Apache Kafka Glossary

New to Apache Kafka® and Confluent or looking for definitions? The terms below provide brief explanations and links to related content for important terms you'll encounter when working with the Confluent event streaming platform.

Search for a term...

Admin API

The Admin API is the Kafka REST API that enables administrators to manage and monitor Kafka clusters, topics, brokers, and other Kafka components.

Ansible Playbooks for Confluent Platform

Ansible Playbooks for Confluent Platform is a set of Ansible playbooks and roles that are designed to automate the deployment and management of Confluent Platform.

Apache Flink

Apache Flink is an open source stream processing framework for stateful computations over unbounded and bounded data streams. Flink provides a unified API for batch and stream processing that supports event-time and out-of-order processing, and supports exactly-once semantics. Flink applications include real-time analytics, data pipelines, and event-driven applications.

Related terms: *bounded stream*, *data stream*, *stream processing*, *unbounded stream*

Related content

- Apache Flink: Stream Processing and SQL on Confluent Cloud
- What is Apache Flink?
- Apache Flink 101 (Confluent Developer course)

Apache Kafka

Apache Kafka is an open source event streaming platform that provides a unified, high- secure data streaming

es distributed

- Introduction to Kafka

audit log

An audit log is a historical record of actions and operations that are triggered when auditable events occurs.

Audit log records can be used to troubleshoot system issues, manage security, and monitor compliance, by tracking administrative activity, data access and modification, monitoring sign-in attempts, and reconstructing security breaches and fraudulent activity.

Related terms: *auditable event*

Related content

- Audit Log Concepts for Confluent Cloud
- Audit Log Concepts for Confluent Platform

auditable event

An auditable event is an event that represents an action or operation that can be tracked and monitored for security purposes and compliance.

When an auditable event occurs, an auditable event method is triggered and an event message is sent to the audit log cluster and stored as an audit log record.

Related terms: *audit log*, *event message*

Related content

- Auditable Events in Confluent Cloud
- Auditable Events in Confluent Platform

authentication

Authentication is the process of verifying the identity of a principal that interacts with a system or application. Authentication is often used in conjunction with authorization to determine whether a principal is allowed to access a resource and perform a specific action or operation on that resource.

Digital authentication requires one or more of the following: something a principal knows (a password or security question), something a principal has (a security token or key), or something a principal is (a biometric characteristic, such as a fingerprint or voiceprint).

Multi-factor authentication (MFA) requires two or more forms of authentication.

Related terms: *authorization*, *identity*, *identity provider*, *identity pool*, *principal*, *role*

authorization

Authorization is the process of evaluating and then granting or denying a principal a set of permissions required to access and perform operations on resources.

Related terms: *authentication*, *group mapping*, *identity*, *identity provider*, *identity pool*, *principal*, *role*

Avro

Avro is a data serialization and exchange framework that provides data structures, remote procedure call (RPC), compact binary data format, a container file, and uses JSON to represent schemas.

Avro schemas ensure that every field is properly described and documented for use with serializers and deserializers. You can either send a schema with every message or use Schema Registry to store and receive schemas for use by consumers and producers to save bandwith and storage space.

Related terms: *data serialization*, *deserializer*, *serializer*

Related content

- Apache Avro - a data serialization system
- Confluent Cloud: Avro Schema Serializer and Deserializer
- Confluent Platform: Avro Schema Serializer and Deserializer

batch processing

Batch processing is the method of collecting a large volume of data over a specific time interval, after which the data is processed all at once and loaded into a destination system.

Batch processing is often used when processing data can occur independently of the source and timing of the data. It is efficient for non-real-time data processing, such as data warehousing, reporting, and analytics.

Related terms: *bounded stream*, *stream processing*, *unbounded stream*

CIDR block

A CIDR block is a group of IP addresses that are contiguous and can be represented as a single block. CIDR blocks are expressed using Classless Inter-domain Routing (CIDR) notation that includes an IP address and a number of bits in the network mask.

Related content

- CIDR notation
- Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan [RFC 4632]

Cluster Linking

Cluster Linking is a highly performant data replication feature that enables links between Kafka clusters to mirror data from one cluster to another. Cluster Linking creates perfect copies of Kafka topics, which keep data in sync across clusters. Use cases include geo-replication of data, data sharing, migration, disaster recovery, and tiered separation of critical applications.

Related content

- Geo-replication with Cluster Linking on Confluent Cloud
- Cluster Linking for Confluent Platform

commit log

A commit log is a log of all event messages about commits (changes or operations made) sent to a Kafka topic.

A commit log ensures that all event messages are processed at least once and provides a mechanism for recovery in the event of a failure.

The commit log is also referred to as a write-ahead log (WAL) or a transaction log.

Related terms: *event message*

Confluent Cloud

Confluent Cloud is the fully managed, cloud-native event streaming service powered by Kora, the event streaming platform based on Kafka and extended by Confluent to provide high availability, scalability, elasticity, security, and global interconnectivity. Confluent Cloud offers cost-effective multi-tenant confgurations as well as dedicated solutions, if stronger isolation is required.

