CONFLUENT

aws

TECHNICAL OVERVIEW

Confluent Streaming Pipelines to Cloud Data Warehouses with AWS

Introduction

To harness the full value of real-time data, organizations are increasingly migrating from existing on-prem data analytics platforms (Teradata, Cloudera, etc.) to cloud-based data warehouses (DW). This process, however, can be long, arduous, and expensive. Some companies have mission-critical ETL jobs running that are very expensive to migrate directly into a cloud DW, or cannot be suddenly disrupted. Other companies are required to work across multiple cloud platforms.

Migrating your on-prem data warehouse or connecting DWs across clouds to unlock real-time analytics and AI/ML innovation doesn't need to be a multiyear lift and shift. Using Confluent's data warehouse streaming pipelines, enterprises can create a bridge across clouds and on-prem environments to start streaming data immediately while unlocking additional value. Pairing Confluent with **AWS** helps you:

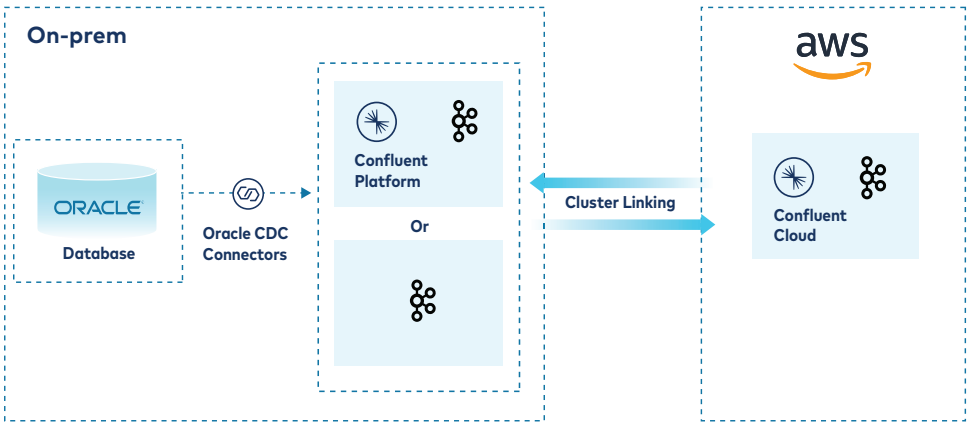
1. **Connect** any data source in real time to Confluent, across hybrid and multicloud environments, to stream more data to and from your Amazon Redshift DW
2. **Process** data streams in flight with Confluent's native stream processing – joining, enriching, and transforming data before they land in your DW
3. **Govern** streaming data to ensure data quality, security, and compliance

Confluent streaming data pipelines enable real-time analytics and decision-making, boost productivity with self-service data, and reduce the total cost of ownership (TCO) and time-to-value of hybrid and multicloud data pipelines. Furthermore, Confluent enables businesses to **build an integrated data analytics platform with AWS** with unified security, billing, and management – all accessible via AWS Marketplace.

This document details different ways to integrate data to the cloud via Confluent, including deployment approach, security considerations, as well as integrating on-prem databases with Confluent Cloud. If you are looking to understand the business value of our solution, please visit our Data Warehouse streaming pipelines webpage.

Stream on-prem data to the cloud reliably and securely with Confluent Cloud

Confluent provides two primary mechanisms for streaming data to and from the cloud. Data already in Kafka can be securely copied from Kafka on-premises, whether it is OSS Apache Kafka or Confluent Platform, to Confluent Cloud via Cluster Linking. Alternatively, data can be streamed into Kafka using a Fully Managed Connector on Confluent Cloud, a self-provided Custom Connector on Confluent Cloud, or a connector running in a self-managed instance of Kafka Connect.

