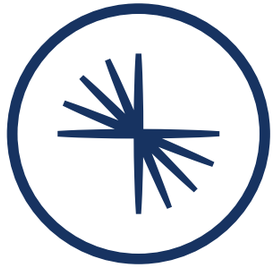
Setting Data in Motion with Confluent Cloud

Version 7.3.3-v1.0.0

Table of Contents

Introduction



CONFLUENT

Global Education

Copyright & Trademarks

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

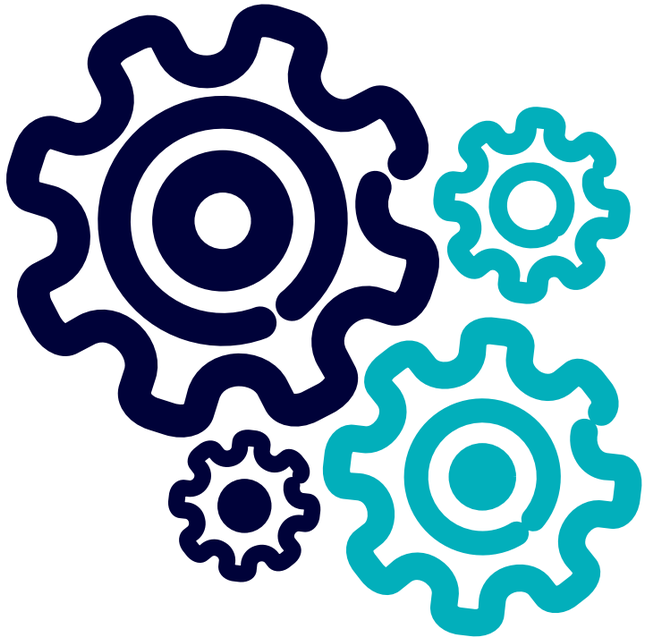
Throughout the course, Hands-On Exercises will reinforce the topics being discussed.

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]

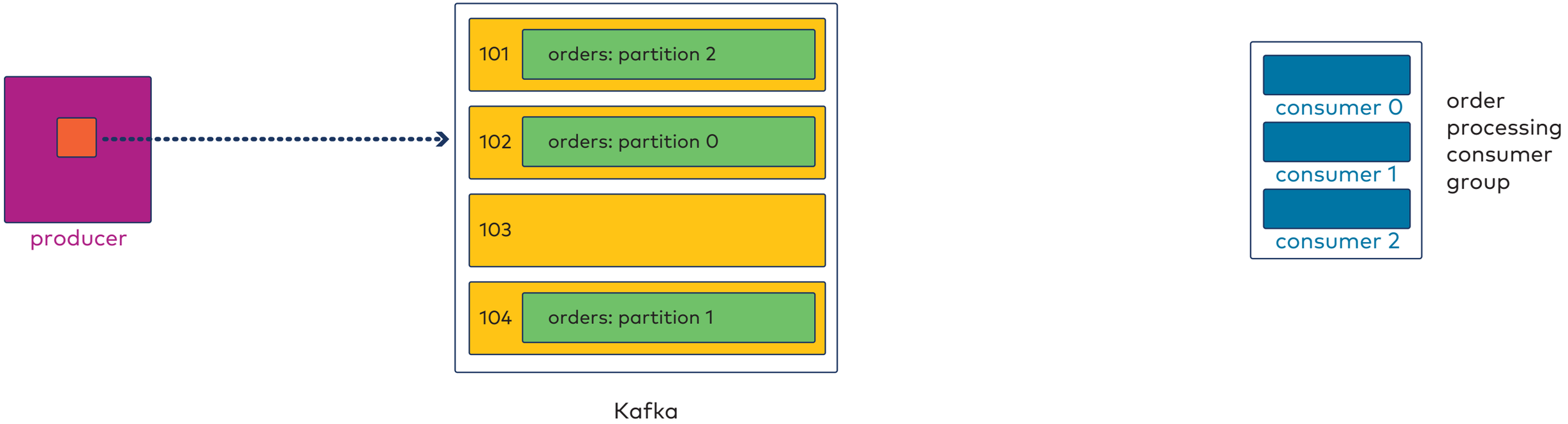
Determine the best answer to each question.

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.

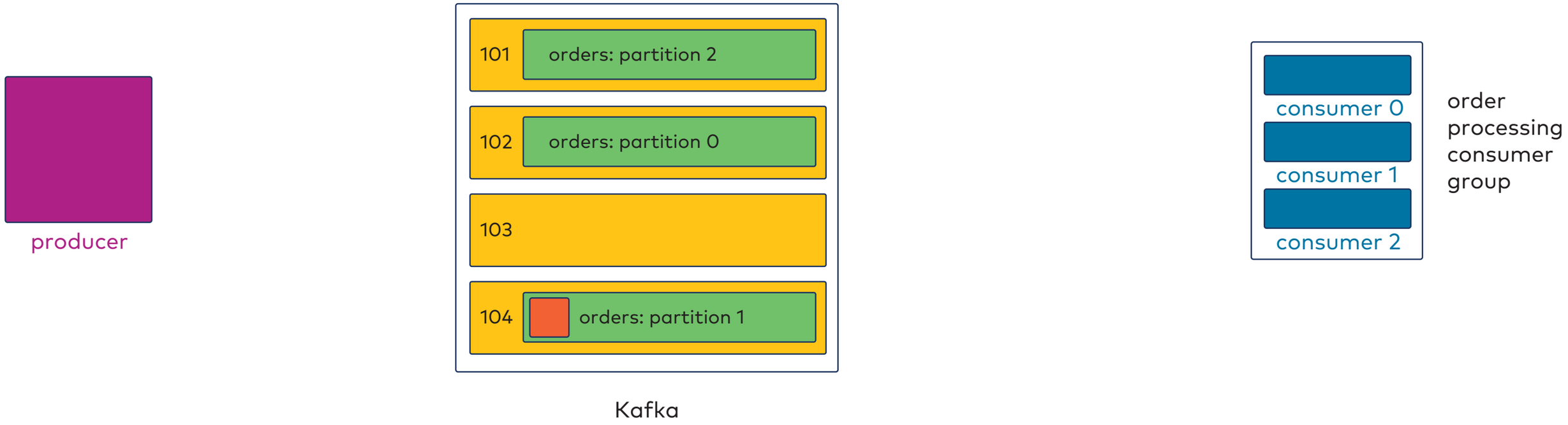
Life Cycle of a Message: Producing (Optional)



- 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)

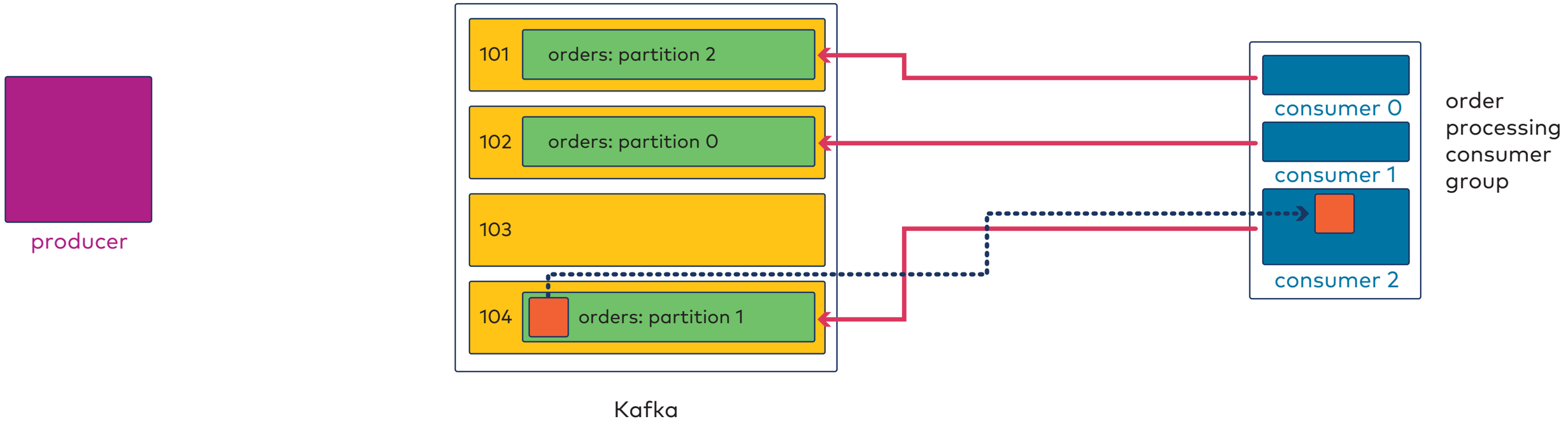
Produced messages live in Kafka, organized by topic.



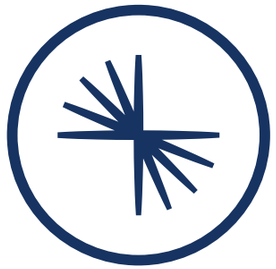
- Kafka consists of brokers
- Brokers contain partitions, which contain messages
- Brokers handle retention and replication

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



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

aws

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:

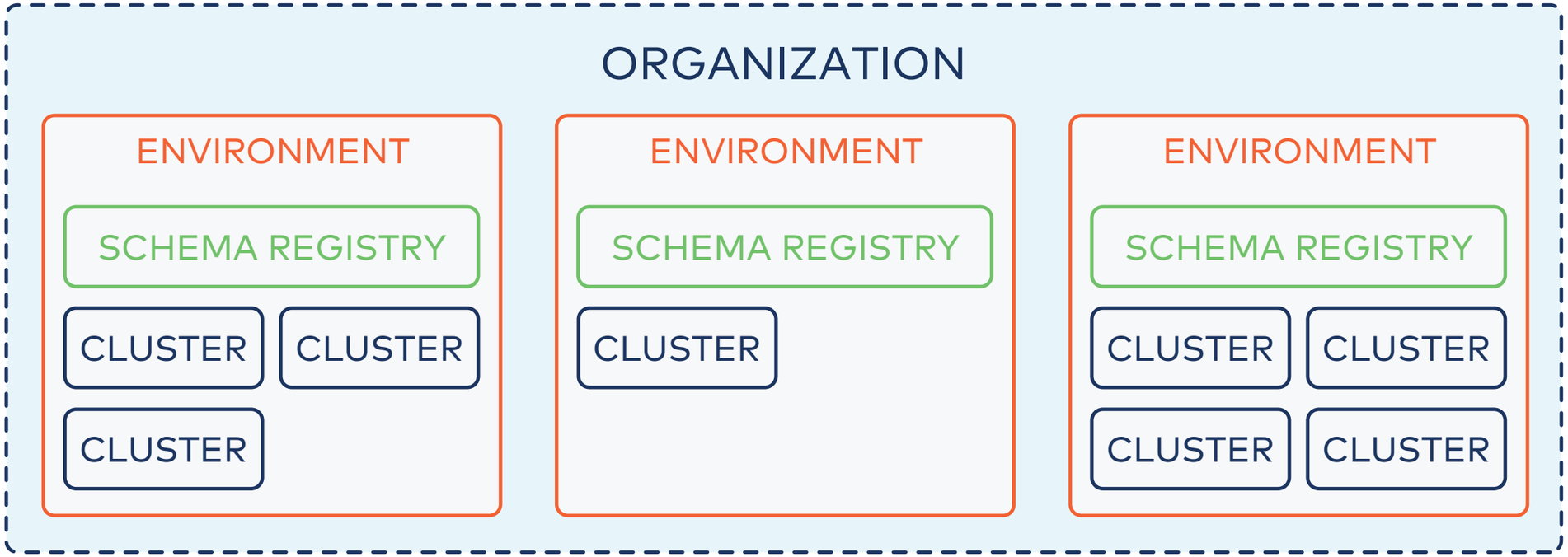
- Definition of how Confluent Cloud is structured.
- Comparison between the three types of clusters and discussion about pricing.
- Description of each component in Confluent Cloud.

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	• Throughput: 250 / 750 MBps • Storage: 5 TB • Partitions: 4096	• Throughput: 250 / 750 MBps • Infinite Storage • Partitions: ~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: $/CKU/hour Minimum: 1 CKU Maximum: 100 CKUs
Based on consumption: • Ingress ($/GB) • Egress ($/GB) • Partitions ($/partition/hour) • Storage ($/GB-month)		Based on consumption: • Ingress ($/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.

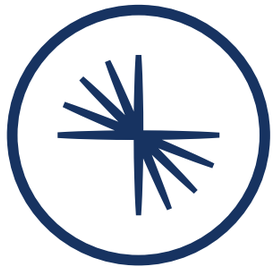
Confluent strongly recommends that you use service accounts for **all production-critical access**.

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

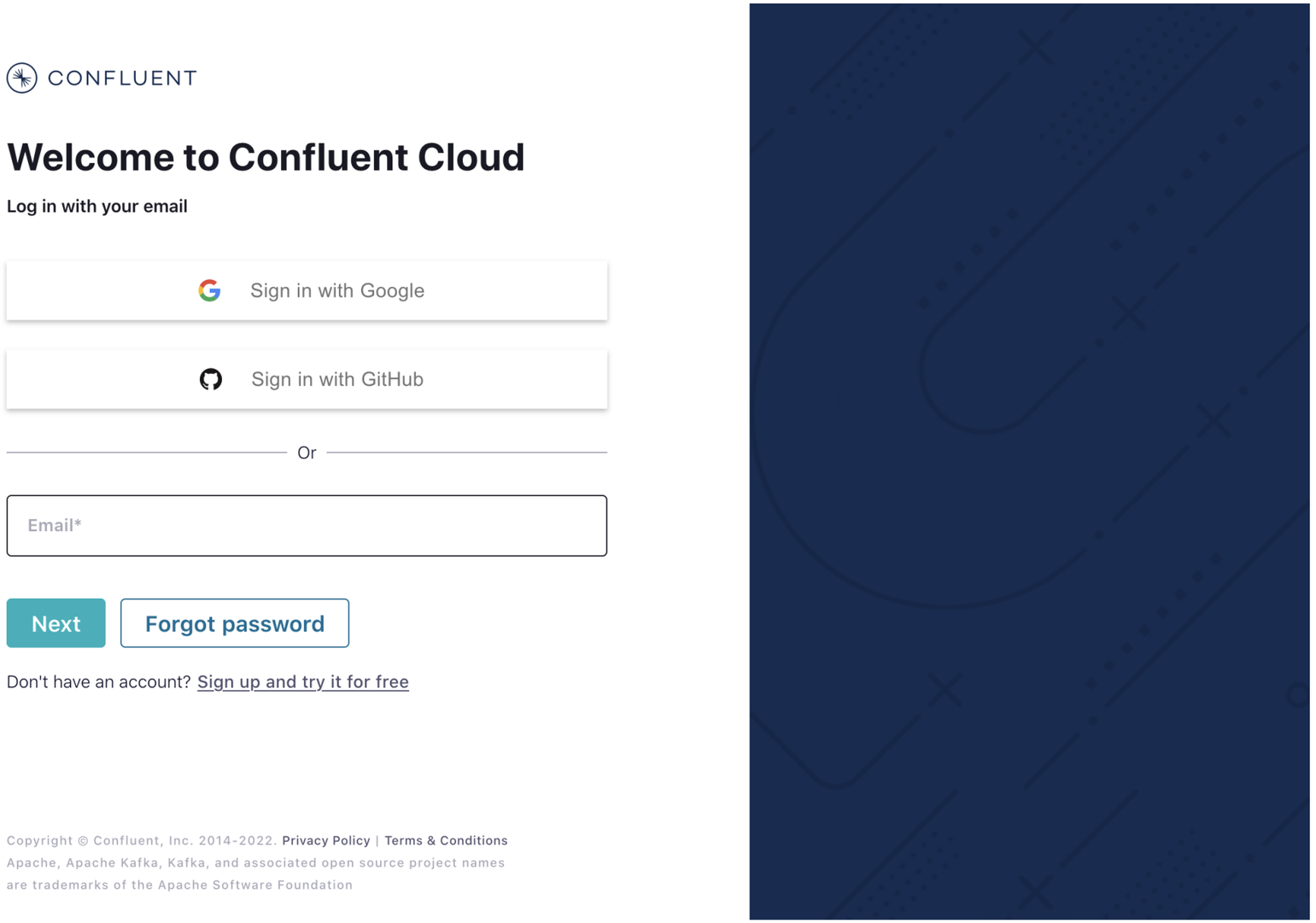
Your instructor will provide a demonstration of the Confluent Cloud console.

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



02b: 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
`confluent help`	-	Get information about all available commands
`confluent login`	-	Login to your account
`confluent environment`	`list`	List all environments you have access to
	`use`	Set your current environment
`confluent kafka cluster`	`list`	List all clusters you have access to in the current environment
	`use`	Set your current cluster in the current environment

Important Commands for Cloud Administrators

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

02d: 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

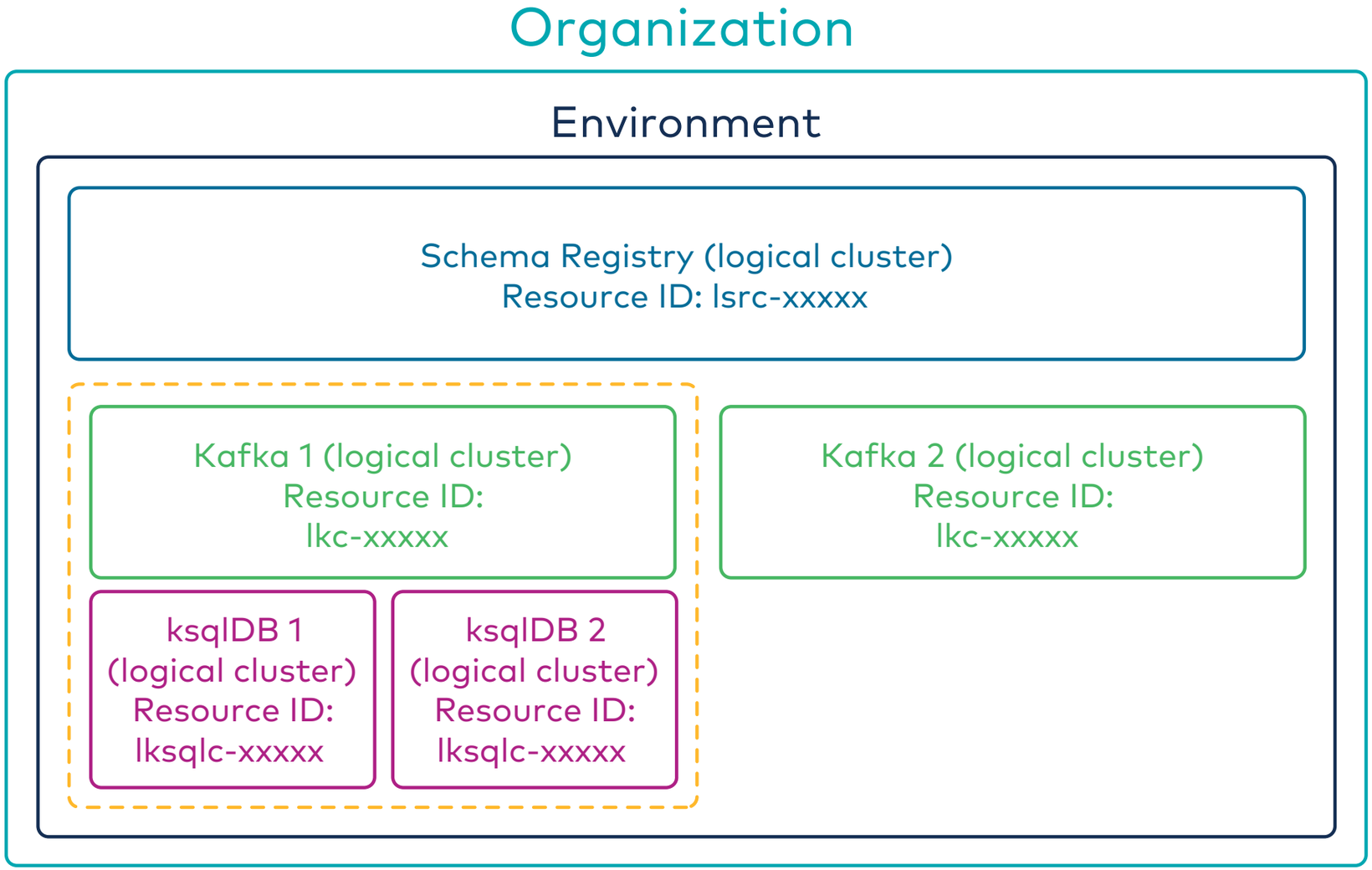
API keys control access to Confluent Cloud components and resources.

