



Confluent Cloud Overview

CC Bootcamp

Created by Sven Erik Knop
Staff Technical Trainer



Course Description

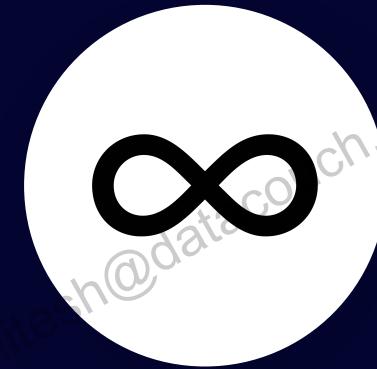
1. Confluent Cloud Overview
2. Labs
 - 2.1. Basic Cluster
 - 2.2. Clients
 - 2.3. ACLs and RBAC
 - 2.4. Managed Connectors
 - 2.5. OIDC
 - 2.6. Audit Logs and Metrics
 - 2.7. Enterprise Cluster
 - 2.8. Dedicated Cluster
 - 2.9. Audit Logs via Cluster Linking
 - 2.10. Cluster and Schema Linking
 - 2.11. Confluent Terraform Provider

nitesh@datacouch.io

Mission Statement



Cloud Native



Complete



Everywhere



What is Confluent Cloud?

- Managed Confluent Kafka Cluster
 - Deployed in AWS, GCP & Azure
 - Shared and Dedicated Cluster

hitesh@datacouch.io

- Scalable through adjustable cluster size (dedicated)
- Internal user management
 - OAuth/OIDC also supported
- Additional services
 - Schema governance
 - ksqlDB
 - Managed connectors
 - Audit Log and Metrics
 - Cluster Linking
 - Schema Linking
 - Flink SQL (preview)



Cloud providers and regions



Amazon AWS

US: 3
Canada: 1
South America: 1

Asia-Pacific: 8 + 1

Europe: 7

Middle East: 2

Africa: 1



GCP

US: 6
Canada: 2
South America: 2

Asia-Pacific: 9 + 2

Europe: 10

Israel: 1



Azure

US: 6
Canada: 1
South America: 1

Asia-Pacific: 5 + 1

Europe: 8

Middle East: 2

Africa: 1



Cluster types

Basic (Shared)

99.5% uptime SLA (2 d/year)

5 TB Storage

Single Zone only

Can be upgraded to Standard (single zone only)

You only pay for the ingress, egress, partitions, and storage

No upfront cost

Standard (Shared)

1 AZ: 99.95% uptime SLA (4 h/year)

3 AZ: 99.99% uptime SLA (1 h/year)

Unlimited Storage

Single & Multi-zone

RBAC and OAuth

Starting at \$1.50 / hr

Enterprise (Shared)

Same as Standard, plus

Private Networking ([Private Link](#))

22,500 client connections

Starts at \$4.50 / hour

Dedicated

Same as Standard, plus

1 GB/s Ingress & Egress

Can be scaled up and down

[Different private networking options](#)

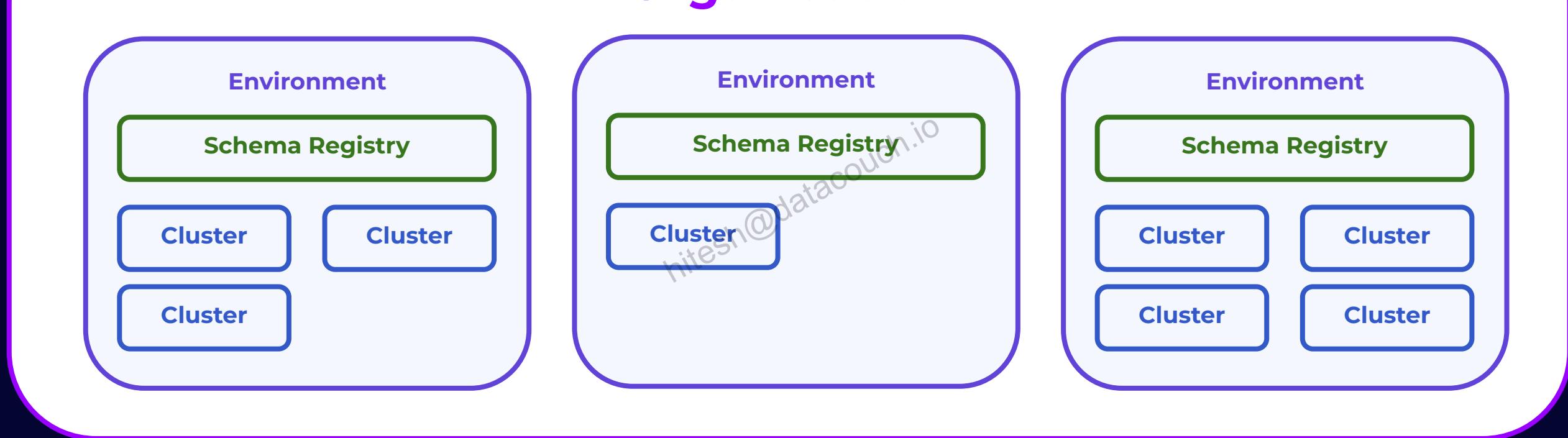
Self-managed keys

Starts at \$2.66 / hr



Confluent Cloud Structure

Organisation





Confluent Cloud Constructs



Organization

- Highest level abstraction
- Billing
- Support
- SSO
- User Management
- Service Accounts
- Contains Multiple Environments



Environment

- Belongs to an organization
- Generally a logical separation of environments+region
 - dev/test/prod
 - us-west-2-prod
- Schema Registry
- Contains Multiple Clusters

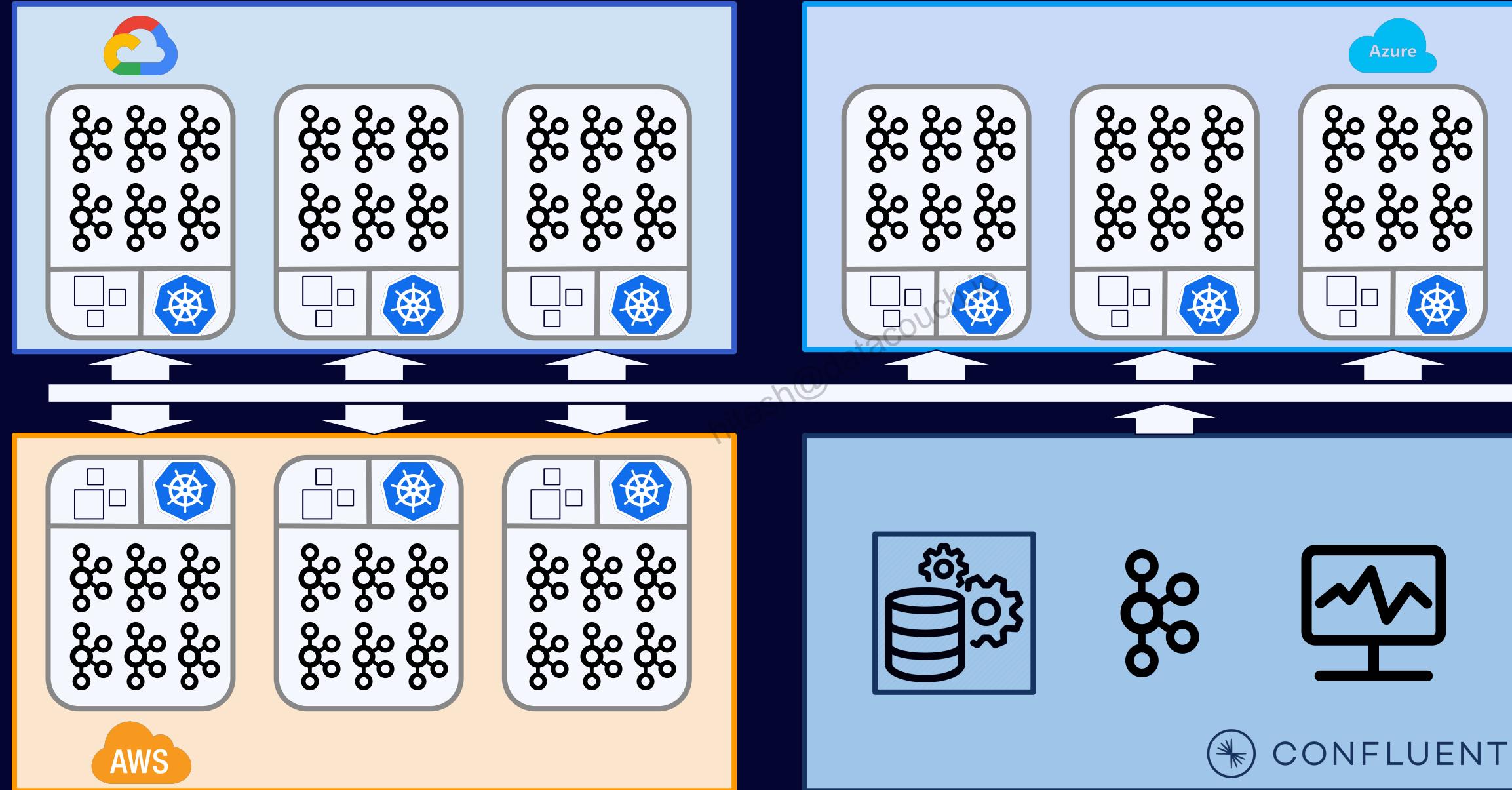


Cluster

- Belongs to an environment
- Topics
- Connectors
- ksqlDB
- Consumer Groups
- API Keys
- ACLs

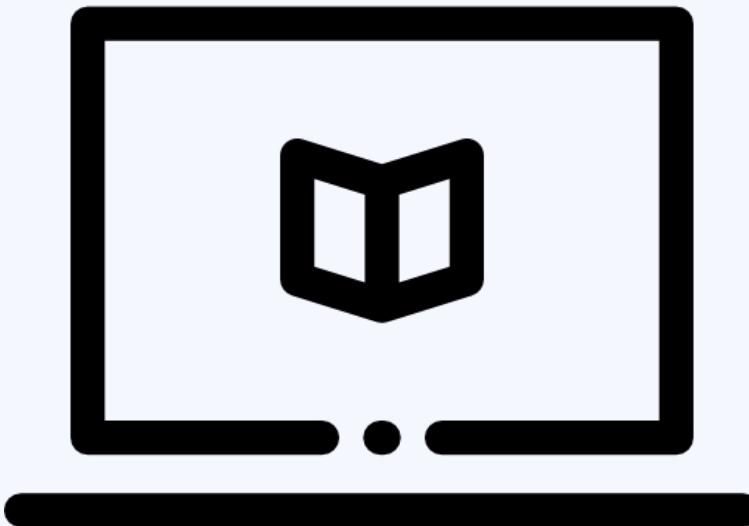


Mothership and Cloud Providers





Labs overview



The lab will use a **Confluent Cloud** organisation.

- This has been either assigned to you
- Or you can create your own environment
 - For **Confluent** employees, create a new organisation using your email address with a subaddress, for example

`username+private@confluent.io`

- For **partners**, please create your own Confluent Cloud organisation. You will get \$400 credits, enough for this bootcamp.



Lab 1: Basic Cluster

nitesh@datacouch.io



(Managed) Schema Registry

https://www.concouch.io



Stream Governance Packages

- Shared Kafka cluster instance
 - Essentials package only available in certain regions
- Single Schema Registry for a whole environment
- Only via public Internet
- Separate API Keys
- Essentials limit of 1,000 schema versions
 - \$0.002 / schema-hr after that
 - Can be upgraded to Advanced
- Advanced limit of 20,000 schema versions

hitesh@datacouch.io

Stream Governance Packages

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

Essentials

The fundamentals for getting started.

