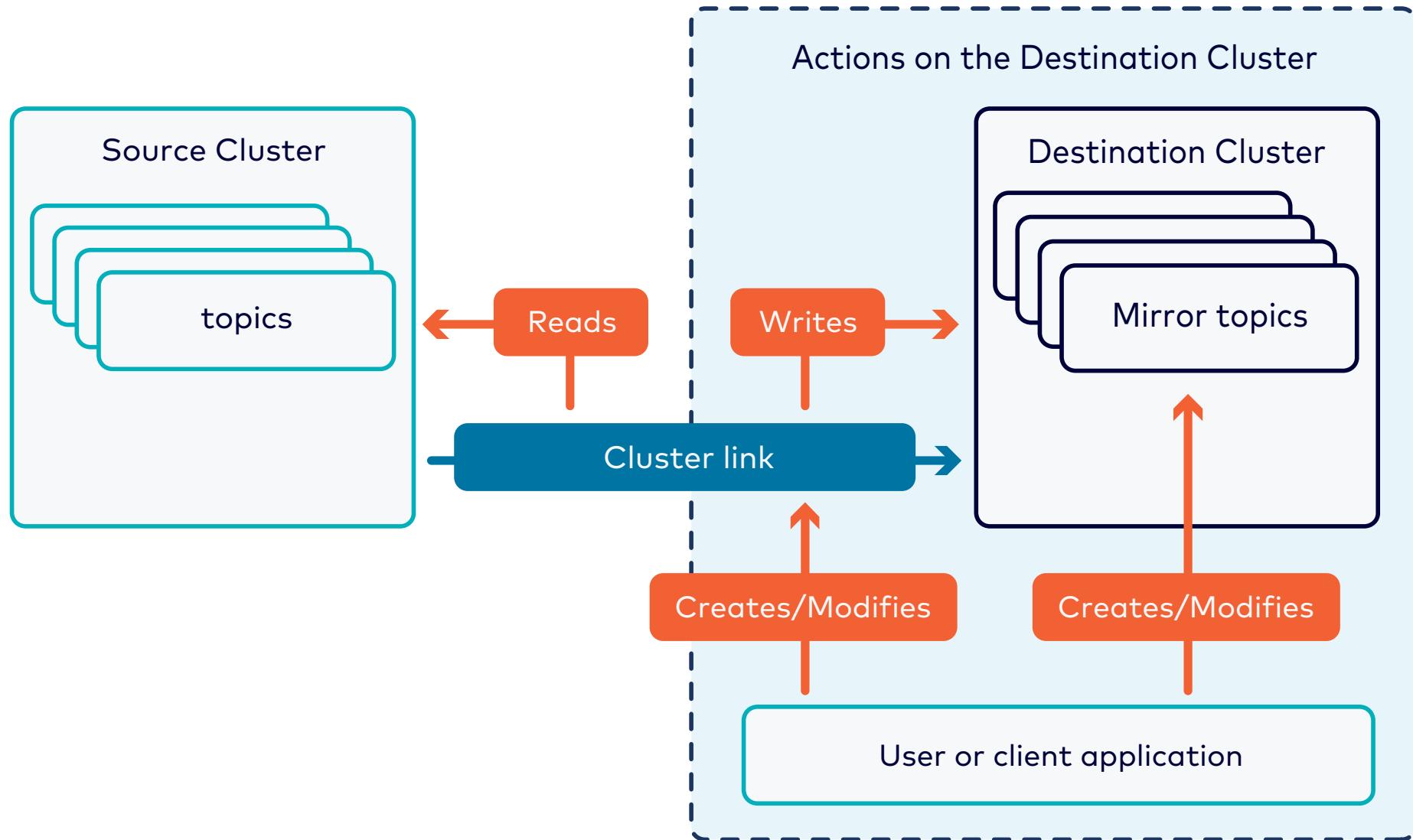
Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Define the operations of cluster linking.
- Describe permissions to manage cluster links and mirror topics.
- Evaluate service accounts for cluster links.
- Define other security considerations.

Cluster Linking Operations



Permissions to Manage Cluster Links and Mirror Topics

Principal	Operation	Cluster	Kafka RBAC Roles	Kafka ACLs
• User • Client app	Create or modify a <i>cluster link</i>	Destination	• CloudClusterAdmin • EnvAdmin • OrgAdmin	• ALTER: Cluster
• User • Client app	Create or modify a <i>mirror topic</i>	Destination	• CloudClusterAdmin • EnvAdmin • OrgAdmin	• ALTER: Cluster & ALTERCONFIGS: Cluster & CREATE: Topic

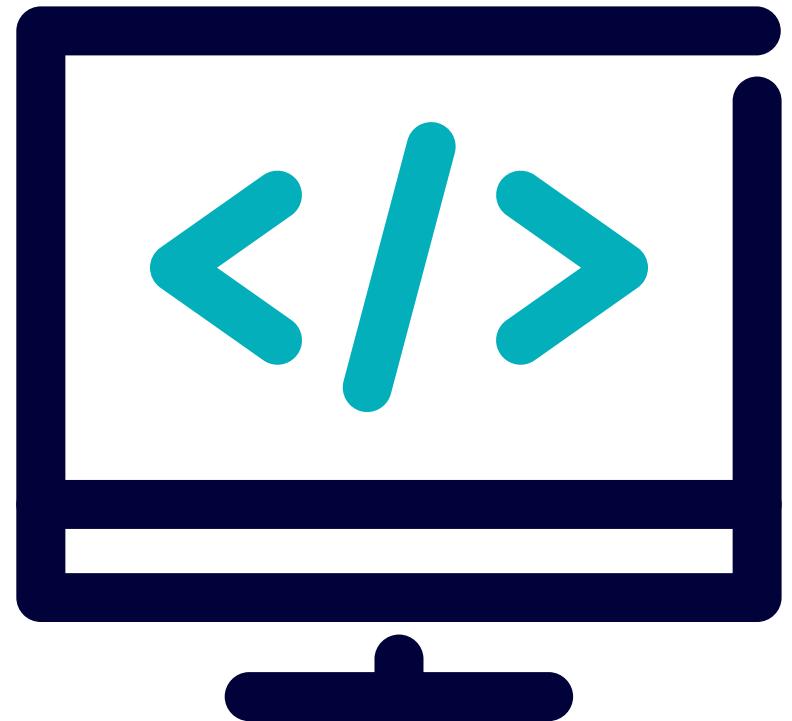


Use RBAC or ACL.

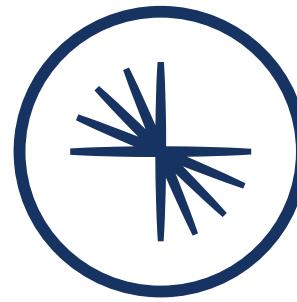
Lab Module 05: Cluster Linking

Please work on **Lab Module 05: Cluster Linking**.

Refer to the Exercise Guide.



06: Kafka Connect in Confluent Cloud



CONFLUENT
Global Education

Module Overview



This module contains 5 lessons:

- What is Connect
- Using Connectors in Confluent Cloud
- Security Considerations
- Networking with External Systems
- Self-Managed Connectors

After this module you will be able to:

- Explain the motivation for Kafka Connect
- Define what the main components are in Kafka Connect
- Use different Connect features in Confluent Cloud
- Apply the correct security settings for your connectors
- Explain how to deploy self-managed connectors

06a: What is Connect?

Description

Connect is used to import and export data into and out of Confluent Cloud.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Describe use cases.
- Define source and sink connectors.
- Scale connectors.
- Evaluate converters.

Required: Data from Another System in Kafka; Kafka Data to Another System

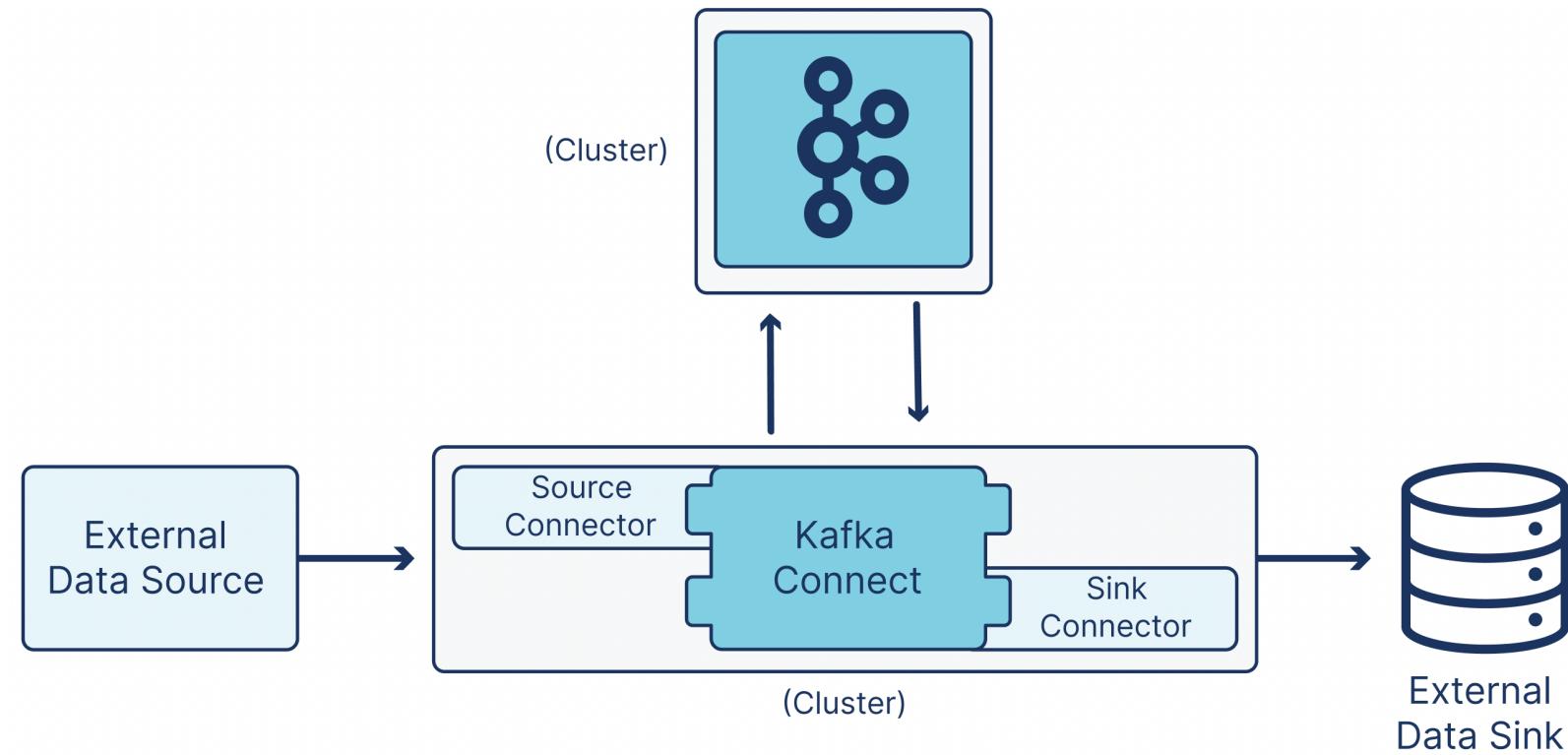
Suppose you have:

- Data in some external system and you want to get it into the Kafka
- Data in the Kafka and want to export it to another external system

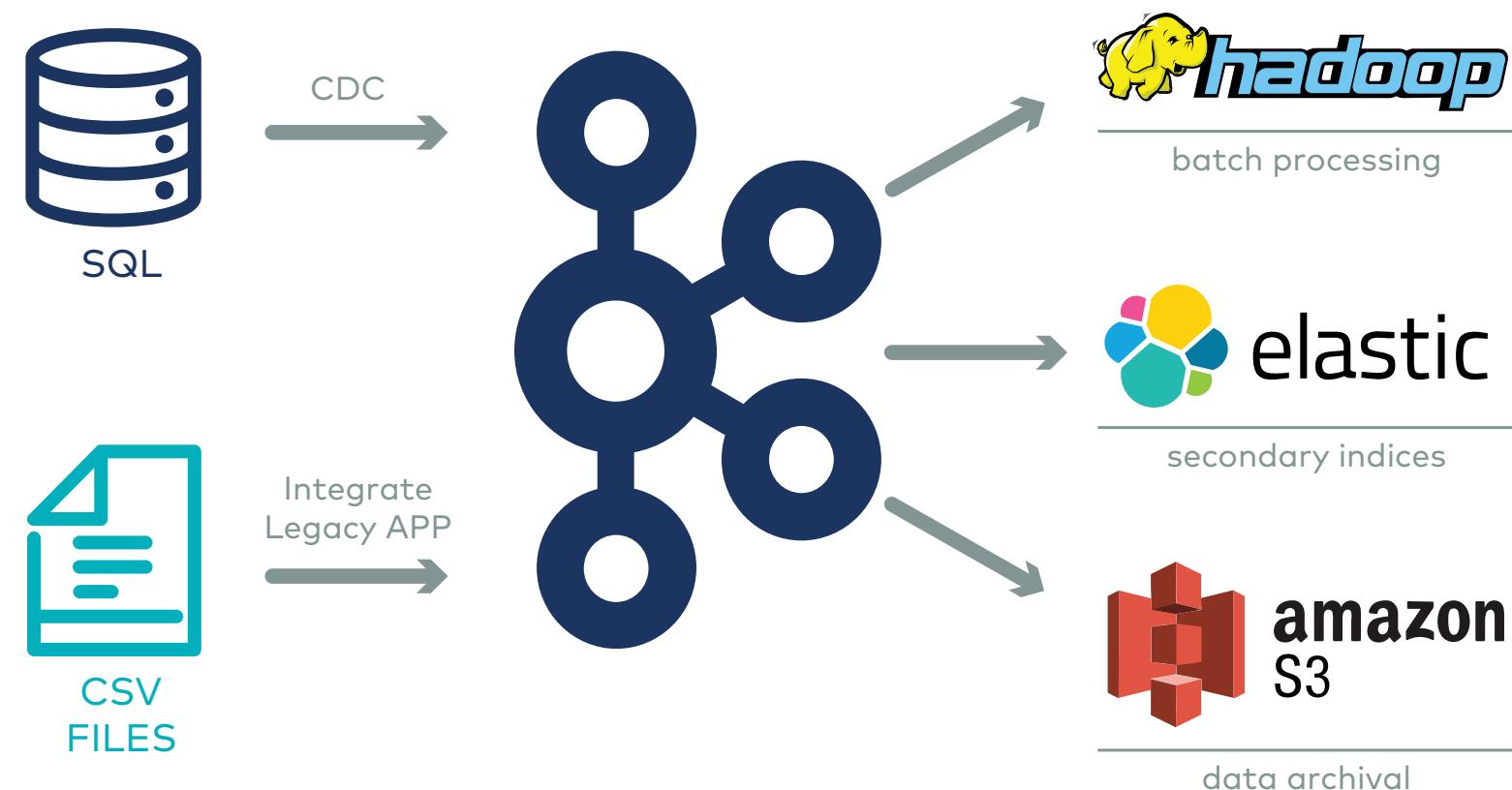
You could program custom producers or consumers with hooks into the external systems to make this happen... But... there's a better and easier way...

Kafka Connect to the Rescue!

- Kafka Connect does the work for us!
- All copying behavior is in Kafka Connect.
- Plugins called connectors contain the logic specific to particular external systems.