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

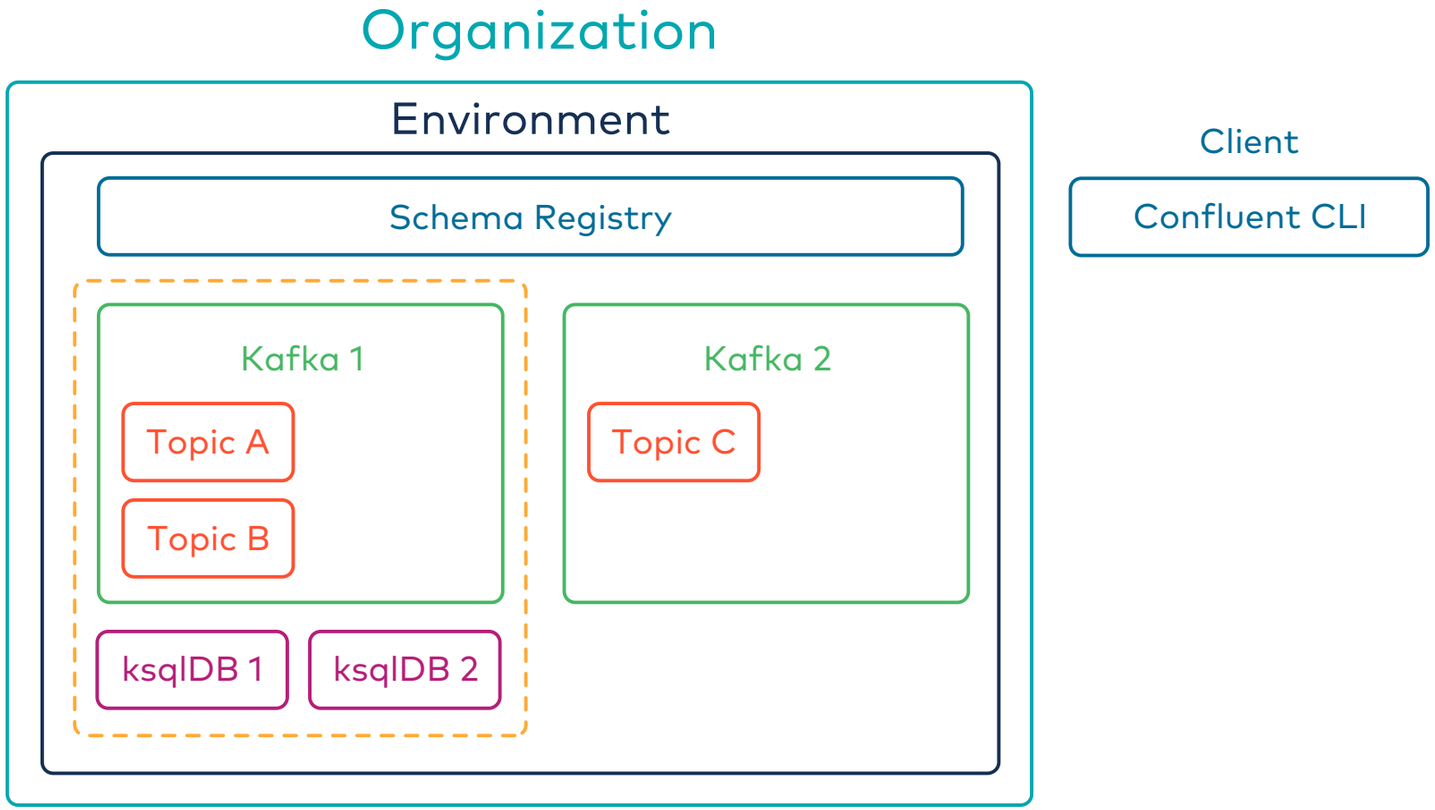
API Keys - Cluster Overview



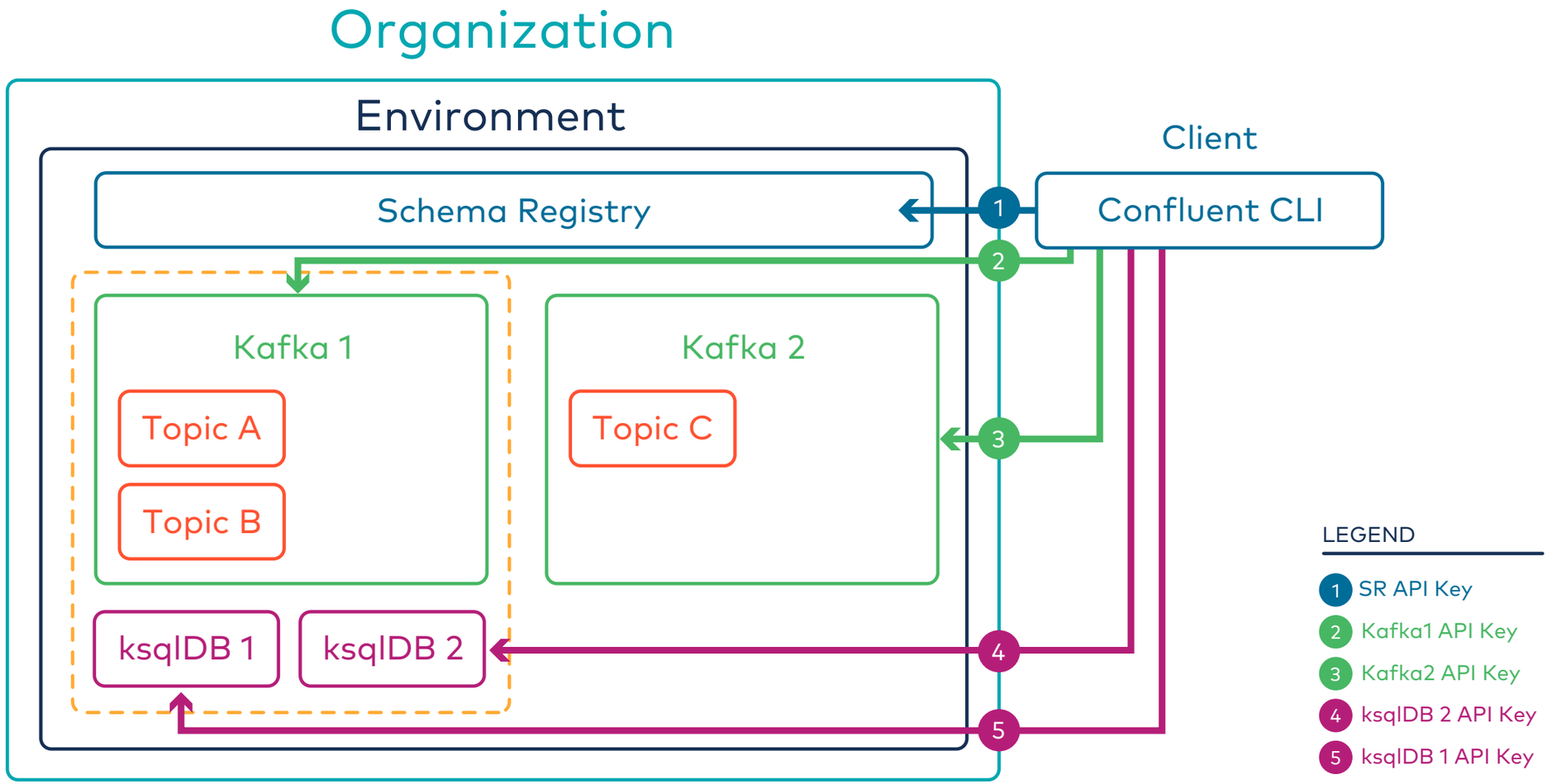
API Keys - Confluent CLI Client (1)

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?



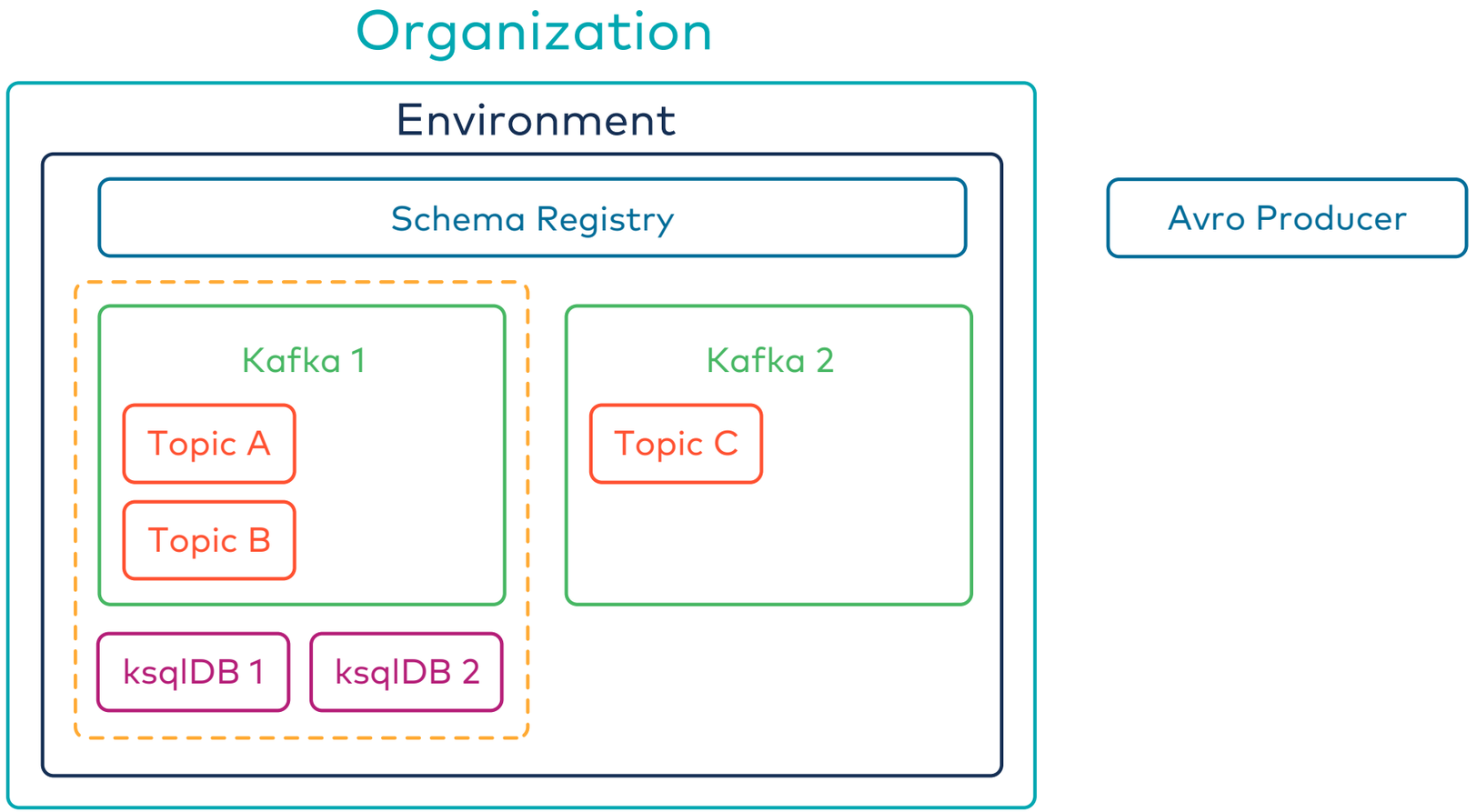
API Keys - Confluent CLI Client (2)



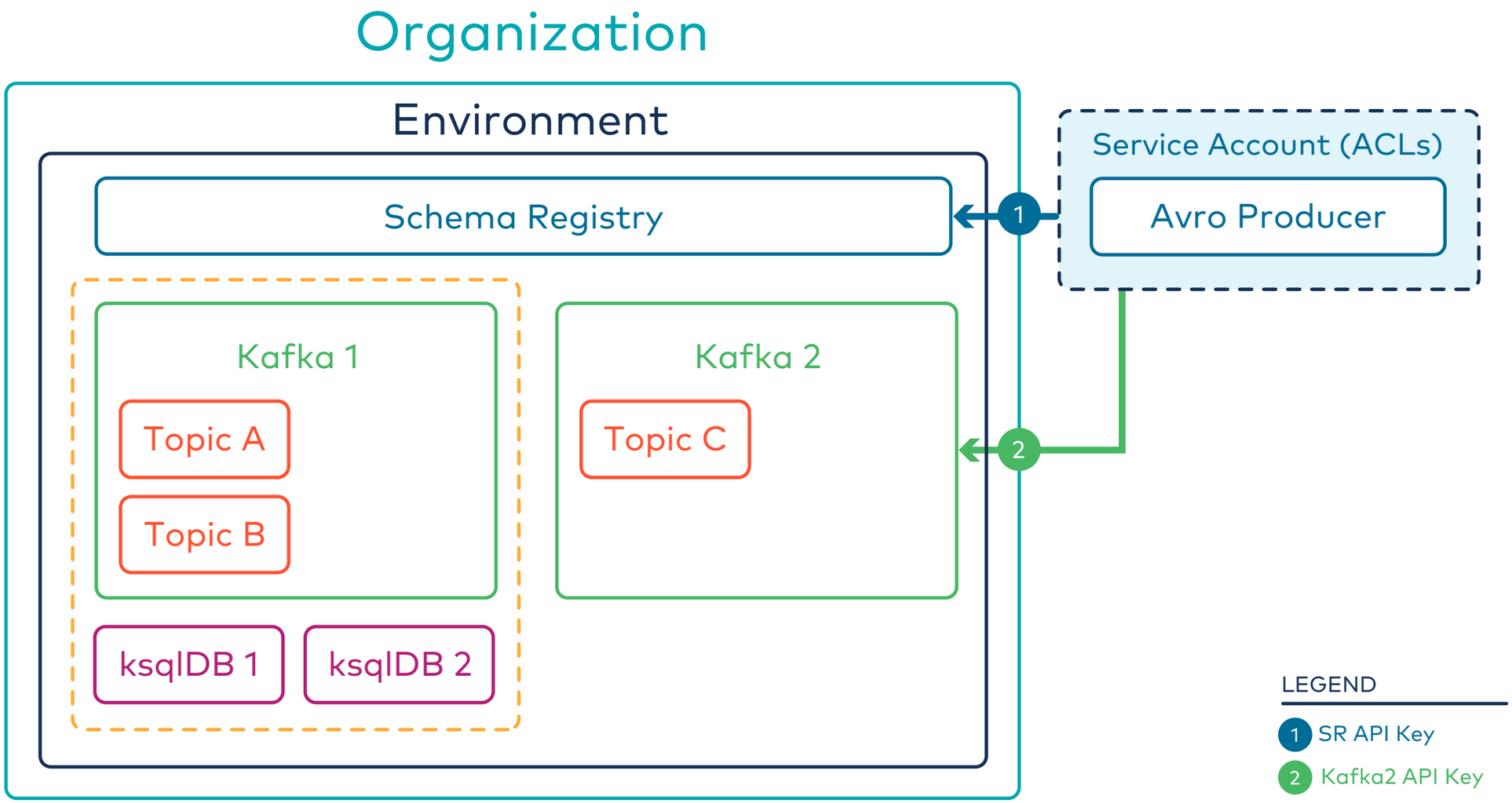
API Keys - Kafka App (1)

Question:

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



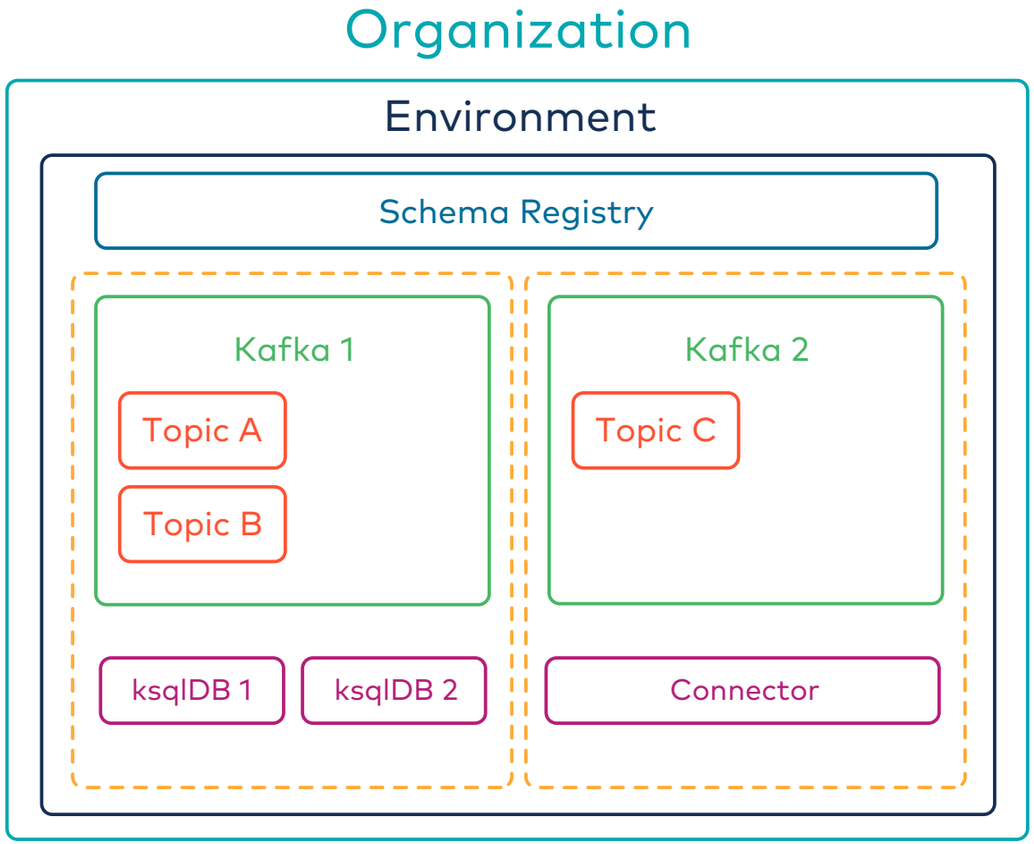
API Keys - Kafka App (2)



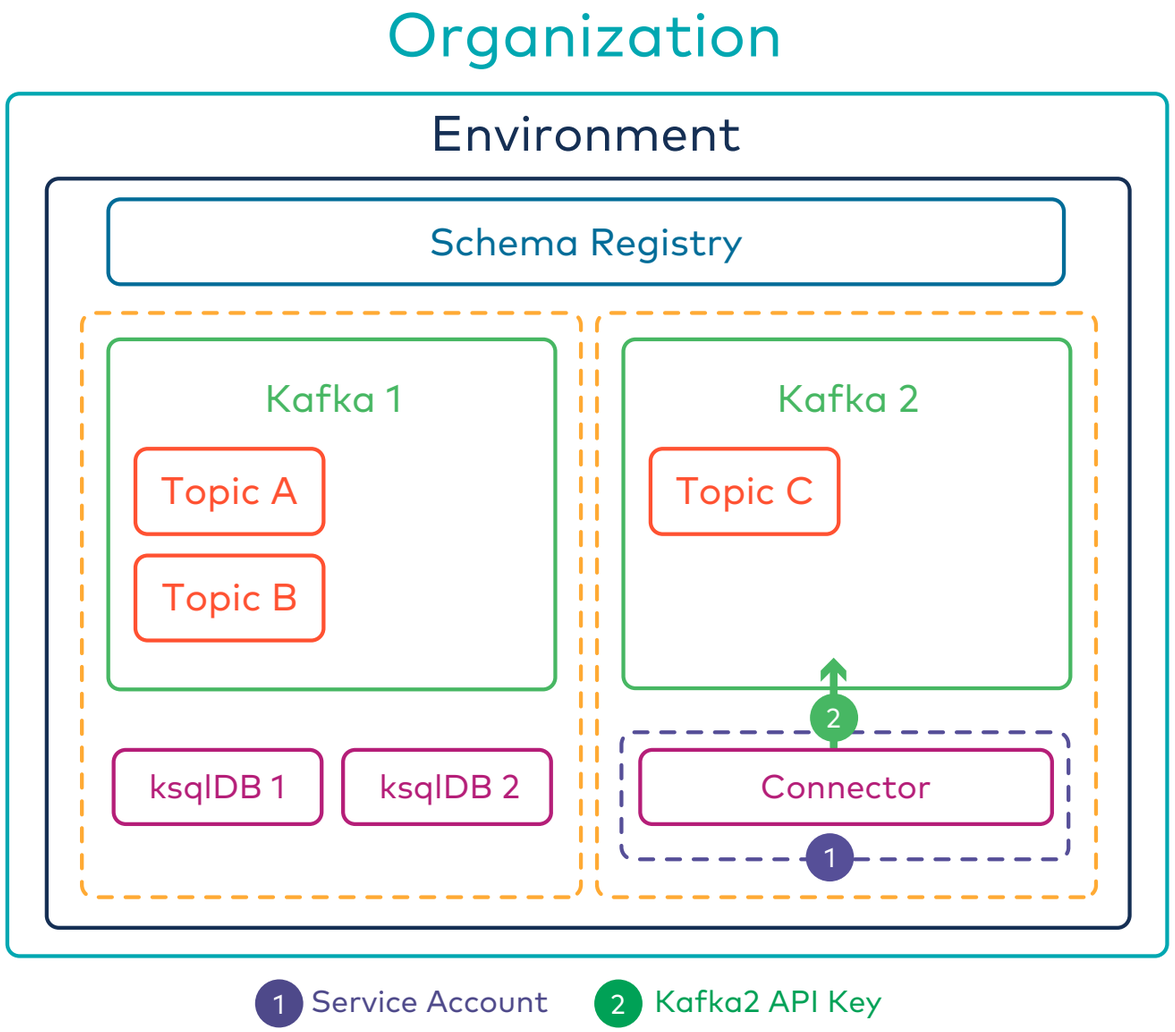
API Keys - Connector (1)

Question:

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



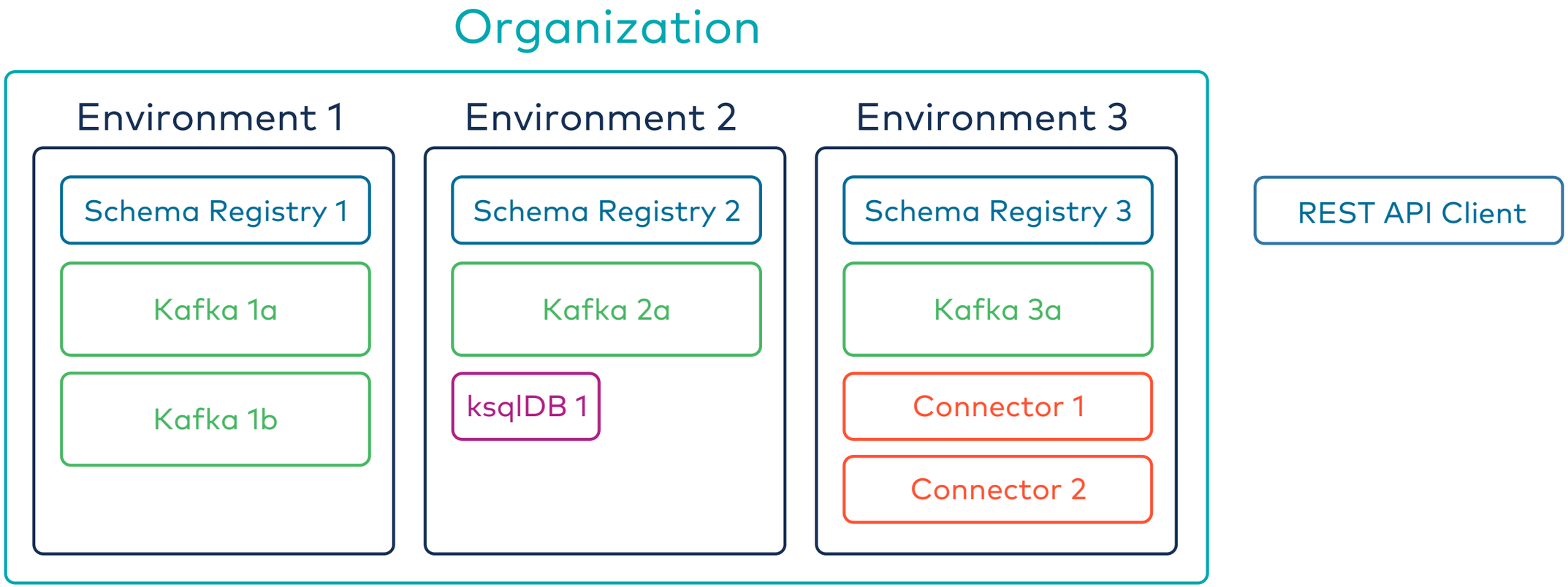
API Keys - Connector (2)