Related terms: *Apache Kafka*, *Kora*

Related content

- Confluent Cloud Overview
- Confluent Cloud

Confluent Cloud network

A Confluent Cloud network is an abstraction for a single tenant network environment that hosts Dedicated Kafka clusters in Confluent Cloud along with their single tenant services, like ksqlDB clusters and managed connectors.

Related content

- Confluent Cloud Network Overview

Confluent for Kubernetes (CFK)

Confluent for Kubernetes (CFK) is a cloud-native control plane for deploying and managing Confluent in private cloud environments through declarative API.

Confluent Platform

Confluent Platform is a specialized distribution of Kafka at its core, with additional components for data integration, streaming data pipelines, and stream processing.

Confluent REST Proxy

Confluent REST Proxy provides a RESTful interface to an Kafka cluster, making it easy to produce and consume messages, view the state of the cluster, and perform administrative actions without using the native Kafka protocol or clients.

Related content

- Confluent Platform: REST Proxy

Confluent Server

Confluent Server is the default Kafka broker component of Confluent Platform that builds on the foundation of Apache Kafka® and provides enhanced proprietary features designed for enterprise use. Confluent Server is fully compatible with Kafka, and adds Kafka cluster support for Role-Based Access Control, Audit Logs, Schema Validation, Self Balancing Clusters, Tiered Storage, Multi-Region Clusters, and Cluster Linking.

Related terms: *Confluent Platform*, *Apache Kafka*, *Kafka broker*, *Cluster Linking*, *multi-region cluster (MRC)*

Confluent Unit for Kafka (CKU)

Confluent Unit for Kafka (CKU) is a unit of horizontal scaling for Dedicated Kafka clusters in Confluent Cloud that provide preallocated resources.

CKUs determine the capacity of a Dedicated Kafka cluster in Confluent Cloud.

Related content

- CKU limits per cluster

Connect API

The Connect API is the Kafka API that enables a connector to read event streams from a source system and write to a target system.

Connect worker

A Connect worker is a server process that runs a connector and performs the actual work of moving data in and out of Kafka topics.

A worker is a server process that runs on hardware independent of the Kafka brokers themselves. It is scalable and fault-tolerant, meaning you can run a cluster of workers that share the load of moving data in and out of Kafka from and to external systems.

Related terms: *connector*, *Kafka Connect*

connector

A connector is an abstract mechanism that enables communication, coordination, or cooperation among components by transferring data elements from one interface to another without changing the data.

connector offset

Connector offset uniquely identifies the position of a connector as it processes data. Connectors use a variety of strategies to implement the connector offset, including everything from monotonically increasing integers to replay ids, lists of files, timestamps and even checkpoint information.

Connector offsets keep track of already-processed data in the event of a connector restart or recovery. While sink connectors use a pattern for connector offsets similar to the offset mechanism used throughout Kafka, the implementation details for source connectors are often much different. This is because source connectors track the progress of a source system as it process data.

consumer

A consumer is a Kafka client application that subscribes to (reads and processes) event messages from a Kafka topic.

The Streams API and the Consumer API are the two APIs that enable consumers to read event streams from Kafka topics.

Related terms: *Consumer API*, *consumer group*, *producer*, *Streams API*

Consumer API

The Consumer API is the Kafka API used for consuming (reading) event messages or records from Kafka topics and enables a Kafka consumer to subscribe to a topic and read event messages as they arrive.

Batch processing is a common use case for the Consumer API.

consumer group

A consumer group is a single logical consumer implemented with multiple physical

consumers for reasons of throughput and resilience.

By dividing topics among consumers in the group into partitions, consumers in the group can process messages in parallel, increasing message throughput and enabling load balancing.

Related terms: *consumer*, *partition*, *partition*, *producer*, *topic*

consumer lag

Consumer lag is the number of consumer offsets between the latest message produced in a partition and the last message consumed by a consumer, that is the number of messages pending to be consumed from a particular partition.

A large consumer lag, or a quickly growing lag, indicates that the consumer is unable to read from a partition as fast as the messages are available. This can be caused by a slow consumer, slow network, or slow broker.

consumer offset

Consumer offset is the unique and monotonically increasing integer value that uniquely identifies the position of an event record in a partition. Consumers use offsets to track their current position in the Kafka topic, allowing consumers to resume processing from where they left off.

Offsets are stored on the Kafka broker, which does not track which records have been read and which have not. It is up to the consumer connection to track this information. When a consumer acknowledges receiving and processing a message, it commits an offset value that is stored in the special internal topic `__commit_offsets`.

cross-resource RBAC role binding

A cross-resource RBAC role binding is a role binding in Confluent Cloud that is applied at the Organization or Environment scope and grants access to multiple resources. For example, assigning a principal the NetworkAdmin role at the Organization scope lets them administer all networks across all Environments in their Organization.

Related terms: *identity pool*, *principal*, *role*, *role binding*

CRUD

CRUD is an acronym for the four basic operations that can be performed on data: Create, Read, Update, and Delete.

custom connector

A custom connector is a connector created using Connect plugins uploaded to Confluent Cloud by users. This includes connector plugins that are built from scratch, modified open-source connector plugins, or third-party connector plugins.

data at rest

Data at rest is data that is physically stored on non-volatile media (such as hard drives, solid-state drives, or other storage devices) and is not actively being transmitted or processed by a system.

data encryption key (DEK)

A data encryption key (DEK) is a symmetric key that is used to encrypt and decrypt data. The DEK is used in client-side field level encryption (CSFLE) to encrypt sensitive

data. The DEK is itself encrypted using a key encryption key (KEK) that is only accessible to authorized users. The encrypted DEK and encrypted data are stored together. Only users with access to the KEK can decrypt the DEK and access the sensitive data.

