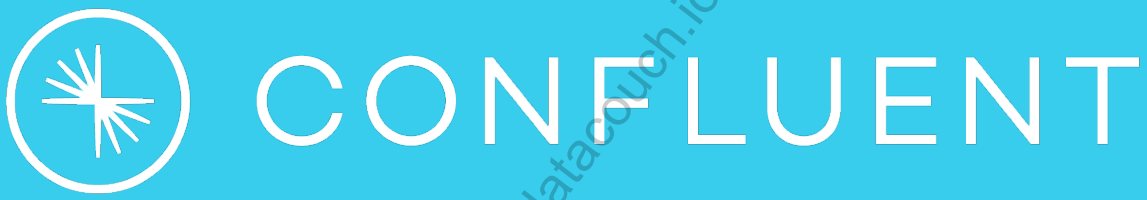
Setting Data in Motion with Confluent Cloud

Exercise Book

Version 7.3.3-v1.0.0



hitesh@datacouch.io

Table of Contents

hitesh@datacouch.io

hitesh@datacouch.io

hitesh@datacouch.io

Copyright & Trademarks

Copyright © Confluent, Inc. 2014-2023. Privacy Policy | Terms & Conditions.
Apache, Apache Kafka, Kafka and the Kafka logo are trademarks of the
Apache Software Foundation

hitesh@datacouch.io

Lab 01 Introduction to Confluent Cloud

a. Introduction

This document provides Hands-On Exercises for the course **Data in Motion with Apache Kafka® Powered by Confluent Cloud**. You will use a setup that includes a virtual machine (VM) configured with Kafka and Confluent tools to manage your data and clusters in Confluent Cloud.

Alternative Lab Environments

As an alternative you can:

- Download the VM to your laptop and run it in VirtualBox. Make sure you have the newest version of VirtualBox installed. Download the VM from this link:

  https://confluent-training-images.s3.eu-central-1.amazonaws.com/training-ubuntu-20-04-CP7-3-3-jun2023.ova

- If you have installed the Kafka and Confluent tools on your Mac or Windows machine then you can run the labs locally in your machine. But please note that your trainer might not be able to troubleshoot any potential problems if you are running the labs locally.

hiles@datarouch.io

Command Line Examples

Most exercises contain commands that must be run from the command line. These commands will look like this:

```
$ pwd
/home/training
```

Commands you should type are shown in **bold**; non-bold text is an example of the output produced as a result of the command.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Preparing the Labs

Welcome to your lab environment! You are connected as user **training**, password **training**.

If you haven't already done so, you should open the **Exercise Guide** that is located on the lab virtual machine. To do so, open the **Confluent Training Exercises** folder that is located on the lab virtual machine desktop. Then double-click the shortcut that is in the folder to open the **Exercise Guide**.

Copy and paste works best if you copy from the Exercise Guide on your lab virtual machine.

- Standard Ubuntu keyboard shortcuts will work: `Ctrl+C` → Copy, `Ctrl+V` → Paste
- In a Terminal window: `Ctrl+Shift+C` → Copy, `Ctrl+Shift+V` → Paste.

If you find these keyboard shortcuts are not working you can use the right-click context menu for copy and paste.

1. Open a terminal window
2. Clone the source code repository to the folder `confluent-ccl` in your **home** directory:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ cd ~
$ git clone --depth 1 --branch 7.3.3-v1.0.0 \
    https://github.com/confluentinc/training-cloud-src.git \
    confluent-ccl
```

If you chose to select another folder for the labs then note that many of our samples assume that the lab folder is `~/confluent-ccl`. You will have to adjust all those command to fit your specific environment.

3. Navigate to the `confluent-ccl` folder:

```
$ cd ~/confluent-ccl
```

4. List the content of the `confluent-ccl` folder:

```
$ ls -l
```

Doublecheck that the content is:

```
total 36
-rw-r--r-- 1 training users  262 May  5 07:55 README.md
-rw-r--r-- 1 training users  779 May  5 07:55 java-client.config
-rw-r--r-- 1 training users  286 May  5 07:55 kafka-cli.properties
drwxr-xr-x 4 training users 4096 May  5 07:55 lab-accessing-ccl
drwxr-xr-x 4 training users 4096 May  5 07:55 lab-cluster-linking
drwxr-xr-x 2 training users 4096 May  5 07:55 lab-schema-linking
drwxr-xr-x 6 training users 4096 May  5 07:55 lab-stream-lineage
-rwxr-xr-x 1 training users  183 May  5 07:55 update-chrome.sh
-rwxr-xr-x 1 training users  311 May  5 07:55 update-completion.sh
```

hnesh@datacouch.io

Testing the pre-installed Apache Kafka command line tools

5. The VM comes with the Confluent Platform CLI tools (binaries) version 7.3.3 which includes Kafka CLI tools 3.3.2.

   To test the installation use the `kafka-console-producer` tool to verify that is correctly installed. Using the option `--help`, you should see all the available options for this tool.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ kafka-console-producer --help
This tool helps to read data from standard input and publish it to
Kafka.
Option                                   Description
------                                   -----------
--batch-size <Integer: size>             Number of messages to send
in a single
                                           batch if they are not
being sent
                                           synchronously. (default:
200)
--bootstrap-server <String: server to    REQUIRED unless --broker
-list
  connect to>                              (deprecated) is specified.
The server
                                           (s) to connect to. The
broker list
                                           string in the form
HOST1:PORT1,HOST2:
                                           PORT2.
--broker-list <String: broker-list>      DEPRECATED, use --bootstrap
-server
                                           instead; ignored if
--bootstrap-
                                           server is specified.  The
broker
                                           list string in the form
HOST1:PORT1,
                                           HOST2:PORT2.
[Not showing all the resulting lines]
```

hitesh@datacouch.io

6. Run the following command to view all the Confluent and Kafka CLI tools included in Confluent Platform 7.3.3:

```
$ ls /usr/bin/ | grep -E 'confluent|kafka|ksql|schema-
registry|zookeeper|^connect|sr-acl|replicator|^control-
center|^cluster'
cluster-information-migration-script
clusterdb
confluent-hub
confluent-rebalancer
connect-distributed
connect-mirror-maker
connect-standalone
...
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Registering to Confluent Cloud

During this course, you will perform all exercises using your own Confluent Cloud account.

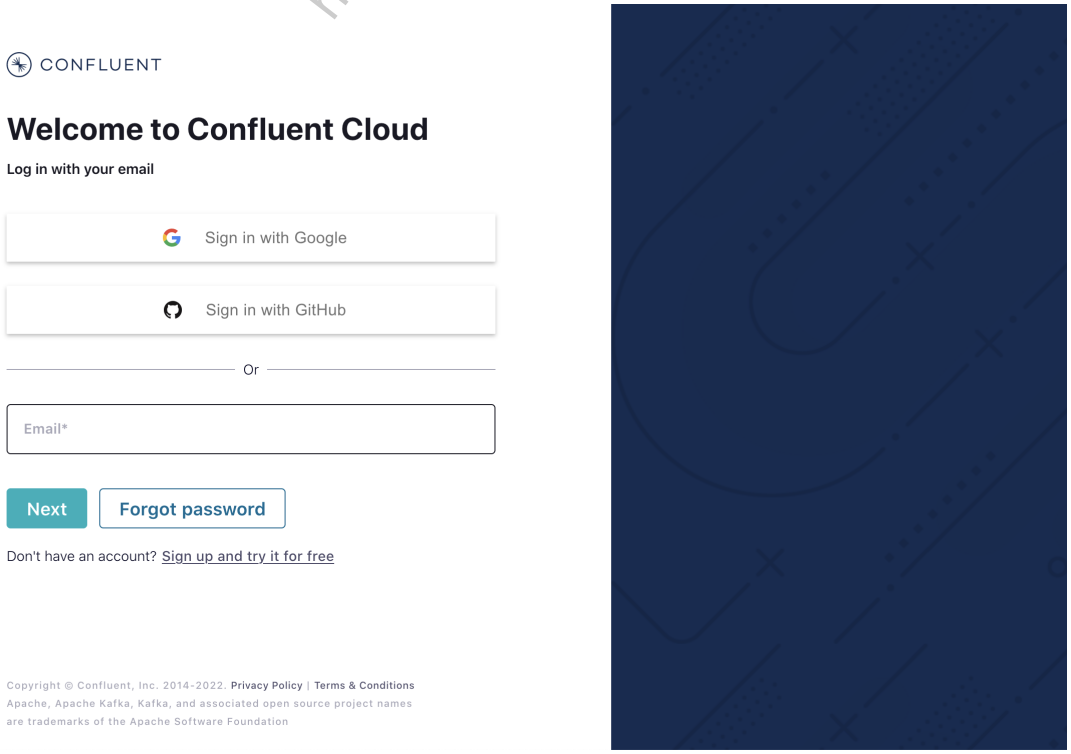
As the use of Confluent Cloud resources have a cost associated, you will be provided of **voucher code** to plenty cover the cost of performing all course exercises.

Make sure you delete all Confluent Cloud resources at the end of the exercises. We will guide through the deletion process. If you don't complete the deletion, you will keep incurring in more costs and potentially consuming all the free credit.

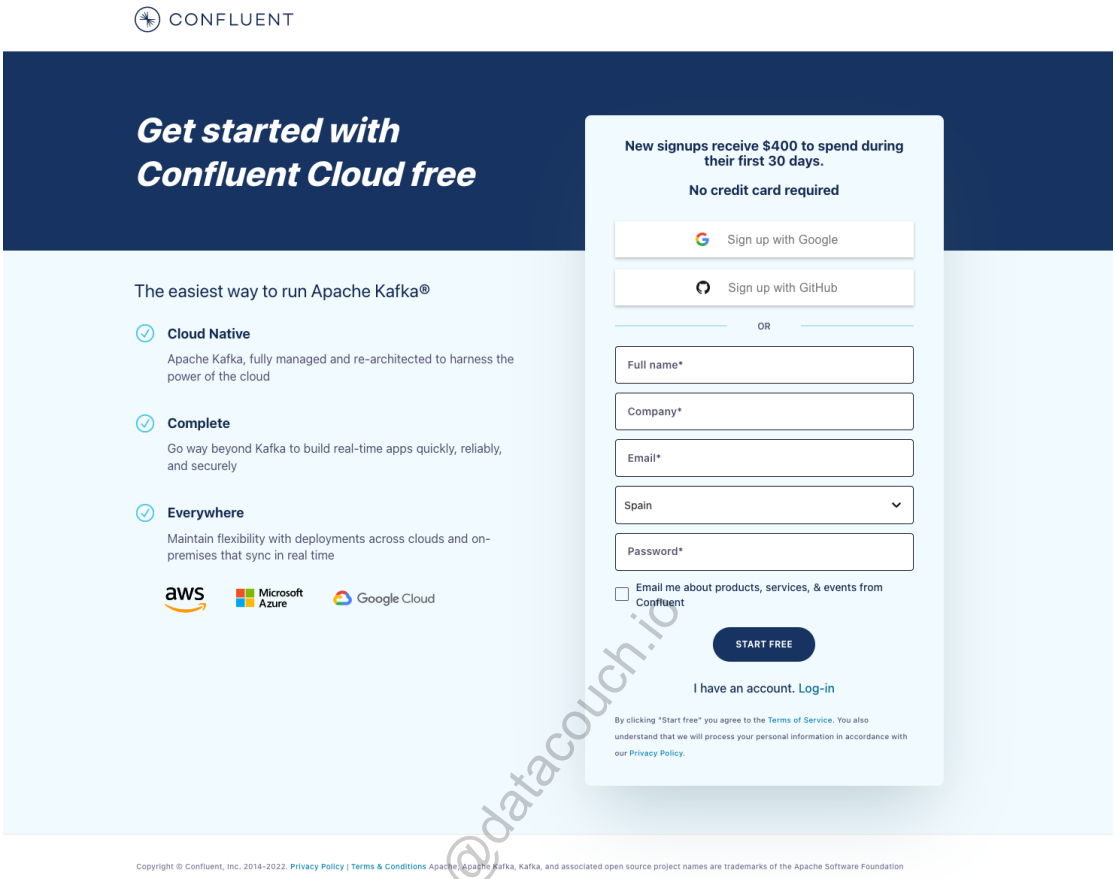
After the free credit is over, your Organization will be suspended stopping further charges. **Unless you previously added a credit card**. More information here.

hitesh@datacouch.io

7. Open a new browser tab.
8. Navigate to Confluent Cloud website at the URL https://confluent.cloud and select **Sign up and try it for free**:



9. Sign up using your Google/GitHub account or entering your full name, organization, country, email and password:



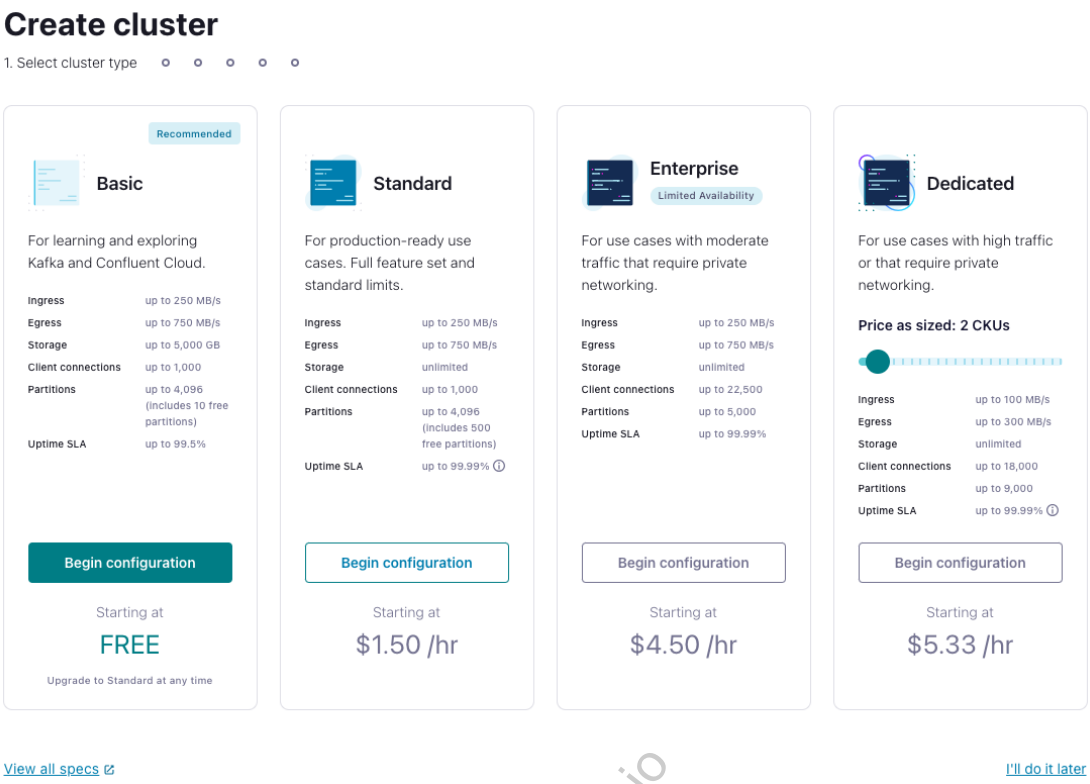
hiresh@datacouch.io

10. Click the **START FREE** button. (Make sure to keep track of your password, as you'll need it to log into Confluent Cloud later on.)

11. Watch your inbox for a confirmation email. Once you get the email, follow the link to proceed to the next step.

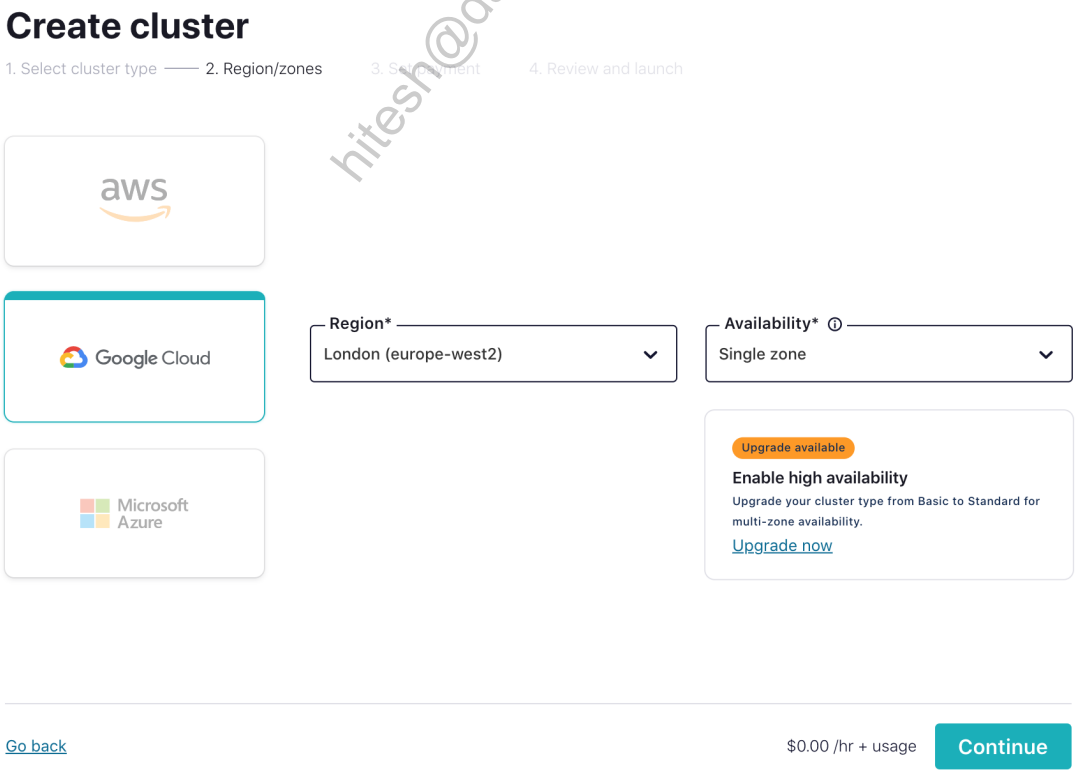
12. **Log in** into your new activated account and **create a new cluster**:
    Choose a **Basic cluster**.

    Select **Begin configuration** to start:



13. Choose your preferred **cloud provider**, **region**, and **availability**. Costs will vary with these choices, but they are clearly shown on the dropdown, so you'll know what you're getting.

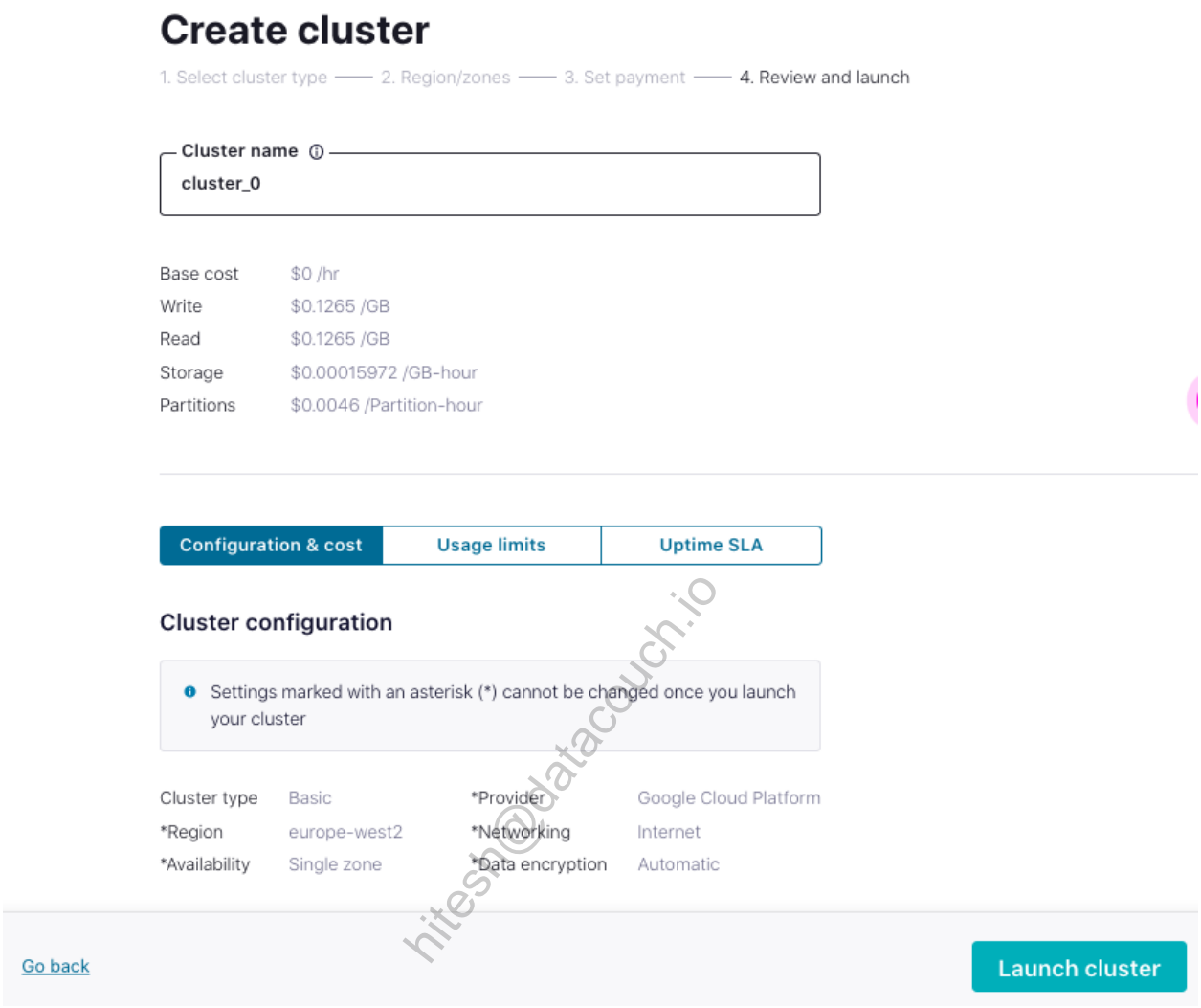
hitesh@datarouch.io



14. Click on **Skip payment**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

15. Give your cluster a **name** and review **Configuration & cost**, **Usage limits** and **Uptime SLA**. Then, select **Launch cluster**.



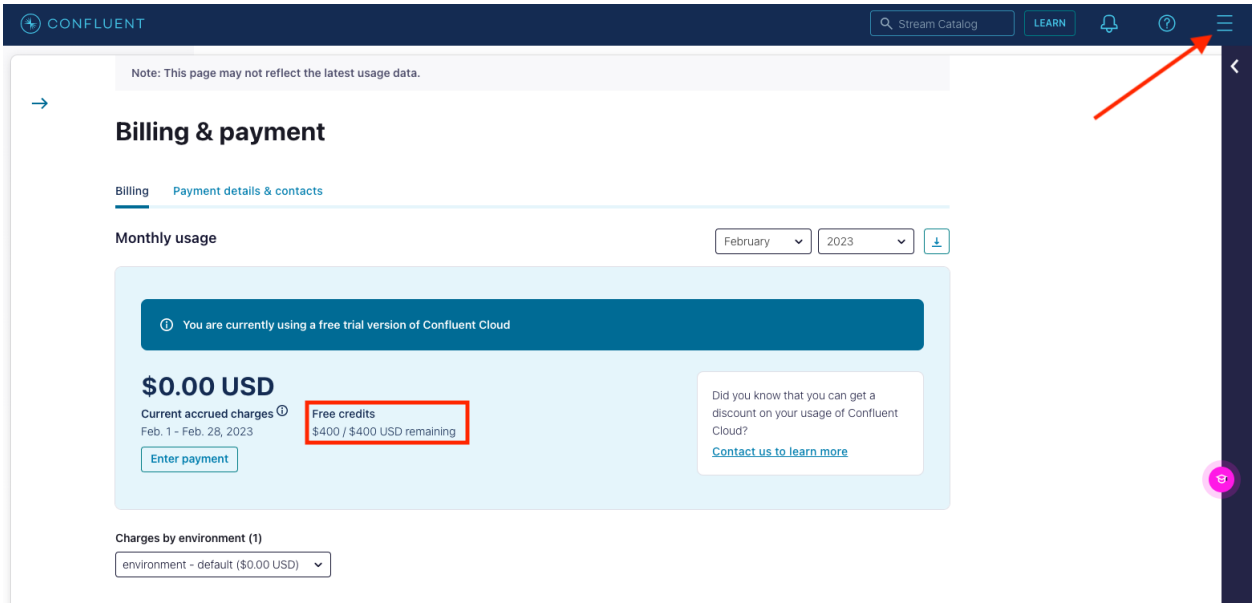
hitesh@datacouch.io

16. It might take a few seconds for your cluster to provision, since quite a bit of work is being performed for you behind the scenes. (It would take a long time to stand up a Kafka cluster yourself, especially the first time.)

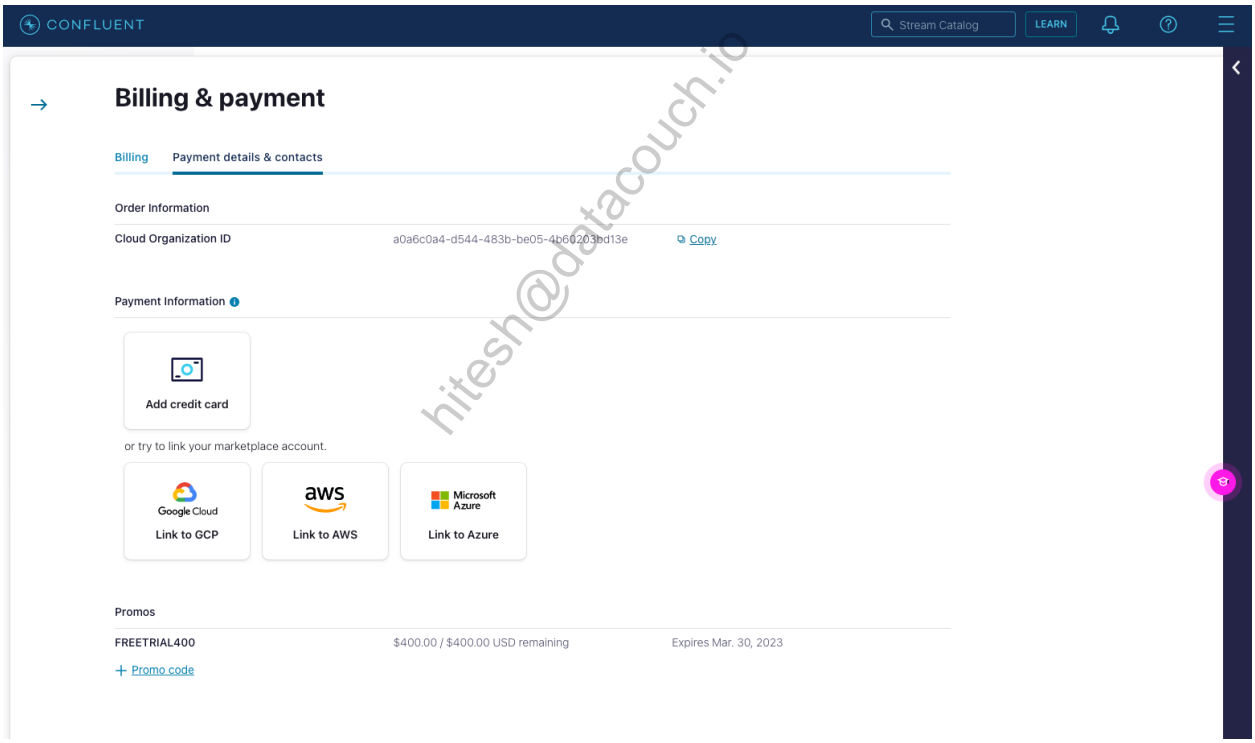
Managing your Free Credit

17. Go to **Main Menu** (three lines icon) and click on **Billing & payment**. In this window you can check your free credit balance:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



18. Now, click on the tab **Payment details & contacts**.



hitesh@datacouch.io

19. To add a new code to your Organization, click on **+ Promo code** button.

20. Enter the **voucher code** provided for this course in the pop-up window and hit **Save**.

Great! You have now created your own Confluent Cloud cluster. You will use this new cluster later in the course. Since this is a Basic cluster with no partitions or data stored and data is not being produced or consumed, you are not incurring any charges.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

In the next few labs, you will create API keys and API secrets to configure your clients, connectors, ksqlDB, etc. Try to store those API keys/secrets locally on your machine. In case the virtual machine crashes, you can recover and reuse them.

Conclusion

In this lab you have prepared and tested the Lab Environment. Finally you have created your Confluent Cloud cluster that will be used in subsequent exercises.

STOP HERE. THIS IS THE END OF THE EXERCISE.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Lab 02 Accessing to Confluent Cloud

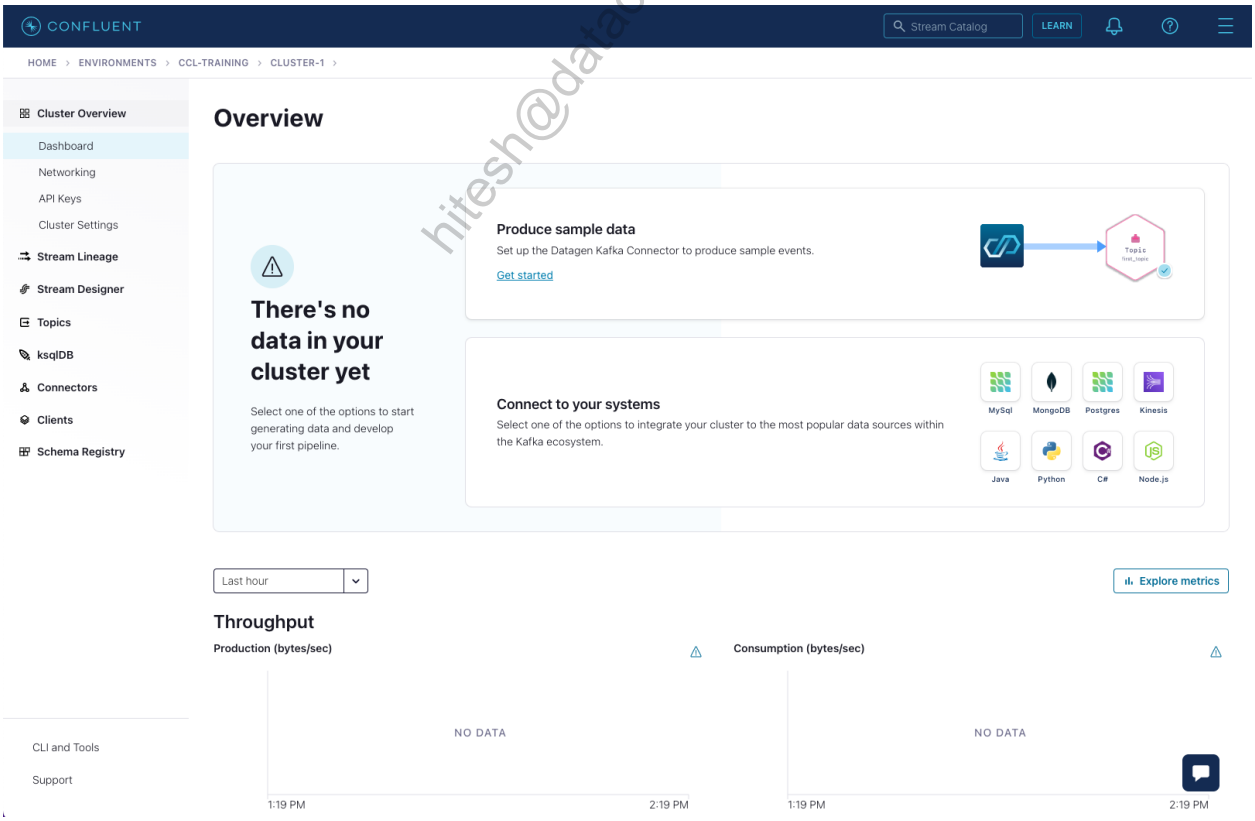
a. Starting working with Confluent Cloud

The goal of this lab is to get familiar with Confluent CLI and Confluent Cloud Console to perform common tasks on the cluster; such as, create topics, produce messages, consume messages or view cluster information. The lab explains how to set up Confluent CLI and Kafka CLI tools with Confluent Cloud, as well as how to connect external Kafka clients to your Confluent Cloud cluster using API Keys.

Create a new API key and API secret

1. Go to [Confluent Cloud website] and log in to your Confluent Cloud account.
2. Navigate to the cluster you created in Lab 1, which should be inside the Environment called **default**.

hitesh@datacouch.io

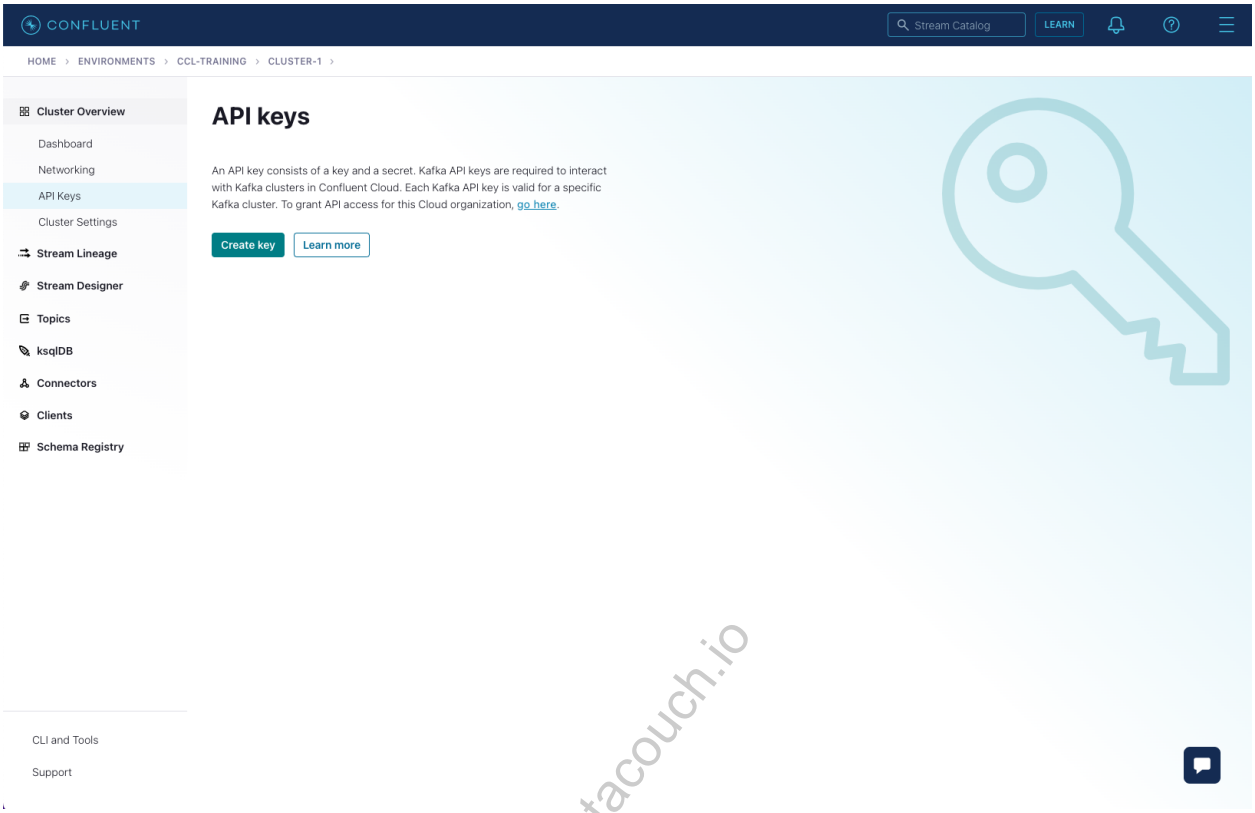


3. In the left pane, select **API Keys** which is under **Cluster Overview**.
4. Create a new **API key/secret pair** to get access to this cluster from the Confluent CLI and the Kafka CLI tools. In the next steps, you will configure these CLI tools with this

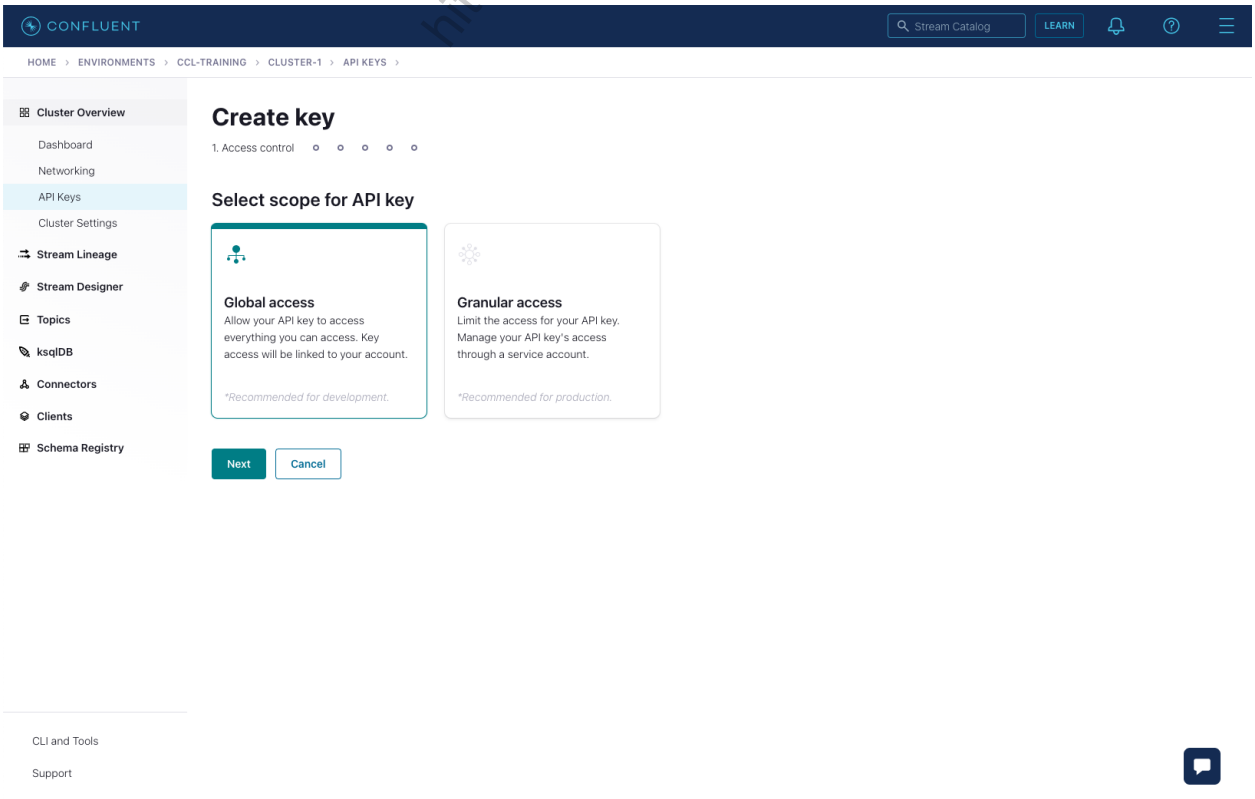
© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

new API key/secret pair.

Now, click on **Create key**.



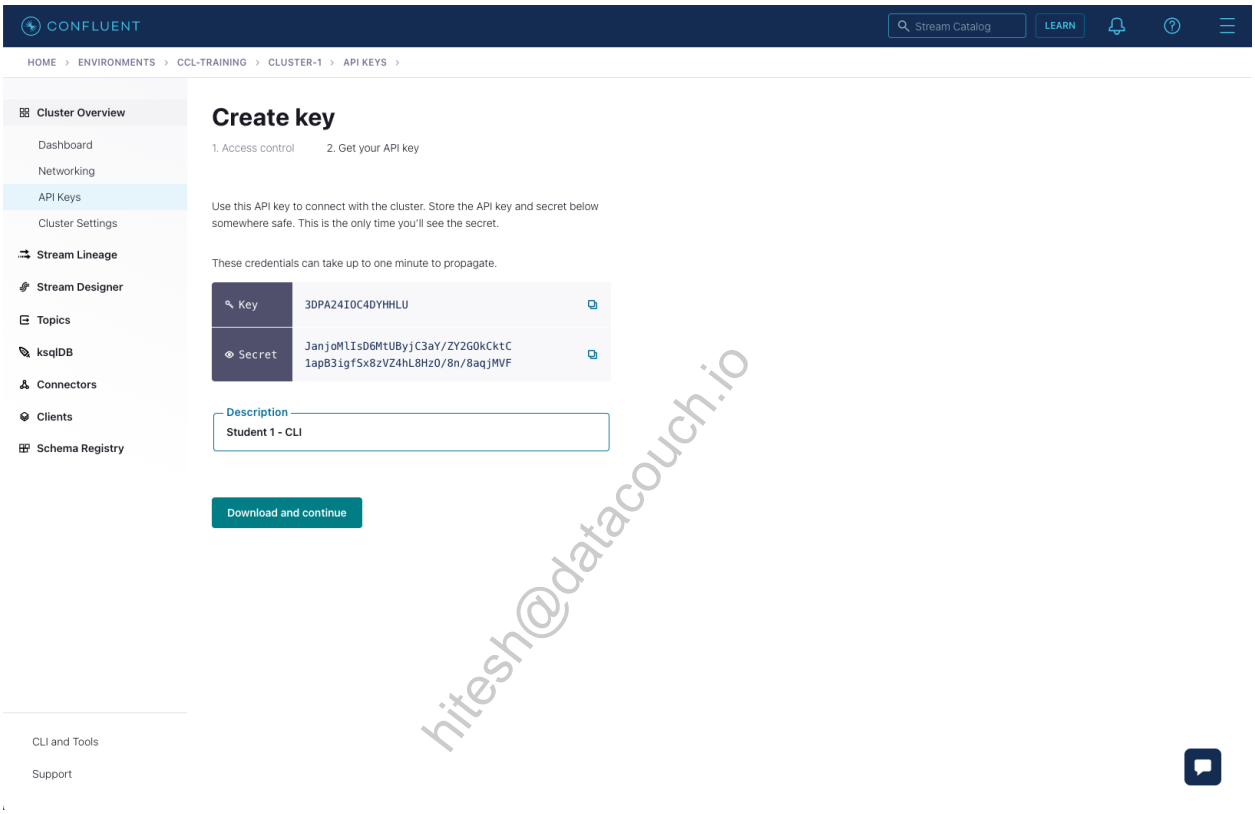
5. Choose **Global access** since this new API key will be used to manage our entire cluster using the Confluent CLI and Kafka CLI tools. Then, click **Next**.



© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

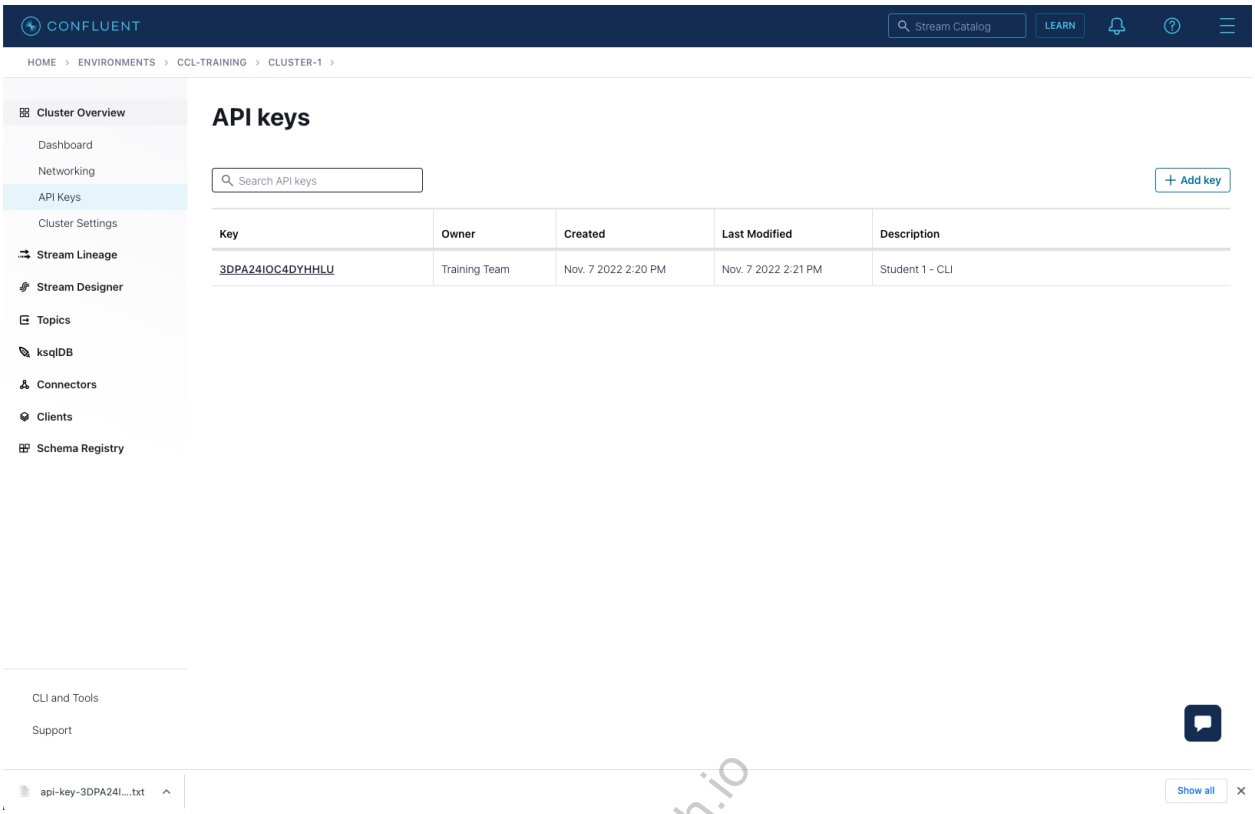
6. Give a **Description** to your new API key/secret pair to be able to recognize it in the future. Then, click on **Download and continue** to store your API key/secret locally.

Make sure you keep your API key/secret pair safely, since the API secret can't be recovered again. If you lose your API secret, you would need to create a new API key/secret pair.



7. Your new **API Key** for your Kafka cluster is now displayed under the path **Cluster Overview → API Keys**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



hitesh@datacouch.io

Using the Confluent CLI

Confluent CLI is already installed in the Virtual Machine provided.

8. Go to the Virtual Machine, open a new terminal window and run the following command to check its version:

```
$ confluent --version
confluent version v3.14.1
```

9. If you want to update the Confluent CLI version to the latest version, run the following command:

   Password: **training**

```
$ sudo confluent update
```

10. Run the following command and log into the CLI by entering your email address and password (note that this is not the API key you just generated):

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ confluent login
Enter your Confluent Cloud credentials:
Email: <enter email>
Password: <enter password>
```

If you use the command **`confluent login --save`**, your login credentials are stored locally in this file **`~/.confluent/config.json`** for non-interactive re-authentication. The password is stored encrypted.

11. Next, get a list of the Kafka clusters in your account, which should be just one:

```
$ confluent kafka cluster list
  Current |     ID     |     Name     | Type  | Provider |   Region   |
Availability | Status
----------+------------+--------------+-------+----------+------------+
--------------+---------
          | lkc-jpyqm | cluster-1 | BASIC | gcp      | us-west4 |
single-zone  | UP
```

hitesh@datacouch.io

12. Set your cluster as the default so you don't need to keep referring to it every time you run a command. Run this command using your **cluster Id** from the previous step:

```
$ confluent kafka cluster use {{ CLUSTER_ID }}
```

Sample

```
$ confluent kafka cluster use lkc-jpyqm
Set Kafka cluster "lkc-jpyqm" as the active cluster for
environment "env-7jj82".
```

13. List again your Kafka clusters. Notice that now there is an **asterisk** in the **Current** column indicating the CLI is currently using this cluster:

```
$ confluent kafka cluster list
  Current |     ID     |     Name     | Type  | Provider |   Region   |
Availability | Status
----------+------------+--------------+-------+----------+------------+
--------------+---------
  *       | lkc-jpyqm | cluster-1 | BASIC | gcp      | us-west4 |
single-zone  | UP
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

14. Store the **API key** that you got from Confluent Cloud by running this command. After pressing **Enter**, you will be prompted to add the secret:

```
$ confluent api-key store {{ API_KEY }}
```

Sample

```
$ confluent api-key store 3DPA24IOC4DYHHLU
Secret:
**********************************************************************

Stored API secret for API key "3DPA24IOC4DYHHLU".
```

15. Set Confluent CLI to use this API key when sending requests to this specific Kafka cluster:

```
$ confluent api-key use {{ API_KEY }} --resource {{ CLUSTER_ID }}
```

Sample

```
$ confluent api-key use 3DPA24IOC4DYHHLU --resource lkc-jpyqm
Using API Key "3DPA24IOC4DYHHLU".
```

hitesh@datacouch.io

Producing and Consuming with Confluent CLI

Confluent CLI has built-in command line utilities to produce messages to a Topic and read messages from a Topic. These are extremely useful to verify that Kafka is working correctly, and for testing and debugging.

16. Before we can start writing data to a topic in Confluent Cloud, we need to first create a topic, for example using the command `confluent kafka topic create`.
Run this command in the terminal:

```
$ confluent kafka topic create -h
```

This will bring up a list of parameters that this command can receive. Take a moment to look through the options.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

17. Now execute the following command to create the topic `cli-test`:

```
$ confluent kafka topic create cli-test
Created topic "cli-test".
```

Use the flag `--partitions` to set the number of topic partitions for the new topic (default: 6).

18. List all topics of your cluster. Run:

```
$ confluent kafka topic list
    Name
------------
  cli-test
```

19. Start producing some messages to `cli-test` topic using the Confluent CLI:

```
$ confluent kafka topic produce cli-test
Starting Kafka Producer. Use Ctrl-C or Ctrl-D to exit.
```

The tool is waiting for your input messages.

hitesh@datatouch.io

20. At this stage type:

```
We love Kafka
```

And hit **Enter**.

21. Now type:

```
Confluent Cloud
```

And hit **Enter**.

22. Type:

```
Data in Motion
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

And hit **Enter**.

23. Now open a new terminal window. We will use a Consumer to retrieve the data that was produced. Run the command:

```
$ confluent kafka topic consume -b cli-test
Starting Kafka Consumer. Use Ctrl-C to exit.
We love Kafka
Confluent Cloud
Data in Motion
```

After a few seconds you should see all messages that you produced earlier:

The optional flag `-b` sets the Consumer to read from the beginning of the topic.

24. Press `Ctrl+C` to exit Confluent CLI Producer.
25. Press `Ctrl+C` to exit Confluent CLI Consumer.

hitesh@datacouch.io

Setting up Kafka CLI tools with Confluent Cloud

Kafka CLI tools such as `kafka-console-producer`, `kafka-console-consumer`, `kafka-consumer-groups` and many others are important for developers who work with Apache Kafka®. In this part, you will learn how to configure them to use them with Confluent Cloud.

These tools are already pre-installed in the VM. If you need to install them in your machine, they are included in Confluent Platform along with other important tools for Replicator, ksqlDB or Schema Registry.

26. To use Kafka CLI tools with Confluent Cloud, we need a configuration properties file containing the following parameters:

```
bootstrap.servers={{ BOOTSTRAP_SERVERS }}
ssl.endpoint.identification.algorithm=https
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginMod
ule required username="{{ CLUSTER_API_KEY }}" password="{{
CLUSTER_API_SECRET }}";
```

27. Run this command to create a new properties file called **`kafka-cli.properties`** using Visual Studio Code:

```
$ code ~/confluent-ccl/kafka-cli.properties
```

28. Replace **`{{ BOOTSTRAP_SERVERS }}`**, **`{{ CLUSTER_API_KEY }}`** and **`{{ CLUSTER_API_SECRET }}`** with the corresponding values.

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

`{{ BOOTSTRAP_SERVERS }}` is the **Host:Port** of your cluster in Confluent Cloud and can be found in Confluent Cloud Console under `Cluster overview > Cluster settings > General > Endpoints > Bootstrap server` or running the following command (bootstrap-servers is highlighted):

```
$ confluent kafka cluster describe
+---------------
+-----------------------------------------------------+
| ID              | lkc-jpyqm
|
| Name            | cluster-1
|
| Type            | BASIC
|
| Ingress         |
100 |
| Egress          |
100 |
| Storage         |
5000 |
| Provider        | aws
|
| Availability    | single-zone
|
| Region          | eu-west-2
|
| Status          | UP
|
| Endpoint        | SASL_SSL://pkc-41wq6.eu-west-
2.aws.confluent.cloud:9092 |
| REST Endpoint | https://pkc-41wq6.eu-west-
2.aws.confluent.cloud:443     |
+---------------
+-----------------------------------------------------+
```

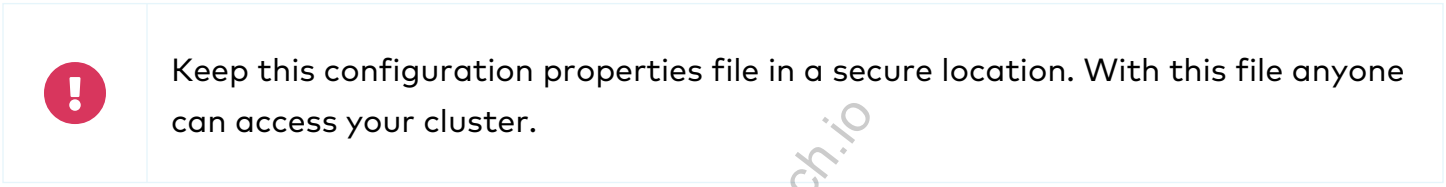
hitesh@datacouch.io

`{{ CLUSTER_API_KEY }}` and `{{ CLUSTER_API_SECRET }}` are the same ones that you used to set up Confluent CLI.
Be careful, note that the API key and API secret are double quoted `" "`, and there is a semicolon `;` at the end of the line. Those characters are required!