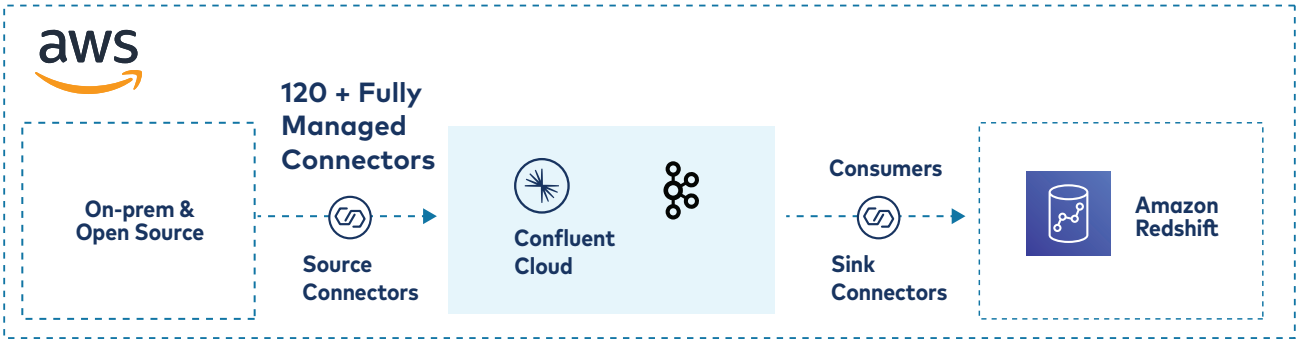


Data Transformation & Filtering

Data can be processed in flight with ksqlDB and Confluent Cloud fully managed connectors. This enables the reduction of throughput and overall data pipeline TCO.

Extend Advanced Analytics Capabilities

Connectors available within the Kafka ecosystem allow developers to easily and securely add data from open-source, on-prem data sources to cloud-native analytics projects via language APIs for .NET, C# and REST. Data in Kafka can be easily accessed by data engineers and developers and allows integrated analysis leveraging big data tools such as Spark, leveraging languages such as Python and R. Access to event-level data enables business stakeholders and analysts to perform real-time data analysis and queries on event-level data. Confluent Cloud's suite of 120+ pre-built connectors allows enterprises to quickly configure connectors to popular cloud-native services, reducing operational overhead.



CASE STUDY: ACMECO—RETAILER

Let's look at how a theoretical retailer, we'll call them AcmeCo, sends on-premises data from Oracle database to AWS for data consolidation and advanced analytics with Amazon Redshift.

This document will walk through the technical considerations and recommendations when implementing the following steps:

Step 1: Prepare the Confluent Environment

- Plan your Confluent installation
- Install Confluent Platform on an on-premise Kubernetes environment
- Prepare the Oracle Transactional database for CDC
- Extract customer data from Oracle Transactional database using the Oracle CDC Source Premium Connector

Step 2: Data Migration/Consolidation—On-Prem to Cloud

- Stream data from Confluent Platform to Confluent Cloud with Cluster Linking

Step 3: Data Transformation & Filtering

- Join data from multiple sources and filter before sending to Confluent Cloud with ksqlDB

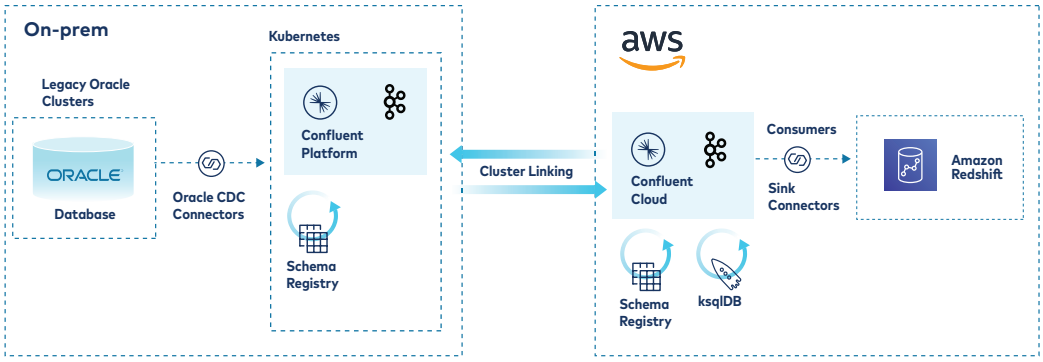
Step 4: Modernize with Additional Data Sources and Real-time Advanced Analytics

- Load the cleaned and merged data into Amazon Redshift, using the fully managed Amazon Redshift Sink Connector.