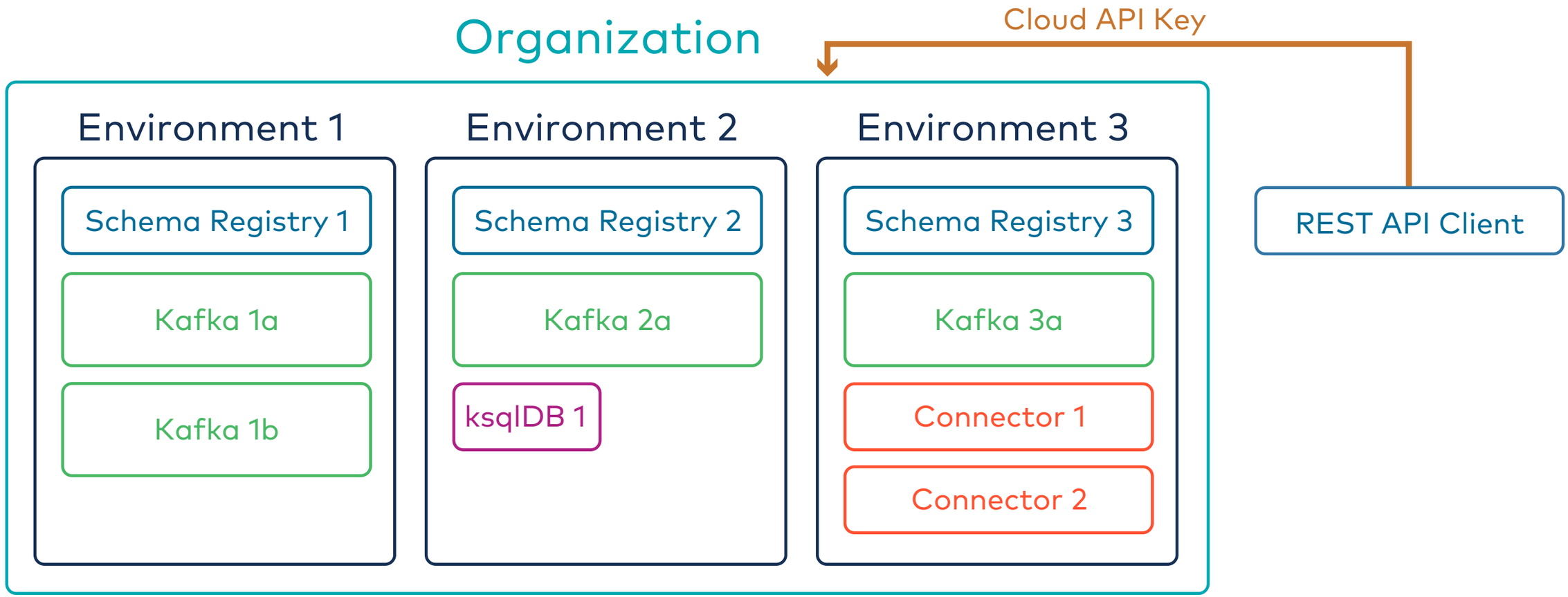
API Keys - Cloud API Key (1)

Question:

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



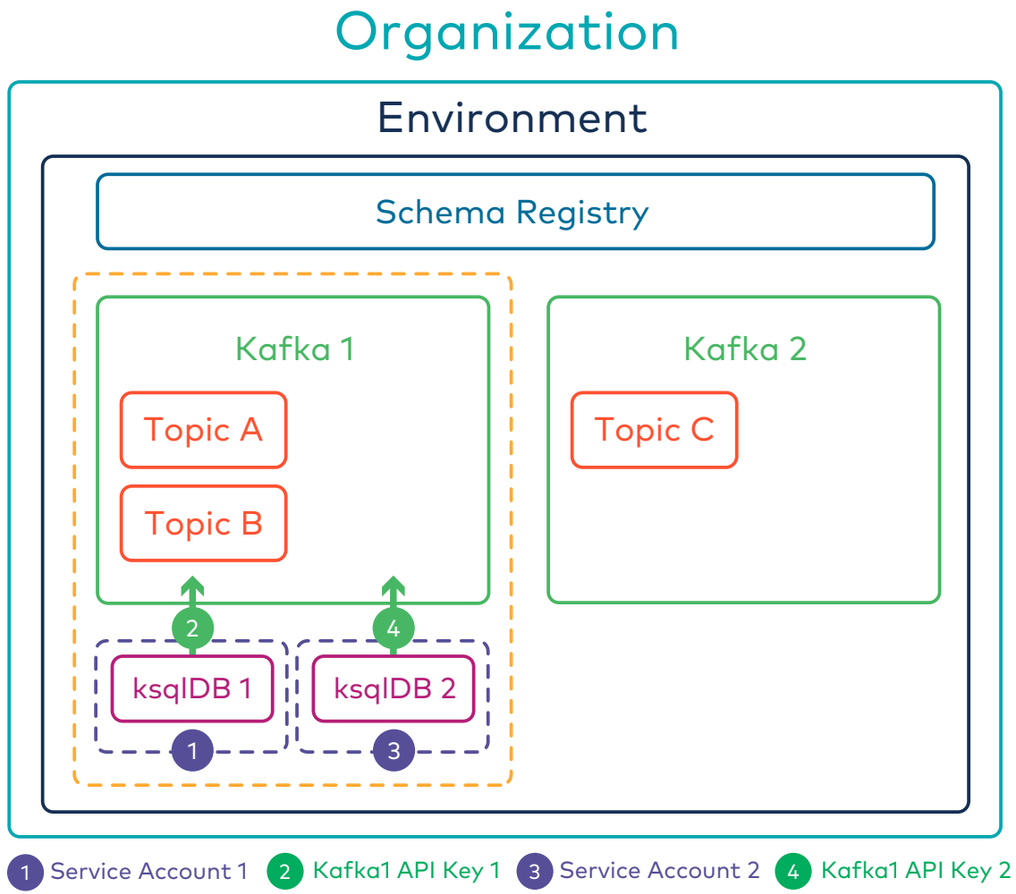
API Keys - Cloud API Key (2)



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.



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 ESP4OMLMFQRNEXOC
$ 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

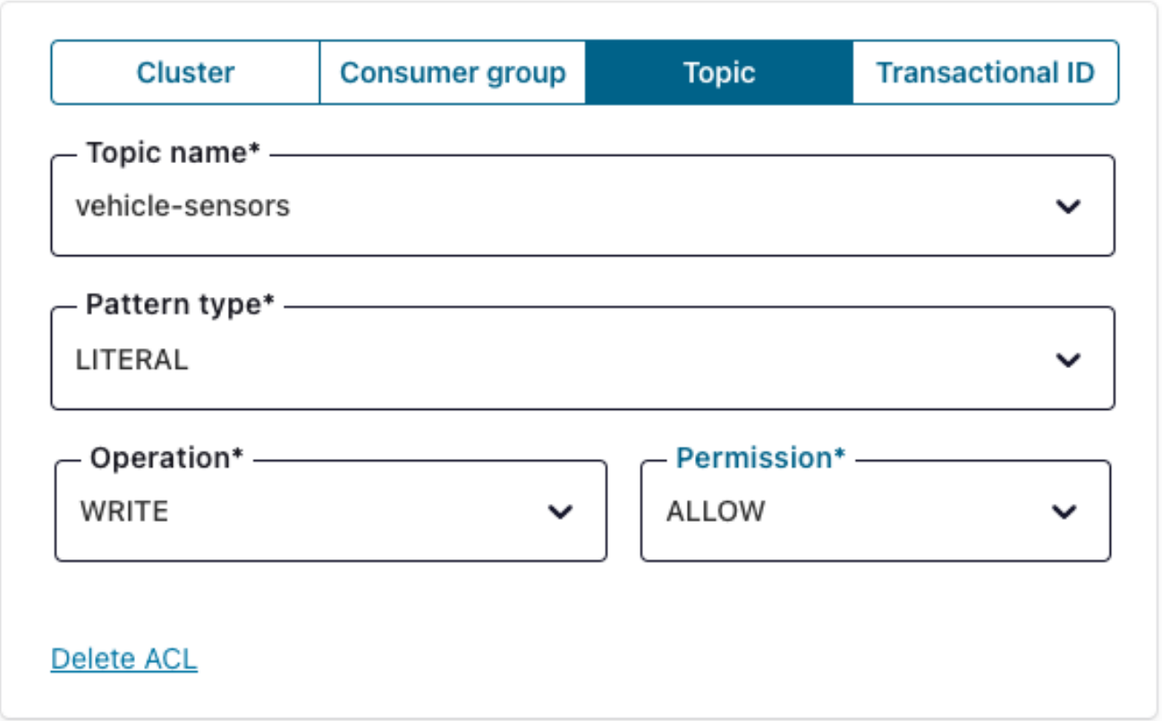
Role-Based Access Control (RBAC)

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** adn **Service** accounts
- Prefix matching supported

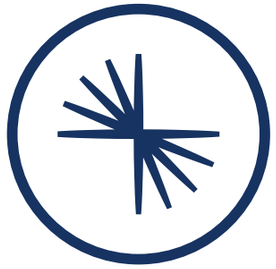


Lab Module 02: Accessing Confluent Cloud

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

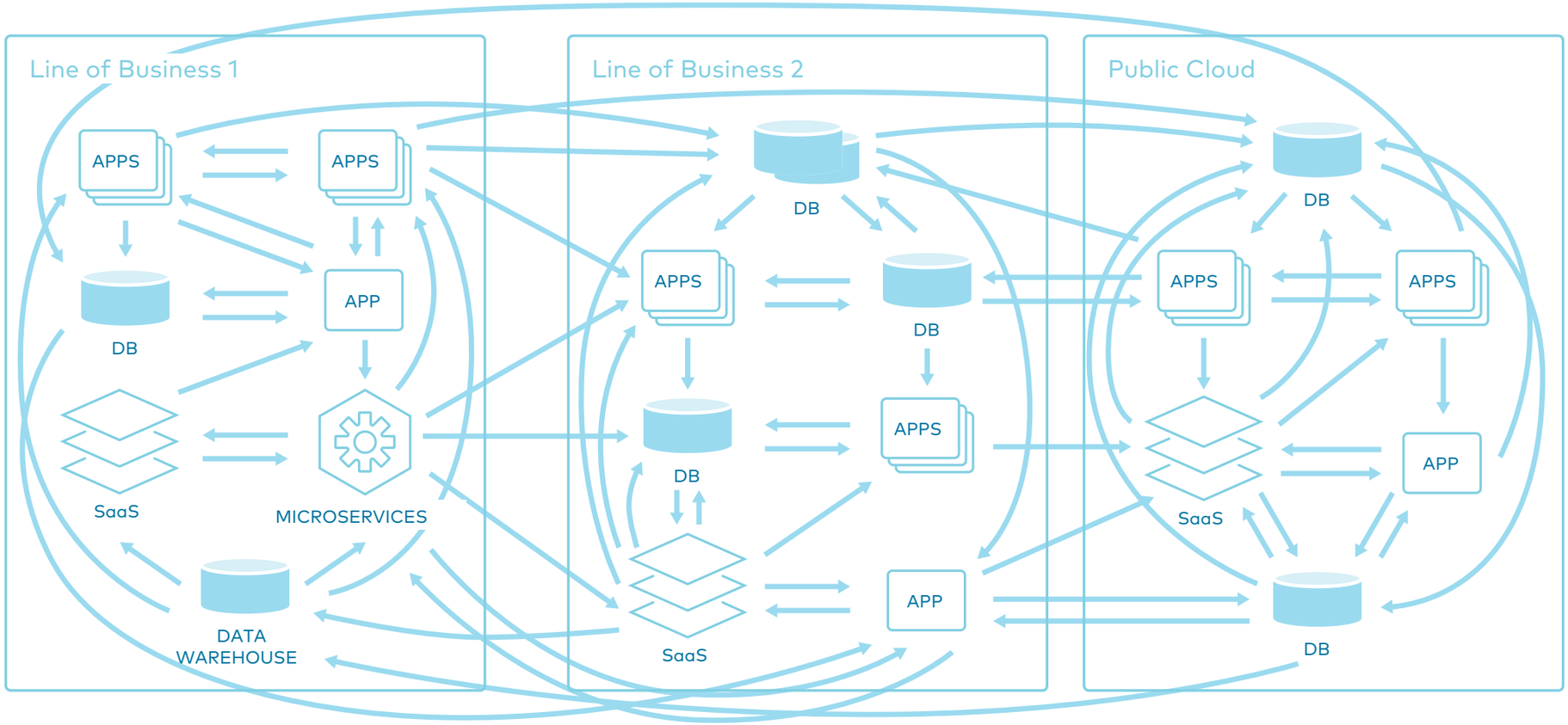
Learn why Stream Governance is important in Confluent Cloud environments.

Learning Objectives

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

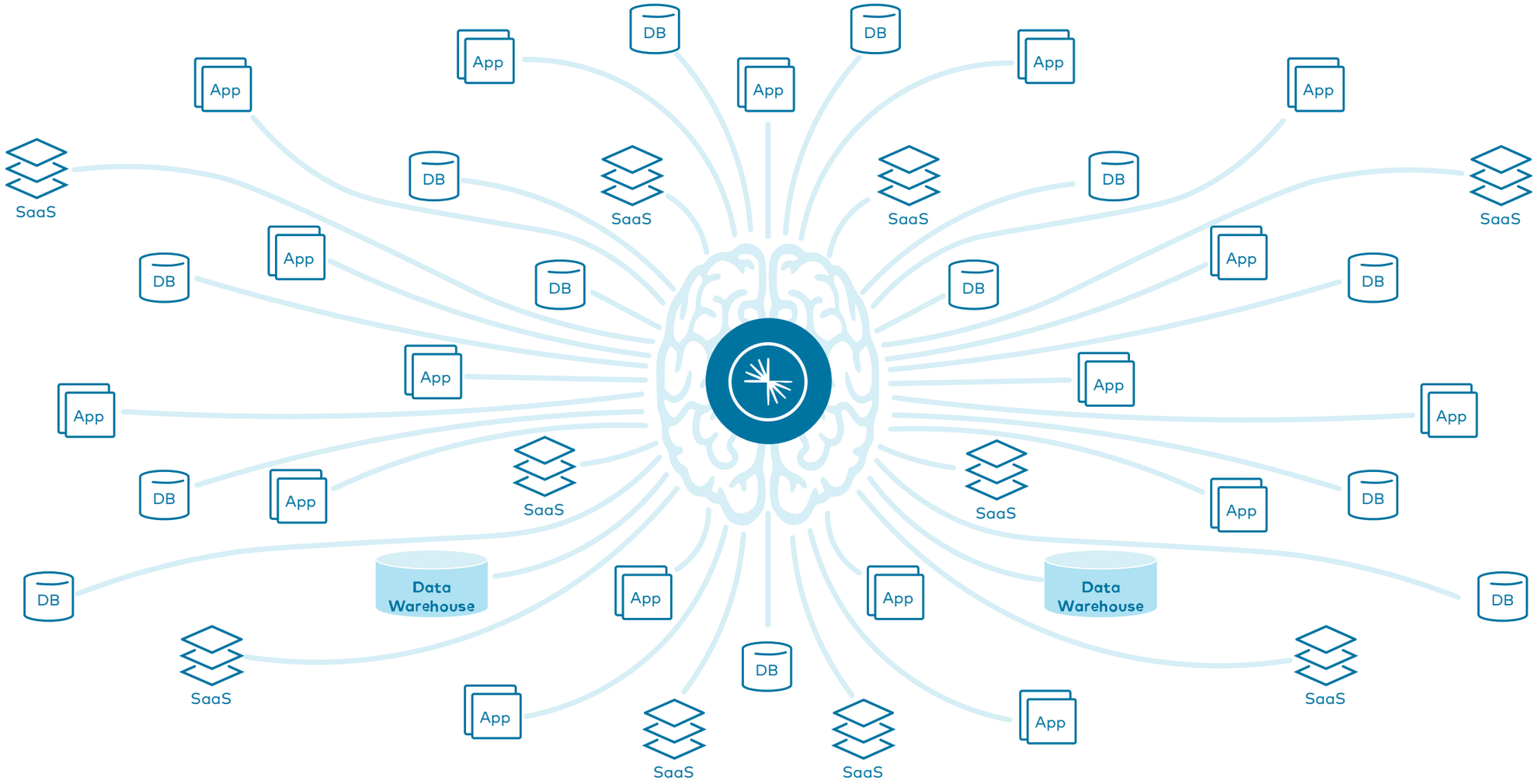
- Motivation for using Stream Governance in Confluent Cloud.

Stream Governance - Data at Rest



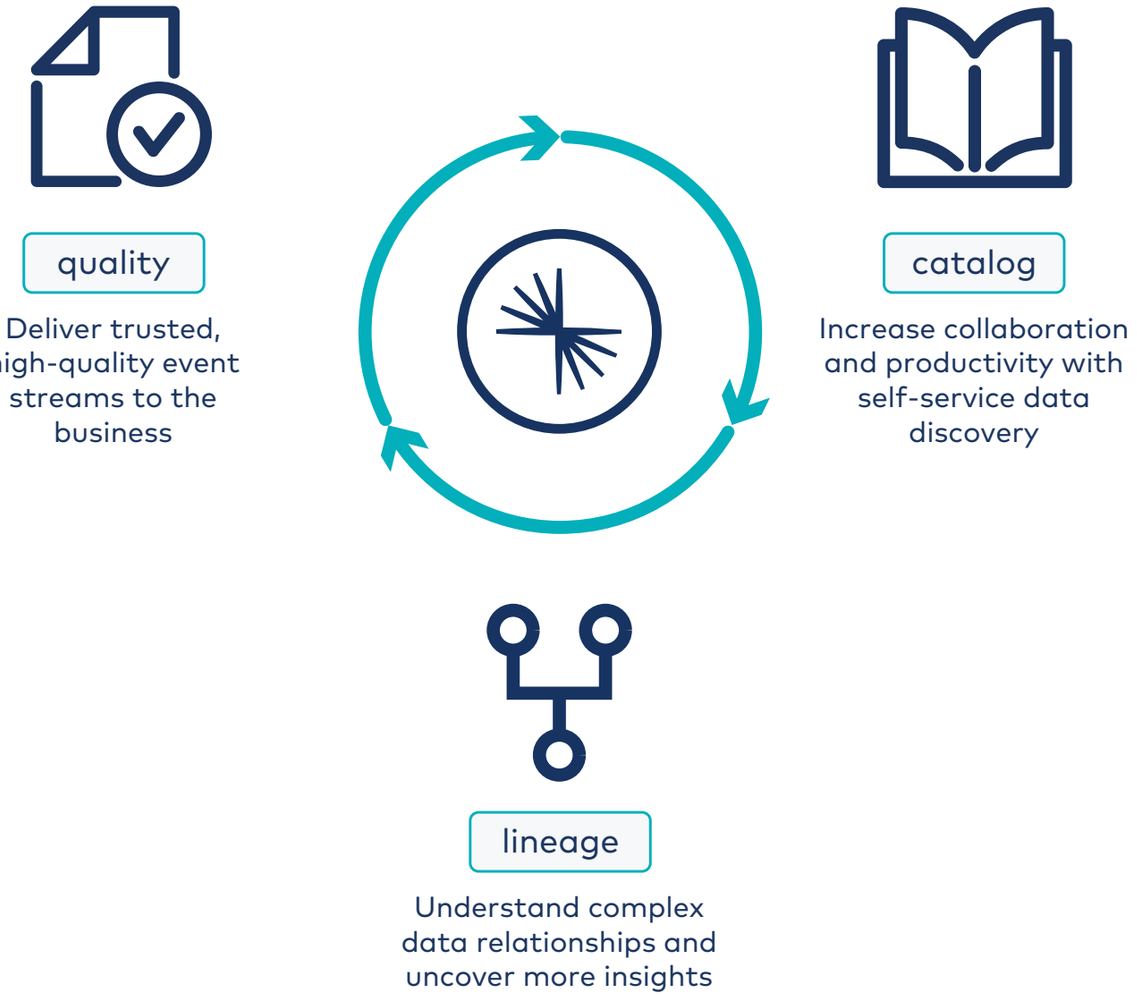
Stream Governance - Data in Motion

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

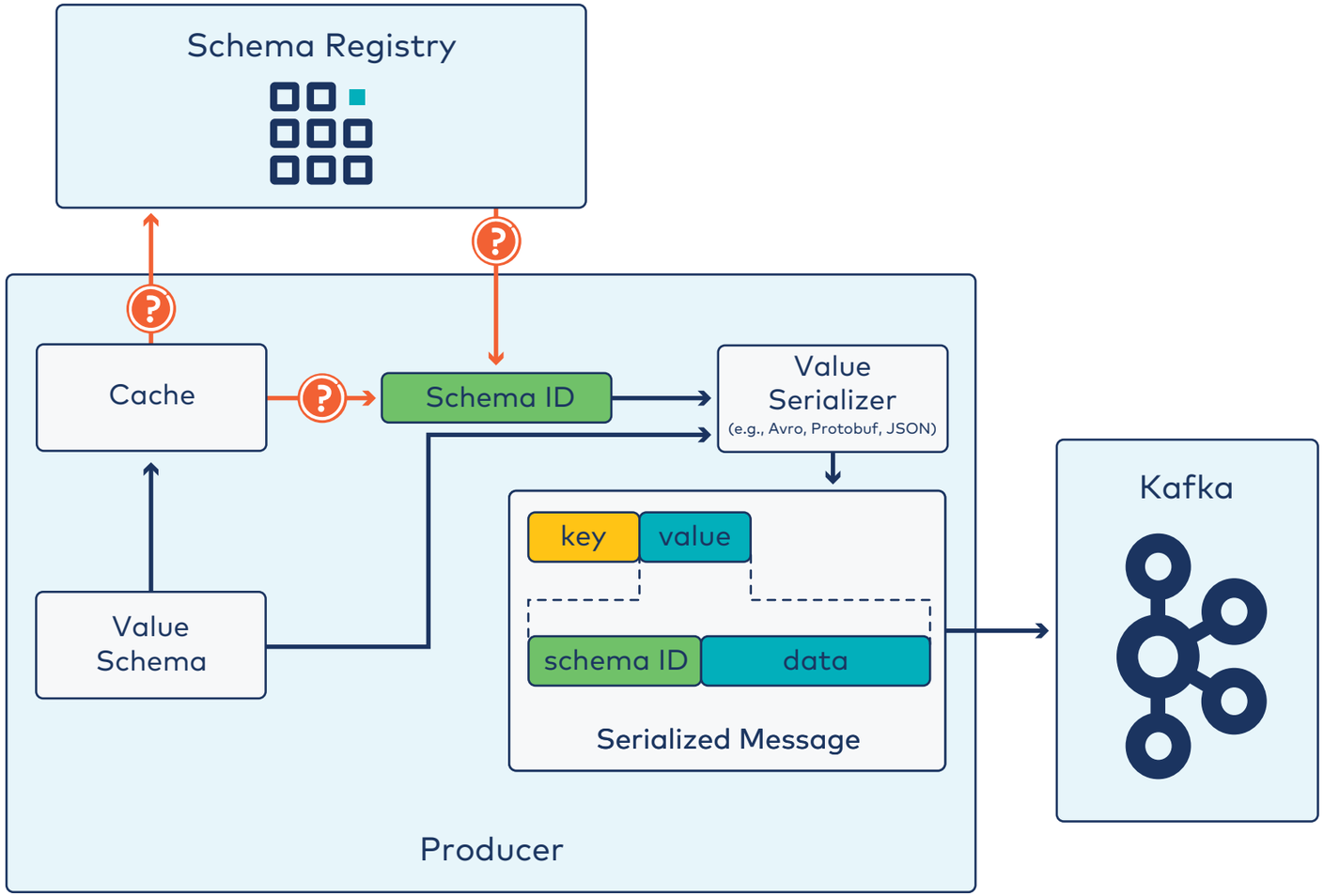
Upon completion of this lesson and associated lab exercises, you will be able to:

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

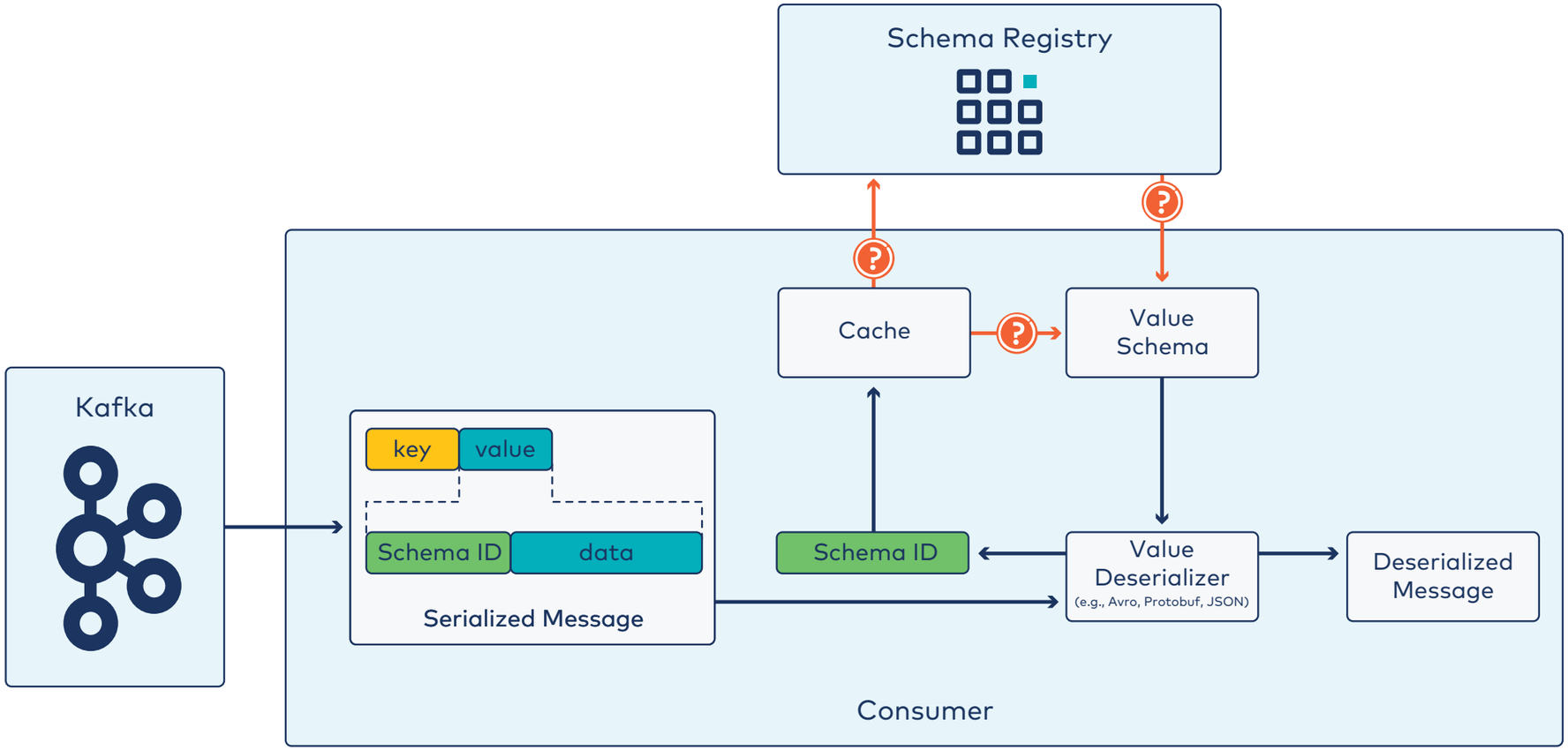
Schema Registry

- 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

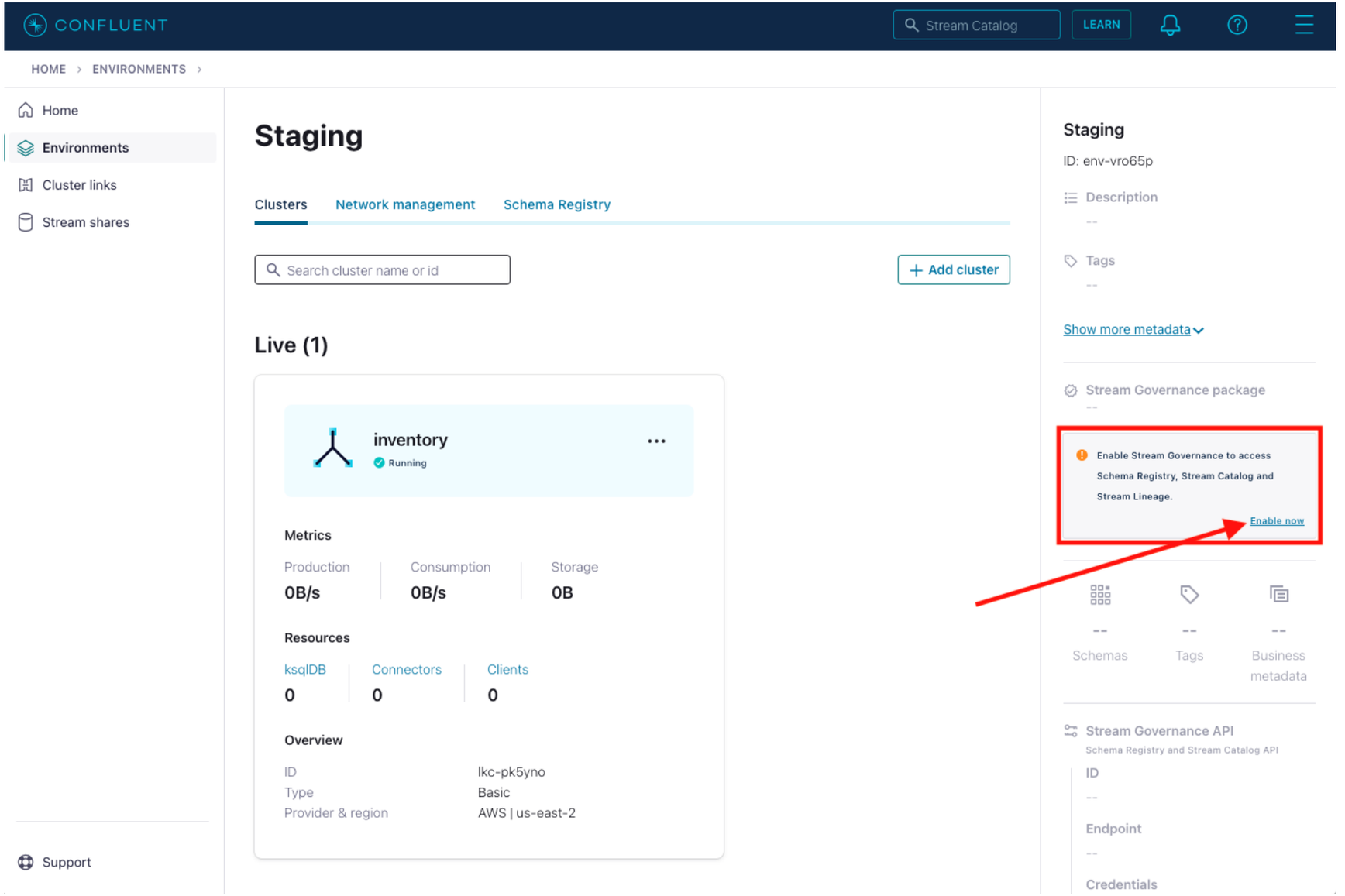
Keep key schema complexity to a minimum to guarantee deterministic serialization and not break topic partitioning.

Key	Value
• Simple non-serialized data type (string, UUID, long ID, etc.) • An Avro record that does not include maps or arrays	• Avro • Protobuf • JSON Schema

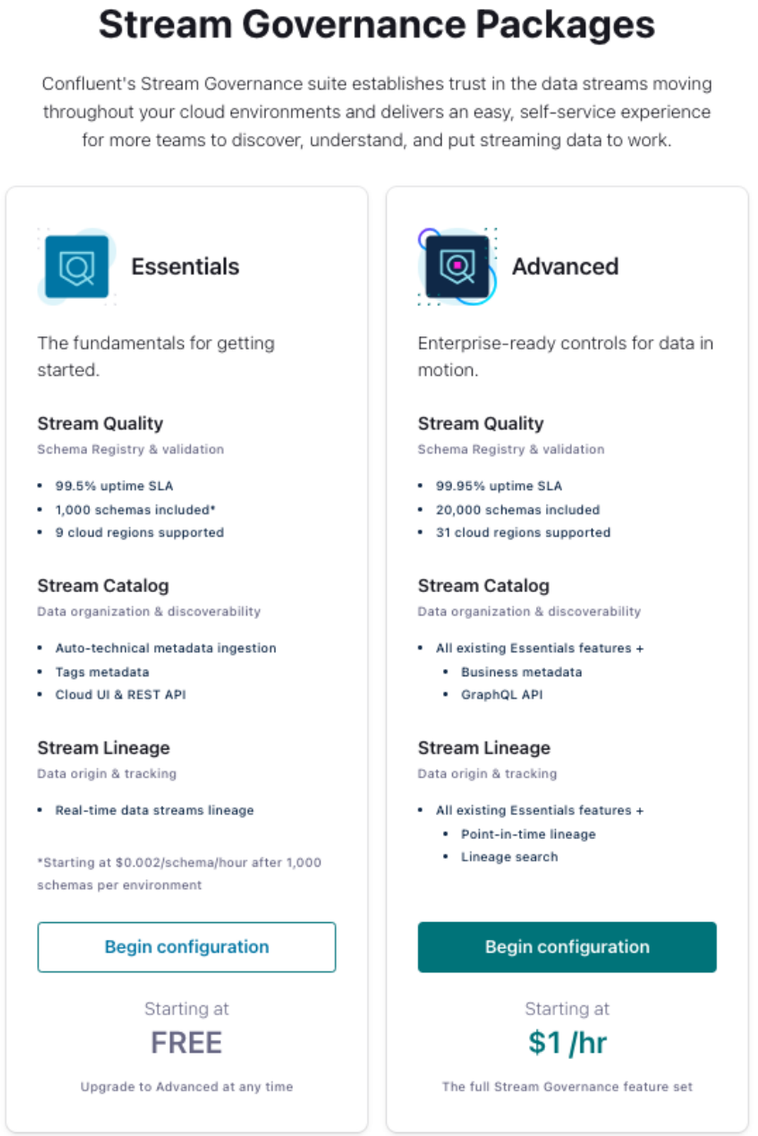
Limits

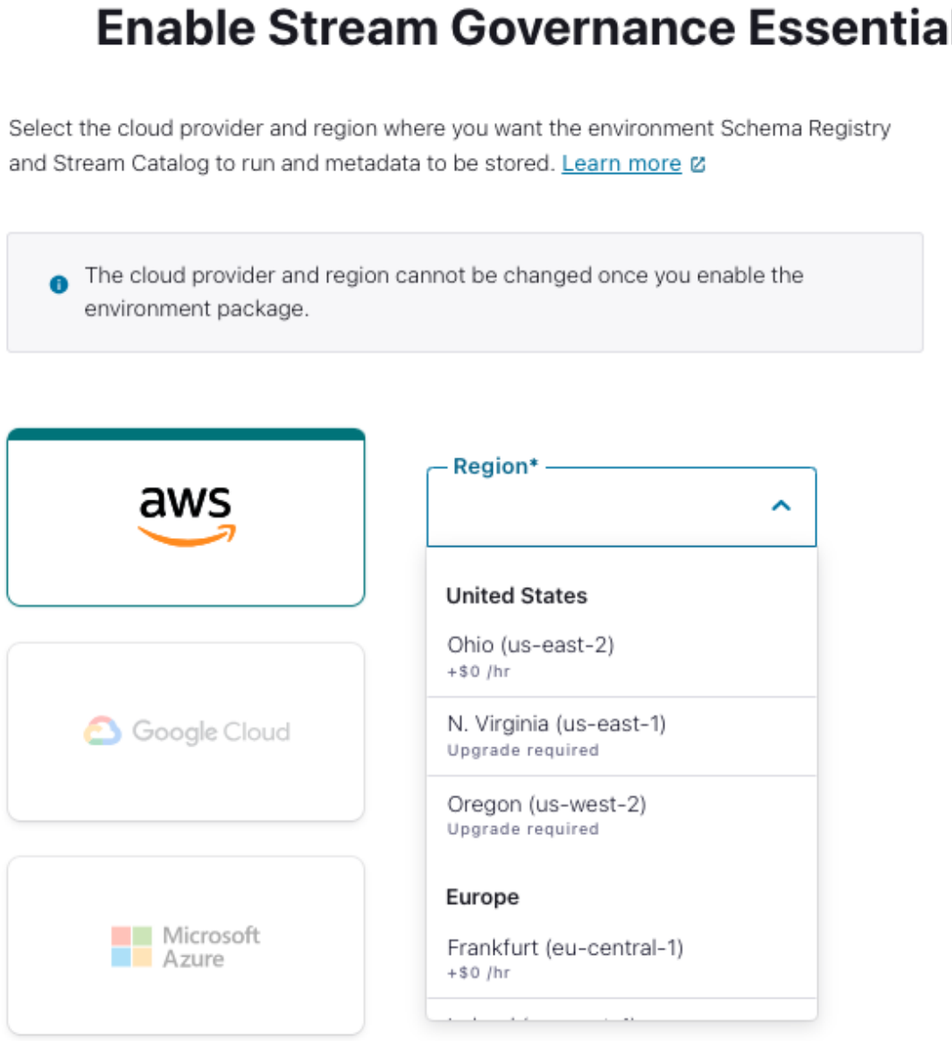
- 1,000 free schemas per environment.
- Rate limits of API requests:
  - 25 requests per second for each API key.

Enable Schema Registry (I)

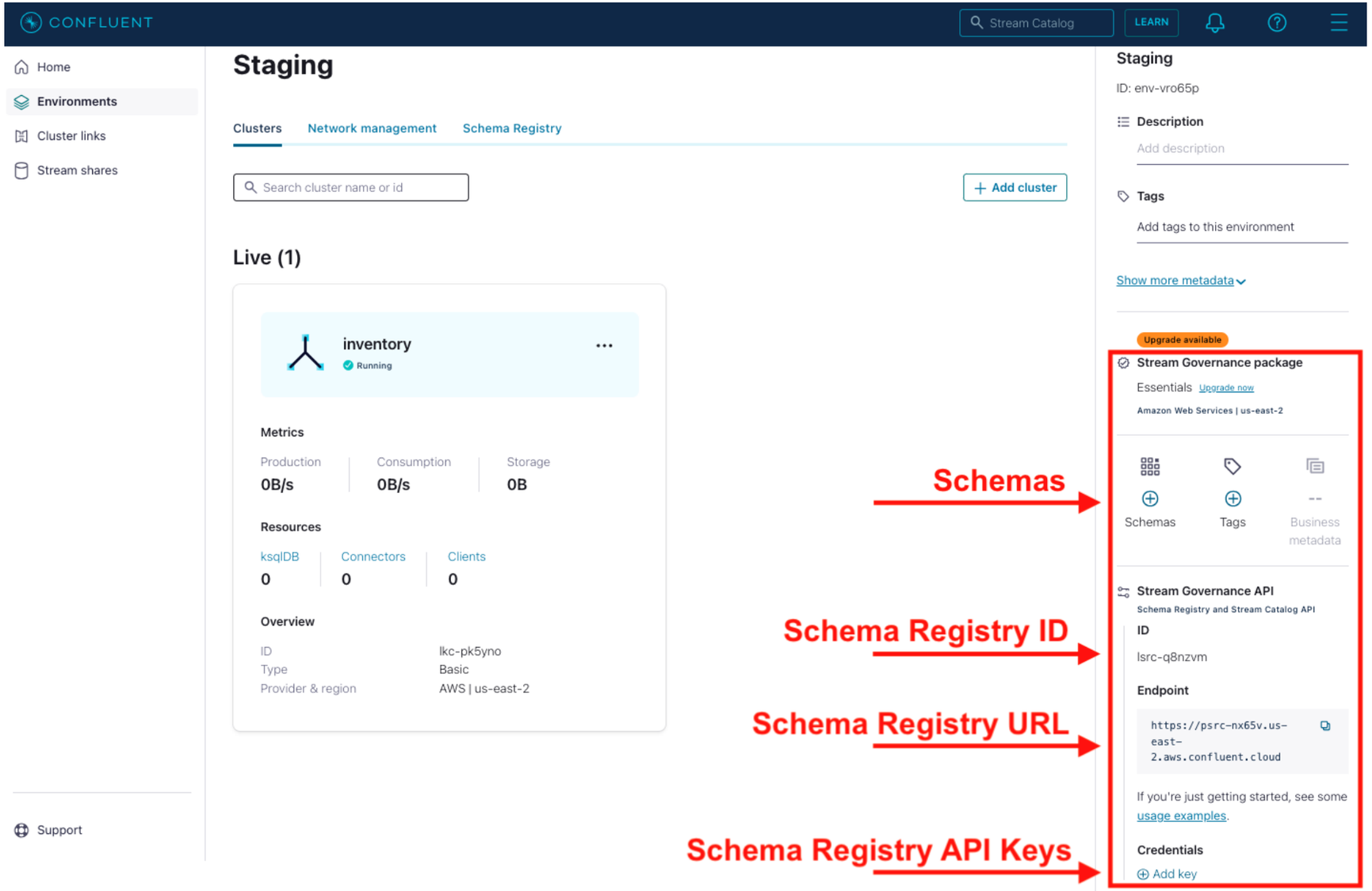


Enable Schema Registry (II)





Enable Schema Registry (III)



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
```

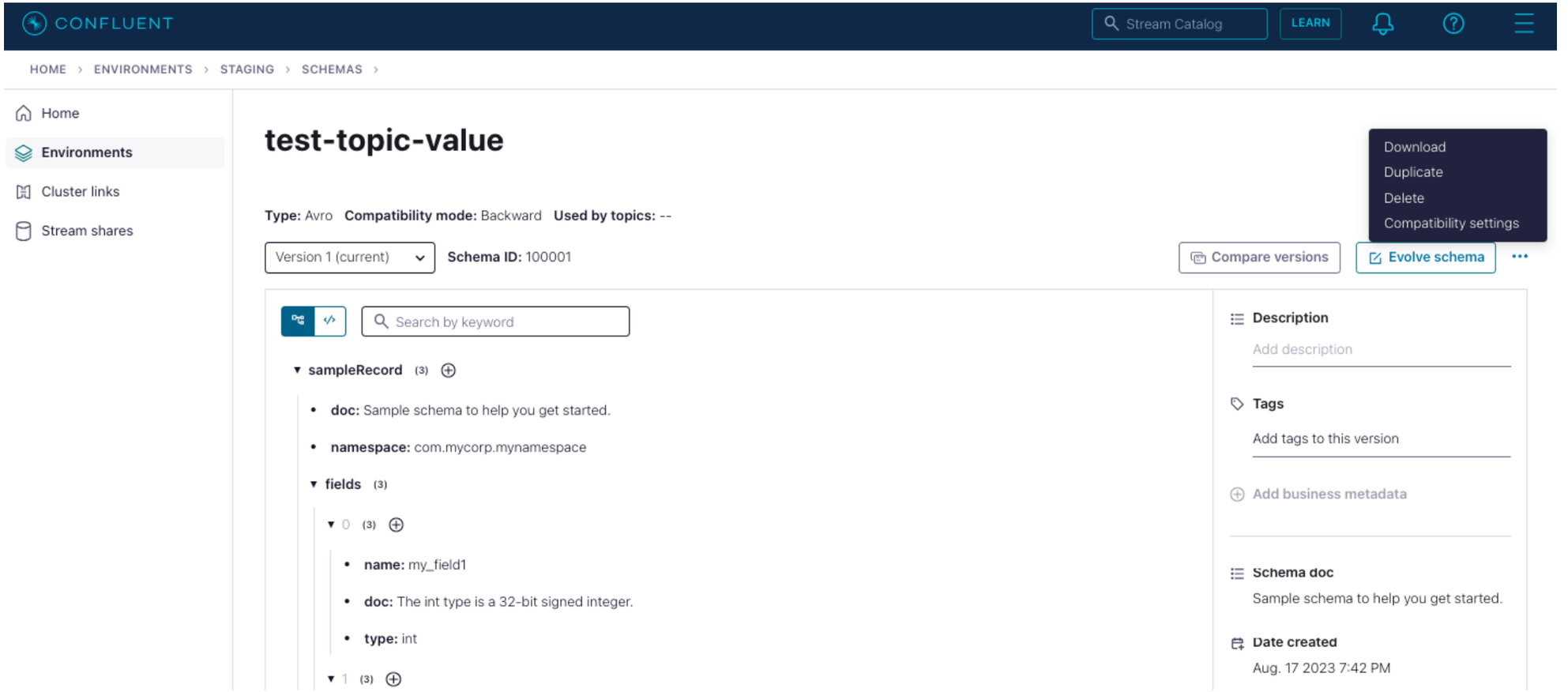
Retrieve a schema by ID:

```
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



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	`order-topic-value`
Key	`order-key`
Value	`order-value`

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
`confluent.key.schema.validation`	When set to `true`, enables schema ID validation on the message key	`false`
`confluent.value.schema.validation`	When set to `true`, enables schema ID validation on the message value	`false`
`confluent.key.subject.name.strategy`	The subject name strategy for the message key	`TopicNameStrategy`
`confluent.value.subject.name.strategy`	The subject name strategy for the message value.	`TopicNameStrategy`

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:

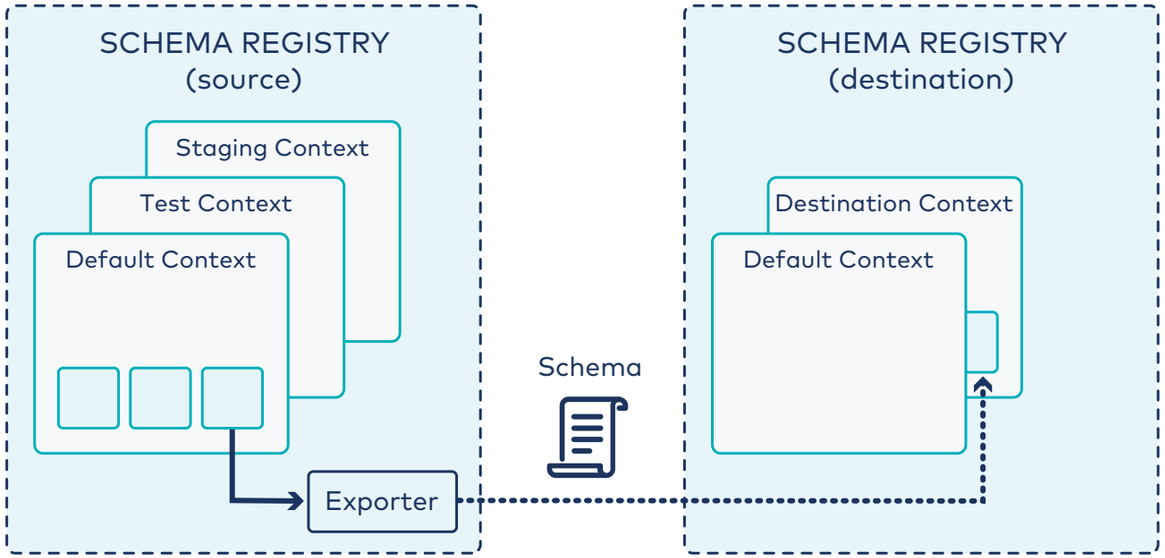
- Definition of schema linking and its components.
- Explanation about schema contexts, how to define them and how to access them using REST API and Kafka clients.
- Description of schema exporters and instructions to manage them.