Use Cases



Source and Sinks

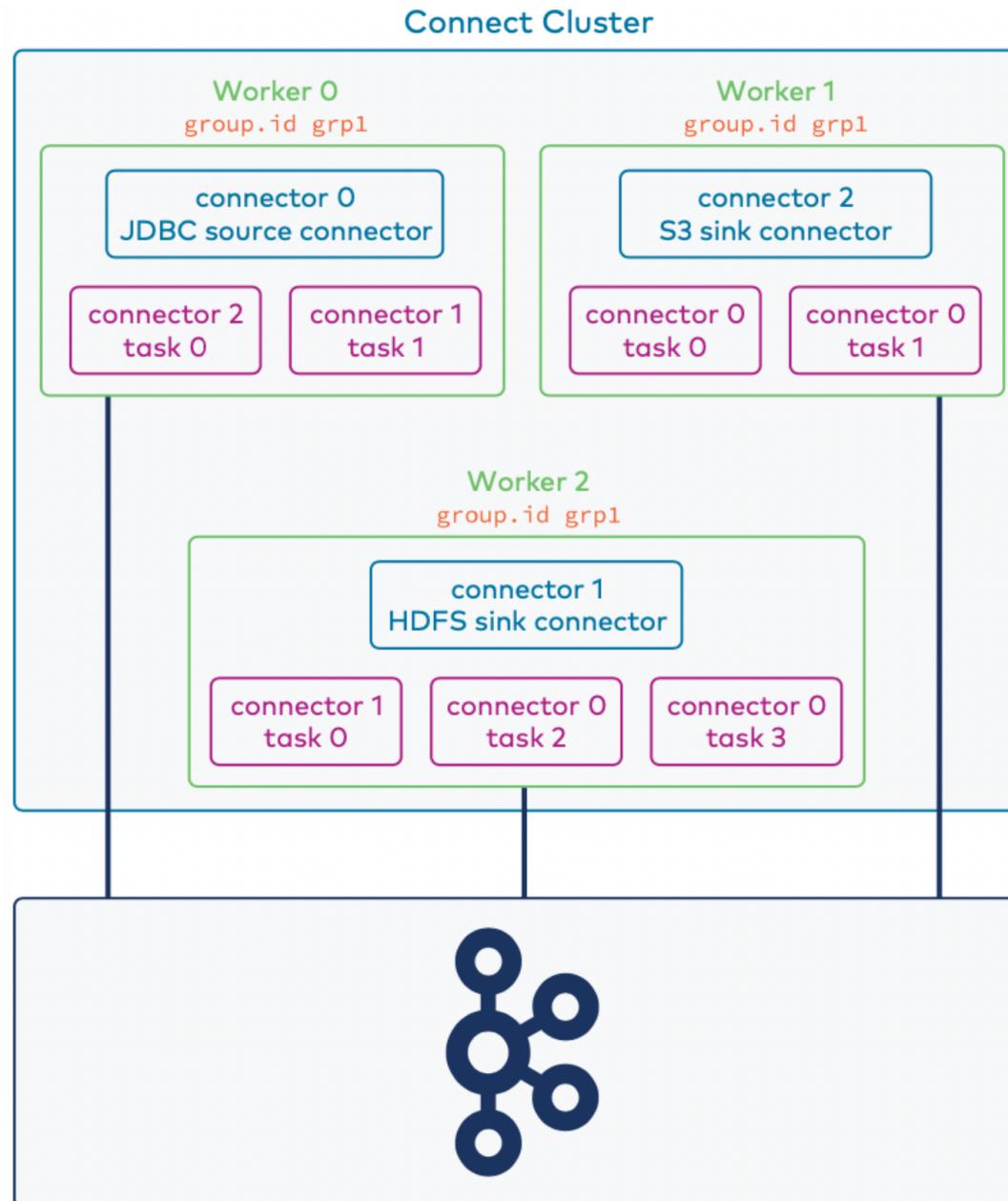
Two kinds of connectors:



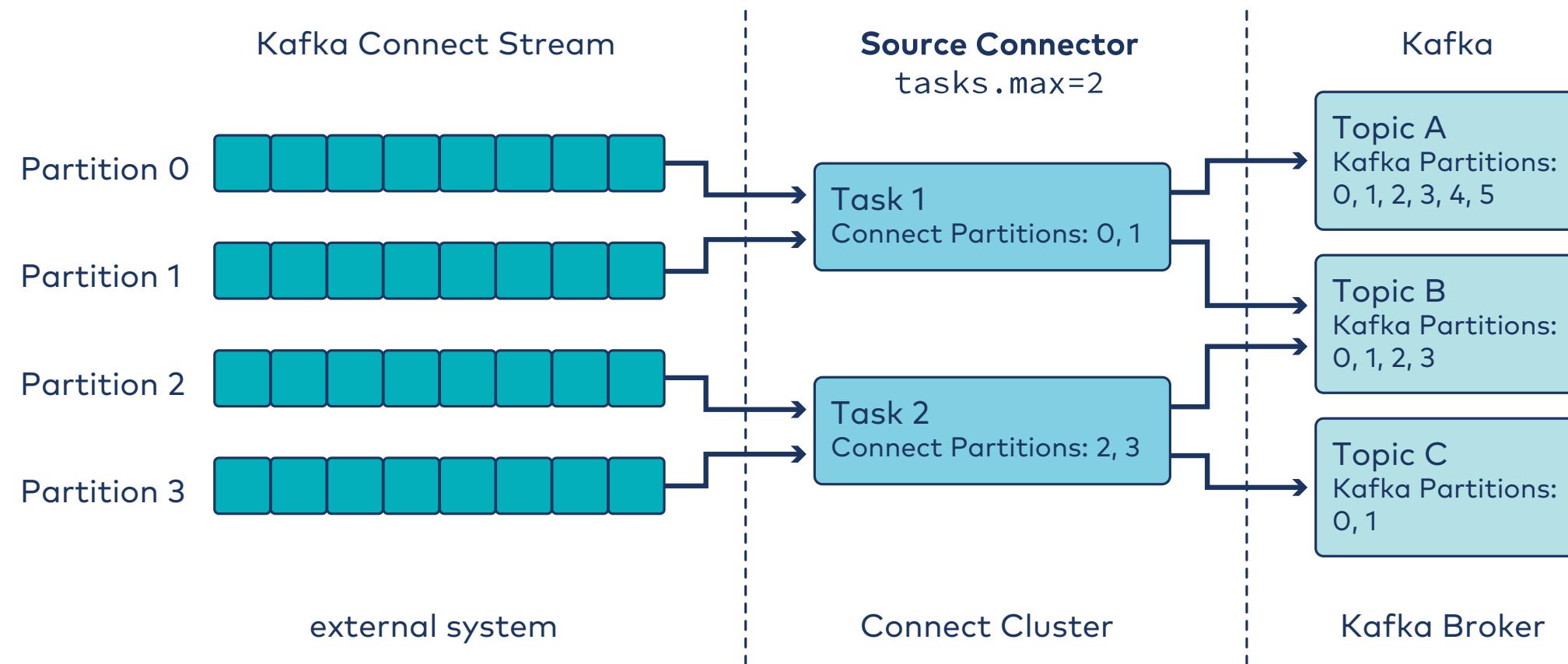
Players in the Kafka Connect World

Player	Description
Kafka Connect	Logic for copying regardless of the external system. A connector has one or more tasks
Worker	Process that runs connectors and/or tasks. A worker runs zero or more connectors. A worker runs zero or more connectors, zero or more tasks
Connector	Logic for copying specific to/from a given external system.
Task	Unit of parallelism into which connector copying logic is broken up.

Connect Cluster Example



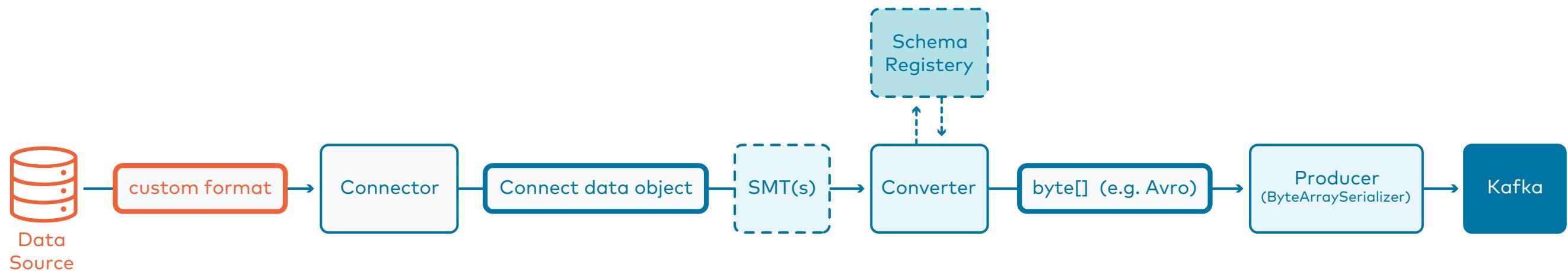
Providing Parallelism & Scalability



Converters

- Converters provide the data format written to or read from Kafka (like Serializers).
- Converters are decoupled from Connectors so they can be reused.
 - Allows any connector to work with any serialization format.

Example of format conversion in a source converter (sink converter is the reverse):



06b: Using Connectors in Confluent Cloud

Description

Build a connect cluster in Confluent Cloud with connectors.

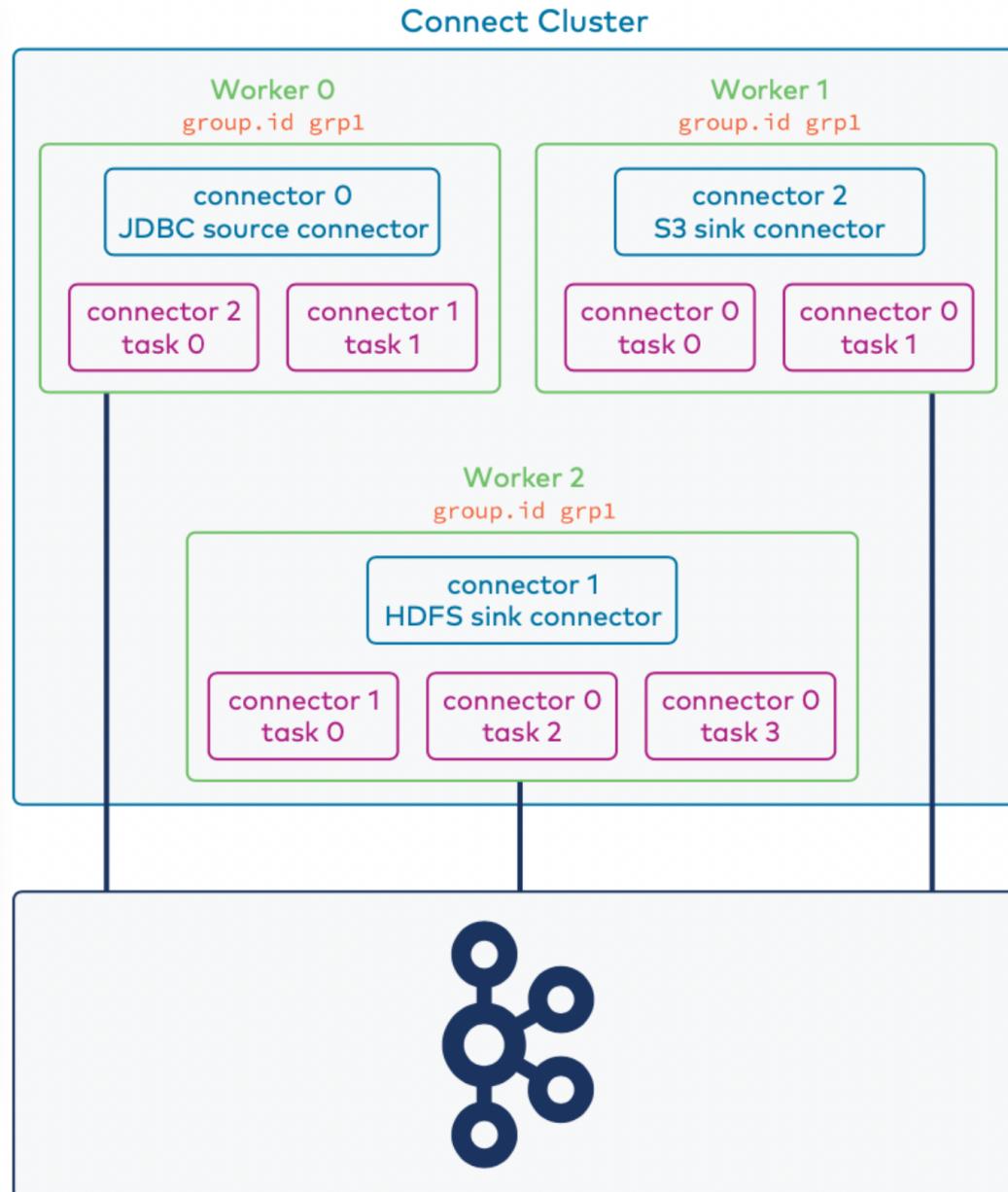
Learning Objectives



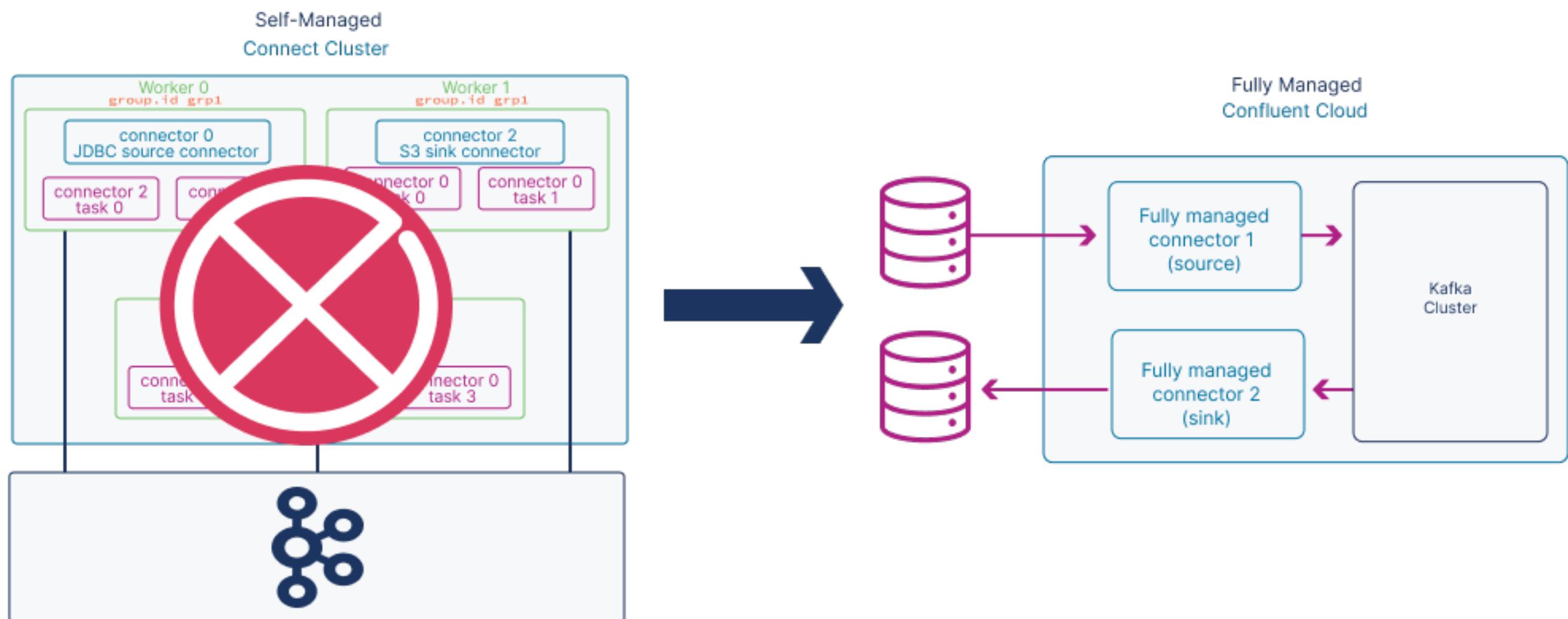
Upon completion of this lesson and associated lab exercises, you will be able to:

- Describe a connect cluster, including:
 - Fully managed connectors
 - Data previews
 - Self-managed connectors
 - Dead Letter Queue (DLQ)
 - Single Message Transforms (SMTs)

Connect Cluster



Fully Managed Connectors in Confluent Cloud



List of Managed Connectors

- Each connector has its own configuration properties.
- There are many fully managed connectors.
- New connectors are added periodically to Confluent Cloud.



Create Managed Connectors

You can create fully managed connectors using:

- Confluent Cloud UI (Demo)
- Confluent CLI:

```
confluent connect cluster create --config-file config.json --cluster <cluster ID>
```

- Confluent REST API:

```
curl --request POST  
  'https://api.confluent.cloud/connect/v1/environments/<env-ID>/clusters/<cluster-ID>/connectors'  
--header 'authorization: Basic <base64-encoded-key-and-secret>'  
--header 'Content-Type: application/json'  
--data "@<my-connector-config>.json"
```

Data Previews

You can sample the output from a connector before you launch it

- Only available using the Confluent UI and for certain connectors
- Previewed records are stored in a separate data preview output topic named `data-preview-ldpc-<id>`
- You must set `tasks.max=1` to run data preview
- Source systems need to produce data



Data preview requests read actual data from the source system.

The screenshot shows the Confluent UI interface. At the top, there's a dark header with the Confluent logo and navigation links: HOME > ENVIRONMENTS > DEFAULT > DEMOS. Below this is a sidebar with links: Cluster Overview, Dashboard, Networking, API Keys, Cluster Settings, Stream Lineage, Stream Designer, Topics, ksqlDB, Connectors (which is highlighted), Clients, and Schema Registry. To the right of the sidebar, the main content area has a title "Connectors". Below it, a sub-header says "Connectors Data preview requests (1)". There's a search bar labeled "Search data preview requests". A single data preview request is listed: "data-preview-ldpc-oqyvpp-Da..." with a checkmark next to "Preview available". Below the request, there's a table with columns "Category" (DataGenSource), "Source" (DatagenSource), "Plugin name", and a delete icon. The entire screenshot is framed by a light gray border.

View the Data Preview Output

The expanded preview record has two sections:

- Metadata
 - Information about how the record is produced
 - Only shown in data preview output
- Record
 - Actual data that is sent to the topic when the connector is running

If data is not as expected:

- Click **Edit and Launch** to modify the connector properties.

```
1 "metadata": {  
2     "source_partition": {  
3         "task.id": "0"  
4     },  
5     "source_offset": {  
6         "task.generation": "1",  
7         "random.seed": -5435027388492592838,  
8         "current.iteration": "60"  
9     },  
10    "topic": "orders-topic",  
11    "partition": null,  
12    "offset": null,  
13    "connector": "ldpc-oqyvpp",  
14    "correlation_id": "6a07f379-39a5-437f-9d59-887ab1f27e90-ldpc-oqyvpp-0-59",  
15    "current_step": 1,  
16    "total_step": 1,  
17    "type": "SOURCE",  
18    "transformation_name": null,  
19    "transformation_type": null  
20 },  
21 }  
22 "record": {  
23     "key": 59,  
24     "value": {  
25         "ordertime": 1492154198176,  
26         "orderid": 59,  
27         "itemid": "Item_1",  
28         "orderunits": 7.827637496301478,  
29         "address": {  
30             "city": "City_5",  
31             "state": "State_",  
32             "zipcode": 13398  
33         }  
34     },  
35     "headers": {  
36         "task.generation": "0",  
37         "task.id": "0",  
38         "current.iteration": "59"  
39     }  
40 }  
41 }
```

Edit and Launch

Connect Events in Confluent Cloud (Connector Logs)

- Currently logged events:
 - `io.confluent.logevents.connect.TaskFailed`
 - `io.confluent.logevents.connect.ConnectorFailed`
- Only available for Standard and Dedicated clusters
- Restricted to the OrganizationAdmin RBAC role

There are two main ways to view Connector events:

- Confluent Cloud UI (easiest)
- Consume the Connector events from the `confluent-connect-log-events` topic using:
 - Confluent CLI
 - Kafka Consumers

View Connect Events Using Confluent Cloud UI

The image shows two screenshots of the Confluent Cloud UI. On the left, the 'Connectors' page is displayed, listing one connector: 'DatagenSourceConnector_0' (Running). It shows metrics like 528B/s and details like Category: Source, ID: lcc-j30yvw, and Plugin name: Datagen Source. A large arrow points from this screen to the right screen. The right screen shows the 'Logs' tab for the same connector. The log entry at 3:13:03 PM details a non-tolerated exception in the error handler, with a detailed JSON log message.