Stream Quality
Schema Registry & validation

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

Stream Catalog
Data organization & discoverability

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

Stream Lineage
Data origin & tracking

- Real-time data streams lineage

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

[Begin configuration](#)

Starting at
FREE

Upgrade to Advanced at any time

Advanced

Enterprise-ready controls for data in motion.

Stream Quality
Schema Registry & validation

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

Stream Catalog
Data organization & discoverability

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

Stream Lineage
Data origin & tracking

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

[Begin configuration](#)

Starting at
\$1 /hr

The full Stream Governance feature set



Enable Stream Governance Essentials

The screenshot illustrates the Stream Governance interface. On the left, there are three service icons: AWS, Google Cloud, and Microsoft Azure. The main interface shows a dropdown menu for selecting a region. An arrow points from the 'N. Virginia (us-east-1)' option in the dropdown to a central modal window titled 'Upgrade required'. This modal states: 'This region is only available in the Stream Governance Advanced package.' It details the 'Stream Governance Advanced' package: Base cost (\$1.1/hr), Included schemas (20,000), and SLA (99.95%). It includes 'Cancel' and 'Apply changes' buttons. To the right of the modal, the 'Stream Governance package' section is shown, indicating it's set to 'Advanced' for 'Amazon Web Services | eu-west-1'. Below this, sections for 'Schemas', 'Tags', and 'Business metadata' are visible. Further down, the 'Stream Governance API' section displays the endpoint: `https://psrc-q55z6.eu-west-1.aws.confluent.cloud`. A note at the bottom suggests reading [usage examples](#).

Region*

United States

- Ohio (us-east-2) +\$0 /hr
- N. Virginia (us-east-1) Upgrade required
- Oregon (us-west-2) Upgrade required

Europe

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

Upgrade required

This region is only available in the Stream Governance Advanced package.

Stream Governance Advanced

Base cost	\$1.1 /hr
Included schemas	20,000
SLA	99.95%

Stream Governance package

Advanced

Amazon Web Services | eu-west-1

Schemas Tags Business metadata

Stream Governance API

Schema Registry and Stream Catalog API

Endpoint

```
https://psrc-q55z6.eu-west-1.aws.confluent.cloud
```

If you're just getting started, see some [usage examples](#).

Credentials

+ Add key

API Keys



Cloud API Keys

For use with CLI, REST Interface or Terraform



Resource API Keys

For applications



Schema registry

For accessing the (managed) schema registry.



ksqlDB

For accessing ksqlDB through REST interface

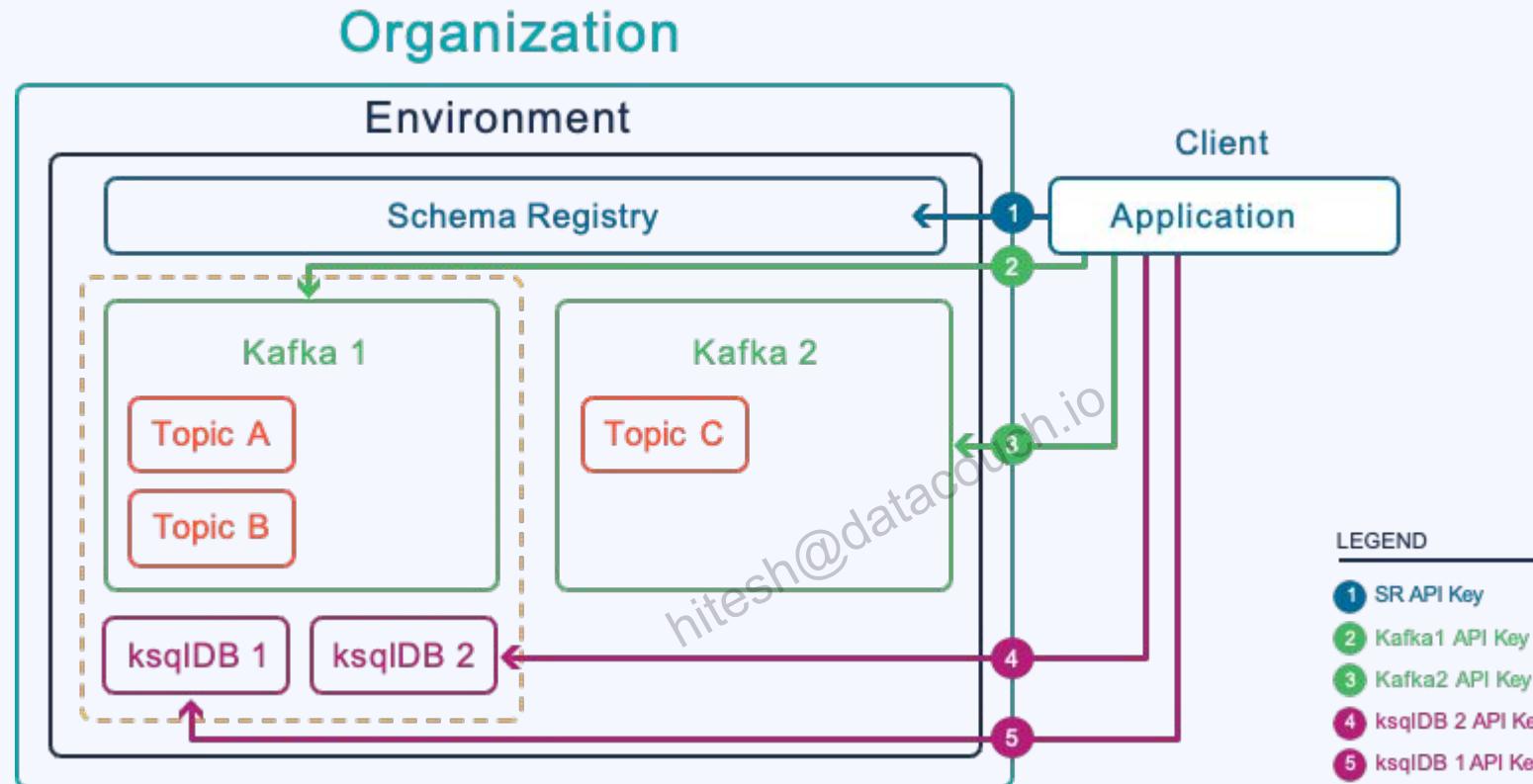


Audit log and Metrics

For accessing the audit logs and metrics



API Keys Details



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



Basic Cluster Lab Goals and Hints



Understand the UI of Confluent Cloud and use it to create an environment and a “basic” cluster.



- Use [Lab 1: Basic Cluster](#)
 - Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the Confluent Cloud UI to create
 - An environment
 - An **Essentials** Schema Registry
 - A Basic Cluster
- Familiarize yourself with the features and configuration of your new cluster

Target time: 1 hour



Lab 2: Clients

nitesh@datacouch.io



Configuring client applications



Confluent CLI

- Confluent CLI basic setup
 - `confluent login`
 - To avoid future cluster selection, you can do:
`confluent environment use <environment-id>`
`confluent kafka cluster use <cluster-id>`
- Your confluent cli configuration will be saved under `~/confluent/config.json`.
- Confluent CLI can be used to perform any admin tasks and to produce and consume from topics
 - `confluent kafka topic produce/consume`

https://dataconflux.io



New client

1 Choose your language

Get the required configuration for your programming language.



Java



Python



C#



Node.js



Spring Boot



Go



C/C++



REST API



Scala



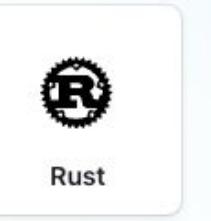
Clojure



Ruby



Ktor



Rust



Groovy

- 2 Copy the configuration snippet for your clients
- 3 Install the required libraries
- 4 Produce data
- 5 Consume data
- 6 Learn more
- 7 Track throughput and consumer lag



Create the client configuration

Java-based clients

C-based clients

Note there is no setting for the schema registry.

Copy the configuration snippet for your clients

Paste the following configuration data into a file called client.properties.

Cluster API Key

An API key is required to connect to your cluster. You can either use an existing key or create a new one here.

[Create Kafka cluster API key](#)

Schema Registry API Key

Schema Registry allows you to enforce a strict schema for your data. [Learn more](#)

[Create Schema Registry API key](#)

[Copy](#)

```
1 # Required connection configs for Kafka producer, consumer, and admin
2 bootstrap.servers=pkcs-e8mp5.eu-west-1.aws.confluent.cloud:9092
3 security.protocol=SASL_SSL
4 sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginM
5 odule required username='{{ CLUSTER_API_KEY }}' password='{{ CLUSTER_API_SECRET }}';
6 sasl.mechanism=PLAIN
7 # Required for correctness in Apache Kafka clients prior to 2.6
8 client.dns.lookup=use_all_dns_ips
9
10 # Best practice for higher availability in Apache Kafka clients
11 prior to 3.0
12 session.timeout.ms=45000
13
14 # Best practice for Kafka producer to prevent data loss
15 acks=all
```

Cluster API Key

An API key is required to connect to your cluster. You can either use an existing key or create a new one here.

[Create Kafka cluster API key](#)

[Copy](#)

```
1 # Required connection configs for Kafka producer, consumer, and admin
2 bootstrap.servers=pkcs-z9d0z.eu-west-1.aws.confluent.cloud:9092
3 security.protocol=SASL_SSL
4 sasl.mechanisms=PLAIN
5 sasl.username={{ CLUSTER_API_KEY }}
6 sasl.password={{ CLUSTER_API_SECRET }}
7
8 # Best practice for higher availability in librdkafka clients prior
9 to 1.7
10 session.timeout.ms=45000
```



Clients Lab Goals and Hints



Run different producers and consumers against your new cluster.

Use API Keys, Service Users, ACLs and RBAC to secure access to the cluster and its resources.



- Use [Lab 2: Clients](#)
 - Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the Confluent Cloud UI to create
 - Some topics
 - Client configurations
 - API Key
- Run different clients as producers and consumers

Target time: 1 hour



Lab 3: ACLs and RBAC

nitesh@datacouch.io



User Accounts, Service Accounts and API Keys

nitesh@datacouch.io



Accounts

User Accounts

- Associated with an email address
 - Users are not created, they are *invited*
- Can access the Confluent Cloud services and log into the Confluent Cloud UI
- API Keys created for a User Account (**Global Access**) have superuser powers and can access all resources, if the correct RBAC role has been assigned.
- Can be created via
 - CCloud UI
 - **confluent** CLI
 - REST API
 - terraform provider (>= 1.37.0)

Service Accounts

- Not associated with a human user
 - Meant for applications and connectors
- Can only access the Confluent Cloud service
- Can have roles assigned (Standard/Dedicated)
- API Keys created for a service account (**Granular Access**) can have ACLs assigned
 - In addition or instead of roles
- Can be created via
 - CCloud UI
 - **confluent** CLI
 - REST API
 - terraform provider



Role-based Access Control (RBAC)

Can control access to:

- Organization
- Environment
- Kafka cluster
- Schema Registry
- ksqlDB cluster
- Pipelines
- Network
- ...

Predefined Roles:

- OrganizationAdmin
- EnvironmentAdmin
- CloudClusterAdmin
- Operator
- KsqlAdmin
- NetworkAdmin
- MetricsViewer
- ResourceOwner
- DeveloperRead
- DeveloperWrite
- ...



Access Control Lists (ACLs)

- Lower Level Access Control
- Applies to Service Accounts (via REST) or API Keys (via UI)

hitesh@datacouch.io

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



ACLs and RBAC Lab Goals and Hints



Learn how to assign permissions in Confluent Cloud via ACLs and RBAC.