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)

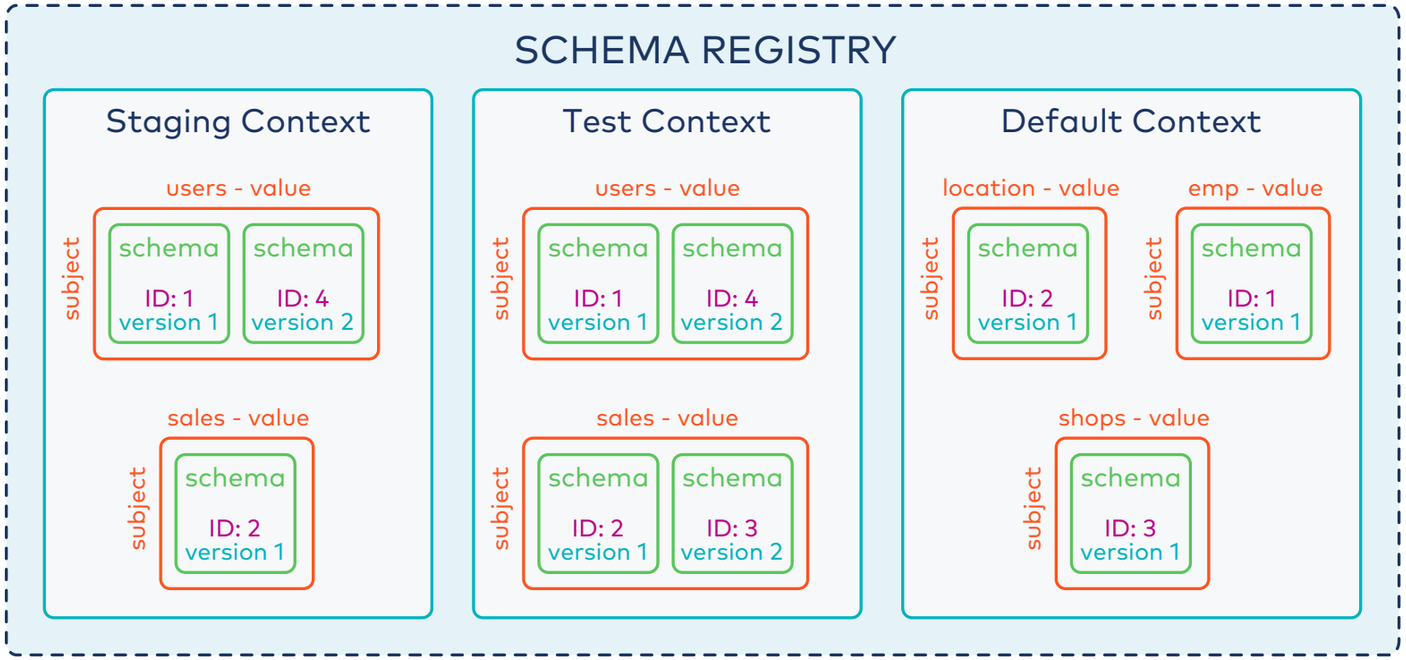
- A schema context is essentially a grouping of subject names and schema IDs.
- A single Schema Registry cluster can host any number of contexts.
- A context can be copied to another Schema Registry cluster, using a schema exporter.
- There is a default context called Default.

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.

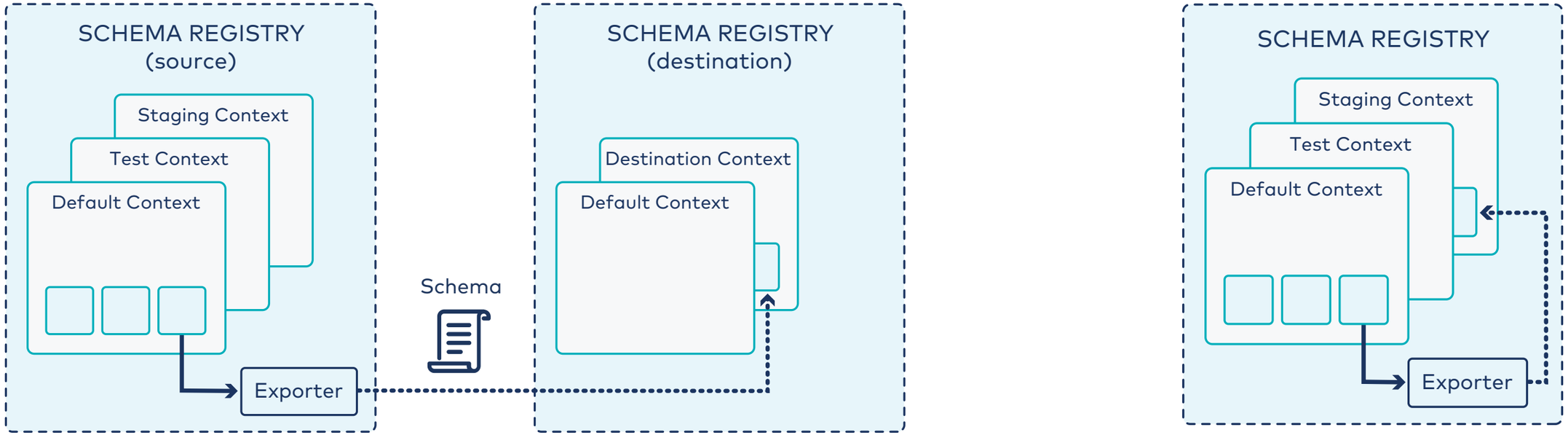
Schema Contexts - Qualified Subject Name

```
:.<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



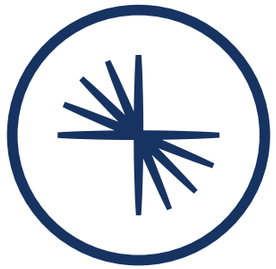
Maximum 10 exporters per Schema Registry.

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 orgins of the data.

Learning Objectives

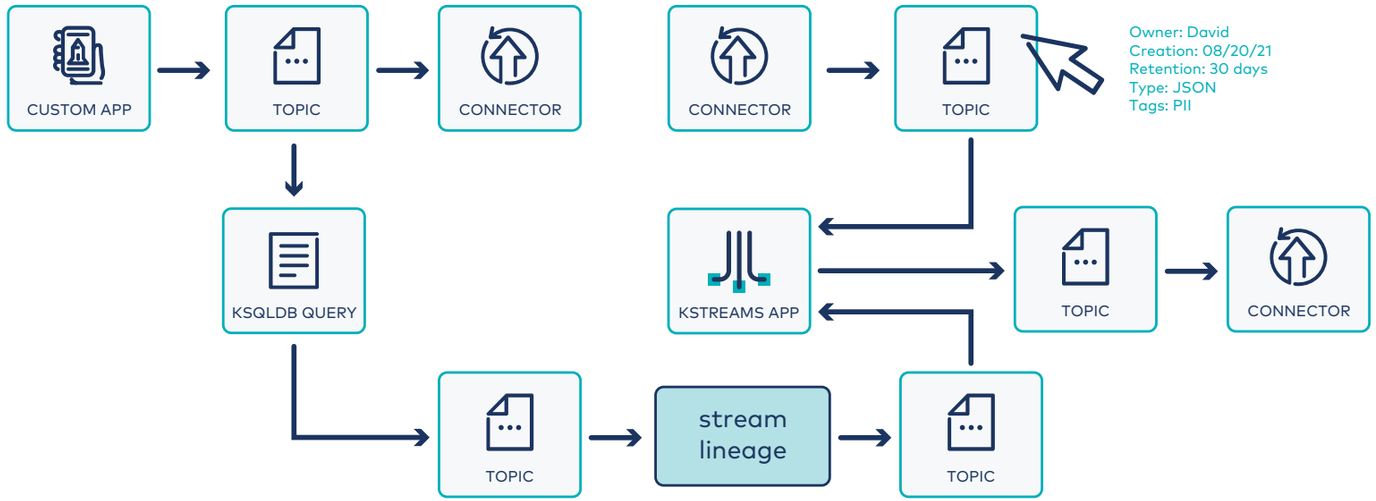
Upon completion of this lesson and associated lab exercises, you will be able to:

- 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

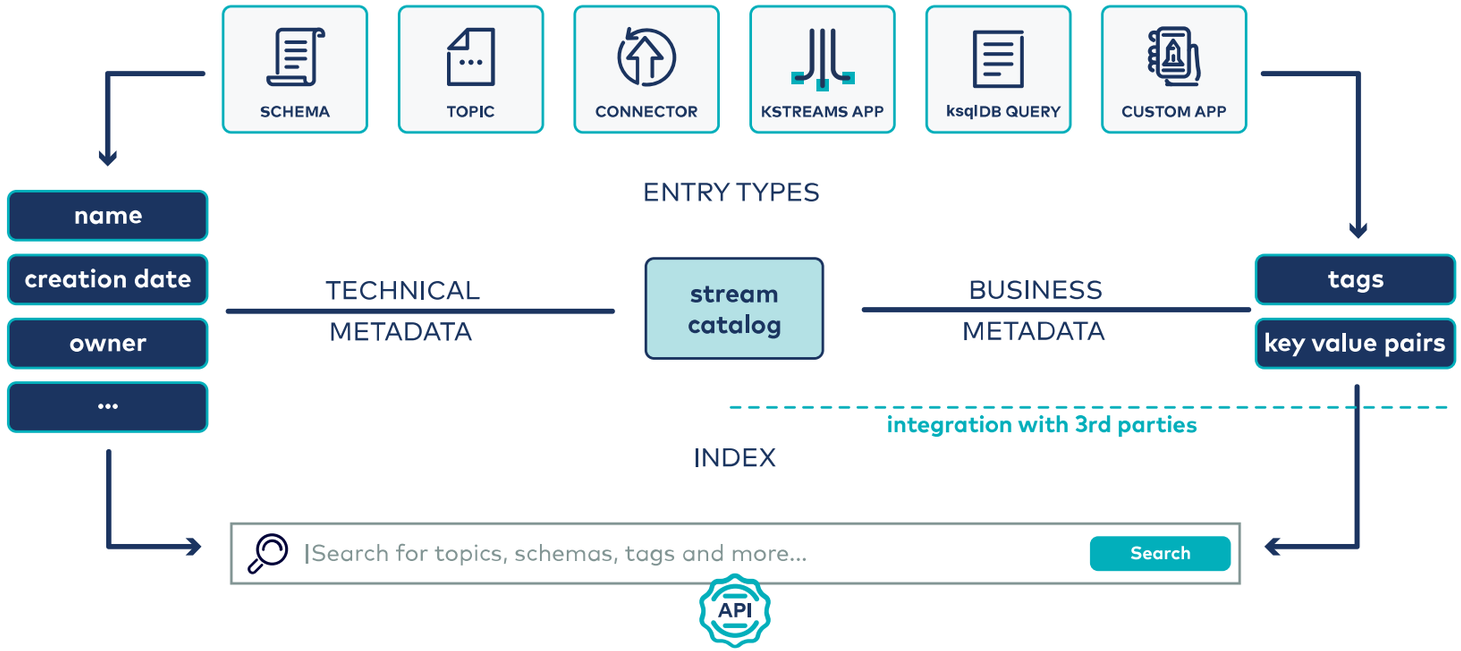
Upon completion of this lesson and associated lab exercises, you will be able to:

- 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:

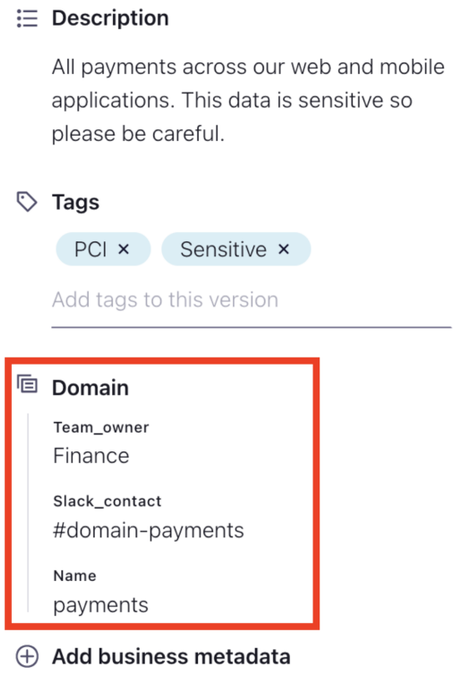
- Tags can be applied to Environments, Topics, Connectors, Schema Subjects, Schema Records, Schema Fields
- Multiple tags can be associated to the same element
- Use provided tags (Public, Private, Sensitive, PII) and custom tags
  - Custom tag:
    - Tag name
    - Description

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:

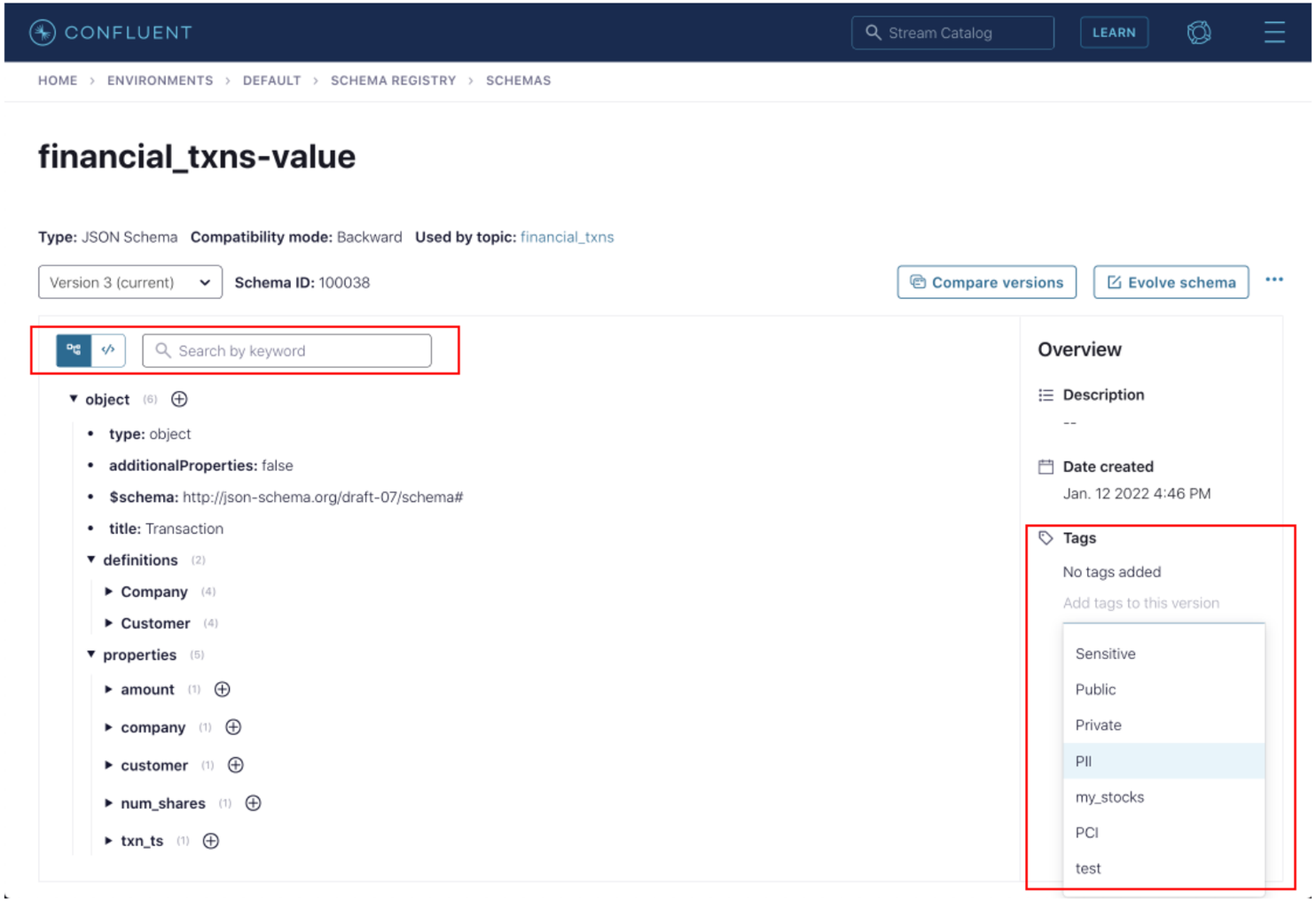
- Which team is responsible for a particular schema?
- Which product domain does a schema belong to?
- What is the GitHub location for a schema?



- 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.

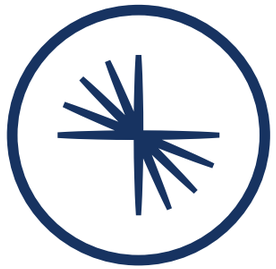


Stream Governance - Catalog - REST API

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 modules 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.

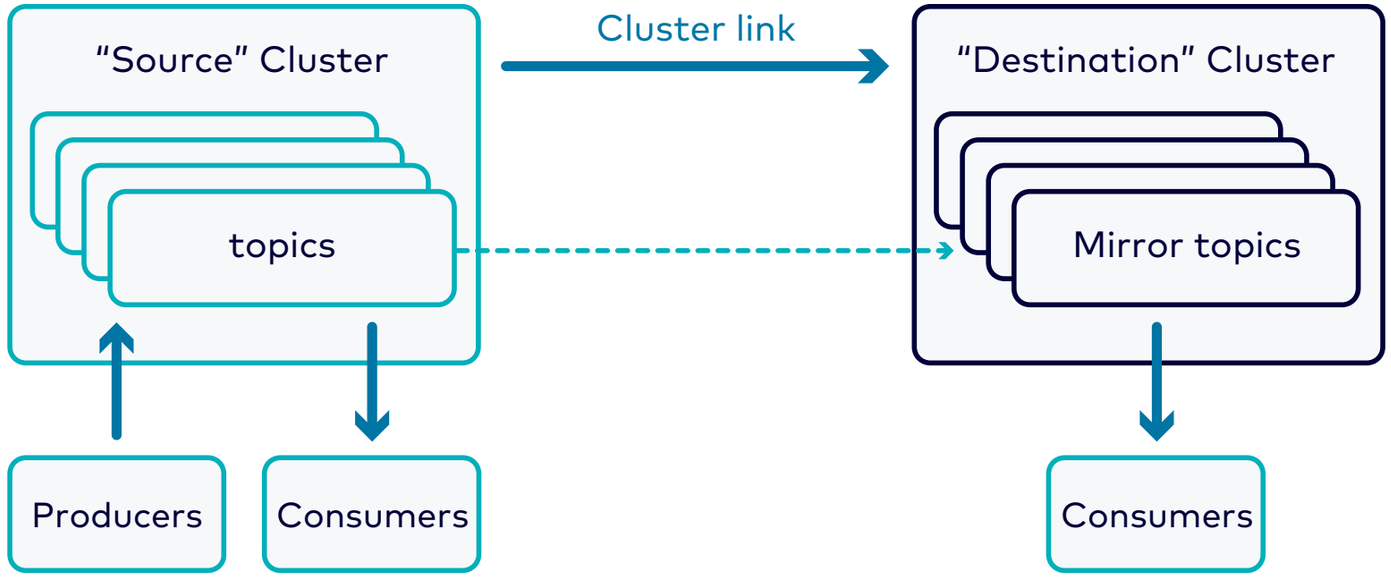
Cluster Linking Overview

- 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.

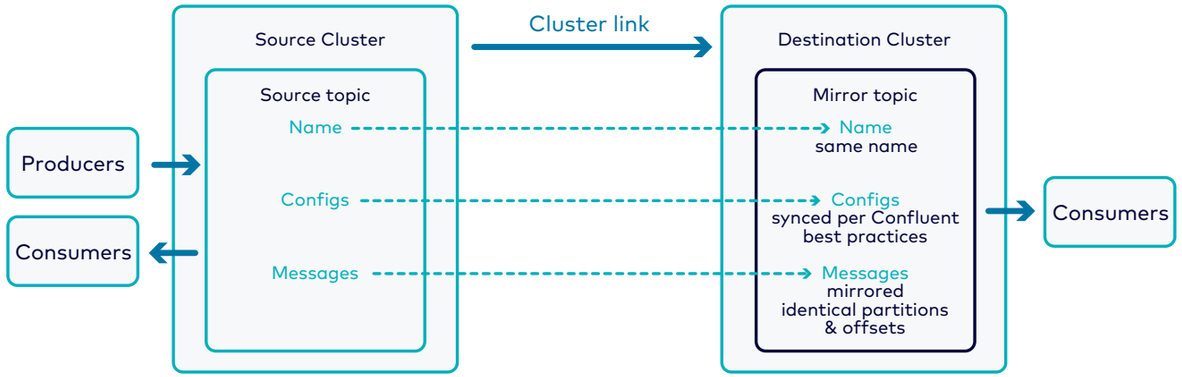
Example of the creation of a Cluster Link:

```
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
• Any Confluent Cloud cluster • Confluent Platform 5.4 • Apache Kafka 2.4+	• 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

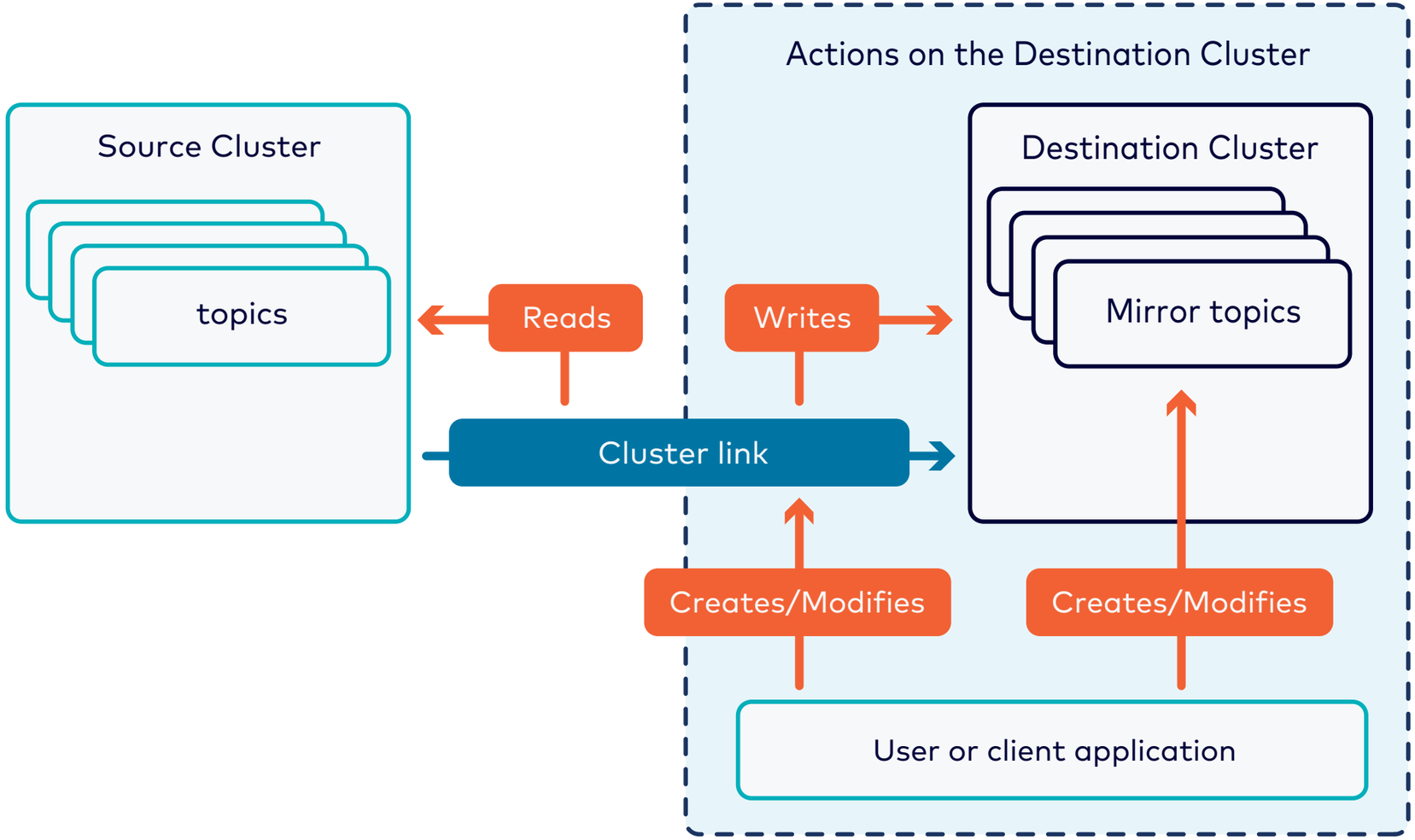
An overview of cluster linking security considerations.