Related terms: *envelope encryption*, *key encryption key (KEK)*

data in motion

Data in motion is data that is actively being transferred between source and destination, typically systems, devices, or networks.

Data in motion is also referred to as data in transit or data in flight.

data in use

Data in use is data that is actively being processed or manipulated in memory (RAM, CPU caches, or CPU registers).

data ingestion

Data ingestion is the process of collecting, importing, and integrating data from various sources into a system for further processing, analysis, or storage.

data mapping

Data mapping is the process of defining relationships or associations between source data elements and target data elements.

Data mapping is an important process in data integration, data migration, and data transformation, ensuring that data is accurately and consistently represented when it is moved or combined.

data pipeline

A data pipeline is a series of processes and systems that enable the flow of data from sources to destinations, automating the movement and tranformation of data for various purposes, such as analytics, reporting, or machine learning.

A data pipeline typically comprised of a source system, a data ingestion tool, a data transformation tool, and a target system. A data pipeline covers the following stages: data extraction, data transformation, data loading, and data validation.

Data Portal

Data Portal is a Confluent Cloud application that uses Stream Catalog and Stream Lineage to provide self-service access throught Confluent Cloud Console for data practiioners to search and discover existing topics using tags and business metaddata, request access to topics and data, and access data in topics to to build streaming applications and data pipelines.

Leverages Stream Catalog and Stream Lineage to provide a data-centric view of Confluent optimized for self-service access to data where users can search, discover and understand available data, request access to data, and use data.

Related terms: *Stream Catalog*, *Stream Lineage*

Related content

- Data Portal on Confluent Cloud

data serialization

Data serialization is the process of converting data structures or objects into a format that can be stored or transmitted, and reconstructed later in the same or another computer environment.

Data serialization is a common technique for implementing data persistence, interprocess communication, and object communication. Confluent Schema Registry (in Confluent Platform) and Confluent Cloud Schema Registry support data serialization using serializers and deserializers for the following formats: Avro, JSON Schema, and Protobuf.

Related content

- Confluent Cloud: Formats, Serializers, and Deserializers
- Confluent Platform: Formats, Serializers, and Deserializers

data steward

A data steward is a person with data-related responsibilities, such as data governance, data quality, and data security.

data stream

A data stream is a continuous flow of data records that are produced and consumed by applications.

dead letter queue (DLQ)

A dead letter queue (DLQ) is a queue where messages that could not be processed successfully by a sink connector are placed. Instead of stopping, the sink connector sends messages that could not be written successfully as event records to the DLQ topic while the sink connector continues processing messages.

deserializer

A deserializer is a tool that converts a serial byte stream back into objects and parallel data. Deserializers work with serializers (known together as Serdes) to support efficient storage and high-speed data transmission over the wire. Confluent provides Serdes for schemas in Avro, Protobuf, and JSON Schema formats.

Related content

- Confluent Cloud: Formats, Serializers, and Deserializers
- Confluent Platform: Formats, Serializers, and Deserializers

Elastic Confluent Unit for Kafka

Elastic Confluent Unit for Kafka (eCKU) is used to express capacity for Basic, Standard, and Enterprise Kafka clusters. These clusters automatically scale up to a fixed ceiling. There is no need to resize these type clusters. When you need more capacity, your cluster expands up to the fixed ceiling. If you're not using capacity above the minimum, you're not paying for it.

ELT

ELT is an acronym for Extract-Load-Transform, where data is extracted from a source system and loaded into a target system before processing or transformation.

Compared to ETL, ELT is a more flexible approach to data ingestion because the data is loaded into the target system before transformation.

envelope encryption

Envelope encryption is a cryptographic technique that uses two keys to encrypt data. The symmetric data encryption key (DEK) is used to encrypt sensitive data. The separate asymmetric key encryption key (KEK) is the master key used to encrypt the DEK. The DEK and encrypted data are stored together. Only users with access to the KEK can decrypt the DEK and access the sensitive data.

In Confluent Cloud, envelope encryption is used to enable client-side field level encryption (CSFLE). CSFLE encrypts sensitive data in a message before it is sent to Confluent Cloud and allows for temporary decryption of sensitive data when required to perform operations on the data.

Related terms: *data encryption key (DEK)*, *key encryption key (KEK)*

ETL

ETL is an acronym for Extract-Transform-Load, where data is extracted from a source system, transformed into a target format, and loaded into a target system.

Compared to ELT, ETL is a more rigid approach to data ingestion because the data is transformed before loading into the target system.

event

An event is a meaningful action or occurrence of something that happened.

Events that can be recognized by a program, either human-generated or triggered by software, can be recorded in a log file or other data store.

Related terms: *event message*, *event record*, *event sink*, *event source*, *event stream*, *event streaming*, *event streaming platform*, *event time*

event message

An event message is a record of an event sent to a Kafka topic, represented as a key-value pair.