Use different setups, and understand how to use authorization effectively.



- Use [Lab 3: ACLs and RBAC](#)
 - Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the Confluent Cloud UI to create
 - Another API Key with specific ACLs
 - Service users with specific roles using RBAC
- Test different clients with specific permissions
 - Verify your assumptions - can you block a user/API key from having access to a resource?

Target time: 1 hour



Lab 4: Managed Connectors

https://nitesh-acouch.io



Connect cluster

- Managed connectors (subset but growing number of existing connectors)
 - New: you can also upload your own connector plugins
- Networking access only possible for
 - Publicly available sources and sinks
 - VPC Peering or AWS Transit Gateway (which enables two-way communication)

The screenshot shows the 'Connector Plugins' page in the Confluent Cloud interface. The left sidebar includes links for Cluster Overview, Dashboard, Networking, API Keys, Cluster Settings, Stream Lineage, Stream Designer, Topics, ksqlDB, Connectors (which is selected), Clients, and Schema Registry. The main area has a title 'Connector Plugins' and a sub-section: 'Confluent Cloud offers pre-built, fully managed Kafka connectors that make it easy to instantly connect your clusters to popular data sources and sinks. Connect to external data systems effortlessly with simple configuration and no ongoing operational burden.' Below this are search, filter, and sort controls: 'Search connectors', 'Filter by: Deployment', 'Type', 'Sort by: Popular', and a 'Generate data' button. A note says 'Fully managed cloud connector'. The page displays 218 connectors in a grid:

Icon	Name	Type	Status
Amazon S3 Sink icon	Amazon S3 Sink	Sink	Popular
Snowflake Sink icon	Snowflake Sink	Sink	Popular
Datagen Source icon	Datagen Source	Source	Tutorial
Elasticsearch Service Sink icon	Elasticsearch Service Sink	Sink	Popular
MongoDB Atlas Source icon	MongoDB Atlas Source	Source	Popular
MongoDB Atlas Sink icon	MongoDB Atlas Sink	Sink	Popular
Microsoft SQL Server CDC Source icon	Microsoft SQL Server CDC Source	Source	Popular
Postgres CDC Source icon	Postgres CDC Source	Source	Popular



Managed Connectors Lab Goals and Hints



Understand how to set up and configure a managed connector, here a Datagen Connector.

Ensure the connector has the correct permissions to access just the topics it needs.



- Use [Lab 4: Managed Connectors](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the Confluent Cloud UI to create
 - A (managed) Datagen connector with global access (JSON format)
 - A Datagen connector with access to specific topics only (AVRO format)
- Use the CLI to create 3rd connector
- Verify your topics: are your connectors producing data?

Target time: 1 hour



Lab 5: OIDC

nitesh@datacouch.io



OpenID Connect (OIDC) Definition

What is OpenID Connect (OIDC)?

OpenID Connect (OIDC) is an identity layer built on top of the OAuth 2.0 framework. It allows **third-party applications** to verify the identity of the end-user and to obtain basic user profile information. OIDC uses JSON web tokens (JWTs), which you can obtain using flows conforming to the OAuth 2.0 specifications.

OpenID vs. OAuth2

While OAuth 2.0 is about resource access and sharing, **OIDC is about user authentication**. Its purpose is to give you one login for multiple sites. Each time you need to log in to a website using OIDC, you are redirected to your OpenID site where you log in, and then taken back to the website. For example, if you chose to sign in to Auth0 using your Google account then you used OIDC. Once you successfully authenticate with Google and authorize Auth0 to access your information, Google sends information back to Auth0 about the user and the authentication performed. This information is returned in a JWT. You'll receive an access token and if requested, an ID token.

OpenID and JWTs

JWTs contain **claims**, which are statements (such as name or email address) about an entity (typically, the user) and additional metadata. The [OpenID Connect specification](#) defines a set of **standard claims**. The set of standard claims include name, email, gender, birth date, and so on. However, if you want to capture information about a user and there currently isn't a standard claim that best reflects this piece of information, you can create custom claims and add them to your tokens.

<https://auth0.com/docs/authenticate/protocols/openid-connect-protocol#openid-vs-oauth2>



OIDC Identity Provider

- External authentication via
 - Azure
 - Okta
 - Any other OIDC provider

Accounts & access

Accounts Access Single sign-on Workload identities Access requests

Identity provider	Provider ID
AWS Cognito	op-bqx
Okta Bootcamp	op-Jgr

+ Add provider

New identity provider

1. Provider type 2. Identity provider

Select an OIDC identity provider type

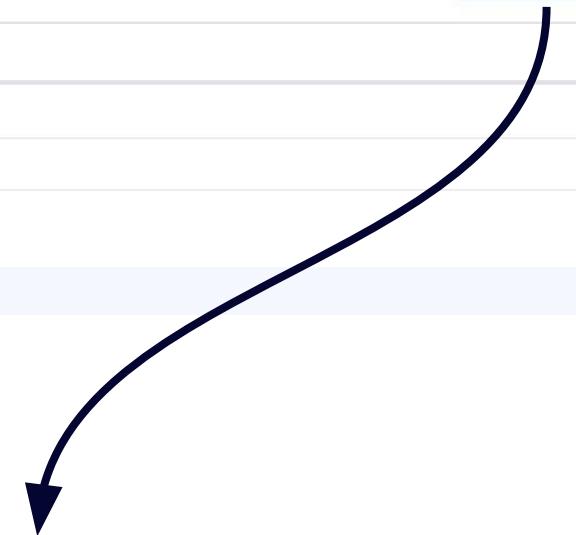
hitesh@datouch.io

Azure AD

okta

Okta

Other OIDC Provider





OIDC Identity Pool

- Identity Pool
 - Connects providers to permissions

Okta Bootcamp

Provider type: Okta

Identity provider	
Name	Okta Bootcamp
Description	Sven's First Attempt at Okta
Issuer URI	https://dev-84536545.okta.com/oauth2/ausdg5a0xigUh97DV5d7
JWKS URI	https://dev-84536545.okta.com/oauth2/ausdg5a0xigUh97DV5d7/v1/keys

+ Add pool

Identity pools ⓘ

Name	Pool ID
Okta Identity Pool	pool-qyJ6
Okta again	pool-6Mzx

New identity pool

1. Identity pool 2. Access control method

Create your identity pool

This will allow you to map permissions to your identity provider.

Set filters (recommended)

Set up filters to specify which identities can authenticate using your identity pool. All identities will be authenticated if no filter is set.

Basic Advanced

+ Add condition



OIDC Lab Goals and Hints



Learn to set up and configure an OIDC provider, and use it to authenticate a client.



- Use [Lab 5: OIDC](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the Confluent Cloud UI to set up
 - A Workload Identity Provider
 - An identity pool with a filter
- Create a client configuration to use that pool.
- Verify your access. Can you produce and consume with these credentials?