SOLUTION DIAGRAM

Today, most enterprise data warehouses are installed on-premises, and in this scenario, we are walking through installing Confluent Platform on-premises, to initiate a data pipeline to the cloud. We will need to install Confluent Platform on-premises, as well as create a Dedicated cluster on Confluent Cloud. We will need networking access between the Kubernetes cluster and a host with Java, Confluent Platform components, the Confluent CLI and Confluent Cloud CLI to allow us to programmatically interact with clusters on both the on-prem and cloud based instances via the Confluent APIs. Access to the Control Center UI can be enabled by either port-forwarding or configuring a load balancer with the external IP address and ports of the Control Center service.

On-prem to AWS Pipeline
Oracle CDC Connector to Amazon Redshift



This solution has been built using the following components:

COMPONENT	DESCRIPTION
Confluent Platform 7.3	Confluent Platform addresses requirements of modern platform streaming applications. It includes: Confluent Control Center, for end-to-end monitoring and management. Confluent Replicator, for managing multi-datacenter deployments. Confluent Auto Data Balancer, for optimizing resource utilization and easy scalability. Tiered Storage, for unlimited retention. Multi- Region Clusters, for high availability. Schema Validation, for data governance. Kubernetes Operator, for containerized installation and operation. Ansible Playbooks and Templates, for non-containerized installation and operation. And Role-Based Access Control (RBAC), Structured Audit Logs, and Secret Protection, for enterprise grade security.
Confluent Cloud Dedicated Cluster	Confluent Cloud is fully managed Kafka with no infrastructure to provision, monitor, or manage. Load is automatically distributed across brokers, consumer groups automatically rebalance when a consumer is added or removed, the state stores used by applications using the Kafka Streams APIs are automatically backed up to Confluent Cloud, and failures are automatically mitigated. Basic and Standard clusters are multi-tenant clusters. Dedicated runs on per- customer dedicated compute resources and supports the most features and custom options.
Confluent CLI 3.5.0	Confluent CLI enables management of Confluent Platform and Confluent Cloud clusters with simplified and unified command line interface.
Confluent for Kubernetes 2.5.0	Confluent for Kubernetes (CFK) is a cloud-native control plane for deploying and managing Confluent in your private cloud environment. CFK leverages Kubernetes-native API approach to configure, deploy, and manage Confluent Platform components.

Preparing the Confluent Environment

Plan your Confluent Installation

Choosing the right deployment model is critical for the success and scalability of the Confluent event streaming platform. You want to provide the right hardware (and cloud instances) for each use case to ensure that the system reliably provides high-throughput and low-latency data streams. You can access the Confluent Platform Reference Architecture for considerations, guidelines and recommendations for deploying Apache Kafka and Confluent Platform in Production. Once you have settled on a deployment model, you can leverage the Sizing Calculator for Apache Kafka and Confluent Platform. Here you can obtain suggestions for VM types and storage for Kafka components, based on anticipated throughput, read fanout and retention, as well as calculate how many partitions a single topic will need.

Confluent Cloud Security Controls

Data security is a primary concern when it is transported in and out of Confluent Cloud as well as when the data is persisted to disk. Confluent provides industry standard and audited protection mechanisms to ensure customers can confidently store data in Confluent Cloud. To further protect data, network-level isolation is available through VPC/VNet isolation and private networking options.

Encryption in Transit

Encryption using TLS 1.2 is required for all client connections to Confluent Cloud and HTTP Strict Transport Security (HSTS) is enabled.

Encryption at Rest

Data at rest uses essentially the same default transparent AES-256 based disk encryption across AWS. The transparent disk encryption is well suited for Kafka since Kafka serializes data into raw bytes before it is being persisted to disk.

Encryption Key Management

Confluent Cloud uses one master key per account/project/tenant using the default cloud provider disk encryption mechanism described above. For Dedicated clusters, Confluent supports Self- Managed Keys, a.k.a. bring-your-own-key (BYOK) encryptions, in AWS.