Each event message consists of a key-value pair, a timestamp, the compression type, headers for metadata (optional), and a partition and offset ID (once the message is written). The key is optional and can be used to identify the event. The value is required and contains details about the event that happened.

Related terms: *event*, *event record*, *event sink*, *event source*, *event stream*, *event streaming*, *event streaming platform*, *event time*

event record

An event record is the record of an event stored in a Kafka topic.

Event records are organized and durably stored in topics. Examples of events include orders, payments, activities, or measurements. An event typically contains one or more data fields that describe the fact, as well as a timestamp that denotes when the event was created by its event source. The event may also contain various metadata, such as its source of origin (for example, the application or cloud service that created the event) and storage-level information (for example, its position in the event stream).

Related terms: *event*, *event message*, *event sink*, *event source*, *event stream*, *event streaming*, *event streaming platform*, *event time*

event sink

An event sink is a consumer of events, which can include applications, cloud services, databases, IoT sensors, and more.

Related terms: *event*, *event message*, **event record*, *event source*, *event stream*, *event streaming*, *event streaming platform*, *event time*

event source

An event source is a producer of events, which can include cloud services, databases, IoT sensors, mainframes, and more.

Related terms: *event*, *event message*, *event record*, *event sink*, *event stream*, *event streaming*, *event streaming platform*, *event time*

event stream

An event stream is a continuous flow of event messages produced by an event source and consumed by one or more consumers.

Related terms: *event*, *event message*, *event record*, *event sink*, *event source*, *event streaming*, *event streaming platform*, *event time*

event streaming

Event streaming is the practice of capturing event data in real-time from data sources.

Event streaming is a form of data streaming that is used to capture, store, process, and react to data in real-time or retrospectively.

Related terms: *event*, *event message*, *event record*, *event sink*, *event source*, *event streaming platform*, *event time*

event streaming platform

An event streaming platform is a platform that events can be written to once, allowing distributed functions within an organization to react in realtime.

Related terms: *event*, *event message*, *event record*, *event sink*, *event source*, *event streaming*, *event time*

event time

Event time is the time when an event occurred on the producing device, as opposed to the time when the event was processed or recorded. Event time is often used in stream processing to determine the order of events and to perform windowing operations.

Related terms: *event*, *event message*, *event record*, *event sink*, *event source*, *event streaming*, *event streaming platform*

exactly-once semantics

Exactly-once semantics is a guarantee that a message is delivered exactly once and in the order that it was sent.

Even if a producer retries sending a message, or a consumer retries processing a message, the message is delivered exactly once. This guarantee is achieved by the broker assigning a unique ID to each message and storing the ID in the consumer offset. The consumer offset is committed to the broker only after the message is

processed. If the consumer fails to process the message, the message is redelivered and processed again.

granularity

Granularity is the degree or level of detail to which an entity (a system, service, or resource) is broken down into subcomponents, parts, or elements.

Entities that are *fine-grained* have a higher level of detail, while *coarse-grained* entities have a reduced level of detail, often combining finer parts into a larger whole.

In the context of access control, granular permissions provide precise control over resource access. They allow administrators to grant specific operations on distinct resources. This ensures users only have permissions tailored to their needs, minimizing unnecessary or potentially risky access.

group mapping

Group mapping is a set of rules that map groups in your SSO identity provider to Confluent Cloud RBAC roles. When a user signs in to Confluent Cloud using SSO, Confluent Cloud uses the group mapping to grant access to Confluent Cloud resources.

Related terms: *identity provider*, *identity pool*, *principal*, *role*

Related content

- Group Mapping for Confluent Cloud

identity

An identity is a unique identifier that is used to authenticate and authorize users and applications to access resources.

Identity is often used in conjunction with access control to determine whether a user or application is allowed to access a resource and perform a specific action or operation on that resource.

Related terms: *identity provider*, *identity pool*, *principal*, *role*

identity pool

An identity pool is a collection of identities that can be used to authenticate and authorize users and applications to access resources.

Identity pools are used to manage permissions for users and applications that access resources in Confluent Cloud. They are also used to manage permissions for Confluent Cloud service accounts that are used to access resources in Confluent Cloud.

identity provider

An identity provider is a trusted provider that authenticates users and issues security tokens that are used to verify the identity of a user.

Identity providers are often used in single sign-on (SSO) scenarios, where a user can log in to multiple applications or services with a single set of credentials.

Infinite Storage

Infinite Storage is the Confluent Cloud storage service that enhances the scalability of Confluent Cloud resources by separating storage and processing. Tiered storage

within Confluent Cloud moves data between storage layers based on the needs of the workload, retrieves tiered data when requested, and garbage collects data that is past retention or otherwise deleted.

If an application reads historical data, latency is not increased for other applications reading more recent data. Storage resources are decoupled from compute resources, you only pay for what you produce to Confluent Cloud and for storage that you use, and CKUs do not have storage limits.

Related content:

- Infinite Storage in Confluent Cloud for Apache Kafka

internal topic

An internal topic is a topic, prefixed with double underscores ("__"), that is automatically created by a Kafka component to store metadata about the broker, partition assignment, consumer offsets, and other information.