Target time: 1 hour

hitesh@datacouch.io

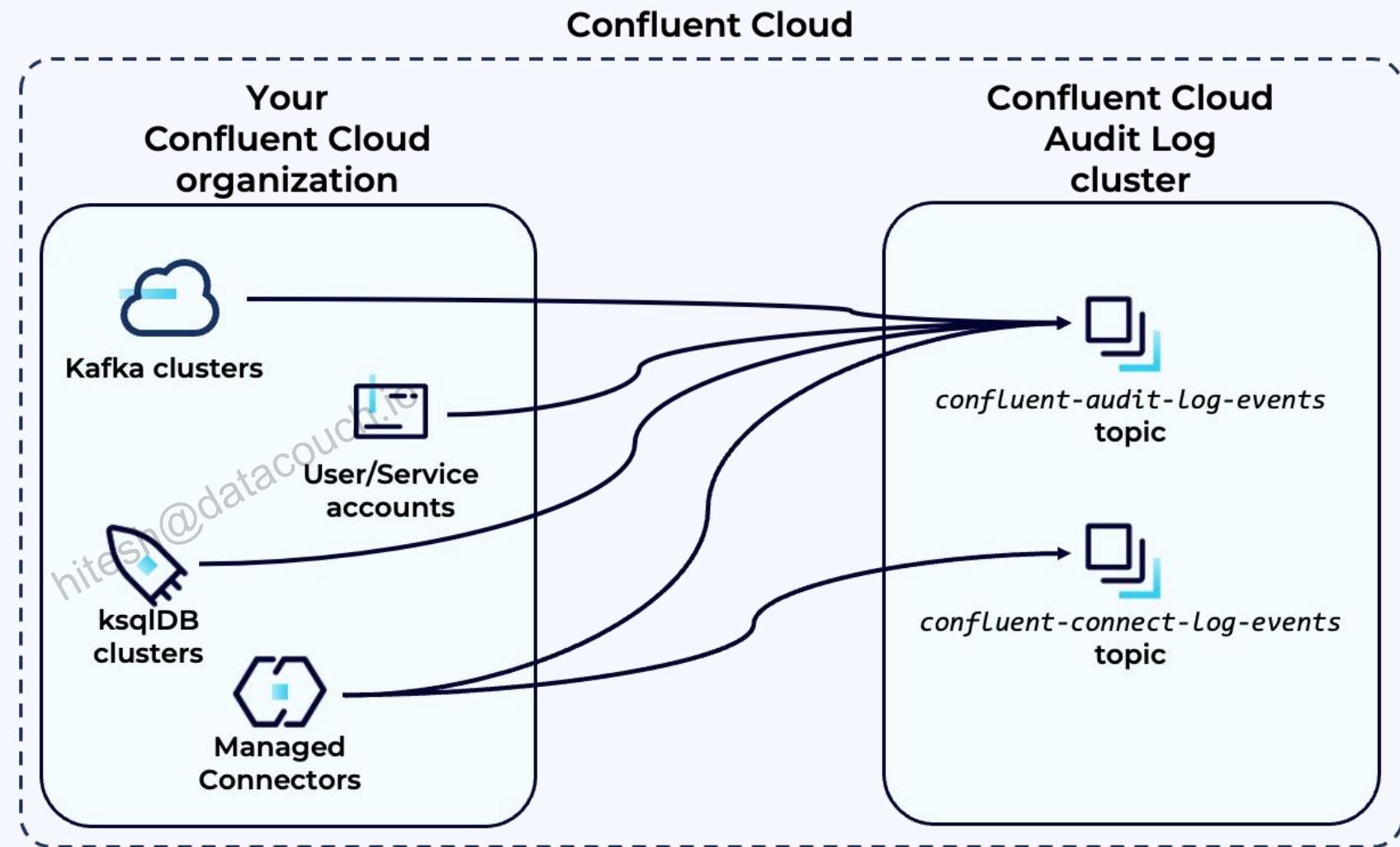


Lab 6: Audit Logs and Metrics

https://www.confluent.io

Audit logs in a separate Kafka cluster

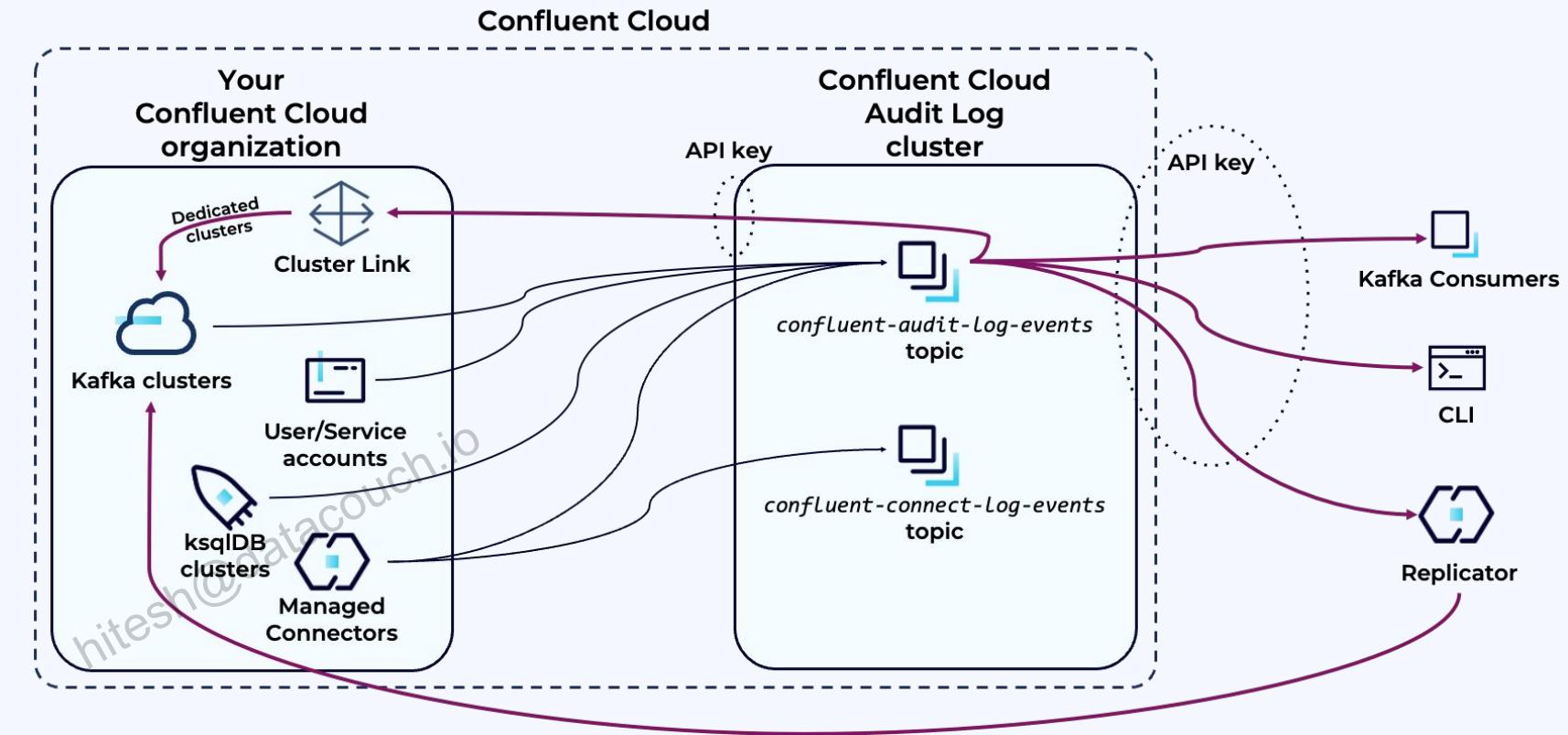
- Audit log events are stored centrally in a separate Audit Log cluster
- Managed connector logs follow the same pattern
- Retained for seven days
- Need a separate API Key to be able to access the audit log
 - Limited to 2 API Keys per organisation





Accessing Audit logs

- Can use a standard Kafka consumer
- Alternatively, use the Replicator to transfer data to another cluster
- In a later lab, we will use a Cluster Link to mirror the audit logs into your own (dedicated) cluster.





Accessing Metrics

- First-class integrations with third-party monitoring providers:
 - Datadog
 - Dynatrace
 - **Grafana Cloud**
 - Prometheus
 - New Relic OpenTelemetry
 - Directly querying the Confluent Cloud Metrics API
- Create a new Service Account:
 - Cloud API Key (Authentication)
 - MetricsViewer role (Authorization)
- **Metrics are retained for seven days**

hitesh@datacouch.io



Audit Log and Metrics Lab Goals and Hints



Learn how to set up and use audit logs and metrics.



- Use [Lab 6: Audit Log and Metrics](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the Confluent Cloud UI to get access to the Audit Log for your cluster
- Use a free Grafana Cloud account to display metrics about your cluster

Target time: 1 hour



End of Day 1

hitesh@datacouch.io



Begin of Day 2

hitesh@datacouch.io



Summary of Day 1

We have

- Created an Environment, Schema Registry and Basic Cluster
- Connected various clients
- Set up authentication and authorization (API Keys and OIDC)
- Created managed connectors
- Investigated the audit log and metrics



Deep Dive: Networking

https://datacouch.io



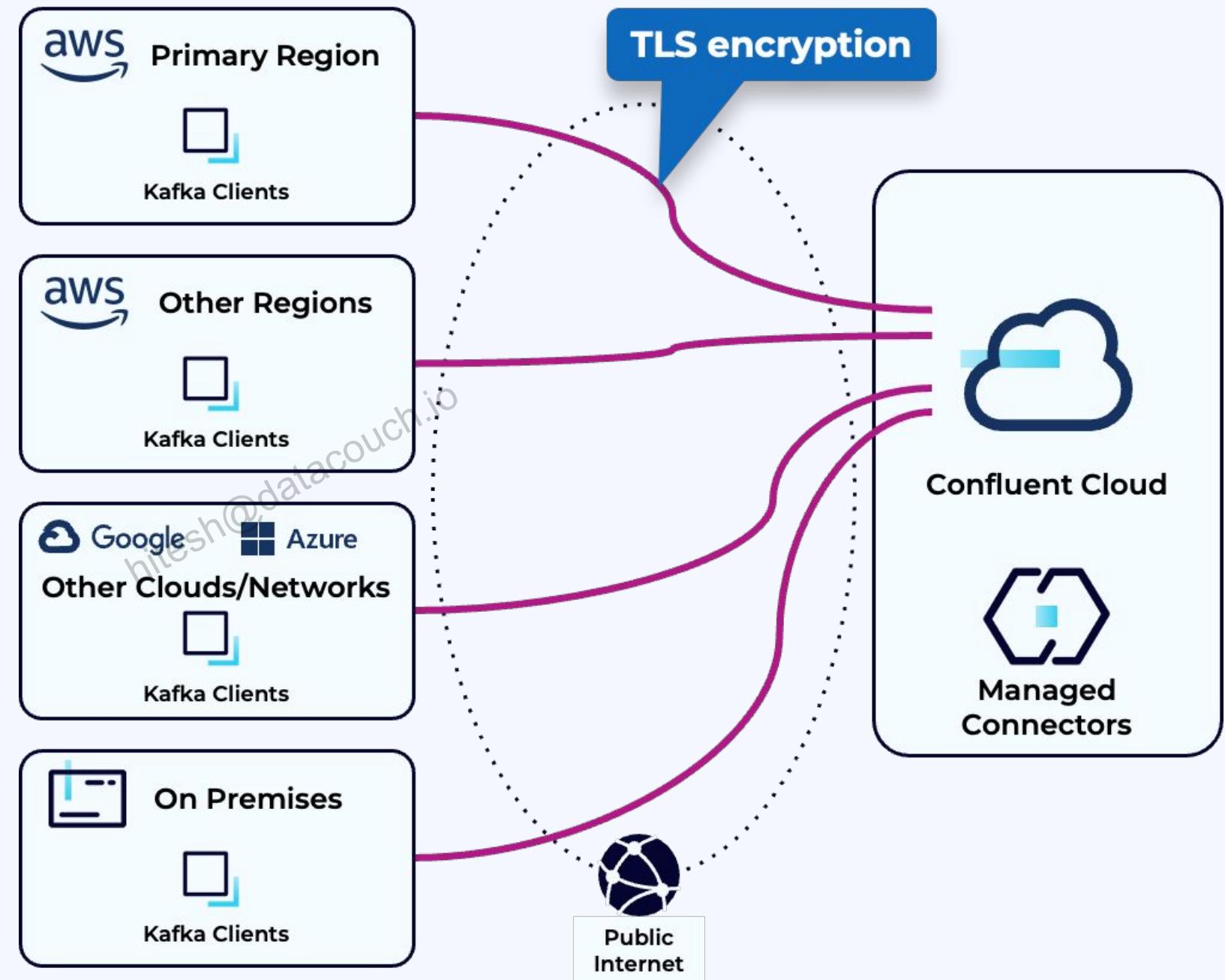
Public endpoints

Shared clusters (Basic and Standard) only have public endpoints.

A dedicated cluster can choose to use a public endpoint.

Public endpoints are visible to anyone who knows the DNS name, and are only protected via the API Key and secret.

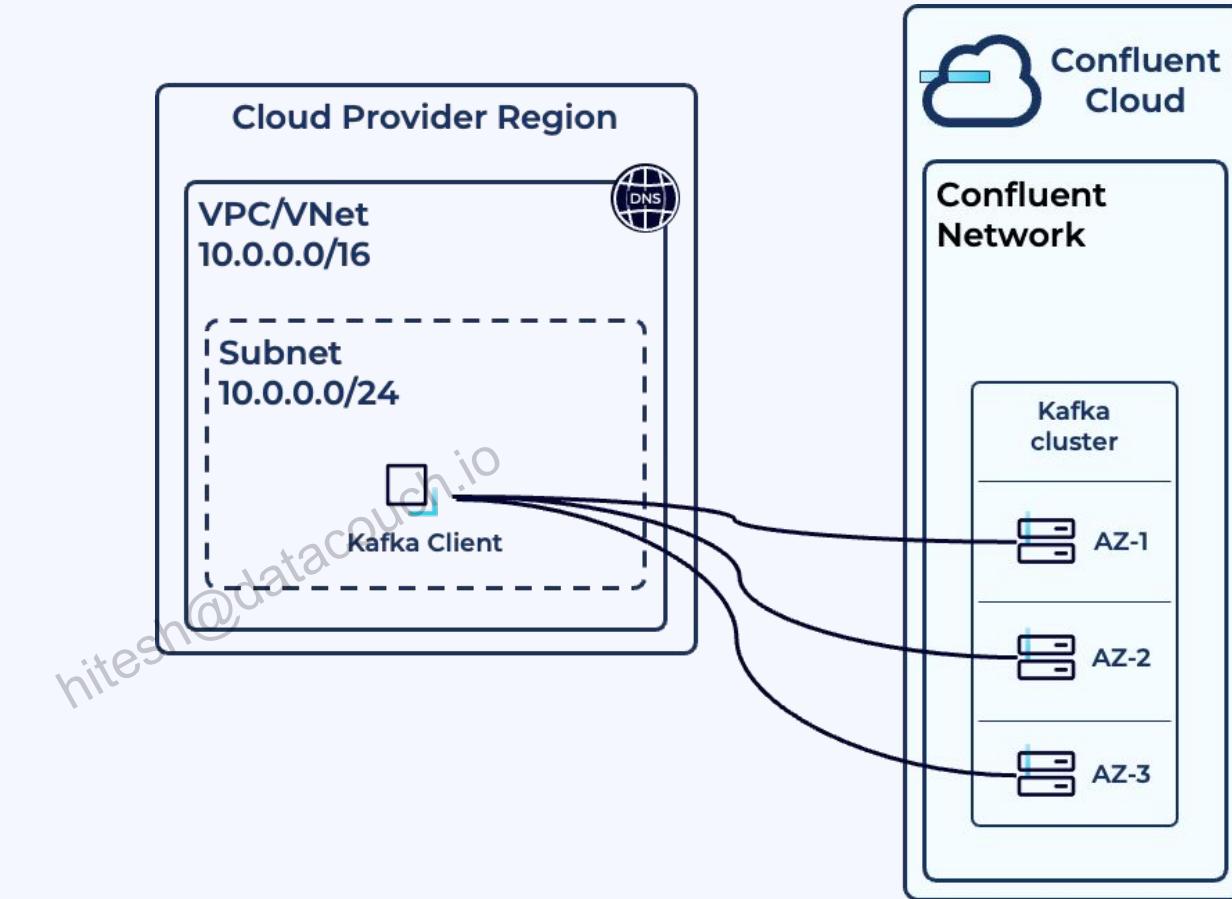
All access to a Confluent Cloud instance is always encrypted.