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 *cluster link*	Destination	• CloudClusterAdmin • EnvAdmin • OrgAdmin	• ALTER: Cluster
• User • Client app	Create or modify a *mirror topic*	Destination	• CloudClusterAdmin • EnvAdmin • OrgAdmin	• ALTER: Cluster & ALTERCONFIGS: Cluster & CREATE: Topic

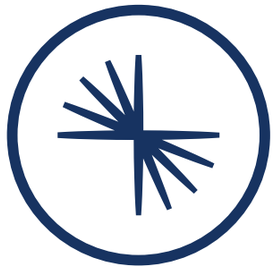
Use RBAC or ACL.

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 modules 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

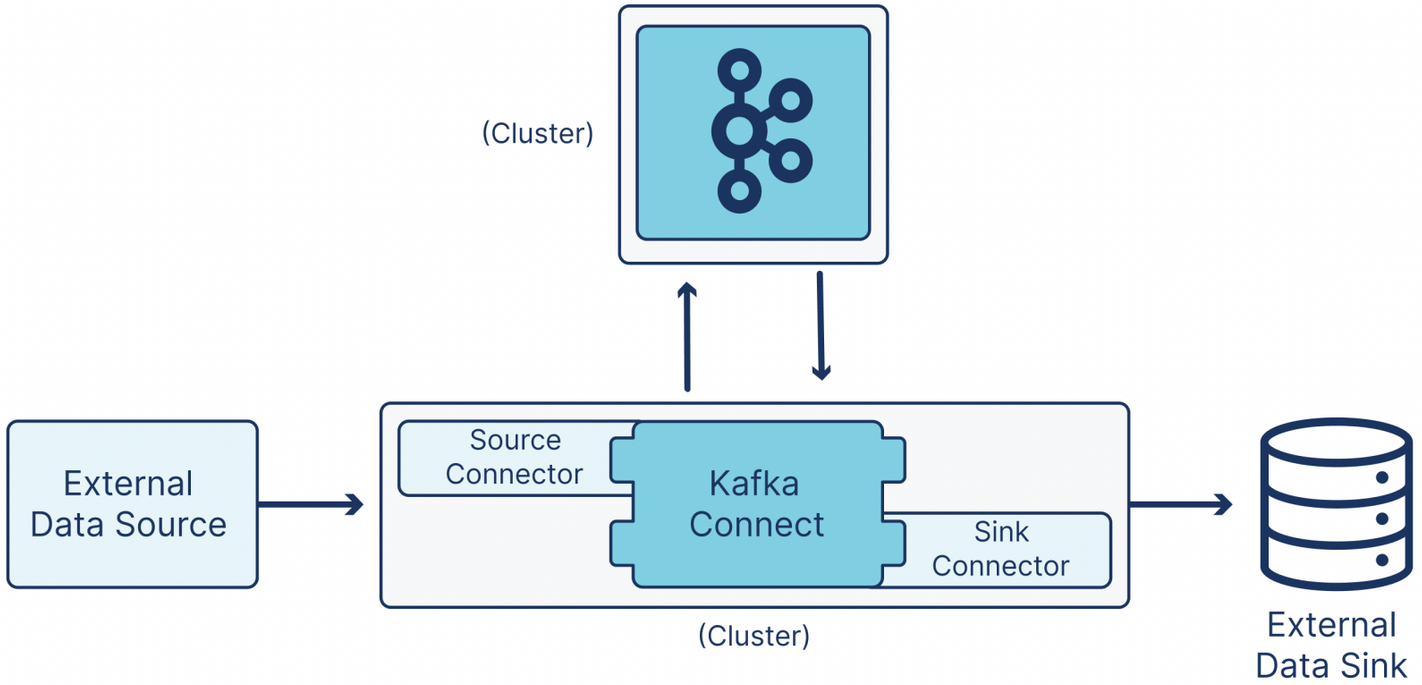
Suppose you have:

- 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

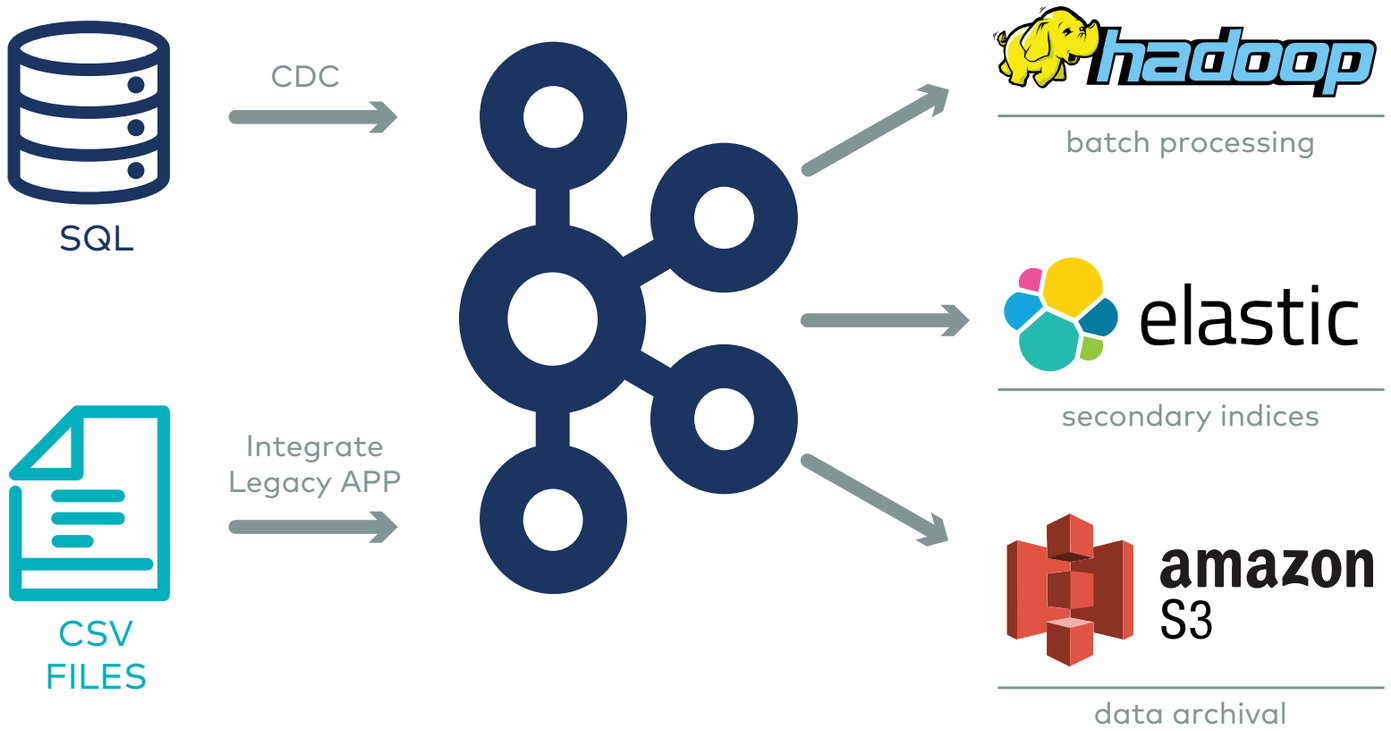
You could program custom producers or consumers with hooks into the externals systems to make this happen… But… there's a better and easier way…

Kafka Connect to the Rescue!

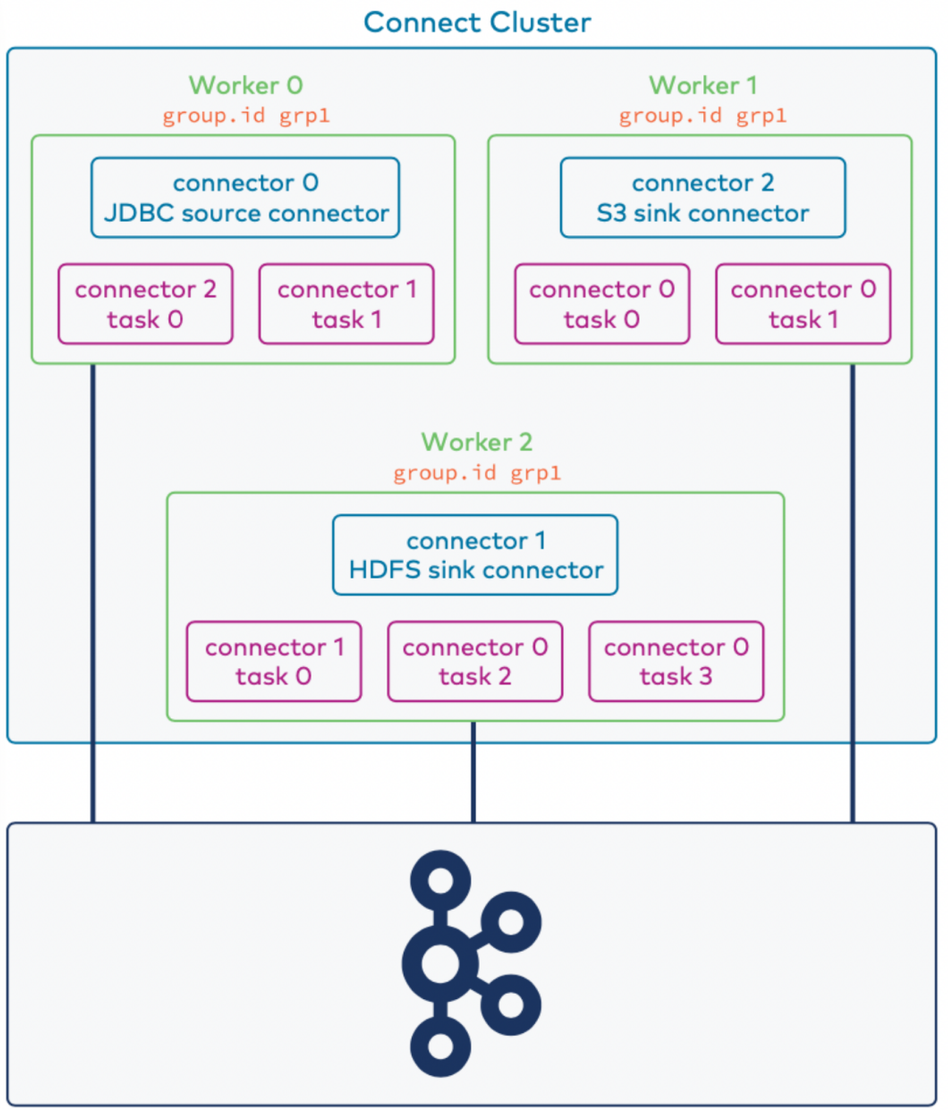
- Kafka Connect does the work for us!
- All copying behavior is in Kafka Connect.
- Plugins called connectors contain the logic specific to particular external systems.



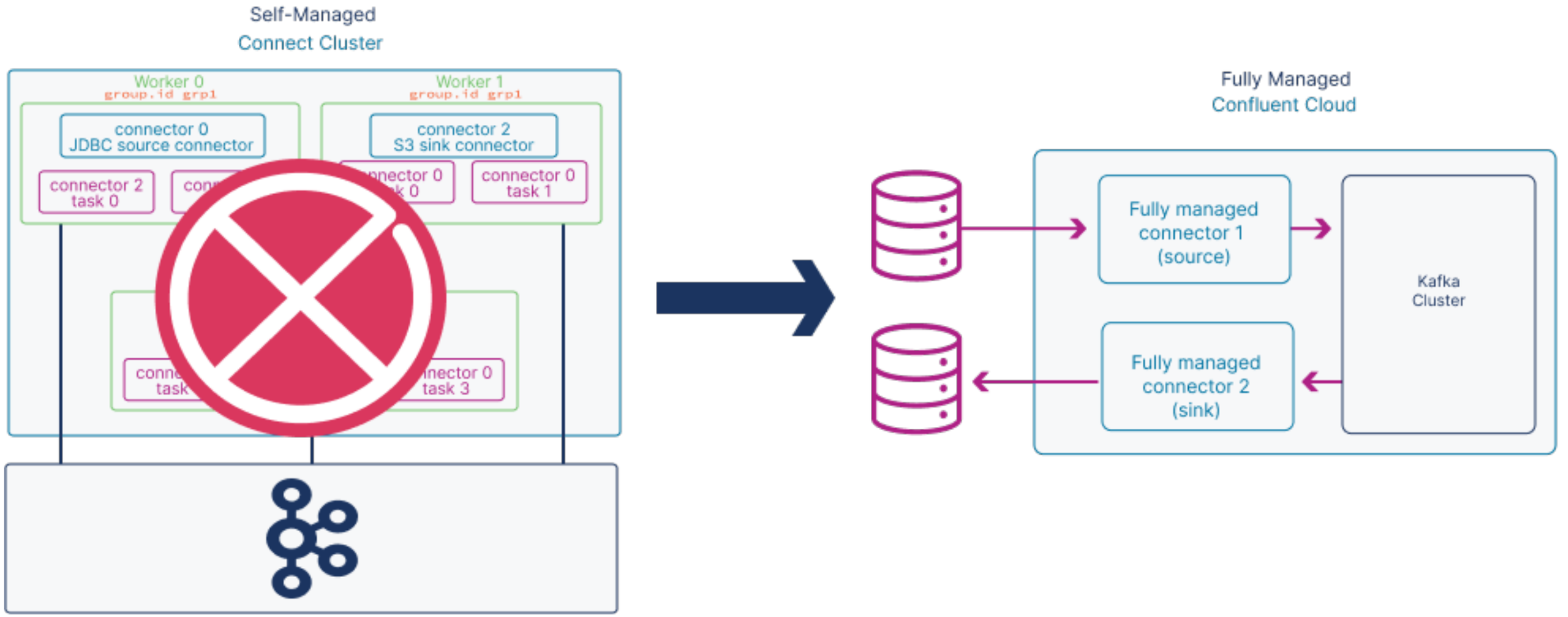
Use Cases



Connect Cluster



Fully Managed Connectors in Confluent Cloud



View the Data Preview Output

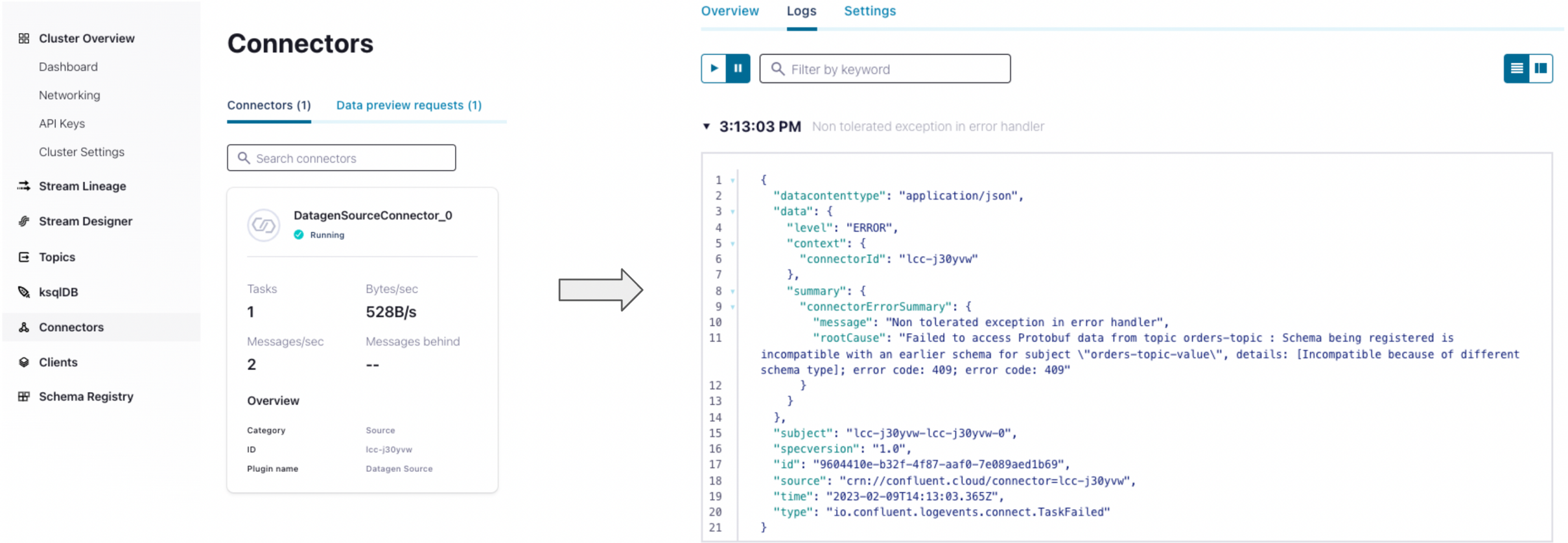
The expanded preview record has two sections:

- 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

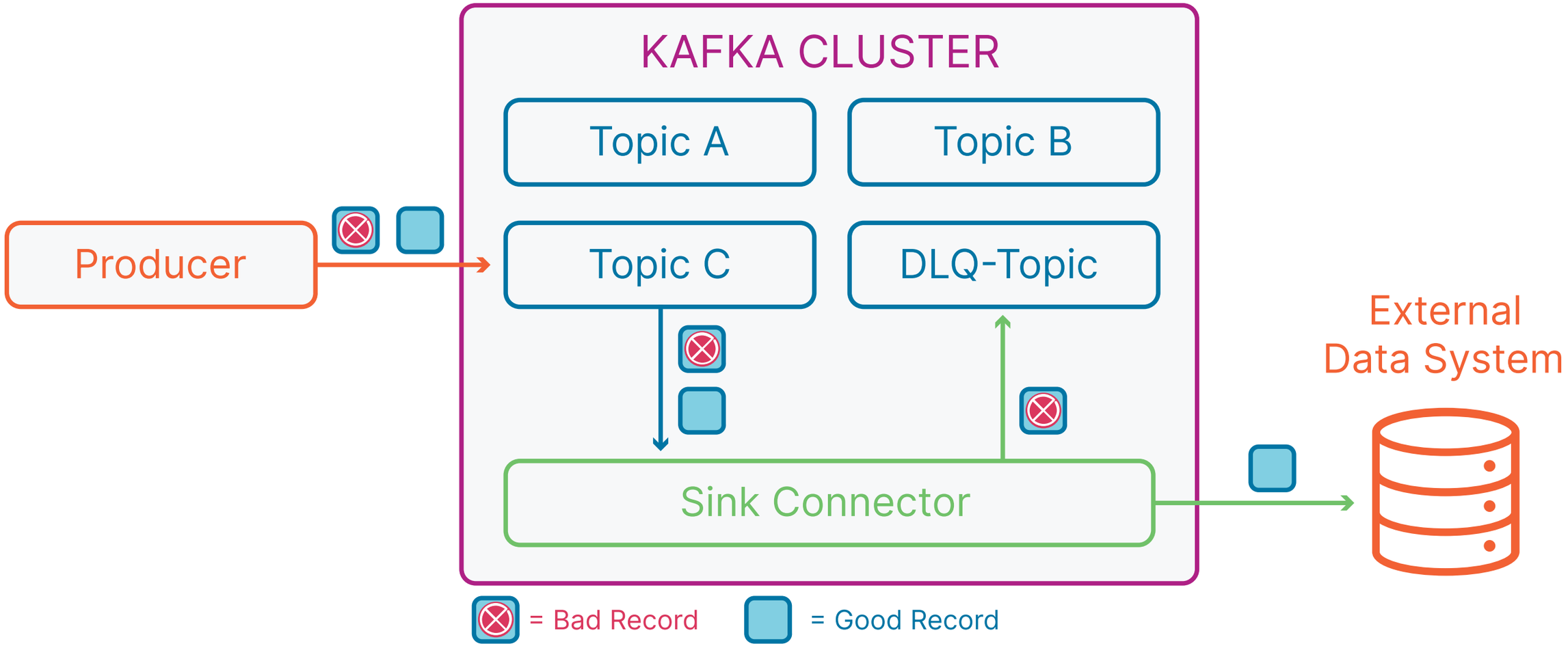
If data is not as expected:

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

View Connect Events Using Confluent Cloud UI



Dead Letter Queue (DLQ)