Sample

```
bootstrap.servers=pkc-41wq6.eu-west-2.aws.confluent.cloud:9092
ssl.endpoint.identification.algorithm=https
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLogin
Module required username="3DPA24IOC4DYHHLU"
password="PSZXWO5cThUGF6LMhlOMo4bWOhrfWYwj1xBPUaM3Z+ivLpQhEp5GrDpF
q5c8SDbH";
```

29. After replacing the placeholders, press **`Ctrl+S`** in VS Code to save the changes.
30. Close VS Code.



hitesh@datacouch.io

Using Kafka CLI tools with Confluent Cloud

31. Kafka CLI tools require to explicitly specify the **`bootstrap.servers`** property in the command.
    Let's capture your **`bootstrap.servers`** value in an environment variable to simplify the commands. Execute the following command replacing **`{{ BOOTSTRAP_SERVERS }}`** with your value:

```
$ export BOOTSTRAP_SERVERS={{ BOOTSTRAP_SERVERS }}
```

Sample

```
$ export BOOTSTRAP_SERVERS=pkc-41wq6.eu-west-
2.aws.confluent.cloud:9092
```

32. List all topics of the cluster using the tool **`kafka-topics`**. Run this command:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ kafka-topics \
    --bootstrap-server $BOOTSTRAP_SERVERS \
    --list \
    --command-config ~/confluent-ccl/kafka-cli.properties
```

Sample

```
$ kafka-topics \
    --bootstrap-server $BOOTSTRAP_SERVERS \
    --list \
    --command-config ~/confluent-ccl/kafka-cli.properties
cli-test
```

33. Let's use the `kafka-console-producer` to produce records to `cli-test` topic:

```
$ kafka-console-producer \
    --broker-list $BOOTSTRAP_SERVERS \
    --topic cli-test \
    --producer.config ~/confluent-ccl/kafka-cli.properties
```

The tool prompts you with a `>`.

34. At this prompt type:

```
> Hi Confluent Cloud
```

And hit **Enter**.

35. Press `Ctrl+C` to exit the Producer.

36. Now we will use a Consumer to retrieve the data that was produced. Run this command:

```
$ kafka-console-consumer \
    --bootstrap-server $BOOTSTRAP_SERVERS \
    --from-beginning \
    --topic cli-test \
    --consumer.config ~/confluent-ccl/kafka-cli.properties
Data in Motion
We love Kafka
Hi Confluent Cloud
Confluent Cloud
```

hitesh@datacoach.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

37. Press `Ctrl+C` to exit the Consumer.

38. Another important Kafka CLI tool is `kafka-consumer-groups`. List all your consumer groups by running this command:

```
$ kafka-consumer-groups \
    --bootstrap-server $BOOTSTRAP_SERVERS \
    --list \
    --command-config ~/confluent-ccl/kafka-cli.properties
confluent_cli_consumer_cf10a1ae-cd3c-4ca3-8973-c34076cf7ab6
console-consumer-36195
```

Connecting Kafka Applications to Confluent Cloud

To manage application access to Confluent Cloud, we should use service accounts. Each service account represents an application programmatically accessing Confluent Cloud. Permissions can be specified using ACLs and role bindings tied to a specific service account. Therefore, you can allow an application to write, read, create and/or delete a specific topic or topics.

hitesh@datacouch.io

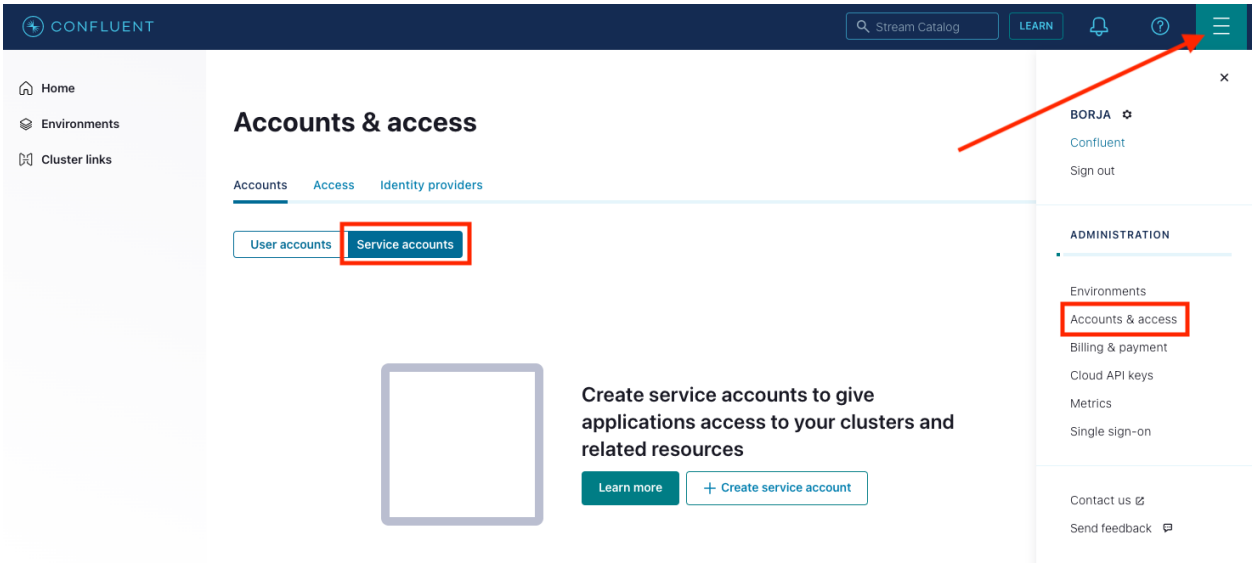
39. Let's first create a new topic called `user-data`. Go back to Confluent Cloud Console and click on **Topics** and then **Add topic**.

40. Type the topic name `user-data` and specify 6 Partitions. Then, click on **Create with defaults**.

Let's now create two Service Accounts, one for a producer and one for a consumer:

41. Go to **Main Menu** (three lines icon) and click on **Accounts & access**. In this window, select **Service accounts** and then click on **Create service account**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



42. Type the name as **JavaProducerSA** and give a description. Then, click on **Next**::

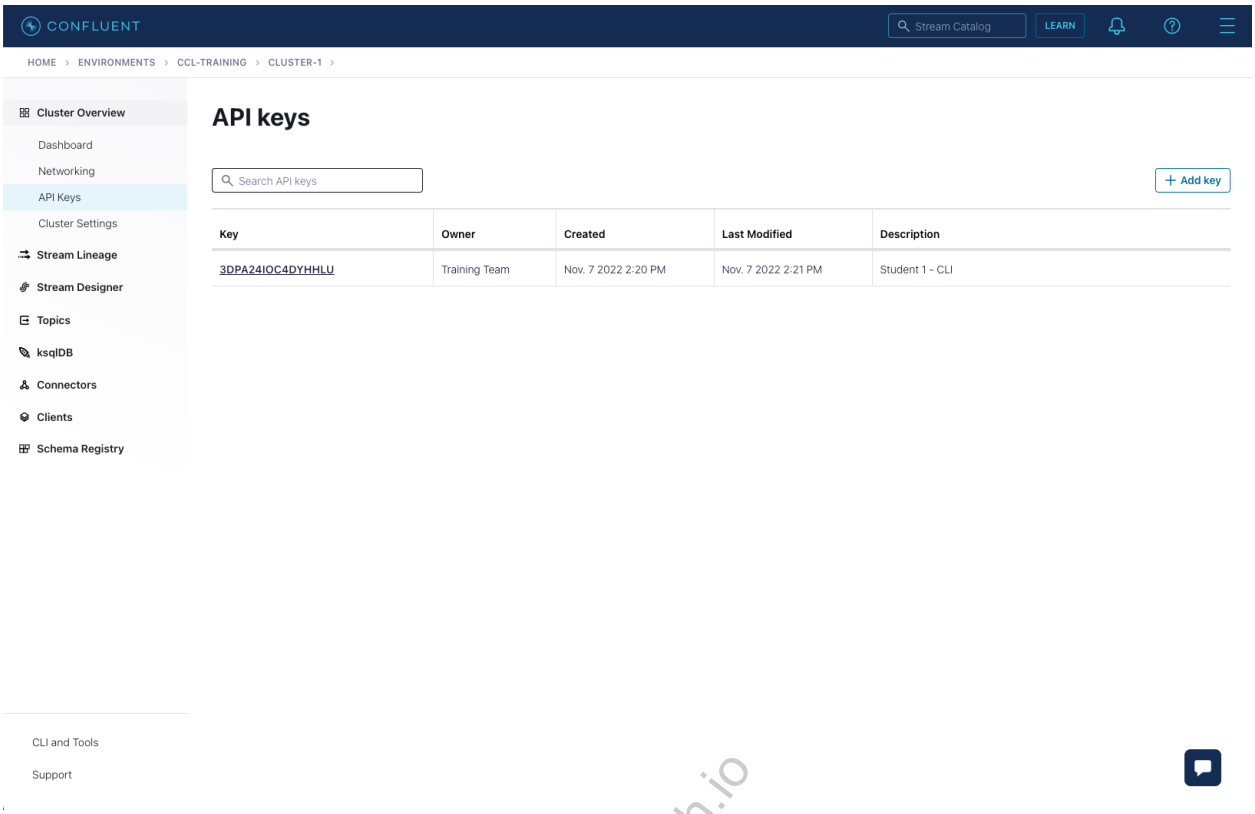
nitesh@datacouch.io

43. For Access, just click on **Next**. We are not assigning any role to this Service Account.
44. Check the information of the Service Account is correct and then click on **Create service account**.
45. Repeat the same steps to create a second Service Account called **JavaConsumerSA**.

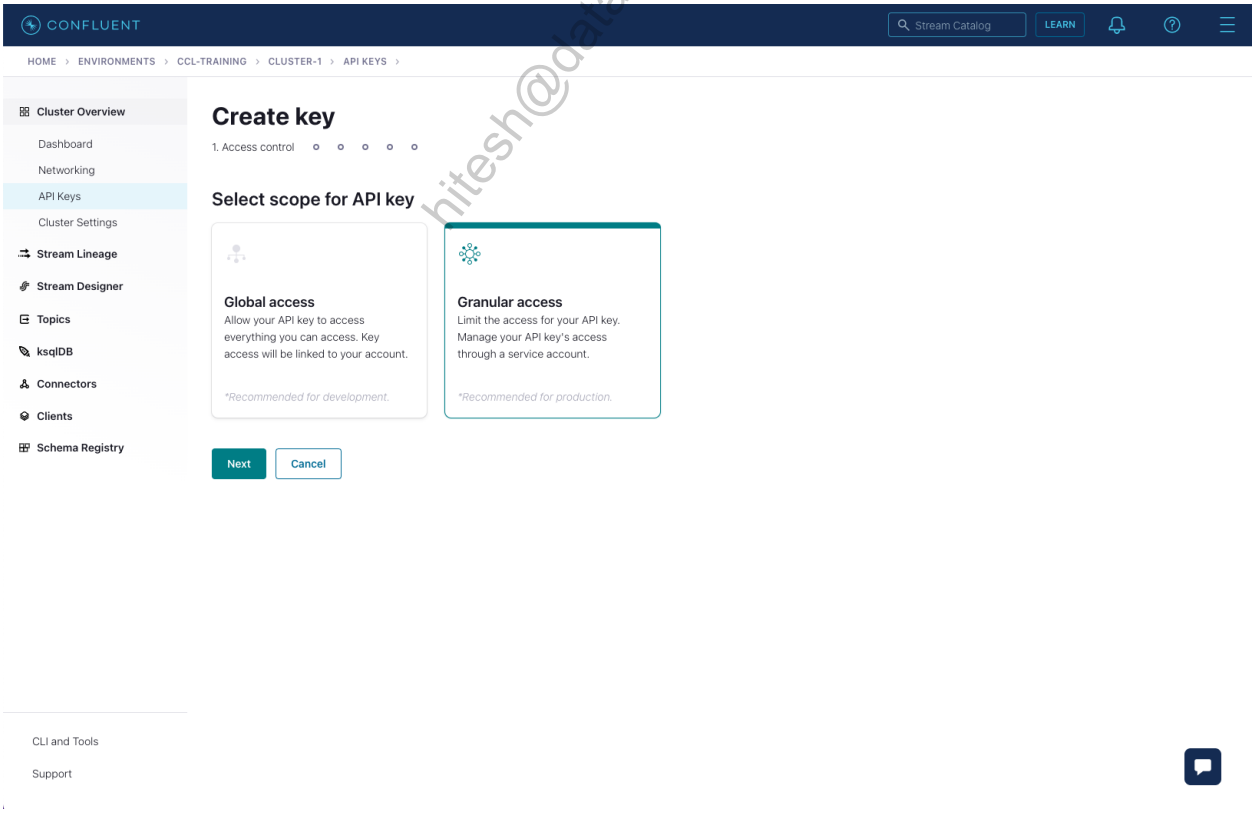
A service account needs to be associated with an API key to access a cluster. So, let's create a new API key/secret pair for our external applications.

46. Go to your Kafka cluster and in the left pane, select **API Keys** which is under **Cluster Overview**:
47. Click on **Add key**:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



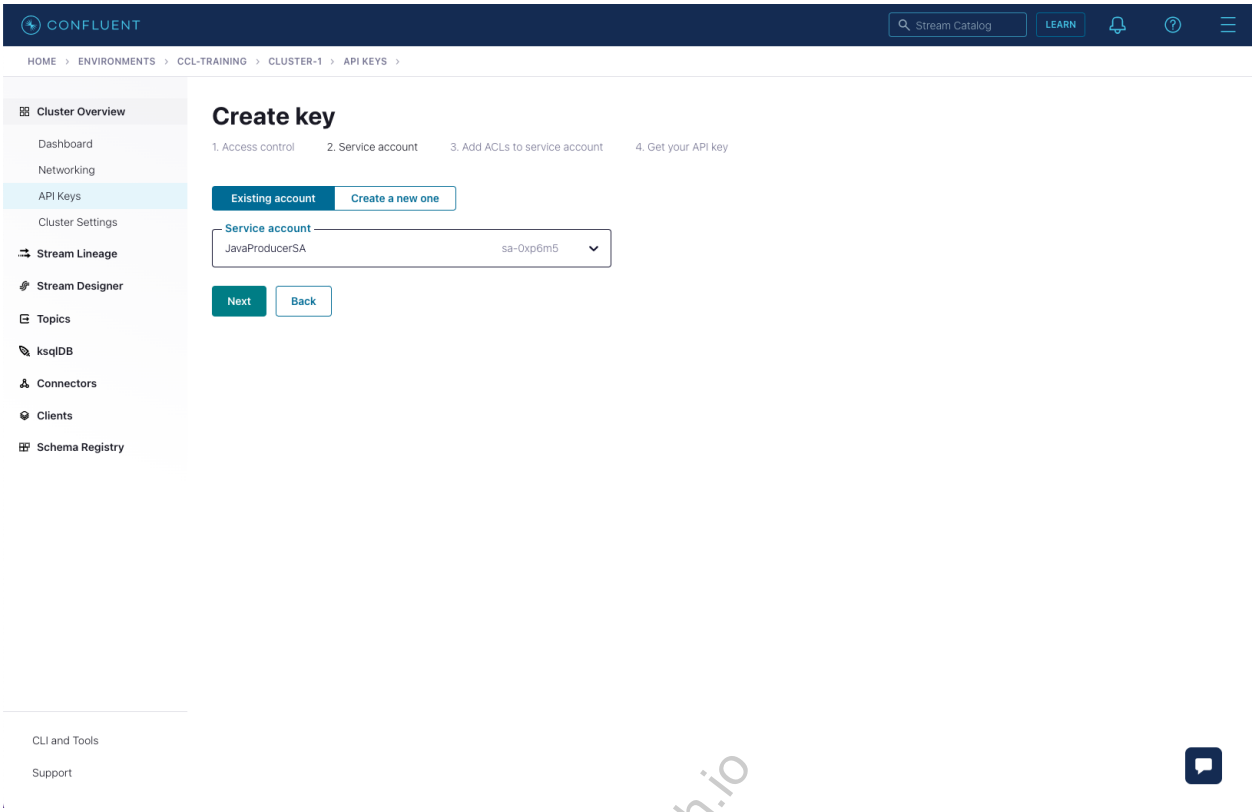
48. Choose **Granular access** and click **Next:**



hitesh@datacouch.io

49. In the dropdown menu, choose **JavaProducerSA** service account and click **Next:**

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

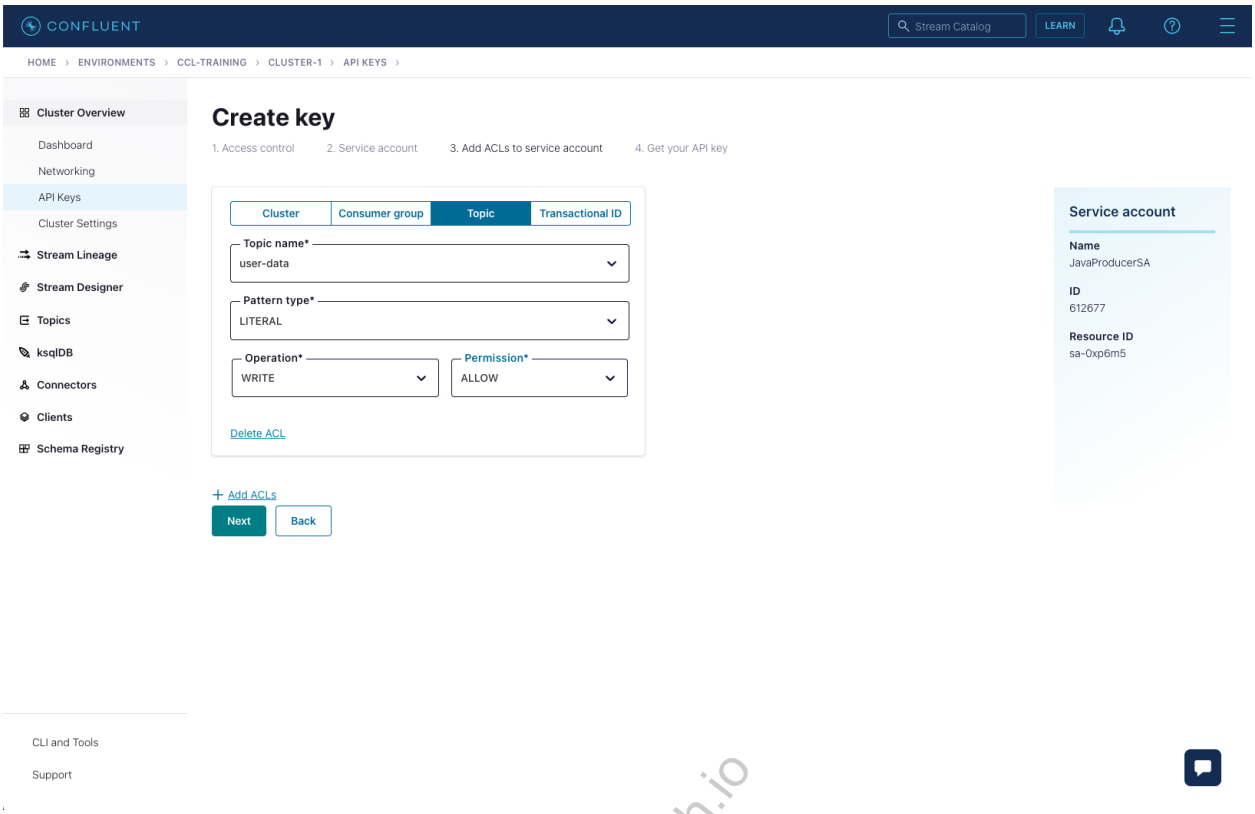


50. To allow an application to write data to `user-data` topic, we need to create an ACL with this configuration:

    - Resource: `Topic`
    - Topic name: `user-data`
    - Pattern type: `LITERAL`
    - Operation: `WRITE`
    - Permission: `ALLOW`

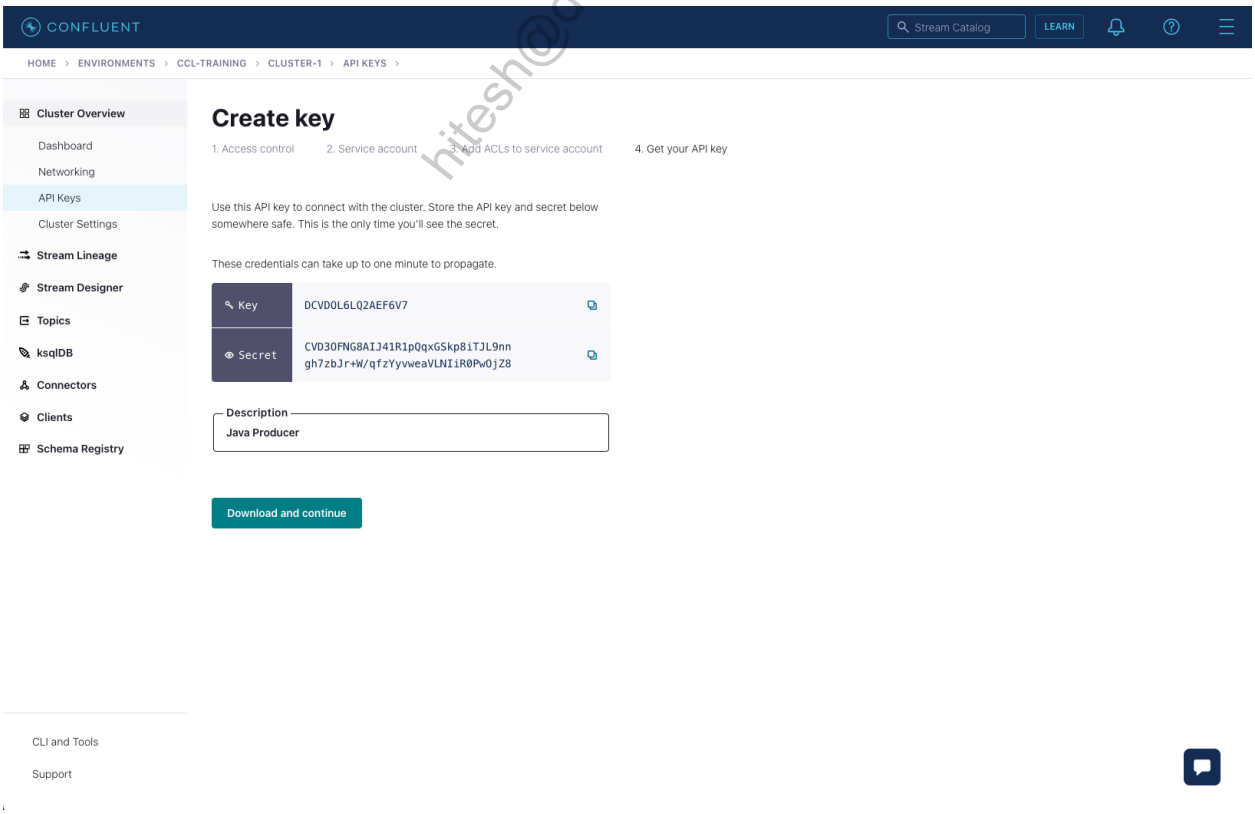
    Once you complete all fields with these values, click **Next**.

hitesh@datacouch.io



51. Give a **Description** to your new API key/secret pair to be able to recognize it in the future. Then, click on **Download and continue** to store your API key/secret locally.

hitesh@datascouch.io



52. Repeat the same steps again to create another API key/secret for the **Java Consumer Service Account**:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

a. Click **Add key**.

b. Choose **Granular acces** and click **Next**.

c. Select **JavaConsumerSA** service account and click **Next**.

d. To consume from a topic, the application requires **two ACLs**:

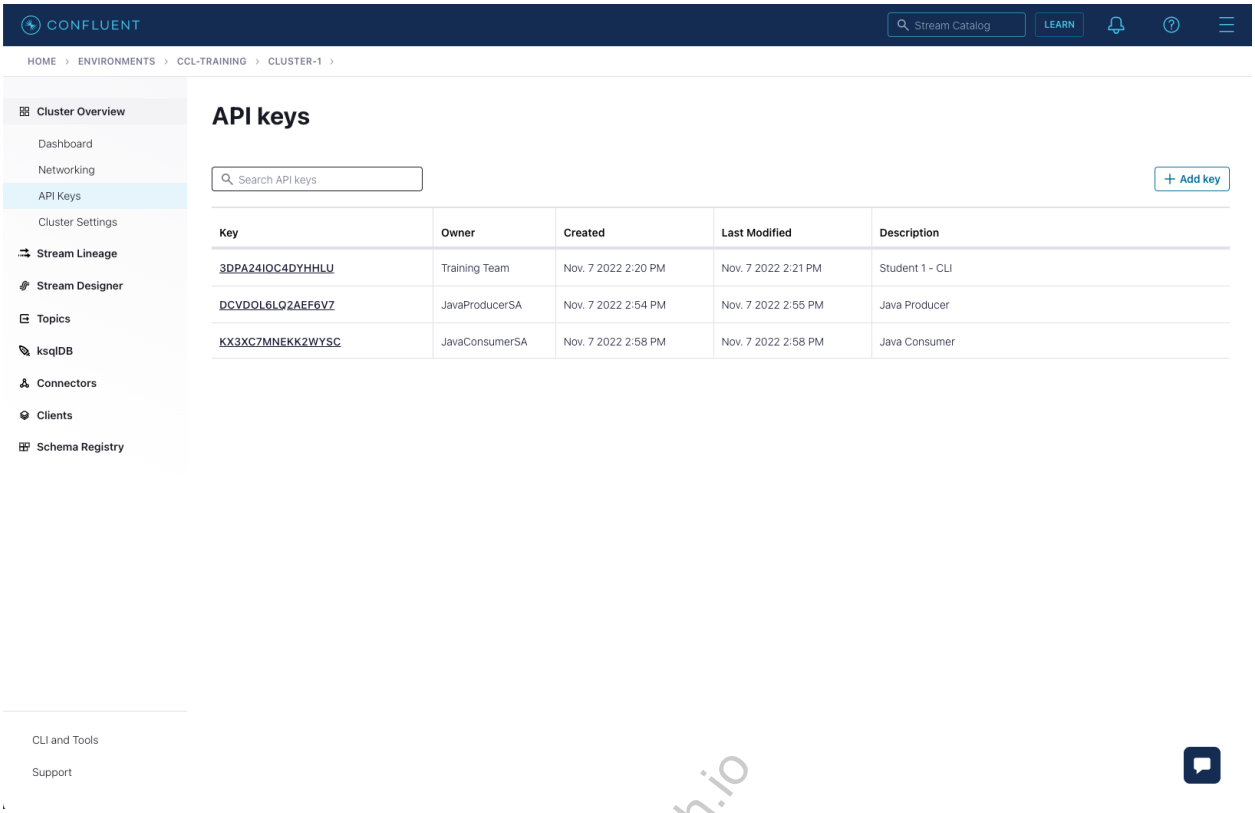
- First ACL:
  i. Resource: `Consumer group`
  ii. Consumer group ID, **Type**: `training-` , then hit **Enter**
  iii. Pattern type: `PREFIXED`
  iv. Operation: `READ`
  v. Permission: `ALLOW`
- Click **Add ACLs** to create a second ACL:
  i. Resource: `Topic`
  ii. Topic name: `user-data`
  iii. Pattern type: `LITERAL`
  iv. Operation: `READ`
  v. Permission: `ALLOW`

After setting both ACLs, click **Next**.

e. Type `Java Consumer` in the **Description** and click on **Download and continue** to store the API key/secret locally.

53. You will end with 3 different API keys:

hitesh@datacouch.io



54. Let's come back to the VM. Now, we need to configure the Java Producer and Consumer to use the new API keys and API secrets we've just created.
Let's start with the Java Producer. Run this command in a terminal window:

```
$ code ~/confluent-ccl/lab-accessing-ccl/java-producer/java-producer.properties
```

55. Replace `{{ BOOTSTRAP_SERVERS }}`, and the `{{ PRODUCER_API_KEY }}` and `{{ PRODUCER_API_SECRET }}` with the API key/secret that you just created for the Service Account `JavaProducerSA`.

56. Press `Ctrl+S` to save the changes.

57. Do the same for the Consumer replacing `{{ BOOTSTRAP_SERVERS }}`, and the `{{ CONSUMER_API_KEY }}` and `{{ CONSUMER_API_SECRET }}` with the API key/secret of `JavaConsumerSA`. Run this command in a terminal window:

```
$ code ~/confluent-ccl/lab-accessing-ccl/java-consumer/java-consumer.properties
```

58. Press `Ctrl+S` to save the changes.

hitesh@catacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

59. Let's start the Java Producer. First, go the **`java-producer`** directory. Run this command:

```
$ cd ~/confluent-ccl/lab-accessing-ccl/java-producer
```

60. Build the Java Producer application. Run:

```
$ ./gradlew build


BUILD SUCCESSFUL in 3s
6 actionable tasks: 4 executed, 2 up-to-date
```

61. Start the Java Producer application. Run this command:

```
$ ./gradlew run --console plain
> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE

> Task :run
Starting Java producer.
Message sent: 1|Neve|Woods|eget@cubiliaCurae.com|Jul 6,
1971|1599300233|true
Message sent: 2|Janna|Walker|quis@elitelitfermentum.ca|Aug 4,
1971|1534725157|true
Message sent: 3|Jolie|Craig|ligula.eu@aliquet.co.uk|Dec 7,
1981|1564221761|true
Message sent: 4|Daria|Perez|penatibus.et@consectetuercursus.net|Oct
19, 1979|1560815636|true
Message sent: 5|Acton|Pitts|Nunc.quis.arcu@pretiumneque.org|Nov 6,
1986|1532799850|false
Message sent: 6|Colby|Winters|in.tempus.eu@vel.org|May 23,
1988|1583641915|false
Message sent: 7|Allegra|Ortega|nisi.Dum.sociis@ategestasa.co.uk|Dec
10, 1969|1609210804|false
```

hitesh@datacouch.io

62. Open a new terminal window. Go the **`java-consumer`** directory. Run this command:

```
$ cd ~/confluent-ccl/lab-accessing-ccl/java-consumer
```

63. Build the Java Consumer application. Run:

```
$ ./gradlew build


BUILD SUCCESSFUL in 3s
6 actionable tasks: 4 executed, 2 up-to-date
```

64. Start the Java Consumer application. Run this command:

```
$ ./gradlew run --console plain
> Task :compileJava
> Task :processResources UP-TO-DATE
> Task :classes

> Task :run
Starting Java Consumer.
Key: 4 - Value:4|Daria|Perez|penatibus.et@consectetuercursus.net|Oct
19, 1979|1560815636|true
Key: 1 - Value:1|Neve|Woods|eget@cubiliaCurae.com|Jul 6,
1971|1599300233|true
Key: 7 - Value:7|Allegra|Ortega|nisi.Dum.sociis@ategestasa.co.uk|Dec
10, 1969|1609210804|false
Key: 3 - Value:3|Jolie|Craig|ligula.eu@aliquet.co.uk|Dec 7,
1981|1564221761|true
Key: 5 - Value:5|Acton|Pitts|Nunc.quis.arcu@pretiumneque.org|Nov 6,
1986|1532799850|false
Key: 6 - Value:6|Colby|Winters|in.tempus.eu@vel.org|May 23,
1988|1583641915|false
Key: 2 - Value:2|Janna|Walker|quis@elitelitfermentum.ca|Aug 4,
1971|1534725157|true
```

hitesh@datacouch.io

65. You can view the messages in Confluent Cloud Console. Go to **Topics** → **user-data** → **Messages** tab.

66. Press `Ctrl+C` to exit the Java Producer.

67. Press `Ctrl+C` to exit the Java Consumer.

Conclusion

In this lab you have learned how to install, configure and use Confluent CLI. You have also configured Kafka CLI tools to use them with Confluent Cloud. Finally you have created API keys and secrets for two Service Accounts providing different permissions with ACLs; and configured a Java producer and consumer to access Confluent Cloud.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

STOP HERE. THIS IS THE END OF THE EXERCISE.

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Lab 03 Working with Schema Linking

a. Working with Schema Linking

Objective

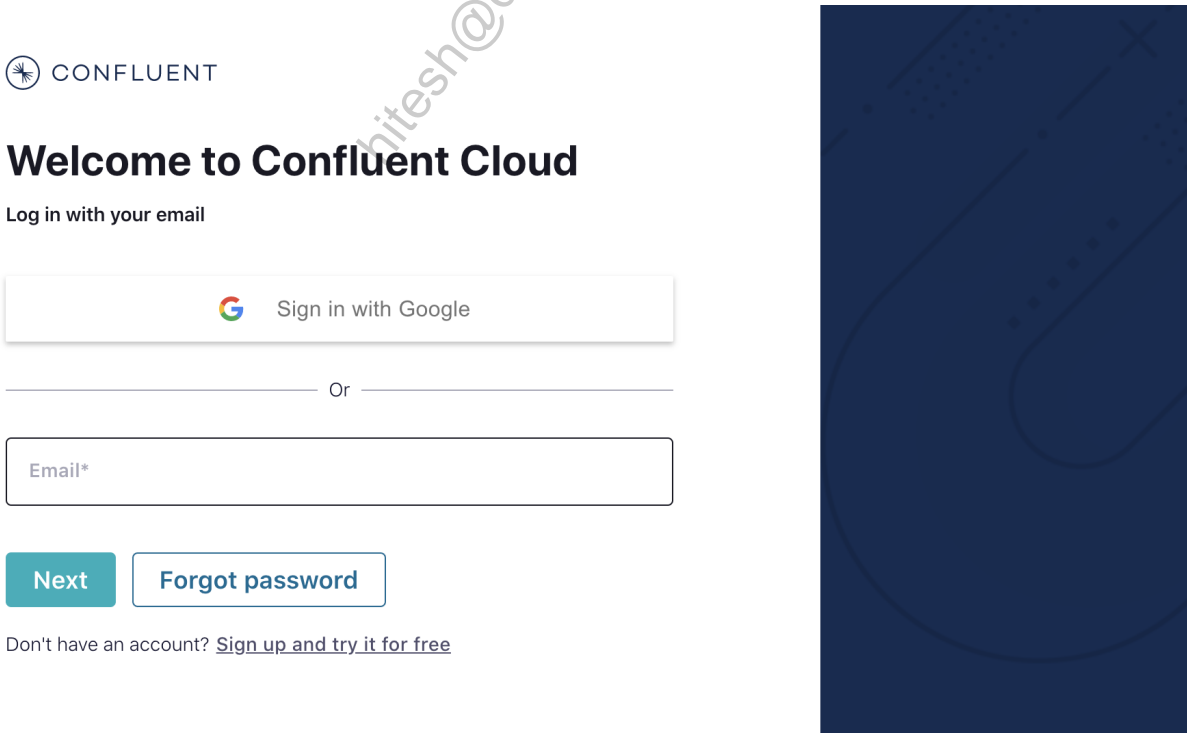
In this lab, we will use Schema Registry in Confluent Cloud creating schemas using the console, the Confluent CLI and the REST API. Then, we will focus on Schema Linking creating an exporter to sync schemas from one Schema Registry cluster to another Schema Registry cluster, and will modify its configuration to see how it will affect at destination.

Creating schemas in Confluent Cloud

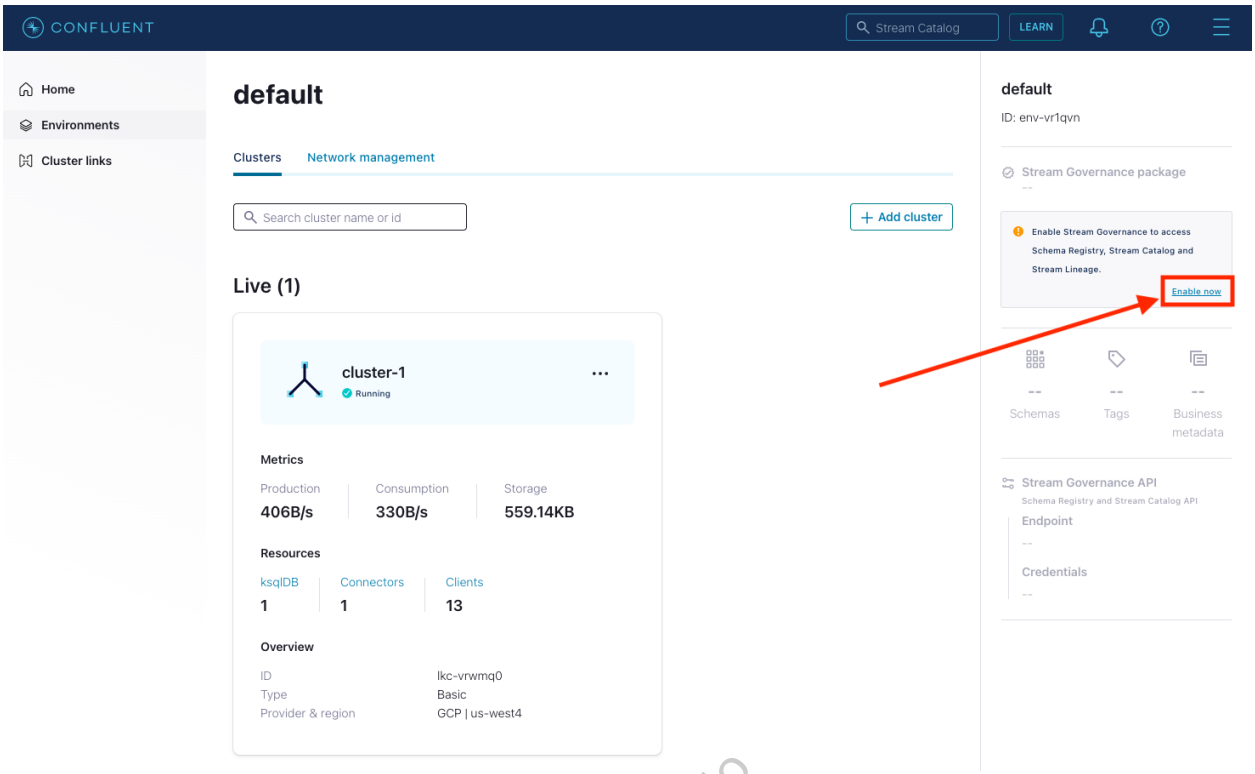
Let's start creating a few schemas in the Schema Registry cluster associated to **default** environment, which will be our source cluster.

hitesh@datacouch.io

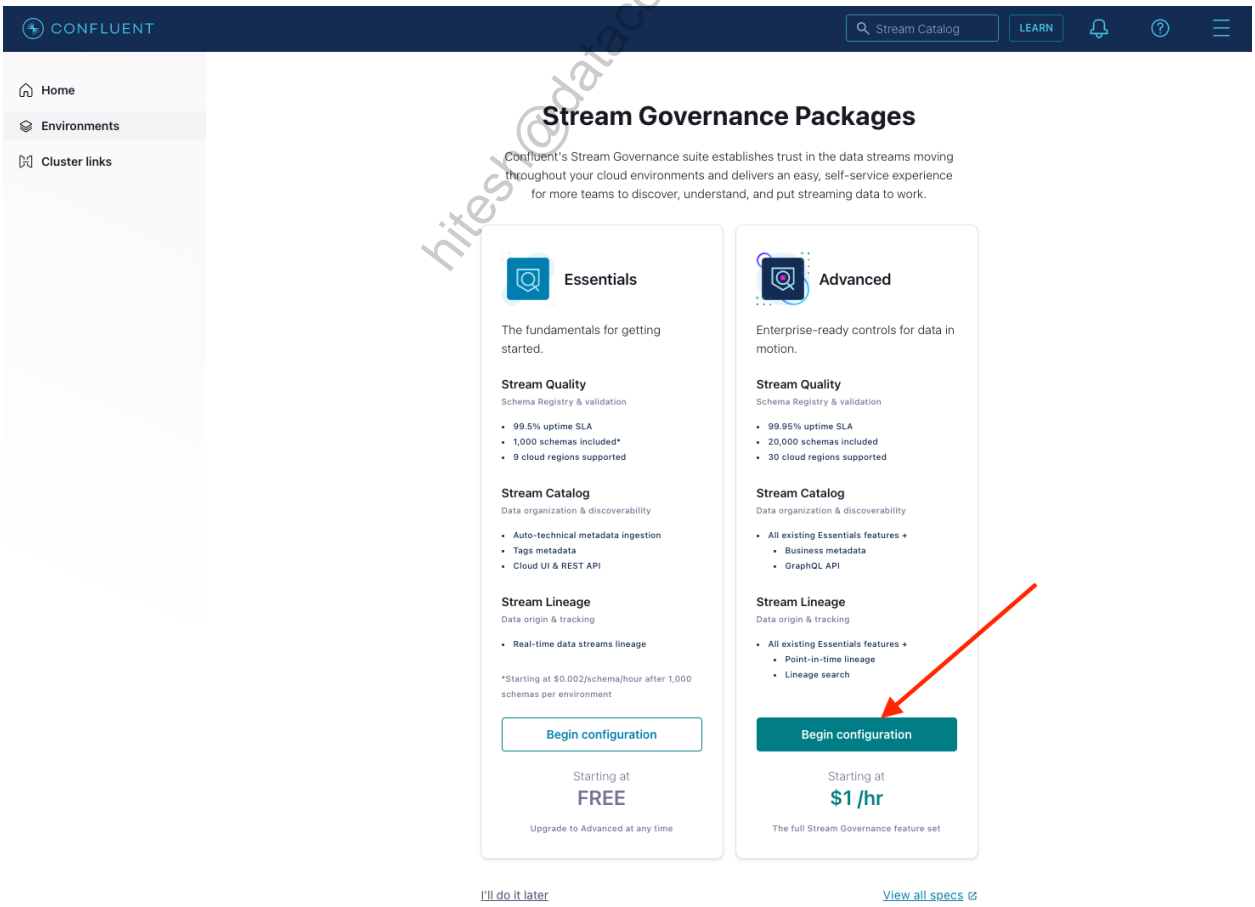
1. Go to [Confluent Cloud website] and log in to your account .



2. Select your **default** environment. In the right pane, click **Enable now** to activate Stream Governance in this environment:



3. Select the **Advanced** package to enable all features:

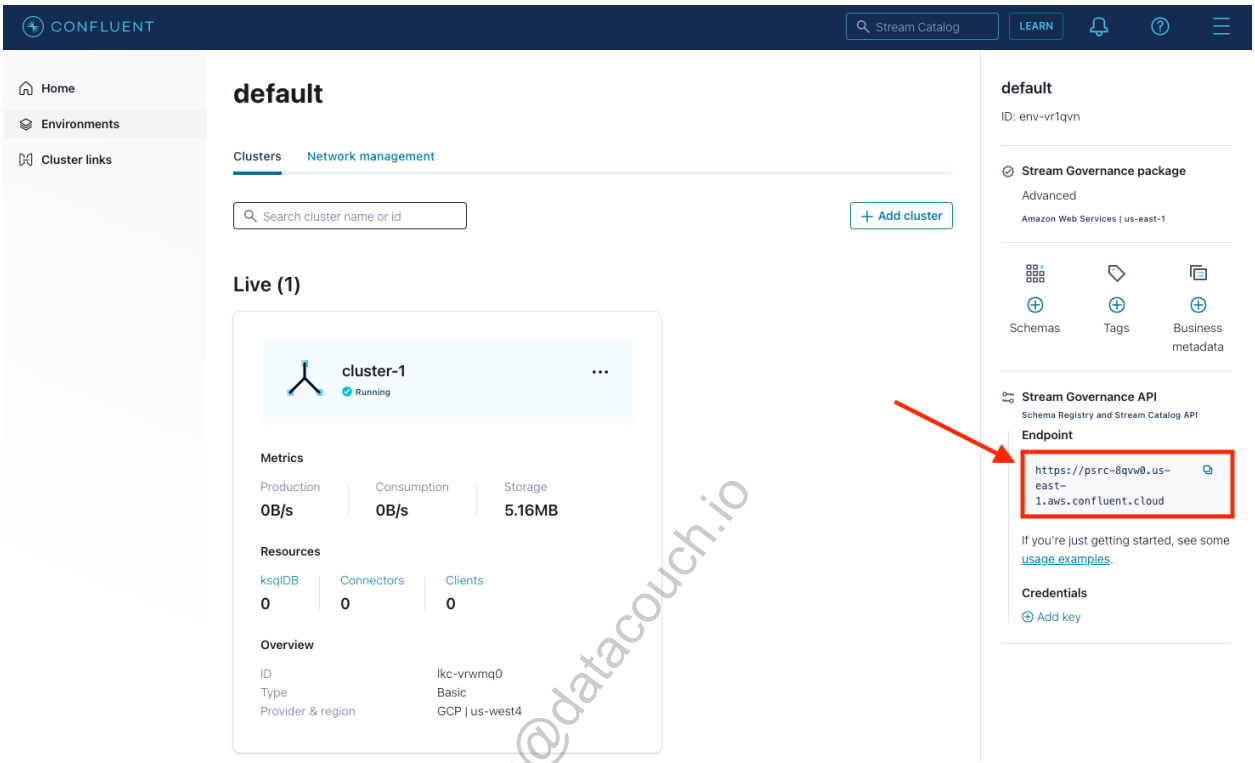


4. Choose your preferred **cloud provider** and **region**. Costs will vary with these choices, but

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

they are clearly shown on the dropdown, so you'll know what you're getting. Then, click on **Enable**.

Schema Registry should be now enabled, so you can find its URL in the right pane of the **default** environment page.



5. Copy the Schema Registry cluster **API endpoint**. Open a terminal and run this command replacing `<Paste your SR URL>` with your **API endpoint**:

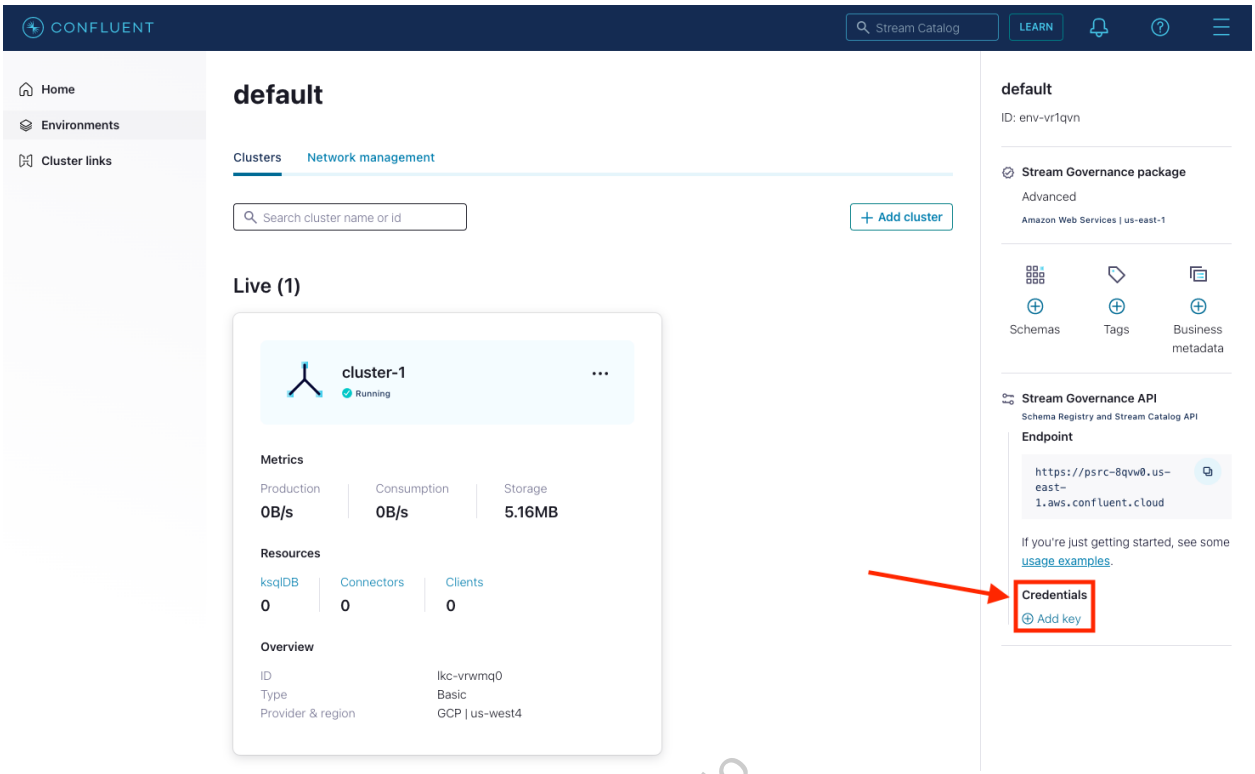
```
$ export SR_URL=<Paste your SR URL>
```

Sample

```
$ export SR_URL=https://psrc-2312y.europe-
west3.gcp.confluent.cloud
```

6. Go back to Confluent Cloud UI. Now, we need a Schema Registry API key and secret to access this logical Schema Registry cluster. Click on **+ Add key** under **Credentials** in the right side of the page.

nitish@datacouch.io



7. Then, follow these instructions:

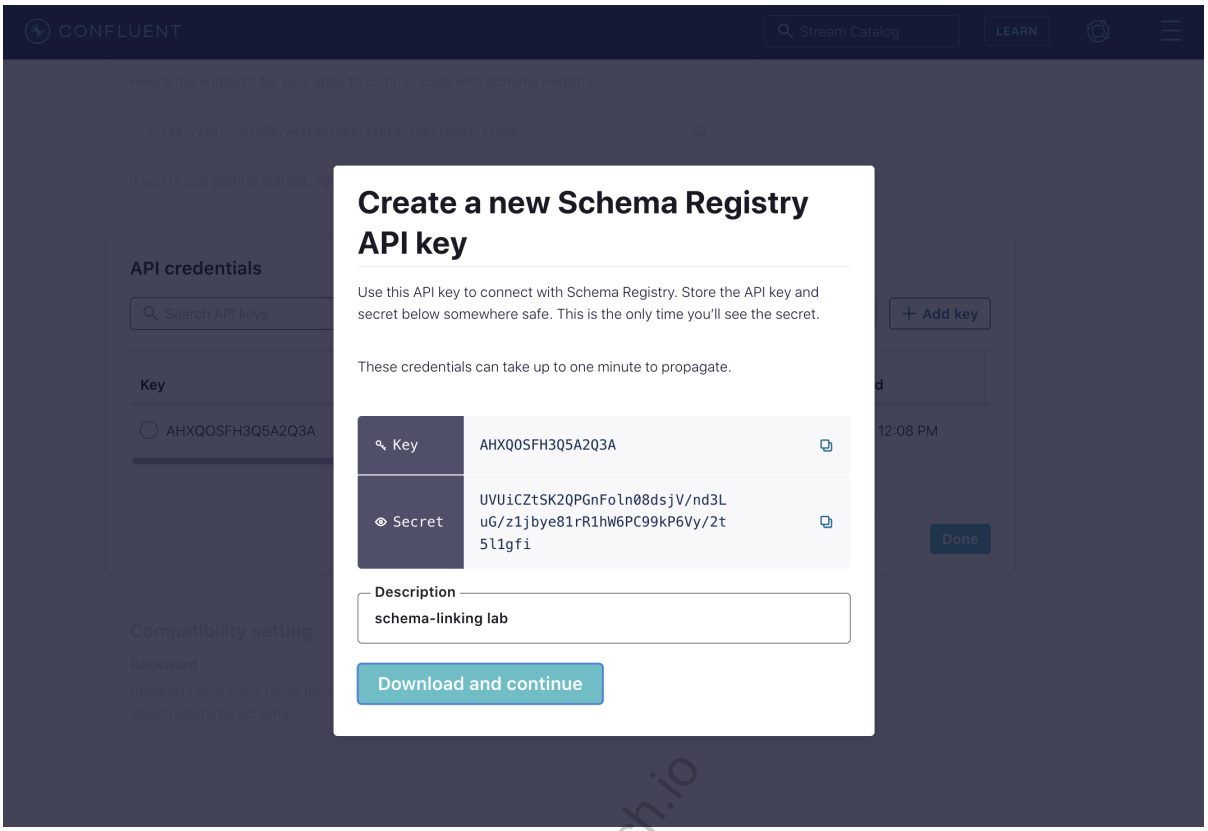
   a. Click **Create key**.

   b. Give a description to identify this new API key in the future.

   c. Click **Download and continue**.

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



8. Go to your terminal window and run these two commands replacing `<Paste your SR KEY>` with your **SR API Key** and `<Paste your SR SECRET>` with your **SR API Secret**:

```
$ export SR_KEY=<Paste your SR KEY>
```

```
$ export SR_SECRET=<Paste your SR SECRET>
```

9. Run the following script to add this Schema Registry information in the `~/.bashrc` file, so these variables will be available in new terminal windows.

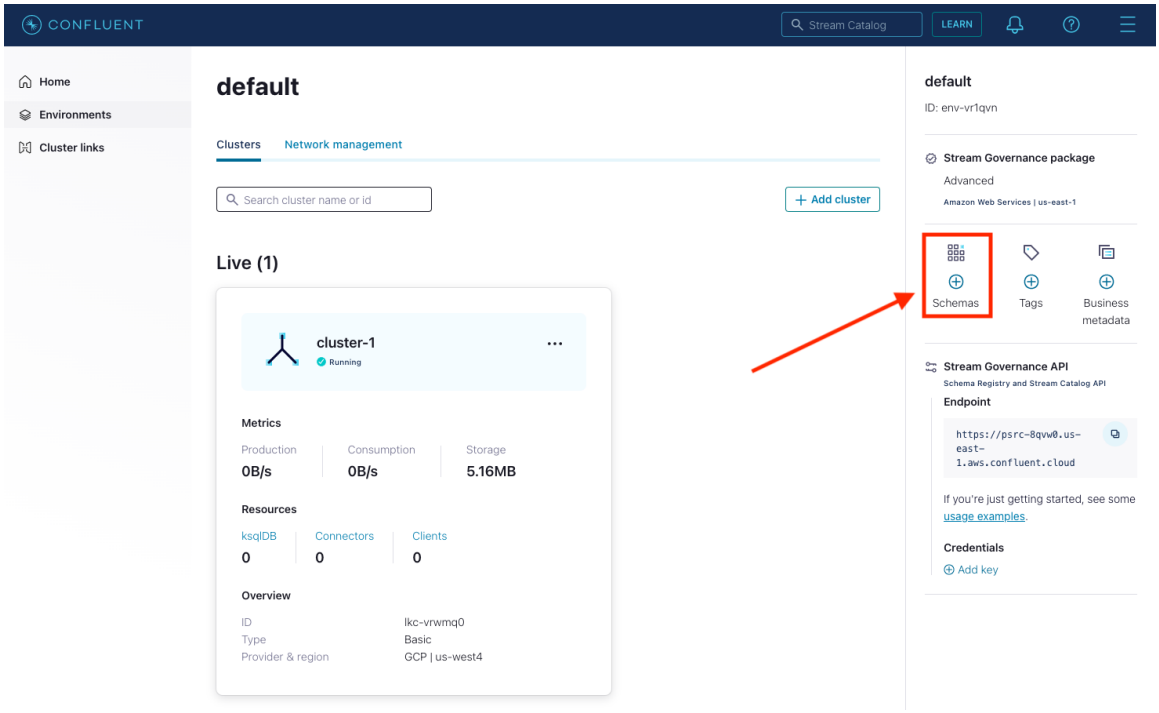
```
$ ~/confluent-ccl/lab-schema-linking/add-sr-info.sh
```

10. Now, let's create schemas using three different methods:

    a. Using Confluent Cloud console:

    - In the **default** environment page, go to the right side of the page and click on **Schemas**.

hitesh@datacouch.io



- Click on **Add schema**.
- Type the subject name with this value: `:.lab:users-value`.
- Select the **Avro** schema.
- Delete the pre-populated schema sample and paste this schema:

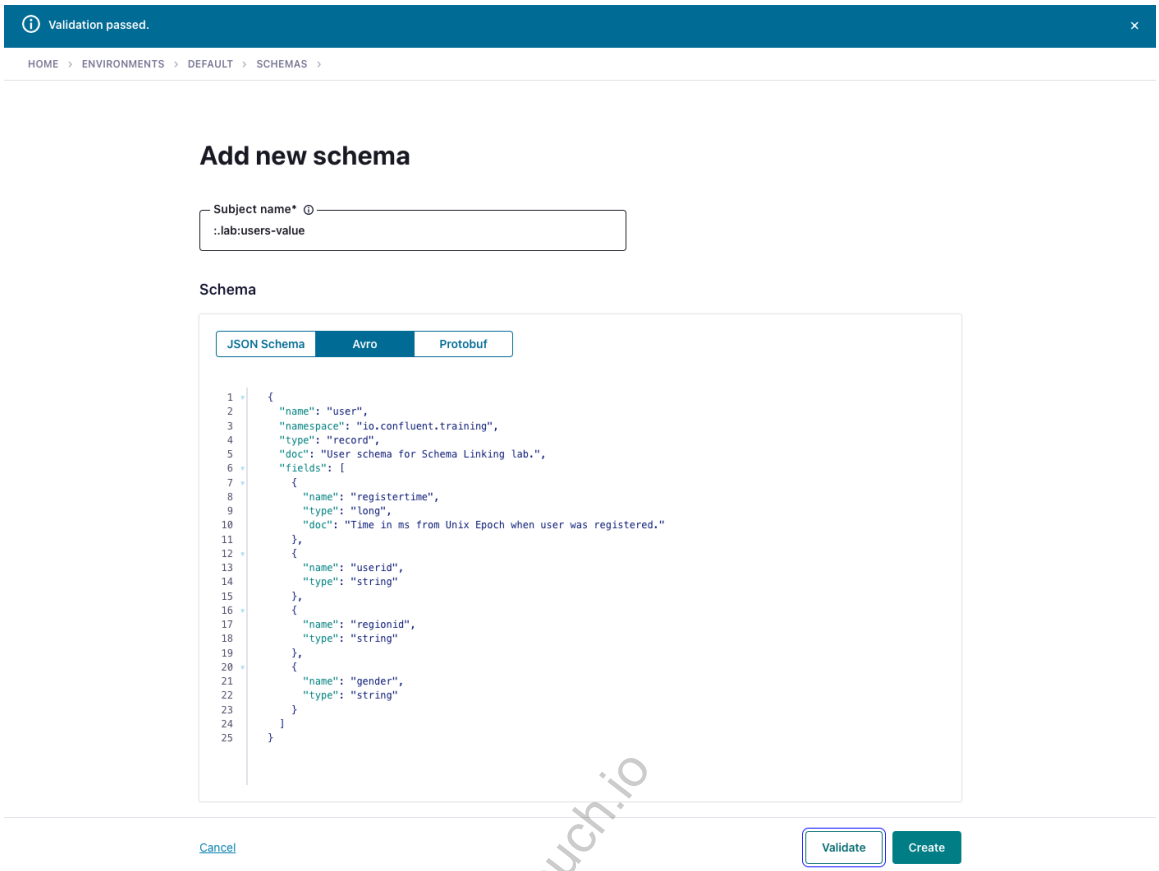
hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
{
  "name": "user",
  "namespace": "io.confluent.training",
  "type": "record",
  "doc": "User schema for Schema Linking lab.",
  "fields": [
    {
      "name": "registertime",
      "type": "long",
      "doc": "Time in ms from Unix Epoch when user was
registered."
    },
    {
      "name": "userid",
      "type": "string"
    },
    {
      "name": "regionid",
      "type": "string"
    },
    {
      "name": "gender",
      "type": "string"
    }
  ]
}
```