Cluster Overview
Dashboard
Networking
API Keys
Cluster Settings
Stream Lineage
Stream Designer
Topics
ksqlDB
Connectors
Clients
Schema Registry

Connectors

Connectors (1) Data preview requests (1)

Search connectors

DatagenSourceConnector_0
Running

Tasks	Bytes/sec
1	528B/s

Messages/sec Messages behind

2	--
---	----

Overview

Category	Source
ID	lcc-j30yvw
Plugin name	Datagen Source

Overview

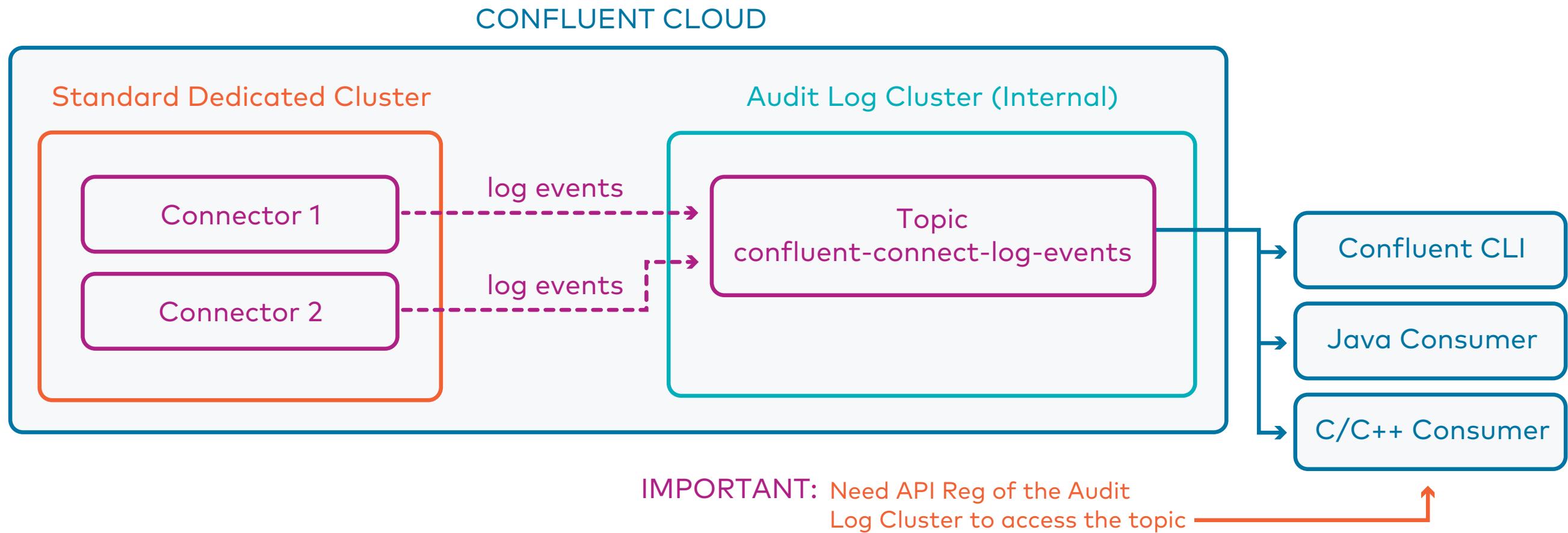
Logs

Filter by keyword

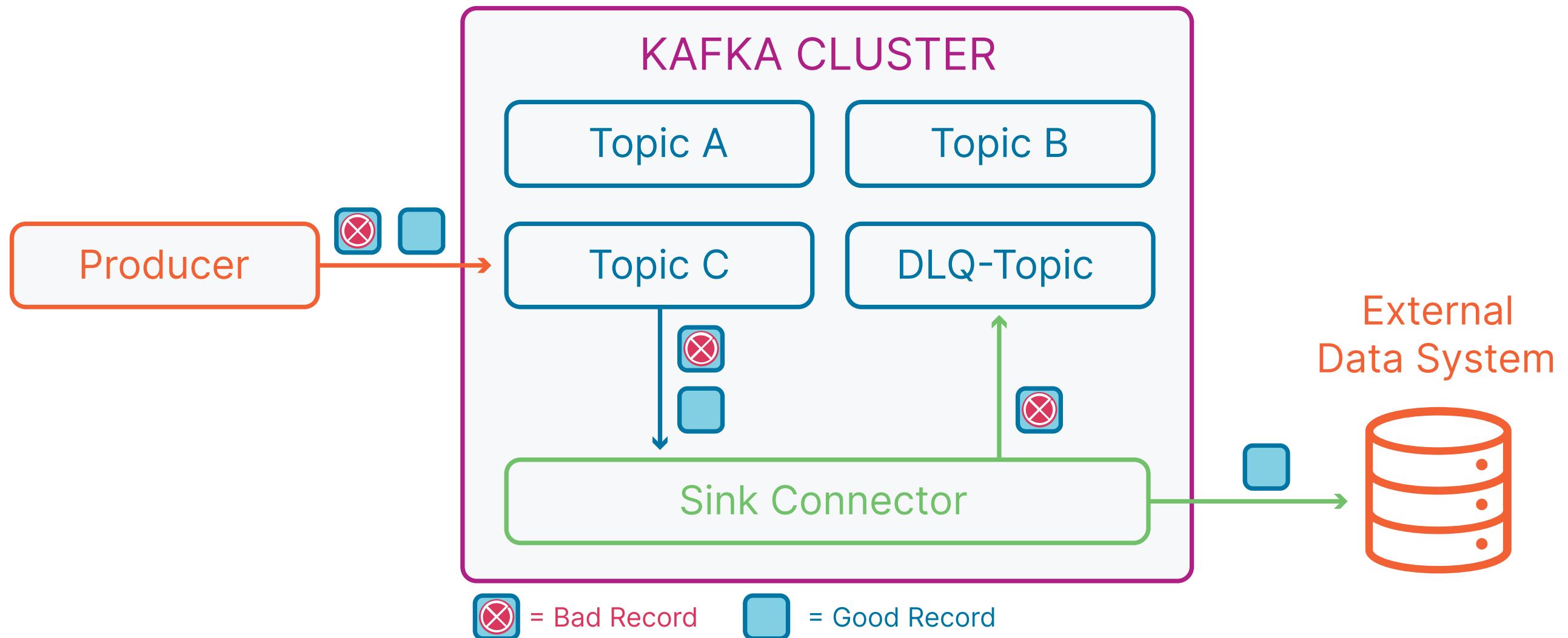
3:13:03 PM Non tolerated exception in error handler

```
1 {  
2   "datacontenttype": "application/json",  
3   "data": {  
4     "level": "ERROR",  
5     "context": {  
6       "connectorId": "lcc-j30yvw"  
7     },  
8     "summary": {  
9       "connectorErrorSummary": {  
10         "message": "Non tolerated exception in error handler",  
11         "rootCause": "Failed to access Protobuf data from topic orders-topic : Schema being registered is incompatible with an earlier schema for subject \"orders-topic-value\", details: [Incompatible because of different schema type]; error code: 409; error code: 409"  
12       }  
13     }  
14   },  
15   "subject": "lcc-j30yvw-lcc-j30yvw-0",  
16   "specversion": "1.0",  
17   "id": "9604410e-b32f-4f87-aaf0-7e089aed1b69",  
18   "source": "crn://confluent.cloud/connector=lcc-j30yvw",  
19   "time": "2023-02-09T14:13:03.365Z",  
20   "type": "io.confluent.logevents.connect.TaskFailed"  
21 }
```

Consuming Connect Events



Dead Letter Queue (DLQ)



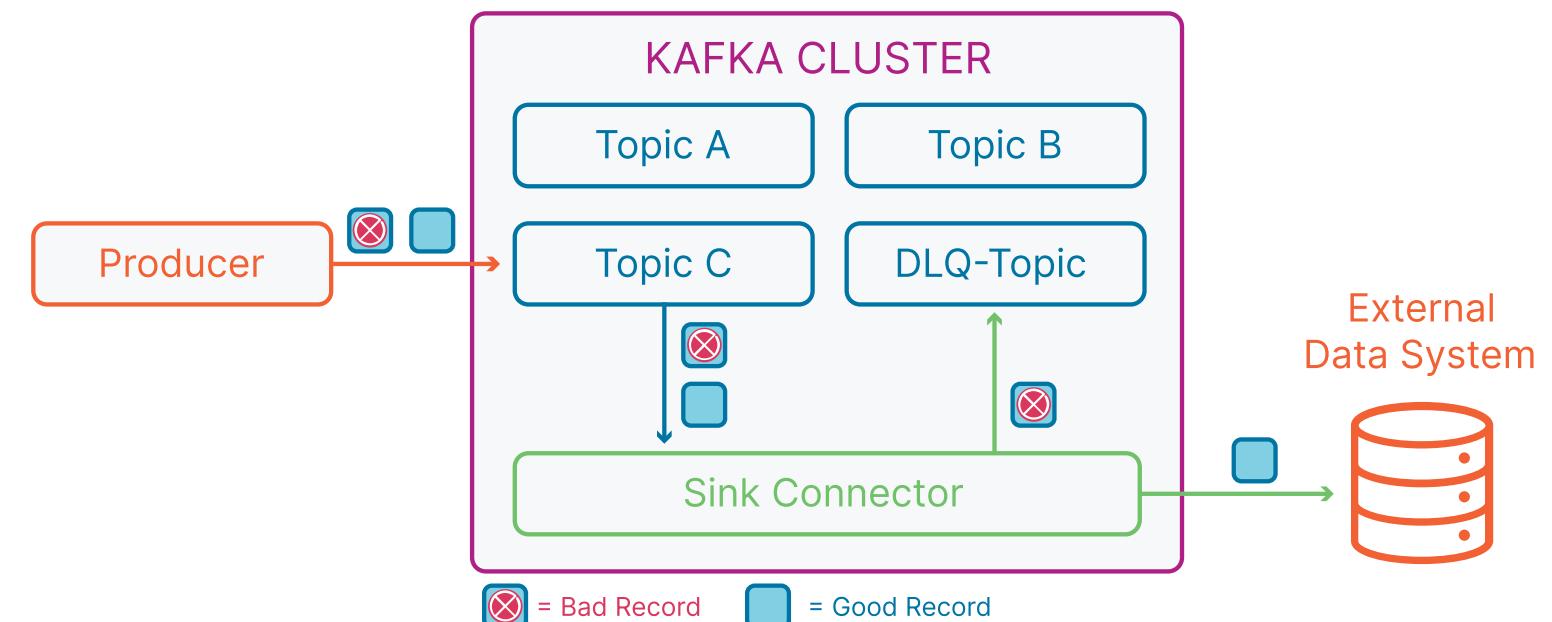
Auto-Created DLQ Topic

When you launch a sink connector, the DLQ topic is automatically created:

- The DLQ topic is a standard topic
- The topic is named `dlq-<connector-ID>`

DLQ record header contains useful information to identify the cause of the error:

- Topic name
- Partition number
- Offset
- Error Class name
- Exception Class name
- Stack Trace



06c: Security Considerations

Description

Security considerations related to Connect clusters.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Review managed connectors that require credentials.

Security - Managed Connectors to Access Kafka

Managed connectors require credentials to access Kafka:

- Authentication:
 - API Key / Secret
- Authorization:
 - ACLs
 - RBAC



Best Practices: Use a Service Account configured with the ACLs and API Key / Secret.

Security - Accounts to Manage Connectors

RBAC roles provide different levels of access for accounts to managed connectors:

Role	Access
• OrganizationAdmin	• Create, configure, pause, restart, delete a connector • Read status and metrics, manage credentials and access • Create and view data preview
• EnvironmentAdmin	
• CloudClusterAdmin	
• Operator	• Pause and restart a connector • Read status and metrics
• DeveloperRead	• Read configuration, status and metrics
• DeveloperWrite	• Configure the connector (DeveloperWrite)

06d: Networking with External Systems

Description

A review of Confluent Cloud options for networking with external systems.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Describe cluster network types.
- Define static egress IP addresses.

Cluster Network Type, IP Address, and IP Range by Connector

Cluster Network Type	IP Address of External Data System	IP Range Used By Connector
• Public Endpoint	Public	Fixed set of static egress IP addresses
• VPC Peering • Transit Gateway	Public	Dynamic public IP/CIDR range from cloud provider region where Confluent Cloud cluster is located
• VPC Peering • Transit Gateway	Private	Source IP address from /16 CIDR range configured by customer for the Confluent Cloud Cluster

Static Egress IP Addresses

- Available for Confluent Cloud clusters on AWS and GCP.
- Public internet only.
- Go to the data source or sink provider platform add the static egress IP addresses to the allow list of authorized clients.

The screenshot shows the Confluent Cloud interface for a cluster named 'DEMONS'. The left sidebar has 'Networking' selected under 'Cluster Overview'. The main content area is titled 'Cluster networking' and shows the 'Type' as 'Internet'. It states 'Cluster is available over the public internet'. Below this is a section titled 'Egress IPs' which lists numerous IP ranges used by the cluster to establish outbound connections. The list includes:

- 100.24.204.241/32
- 107.21.116.39/32
- 18.204.235.237/32
- 18.232.30.126/32
- 3.217.171.197/32
- 3.227.73.0/32
- 3.234.42.230/32
- 34.204.253.120/32
- 34.205.233.46/32
- 34.231.151.179/32
- 35.173.154.142/32
- 52.20177.94/32
- 52.3.108.122/32
- 52.4.168.0/32
- 52.45.226.34/32
- 52.5.120.217/32
- 52.65.155.248/32
- 52.73.34.234/32
- 54.146.88.203/32
- 54.156.105.40/32
- 54.172.40.193/32
- 54.204.102.217/32
- 54.204.113.211/32
- 54.237.164.14/32
- 54.242.207.111/32
- 54.243.244.175/32
- 54.83.252.163/32

06e: Self-Managed Connectors

Description

Self-managed connectors in Confluent Cloud maybe used when a managed-connector does not yet exist.

Learning Objectives

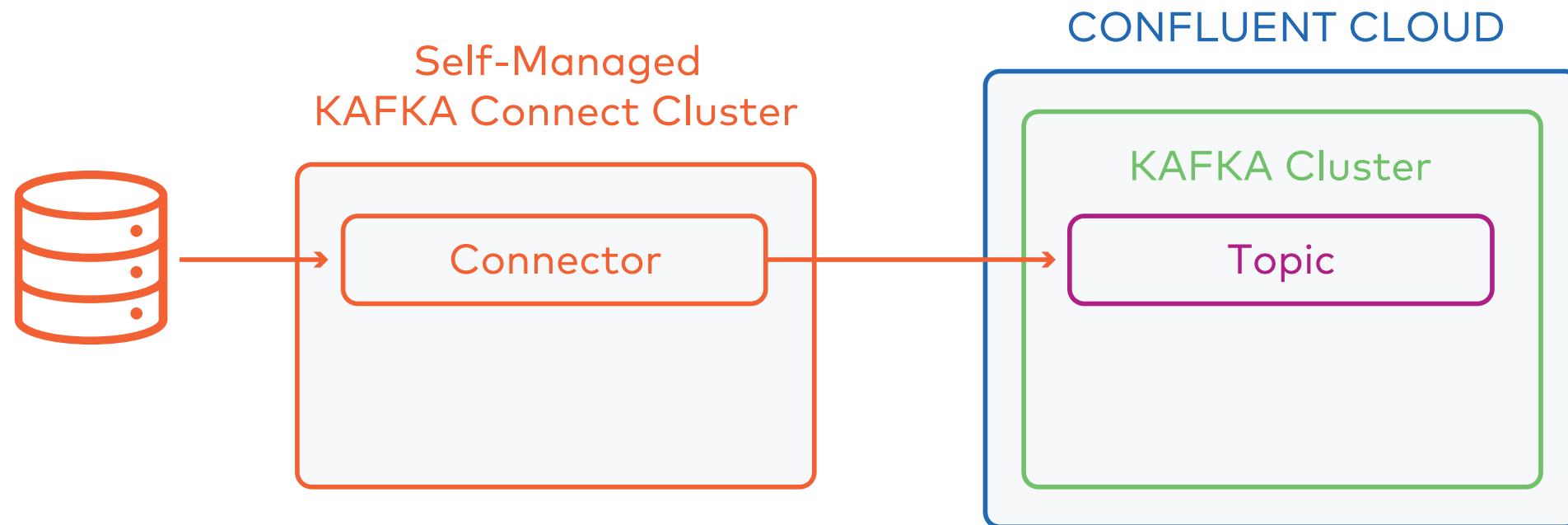


Upon completion of this lesson and associated lab exercises, you will be able to:

- Define self-managed connectors in Confluent Cloud, including where to find them.