Private networking options

- Only accessible from private VPC
- Requires separate networking step to configure the cluster



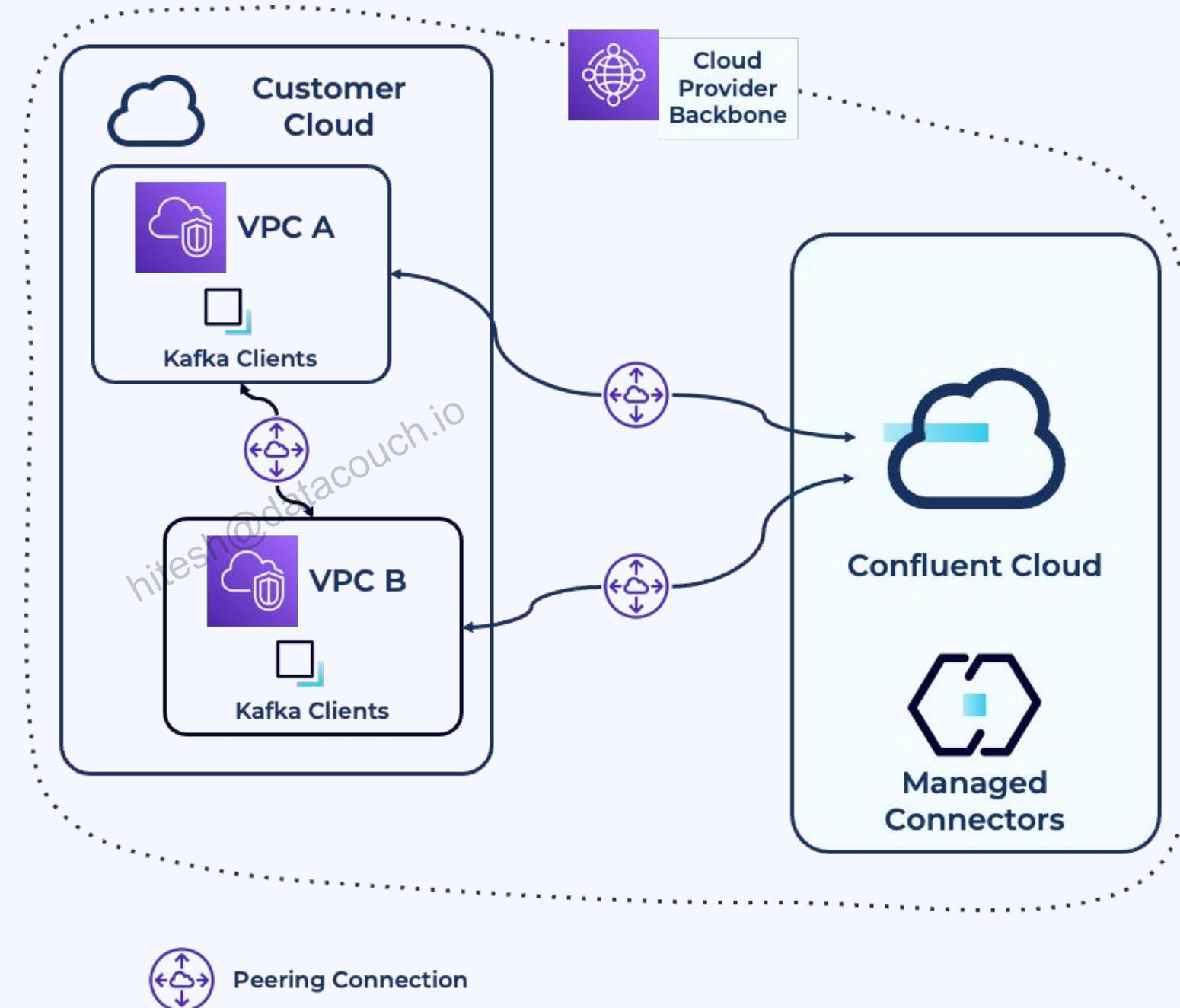


VPC Peering

VPC Peering connects two VPCs so that resources in each network can communicate with each other. It is **bi-directional**.

Ensure that the IP address ranges of CCloud and Customer VPC do not overlap.

Only the customer VPC can access CCloud.



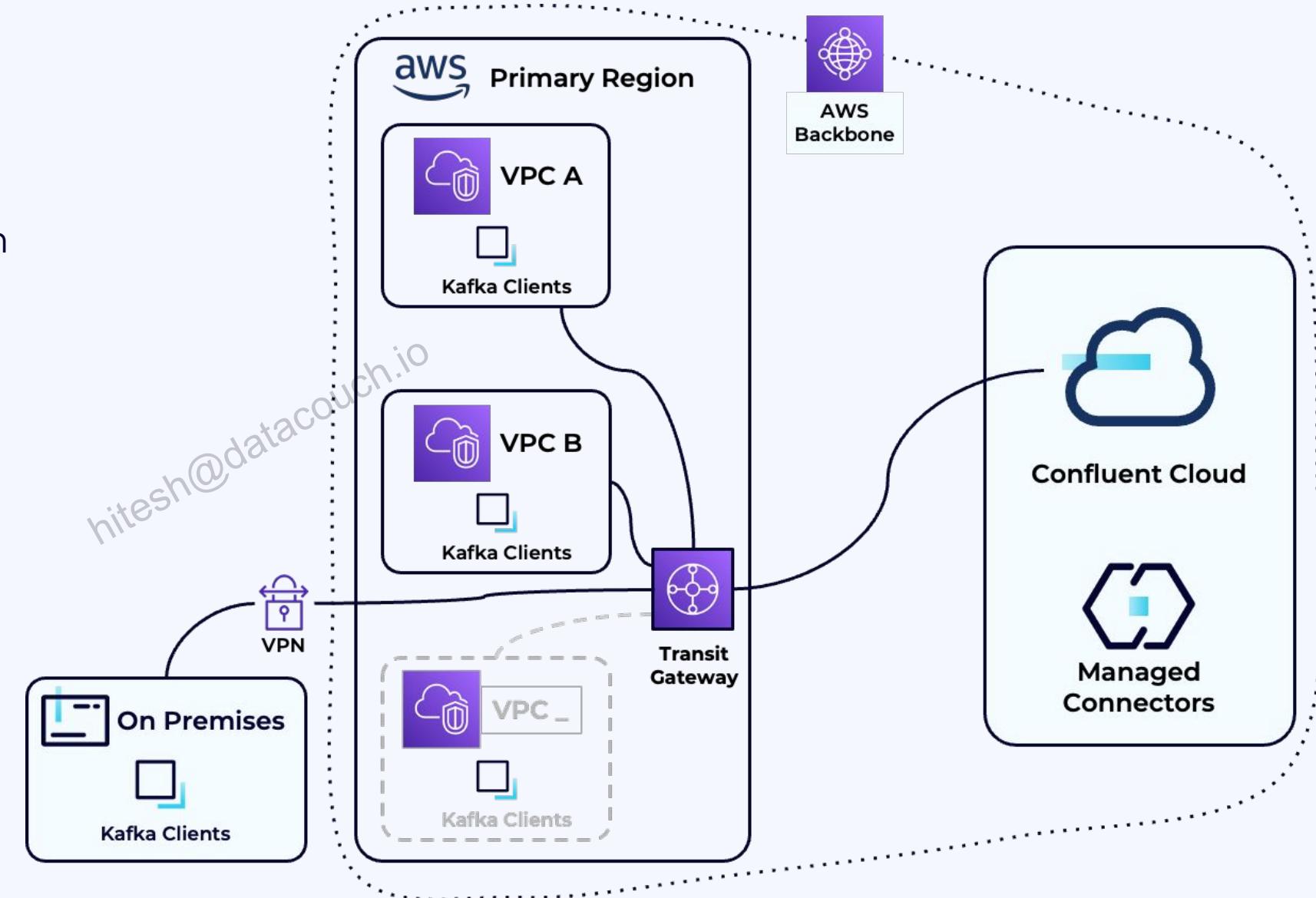


AWS Transit Gateway

AWS Transit Gateway connects VPCs and on-premises networks through a central hub.

This simplifies your network and puts an end to complex peering relationships.

It acts as a cloud router – each new connection is only made once.



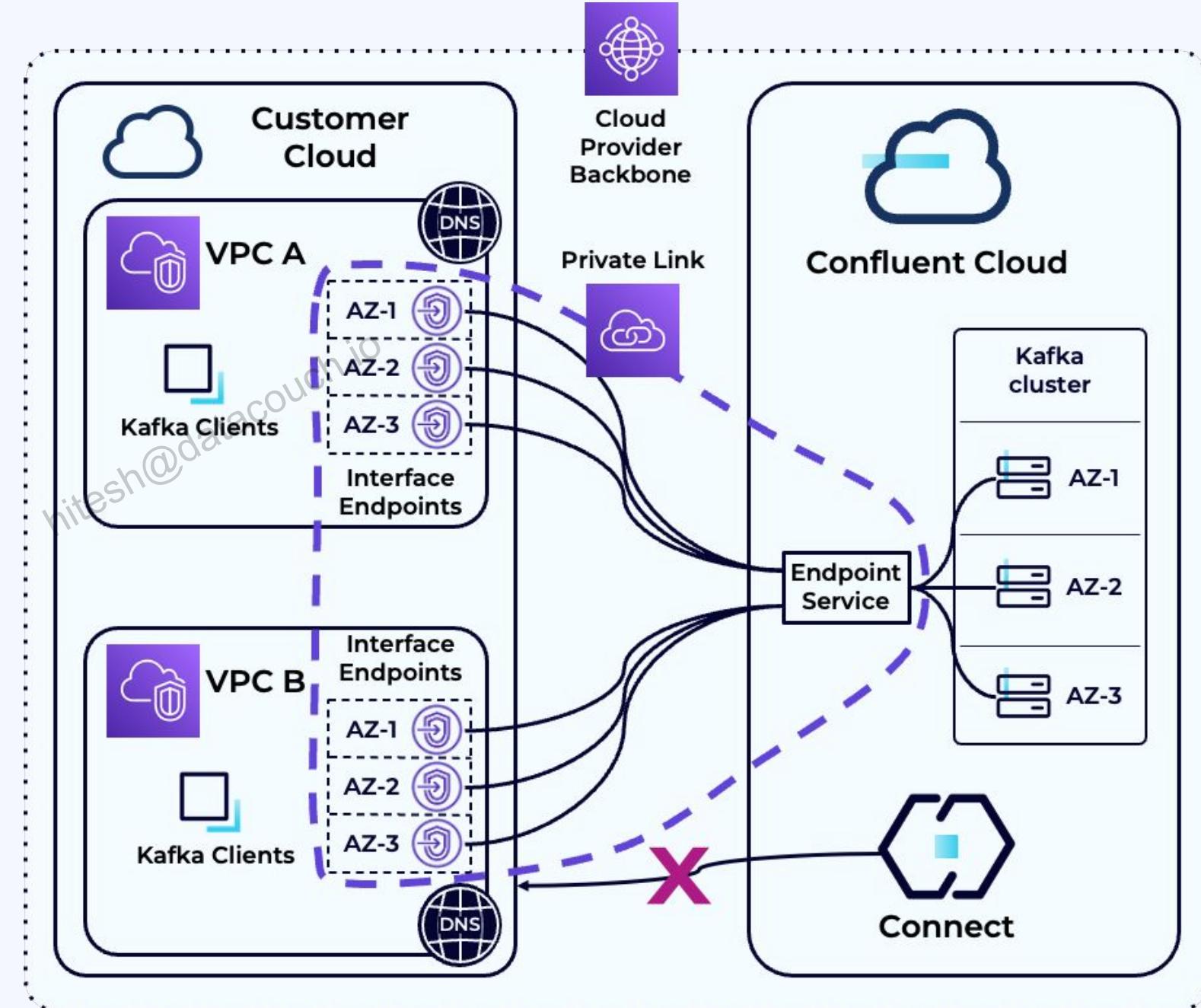


Private Link (all clouds)

PrivateLink offers **one-way** connectivity from your VPC to a Confluent Cloud cluster, without any coordination of CIDR ranges.