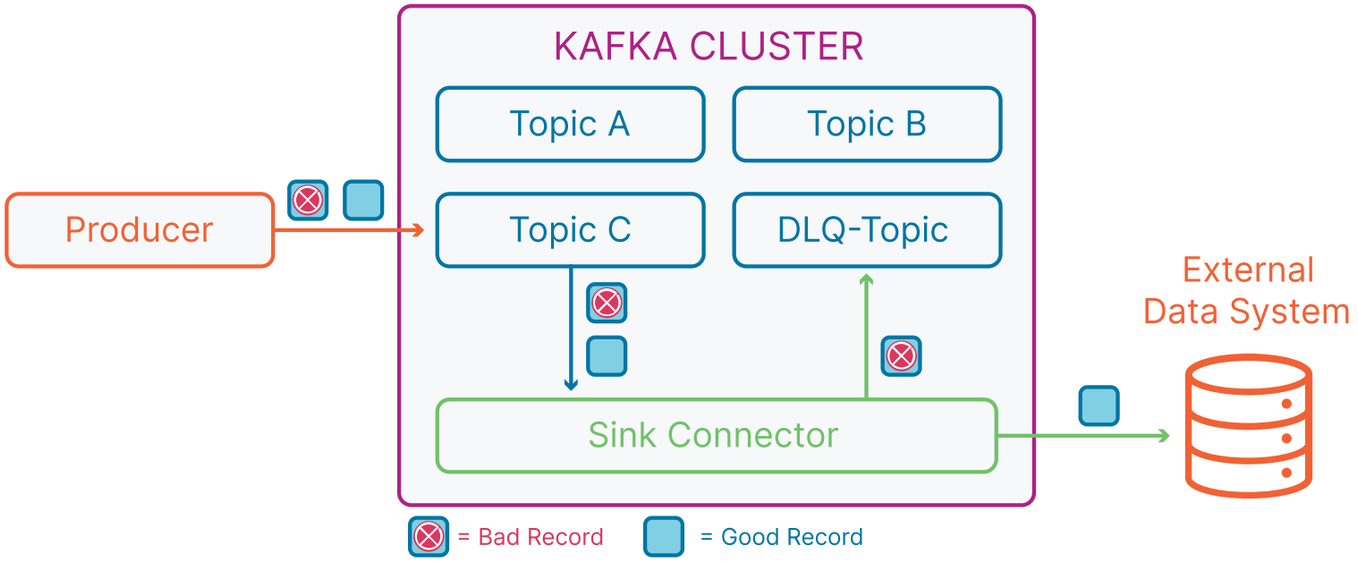
Auto-Created DLQ Topic

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



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:

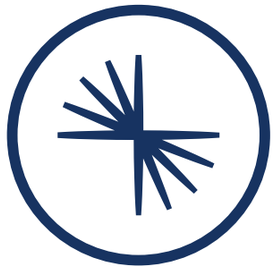
Role	Access
• `OrganizationAdmin` • `EnvironmentAdmin` • `CloudClusterAdmin`	• Create, configure, pause, restart, delete a connector • Read status and metrics, manage credentials and access • Create and view data preview
• `Operator`	• Pause and restart a connector • Read status and metrics
• `DeveloperRead`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

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 modules 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:

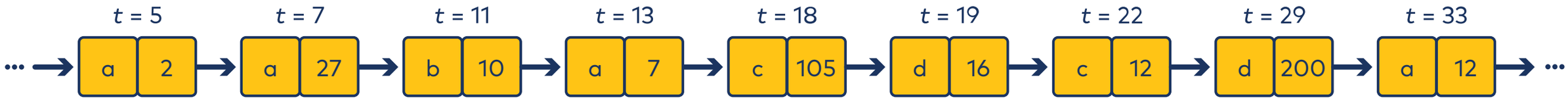
- Define these stream concepts:
  - A sample stream
  - Operating on the stream
  - Windowing the stream
  - Stream-table duality

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:

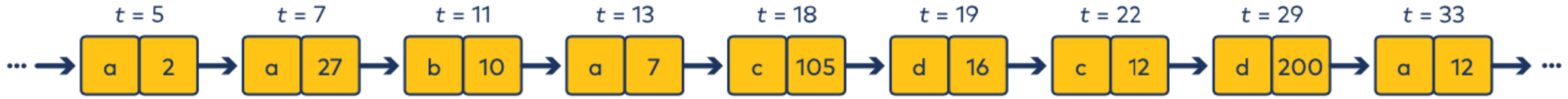


Output stream:

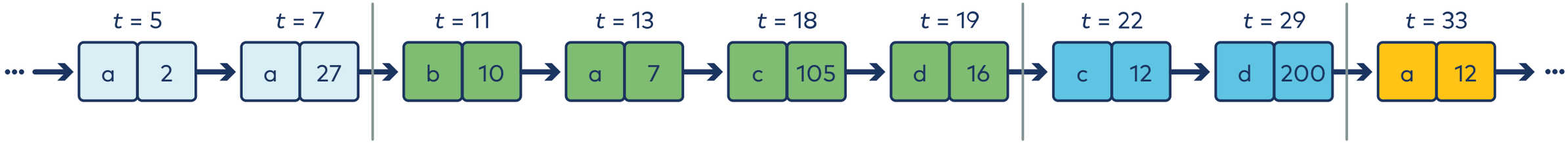
Windowing the Stream

We can split up time into windows.

- Here's our input stream:



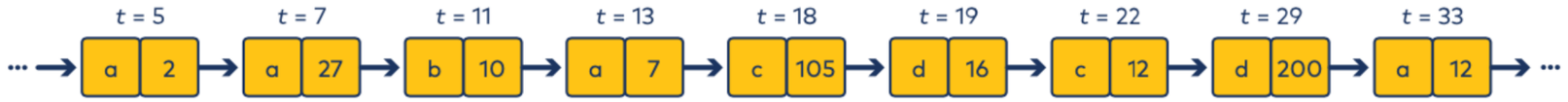
Here's the same stream, divided into windows of size 10:

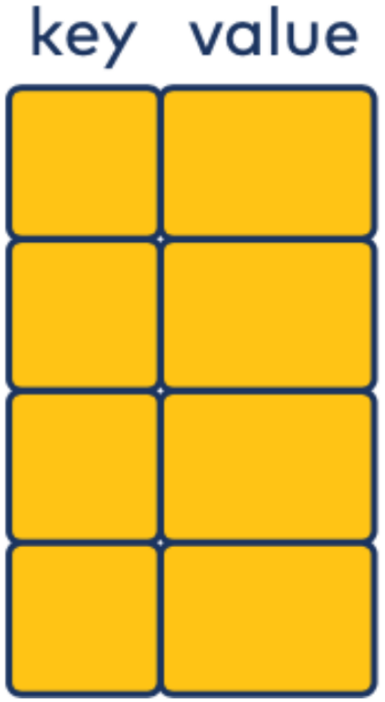


Stream-Table Duality

- We can view a stream as a table

Stream	Table
Records are events in time	Records are updates to same-key table entries





07b: What is ksqlDB?

Description

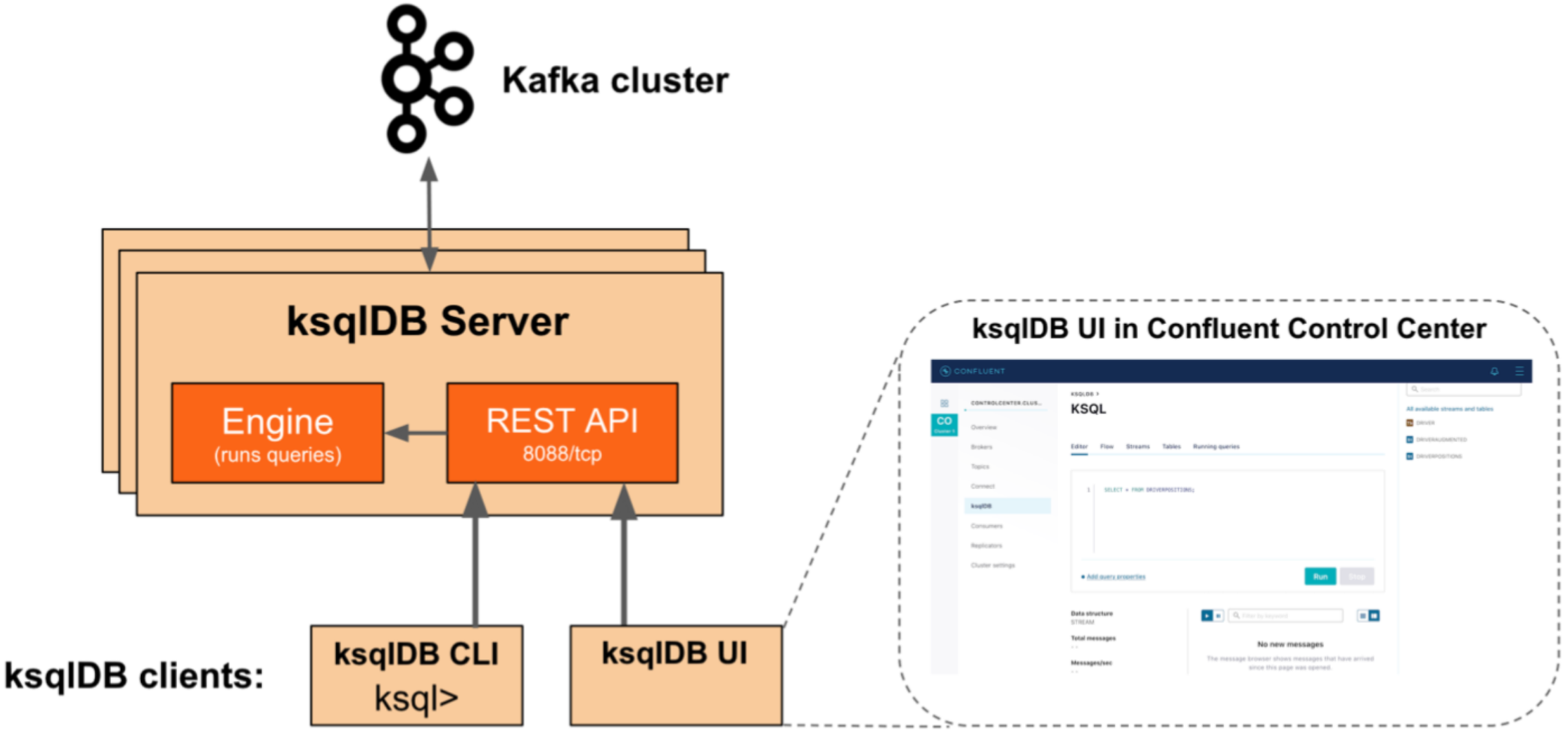
A review of ksqlDB, including stream processing apps, simple architecture, and components.

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

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

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

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 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

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

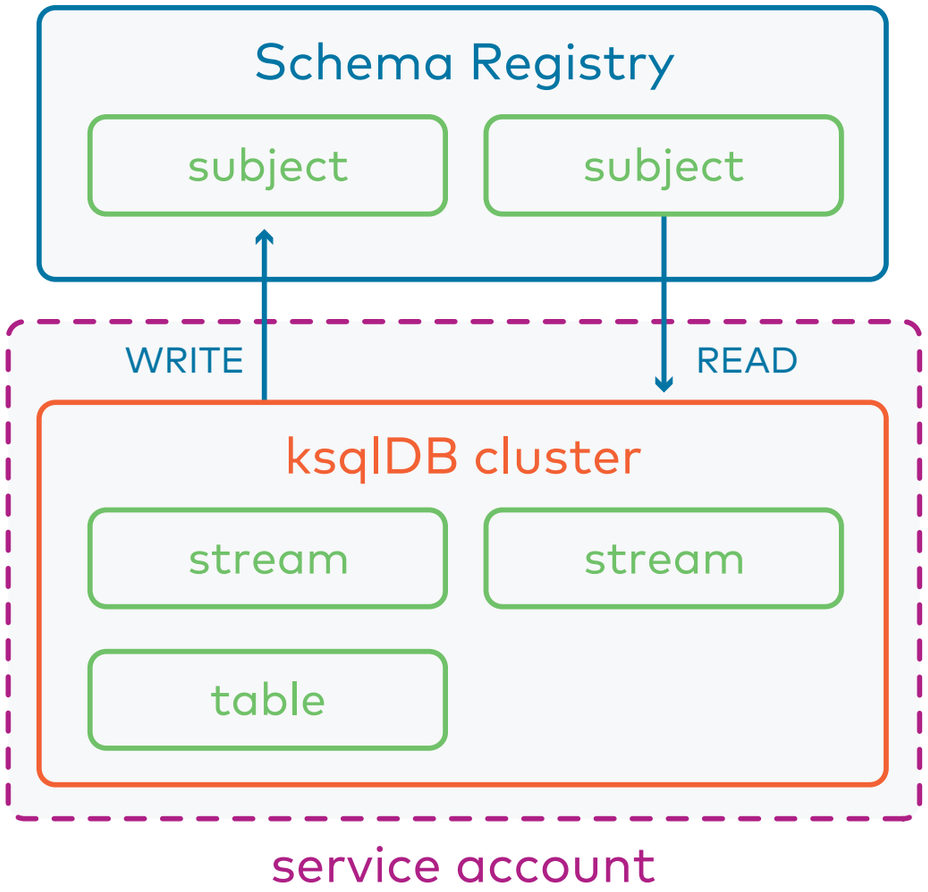
Upon completion of this lesson and associated lab exercises, you will be able to:

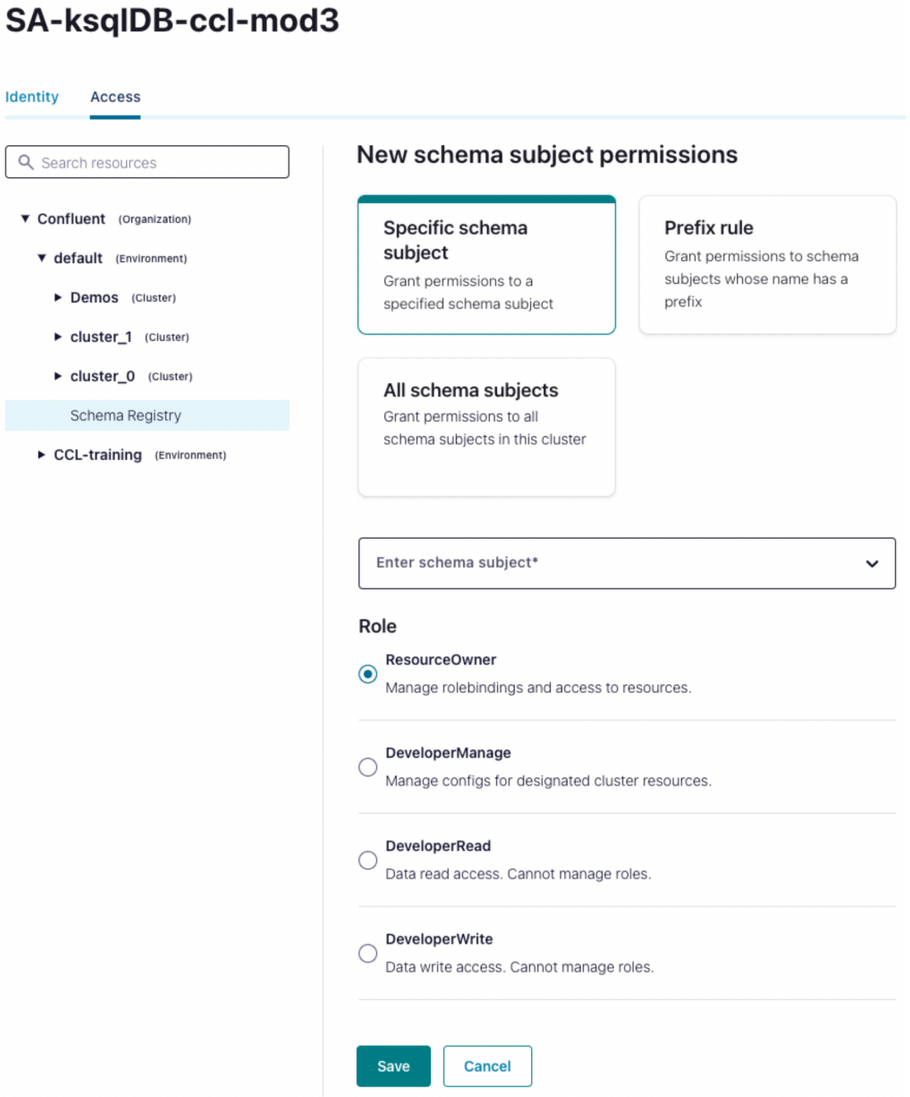
Evaluate security considerations, including:

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

Enable ksqlDB Access to Schema Registry

This configuration is required

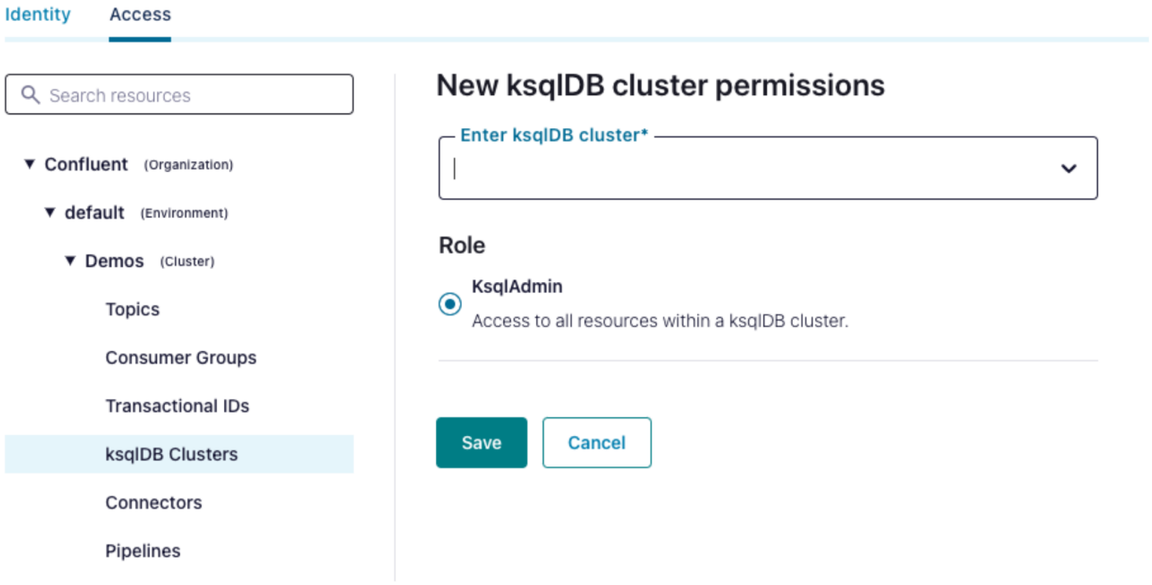


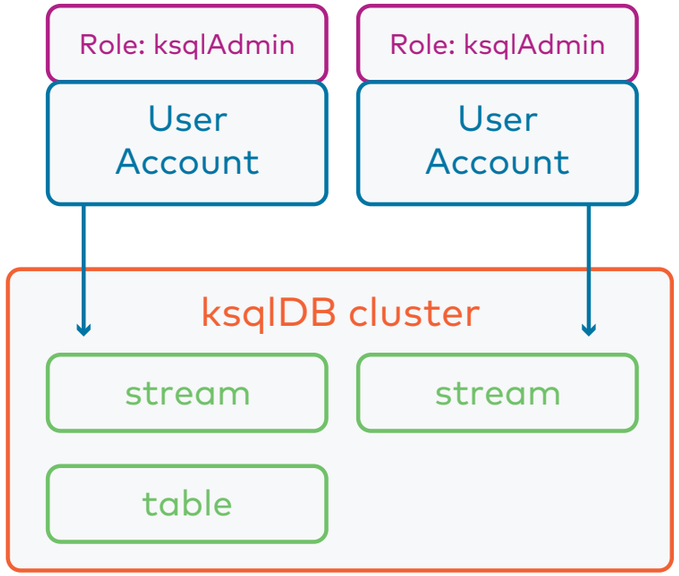


RBAC for ksqlDB Administration

There is only 1 role for ksqlDB cluster permissions:

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





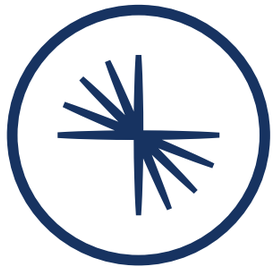
Only available on Standard and Dedicated clusters.

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 modules 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 Confuent Cloud.