Run Connectors Not Yet Available in Confluent Cloud

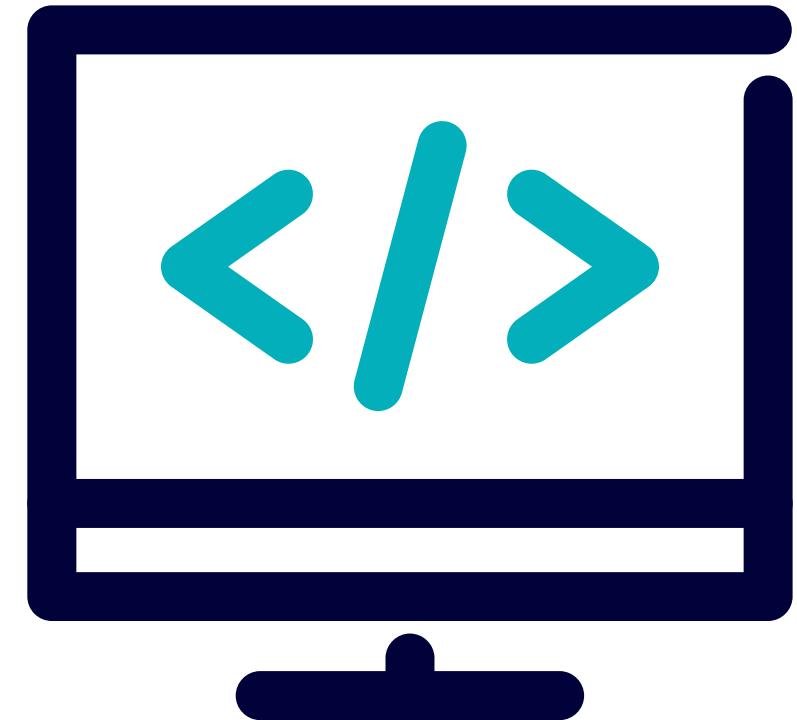
- What if you want to run a connector not yet available in Confluent Cloud?
 - You may run it yourself in a self-managed Kafka Connect cluster.



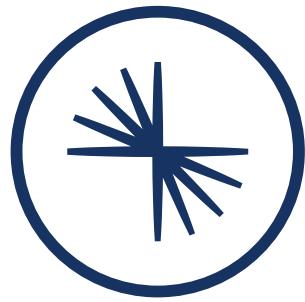
Lab Module 06: Connect in Confluent Cloud

Please work on **Lab Module 06: Connect in Confluent Cloud**.

Refer to the Exercise Guide.



07: ksqlDB in Confluent Cloud



CONFLUENT
Global Education

Module Overview



This module contains five lessons:

- Stream Concepts
- What is ksqlDB
- Creating ksql Applications
- ksqlDB in Confluent Cloud
- Security Considerations

After this module you will be able to:

- Define the Stream - Table duality in stream processing
- Explain the motivation for ksqlDB
- Write push and pull queries and explain the differences
- Write other important ksqlDB queries

07a: Stream Concepts

Description

A review of fundamental stream concepts.

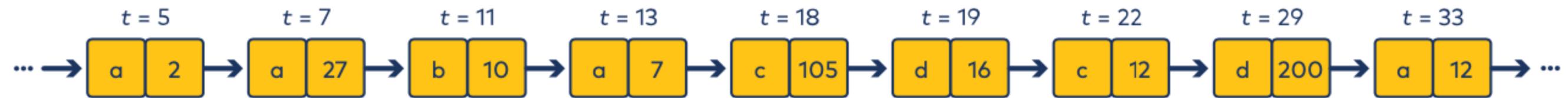
Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Define these stream concepts:
 - A sample stream
 - Operating on the stream
 - Windowing the stream
 - Stream-table duality

A Sample Stream



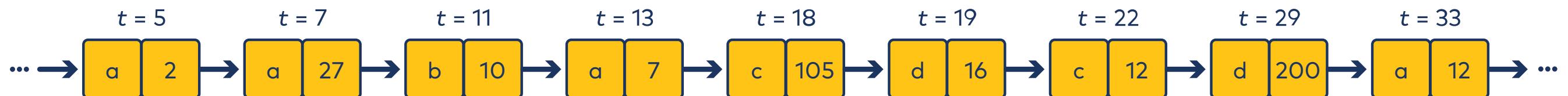
- Events are key/value pairs
- Ours will be sourced from Kafka topics Time is a big deal

Operating on the Stream

We can do **stateless operations** on a stream, like filtering.

What would happen if we filtered to keep those records whose value exceeded 50?

Input stream:



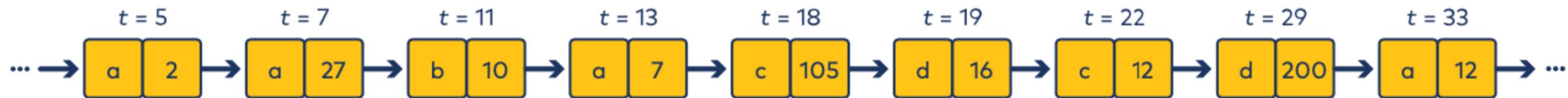
Output stream:



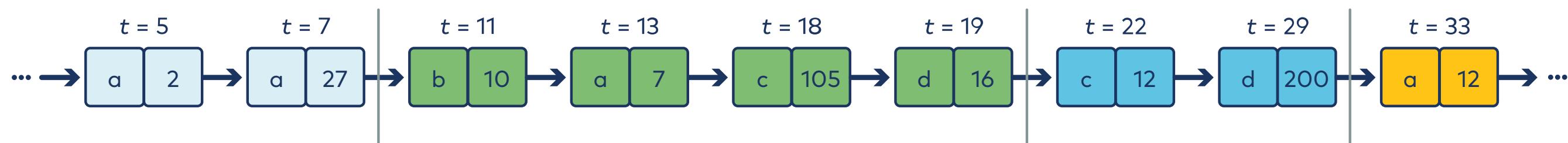
Windowing the Stream

We can split up time into windows.

- Here's our input stream:



Here's the same stream, divided into windows of size 10:

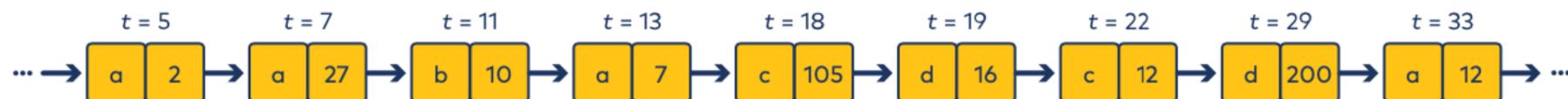


Stream-Table Duality

- We can view a stream as a table

Stream

Records are events in time

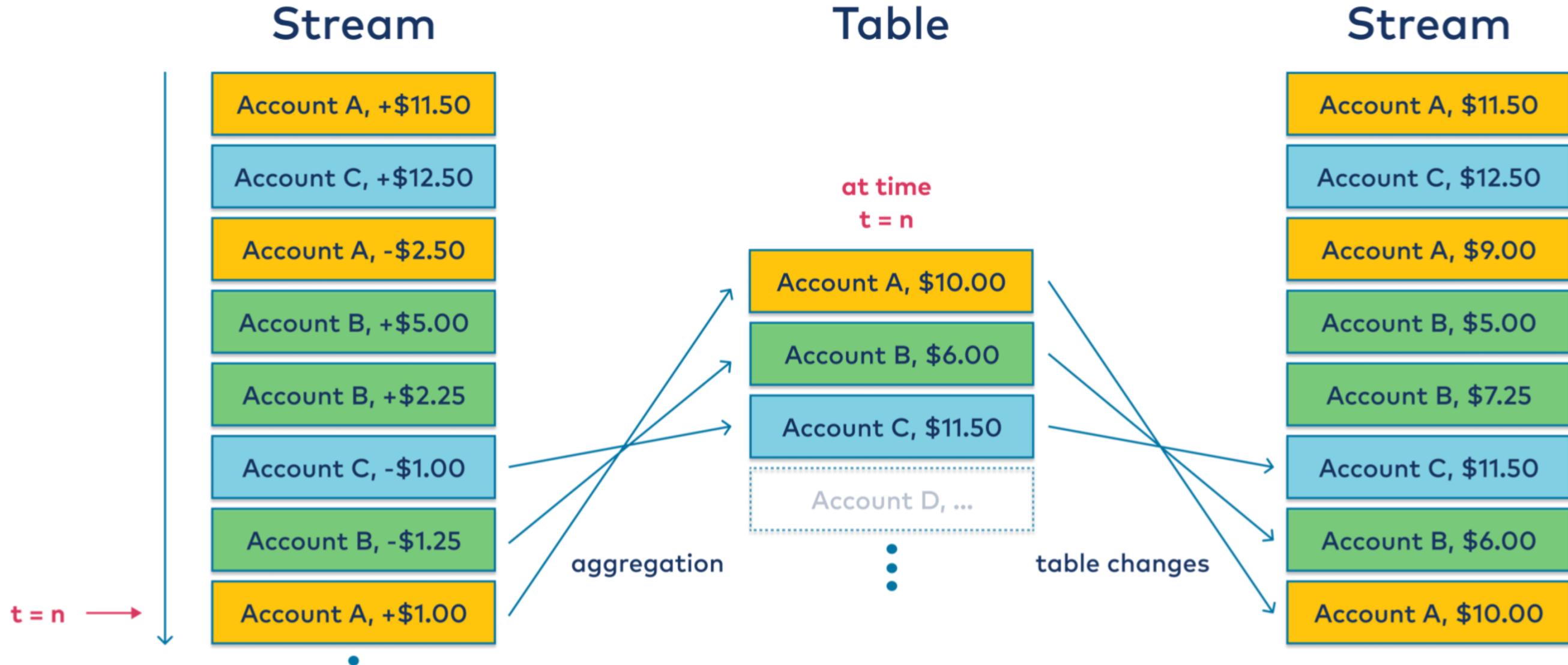


Table

Records are updates to same-key table entries



Stream-Table Duality - Bank Account Example



07b: What is ksqlDB?

Description

A review of ksqlDB, including stream processing apps, simple architecture, and components.

Learning Objectives



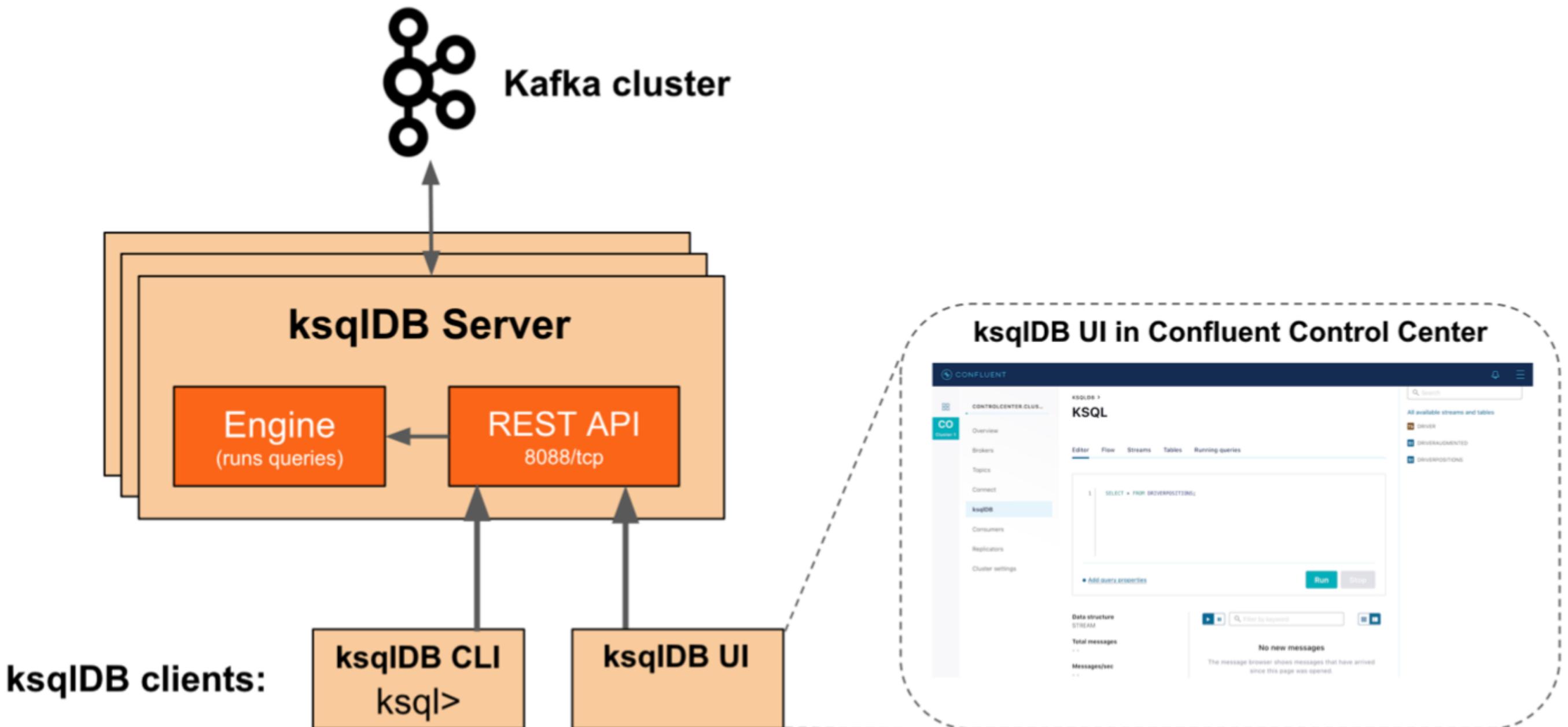
Upon completion of this lesson and associated lab exercises, you will be able to:

- Define what is ksqlDB, including:
 - A simpler architecture for event streaming apps
 - ksqlDB architecture and components

A Simpler Architecture for Event Streaming Apps



ksqldb Architecture and Components



07c: Creating ksql Applications

Description

Describe writing ksql queries and applications in Confluent Cloud.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Describe creating ksql apps, including:
 - SQL-like semantics
 - ksql statements
 - Non-persistent and persistent queries
 - Pull vs. push queries
 - ksqlDB functions and clauses
 - Windowing modes in ksqlDB

SQL-Like Semantics

```
CREATE TABLE possible_fraud AS
    SELECT card_number, count(*)
    FROM authorization_attempts
    WINDOW HOPPING (SIZE 5 SECONDS, ADVANCE BY 1 SECOND)
    GROUP BY card_number
    HAVING count(*) > 3;
```

ksql Statements

Command	Description
CREATE	Create streams, tables, Kafka Connect connectors, custom "types"
SELECT	Create push or pull queries
SHOW	Display information about streams, tables, types, properties, functions, topics queries, etc.
DESCRIBE	Display column names and types, metrics for streams and tables, or check status of connector
DROP	Remove an existing stream, table, connector, or type

Non-Persistent vs. Persistent Queries

Non-Persistent Query

- The result of a non-persistent query will not be persisted into a Kafka topic and will only be printed out in the console.

Persistent Query

- The result of a persistent query will be persisted into a Kafka topic.