- Click **Validate** to check the schema corresponds to an Avro schema.

hitesh@datacouch.io



- Click **Create**.

b. Using Confluent CLI:

- In the Terminal, navigate to the lab folder:

```
$ cd ~/confluent-ccl/lab-schema-linking
```

- Log in to your Confluent Cloud account using the Confluent CLI:

```
$ confluent login
Enter your Confluent Cloud credentials:
Email: <enter email>
Password: <enter password>
```

- Run this command to create a new schema called `product` within the subject `:.lab:products-value`. The schema is defined in a JSON file called `product-schema.json` in the lab folder.

```
$ confluent schema-registry schema create --subject
:.lab:products-value --schema product-schema.json
Successfully registered schema with ID "100002".
```

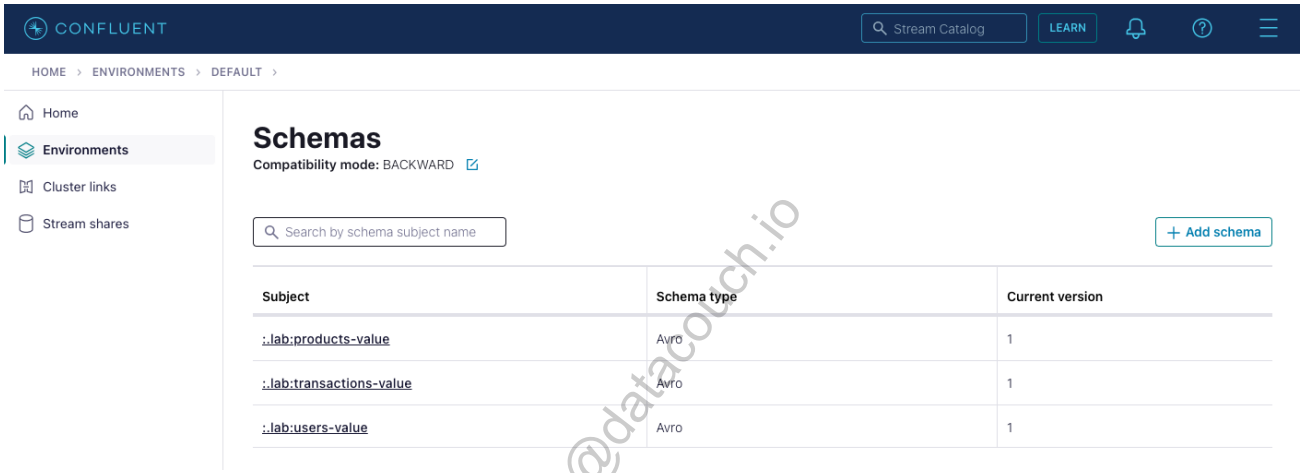
hitesh@datacouch.io

c. Using Confluent Cloud REST API:

- In the Terminal, run this command using the tool **curl** to create a new schema called `transaction` within the subject `:.lab:transactions-value`.

```
$ curl -u $SR_KEY:$SR_SECRET -H "Content-Type: application/json"
$SR_URL/subjects/:.lab:transactions-value/versions --data
@transaction-schema.json -X POST
{"id":100003}
```

You should end up with three subjects within the same context `.lab`.

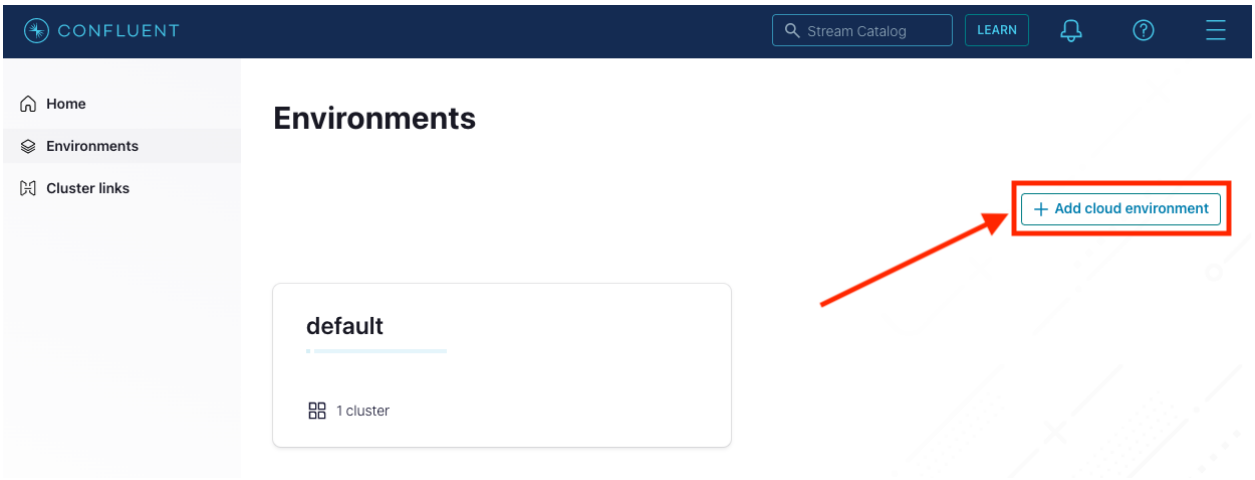


mesh@datacouch.io

Creating a new Schema Registry cluster

In this part, we will use Schema Linking to export all the subjects in the context `.lab` to another Schema Registry cluster associated to a new environment. So, let's create it:

11. In Confluent Cloud UI, go to **Environments** and click on **Add cloud environment**.



© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

12. Give it the name **destination-env**.

13. In the Stream Governance Packages window, select the **Essentials** package which should be more than enough to complete this lab as we don't need the other extra features for this new environment.
    Click on **Begin configuration** under the **Essentials** package.

14. Choose a **cloud provider** and a **region**. Schema Linking can sync schemas between any cloud providers and regions. Then, click on **Enable**.

Using Schema Linking

Let's start first setting up the Schema Link, and then we will make changes in the Source cluster to visualize how those changes are automatically applied in the Destination Schema Registry cluster.

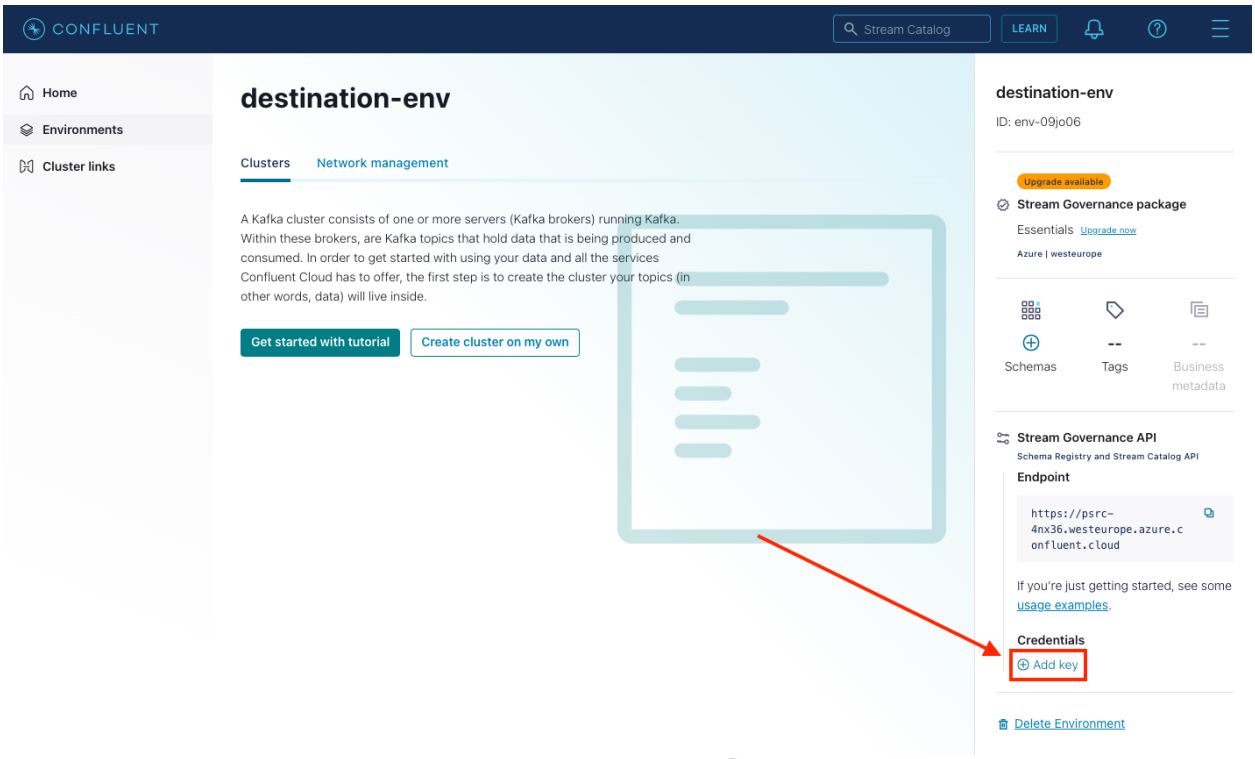
Since the Schema Linking Exporter is located in the Source cluster, it needs credentials to access the Destination Schema Registry cluster and to know where it is located:

- Destination SR URL
- Destination SR API Key
- Destination SR API Secret

hitesh@datacouch.io

15. Click on **Add key** under **Credentials** in the right side of the page.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



16. Then, follow these instructions:

    a. Click **Create key**.

    b. Give a description to identify this new API key in the future.

    c. Click **Download and continue**.

hitesh@datacouch.io

Now, let's create a configuration file with this credential information and the Schema Registry Endpoint. We will pass this configuration file to the Schema Linking Exporter, so it will be able to access the Destination cluster.

17. Open the configuration file `dest-SR.config` in VS Code:

```
$ code ~/confluent-ccl/lab-schema-linking/dest-SR.config
```

- Replace `{{ DEST_SR_URL }}` with the Schema Registry Endpoint of the Destination cluster
- Replace `{{ DEST_SR_KEY }}` with the API Key you have just generated
- Replace `{{ DEST_SR_SECRET }}` with the API Secret you have just generated

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
schema.registry.url={{ DEST_SR_URL }}
basic.auth.credentials.source=USER_INFO
basic.auth.user.info={{ DEST_SR_KEY }}:{{ DEST_SR_SECRET }}
```

Press **Ctrl+S** to save the changes.

Sample

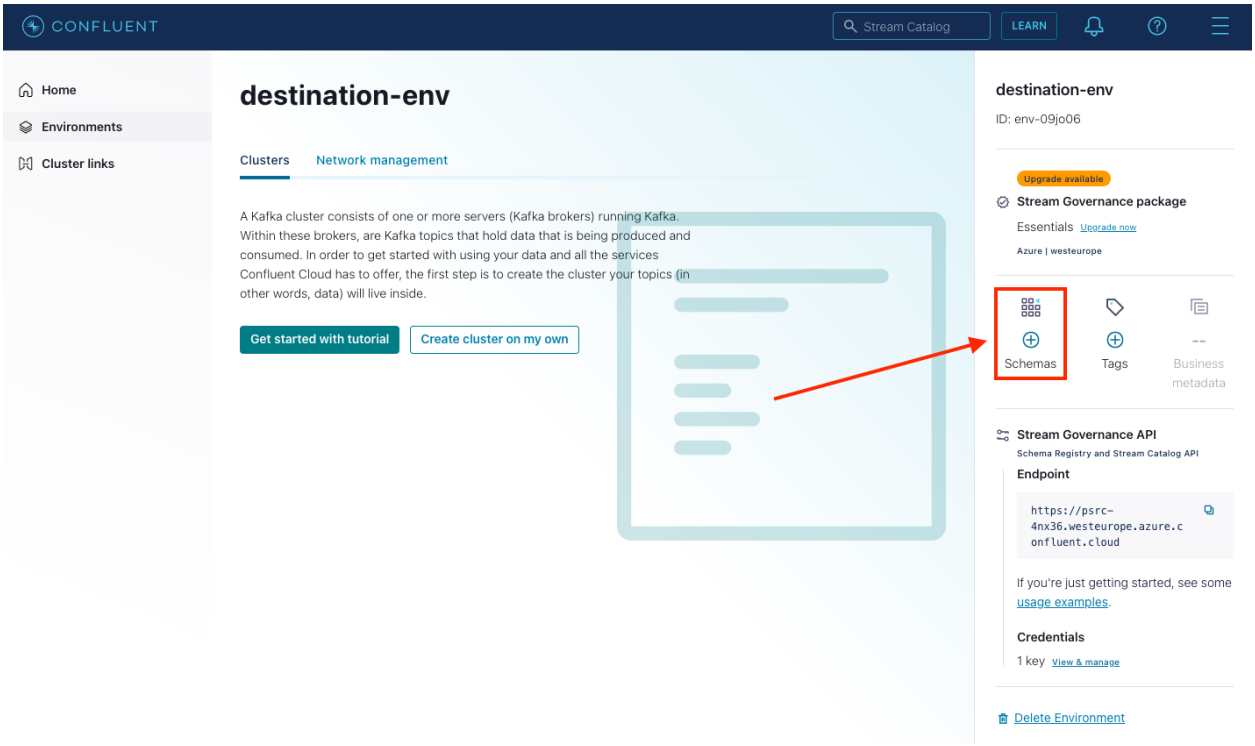
```
schema.registry.url=https://psrc-
4ny36.westeurope.azure.confluent.cloud
basic.auth.credentials.source=USER_INFO
basic.auth.user.info=QYZSOSFMQNTTTW3C:inUhDqXiy4AaOLx5TML1loPoR
vodFxdAH9g6YR2RBL05hkQaMYNAuAxHpeueGteq
```

18. Go back to the Terminal window and use the Confluent CLI to create a new exporter called **lab-exporter**. Configure it to export all the subjects within the context **.lab**. Run this command:

```
$ confluent schema-registry exporter create lab-exporter --subjects
":.lab:*" --context-type NONE --config-file dest-SR.config
Created schema exporter "lab-exporter".
```

19. Open Confluent Cloud Console and go to the **Schema Registry** in the **destination-env** environment by clicking the button **Schemas** on the right pane.

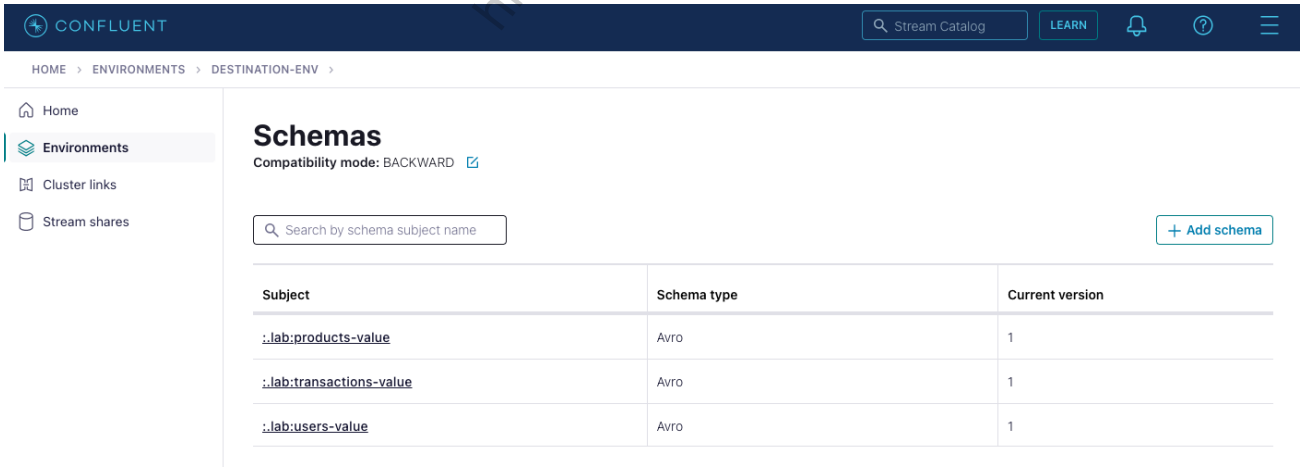
hitesh@datatouch.io



20. In **Schema Registry**, you should see the subjects exported by the new `lab-exporter`.

nitesh@danacouch.io

Note that the context name at destination is exactly the same as the context name at source. Later in this lab you will change the exporter configuration to change the context name at destination.



21. Now, let's add a new schema in the **Source Schema Registry cluster** to see if it is automatically exported to the Destination cluster. Run this command in a Terminal window to create the new schema `order-schema` in the subject `:.lab:orders-value`:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ confluent schema-registry schema create --subject :.lab:orders-value --schema order-schema.json
Successfully registered schema with ID 100004
```

22. Check in Confluent Cloud Console in the **Destination Schema Registry cluster** that this new subject has been automatically exported.

23. Now in the **Source Schema Registry cluster**, let's create a new schema version in the subject `:.lab:transactions-value` by adding an optional field called `amount` to the previous version. Run this command in a Terminal window to evolve the schema to version 2 :

```
$ curl -u $SR_KEY:$SR_SECRET -H "Content-Type: application/json" $SR_URL/subjects/:.lab:transactions-value/versions --data @transaction-schema-v2.json -X POST
{"id":100005}
```

24. Check again in Confluent Cloud website in the **Destination Schema Registry cluster** that this new version has been automatically exported.

hitesh@datacouch.io

Updating the Context Type configuration of the Exporter

Previously, you've created the Exporter using the `--context-type=NONE`, which copies the source context as-is, without prepending anything. This is useful to make an exact copy of the source Schema Registry in the destination.
In this part, you'll investigate the other two context types `AUTO` and `CUSTOM`.

Context Type	Description
`AUTO`	Prepends the source context with an automatically generated context, which is `.lsrc-xxxxxx` for Confluent Cloud. This is the **default**.
`CUSTOM`	Prepends the source context with a custom context name.
`NONE`	Copies the source context as-is, without prepending anything.

25. First, you will update the Exporter configuration to use `--context-type=AUTO`. In order to apply an update to an active Exporter, we must first pause the Exporter. Run the

following command :

```
$ confluent schema-registry exporter pause lab-exporter
Paused schema exporter "lab-exporter".
```

26. Now, you can apply the configuration update modifying the `--context-type=AUTO` by running this command :

```
$ confluent schema-registry exporter update lab-exporter --subjects
":.lab:*" --context-type AUTO --config-file dest-SR.config
Updated schema exporter "lab-exporter".
```

27. Since this Exporter has already exported all the schemas in the previous steps of the lab, we need to reset its offset so it can re-export all schemas again but to the new context name `.lsrc-xxxxxx.lab`.

```
$ confluent schema-registry exporter reset lab-exporter
Reset schema exporter "lab-exporter".
```

Remember that the Exporter is consuming the schemas from the internal topic `_schemas`, so we need to reset its offset to the beginning of the topic to re-consume all schemas and re-export them to destination.

hitesh@datacouch.io

28. You can check the Exporter offset by running this command:

```
$ confluent schema-registry exporter get-status lab-exporter
+--------------------+--------------+
| Name               | lab-exporter |
| Exporter State     | PAUSED       |
| Exporter Offset    |           -1 |
| Exporter Timestamp |            0 |
| Error Trace        |              |
+--------------------+--------------+
```

29. Now, you can resume the Exporter `lab-exporter` by running this command:

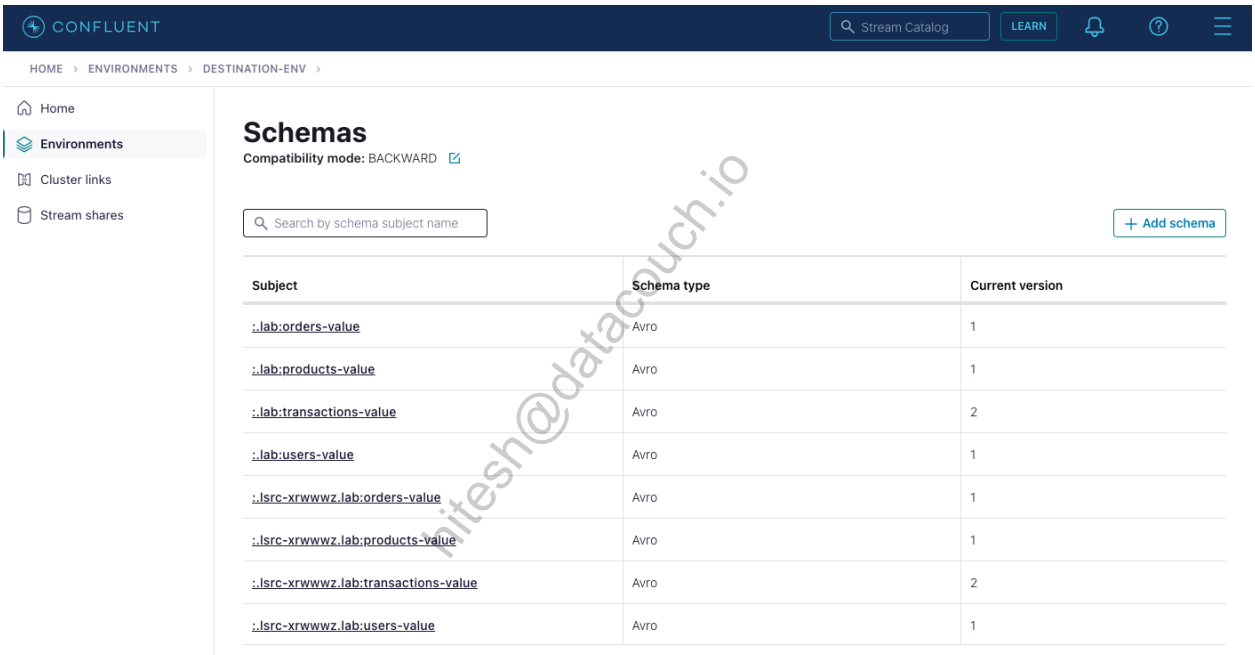
```
$ confluent schema-registry exporter resume lab-exporter
Resumed schema exporter "lab-exporter".
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

30. Check the offset of your exporter again. Run:

```
$ confluent schema-registry exporter get-status lab-exporter
+--------------------+---------------+
| Name               | lab-exporter  |
| Exporter State     | RUNNING       |
| Exporter Offset    |       4928650 |
| Exporter Timestamp | 1645535538242 |
| Error Trace        |               |
+--------------------+---------------+
```

31. Go to Confluent Cloud website in the Destination Schema Registry cluster. Check that the subjects have been exported in the new context name `.lsrc-xxxxxx.lab`.



nitesh@datacouch.io

32. Repeat the same steps explained previously to update the Exporter configuration to `--context-type=CUSTOM`:

    a. Pause the Exporter:

```
$ confluent schema-registry exporter pause lab-exporter
Paused schema exporter "lab-exporter".
```

    b. Update the Exporter configuration. Choose a `WORD` to prefix your `CUSTOM` context name:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ confluent schema-registry exporter update lab-exporter
--subjects ":.lab:*" --context-type CUSTOM --context-name {{ WORD
}} --config-file dest-SR.config
Updated schema exporter "lab-exporter".
```

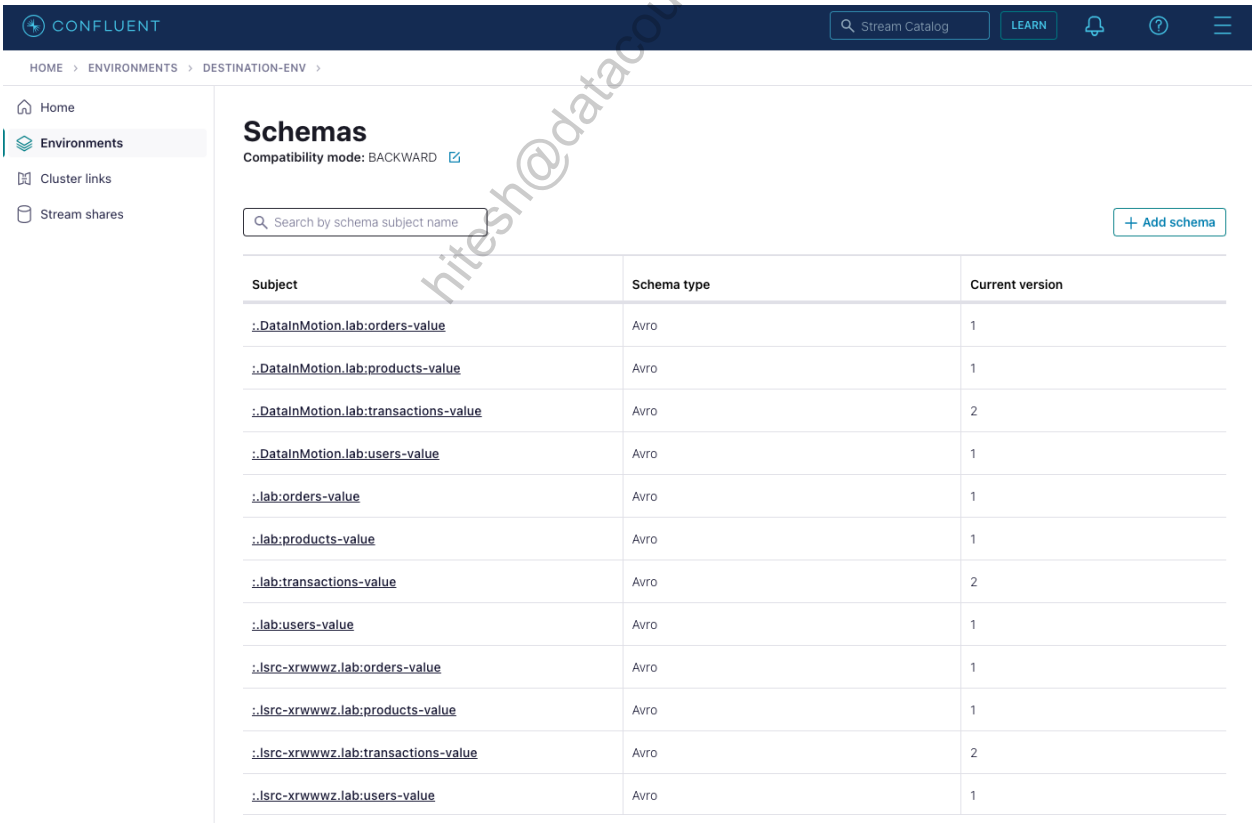
c. Reset the Exporter:

```
$ confluent schema-registry exporter reset lab-exporter
Reset schema exporter "lab-exporter".
```

d. Resume the Exporter:

```
$ confluent schema-registry exporter resume lab-exporter
Resumed schema exporter "lab-exporter".
```

33. Check in Confluent Cloud Console in the Destination SR cluster that the subjects have been exported in your custom context name `.{{ WORD }}.lab`.



hitesh@datacouch.io

Clean-up

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

IMPORTANT: Follow these instructions to stop incurring in additional charges.