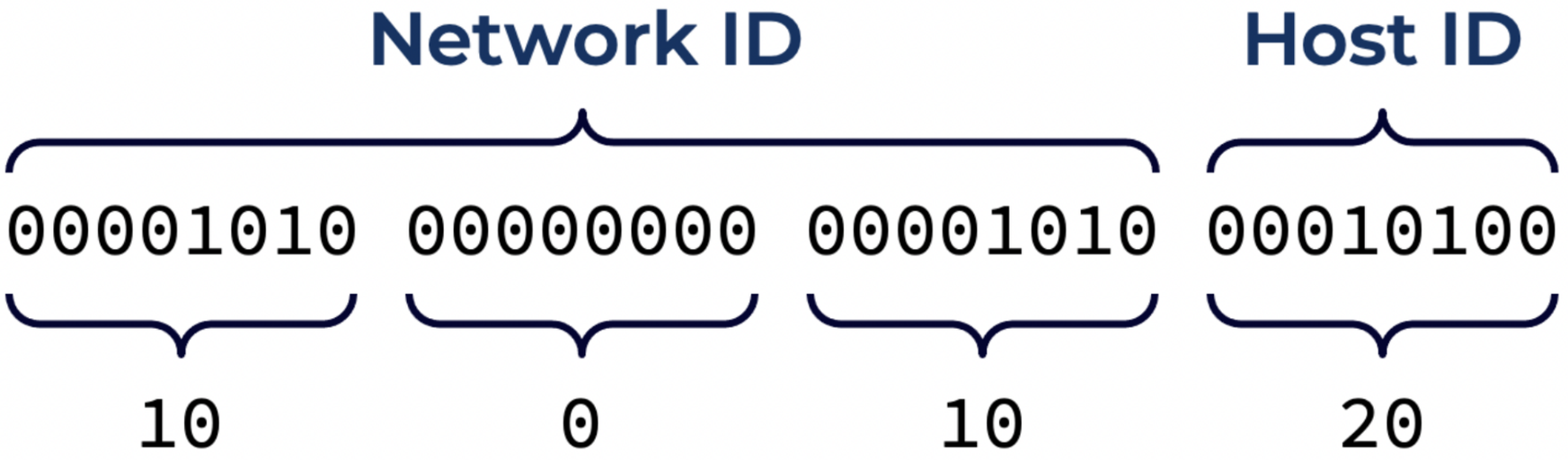
Learning Objectives

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

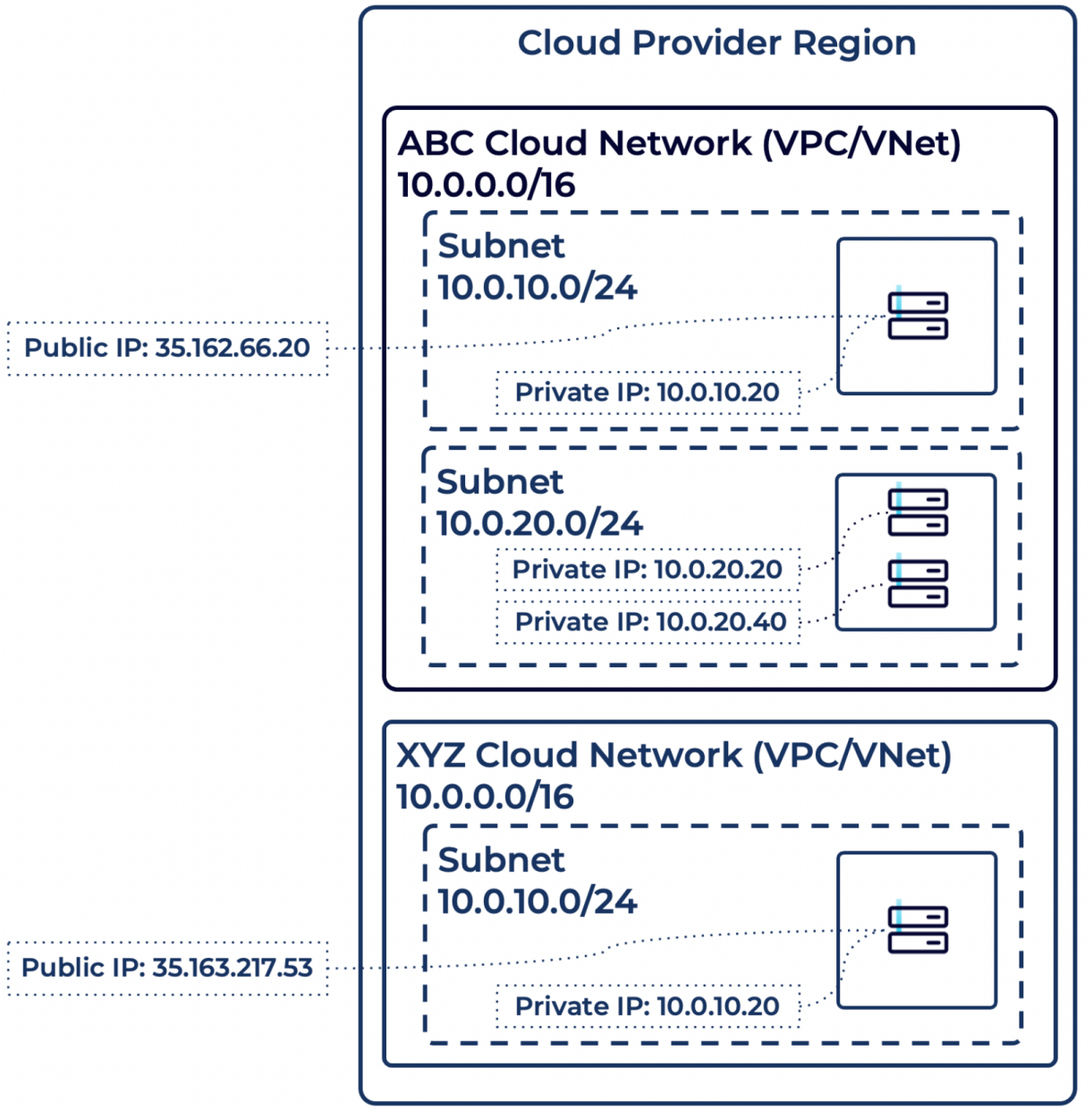
IPv4 Addresses (OPTIONAL)

- 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)



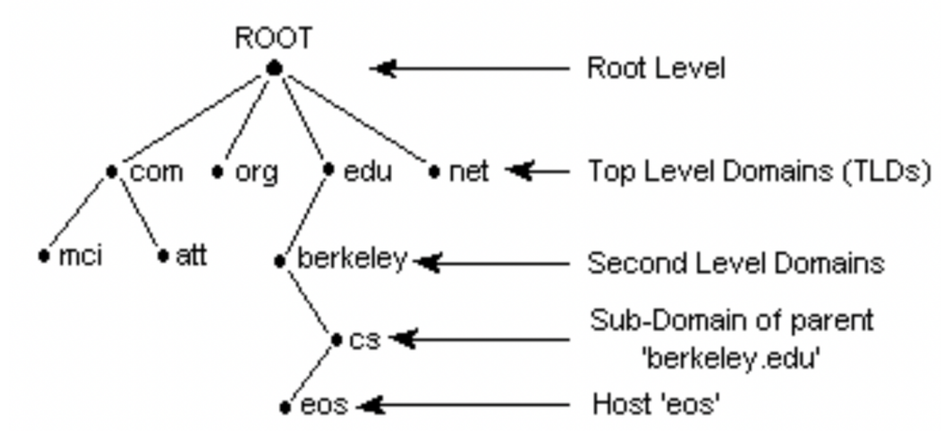
Public vs. Private (OPTIONAL)

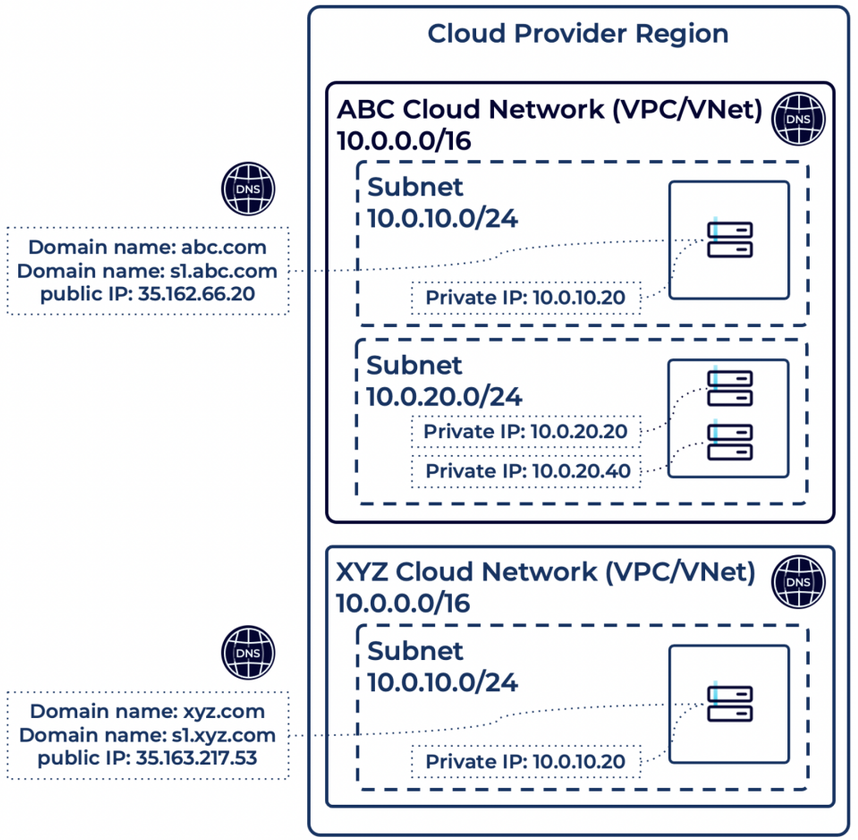
- Public IP addresses are accessible from the public Internet (globally unique)
- Private IP addresses are accessible from their private network (unique per network)
- All devices in cloud networks are assigned a private IP address
- Devices can be assigned both private and public IP addresses



Domain Name System - DNS (OPTIONAL)

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

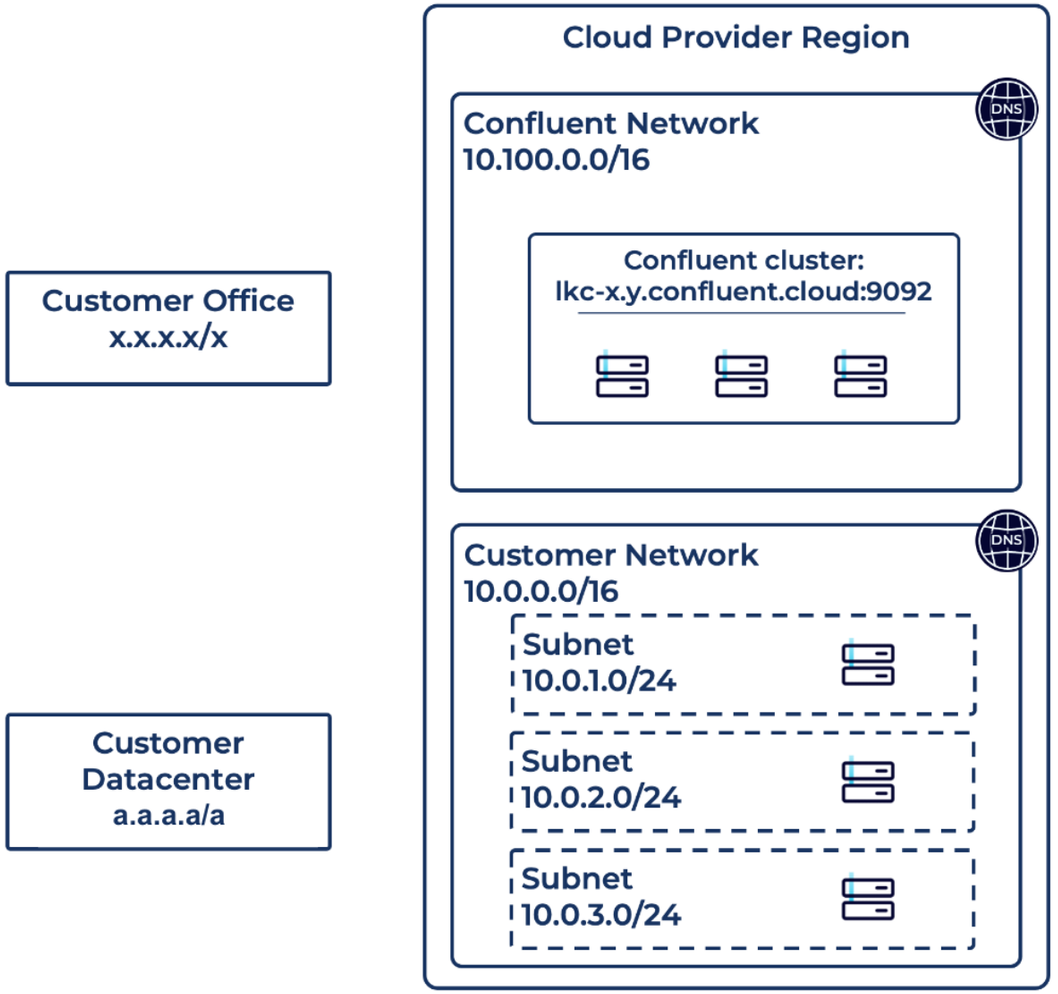




Cloud Networking (OPTIONAL)

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



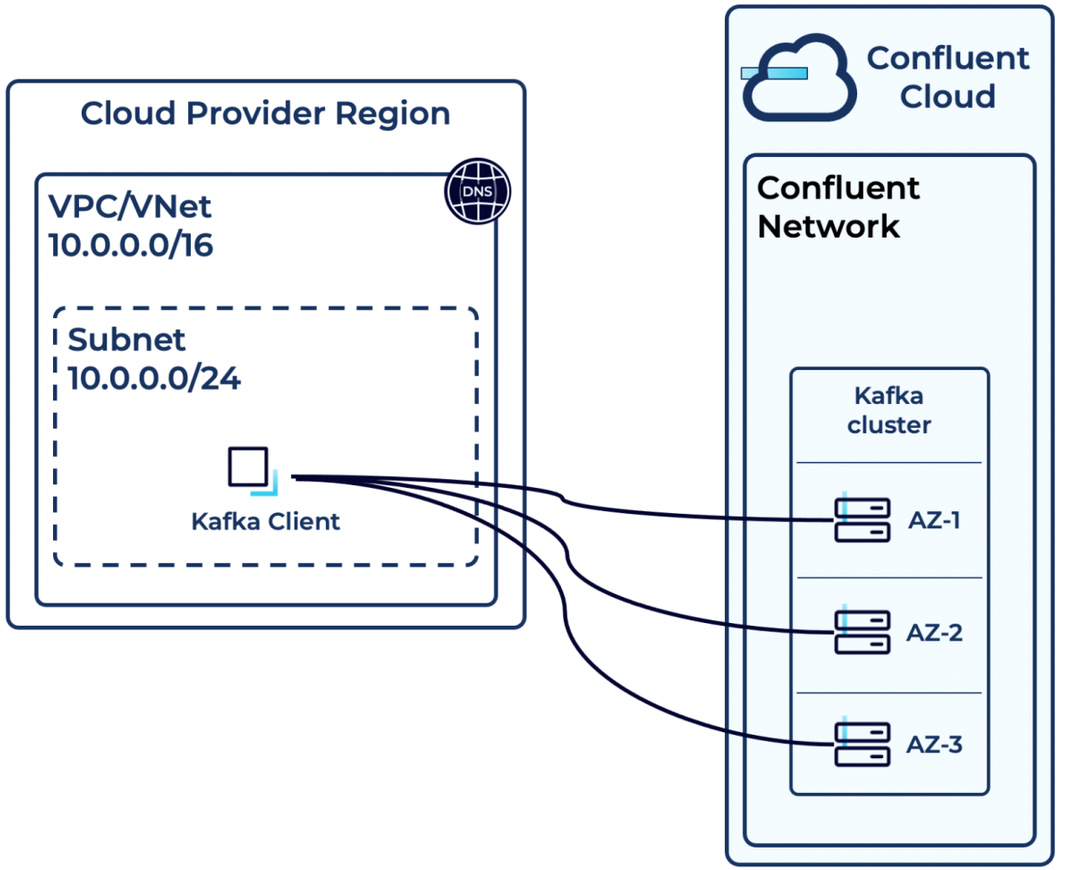
Cloud Provider Networks - VPC / VNet (OPTIONAL)

- 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

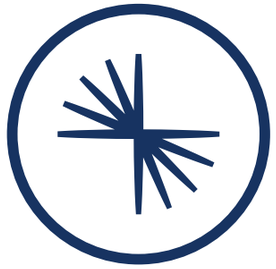
Kafka Networking (OPTIONAL)

- 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 modules contains two lessons:

- Terraform and the Confluent Provider
- Integrate Pulumi wtih 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

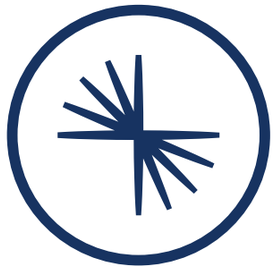
- Using Confluent Cloud and Terraform together provides a powerful way to manage your streaming infrastructure.
- With Terraform, you can define and manage your infrastructure as code, enabling you to easily manage your infrastructure across multiple environments and cloud providers.

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 modules 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

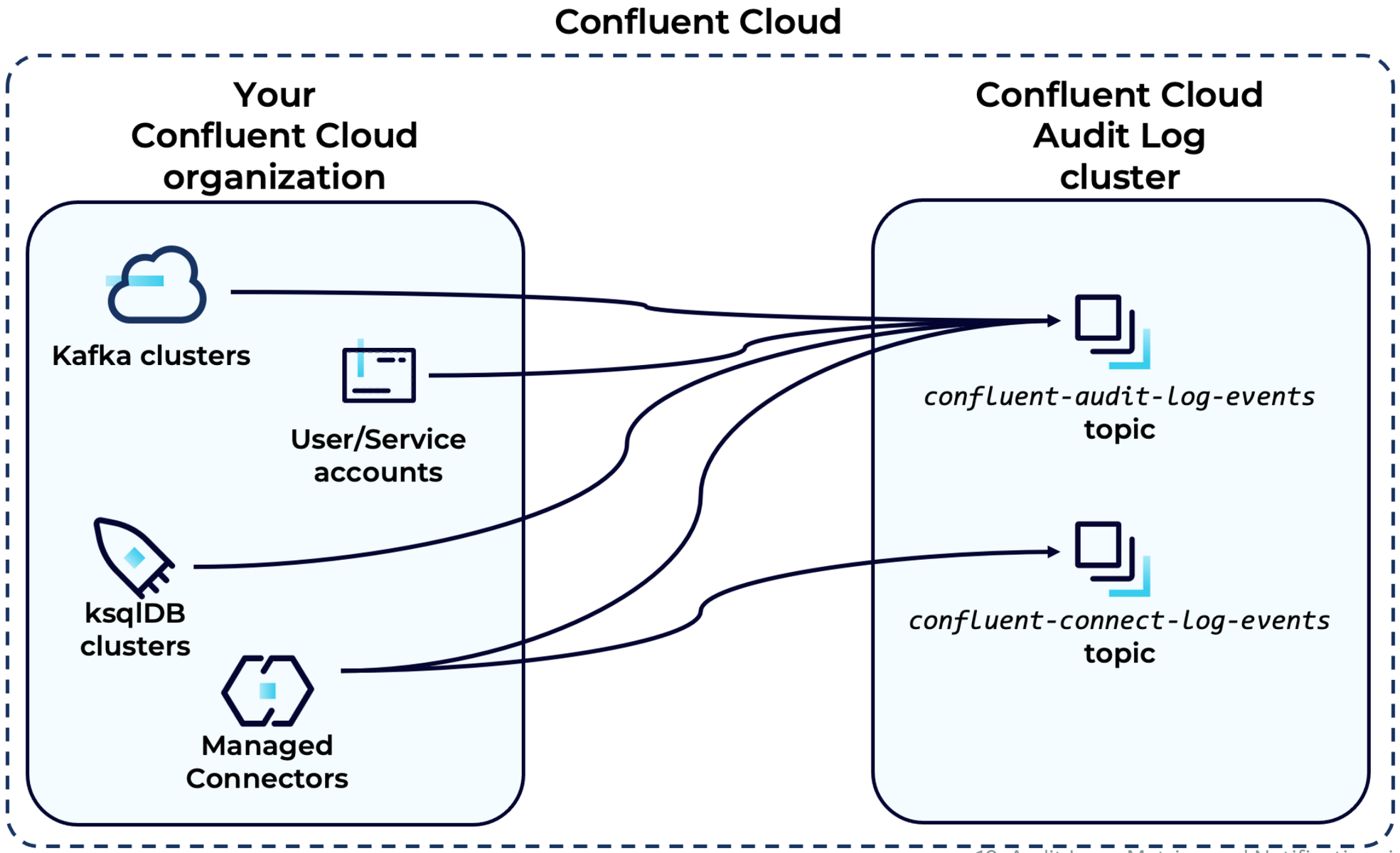
There are 3 types of Audit Log Events:

- 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...

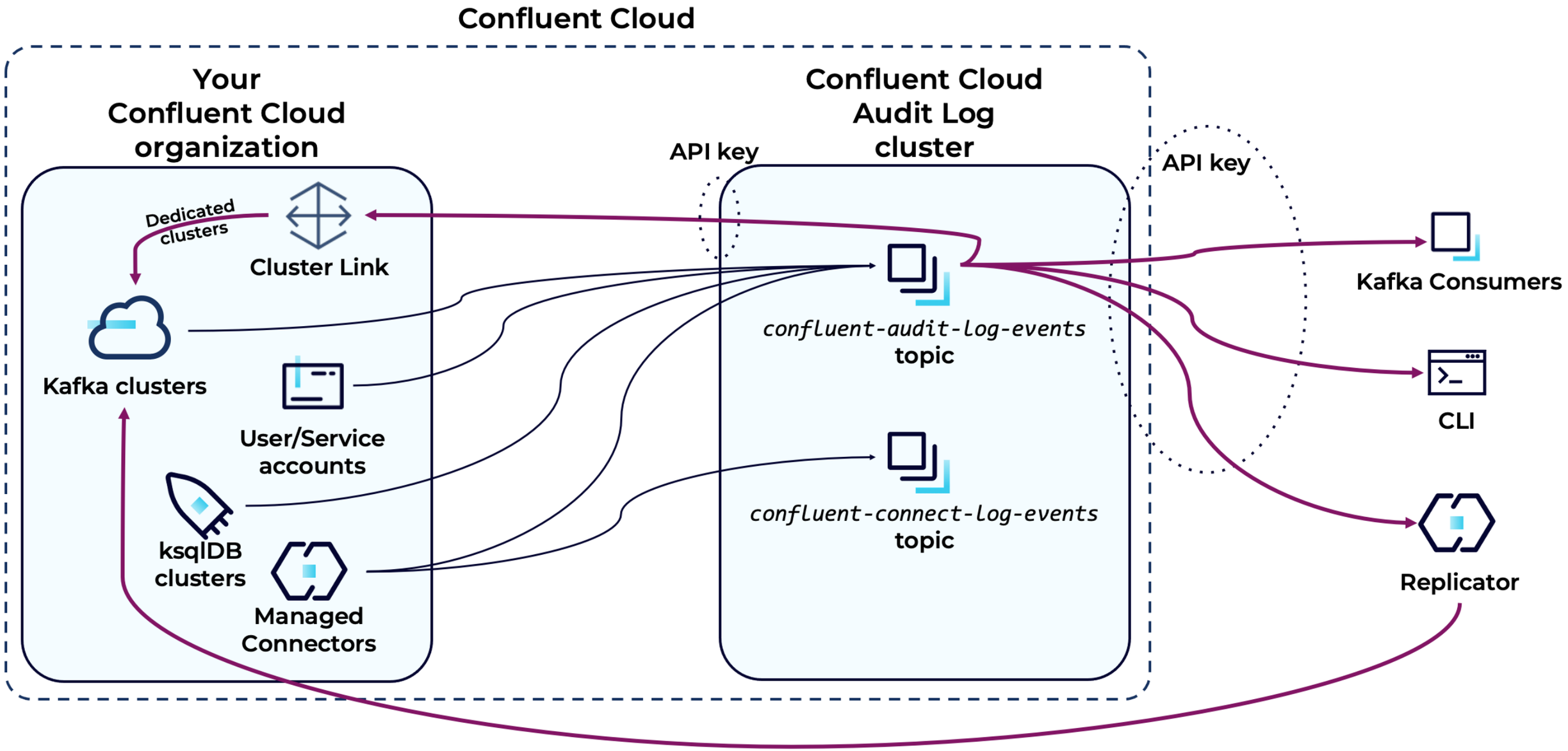
Audit Log Record Example

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

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*

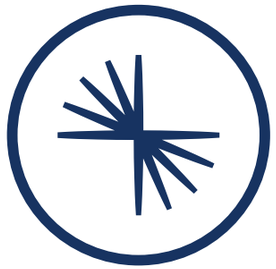
(Show Instructions in the Confluent Cloud UI)

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 modules contains one lab to practice everything you haver learned during these three days.

The objectives of this lab are:

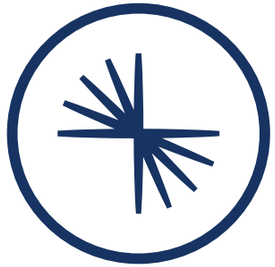
- Apply best practices when designing your Confluent Cloud Org
- Set appropriate security configurations to your User and Service accounts using API Keys, RBAC, and ACLs

- 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.



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.