Pull vs. Push Queries

Pull Queries (point-in-time queries)

- Look up value in materialized table
- Single response with the result set

```
SELECT * FROM viewsPerUserSession
  WHERE user_id = 'cool-kafka-user-5'
    AND 1570051876000 <= WINDOWSTART
    AND WINDOWSTART <= 1570123744000;
```

Push Queries (continuous queries)

- Subscribe to a stream
- Response of indefinite length Denoted by EMIT CHANGES

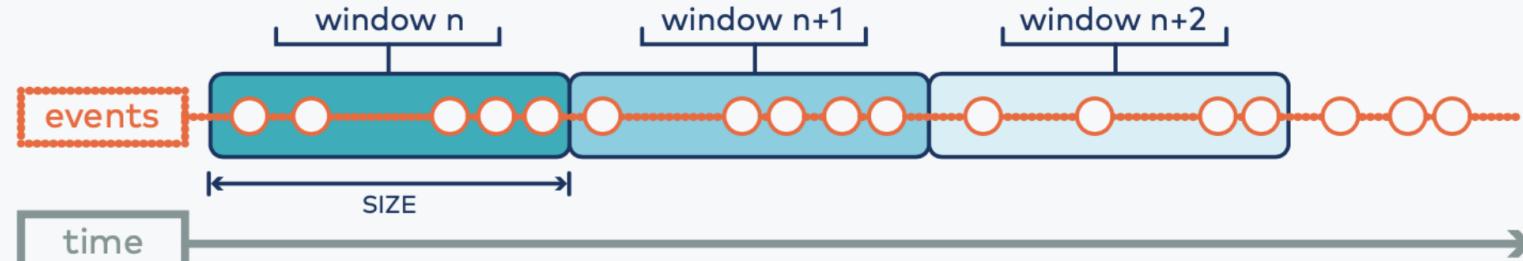
```
SELECT * FROM user_table
  EMIT CHANGES;
```

ksqldb Functions and Clauses

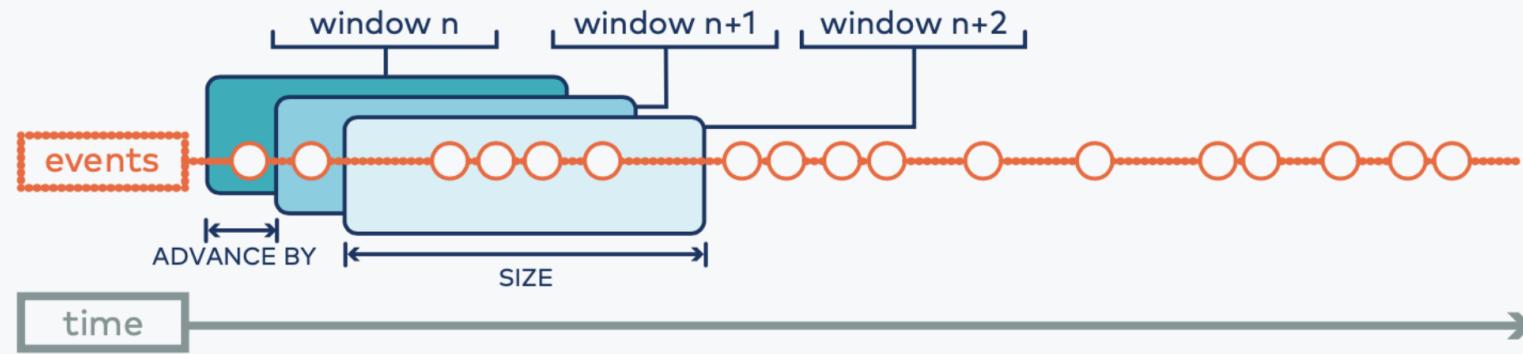
- Many popular scalar functions:
 - `ABS`, `CEIL`, `CONCAT`, `LEN`, `ROUND`, `TRIM`, `SUBSTRING`, and many more
 - `EXTRACTJSONFIELD`: Given a string column in JSON format, extract the field that matches
- You can `JOIN` streams and tables
- Aggregations `COUNT`, `MAX`, `MIN`, `SUM`, etc
- Re-key the streams
 - `PARTITION BY column_name` → resulting stream will have new key set by specified column
 - `PARTITION BY MY_FUNCTION(x, 2)` → resulting stream will have new key set by a UDF response

Windowing Modes in ksqlDB

Tumbling



Hopping



Session



Tumbling Windows

- In ksqlDB SQL, to do windowing, we simply add a WINDOW clause to a query.
- We specify the type of window in the clause, e.g.

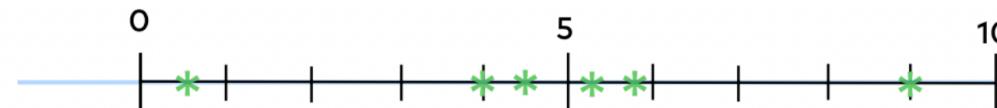
```
CREATE TABLE top_sellers AS
    SELECT song, artist, TOPK(count(*), 10)
    FROM song_sales_stream
    WINDOW TUMBLING (SIZE 604800 SECONDS)
    GROUP BY song, artist;
```

Fraud Detection Example 1

Query:

```
CREATE TABLE possible_fraud AS
    SELECT card_number, count(*)
    FROM authorization_attempts
    WINDOW TUMBLING (SIZE 5 SECONDS)
    GROUP BY card_number
    HAVING count(*) > 3
    EMIT CHANGES;
```

Example timeline



*(Each * represents an authorization attempt.)*

Question: Will the query report possible fraud?

Fraud Detection Example 2

Query:

```
CREATE TABLE possible_fraud AS
    SELECT card_number, count(*)
    FROM authorization_attempts
    WINDOW HOPPING (SIZE 5 SECONDS,
                     ADVANCE BY 1 SECOND)
    GROUP BY card_number
    HAVING count(*) > 3
    EMIT CHANGES;
```

Example timeline



(Each * represents an authorization attempt.)

Question: Will the query report possible fraud?

Data Enrichment and Joins

Supported JOIN Types:

Join operands	Type	(INNER) JOIN	LEFT JOIN	OUTER JOIN
Stream + Stream → Stream	windowed	✓	✓	✓
Table + Table → Table	non windowed	✓	✓	✓
Stream + Table → Stream	non windowed	✓	✓	✗

Stream-Table Join

Now let's bring a stream and a table together:

```
CREATE STREAM driver_positions_enriched AS
    SELECT pos.driver_id,
        pos.latitude,
        pos.longitude,
        prof.name,
        prof.postal_code,
        ...
    FROM    driver_positions pos LEFT JOIN driver_profiles prof
    ON      pos.driver_id = prof.driver_id
    EMIT CHANGES;
```

Stream-Stream Join

We can also join a stream with a stream, e.g.

```
CREATE STREAM shipped_orders AS
    SELECT o.id AS order_id,
        TIMESTAMPTOSTRING(o.rowtime, 'yyyy-MM-dd HH:mm:ss', 'UTC') AS order_ts,
        o.total_amount,
        o.customer_name,
        s.id AS SHIPMENT_ID,
        s.warehouse
        ...
    FROM     orders o INNER JOIN shipments s
    WITHIN  7 DAYS
    ON      o.id = s.order_id
    EMIT CHANGES;
```

07d: ksqlDB in Confluent Cloud

Description

A review of ksqlDB in Confluent Cloud, including supported features, limitations, ksqlDB capacity, ksqlDB high availability, and scaling CSUs.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Describe ksqlDB in Confluent Cloud, including:
 - Supported features
 - Limitations
 - ksqlDB capacity in Confluent Cloud
 - ksqlDB high availability
 - Scaling CSUs

Supported Features

- Web interface for managing your ksqlDB clusters
- SQL editor to write, develop, and execute SQL queries with auto completion
- Integration with Confluent Cloud Schema Registry
- SQL-based Connect integration
- Available in all cloud providers, in all regions
- Private networking with Private Link

Limitations

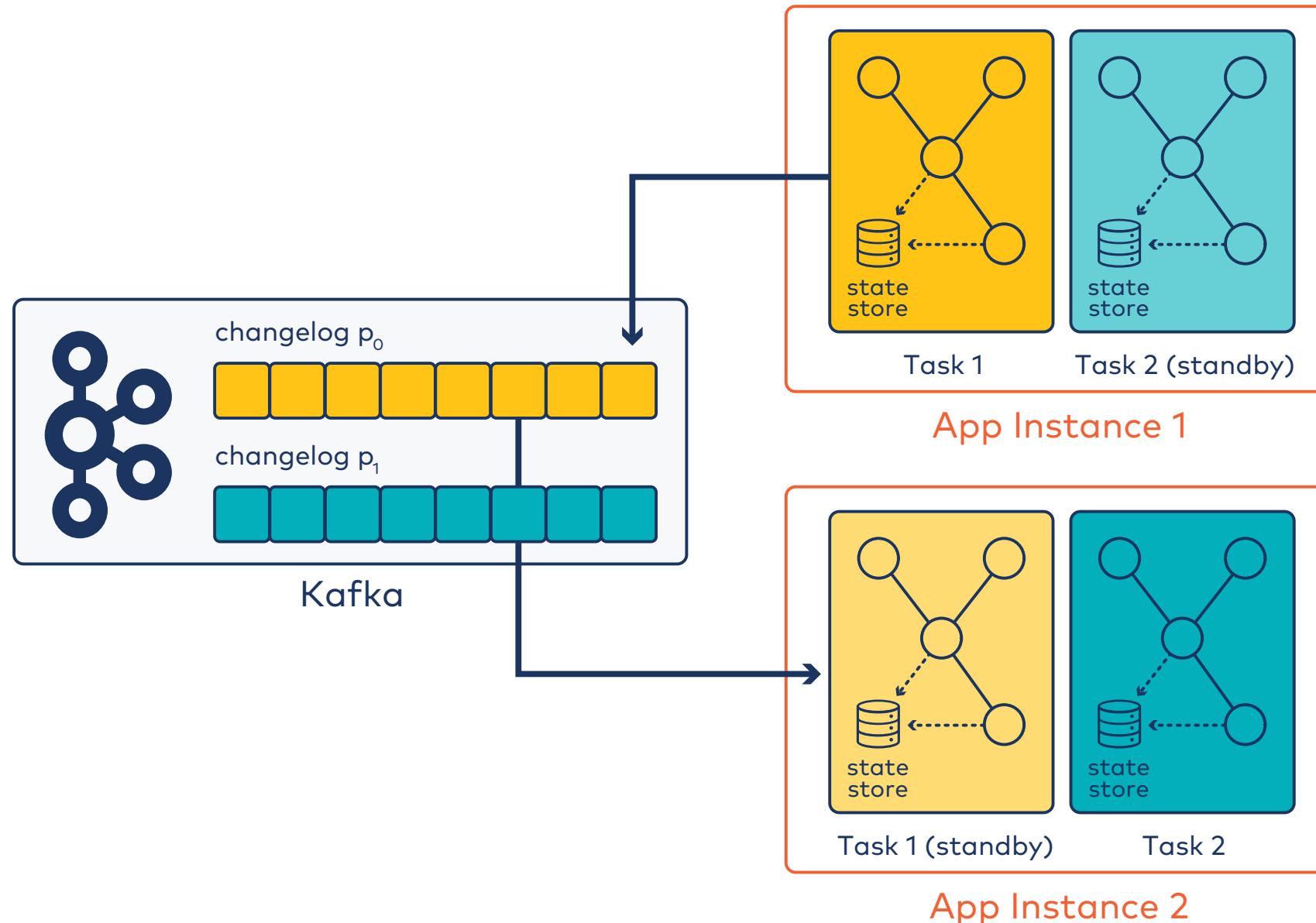
- User-defined functions (UDFs, UDAFs, and UDTFs) aren't supported
- Maximum of 20 persistent queries per cluster
- Maximum of 10 ksqlDB clusters per environment
- Pull queries have specific limitations in Confluent Cloud

ksqldb Capacity in Confluent Cloud

Capacity unit in Confluent Cloud ksqldb is the Confluent Streaming Unit (CSU):

- Your cluster can have 1, 2, 4, 8, 12, 16, 20, 24 and 28 CSUs
- **High availability** cannot be enabled for clusters with less than 8 CSUs

ksqldb High Availability



Storage

125 GB of storage space with each CSU

Number of CSUs	Storage	User-available storage
1	125 GB	125 GB
2	250 GB	250 GB
4	500 GB	500 GB
8	1000 GB	500 GB*
12 - 28	1500 - 3500 GB	750 - 1750 GB*

- High-availability clusters with half of storage used for data replicas.

Scaling CSUs

Scaling ksqlDB cluster after initial provisioning:

- Provision a new cluster and migrate to your new one
- Contact your Confluent team to scale the cluster (for 4+ CSUs clusters)

07e: Security Considerations

Description

An overview of important security considerations in Confluent Cloud.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

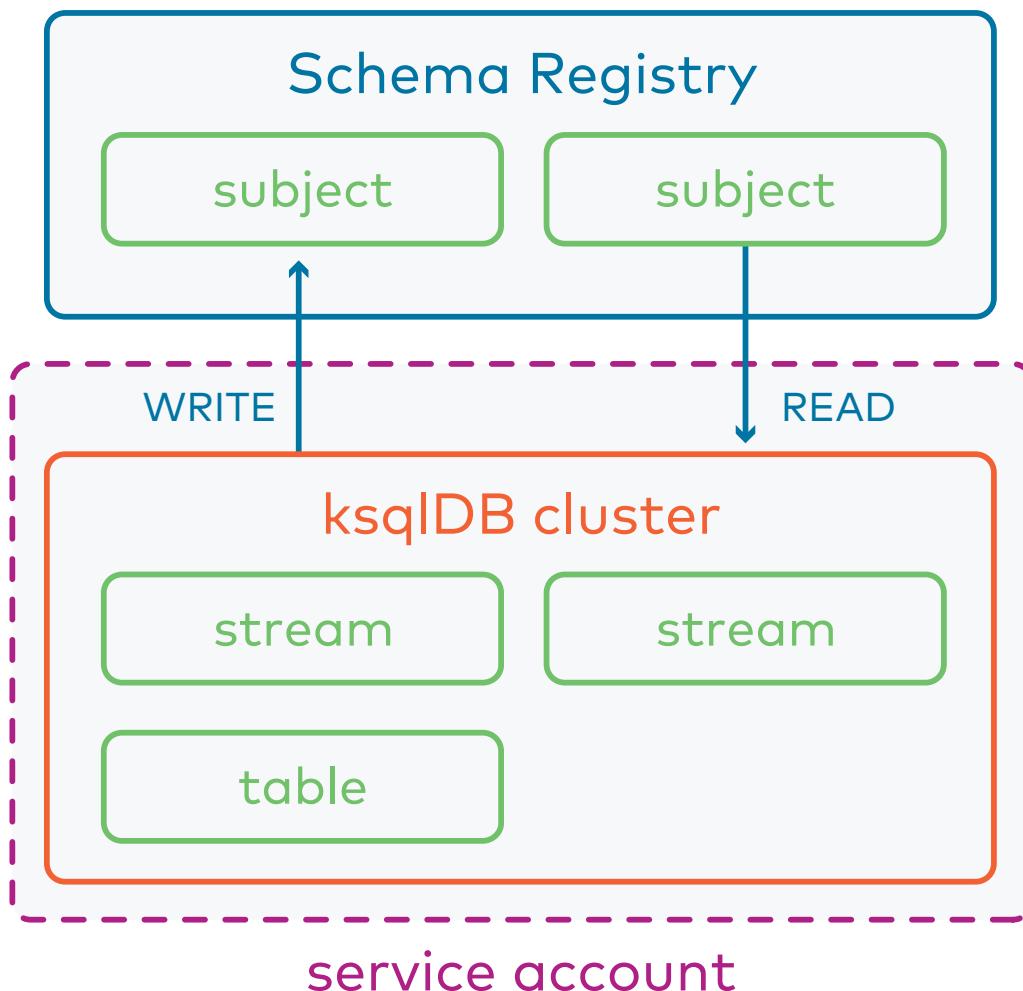
Evaluate security considerations, including:

- RBAC for ksqlDB Clusters
- Enable ksqlDB access to Schema Registry

Enable ksqlDB Access to Schema Registry



This configuration is required



SA-ksqlDB-ccl-mod3

Identity **Access**

Search resources

▼ Confluent (Organization)
 ▼ default (Environment)
 ► Demos (Cluster)
 ► cluster_1 (Cluster)
 ► cluster_0 (Cluster)
 ► Schema Registry
 ► CCL-training (Environment)

New schema subject permissions

Specific schema subject
Grant permissions to a specified schema subject

Prefix rule

Grant permissions to schema subjects whose name has a prefix

All schema subjects
Grant permissions to all schema subjects in this cluster

Enter schema subject*

Role

ResourceOwner
Manage rolebindings and access to resources.

DeveloperManage
Manage configs for designated cluster resources.

DeveloperRead
Data read access. Cannot manage roles.

DeveloperWrite
Data write access. Cannot manage roles.

Save

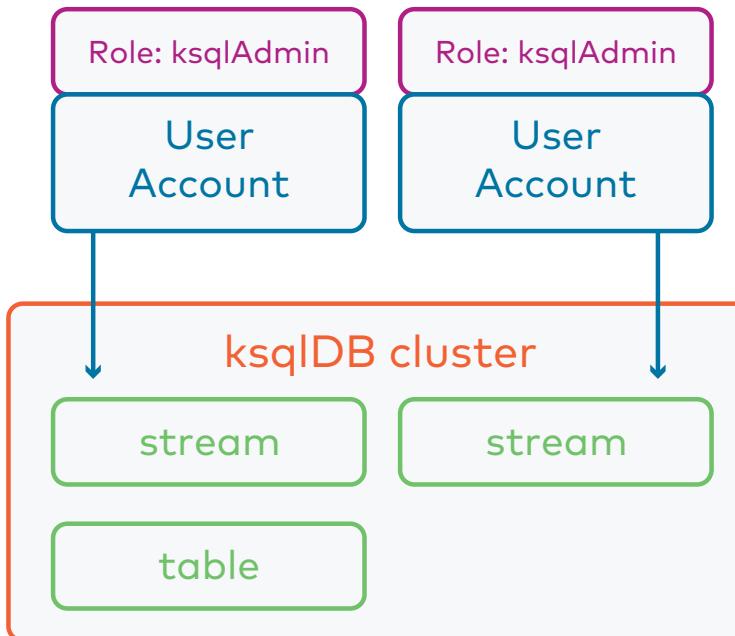
Cancel

RBAC for ksqlDB Administration

There is only 1 role for ksqlDB cluster permissions:

- **ksqlAdmin** role Access to all resources within a ksqlDB cluster.

The screenshot shows the 'Access' tab selected in the top navigation bar. On the left, a sidebar lists various resources: Confluent (Organization), default (Environment), Demos (Cluster), Topics, Consumer Groups, Transactional IDs, KsqlDB Clusters (which is highlighted in blue), Connectors, and Pipelines. A search bar at the top says 'Search resources'. The main panel is titled 'New ksqlDB cluster permissions' and contains a dropdown labeled 'Enter ksqlDB cluster*' with a placeholder. Below it, a 'Role' section shows 'KsqlAdmin' selected with the note: 'Access to all resources within a ksqlDB cluster.' At the bottom are 'Save' and 'Cancel' buttons.