Secure Networks with AWS VNet Peering or PrivateLink

VPC peering between Confluent Cloud can be established by providing Confluent with the CIDR range and adding the peering via the Confluent Cloud UI with parameters for AWS Account ID, VPC ID, and VPC CIDR. Confluent Cloud Console components, like topic management and ksqlDB, are set up with private endpoints that are not publicly reachable. You must configure internal access to these components.

AWS PrivateLink offers one-way connectivity from your VPC to a Confluent Cloud cluster, ensuring secure access by limiting traffic to only that which originates from within your VPC, without any coordination of CIDR block ranges. PrivateLink connections to Confluent Cloud can only be made from VNets in registered customer AWS subscriptions. With PrivateLink, Confluent exposes service alias(es) for each new cluster, for which customers can create corresponding private endpoints in their own AWS VPCs. Some Confluent Cloud Console components, like topic management, use cluster endpoints that are not publicly reachable when PrivateLink is enabled for a cluster. Unlike VPC peering, AWS PrivateLink does not require the use of a proxy to forward traffic from your browser through your VPC to the Confluent Cloud cluster. You must configure your network to route requests for these components over the PrivateLink connection. A Dedicated Cluster is required for all private networking options.

Confluent also offers AWS Transit Gateway as a secure network option.

Compliance

Confluent maintains a number of compliance certifications and supports industry security standards, such as SOC 1, 2, and 3, ISO 27001, PCI DSS, CSA Star Level 1 and HIPAA. For additional information or to contact the compliance team please refer to Confluent's Trust and Security Page. Confluent's Cloud Data Processing Addendum addresses both CCPA and GDPR requirements and you can view how Confluent handles personal data via Confluent's Privacy Policy.

Install Confluent Platform to an on-premises Kubernetes environment

The Confluent for Kubernetes Quickstart is available as a starting point to install the following Confluent Platform components: Confluent Operator (1), Connect(1), Control Center (1), Brokers (3), Schema registry (1) and ZooKeepers (3). The confluent-platform.yaml file can be easily updated with configuration overrides to customize Confluent Platform components for bespoke installations. For example, this deployment will require Kafka server overrides to enable cluster linking as well as Connect overrides to deploy a Docker image of Connect stored within a cloud hosted container registry service. More complex scenarios are available via Confluent Kubernetes-Examples github repo.

To prepare for deployment, verify Kubernetes environment supports persistent volume claims and install kubectl and Helm 3. To deploy, create a namespace 'Confluent' to host pods, then set it as your default namespace. First, install Confluent operator using Helm, then Confluent-for-Kubernetes, then apply Confluent Platform customizations via the confluent-platform.yaml.

The available Helm 3 charts reference the namespace 'confluent', create namespace confluent and set current context.

```
kubectl create namespace confluent
kubectl config set-context —-current --namespace confluent
```

Add the Confluent Helm packages to Helm repo

```
helm repo add confluentinc https://packages.confluent.io/helm
helm repo update
```

Deploy Confluent Operator Pod

```
helm upgrade --install confluent-operator confluentinc/confluent-for-kubernetes
```

Deploy Confluent Platform Pods

```
kubectl apply -f
https://raw.githubusercontent.com/confluentinc/confluent-kubernetes-examples/master/quickstart-deploy/confluent-platform.yaml
```

Verify Pods are deployed

```
kubectl get pods
```

Post installation, verify connectivity with your host with Confluent components, and access Control Center via port-forwarding or load balancer.

Prepare the Oracle database for CDC

Please refer to Database prerequisites for guidelines.

Extract customer data from Oracle DB using the Oracle CDC Source Premium Connector

Confluent's ecosystem of 120+ connectors unlock data from a variety of sources, whether on-premises or multi-cloud, and proprietary or open source. On-premises relational database systems often hold highly critical enterprise transaction workloads. Organizations often find that they want to use the data that it stores elsewhere, such as their analytics platform or for driving real-time applications. Change data capture (CDC) solves this challenge by efficiently identifying and capturing data that has been added to, updated, or removed from tables.