34. Before deleting the Exporter, we need to pause it. Run this command:

```
$ confluent schema-registry exporter pause lab-exporter
Paused schema exporter "lab-exporter".
```

35. Run the following command to delete the Exporter:

```
$ confluent schema-registry exporter delete lab-exporter
Deleted schema exporter "lab-exporter".
```

36. OPTIONAL - Delete all subjects you have created. You can delete them individually using:

    a. Confluent Cloud Schema Registry console and delete each subject entirely.

    b. Confluent CLI specifying the qualified subject name. But first you need to select the appropriate **Environment**:

```
$ confluent environment list
  Current |     ID     |       Name
----------+------------+-------------------
  *       | env-d9q617 | default
          | env-knr3v6 | destination-env
```

```
$ confluent environment use {{ ENVIRONMENT_ID }}
```

    Now, you can use the Confluent CLI specifying the qualified subject name:

```
$ confluent schema-registry schema delete --subject {{ qualified
subject name }} --version all --force
```

    c. Confluent REST API specifying the qualified subject name:

```
$ curl -u $SR_KEY:$SR_SECRET -H "Accept: application/json"
$SR_URL/subjects/{{ qualified subject name }} -X DELETE
```

nitesh@datacouch.io

Conclusion

In this lab you have learned how to create schemas using contexts in three different ways. You have also learned how to create an Exporter to sync schemas between two different clusters. And finally you have worked with the exporter parameter `--context-type` to see the implications of changing its value at destination cluster.

STOP HERE. THIS IS THE END OF THE EXERCISE.

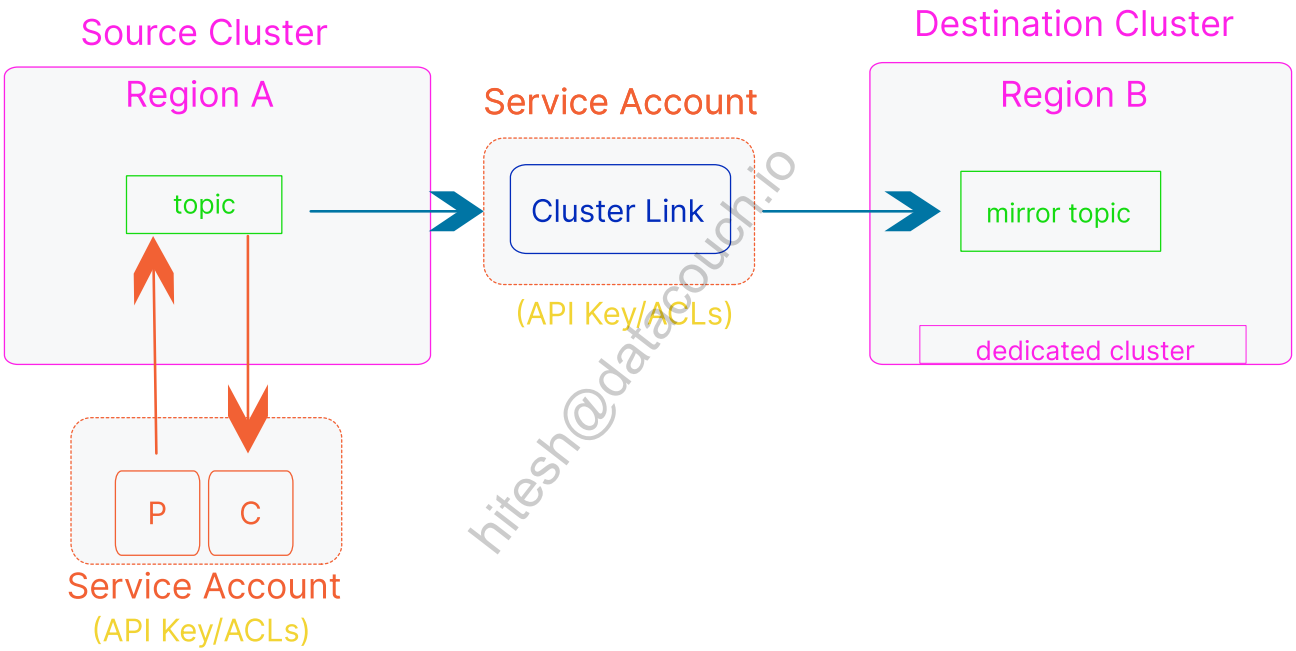
hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Lab 05 Cluster Linking

a. Setting up Cluster Linking in Confluent Cloud

The goal of this lab is to get familiar with Cluster Linking by creating a cluster link, updating its configuration and examining its behavior. You will perform all these common tasks following recommended best practices. Finally, you will also migrate Kafka clients from the source to the destination cluster.



hitesh@datascouch.io

Create a Dedicated cluster as the Destination cluster

Cluster Linking requires the destination cluster to be a Dedicated cluster. So in this section, you will use the Environment **destination-env** to create a new Dedicated cluster named **destination-cluster**.

1. Go to Confluent Cloud website and log in to your Confluent Cloud account.
2. Navigate to **Environments**. Then, click on **destination-env** environment.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

If you don't have the **destination-env** environment, go to Lab 4 and re-do Steps 9 to 12.

3. Click on **Create cluster on my own**. Then, follow these instructions:
   a. Cluster type: Go to the **Dedicated** cluster and select **1 CKU** by dragging the slider. Then click on **Begin Configuration**.
   b. Region/zones: Choose your preferred Cloud Provider and Region. Select **Single zone**. Click on **Continue**.
   c. Networking: Select **Internet**, then click on **Continue**.
   d. Security: Select **Automatic**, then click on **Continue**.
   e. Set payment: Click on **Skip payment**.
   f. Review and launch: Set the cluster name to **destination-cluster**. Review and click on **Launch cluster**.

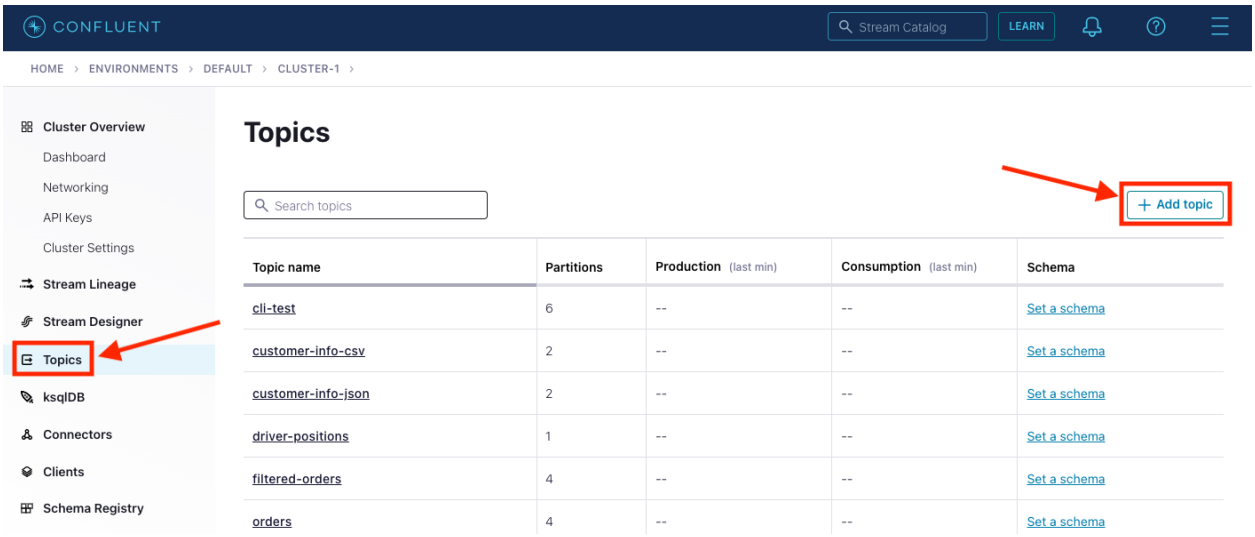
It will take a few minutes for the Dedicated to be available. In the meantime, let's create a topic and set up the API key/secret configurations in the Source cluster.

hitesh@datacouch.io

Create a topic in the Source cluster

4. In Confluent Cloud website, go to the **default** environment.
5. Navigate to the cluster you created in Lab 1, which should be inside the Environment called **default**.
6. In the left pane, click on **Topics**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



7. Click on **Add topic**. Use this configuration. Then, click on **Create with defaults**:
   - Topic Name: **courier-positions**
   - Partitions: **1**
8. Click on **Skip** for the definition of a data contract.

hitesh@datacouch.io

Create API key/secret for the Clients to access the Source cluster

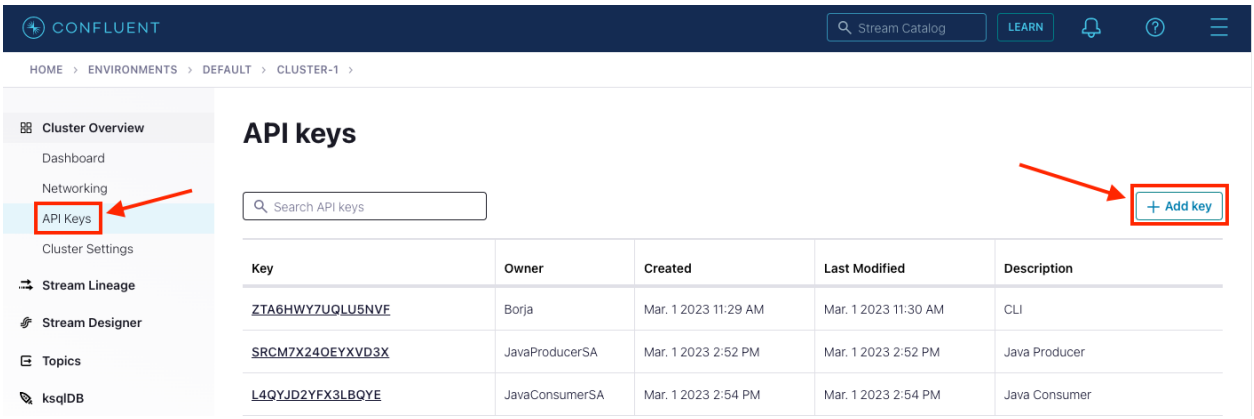
As you can see in the diagram at the beginning of the lab, you are going to use a producer and a consumer. These two clients need an API Key/Secret to access the cluster.

Confluent best practices recommend to use Service Accounts for creating API key/secret that will be applied to applications.

For this lab, you are going to create a single Service Account that will be used for both clients (producer and consumer).

9. Go to the **default** environment.
10. Navigate to the cluster that is inside the **default** environment.
11. In the left pane, select **API Keys** which is under **Cluster Overview**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



12. Click **Add key**. Then, follow these instructions:

    a. Access control: Select **Granular access**, then click **Next**.

    b. Service account: Select **Create a new one**. Then, type name and description:

    - Name: **sa-lab5-clients**
    - Description: **Service Account for the producer and consumer in Lab 5**

    c. Add ACLs to service account: This service account will be used by a producer and a consumer so you will need these **three ACLs**:

    - First ACL:
        i. Resource: `Consumer group`
        ii. Consumer group ID, **Type**: `courier-` , then hit **Enter**
        iii. Pattern type: `PREFIXED`
        iv. Operation: `READ`
        v. Permission: `ALLOW`
    - Second ACL. Click **Add ACLs**:
        i. Resource: `Topic`
        ii. Topic name: `courier-positions`
        iii. Pattern type: `LITERAL`
        iv. Operation: `READ`
        v. Permission: `ALLOW`
    - Third ACL. Click **Add ACLs**:

ritesh@datatouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

i. Resource: `Topic`

ii. Topic name: `courier-positions`

iii. Pattern type: `LITERAL`

iv. Operation: `WRITE`

v. Permission: `ALLOW`

After setting the three ACLs, click **Next**.

d. Type `Lab5 - Clients` in the **Description** and click on **Download and continue** to store the API key/secret locally.

Keep this API key and secret as you will use it in the following section.

Create the java-client.config file for the Source cluster

13. Go to the **VM** and open a Terminal window.

14. Change the working directory to the Cluster Linking folder by running this command:

```
$ cd ~/confluent-ccl/lab-cluster-linking/
```

15. Open Visual Studio Code to modify the file `SOURCE-java-client.config`. Run this command:

```
$ code SOURCE-java-client.config
```

- Replace `{{ SOURCE_CLUSTER_URL }}` with the Cluster Bootstrap Servers Endpoint
- Replace `{{ SOURCE_CLUSTER_API_KEY }}` with the API Key you have just generated
- Replace `{{ SOURCE_CLUSTER_API_SECRET }}` with the API Secret you have just generated

```
bootstrap.servers={{ SOURCE_CLUSTER_URL }}
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLogin
Module   required username='{{ SOURCE_CLUSTER_API_KEY }}'
password='{{ SOURCE_CLUSTER_API_SECRET }}';
```

nitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Press `Ctrl+S` to save the changes.

Sample

```
bootstrap.servers=pkc-6ojv2.us-west4.gcp.confluent.cloud:9092
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLo
ginModule required username='MURRTS4WL'
password='HJeLz5o+d1/Qc5eP0Z9ouhdFzIW24NQWwrU8gDd8mj3sqiGuIhJ';
```

Create API key/secret for the Clients to access the Destination cluster

You need a new API key/secret for the new dedicated cluster **destination-cluster** so that clients can access it. Repeat the same steps you did before but for the Destination cluster.

hitesh@datacouch.io

16. In Confluent Cloud website, go to the cluster **destination-cluster** in the **destination-env** environment.
17. In the left pane, select **API Keys** which is under **Cluster Overview**.
18. Click on **Create key**. Then, follow these instructions:
    a. Access control: Select **Granular access**, then click **Next**.
    b. Service account: Use the existing account **sa-lab5-clients**. Then, click **Next**.
    c. Add ACLs to service account: This service account will be used by a producer and a consumer so you will need these **three ACLs**:
       - First ACL:
         i. Resource: `Consumer group`
         ii. Consumer group ID, **Type**: `courier-` , then hit **Enter**
         iii. Pattern type: `PREFIXED`
         iv. Operation: `READ`
         v. Permission: `ALLOW`

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

- Second ACL. Click **Add ACLs**:
  i. Resource: `Topic`
  ii. Topic name, **Type**: `courier-positions` , then hit **Enter**
  iii. Pattern type: `LITERAL`
  iv. Operation: `READ`
  v. Permission: `ALLOW`
- Third ACL. Click **Add ACLs**:
  i. Resource: `Topic`
  ii. Topic name, **Type**: `courier-positions` , then hit **Enter**
  iii. Pattern type: `LITERAL`
  iv. Operation: `WRITE`
  v. Permission: `ALLOW`

After setting the three ACLs, click **Next**.

d. Type `Lab5 - Clients` in the **Description** and click on **Download and continue** to store the API key/secret locally.

hitesh@datacouch.io

Keep this API key and secret as you will use it in the following section.

Create the java-client.config file for the Destination cluster

19. Go to the **VM** and open a Terminal window.
20. Change the working directory to the Cluster Linking folder by running this command:

```
$ cd ~/confluent-ccl/lab-cluster-linking/
```

21. Open Visual Studio Code to modify the file `DESTINATION-java-client.config`. Run this command:

```
$ code DESTINATION-java-client.config
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

- Replace `{{ DESTINATION_CLUSTER_URL }}` with the Cluster Bootstrap Servers Endpoint
- Replace `{{ DESTINATION_CLUSTER_API_KEY }}` with the API Key you have just generated
- Replace `{{ DESTINATION_CLUSTER_API_SECRET }}` with the API Secret you have just generated

```
bootstrap.servers={{ DESTINATION_CLUSTER_URL }}
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLogin
Module   required username='{{ DESTINATION_CLUSTER_API_KEY }}'
password='{{ DESTINATION_CLUSTER_API_SECRET }}';
```

Press `Ctrl+S` to save the changes.

Sample

```
bootstrap.servers=pkc-6ojv2.us-west4.gcp.confluent.cloud:9092
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLo
ginModule required username='MURRTS4WL'
password='HJeLz5o+d1/Qc5eP0Z9ouhdFzIW24NQWwrU8gDd8mj3sqiGuIhJ';
```

hitesh@datacouch.io

Start the producer and consumer apps

Let's start the producer to write data to the `courier-positions` topic in the Source cluster.

22. In the **Virtual Machine**, run this command in a Terminal window:

```
$ cd ~/confluent-ccl/lab-cluster-linking/java-producer/
```

23. Start the producer by running the following command:

```
$ ./gradlew run --console plain --args="../SOURCE-java-client.config"
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Now, let's start the consumer to read `courier-positions` topic from the Source cluster.

24. Open a new Terminal window and run this command:

```
$ cd ~/confluent-ccl/lab-cluster-linking/java-consumer/
```

25. Start the consumer by running the following command:

```
$ ./gradlew run --console plain --args="../SOURCE-java-client.config"
```

Leave the producer and consumer running.

Create Service Account for Cluster Link

In this section, you are going to create a Service Account using the Confluent CLI.

hitesh@datacouch.io

26. In the **Virtual Machine**, open a new Terminal window and log in to your account using the Confluent CLI, if you are not already:

```
$ confluent login
```

27. Make sure the **default** environment is selected by running the following two commands:

```
$ confluent environment list
  Current |     ID     |      Name
----------+------------+-----------------
  *       | env-d9q617 | default
          | env-knr3v6 | destination-env
```

```
$ confluent environment use {{ ENVIRONMENT_ID }}
```

28. Create a Service Account for your new Cluster Link:

```
$ confluent iam service-account create sa-cluster-link --description
"Service Account for the cluster link in Lab 5"
+-------------+---------------------------------+
| ID          | sa-xx6g61                       |
| Name        | sa-cluster-link                 |
| Description | Service Account for the         |
|             | cluster link in Lab 5           |
+-------------+---------------------------------+
```

29. Store the Service Account ID as environment variable. Run the command replacing `<service-account-ID>` with the ID from the previous output:

```
$ export SA_ID=<service-account-ID>
```

Sample

```
$ export SA_ID=sa-xx6g61
```

30. Before creating the API key and secret to access the Source cluster, check the cluster IDs and store them as environment variables. Run:

```
$ confluent kafka cluster list --all
Current |     ID     |          Name          |   Type    | Provider |
Region    | Availability | Status
--------+------------+------------------------+-----------+----------+
--------------+--------------+-------
        | lkc-1jkq2z | destination-cluster | DEDICATED | gcp      |
europe-west1 | single-zone  | UP
*       | lkc-vrwmq0 | cluster-1           | BASIC     | gcp      |
us-west4     | single-zone  | UP
```

- Run the following command after replacing `<source-cluster-ID>` with your cluster ID from the output of the previous command:

```
$ export SRC_CLUSTER_ID=<source-cluster-ID>
```

Sample

```
$ export SRC_CLUSTER_ID=lkc-vrwmq0
```

hitesh@datatouch.io

- Do the same with the **<destination-cluster-ID>**. Run:

```
$ export DST_CLUSTER_ID=<destination-cluster-ID>
```

Sample

```
$ export DST_CLUSTER_ID=lkc-1jkq2z
```

31. Create the API key and secret to access the Source cluster by running:

```
$ confluent api-key create --resource $SRC_CLUSTER_ID --service
-account $SA_ID
It may take a couple of minutes for the API key to be ready.
Save the API key and secret. The secret is not retrievable later.
+------------
+-----------------------------------------------------------------+
| API Key    | RHSS5DQC74OMSZ5H
|
| API Secret |
OKOGh/hBS8V4NCjzQCojJLLy8+cxqyYWktayO/S1n/wbjP57zZI7i20uTQlUAidj |
+------------
+-----------------------------------------------------------------+
```

nitesh@datapouch.io

32. Let's store this API key and secret in environment variables for future reference:

- Run the following command replacing the **<service-account-source-key>**:

```
$ export SA_SRC_KEY=<service-account-source-key>
```

Sample

```
$ export SA_SRC_KEY=RHSS5DQC74OMSZ5H
```

- Do the same with the secret by replacing **<service-account-source-secret>**. Run:

```
$ export SA_SRC_SECRET=<service-account-source-secret>
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Sample

```
$ export
SA_SRC_SECRET=OKOGh/hBS8V4NCjzQCojJLLy8+cxqyYWktayO/S1n/wbjP57z
ZI7i20uTQlUAidj
```

33. Apply ACLs on the Cluster Link service account to allow READ from the topic **courier-positions**:

```
$ confluent kafka acl create --allow --service-account $SA_ID
--operations READ,DESCRIBE_CONFIGS --topic "courier-positions"
--cluster $SRC_CLUSTER_ID
    Principal    | Permission |    Operation     | Resource Type |
Resource Name   | Pattern Type
-----------------+------------+------------------+---------------+
-------------------+--------------
  User:sa-xx6g61 | ALLOW      | DESCRIBE_CONFIGS | TOPIC         |
courier-positions | LITERAL
  User:sa-xx6g61 | ALLOW      | READ             | TOPIC         |
courier-positions | LITERAL
```

34. Allow the Cluster Link to sync consumer group offsets, by setting the following ACLs:

```
$ confluent kafka acl create --allow --service-account $SA_ID
--operations DESCRIBE --topic "courier-positions" --cluster
$SRC_CLUSTER_ID
    Principal    | Permission | Operation | Resource Type |
Resource Name   | Pattern Type
-----------------+------------+-----------+---------------+
-------------------+--------------
  User:sa-xx6g61 | ALLOW      | DESCRIBE  | TOPIC         | courier-
positions | LITERAL
```

```
$ confluent kafka acl create --allow --service-account $SA_ID
--operations READ,DESCRIBE --consumer-group "courier-" --prefix
--cluster $SRC_CLUSTER_ID
    Principal    | Permission | Operation | Resource Type | Resource
Name | Pattern Type
-----------------+------------+-----------+---------------+
---------------+--------------
  User:sa-xx6g61 | ALLOW      | DESCRIBE  | GROUP         | courier-
| PREFIXED
  User:sa-xx6g61 | ALLOW      | READ      | GROUP         | courier-
| PREFIXED
```

hitesh@datacouch.io

Create the Cluster Link

35. In order to create the Cluster Link, you need to know the Source cluster Bootstrap Server. Run the following command to find out and store it in an environment variable:

```
$ confluent kafka cluster describe $SRC_CLUSTER_ID
+----------------------
+-----------------------------------------------------+
| Current                | true
|
| ID                     | lkc-vrwmq0
|
| Name                   | cluster-1
|
| Type                   | BASIC
|
| Ingress Limit (MB/s)  |
250 |
| Egress Limit (MB/s)   |
750 |
| Storage                | 5 TB
|
| Provider               | gcp
|
| Region                 | us-west4
|
| Availability           | single-zone
|
| Status                 | UP
|
| Endpoint               | SASL_SSL://pkc-6ojv2.us-
west4.gcp.confluent.cloud:9092 |
| REST Endpoint          | https://pkc-6ojv2.us-
west4.gcp.confluent.cloud:443     |
| Topic Count            |
13 |
+----------------------
+-----------------------------------------------------+
```

htesh@datacouch.io

- Replace `<source-bootstrap-server>` with the highlighted part and run the command:

```
$ export SRC_BOOTSTRAP_SERVER=<source-bootstrap-server>
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Sample

```
$ export SRC_BOOTSTRAP_SERVER=pkc-6ojv2.us-
west4.gcp.confluent.cloud:9092
```

36. Remember that the Cluster Link itself will reside on the Destination cluster. So, your Confluent CLI should be selecting the Environment that contains the Destination cluster in order to create the Link. Run the following commands to change the **`Current`** cluster:

    - List all your environments:

```
$ confluent environment list
  Current |     ID      |      Name
----------+-------------+-------------------
          | env-09jo06  | destination-env
  *       | env-vr1qvn  | default
```

    - Use the environment that contains the Destination cluster replacing the **`<environment-id>`**:

```
$ confluent environment use <environment-id>
```

Sample

```
$ confluent environment use env-09jo06
```

37. Create a Cluster Link in the Destination cluster to mirror data from the Source cluster. Run:

hitesh@datacouch.io

```
$ confluent kafka link create lab5-cluster-link \
    --cluster $DST_CLUSTER_ID \
    --source-cluster $SRC_CLUSTER_ID \
    --source-bootstrap-server $SRC_BOOTSTRAP_SERVER \
    --source-api-key $SA_SRC_KEY \
    --source-api-secret $SA_SRC_SECRET \
    --config-file ~/confluent-ccl/lab-cluster-linking/cluster-
link.config

Created cluster link "lab5-cluster-link" with configs:
"bootstrap.servers"="pkc-zpjg0.eu-central-1.aws.confluent.cloud:9092"
"consumer.offset.group.filters"="{"groupFilters": [{"name": "courier-
","patternType": "PREFIXED","filterType": "INCLUDE"}]}"
"consumer.offset.sync.enable"="true"
"consumer.offset.sync.ms"="5000"
"link.mode"="DESTINATION"
"sasl.mechanism"="PLAIN"
"security.protocol"="SASL_SSL"
```

Note that the last line of the previous command provides a config file. This file contains the required properties to configure the Cluster Link to mirror the consumer offsets for consumer group names starting by **courier-**.

```
consumer.offset.sync.enable=true
consumer.offset.group.filters={"groupFilters": [{"name":
"courier-","patternType": "PREFIXED","filterType":
"INCLUDE"}]}
```

hitesh@datacouch.io

38. List the full configuration of the **lab5-cluster-link** by running this command:

```
$ confluent kafka link configuration list lab5-cluster-link --cluster
$DST_CLUSTER_ID
```

Note that the property **consumer.offset.sync.enable** is set to **true**.

Mirror the *courier-positions* topic

Now that you have a cluster link, you can mirror topics across it; from source to destination.