Examples of internal topics: `__cluster_metadata`, `__consumer_offsets`, `__transaction_state`, `__confluent.support.metrics`, and `__confluent.support.metrics-raw`.

JSON Schema

JSON Schema is a declarative language used for data serialization and exchange to define data structures, specify formats, and validate JSON documents. It is a way to encode expected data types, properties, and constraints to ensure that all fields are properly described for use with serializers and deserializers.

Related terms: *data serialization*, *deserializer*, *serializer*

Related content

- JSON Schema - a declarative language that allows you to annotate and validate JSON documents.
- Confluent Cloud: JSON Schema Serializer and Deserializer
- Confluent Platform: JSON Schema Serializer and Deserializer

Kafka bootstrap server

A Kafka bootstrap server is a Kafka broker that a Kafka client initiates a connection to a Kafka cluster and returns metadata, which includes the addresses for all of the brokers in the Kafka cluster.

Although only one bootstrap server is required to connect to a Kafka cluster, multiple brokers can be specified in a bootstrap server list to provide high availability and fault tolerance in case a broker is unavailable. In Confluent Cloud, the bootstrap server is the general cluster endpoint.

Kafka broker

A Kafka broker is a server in the Kafka storage layer that stores event streams from one or more sources.

A Kafka cluster is typically comprised of several brokers. Every broker in a cluster is also a bootstrap server, meaning if you can connect to one broker in a cluster, you can connect to every broker.

Kafka client

A Kafka client allows you to write distributed applications and microservices that read, write, and process streams of events in parallel, at scale, and in a fault-tolerant manner, even in the case of network problems or machine failures.

The Kafka client library provides functions, classes, and utilities that allow developers to create Kafka producer clients (Producers) and consumer clients (Consumers) using various programming languages. The primary way to build production-ready Producers and Consumers is by using your preferred programming language and a Kafka client library.

Related content

- Build Client Applications for Confluent Cloud
- Build Client Applications for Confluent Platform
- Getting Started with Apache Kafka and Java (or Python, Go, .Net, and others)

Kafka cluster

A Kafka cluster is a group of interconnected Kafka brokers that manage and distribute real-time data streaming, processing, and storage as if they are a single system.

By distributing tasks and services across multiple Kafka brokers, the Kafka cluster improves availability, reliability, and performance.

Kafka Connect

Kafka Connect is the component of Kafka that provides data integration between databases, key-value stores, search indexes, file systems, and Kafka brokers.

Kafka Connect is an ecosystem of a client application and pluggable connectors. As a client application, Connect is a server process that runs on hardware independent of the Kafka brokers themselves. It is scalable and fault-tolerant, meaning you can run a cluster of Connect workers that share the load of moving data in and out of Kafka from and to external systems. Connect also abstracts the business of code away from the user and instead requires only JSON configuration to run.

Related content

- Confluent Cloud: Kafka Connect
- Confluent Platform: Kafka Connect

Kafka controller

A Kafka controller is the node in a Kafka cluster that is responsible for managing and changing the metadata of the cluster. This node also communicates metadata changes to the rest of the cluster. When Kafka uses ZooKeeper for metadata management, the controller is a broker, and the broker persists the metadata to ZooKeeper for backup and recovery. With KRaft, you dedicate Kafka nodes to operate as controllers and the metadata is stored in Kafka itself and not persisted to ZooKeeper. KRaft enables faster recovery because of this.

For more information, see KRaft overview.

Kafka listener

A Kafka listener is an endpoint that Kafka brokers bind to use to communicate with clients.

For Kafka clusters, Kafka listeners are configured in the `listeners` property of the

`server.properties` file. Advertised listeners are publicly accessible endpoints that are used by clients to connect to the Kafka cluster.

Related content

- Kafka Listeners – Explained

Kafka metadata

Kafka metadata is the information about the Kafka cluster and the topics that are stored in it. This information includes details such as the brokers in the cluster, the topics that are available, the partitions for each topic, and the location of the leader for each partition.

Kafka metadata is used by clients to discover the available brokers and topics, and to determine which broker is the leader for a particular partition. This information is essential for clients to be able to send and receive messages to and from Kafka.

Kafka Streams

Kafka Streams is a stream processing library for building streaming applications and microservices that transform (filter, group mapping, aggregate, join, and more) incoming event streams in real-time to Kafka topics stored in an Kafka cluster.

The Streams API can be used to build applications that process data in real-time, analyze data continuously, and build data pipelines.

Related content

- Kafka Streams

Kafka topic

A Kafka topic is a user-defined category or feed name where event messages are stored and published by producers and subscribed to by consumers.

Each topic is a log of event messages. Topics are stored in one or more partitions, which distribute topic records brokers in a Kafka cluster. Each partition is an ordered, immutable sequence of records that are continually appended to a topic.

Related content

- Manage Topics in Confluent Cloud

key encryption key (KEK)

A key encryption key (KEK) is a master key that is used to encrypt and a decrypt other keys, specifically the data encryption key (DEK). Only users with access to the KEK can decrypt the DEK and access the sensitive data.

Related terms: *data encryption key (DEK)*, *envelope encryption*.