PrivateLink connections to Confluent Cloud can only be made from VPCs in registered customer cloud accounts.

Managed connectors **cannot** route into resources in the Customer Cloud.





Lab 7: Enterprise Cluster

https://niteshacouch.io



Setting up an Enterprise Cluster

- An Enterprise cluster will always use a Private Link for networking
- Limited access via UI
 - Cannot see or modify topics or KSQL queries
 - Either:
 - Access via CLI only
 - Alternative:
 - Use a Proxy
- After creating the Enterprise Cluster, need to set up the Private Link Connection to your VPC

hitesh@datacouch.io



Proxy Setup

nitesh@datacouch.io

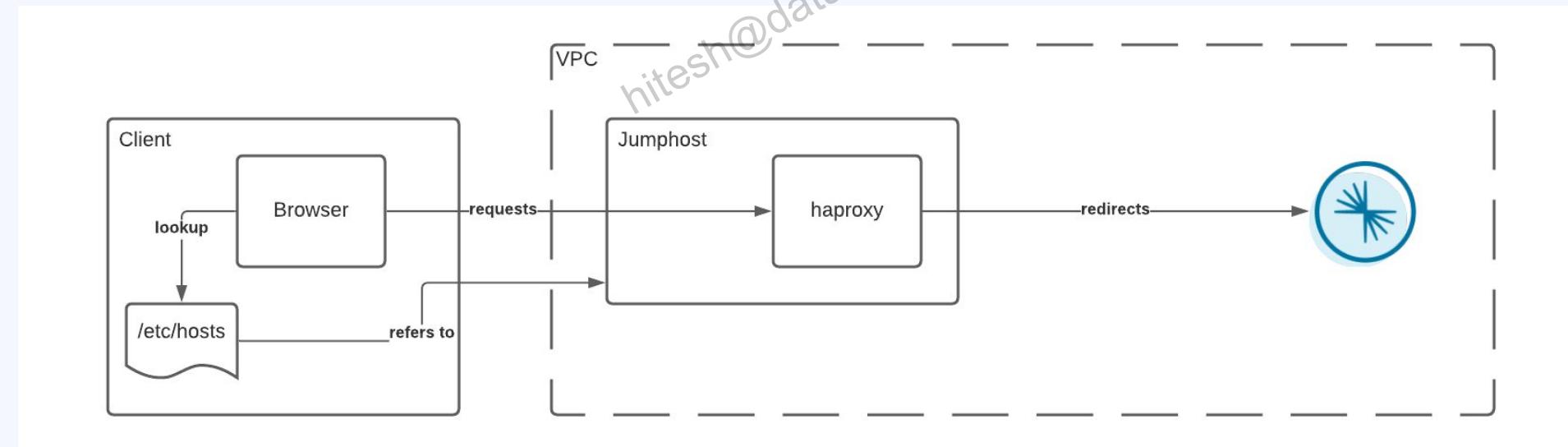


haproxy

- When private networking is set up for CCloud, you cannot access topics anymore through the GUI.
 - Same is true for ksqlDB queries

ⓘ This cluster uses private networking and is not accessible over the internet. To use this feature [follow the steps here](#). The endpoint you need to route to is `lkc-oq099o-6x32l7.eu-west-1.aws.glb.confluent.cloud`.

- We need to set up a proxy and redirect your request to CCloud in order to use the UI correctly.
- Can use **haproxy**, **nginx**, or any other Proxy technology



- Customers will do this using their networking team



Alternative: dynamic (SOCKS5) proxy

- Connect to your jumphost with the option -D <portnumber>, for example
 - ssh -D 8081 -i ~/ssh/bootcamp.pem ubuntu@63.33.215.171
- Set up a Proxy in your browser, for example FoxyProxy
 - Configure a proxy connection to localhost:8081
 - Configure an URL for which the proxy should redirect, your CCloud instance (shown in a red banner)

The screenshot shows the FoxyProxy configuration interface. At the top, there's an 'Add' button, a 'filter' input field, and a 'Get Location' button. Below this, a proxy entry for 'localhost:8081' is listed. The configuration fields include:

- Title:** localhost:8081
- Type:** SOCKS5
- Country:** (empty)
- City:** city
- Color:** A blue square with a circular arrow icon.
- Hostname:** localhost
- Port:** 8081
- Username:** username
- Password:** ***
- PAC URL:** PAC URL

Below the proxy entry, there's a table for defining URL patterns to be proxied. The columns are 'Quick Add', 'Include', 'Type', 'Title', 'Pattern', and 'Actions'. The rows are:

Quick Add	Include	Type	Title	Pattern	Actions
✓	Include	Reg Exp	EMEA Bootcamp	http[s]://controlcenter-0.sven.bootcamp-emea.confluent.io.*	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
✓	Include	Reg Exp	HTTPS	https://.*lkc-m2yg51.dom1w4k12gl.eu-west-1.aws.confluent.cloud*	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>
✓	Include	Wildcard	KSQL	*pksqlc-6j91q.dom1w4k12gl.eu-west-1.aws.confluent.cloud*	<input type="checkbox"/> <input type="checkbox"/> <input type="checkbox"/>

At the bottom center is a large orange 'Save' button.



Enterprise Cluster Lab Goals and Hints



Create an Enterprise cluster and connect it via a private link to an AWS VPC.



- Use [Lab 7: Enterprise Cluster](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Create an AWS VPC
- Use the Confluent Cloud UI to create an Enterprise Cluster
- Configure a private link to your VPC
- Test the cluster
- Create a Proxy solution to enable topics management from the UI

Target time: 1 hour



Lab 8: Dedicated Cluster

https://niteshacouch.io



CKU - Confluent Unit for Kafka

Unit	Limit
MBps ingress	50
MBps egress	150
TB storage	Infinite
Partitions	4,500
Connections	3,000
Connections/sec	500
Requests/sec	15,000

CKUs	Brokers
1	4
2	6
3	9
N (N>1)	3 * N

hitesh@datacouch.io

At least 2 CKU are required for multizone clusters.

For organizations with credit card billing, the upper limit is 4 CKUs per dedicated cluster.

Clusters up to 100 CKUs are available by request.

For organizations with integrated cloud provider billing or payment using an invoice, the upper limit is 24 CKUs per dedicated cluster. Clusters up to 100 CKUs are available by request



Dedicated Cluster Lab Goals and Hints



Create an dedicated cluster and connect it via a private link to an AWS VPC.



- Use [Lab 8: Dedicated Cluster and Private Links](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the VPC and dedicated cluster you used in the previous (Enterprise Cluster) lab
- Configure a private link to your VPC
- Test the cluster
- Use your proxy solution to enable topics management from the UI

Target time: 1 hour



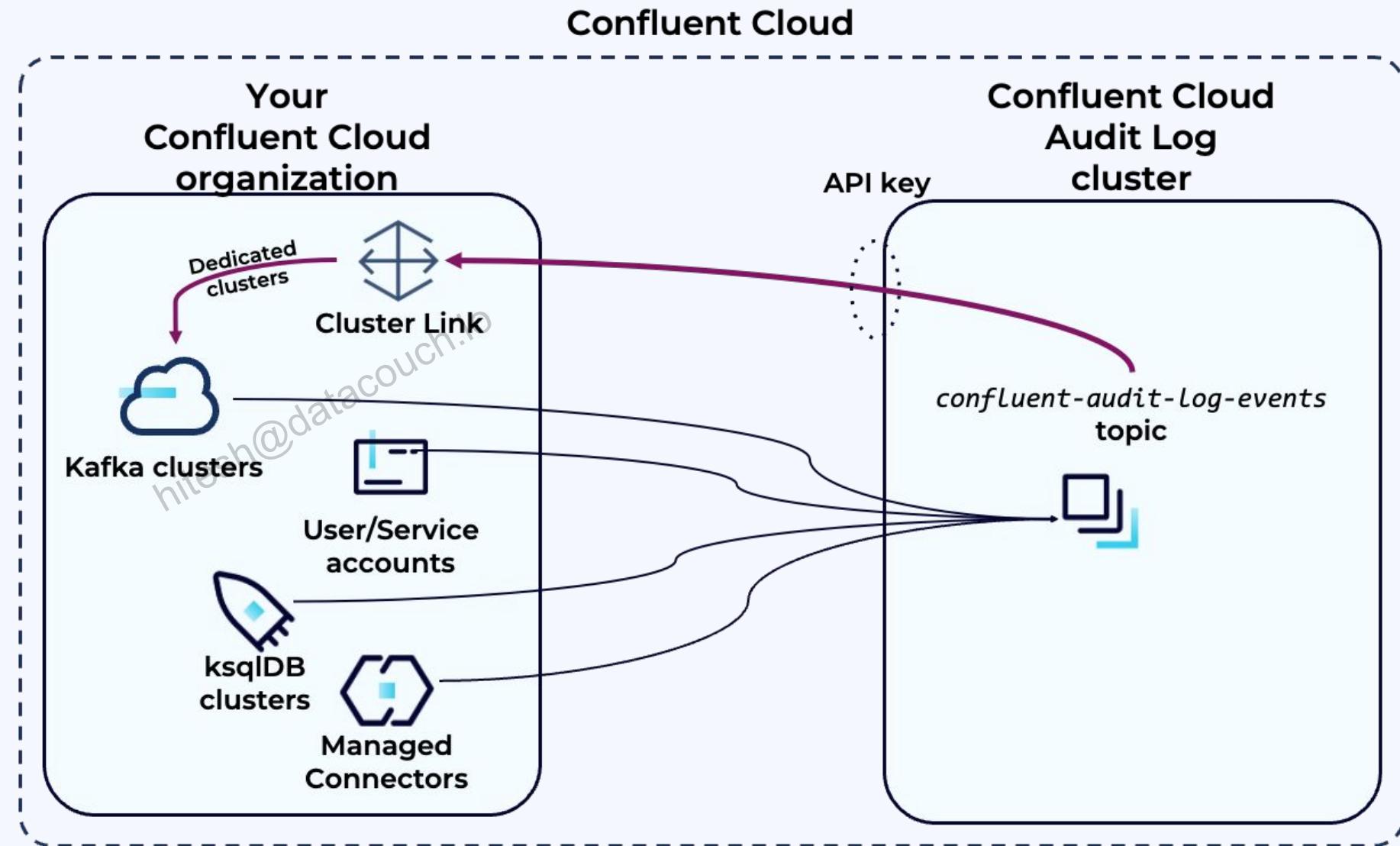
Lab 9: Audit Log via Cluster Linking

https://niteshmcouch.io



Connect audit log via Cluster Linking

- Use Cluster Linking to mirror audit log topic into your own cluster
- Allows you to analyse events via managed ksqlDB





ksqldb

- Hosted ksqldb
- Limitations?
 - UDAF & UDTF are not possible on the Cloud.
 - 20 Persistent Queries
 - You can have a maximum of 10 ksqldb clusters per environment
 - Some limitations on pull queries
- Alternative: self-hosted ksqldb
 - Full flexibility

New application

1. Access control

Global access

Allow your ksqldb application to access everything you can access. Application access will be linked to your account

*Recommended for development.