39. Create the mirror topic `courier-positions`.

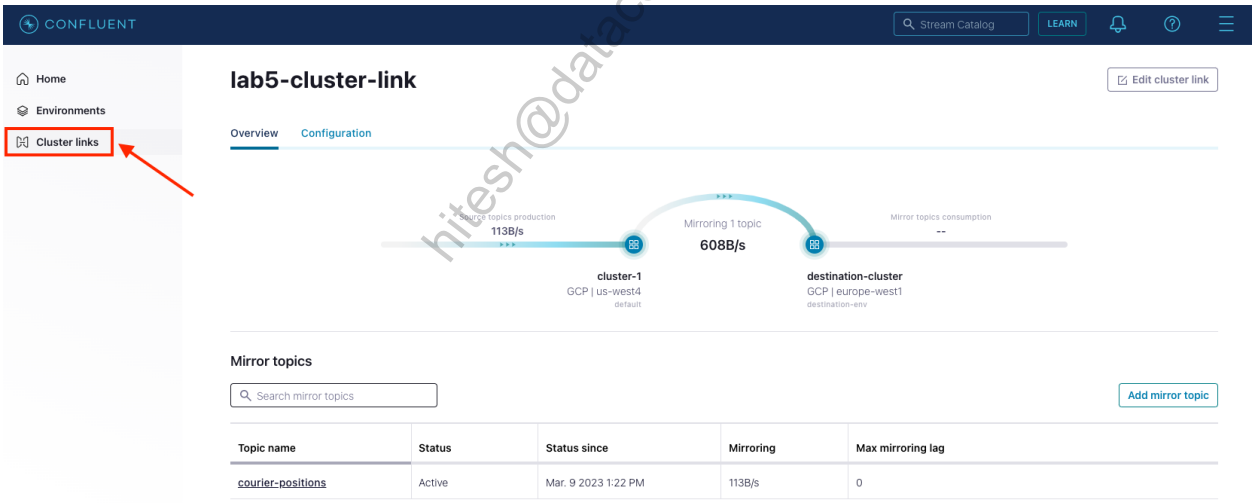
```
$ confluent kafka mirror create courier-positions --link lab5-
cluster-link --cluster $DST_CLUSTER_ID
Created mirror topic "courier-positions".
```

40. Run the following command to list all the mirror topics in your Destination cluster:

```
$ confluent kafka mirror list --cluster $DST_CLUSTER_ID
      Link Name      | Mirror Topic Name | Source Topic Name | Mirror
Status | Status Time (ms) | Num...
-------------------+-------------------+-------------------+
---------------+------------------+----...
  lab5-cluster-link | courier-positions | courier-positions | ACTIVE
|    1678364531232 |    ...
```

41. You can also check the status and configuration of your Cluster Link directly in Confluent Cloud UI.

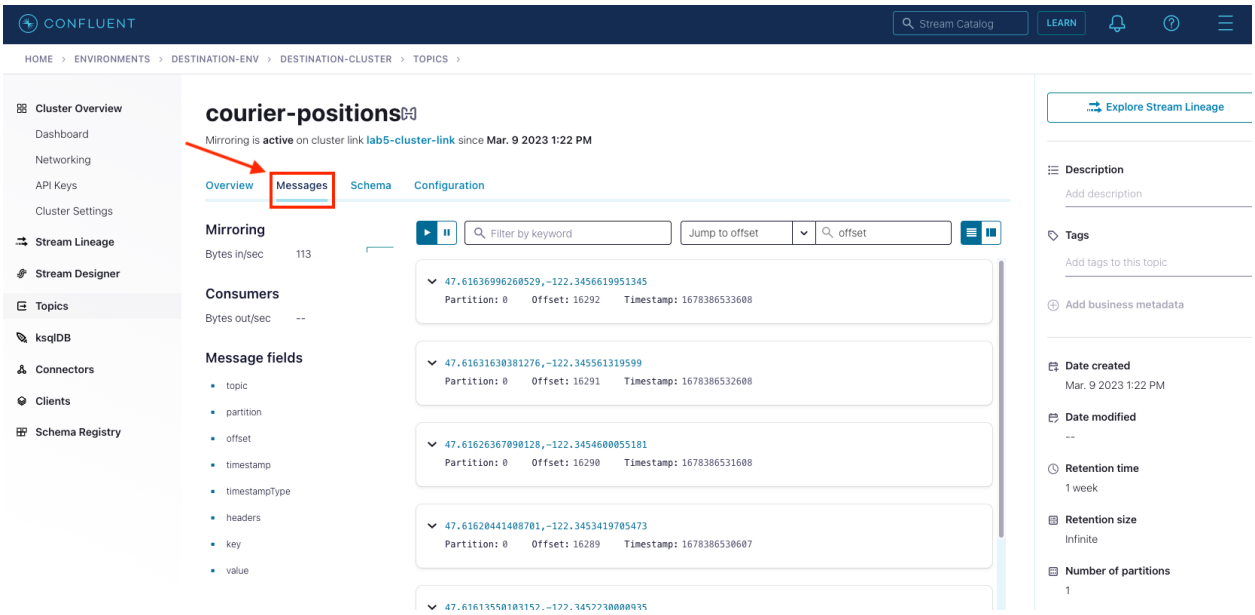
    Go to **Cluster links** and click on **lab5-cluster-link**:



hitesh@datacouch.io

42. Check the data is being synced by looking into the mirror topic. Click on **courier-positions** and then click on the Messages tab:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



Migrate the consumer from Source to Destination cluster

nitesh@datacouch.io

43. Go to the **VM** and take a look at the Terminal window where the consumer is running. It should look like that:

```
...
Consumed Key:courier-1 Value:47.60757821681359,-122.3373687693654  --
Partition:0 Offset:646
Consumed Key:courier-1 Value:47.60756356070046,-122.3373552333628  --
Partition:0 Offset:647
Consumed Key:courier-1 Value:47.60756724789032,-122.3373586988791  --
Partition:0 Offset:648
Consumed Key:courier-1 Value:47.60755857124403,-122.3373506694099  --
Partition:0 Offset:649
Consumed Key:courier-1 Value:47.60755539743309,-122.3373479325632  --
Partition:0 Offset:650
```

44. Press `Ctrl+C` to stop the consumer.

45. Take a note of the last message consumed and its offset `lastOffset`.

```
Consumed Key:courier-1 Value:47.61338524621169,-122.3405219311683  --
Partition:0 Offset:813
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

After migrating the consumer to read from the Destination cluster, the consumer should start consuming the message `lastOffset + 1`, as Cluster Link was also configured to mirror the offsets of the consumer groups prefixed by `courier-`.

46. Before restarting the consumer in the Destination cluster, you need to update the Cluster Link configuration to stop mirroring the consumer offsets:
    - Open the `cluster-link.config` file:

```
$ code ~/confluent-ccl/lab-cluster-linking/cluster-link.config
```

    - Change the property `consumer.offset.sync.enable` to `false`.
    - Press `Ctrl+S` to save the file.
    - Run the following command to update the Cluster Link configuration:

```
$ confluent kafka link configuration update lab5-cluster-link \
          --cluster $DST_CLUSTER_ID \
          --config-file ~/confluent-ccl/lab-cluster-
linking/cluster-link.config

Updated cluster link "lab5-cluster-link".
```

hlesh@gmacouch.io

Make sure you run the previous command in the Terminal window where the environment variables are stored. Do not run it in the consumer Terminal window.

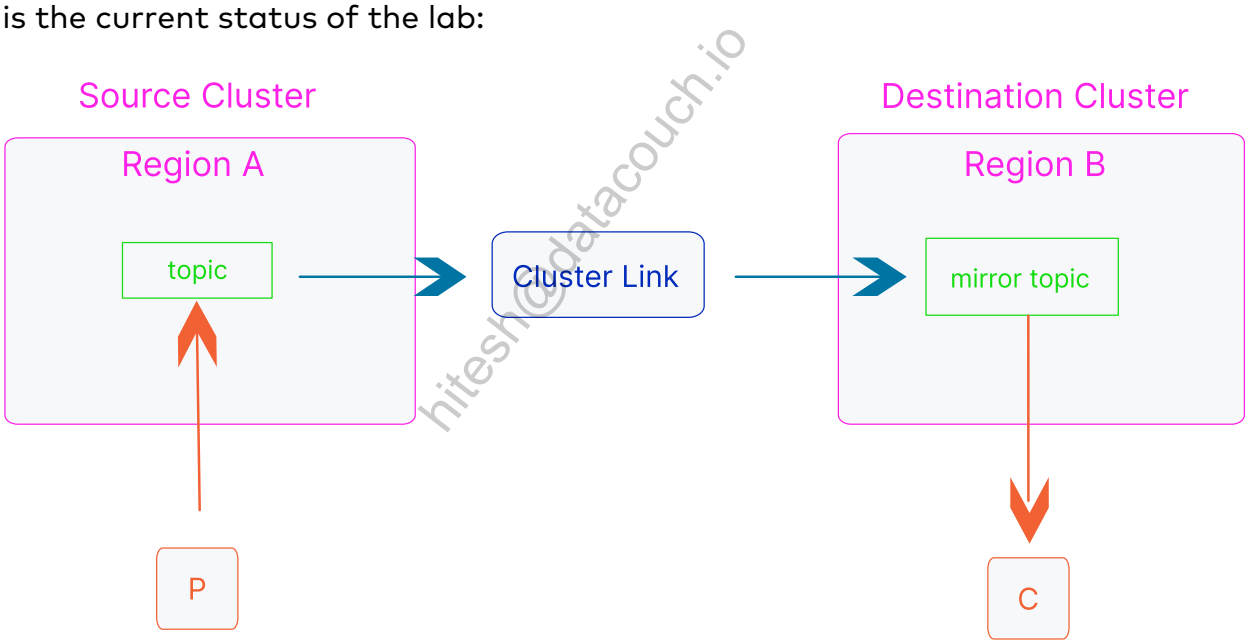
    - You can double check the property was update by listing the Cluster Link configuration:

```
$ confluent kafka link configuration list lab5-cluster-link
--cluster $DST_CLUSTER_ID
```

47. Go back to the consumer Terminal window and restart the consumer to read from the Destination cluster. Run this command:

```
$ ./gradlew run --console plain --args="../DESTINATION-java
-client.config"

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE

> Task :run
Starting Java Consumer.
Consumed Key:courier-1 Value:47.61860447161879,-122.3553537350844  --
Partition:0 Offset:814
Consumed Key:courier-1 Value:47.61865452444275,-122.3554362772165  --
Partition:0 Offset:815
Consumed Key:courier-1 Value:47.61871524477854,-122.3554968416097  --
Partition:0 Offset:816
```

48. Check that the offset of the first message consumed corresponds to `lastOffset + 1`. See how easy is migrating consumers from one cluster to another using Cluster Linking. This is the current status of the lab:



hitesh@datacouch.io

Migrate the producer from Source to Destination cluster

49. Open the Terminal window where the producer is running.
50. To perform the migration, first, you need to stop the producer. Press `Ctrl+C` to stop it.
51. Now, the mirror topic needs to be promoted, since producers cannot write to mirror topics. Run this command to convert the mirror topic into a normal topic:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ confluent kafka mirror promote courier-positions --link lab5-
cluster-link --cluster $DST_CLUSTER_ID

  Mirror Topic Name | Partition | Partition Mirror Lag | Last Source
Fetch Offset | Error Message ...
--------------------+-----------+----------------------+
-------------------------+--------------...
  courier-positions |         0 |                    0 |
1132 |                ...
```

As long as the source cluster is online, use the **promote** command as it will ensure there is no mirroring lag, config sync lag, or consumer offset lag before conversion. The **failover** command will succeed regardless of the mirroring lag or the source cluster's reachability.

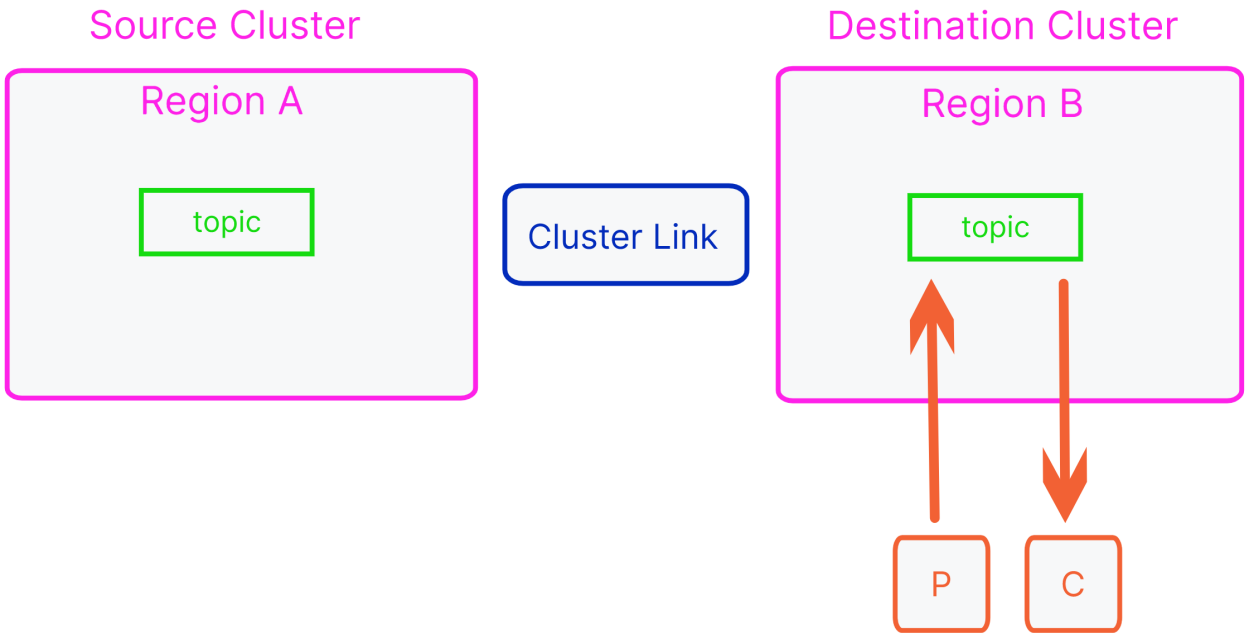
52. Go back to the producer Terminal window and restart the producer pointing the Destination cluster. Run this command:

```
$ ./gradlew run --console plain --args="../DESTINATION-java
-client.config"

> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE

> Task :run
Starting Java producer.
Produced Key:courier-1 Value:47.61555078539428,-122.350074928362  --
Partition:0 Offset:1133
Produced Key:courier-1 Value:47.6156011233706,-122.3501682806703  --
Partition:0 Offset:1134
Produced Key:courier-1 Value:47.61565226418686,-122.350239823045  --
Partition:0 Offset:1135
```

hitesh@datatouch.io

53. This is the current status of the lab:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



Delete the Cluster Link

Now that we have finished the migration, we can safely delete the Cluster Link.

54. Delete the Cluster Link providing its name:

```
$ confluent kafka link delete lab5-cluster-link --cluster
$DST_CLUSTER_ID
Are you sure you want to delete cluster link "lab5-cluster-link"?
To confirm, type "lab5-cluster-link". To cancel, press Ctrl-C: lab5-
cluster-link
Deleted cluster link "lab5-cluster-link".
```

ilesh@datacouch.io

Remember! To delete a Cluster Link, it requires to not have mirror topics on it.

Clean up

55. First, stop the producer and consumer by pressing `Ctrl+C` in both Terminal windows.

56. Delete the Dedicated cluster running this command:

```
$ confluent kafka cluster delete $DST_CLUSTER_ID
Are you sure you want to delete Kafka cluster "lkc-1jkq2z"?
To confirm, type "destination-cluster". To cancel, press Ctrl-C:
destination-cluster
Deleted Kafka cluster "lkc-1jkq2z".
```

Conclusion

In this lab you have learned how to use Cluster Linking following best practices. You have created a Service Account for your Cluster Link, applied the required ACLs and generated API key/secret for the Source cluster.

You have learned how to create and configure the Link to mirror one specific topic. Then, you have migrated your clients from the Source to the Destination cluster and converted the mirror topic into a normal topic. Finally, you have deleted the Cluster Link.

STOP HERE. THIS IS THE END OF THE EXERCISE.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Lab 06 Connect

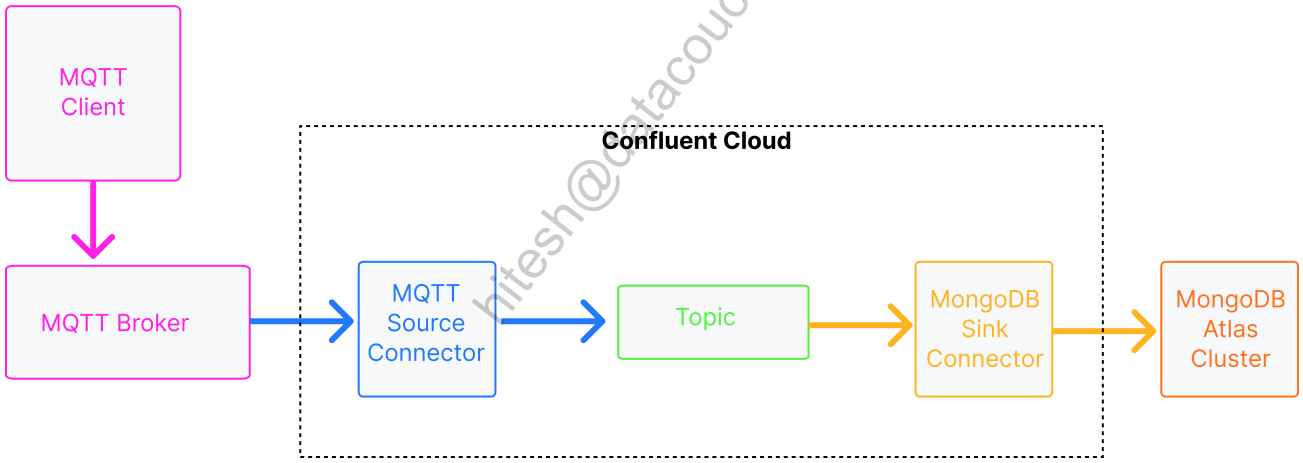
a. Using Connectors with Confluent Cloud

The goal of this lab is to get familiar with connectors in Confluent Cloud. You will create two connectors, use egress IP addresses and experience with Dead Letter Queue topics.

Connect Pipeline

In this section, you will run two fully-managed Connectors in Confluent Cloud: a MQTT source Connector and a MongoDB Atlas sink Connector. The MQTT source Connector writes the events from a topic in a free public MQTT broker to a Confluent Cloud Topic. The MongoDB Atlas sink Connector reads data from the same Topic and writes those messages to a Collection in your MongoDB Atlas cluster.

hitesh@datacouch.io



Create a new Basic cluster

For this lab you need a Basic cluster in one of the following regions:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

AWS

```
Oregon (us-west-2)
N. Virginia (us-east-1)
Frankfurt (eu-central-1)
Ireland (eu-west-1)
Paris (eu-west-3)
Stockholm (eu-north-1)
São Paulo (sa-east-1)
Hong Kong (ap-east-1)
Mumbai (ap-south-1)
Tokyo (ap-northeast-1)
Seoul (ap-northeast-2)
Singapore (ap-southeast-1)
Sydney (ap-southeast-2)
Cape Town (af-south-1)
```

Google Cloud

```
Iowa (us-central1)
Belgium (europe-west1)
São Paulo (southamerica-east1)
Mumbai (asia-south1)
Singapore (asia-southeast1)
Jakarta (asia-southeast2)
Taiwan (asia-east1)
Tokyo (asia-northeast1)
Seoul (asia-northeast3)
```

hitesh@datacouch.io

Azure

```
Virginia (eastus2)
Ireland (northeurope)
Netherlands (westeurope)
Toronto (canadacentral)
Hong Kong (eastasia)
Pune (centralindia)
```

If you don't have a Basic cluster in any of those regions, create a new one.

1. Go to Confluent Cloud website and log in to your Confluent Cloud account.
2. Navigate to **Environments**. Then, click on **default** environment.

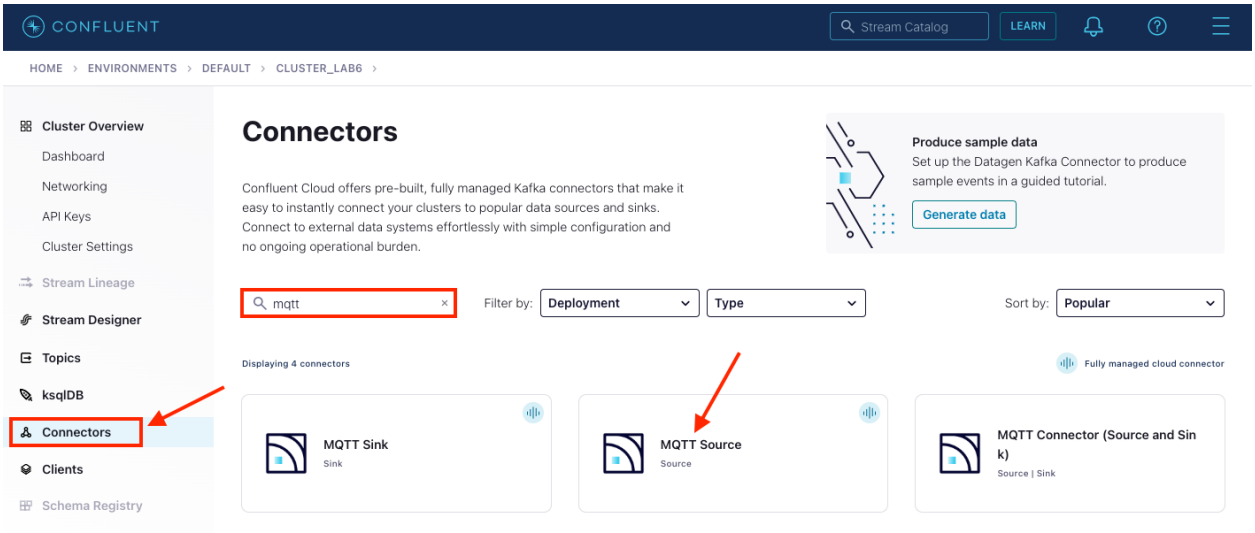
© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

3. Click on **Add cluster**. Then, follow these instructions:
    a. Cluster type: Click on **Begin Configuration** under **Basic**.
    b. Region/zones: Choose your preferred cloud provider and region from those specified above. Select **Single zone**. Click on **Continue**.
    c. Set payment: Click on **Skip payment**.
    d. Review and launch: Set the cluster name to **cluster_lab6**. Review and click on **Launch cluster**.

Configure a MQTT Source connector

You are going to connect a free public MQTT broker to Confluent Cloud.

4. Check the website of the MQTT HiveMQ Broker.
5. See under the **Public MQTT Broker** section that it can be accessed through:
    - Broker: **broker.hivemq.com**
    - TCP Port: **1883**
6. Now, come back to your Confluent Cloud account and go to your cluster.
7. In the left sidebar, select **Connectors**.
8. Type `mqtt` in the Search box and then select the **MQTT Source** connector.



9. Click on **Add a new topic**:

hitesh@datacouch.io

- Topic name: **mqtt-topic**
- Partitions: **1**

10. Click on **Create with defaults** to finish the creation of the topic.
11. Select the topic **mqtt_topic** and click **Continue**.
12. In Kafka Credentials, select **Granular access** and choose **Create a new one**:
    - New service account name: **sa-mqtt-connector**
    - Description: **Service Account for the MQTT connector of Lab6**
    - **Check the box** for "Adding all required ACLs" and click on **Continue**
13. In Authentication, use the following settings and then click on **Continue**:
    - List of Server URIs: **tcp://broker.hivemq.com:1883**
    - MQTT Topics: **testtopic/lab6**
14. In Configuration, just click on **Continue**.
15. In Sizing, leave the number of tasks to 1 and click on **Continue**.
16. In Review and launch, set the name of the connector to **MqttSourceConnector_Lab6** and click on **Continue**.

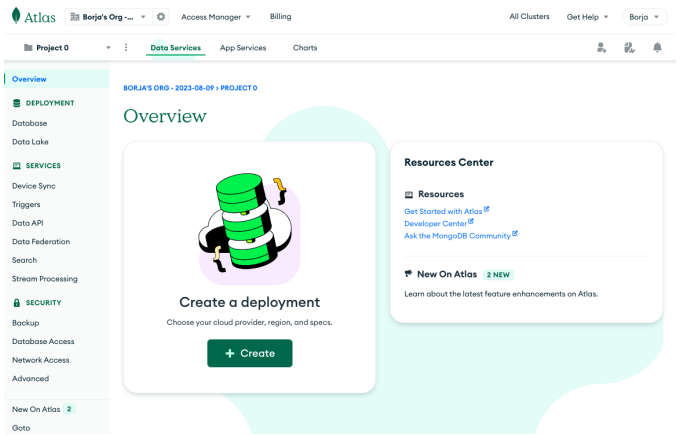
nitesh@datacouch.io

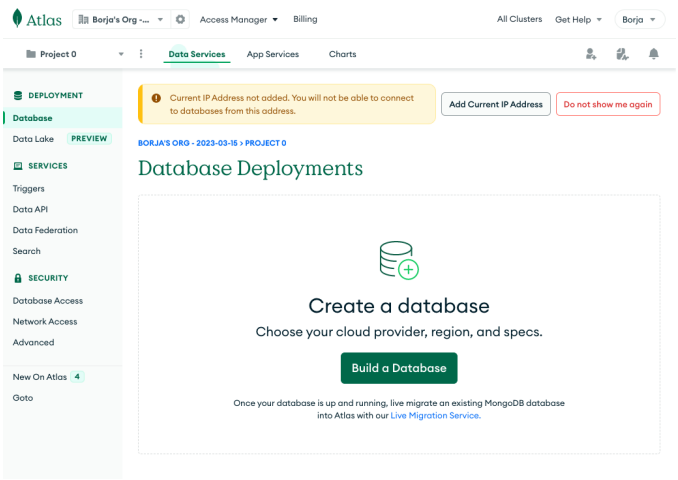
Prepare your MongoDB Atlas cluster

To complete this Lab, you need an account in MongoDB Atlas.

17. If you don't have an account, sign up to create one.
18. Once you are logged in, create a new cluster by clicking on **Create a Deployment** or **Build a Database** depending on whether or not you're new to MongoDB Atlas.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.





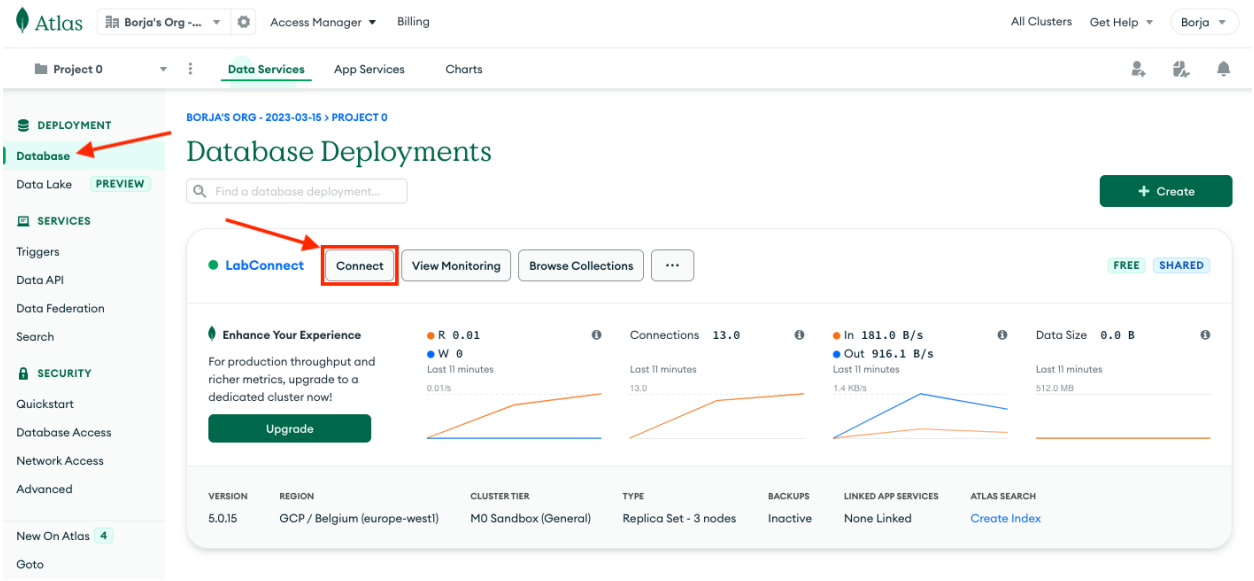
19. In the "Deploy your database" window:
    - Choose the FREE cluster **M0**.
    - Select **the same provider and region** as the Confluent Cloud Basic cluster you are using from the beginning of this Lab.
    - Name your cluster: **LabConnect**
    - Click on **Create**.
20. In the "Security Quickstart" window:
    - Choose the **Username and Password** authentication.
    - Set your Username and Password, then click on **Create User**.

hitesh@datacouch.io

> Keep your **username** and **password**, as you will need them to configure the Connector later.