Kora

Kora is the cloud-native streaming data service based on Kafka technology that powers the Confluent Cloud event streaming platform for building real-time data pipelines and streaming applications. Kora abstracts low-level resources, such as Kafka brokers, and hides operational complexities, such as system upgrades.

Kora is built on the following foundations: a tiered storage layer that improves cost and performance, elasticity and consistent performance through incremental load

balancing, cost effective multi-tenancy with dynamic quota management and cell-based isolation, continuous monitoring of both system health and data integrity, and clean abstraction with standard Kafka protocols and CKUs to hide underlying resources.

Related terms: *Apache Kafka*, *Confluent Cloud*, *Confluent Unit for Kafka (CKU)*

Related content

- Kora: The Cloud Native Engine for Apache Kafka
- Kora: A Cloud-Native Event Streaming Platform For Kafka

KRaft

KRaft (or Apache Kafka Raft) is a consensus protocol introduced in Kafka 2.4 to provide metadata management for Kafka with the goal to replace ZooKeeper. KRaft simplifies Kafka because it enables the management of metadata in Kafka itself, rather than splitting it between ZooKeeper and Kafka. As of Confluent Platform 7.5, KRaft is the default method of metadata management in new deployments. For more information, see KRaft overview.

ksqlDB

ksqlDB is a streaming SQL database engine purpose-built for creating stream processing applications on top of Kafka.

logical Kafka cluster (LKC)

A logical Kafka cluster (LKC) is a subset of a physical Kafka cluster (PKC) that is isolated from other logical clusters within Confluent Cloud. Each logical unit of isolation is considered a tenant and maps to a specific organization. If the mapping is one-to-one, one LKC maps to one PKC (a Dedicated cluster). If the mapping is many-to-one, one LKC maps to one of the multitenant Kafka cluster types (Basic, Standard, and Enterprise).

Related terms: *Confluent Cloud*, *Kafka cluster*, *physical Kafka cluster (PKC)*

Related content

- Kafka Cluster Types in Confluent Cloud
- Multi-tenancy and Client Quotas on Confluent Cloud

multi-region cluster (MRC)

A multi-region cluster (MRC) is a single Kafka cluster that replicates data between datacenters across regional availability zones.

multi-tenancy

Multi-tenancy is a software architecture in which a single physical instance is shared among multiple logical instances, or tenants. In Confluent Cloud, each Basic, Standard, and Enterprise cluster is a logical Kafka cluster (LKC) that shares a physical Kafka cluster (PKC) with other tenants. Each LKC is isolated from other L and has its own resources, such as memory, compute, and storage.

Related terms: *Confluent Cloud*, *logical Kafka cluster (LKC)*, *physical Kafka cluster (PKC)*

Related content

- From On-Prem to Cloud-Native: Multi-Tenancy in Confluent Cloud

- Multi-tenancy and Client Quotas on Confluent Cloud

offset

An offset is an integer assigned to each message that uniquely represents its position within the data stream, guaranteeing the ordering of records and allowing offset-based connections to replay messages from any point in time.

Related terms: *consumer offset*, *connector offset*, *offset commit*, *replayability*

offset commit

An offset commit is the process of keeping track of the current position of an offset-based connection (primarily Kafka consumers and connectors) within the data stream.

The offset commit process is not specific to consumers, producers, or connectors. It is a general mechanism in Kafka to track the postion of any application that is reading data.

When a consumer commits an offset, the offset identifies the next message the consumer should consume. For example, if a consumer has an offset of 5, it has consumed messages 0 through 4 and will next consume message 5.

If the consumer crashes or is shut down, its partitions are reassigned to another consumer which initiates consuming from the last committed offset of each partition.

The committed offset for consumers is stored on a Kafka broker. When a consumer commits an offset, it sends a commit request to the Kafka cluster, specifying the partition and offset it wants to commit for a particular consumer group. The Kafka broker receiving the commit request then stores this offset in the `__consumer_offsets` internal topic.

Related terms: *consumer offset*, *offset*

parent cluster

The Kafka cluster that a resource belongs to.

Related terms: *Kafka cluster*

partition

A partition is a unit of data storage that divides a topic into multiple, parallel event streams, each of which is stored on separate Kafka brokers and can be consumed independently.

Partitioning is a key concept in Kafka because it allows Kafka to scale horizontally by adding more brokers to the cluster. Partitions are also the unit of parallelism in Kafka. A topic can have one or more partitions, and each partition is an ordered, immutable sequence of event records that is continually appended to a partition log.

physical Kafka cluster (PKC)

A physical Kafka cluster (PKC) is a Kafka cluster comprised of multiple brokers.

Each physical Kafka cluster is created on a Kubernetes cluster by the control plane. A PKC is not directly accessible by clients.

principal

A principal is an entity that can be authenticated and granted permissions based on

roles to access resources and perform operations. An entity can be a user account, service account, group mapping, or identity pool.

Related terms: *group mapping*, *identity*, *identity pool*, *role*, *service account*, *user account*

private internet

A private internet is a closed, restricted computer network typically used by organizations to provide secure environments for managing sensitive data and resources.

processing time

Processing time is the time when an event is processed or recorded by a system, as opposed to the time when the event occurred on the producing device. Processing time is often used in stream processing to determine the order of events and to perform windowing operations.

producer

A producer is a client application that publishes (writes) data to a topic in an Kafka cluster.