Granular access

Limit the access for your application. Manage application access through a service account.

*Recommended for production.

Choose service account

Select or create a new service account that your ksqldb application will use to communicate with your Kafka cluster.

Existing account Create a new one

Service account

Service Account	Cloud ID	Note
AdminUser	sa-l95rx7	provisioning may fail. View required ACLs
AnotherUser	sa-ldwwg7	
DemoService	sa-epr0vm	
EnvironmentTest	sa-l68156	
HUKServiceAccount	sa-Oxdorp	
testUser	sa-dlzn3l	
TopologyTest1	sa-ldw3my	
TopologyTest2	sa-l681mq	



Dedicated Cluster Lab Goals and Hints



Use Cluster Linking to bring the audit logs for your organization into your dedicated cluster.

Use ksqlDB to shape, filter, and analyse the data.



- Use [Lab 9: Audit Logs revisited](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the dedicated cluster from the previous lab
- Configure a cluster link to the audit log data
- Create a ksqlDB cluster and use it to analyse your audit data
 - You will have to use your proxy solution to enable ksqlDB management from the UI

Target time: 1 hour



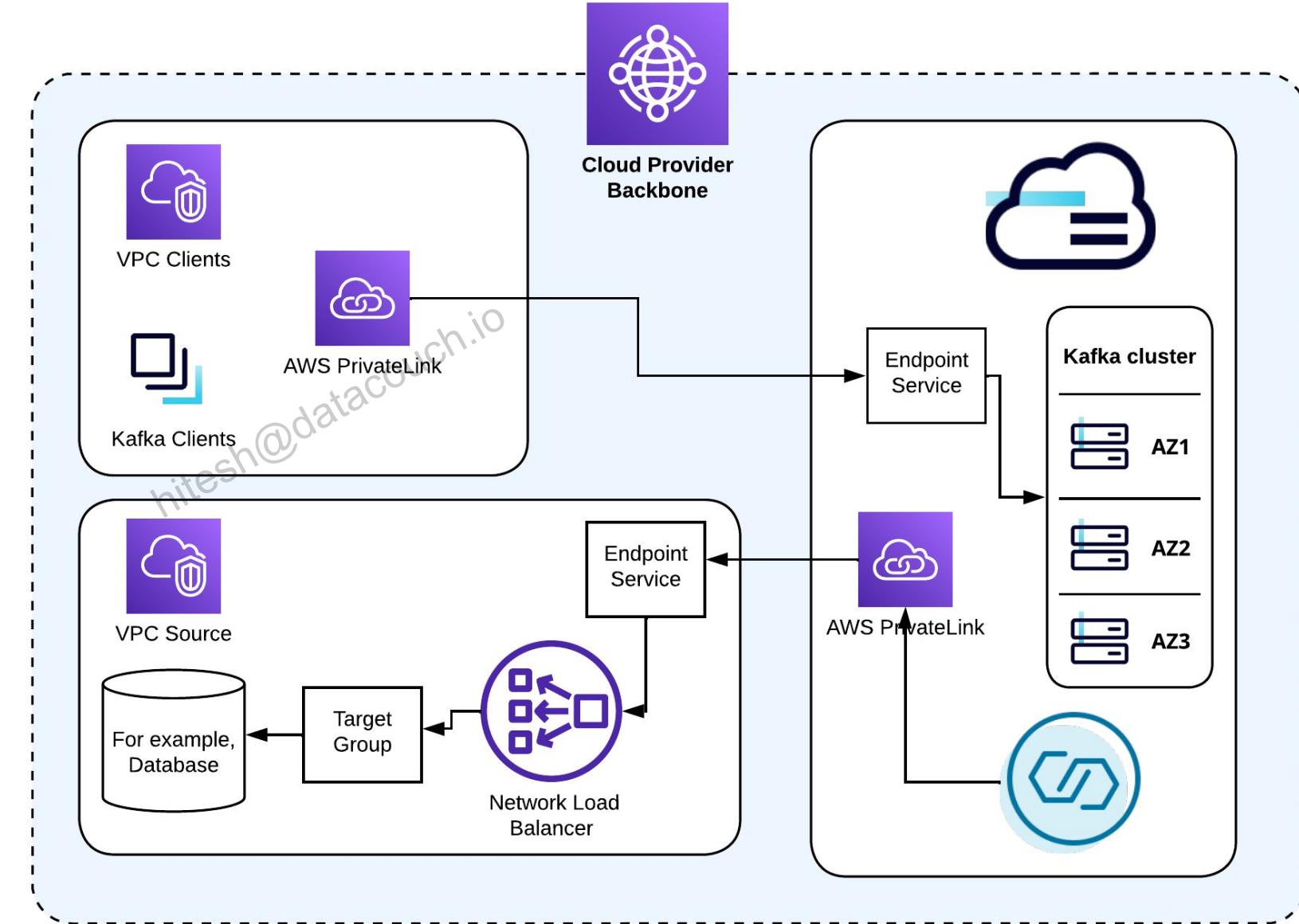
Lab 9a: Egress Access Points

https://niteshacough.io



Egress Access Points

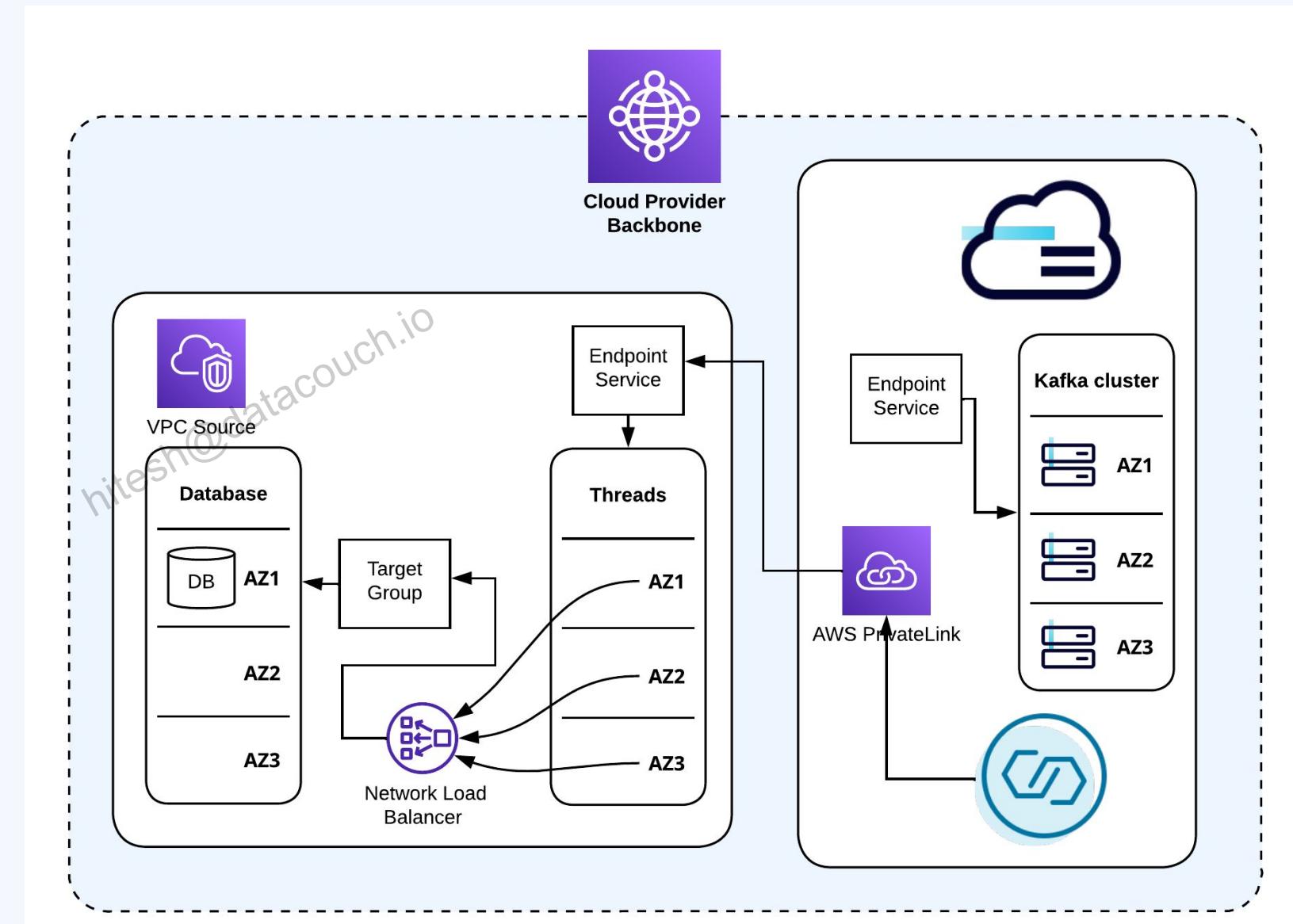
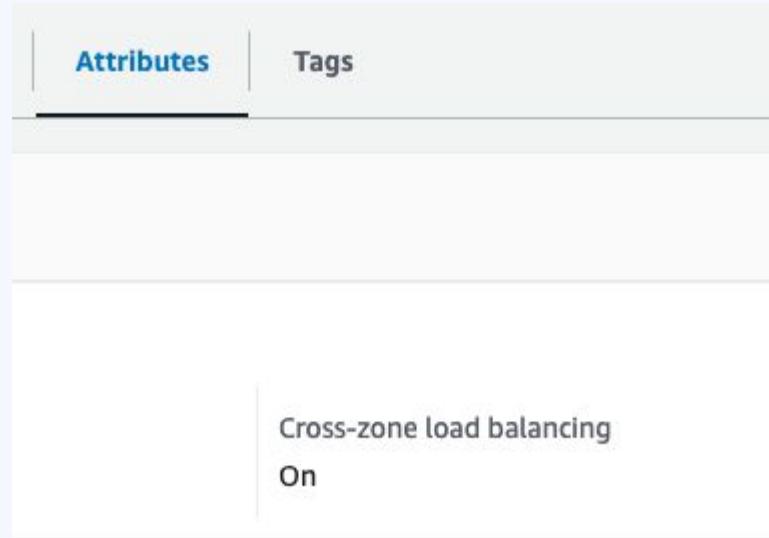
- PrivateLink Connection to instances in public and private subnets
- Enables customers to use managed connectors against secured sources and sinks





Load balancers for Multi-AZ

- Endpoint services can be configured for multiple AZ using the same load balancer
- The load balancer needs to be configured for **cross-zone load balancing** to make that possible





Connectors use endpoint DNS name as URL

PL-Postgres-Ireland-3-AZ

• Ready • Last updated: May. 13 2024 12:54 PM

ID	Service	Endpoint ID
ap-n453w8	com.amazonaws.vpce.e...vc-08c1c12c0434e8d87	vpce-006c85cd3bee313c8
VPC endpoint DNS name	DNS records	
vpce-006c85cd3bee313...1.vpce.amazonaws.com	Create record	

Authentication

Database hostname	vpce-006c85cd3bee313c8-ci6qrk93.vpce-svc-08c1c12c0434e8d87.eu-west-1.vpce.amazonaws.com
Database port	5432
Database username	admin
Database password	*****
Database name	movielens
SSL mode	prefer