Kafka Connect and the ecosystem of available connectors makes change data available to the rest of the organization. Whether in single node (standalone) or scaled to an organization-wide service (distributed), Connect provides lower time to production. Fully managed connectors hosted on Confluent Cloud abstract away all the operational burden of Connect cluster management. Most fully managed connectors can be configured and launched within minutes, with simply the connection details, credentials and a few additional parameters.

The Oracle CDC Source premium connector is a self-managed connector. Self-managed connectors can be manually installed on a Connect instance that you maintain yourself. To add additional connectors to your Confluent on Kubernetes clusters, connectors can be downloaded and added to the default Connect Docker image, then included during Confluent Platform deploy. This can be done by creating a Dockerfile, building the docker image: `docker build -t <name of image>`, and copying the Docker image to a container registry service where it can be referenced via the confluent-platform.yaml file, `spec → image → application→ <url of repo>`.

```
spec:
  replicas: 1
  image:
    application: [AWS Account #}.dkr.ecr.us-west-2.amazonaws.com/dwm-connect:latest
```

Once the Connect instance is up, a new connector tile will be available for selection within the Confluent Control Center UI. After selecting, the connector can be configured by adding in connector parameters similar to below, or you can import the configuration file from an existing running connector.

```
name = OracleCdcSourceConnectorConnector_0
connector.class = io.confluent.connect.oracle.cdc.OracleCdcSourceConnector
tasks.max = 1
key.converter = org.apache.kafka.connect.storage.StringConverter
value.converter = io.confluent.connect.avro.AvroConverter
topic.creation.groups = redo
oracle.server = dwm-demo.cjg294eidfaj.us-west-2.example.com
oracle.port = 1521
oracle.sid = ORCL
oracle.username = <insert username>
oracle.password = <insert password>
start.from = SNAPSHOT
redo.log.row.fetch.size = 1
max.batch.size = 100
lob.topic.name.template = ${databaseName}.${schemaName}.${tableName}.${columnName}
table.inclusion.regex = ORCL.ADMIN.*
table.topic.name.template = ${databaseName}.${schemaName}.${tableName}
topic.creation.default.partitions = 5
topic.creation.redo.partitions = 1
topic.creation.redo.include = oracle-redo-log-topic
topic.creation.default.replication.factor = 3
topic.creation.redo.retention.ms = 1209600000
topic.creation.redo.replication.factor = 3
topic.creation.default.cleanup.policy = compact
topic.creation.redo.cleanup.policy = delete
value.converter.schema.registry.url = http://schemaregistry.confluent.svc.cluster.local:8081
```

There are a few config parameters to highlight:

- `table.inclusion.regex` identifies the regular expression that identifies tables that this connector will capture
- `table.topic.name.template` specifies the rule for the names of the topics to which the events are written
- Since we have a few tables, including a CLOB data type, we also use `lob.topic.name.template` to specify the names of the topics where LOB values are written
- We also have a few columns with NUMERIC data types in Oracle Database, and with `"numeric.mapping":"best_fit"`; the connector will store them as an integer, a float, or a double in Kafka topics rather than as arbitrarily high precision numbers

The last group of configuration parameters uses the functionality added in Apache Kafka 2.6 to define how Connect creates topics to which the source connector writes. The following parameters create a redo log topic called oracle-redo-log-topic with one partition and create other topics (used for table- specific change events) with five partitions.

When the connector is running, it logs that it cannot find a redo log topic if there are no changes (INSERT, UPDATE, or DELETE) in the database five minutes after it completes a snapshot. To prevent this from happening, you can `increase redo.log.startup.polling.limit.ms`, or you can create a redo log topic before running the connector with `config.cleanup.policy=delete`.

Data Transformation & Filtering

ETL tools are usually required to map source to destination data and perform data transformation. ksqlDB offers a single solution for collecting streams of data, enriching them, and serving queries on new derived streams and tables. Developers can build real-time applications with the same ease and familiarity as building traditional apps on a relational database – all through a familiar, lightweight SQL syntax. ksqlDB is built on top of Kafka Streams, a lightweight, powerful Java library for enriching, transforming, and processing real-time streams of data. Having Kafka Streams at its core means ksqlDB is built on well-designed and easily understood layers of abstractions.