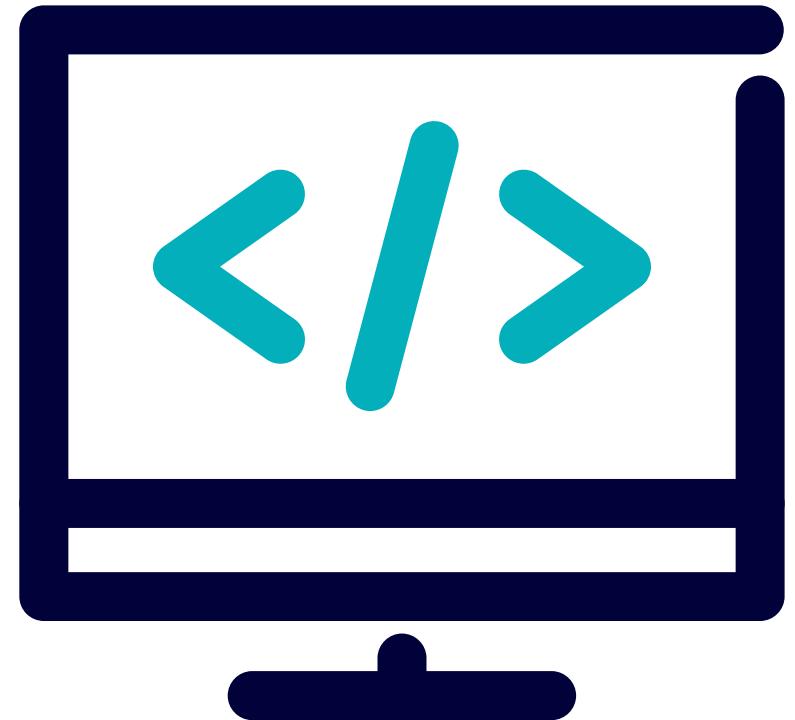


Only available on Standard and Dedicated clusters.

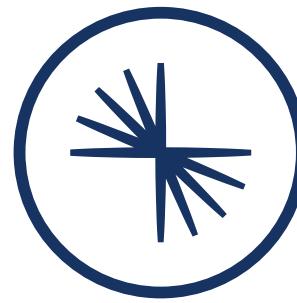
Lab Module 07: Using ksqlDB in Confluent Cloud for Stream Processing

Please work on **Lab Module 07: Using ksqlDB in Confluent Cloud for Stream Processing.**

Refer to the Exercise Guide.



08: Confluent Cloud Networking



CONFLUENT
Global Education

Module Overview



This module contains seven lessons:

- Networking Fundamentals (OPTIONAL)
- Confluent Cloud Overview
- Secure Public Endpoints
- VPC and VNet Peering
- AWS Transit Gateway
- Private Link
- Networking Summary

After this module you will be able to:

- Describe the Confluent Cloud networking structure
- List different networking options, benefits, and limitations
- Choose a suitable networking solution for a use case

08a: Networking Fundamentals (OPTIONAL)

Description

Understand fundamental networking topics in Confluent Cloud.

Learning Objectives

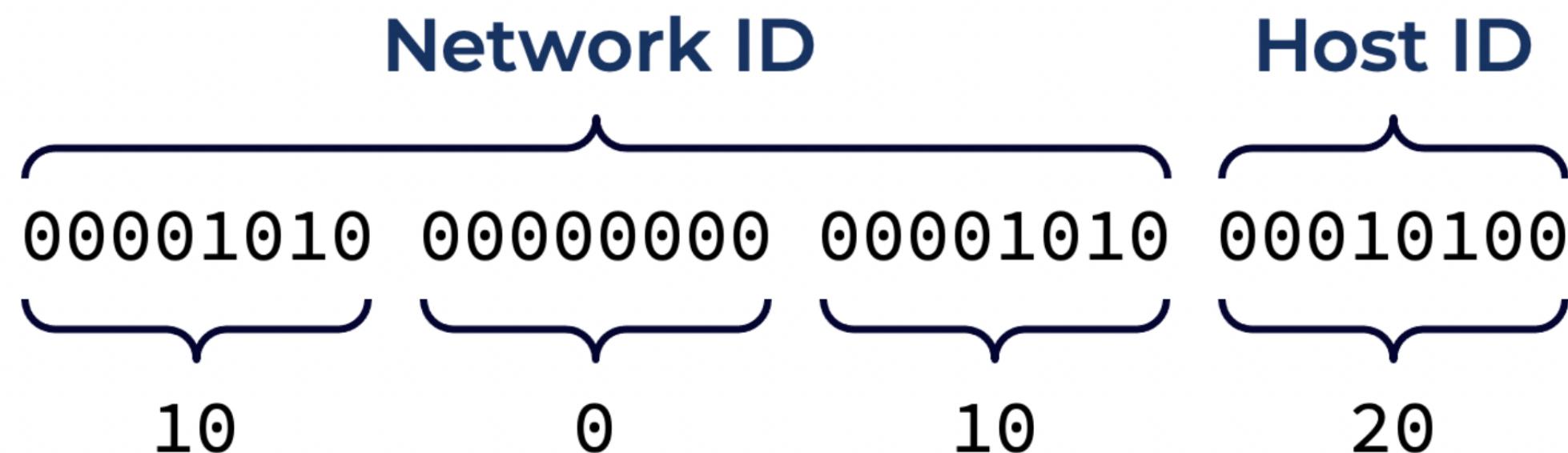


Upon completion of this lesson, you will be able to:

- Understand basic networking fundamentals, including:
 - IPv4 Addresses
 - DNS
 - VPCs and VNETs
 - Security groups, rules
 - Kafka connectivity

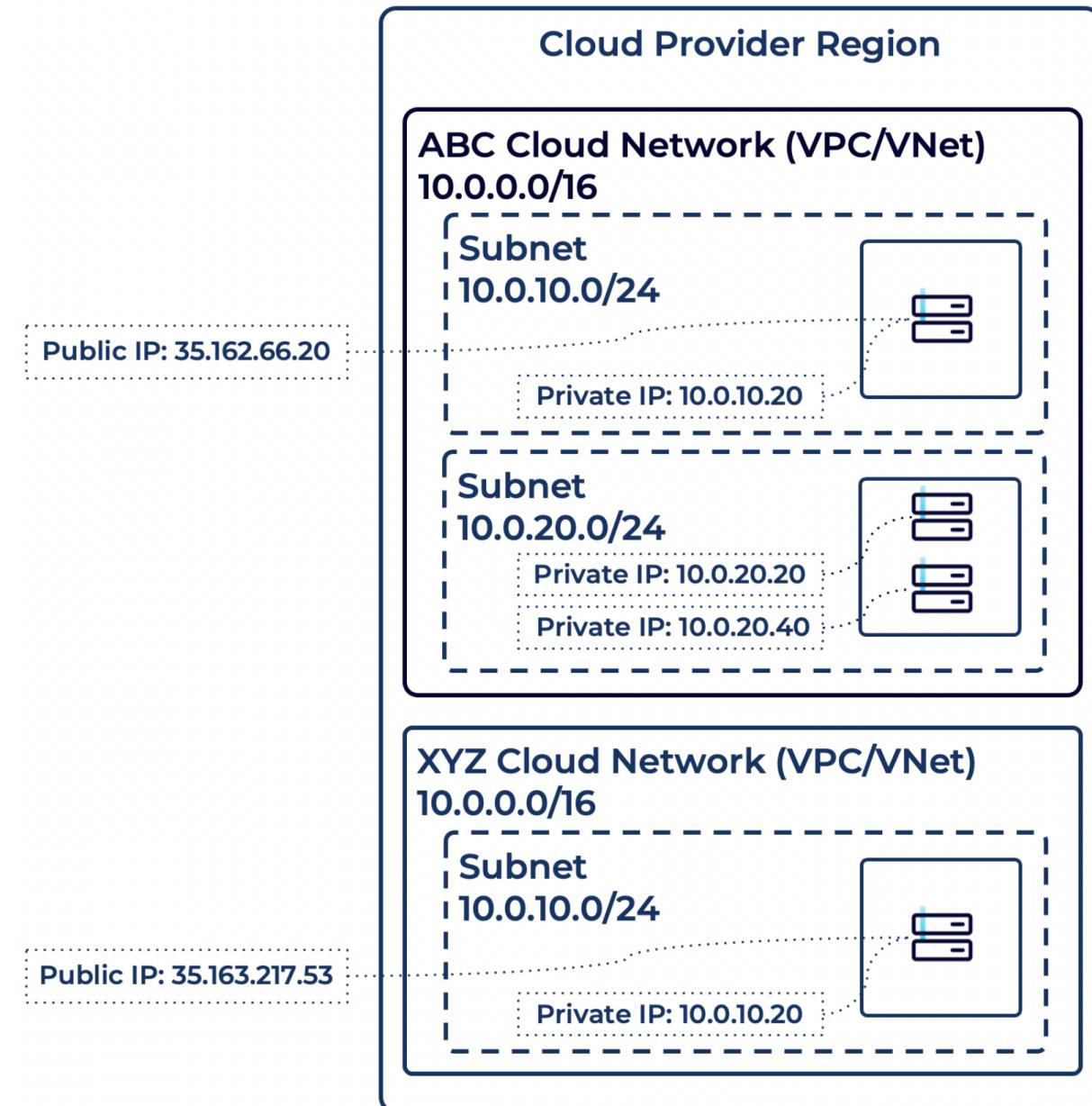
IPv4 Addresses (OPTIONAL)

- Devices that are part of a network are uniquely identified by IP address
- IPv4 Addresses:
 - 32 bit address (four "octets"): 10.0.10.20
 - Network "Mask" (e.g. /24): indicates which portion of address is network ID, and which is host ID
 - CIDR notation refers to the network range (e.g. 10.0.10.0/24)



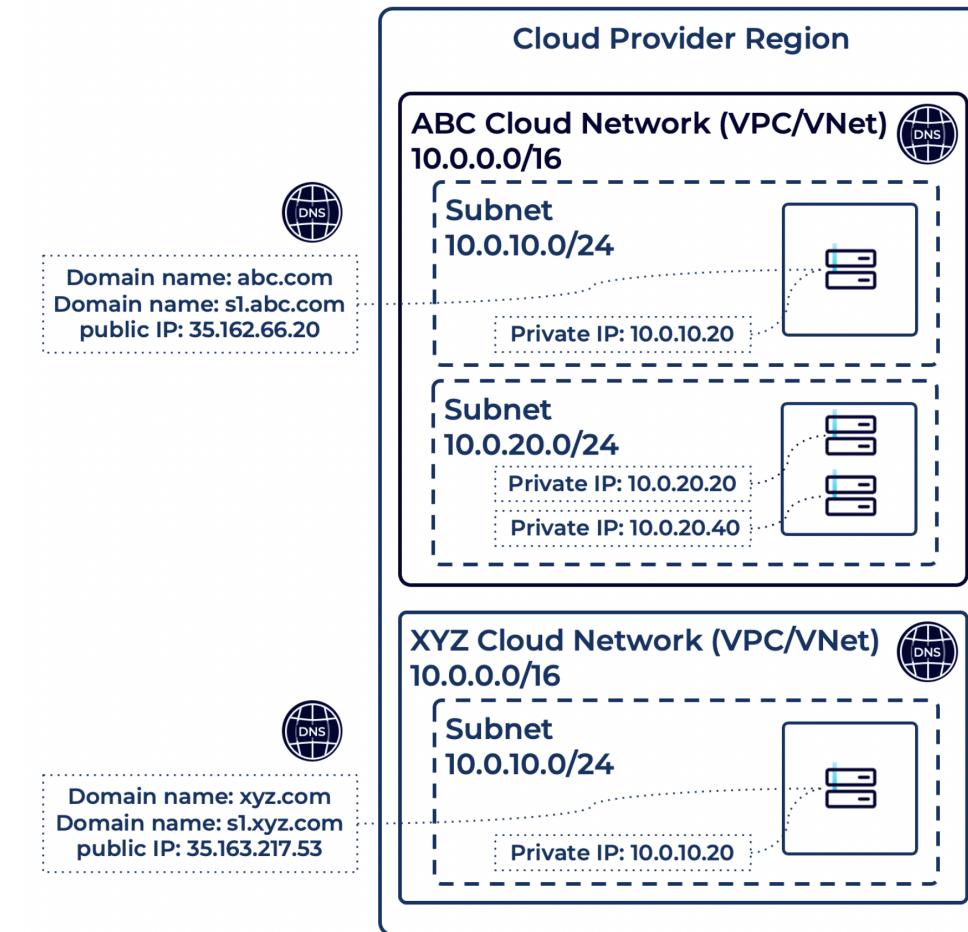
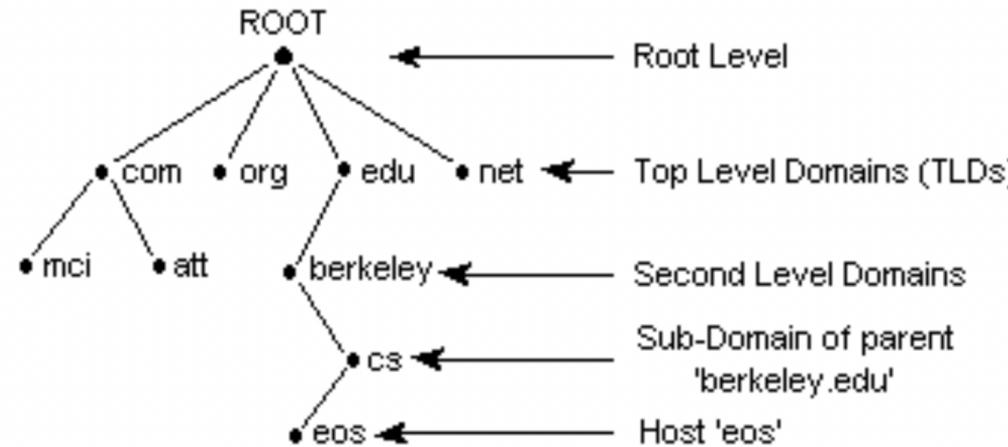
Public vs. Private (OPTIONAL)

- Public IP addresses are accessible from the public Internet (globally unique)
- Private IP addresses are accessible from their private network (unique per network)
- All devices in cloud networks are assigned a private IP address
- Devices can be assigned both private and public IP addresses



Domain Name System - DNS (OPTIONAL)

- Hierarchical and distributed system to provide information about domain names
- DNS records define “domain name” and “IP address” relationships



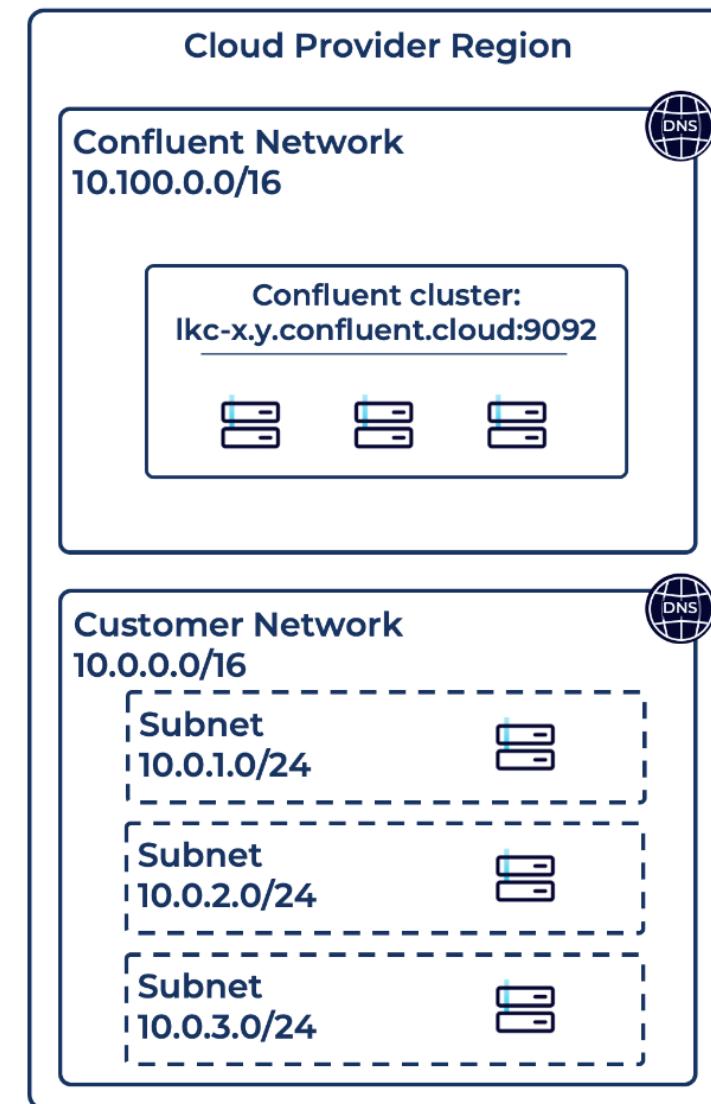
Cloud Networking (OPTIONAL)

Cloud service providers (AWS, Google, Azure) offers "Infrastructure as a Service (IaaS)":

- Virtual Machines
- Virtual Networks
 - AWS and Google → Virtual Private Clouds or VPCs
 - Azure → Virtual Networks or VNets