[Edit](#)



Egress Access Point Lab Goals and Hints



Configure an Endpoint Service for your VPC to enable access to a private database

Use an Egress access point to connect to the endpoint service and set up the connector



- Use [Lab 9 Alternative: Private Link for Managed Connectors](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the dedicated cluster from the previous lab
- Configure an endpoint service in your VPC
- Configure an Egress access point for your Confluent Cloud dedicated cluster
- Configure a Managed CDC Source Connector to retrieve data from the database

Target time: 1 hour



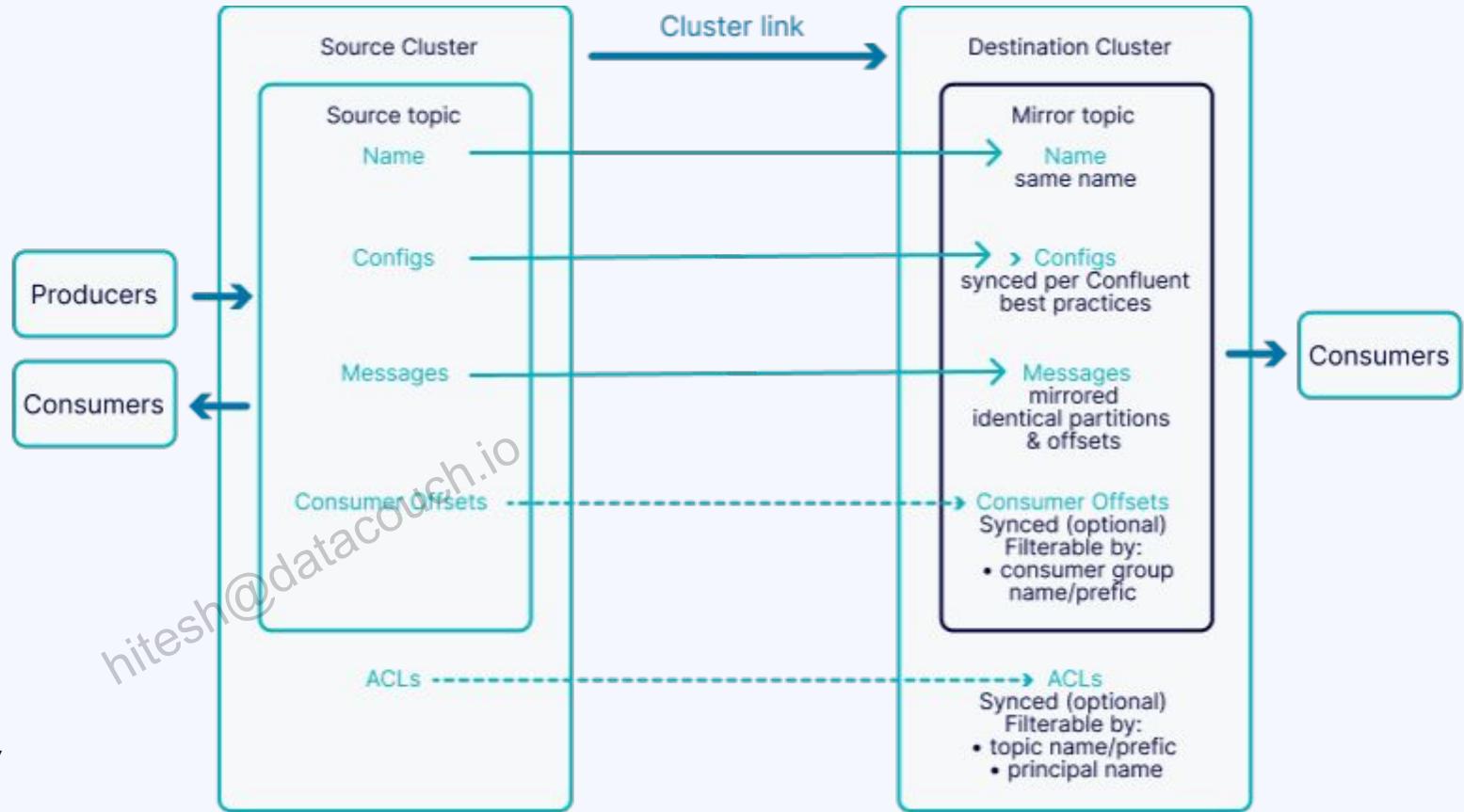
Lab 10: Cluster and Schema Linking

nitesh@nacouch.io



Cluster linking

- Mirror topics from source to destination
- Optionally sync:
 - Consumer offsets
 - Acls
- Can add prefix to mirror topic
- Mirror topics are read-only
- Cluster linking uses byte-for-byte copy
 - Preserves the offsets
- Cluster link is created and resides on the destination cluster
 - **Must be a dedicated cluster**

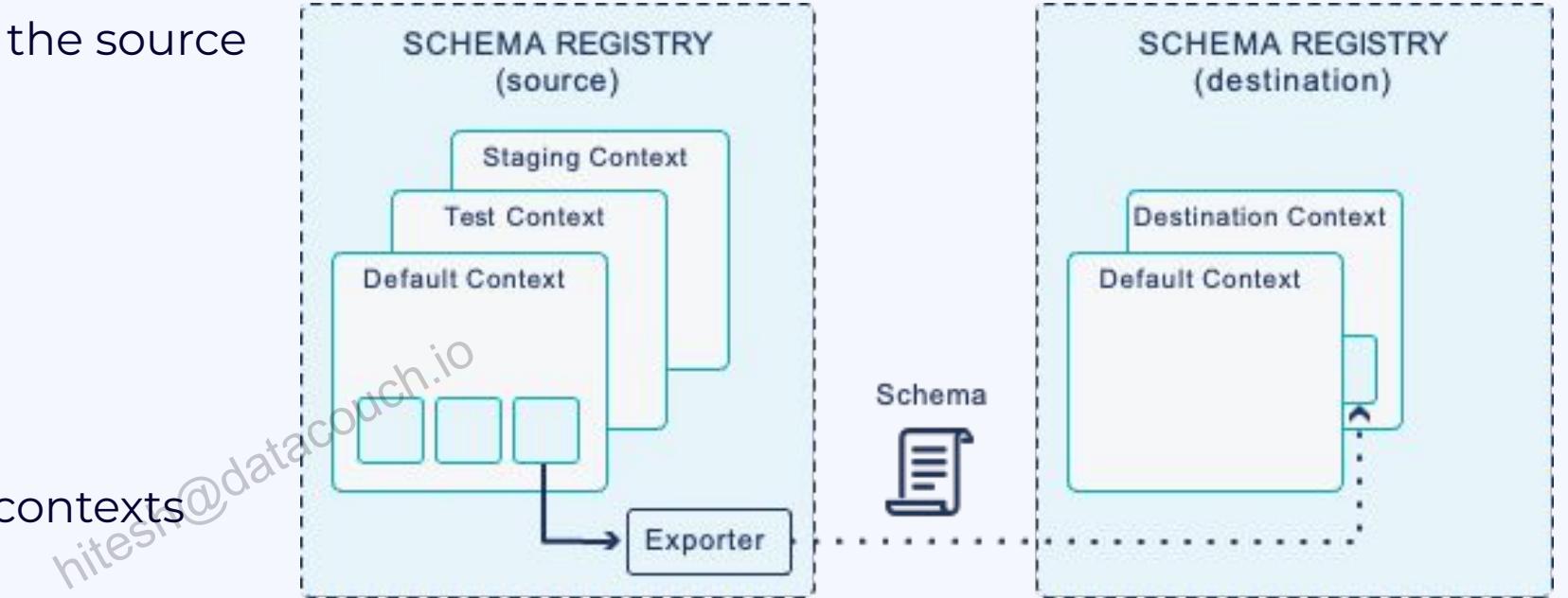




Schema Linking

- Schema linking requires an **exporter** in the source
- Schemas are grouped into contexts
 - Preserves schema ids

Useful REST API Calls to read schemas and contexts



```
curl -u <api-key:api-secret> <schema-registry>/subjects  
curl -u <api-key:api-secret> <schema-registry>/schemas  
curl -u <api-key:api-secret> <schema-registry>/contexts  
curl -u <api-key:api-secret> <schema-registry-url>/contexts/<context>/schemas  
curl -u <api-key:api-secret> <schema-registry-url>/contexts/<context>/schemas/ids/<schema-id>
```



Cluster and Schema Linking Lab Goals and Hints



Learn more about cluster and schema linking and gain practical experience using these techniques.



- Use [Lab 10: Cluster and Schema Linking](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Use the dedicated cluster from the previous two labs
- Configure a cluster link to some or all of the topics in your Basic Cluster
- Create a second environment with its own schema registry
- Use schema linking to link the schema registry of both environments together and use it

Target time: 1 hour



Lab 11: Confluent Terraform Provider

https://niteshacouch.io



Confluent Terraform Provider

nitesh@nitesh-laptop:~/Desktop\$ terraform init
nitesh@nitesh-laptop:~/Desktop\$ terraform apply



Confluent Terraform provider

- Infrastructure as code:
 - Cluster
 - Networking
 - Schema registry
 - User accounts
 - Service accounts
 - Topics
 - ACLs
 - ...
- Constantly evolving and improving
- Repeatable, complete provisioning of cluster infrastructure without the need of human interaction
- Details here:
<https://registry.terraform.io/providers/confluentinc/confluent/latest/docs>

hitesh@datacouch.io



Code example

- Choose the correct provider
 - Fixing the version provides code stability
- The API key and secret should be injected via variables
- This example creates a STANDARD cluster in AWS in a defined region

```
terraform {  
  required_providers {  
    confluent = {  
      source  = "confluentinc/confluent"  
      version = "1.35.0"  
    }  
  }  
  
provider "confluent" {  
  cloud_api_key    = var.confluent_api_key  
  cloud_api_secret = var.confluent_api_secret  
}  
  
resource "confluent_environment" "stream_bootcamp" {  
  display_name = var.confluent_environment  
}  
  
locals {  
  env_id = confluent_environment.stream_bootcamp.id  
}  
  
resource "confluent_kafka_cluster" "bootcamp-cluster" {  
  display_name = "bootcamp-cluster"  
  availability = "SINGLE_ZONE"  
  cloud        = "AWS"  
  region       = var.confluent_region  
  standard {}  
  
  environment {  
    id = local.env_id  
  }  
}
```



Client authentication to run terraform

Set environment variables with the API Key and Secret.

The code on the previous page expects these values to be set.

```
$ export TF_VAR_confluent_api_key="XXXXXXXXXXXXXXXXXX"  
$ export TF_VAR_confluent_api_secret="YYYY+YYYY+YYYY+YYYY+YYYY+YYYY+YYYY+YYYY+YYYY+YYYY+YYYY+YYYY"  
$ terraform apply
```

Do not store the API Key and Secret in your Terraform file (and *definitely* do not push it to GitHub)

hitesh@datacouch.io



Confluent Terraform Provider Lab Goals and Hints



Build up a set of Terraform scripts to create a production-ready environment.



- Use [Lab 11: Confluent Terraform Provider](#)
- Make a copy of the document
 - Use the checkboxes to mark your own progress
- Follow the lab to slowly build up a set of terraform scripts that will create
 - An environment and a schema registry
 - A Kafka cluster
 - Topics, API Keys and authorizations
 - Managed Connectors
 - ksqlDB cluster

Target time: 4 hours

Well done, you have successfully completed the labs of the Confluent Cloud bootcamp.

Please do not forget to give us feedback on ...

Good luck with the exams to complete your certification.



nitesh@datacouch.io