Source Data stored in Oracle Database

Field Name	Data Type
CUSTOMER_ID	INT
CUSTOMER_NAME	STRING
CUSTOMER_LOYALTY	STRING
CUSTOMER_EMAIL	STRING
CUSTOMER_STREET	STRING
CUSTOMER_ZIPCODE	STRING
CUSTOMER_COUNTRY	STRING
CUSTOMER_PREFCC	STRING

Join on customer_id

Source Data stored in POS Database

Field Name	Data Type
customer_id	INT
product_name	STRING
order_number	INT
order_date	STRING
cost	INT

After pulling data from separate sources into Confluent, data can be joined from both sources and pre- filtered using ksqlDB to reduce the amount of data being sent to the cloud and reduce ingress/egress expenses. Example queries enriching order data with customer details by joining on customer_id below.

Create Stream joining Customers and Orders data stored in Confluent Cloud

```
CREATE STREAM enriched_orders WITH
(VALUE_FORMAT='AVRO') AS SELECT c.customer_id,
c.customer_name,  c.customer_loyalty,
o.product_name, o.order_number, o.cost,
o.order_date FROM customers c INNER JOIN orders
o WITHIN 24 HOURS GRACE PERIOD 60 MINUTES
ON (o.customer_id = c.customer_id) emit changes;
```

Create Table from Enriched_Orders Stream

```
CREATE TABLE orders_enhanced WITH
(KAFKA_TOPIC='orders_enhanced',
KEY_FORMAT+'JSON', VALUE_FORMAT+'AVRO') AS
SELECT c_customer_id, customer_name,
COUNT(ordeer_number) AS num_orders, SUM(cost)
AS total_spend,
MAX(STRINGTOTIMESTAMP(ORDER_DATE,
'MM/dd/yyyy')) AS last_ordeer FROM
ENRICHED_ORDERS  GROUP BY c_customer _id,
customer_name HAVINNG
MAX(STRINGTOTIMESTAMP(ORDER_DATE,
'MM/dd/yyyy'))
STRINGTOTIMESTAMP('01/01/2017',
'MM/dd/yyyy');
```

Replicate data from Confluent Platform to Confluent Cloud with Cluster Linking

Cluster Linking enables topic data sharing between hybrid environments, as well as expedites migration of on-premises to Confluent Cloud clusters. Consumers on the destination cluster can read from local, read-only, mirrored topics to read messages produced on the source cluster. If an original topic on the source cluster is removed for any reason, you can stop mirroring that topic, and convert it to a read/write topic on the destination. Unlike, Replicator and MirrorMaker2, Cluster Linking does not require running Connect to move messages from one cluster to another, ensuring that the offsets are preserved from one cluster to another. We call this "byte-for-byte" replication. Whatever is on the source, will be mirrored precisely on the destination cluster. This simplifies moving from an on- premises Kafka cluster to a Confluent Cloud cluster. The native offset preservation you get by leveraging Confluent Server on the brokers makes this much easier to do with Cluster Linking than with other Connect based methods. A Dedicated Cluster is required for Cluster Linking.

There are two steps involved in on-premises to Confluent Cloud cluster linking. First, a source-based cluster link will be created. Then, a mirror topic will be created on the destination cluster hosted by Confluent Cloud and associated with the cluster link. Please reach out to Confluent for implementation questions.

Load the cleaned and merged data into Amazon Redshift, using the fully managed Redshift Sink Connector

Once on-premises data is in Confluent Cloud, Confluent's suite of fully managed connectors provide streamlined access to building pipelines to AWS native services. With fully managed connectors, Confluent Cloud has optimized the process for configuring connectors by reducing the effort needed for set-up. Confluent Cloud resources can also be configured and associated with cloud platform subscriptions, and provide unified security, billing, and management. Once in Confluent Cloud, live streaming data that feeds into live streaming analytics can also be served to developers enabling them to build real-time apps or services.