Producers write data to a topic and are the only clients that can write data to a topic. Each record written to a topic is appended to the partition of the topic that is selected by the producer.

Producer API

The Producer API is the Kafka API that allows you to write data to a topic in an Kafka cluster.

The Producer API is used by producer clients to publish data to a topic in an Kafka cluster.

Protobuf

Protobuf (or Protocol Buffers) is an open-source data format used to serialize structured data for storage.

Related terms: *data serialization*, *deserializer*, *serializer*

Related content

- Protocol Buffers
- Getting Started with Protobuf in Confluent Cloud
- Confluent Cloud: Protobuf Serializer and Deserializer
- Confluent Platform: Protobuf Serializer and Deserializer

public internet

The public internet is the global system of interconnected computers and networks that use TCP/IP to communicate with each other.

rebalancing

Rebalancing is the process of redistributing the partitions of a topic among the consumers of a consumer group for improved performance and scalability.

A rebalance can occur if a consumer has failed the heartbeat and has been excluded from the group, it voluntarily left the group, metadata has been updated for a

consumer, or a consumer has joined the group.

replayability

Replayability is the ability to replay messages from any point in time.

Related terms: *consumer offset*, *offset*, *offset commit*

replication

Replication is the process of creating and maintaining multiple copies (or *replicas*) of data across different nodes in a distributed system to increase availability, reliability, redundancy, and accessibility.

replication factor

A replication factor is the number of copies of a partition that are distributed across the brokers in a cluster.

role

A role is a Confluent-defined job function assigned a set of permissions required to perform specific actions or operations on Confluent resources bound to a principal and Confluent resources. A role can be assigned to a user account, group mapping, service account, or identity pool.

Related terms: *group mapping*, *identity*, *identity pool*, *principal*, *service account*

Related content

- Predefined RBAC Roles in Confluent Cloud
- Role-Based Access Control Predefined Roles in Confluent Platform

rolling restart

A rolling restart restarts the brokers in a Kafka cluster with zero downtime by incrementally restarting a Kafka broker after verifying that there are no under-replicated partitions on the broker before proceeding to the next broker.

Restarting the brokers one at a time allows for software upgrades, broker configuration updates, or cluster maintenance while maintaining high availability by avoiding downtime.

Related content

- Rolling restart

schema

A schema is the structured definition or blueprint used to describe the format and structure event messages sent through the Kafka event streaming platform.

Schemas are used to validate the structure of data in event messages and ensures that producers and consumers are sending and receiving data in the same format. Schemas are defined in the Schema Registry.

Schema Registry

Schema Registry is a centralized repository for managing and validating schemas for topic message data that stores and manages schemas for Kafka topics. Schema Registry is built into Confluent Cloud as a managed service, available with the Advanced Stream Governance package, and offered as part of Confluent Enterprise for

self-managed deployments.

The Schema Registry is a RESTful service that stores and manages schemas for Kafka topics. The Schema Registry is integrated with Kafka and Connect to provide a central location for managing schemas and validating data. Producers and consumers to Kafka topics use schemas to ensure data consistency and compatibility as schemas evolve. Schema Registry is a key component of Stream Governance.

Related content

- Confluent Cloud: Manage Schemas in Confluent Cloud
- Confluent Platform: Schema Registry Overview

schema subject

A schema subject is the namespace for a schema in Schema Registry. This unique identifier defines a logical grouping of related schemas. Kafka topics contain event messages serialized and deserialized using the structure and rules defined in a schema subject. This ensures compatibility and supports schema evolution.

Related content

- Confluent Cloud: Manage Schemas in Confluent Cloud
- Confluent Platform: Schema Registry Concepts
- Understanding Schema Subjects

Serdes

Serdes are serializers and deserializers that convert objects and parallel data into a serial byte stream for efficient storage and high-speed data transmission over the wire. Confluent provides Serdes for schemas in Avro, Protobuf, and JSON Schema formats.

Related content

- Confluent Cloud: Formats, Serializers, and Deserializers
- Confluent Platform: Formats, Serializers, and Deserializers
- Serde

serializer

A serializer is a tool that converts objects and parallel data into a serial byte stream. Serializers work with deserializers (known together as Serdes) to support efficient storage and high-speed data transmission over the wire. Confluent provides serializers for schemas in Avro, Protobuf, and JSON Schema formats.

Related content

- Confluent Cloud: Formats, Serializers, and Deserializers
- Confluent Platform: Formats, Serializers, and Deserializers

service account

A service account is a non-person entity used by an application or service to access resources and perform operations.

Because a service account is an identity independent of the user who created it, it can be used programmatically to authenticate to resources and perform operations without the need for a user to be signed in.

Related content

- Service Accounts for Confluent Cloud

service quota

A service quota is the limit, or maximum value, for a specific Confluent Cloud resource or operation that might vary by the resource scope it applies to.

Related content

- Service Quotas for Confluent Cloud

single message transform (SMT)

A single message transform (SMT) is a transformation or operation applied in realtime on an individual message that changes the values, keys, or headers of a message before being sent to a sink connector or after being read from a source connector. SMTs are convenient for inserting fields, masking information, event routing, and other minor data adjustments.

single sign-on (SSO)

Single sign-on (SSO) is a centralized authentication service that allows users to use a single set of credentials to log in to multiple applications or services.