Customer Office
x.x.x.x/x

Customer Datacenter
a.a.a/a/a

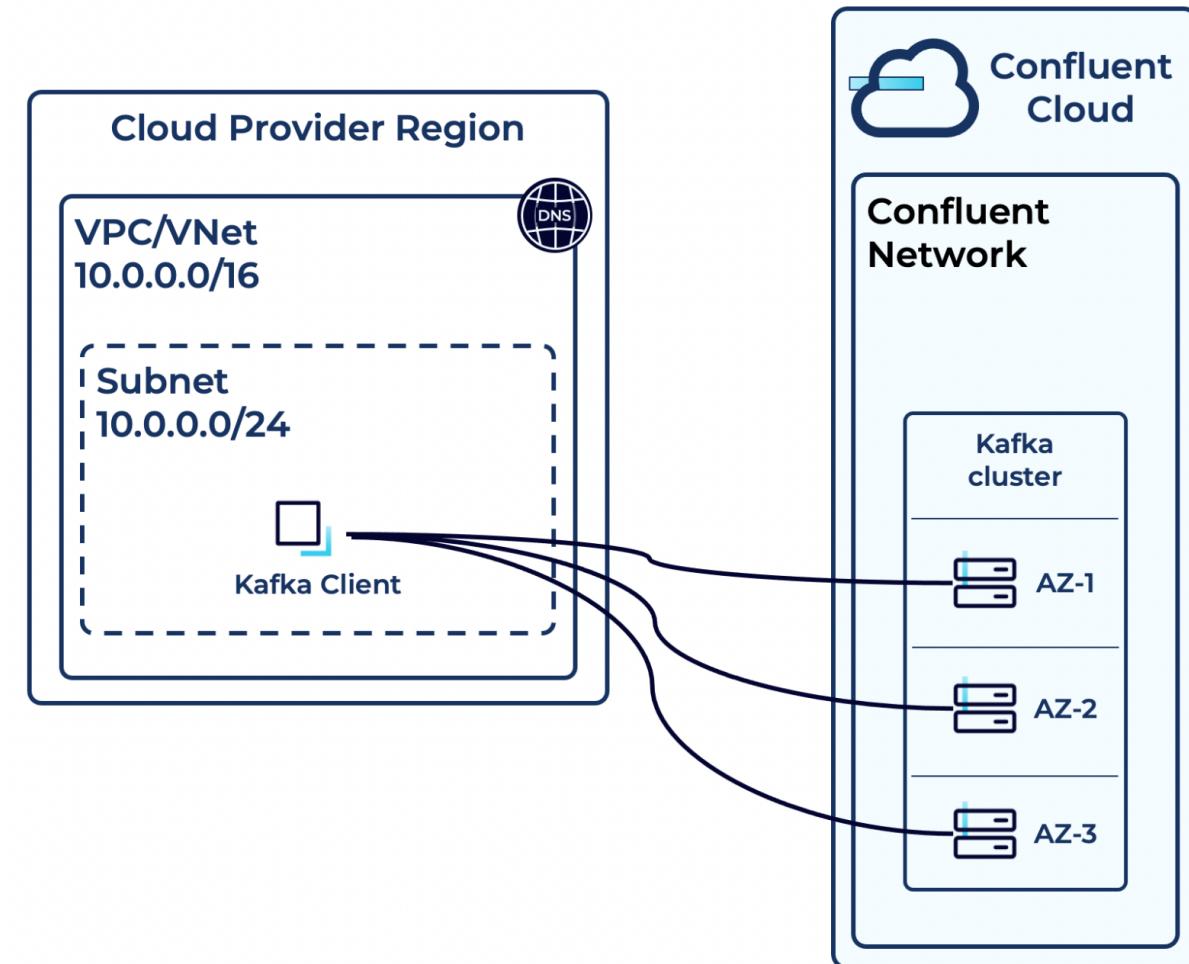


Cloud Provider Networks - VPC / VNet (OPTIONAL)

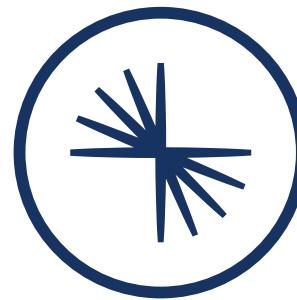
- Assigned a /16 CIDR block of IP addresses
 - Broken into subnets (/24 CIDR blocks)
 - Virtual machines assigned IP addresses (private and public)
- Cloud Regions and Availability Zones
 - VPCs or VNets run in a specific cloud provider region
 - Connect VPCs or VNets within the same region using something called VPC/VNet peering
- Routing and security rules control access
 - Traffic leaving the VPC/VNet
 - Traffic entering/traversing the VPC/VNet

Kafka Networking (OPTIONAL)

- Kafka clients communicate with Kafka brokers using TCP
- Kafka client connection data flow:
 1. The client is given one or more Kafka 'bootstrap' Kafka URLs
 2. Request cluster metadata using the bootstrap endpoint
 3. This metadata includes connectivity information for all brokers (all broker endpoints)
 4. The client will connect to individual brokers directly to send produce (write) and consume (read) requests



09: Automate, Deploy, and Manage Confluent Cloud



CONFLUENT
Global Education

Module Overview



This module contains two lessons:

- Terraform and the Confluent Provider
- Integrate Pulumi with Terraform and the Confluent Provider

After this module you will be able to:

- Describe using the Terraform Confluent provider.
- Define how the Pulumi template can be used with Terraform and Confluent Cloud.

09a: Terraform and the Confluent Provider

Description

A review of using Terraform and the Confluent Cloud provider.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Describe using Terraform with Confluent Cloud.
- Complete the steps for using the Terraform Confluent Cloud provider, from creating a Confluent Cloud account, installing Terraform, configuring and deploying the Terraform Confluent Cloud provider.

What is Terraform?

- Open-source **infrastructure-as-code** (IaC) tool
- Very popular tool with an active community
- **Declarative** and **version-controlled** configuration
- Easy to create, modify and destroy infrastructure components across various cloud providers
- Operates in three main phases (Plan, Apply and Destroy)
- Automate the provisioning and management of resources (reduce manual intervention)



Efficient and reliable way of deploying infrastructure

Why Terraform with Confluent Cloud?

Use the **Confluent Terraform provider** to deploy and manage Confluent Cloud infrastructure:

- Human Readable Configuration
- Manage Critical Confluent Cloud Resources
- Consistent Deployability
- Multi-Cloud With Ease
- Scale Quickly
- Industry Standard

Terraform Directory

In a typical Terraform directory, you may find the following files:

- `main.tf`
- `variables.tf`
- `terraform.tfvars`
- `outputs.tf`
- `terraform.tfstate`

Understanding the `main.tf` file (I)

```
terraform {
  required_providers {
    confluent = {
      source  = "confluentinc/confluent"
      version = "1.50.0"
    }
  }
}

provider "confluent" {
  cloud_api_key    = var.confluent_cloud_api_key
  cloud_api_secret = var.confluent_cloud_api_secret
}

resource "confluent_environment" "staging" {
  display_name = "Staging"
}
```



`main.tf` file continues in the next slide

Understanding the `main.tf` file (II)

```
data "confluent_schema_registry_region" "essentials" {
    cloud      = "AWS"
    region     = "us-east-2"
    package    = "ESSENTIALS"
}

resource "confluent_schema_registry_cluster" "essentials" {
    package = data.confluent_schema_registry_region.essentials.package

    environment {
        id = confluent_environment.staging.id
    }

    region {
        id = data.confluent_schema_registry_region.essentials.id
    }
}
```

resource vs data

- **Resource**

Declarative representation of a specific resource that will be created, updated, or deleted

```
resource "confluent_environment" "staging" {  
    display_name = "Staging"  
}
```

- **Data**

Query that retrieves information or metadata from a cloud provider without creating an actual resource

```
data "confluent_schema_registry_region" "essentials" {  
    cloud      = "AWS"  
    region     = "us-east-2"  
    package    = "ESSENTIALS"  
}
```

Understanding the `variables.tf` file

```
variable "confluent_cloud_api_key" {
    description = "Confluent Cloud API Key (also referred as Cloud API ID)"
    type        = string
}

variable "confluent_cloud_api_secret" {
    description = "Confluent Cloud API Secret"
    type        = string
    sensitive   = true
}
```

Understanding the `terraform.tfvars` file

```
# The Confluent Cloud API Key to authenticate Terraform with Confluent Cloud
confluent_cloud_api_key = ""

# The Confluent Cloud API Secret to authenticate Terraform with Confluent Cloud
confluent_cloud_api_secret = ""
```

Understanding the `outputs.tf` file

```
output "resource-ids" {
    value = <<-EOT
        Environment ID: ${confluent_environment.staging.id}
        Kafka Cluster ID: ${confluent_kafka_cluster.basic.id}
        Kafka topic name: ${confluent_kafka_topic.orders.topic_name}

        Service Account and their Kafka API Key (API Key inherit the permissions granted to the owner):
        ${confluent_service_account.app-manager.display_name}:
        ${confluent_service_account.app-manager.id}
        ${confluent_service_account.app-manager.display_name}'s Kafka API Key:      "${confluent_api_key.app-
manager-kafka-api-key.id}"
        ${confluent_service_account.app-manager.display_name}'s Kafka API Secret:  "${confluent_api_key.app-
manager-kafka-api-key.secret}"

    EOT
    sensitive = true
}
```

Using Confluent Terraform Provider

Typical process to get started with Terraform:

- **Step 1:** Set up a Confluent Cloud Account and create a Cloud API Key / Secret
- **Step 2:** Install Terraform
- **Step 3:** Create the Terraform Configuration Files
- **Step 4:** Initialize Terraform:

```
terraform init
```

- **Step 5:** View Plan of Changes Before Deployment:

```
terraform plan
```

- **Step 6:** Deploy Confluent Cloud Resources:

```
terraform apply
```

- **Step 7:** Destroy Confluent Cloud Resources:

```
terraform destroy
```

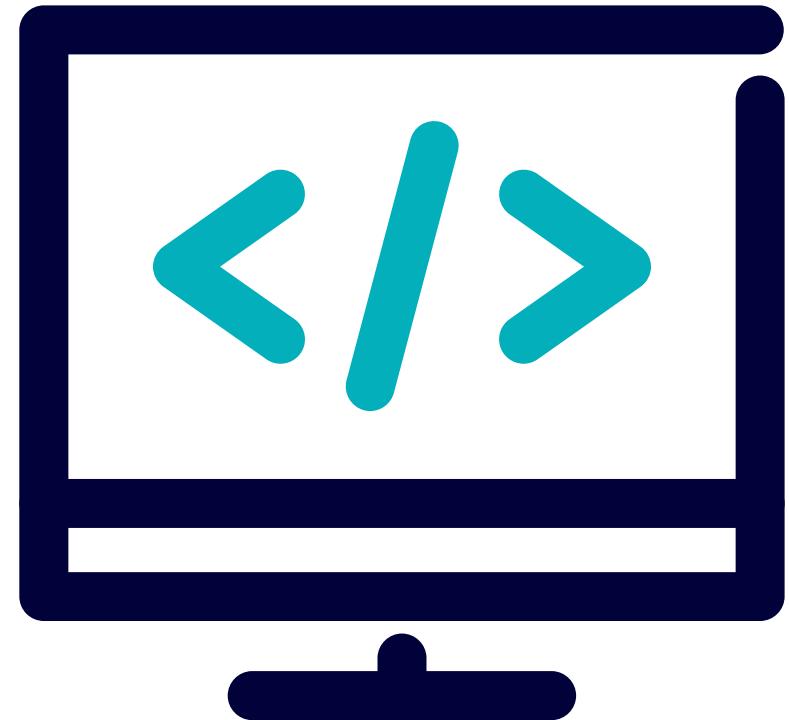
Conclusion

- Using Confluent Cloud and Terraform together provides a powerful way to manage your streaming infrastructure.
- With Terraform, you can define and manage your infrastructure as code, enabling you to easily manage your infrastructure across multiple environments and cloud providers.

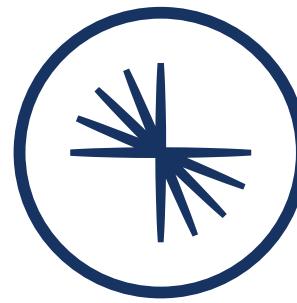
Lab Module 09: Automate, Deploy, and Manage in Confluent Cloud

Please work on **Lab Module 09: Automote, Deploy, and Manage in Confluent Cloud**.

Refer to the Exercise Guide.



10: Audit Logs, Metrics, and Notifications in Confluent Cloud



CONFLUENT
Global Education

Module Overview



This module contains three lessons:

- Confluent Cloud Audit Logs
- Confluent Cloud Metrics
- Confluent Cloud Notifications

After this module you will be able to:

- Examine Confluent Cloud audit logs, including key concepts and examples.
- Evaluate Confluent Cloud metrics, including metrics pre-requisites, using the metric API.
- List available notifications in Confluent Cloud, including key terms, subscriptions, and integrations.

10a: Audit Logs

Description

A review of audit logs in Confluent Cloud.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Examine Confluent Cloud audit logs, including key concepts and examples.

What are Audit Logs?

Information to assess security risks in your Confluent Cloud clusters:

- Authentication actions (e.g. API keys)
- Authorization actions (e.g. ACLs and RBAC)
- Organization operations (e.g. create, delete, and modify Confluent Cloud resources)

Auditable event record includes:

- Who tried to do what
- When they tried
- Whether or not the system gave permission to proceed

Audit Logs

- **Standard and Dedicated** clusters only (enabled by default)
- All audit log messages are captured in the **audit log topic**:
 - Are retained for **7 days**
 - The audit log topic name is **confluent-audit-log-events**
 - This **audit log topic** lives on an **independent Confluent Cloud cluster**
- Audit log entries cannot be:
 - modified
 - deleted
 - produced directly to the audit log topic

Audit Log Message Types

There are 3 types of Audit Log Events:

- Authentication ([io.confluent.kafka/authentication](#))
 - Authentication using an API key or token
- Authorization ([io.confluent.kafka/authorization](#))
 - Actions that are protected by ACLs or RBAC
- Request ([io.confluent.cloud/request](#))
 - Organization operations to create, delete, and modify Confluent Cloud resources:
 - API keys
 - Kafka clusters
 - User accounts
 - Service accounts
 - Connectors
 - and more...

Audit Log Record Example

Each audit log record is a JSON message comprising these two parts:

```
{  
    "id": "fc0f727d-899a-4a22-ad8b-a866871a9d37",  
    "source": "crn://confluent.cloud/kafka=lkc-a1b2c",  
    "specversion": "1.0",  
    "type": "io.confluent.kafka.server/authorization",  
    "datacontenttype": "application/json",  
    "subject": "crn://confluent.cloud/kafka=lkc-a2b2c",  
    "time": "2021-01-01T12:34:56.789Z",  
    "data": {  
        "serviceName": "crn://confluent.cloud/kafka=lkc-a1b2c",  
        "methodName": "kafka.CreateTopics",  
        "resourceName": "crn://confluent.cloud/kafka=lkc-a1b2c/topic=departures",  
        "authenticationInfo": {  
            "principal": "User:123456"  
        },  
        "authorizationInfo": {  
            "granted": true,  
            "operation": "DescribeConfigs",  
            "resourceType": "Topic",  
            "resourceName": "departures",  
            "patternType": "LITERAL",  
            "superUserAuthorization": true  
        },  
        "request": {  
            "correlationId": "123",  
            "clientId": "adminclient-42"  
        }  
    }  
}
```

context

data

Important Message Properties (I)

Event Property	Description
id	A randomly-generated UUID that ensures uniqueness across all sources
source	The source location of the auditable event in the form of CRN (Confluent Resource Name)
subject	The resource affected by the auditable event in the form of CRN (Confluent Resource Name)
type	<ul style="list-style-type: none">• io.confluent.kafka.server/authentication• io.confluent.kafka.server/authorization• io.confluent.cloud/request
time	The timestamp, in RFC 3339 format, for when the event occurred

Important Message Properties (II)

Event Property	Description
methodName	The audit log method that triggered the auditable event message
serviceName	The service that generated this audit log in the form of CRN (Confluent Resource Name)
resourceName	The CRN of the affected resource

Confluent Resource Names (CRNs)

- It provides a uniform way to uniquely identify resources
- Is prefixed with `crn://`
 - Cluster Resource CRN:

```
crn://confluent.cloud/organization=fc5ba16d-661d-474c-85df-c2a1ed26032c/environment=env-pqr45/cloud-cluster=lkc-a1b2c/kafka=lkc-a1b2c
```

- Topic CRN:

```
crn://confluent.cloud/kafka=lkc-a1b2c/topic=my-topic
```
- Service Account CRN:

```
crn://confluent.cloud/organization=fc5ba16d-661d-474c-85df-c2a1ed26032c/service-account=sa-ymnkzp
```
- Schema Registry CRN:

```
crn://confluent.cloud/organization=fc5ba16d-661d-474c-85df-c2a1ed26032c/environment=env-pqr45/schema-registry=lsrc-m1n2o
```

Audit Log Message Schema

- The schema is a JSON Schema format
- Compatibility policy based on BACKWARDS compatibility:
 - Non-breaking changes to the schema without advance notice
 - Breaking changes will be widely communicated at least 180 days in advance



Exceptions to this policy apply in case of critical security vulnerabilities or functional defects

Audit Log Message Schema File - SAMPLE

```
{  
  "$schema": "http://json-schema.org/draft-07/schema#",  
  "type": "object",  
  "description": "This is v1.2 of the Confluent Cloud audit log event schema. It is based on CloudEvents. Links to the latest version can be found at https://docs.confluent.io/cloud/current/monitoring/audit-logging. Confluent will make non-breaking changes to the schema without advance notice. Breaking changes will be widely communicated at least 180 days in advance, and we will continue to maintain compatibility during this time. Exceptions to this policy apply in case of critical security vulnerabilities or functional defects. Check the above web page for links to our announcement channels and details about our compatibility policy.",  
  "properties": {  
    "id": {  
      "type": "string",  
      "minLength": 1,  
      "description": "Uniquely identifies the event.",  
      "examples": [  
        "c72a3d0c-e6f3-4196-9b49-a835614452df"  
      ]  
    },  
  ...  
}
```