Amazon Redshift Fully Managed Connector on Confluent Cloud

Amazon Redshift is a fully managed, petabyte-scale data warehouse service in the cloud. Users can start with just a few hundred gigabytes of data and scale to a petabyte or more. This enables customers to use their data to acquire new insights for your business and customers. Confluent Cloud's fully-managed AmazonRedshift connector makes it easier than ever to pull in data from disparate data, aggregate, and insert to the platform.

Topics in Confluent Cloud can be quickly loaded into Amazon Redshift databases, with dynamically created tables and schema by setting `auto.create='true'` and `auto.evolve='true'`. To configure the connector within Confluent Cloud, pass in the following parameters: domain, database name, and credentials.

Add Amazon Redshift Sink Connector

```
{
  "name": "RedshiftSinkConnector_0",
  "config": {
    "connector.class": "RedshiftSink",
    "name": "RedshiftSinkConnector_0",
    "input.data.format": "AVRO",
    "kafka.api.key": "****************",
    "kafka.api.secret":
"**********************************************************************",
    "topics": "orders_enhanced",
    "aws.redshift.domain": "redshift",
    "aws.redshift.port": "5349",
    "aws.redshift.user": "awsuser",
    "aws.redshift.password": "*********",
    "aws.redshift.database": "enchanced_orders",
    "db.timezone": "America/Los_Angeles",
    "auto.create": "true",
    "auto.evolve": "true",
    "tasks.max": "1"
  }
}
```

CONSIDERATIONS

- At minimum, INSERT access privilege is required for this connector. See Amazon Redshift Grant. If delete.enabled=true, DELETE access privilege is required.
- The topic you want to sink into Amazon Redshift must be schema backed, so please use JSON-SR, AVRO, or ProtoBuff
- At least one delivery—This connector guarantees that records from the Kafka topic are delivered at least once.
- Performance—The Amazon Redshift Sink Connector supports running one or more tasks. You can specify the number of tasks in the tasks.max configuration parameter. This can lead to huge performance gains when multiple files need to be parsed.

Stream Governance

Stream Governance provides businesses with a modern governance solution, built for data streaming, that allows them to operate at scale around a central nervous system of real-time data. Individual users across teams have access to the data they need through self-service discovery in a system with the controls required to ensure long-term data compatibility, risk management, and regulatory compliance.

Stream Quality

Schema Registry allows teams to define and enforce universal data standards that enable scalable data compatibility while reducing operational complexity. Schemas shared across cloud and hybrid environments sync in real time with Schema Linking. Avro, JSON and Protobuf serialization formats are supported.

Schema Validation, enabled at the topic-level, ensures broker/registry coordination by verifying that schemas tied to incoming messages are both valid and assigned to the specific destination topic in order to publish.

Stream Catalog

Available through the Confluent Cloud UI, REST API, and GraphQL API, Stream Catalog allows users across teams to collaborate within a centralized, organized library designed for sharing, finding, and understanding the data needed for projects. Built and organized through automatic metadata collection, this catalog includes topics, schemas, fields in schemas, clusters, connectors, and more.

Stream Lineage

Stream Lineage provides a GUI of data streams and data relationships with both a bird's eye view and drill-down magnification for answering questions like: Where did data come from? Where is it going? Where, when, and how was it transformed? When and how was a pipeline last updated? Live metrics and metadata inspection embedded directly within lineage graphs enables teams to learn quickly and make more informed decisions. Historical, point-in-time insights and search across graphs simplify maintenance of mission-critical data streams.

Please see Confluent's Stream Governance docs to learn more.

For more information on building streaming data pipelines with Confluent and AWS, please visit our Data Warehouse streaming pipelines webpage or read our blog.

Resources

Get started for free on Confluent Cloud with $400 free credits at sign-up.
Try **FREE** on AWS Marketplace

Connectors on Confluent Hub

MySQL CDC Source Connector

SQL Server CDC Source Connector

PostgreSQL CDC Source Connector

Oracle CDC Source Connector

Amazon Redshift Sink Connector

See all

Source Code on GitHub

GitHub Repo

Documents and Training

Set up a Confluent Cloud Account

Set up a Confluent Platform Account