Related terms: *authentication*, *group mapping*, *identity provider*

Related content

- Single Sign-On for Confluent Cloud

sink connector

A sink connector is a Kafka Connect connector that publishes (writes) data from a Kafka topic to an external system.

source connector

A source connector is a Kafka Connect connector that subscribes (reads) data from a source (external system), extracts the payload and schema of the data, and publishes (writes) the data to Kafka topics.

standalone

Standalone refers to a configuration in which a software application, system, or service operates independently on a single instance or device. This mode is commonly used for development, testing, and debugging purposes.

For Kafka Connect, a standalone worker is a single process responsible for running all connectors and tasks on a single instance.

static egress IP address

A static egress IP address is an IP address used by a Confluent Cloud managed connector to establish outbound connections to endpoints of external data sources and sinks over the public internet.

Related content

- Use Static IP Addresses on Confluent Cloud for Connectors and Cluster Linking
- Static Egress IP Addresses for Confluent Cloud Connectors

Stream Catalog

Stream Catalog is a pillar of Confluent Cloud Stream Governance that provides a centralized inventory of your organization's data assets that supports data governance and data discovery. With Data Portal in Confluent Cloud Console, users can find event streams across systems, search topics by name or tags, and enrich event data to increase value and usefulness. REST and GraphQL APIs can be used to search schemas, apply tags to records or fields, manage business metadata, and discover relationships across data assets.

Related content

- Stream Catalog on Confluent Cloud: User Guide to Manage Tags and Metadata
- Stream Catalog in Streaming Data Governance (Confluent Developer course)

Stream Designer

Stream Designer is a graphical tool that lets you visually build streaming data pipelines powered by Apache Kafka.

Related content

- Stream Designer for Confluent Cloud

Stream Governance

Stream Governance is a collection of tools and features that provide data governance for data in motion. These include data quality tools such as Schema Registry, schema ID validation, and schema linking; built-in data catalog capabilities to classify, organize, and find event streams across systems; and stream lineage to visualize complex data relationships and uncover insights with interactive, end-to-end maps of event streams.

Taken together, these and other governance tools enable teams to manage the availability, integrity, and security of data used across organizations, and help with standardization, monitoring, collaboration, reporting, and more.

Related terms: *Stream Catalog*, *Stream Lineage*

Related content

- Stream Governance on Confluent Cloud

stream lineage

Stream lineage is the life cycle, or history, of data, including its origins, tranformations, and consumption, as it moves through various stages in data pipelines, applications, and systems.

Stream lineage provides a record of data's journey from its source to its destination, and is used to track data quality, data governance, and data security.

Related terms: **Data Portal**, *Stream Governance*

Related content

- Stream Lineage on Confluent Cloud

stream processing

Stream processing is the method of collecting event stream data in real-time as it arrives, transforming the data in real-time using operations (such as filters, joins, and aggregations), and publishing the results to one or more target systems.

Stream processing can be used to analyze data continuously, build data pipelines, and

process time-sensitive data in real-time. Using the Confluent event streaming platform, event streams can be processed in real-time using Kafka Streams, Kafka Connect, or ksqlDB.

Streams API

The Streams API is the Kafka API that allows you to build streaming applications and microservices that transform (for example, filter, group, aggregate, join) incoming event streams in real-time to Kafka topics stored in a Kafka cluster.

The Streams API is used by stream processing clients to process data in real-time, analyze data continuously, and build data pipelines.

Related content

- Introduction Kafka Streams API

throttling

Throttling is the process Kafka clusters in Confluent Cloud use to protect themselves from getting to an over-utilized state. Also known as backpressure, throttling in Confluent Cloud occurrs when cluster load reaches 80%. At this point, applications may start seeing higher latencies or timeouts as the cluster must begin throttling requests or connection attempts.

unbounded stream

An unbounded stream is a stream of data that is continuously generated in real-time and has no defined end. Examples of unbounded streams include stock prices, sensor data, and social media feeds.

Processing unbounded streams requires a different approach than processing bounded streams. Unbounded streams are processed incrementally as data arrives, while bounded streams are processed as a batch after all data has arrived. Kafka Streams and Flink can be used to process unbounded streams.

Related terms: *bounded stream*, *stream processing*, *unbounded stream*

under replication

Under replication is a situation when the number of in-sync replicas is below the number of all replicas.

Under Replicated partitions can occur when a broker is down or cannot replicate fast enough from the leader (replica fetcher lag).

user account

A user account is an account representing the identity of a person who can be authenticated and granted access to Confluent Cloud resources.

Related content

- User Accounts for Confluent Cloud

watermark

A watermark in Flink is a marker that keeps track of time as data is processed. A watermark means that all records until the current moment in time have been “seen”. This way, Flink can correctly perform tasks that depend on when things happened, like calculating aggregations over time windows.

Related content

- Time and Watermarks

Was this doc page helpful?

Give us feedback

Do you still need help?

Confluent support portal

Ask the community

Be the first to get updates and new content

Email

SIGN UP

By clicking "SIGN UP" you agree that your personal data will be processed in accordance with our Privacy Policy.