Auditable Event Methods

- Not all operations generate an Audit Log Record (e.g. describe a topic)
- There are more than 100 **Auditable Event Methods** that generate **Audit Log Records**:
 - Authentication and Authorization (Kafka Cluster & Schema Registry)
 - Kafka Cluster Management
 - OAuth/OIDC Identity Provider and Identity Pool
 - Organization
 - Networking
 - Pipeline (Stream Designer)
 - Role-based Access Control (RBAC)

Auditable Event Methods - Examples

Event Method	Description
<code>kafka.Authentication</code>	Method triggered by a request for authentication using an API key or token
<code>DeleteUser</code>	Method triggered by a request to delete a user account
<code>PauseConnector</code>	Method triggered by a request to pause a connector
<code>CreatePrivateLinkAccess</code>	Method triggered by a request to create a private link access
<code>BindRoleForPrincipal</code>	Method triggered by a request to bind a role for a principal
<code>kafka.CreateTopics</code>	Method triggered by a request to authorize creating topics
<code>kafka.CreateTopics</code>	Method triggered by a request to create Kafka topics

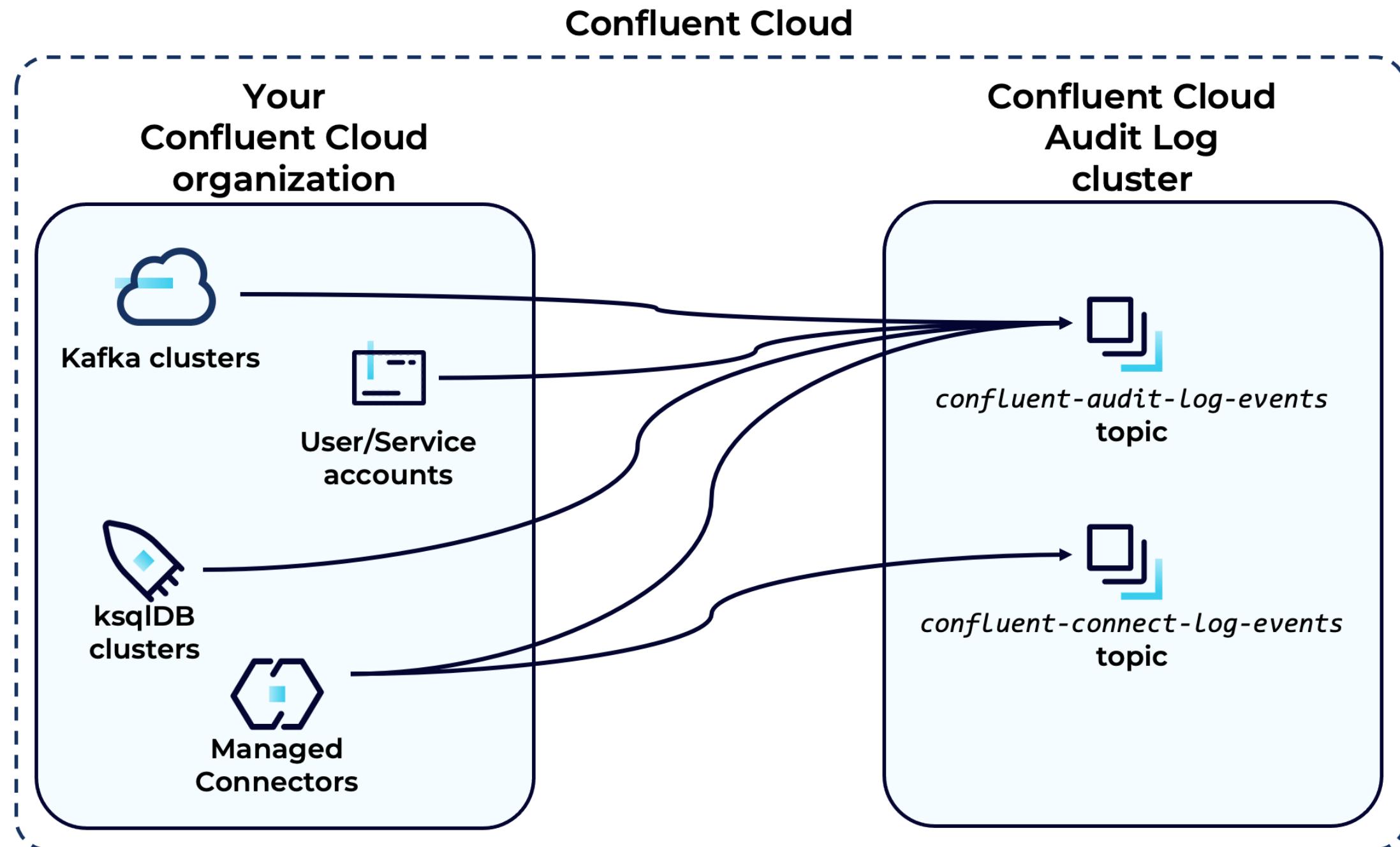
kafka.CreateTopics (authorization)

```
{  
  "type": "io.confluent.kafka.server/authorization",  
  "data": {  
    "methodName": "kafka.CreateTopics",  
    "serviceName": "crn://confluent.cloud/kafka=lkc-a1b2c",  
    "resourceName": "crn://confluent.cloud/kafka=lkc-a1b2c",  
    "authenticationInfo": {  
      "principal": "User:123456"  
    },  
    "authorizationInfo": {  
      "granted": true,  
      "operation": "Create",  
      "resourceType": "Cluster",  
      "resourceName": "kafka-cluster",  
      "patternType": "LITERAL",  
      "superUserAuthorization": true  
    },  
    "request": {  
      "correlationId": "123",  
      ...  
    }  
  }  
}
```

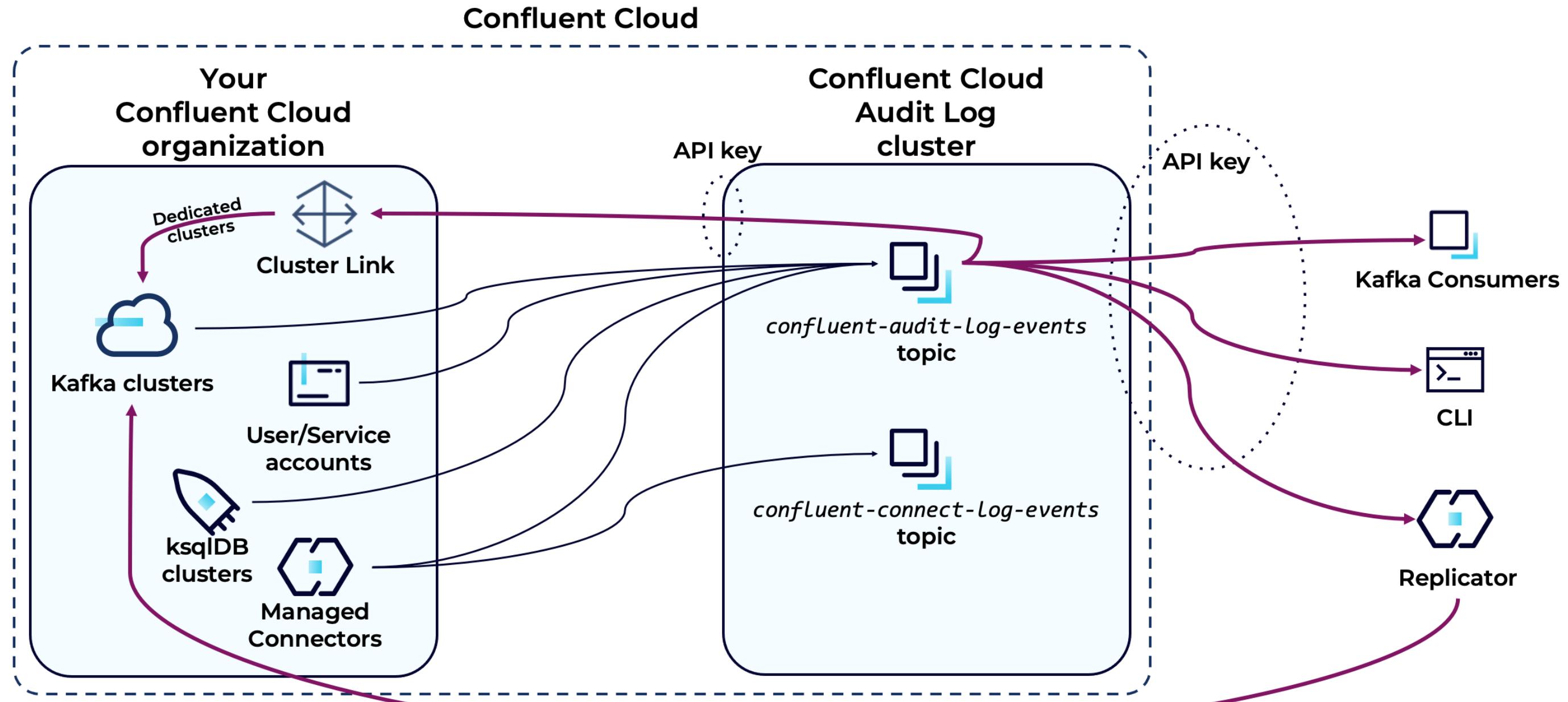
kafka.CreateTopics (topic creation)

```
{  
  "type": "io.confluent.kafka.server/request",  
  "data": {  
    "methodName": "kafka.CreateTopics",  
    "serviceName": "crn://confluent.cloud/",  
    "resourceName": "crn://confluent.cloud/organization=91a07ac9-9a13-491e-a2cf-  
46a9a5cf339c/environment=env-311k0/cloud-cluster=lkc-rjy79/kafka=lkc-rjy79/topic=test-topic",  
    "cloudResources": [  
      {  
        "scope": {  
          "resources": [  
            {  
              "type": "ORGANIZATION",  
              "resourceId": "91a07ac9-9a13-491e-a2cf-46a9a5cf339c"  
            },  
            {  
              "type": "ENVIRONMENT",  
              "resourceId": "env-311k0"  
            },  
            ...  
          ]  
        }  
      ]  
    ]  
  }  
}
```

Audit Log Architecture



Consume Audit Log Records



Audit Log Cluster - API Keys

- Limit of **2 API keys** per Audit Log cluster (Audit logs AND Connect log events)
- Use the Confluent CLI to create the API keys:
 - Describe the Audit Log cluster:
 - *Environment ID*
 - *Cluster ID*
 - *Service Account ID*
 - *Topic Name*
 - Select the correct cluster using the *Environment ID* and *Cluster ID*
 - Create the API key providing the *Service Account ID* and *Cluster ID*

(Show Instructions in the Confluent Cloud UI)

10b: Metrics

Description

A review of metrics in Confluent Cloud.

Learning Objectives



Upon completion of this lesson and associated lab exercises, you will be able to:

- Evaluate Confluent Cloud metrics, including metrics prerequisites, using the metric API.

Confluent Cloud Metrics

Metrics are available using:

- First-class integrations with third-party monitoring providers:
 - Datadog
 - Dynatrace
 - Grafana Cloud
 - Prometheus
 - New Relic OpenTelemetry
- Directly querying the Confluent Cloud Metrics API



Metrics are retained for seven days

Metrics Prerequisites

- Create a new Service Account:
 - Cloud API Key (Authentication)
 - MetricsViewer role (Authorization)

Confluent Cloud Metrics API

- It provides actionable operational metrics about your Confluent Cloud deployment
- It is a queryable HTTP API:
 - POST a query written in JSON
 - Get back a time series of metrics specified by the query

Discovery using the Metrics API

Programmatic discovery of endpoints can be used to avoid hardcoding **metrics** and **resources** into client scripts

Get all the available **metrics** by sending a **GET** request to the **descriptors/metrics** endpoint:

```
http 'https://api.telemetry.confluent.cloud/v2/metrics/cloud/descriptors/metrics?resource_type=kafka'  
--auth '<API_KEY>:<SECRET>'
```

Get all the available **resources** by sending a **GET** request to the **descriptors/resources** endpoint:

```
http 'https://api.telemetry.confluent.cloud/v2/metrics/cloud/descriptors/resources' --auth  
'<API_KEY>:<SECRET>'
```

Metrics API Query - Example

Objective - Get average consumer lag over the last hour grouped by topic and consumer group

1. Create a JSON file containing the query:

```
{  
  "aggregations": [{"metric": "io.confluent.kafka.server/consumer_lag_offsets"}],  
  "filter": {  
    "field": "resource.kafka.id",  
    "op": "EQ",  
    "value": "lkc-XXXXXX"  
  },  
  "granularity": "PT1H",  
  "group_by": [  
    "metric.consumer_group_id",  
    "metric.topic"  
  ],  
  "intervals": ["PT1M/now"],  
  "limit": 25  
}
```

Metrics API Query - Example Cont'd

2. Submit the query as a **POST** including the JSON file and the API Key/Secret:

```
http 'https://api.telemetry.confluent.cloud/v2/metrics/cloud/query' \
--auth '<API_KEY>:<SECRET>' < file.json
```

3. The response should resemble:

```
{ "data": [
  {
    "metric.consumer_group_id": "group_1",
    "metric.topic": "test_topic_1",
    "timestamp": "2022-03-23T21:00:00Z",
    "value": 0.0
  },
  {
    "metric.consumer_group_id": "group_2",
    "metric.topic": "test_topic_2",
    "timestamp": "2022-03-23T21:00:00Z",
    "value": 6.0
  }
]
```

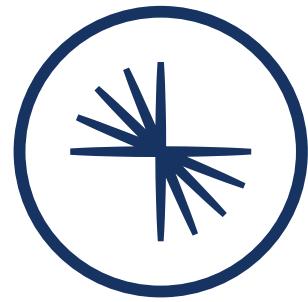
Lab Module 10: Audit Logs, Metrics, and Notifications in Confluent Cloud

Please work on **Lab Module 10: Audit Logs, Metrics, and Notifications in Confluent Cloud**.

Refer to the Exercise Guide.



11: Real-Life Use Case



CONFLUENT
Global Education

Module Overview



This module contains one lab to practice everything you have learned during these three days.

The objectives of this lab are:

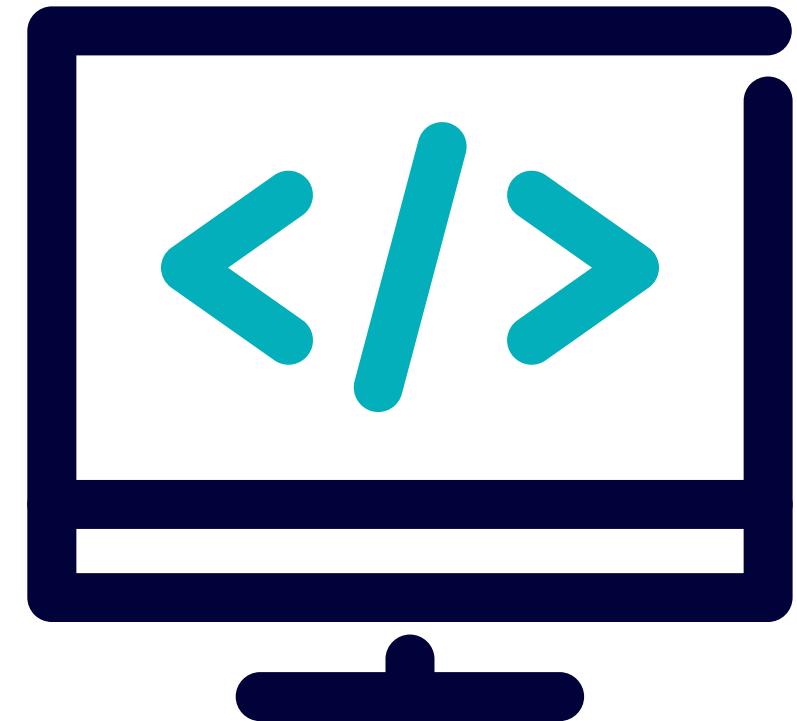
- Apply best practices when designing your Confluent Cloud Org
- Set appropriate security configurations to your User and Service accounts using API Keys, RBAC, and ACLs

- Use Confluent CLI and APIs to perform common operations.
- Integrate Schema Registry in your workloads
- Mirror data between Kafka clusters using Schema Linking and Cluster Linking
- Understand your streams and their relationships using Stream Lineage & Catalog
- Perform stream processing in Confluent Cloud with ksqlDB
- Integrate external data systems in Confluent Cloud using managed connectors

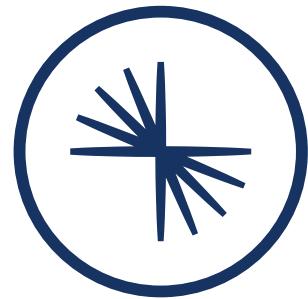
Lab Module 11: Real-Life Use Case

Please work on **Lab Module 11: Real-Life Use Case**.

Refer to the Exercise Guide.



Course Conclusion



CONFLUENT
Global Education

Confluent Certified Developer for Apache Kafka

Duration: 90 minutes

Qualifications: Solid understanding of Apache Kafka and Confluent products, and 6-to-9 months hands-on experience

Availability: Live, online, 24-hours a day!

Cost: \$150

Register online: www.confluent.io/certification



Confluent Certified Administrator for Apache Kafka

Duration: 90 minutes

Qualifications: Solid work foundation in Confluent products and 6-to-9 months hands-on experience

Availability: Live, online, 24-hours per day!

Cost: \$150

Register online: www.confluent.io/certification



We Appreciate Your Feedback!



Please complete the course survey now.