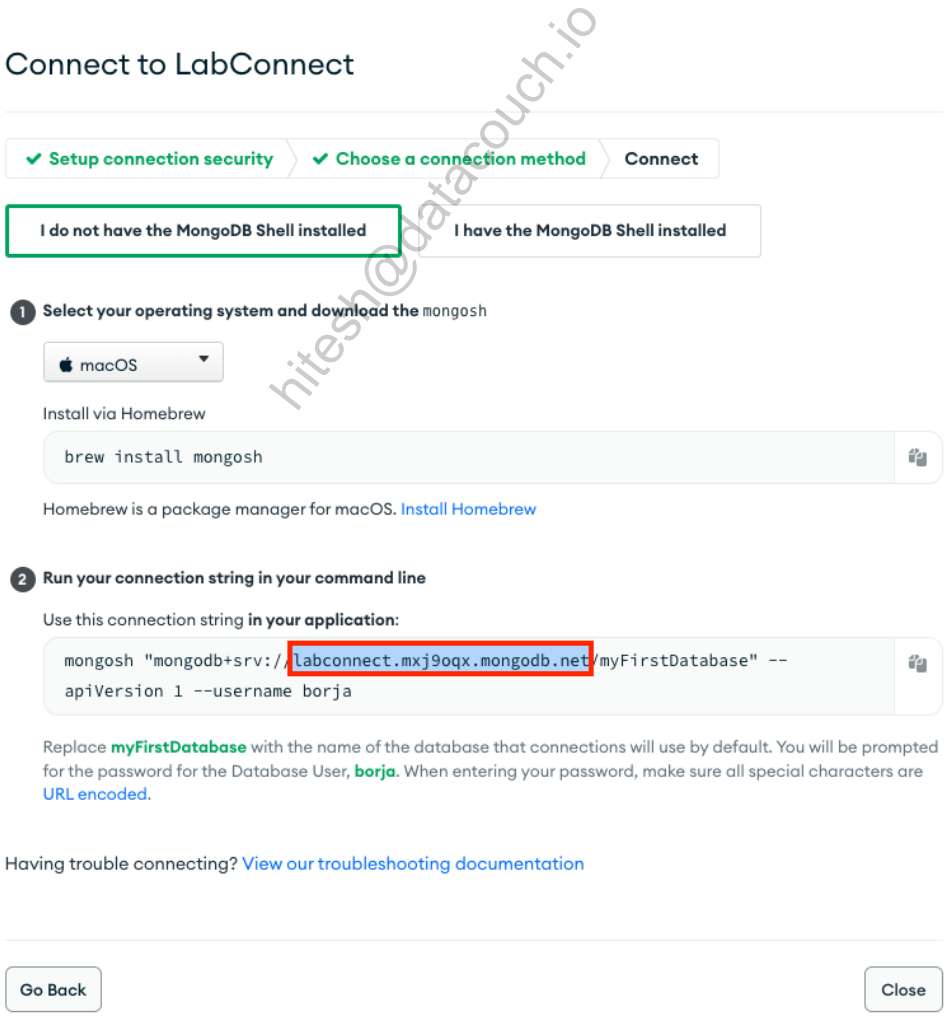
    - Select **My Local Environment** to connect from.
    - Click on **Finish and Close**.
21. Go to **Database Deployments** and then click on **Connect**:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



22. Under "Access your data through tools", click on **Shell**.

23. Check the **Connection Host**, which is the highlighted part in the following screenshot:

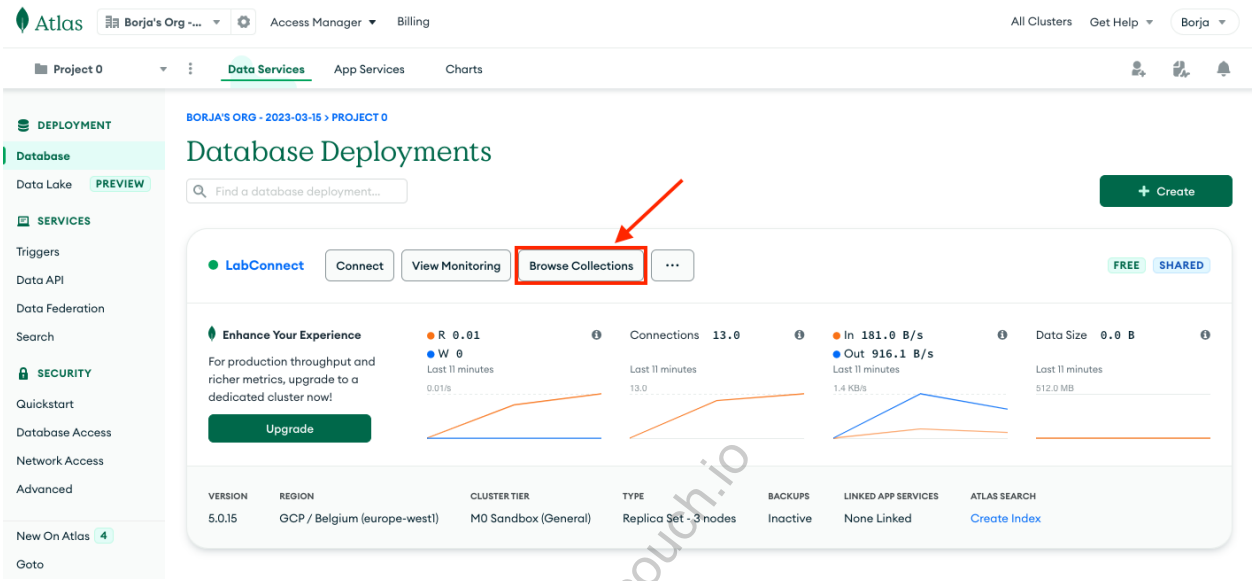


hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Copy this **Connection Host** into a Notepad, as you will need it later to configure the Connector.

24. Click on **Close** and then click on **Browse Collections**.



25. Select **Add My Own Data** and then use the following names:
    - Database name: **ConfluentCloud**
    - Collection name: **mqttTopicData**
26. Finally, click on **Create**.

hitesh@datacouch.io

Configure Static Egress IP Addresses for Confluent Cloud Connectors

In Confluent Cloud clusters, static egress IP addresses are public IP addresses associated with your Confluent Cloud cluster that are used to communicate with external resources (such as MongoDB Atlas) over the public internet and include the following features and options:

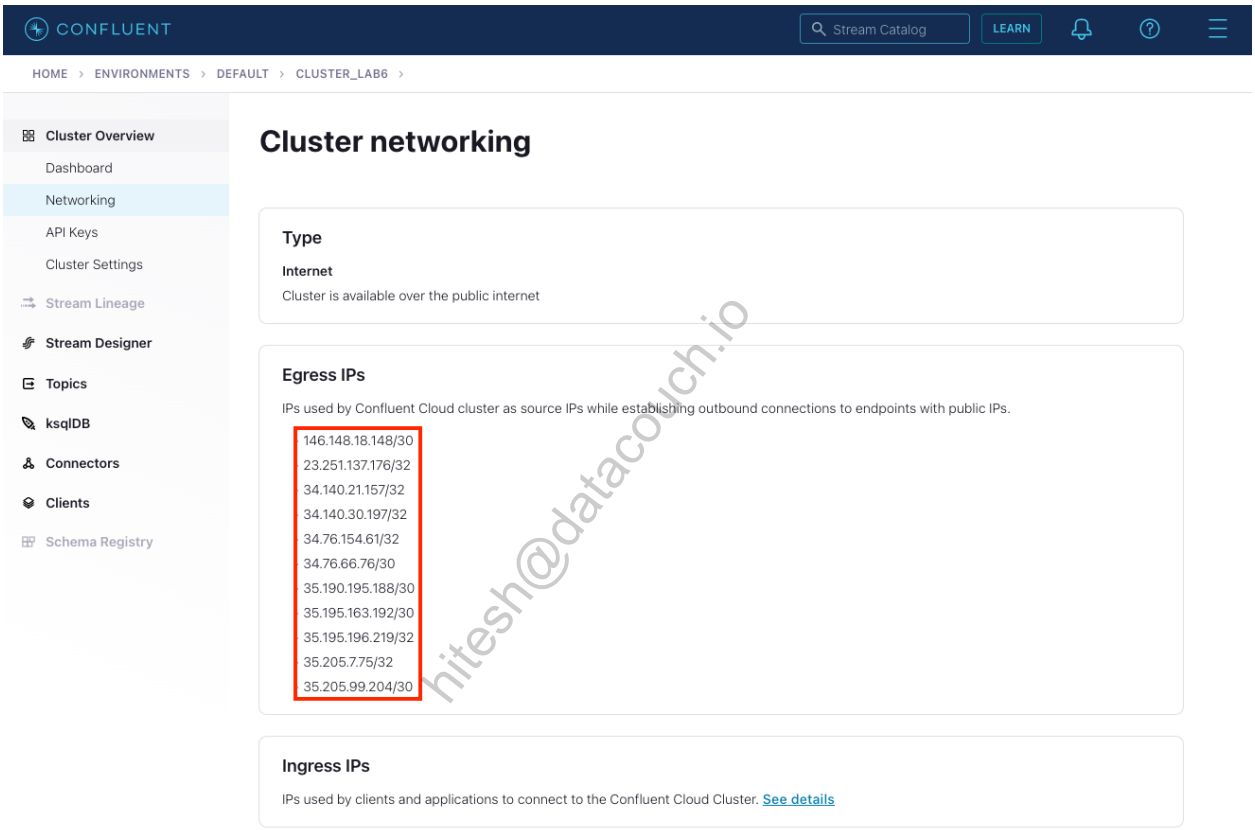
- Never change — stable throughout the lifecycle of the Confluent Cloud cluster and managed connectors.
- Can be used to restrict access to authorized clients on an external resource.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

- Static egress IP addresses are shared across Confluent Cloud cluster types in the same cloud service provider region.

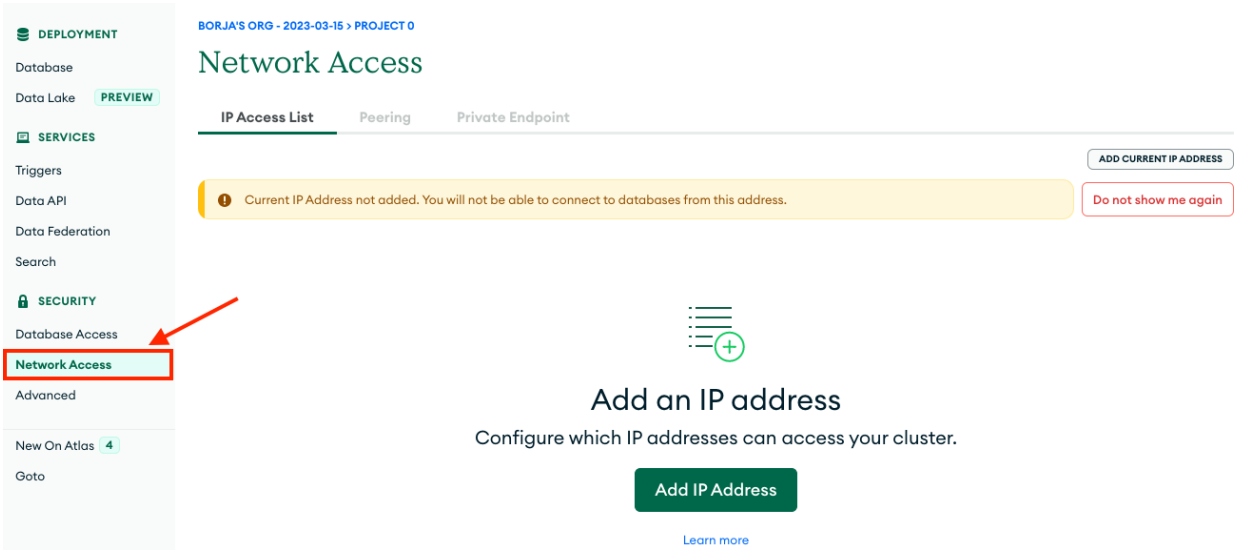
For the connector to work correctly, you need to add all Confluent Cloud egress IP addresses to the Access List of the MongoDB Atlas cluster to allow connections from Confluent Cloud.

27. Go to your Confluent Cloud cluster.
28. In the left sidebar, click on **Networking**. Here, you will find all the egress IP addresses.



29. Now, go to your MongoDB Atlas cluster and in the left sidebar click on **Network Access**:

hitesh@datacouch.io

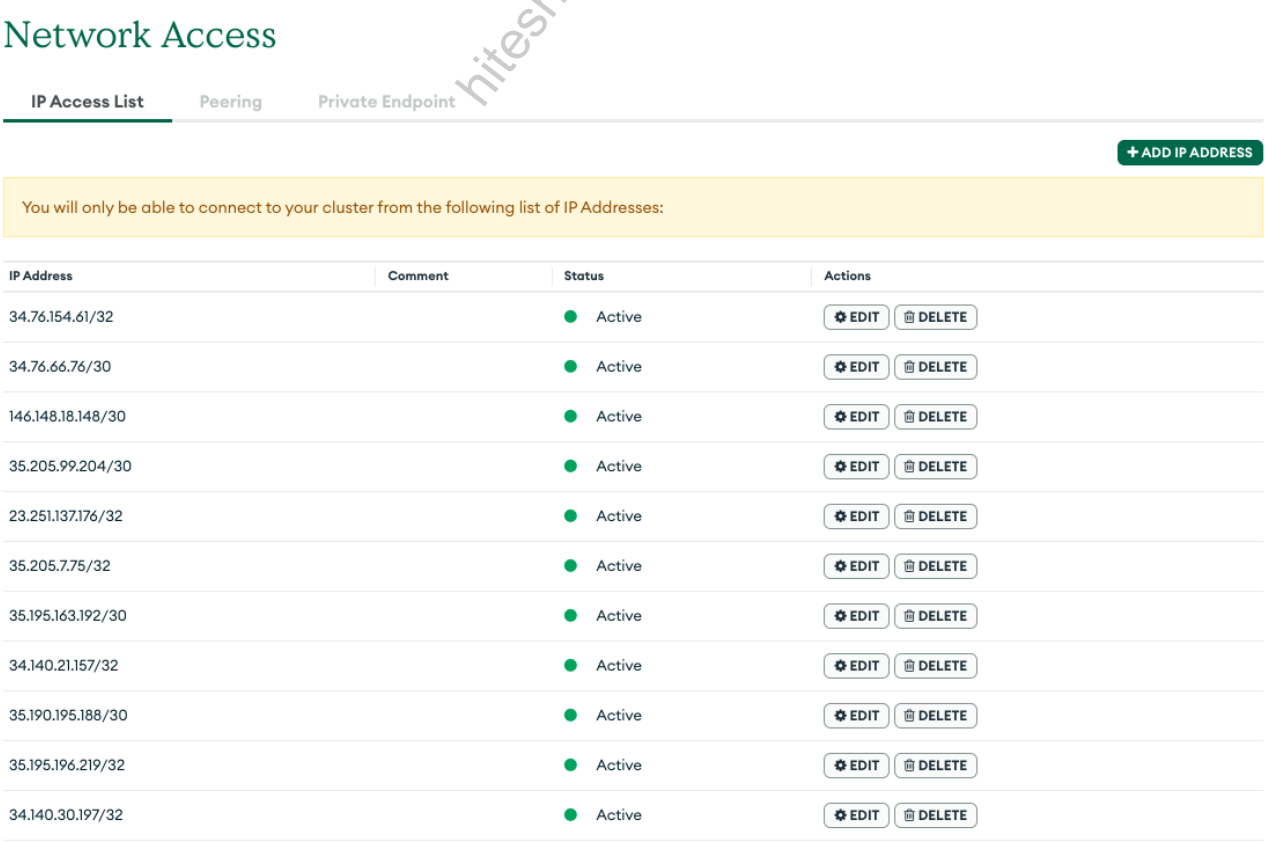


30. Click on **Add IP Address**, copy the first **Egress IP** from Confluent Cloud and paste it in **Access List Entry**. Then, click on **Confirm**.

31. Repeat this process with the rest of **Egress IP addresses** from Confluent Cloud.

If you don't want to add all the egress IP addresses, then allow access from anywhere using the CIDR notation `0.0.0.0/0`. This is not recommended for production environments.

hitesh@datacouch.io

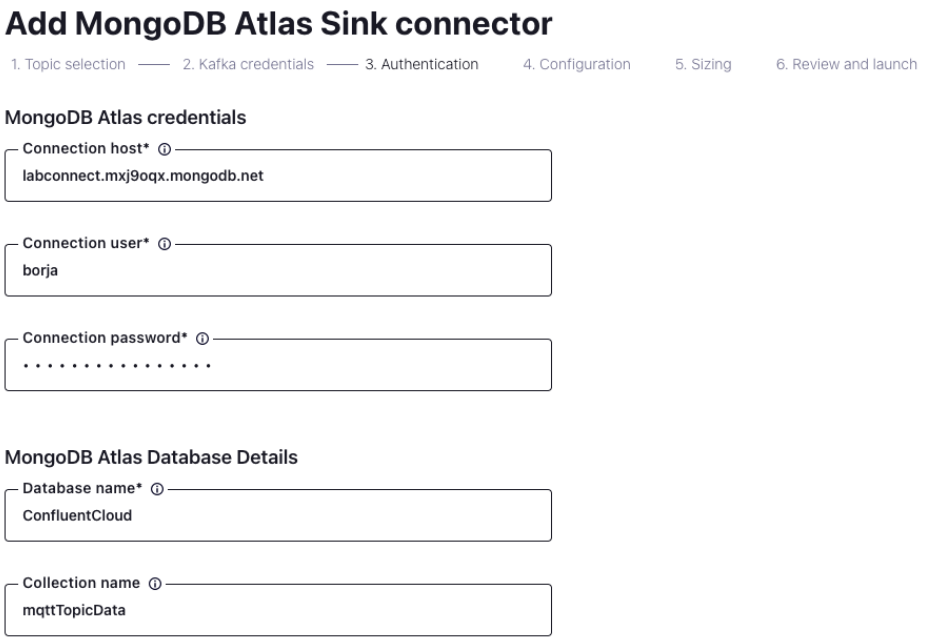


© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Configure the MongoDB Atlas Sink Connector

32. Go back to your cluster in Confluent Cloud.
33. In the left sidebar, select **Connectors**.
34. Click on **Add Connector**.
35. Type `mongodb` in the Search box and then select the **MongoDB Atlas Sink** connector.
36. Select the topic **mqtt_topic** and click **Continue**.
37. In Kafka Credentials, select **Granular access** and choose **Create a new one**:
    - New service account name: **sa-mongodb-connector**
    - Description: **Service Account for the MongoDB Atlas Sink connector of Lab6**
    - **Check the box** for "Adding all required ACLs" and click on **Continue**
38. In Authentication, use the following settings and then click on **Continue**:
    - Connection host: **Use the host you copied in a notepad in step 23**
    - Connection user: **Use the username you created in step 20**
    - Connection password: **Use the password you created in step 20**
    - Database name: **ConfluentCloud**
    - Collection name: **mqttTopicData**

nitesh@datacouchio



© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

39. In Configuration, select **JSON** as the input Kafka record value format and click on **Continue**.

By selecting **JSON**, you are actually configuring the deserializer that the sink connector will use to deserialize the messages consumed from Kafka (**JsonDeserializer**).

40. In Sizing, leave the number of tasks to 1 and click on **Continue**.
41. In Review and launch, set the name of the connector to **MongoDbAtlasSinkConnector_Lab6** and click on **Continue**.
42. After a few seconds, you will see that a new topic has been automatically created with a name that starts with `dlq-lcc-`, which stands for `dead letter queue - logical connect cluster -`. This topic will receive the messages that cause a deserialization error in the MongoDB Atlas Sink connector.

hitesh@datatouch.io

Start Writing Data to the MQTT Broker

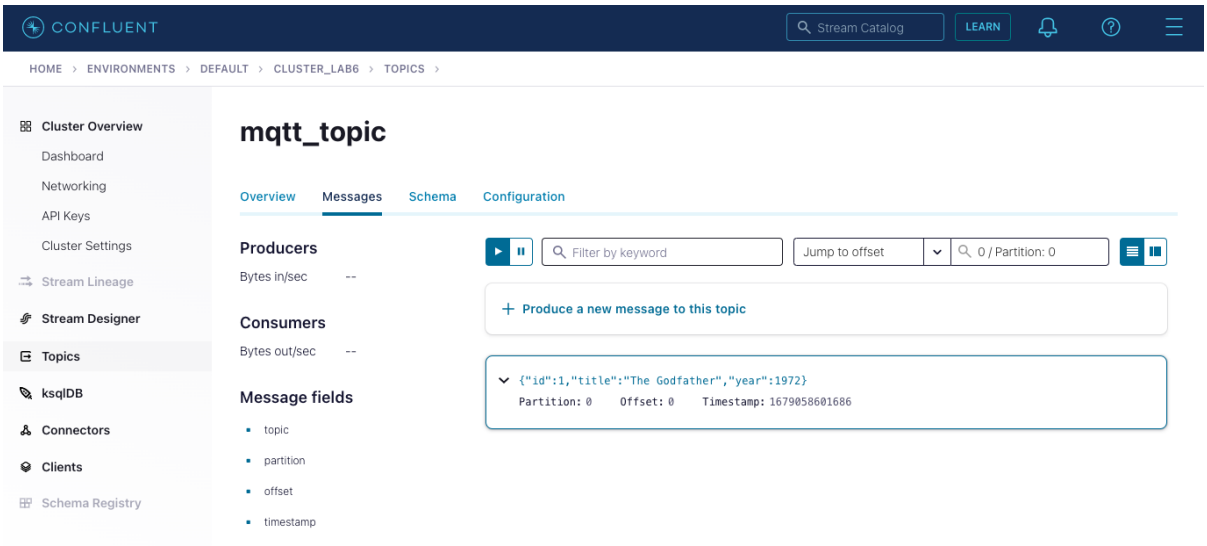
Let's start producing some data to the MQTT broker.

43. Go the HiveMQ websocket client.
44. Click on **Connect** to establish the connection with the HiveMQ Broker.
45. In the **Publish** section, change the topic name to: **testtopic/lab6**
46. Copy the following JSON payload, paste it in the **Message** box and click **Publish**:

```
{"id":1,"title":"The Godfather","year":1972}
```

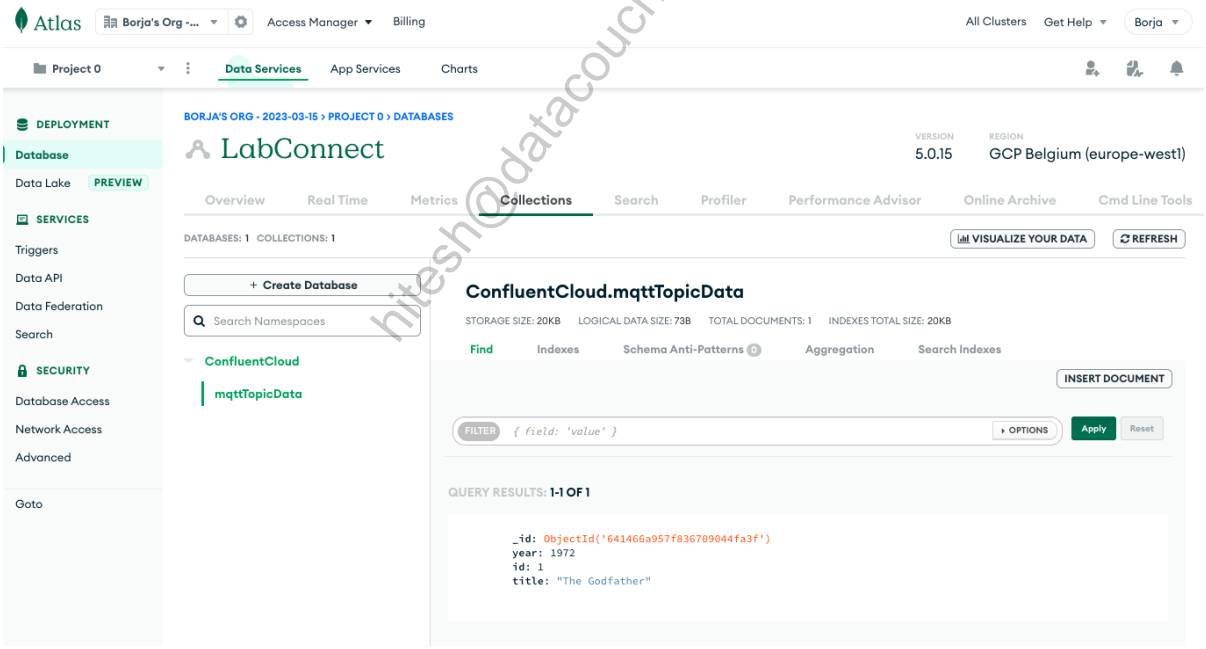
47. Verify that the JSON payload was sent to Confluent Cloud:
    - Go to Confluent Cloud → **<your-cluster>** → **Topics** → **mqtt_topic** → **Messages**
    - In the **offset** box, type **0** to view messages from offset 0 and then, select `0 / Partition: 0`
    - You should see the message that you just produced to the MQTT broker

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



48. Verify that the JSON payload was sent to MongoDB Atlas:

    - Go to MongoDB Atlas → **LabConnect** cluster → **Collections**
    - You should see the document with the data you just produced to the MQTT broker



hitesh@datacouch.io

49. Produce more messages using the HiveMQ websocket client and verify that are received in Confluent Cloud and MongoDB Atlas:

```
{"id":2,"title":"Pulp Fiction","year":1994}
```

```
{"id":3,"title":"Star Wars","year":1977}
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
{"id":4,"title":"Die Hard","year":1988}
```

```
{"id":5,"title":"Lost in Translation","year":2003}
```

```
{"id":6,"title":"Jaws","year":1975}
```

Dead Letter Queues in Confluent Cloud

For Connect, errors that may occur are typically serialization and deserialization (serde) errors. In Confluent Cloud, the connector does not stop when serde errors occur. Instead, the connector continues processing records and sends the errors to a Dead Letter Queue (DLQ).

Example Deserialization Error

```
A record arrives at the sink connector in JSON format, but the sink
connector configuration is expecting another format, like Avro.
```

ritesh@datacouch.io

You can use the record headers in a DLQ topic record to identify and troubleshoot an error when it occurs.

50. In Confluent Cloud, go to **Topics** inside your cluster.
51. Click on the topic that starts with **`dlq-lcc-`** and go to the tab **Messages**.

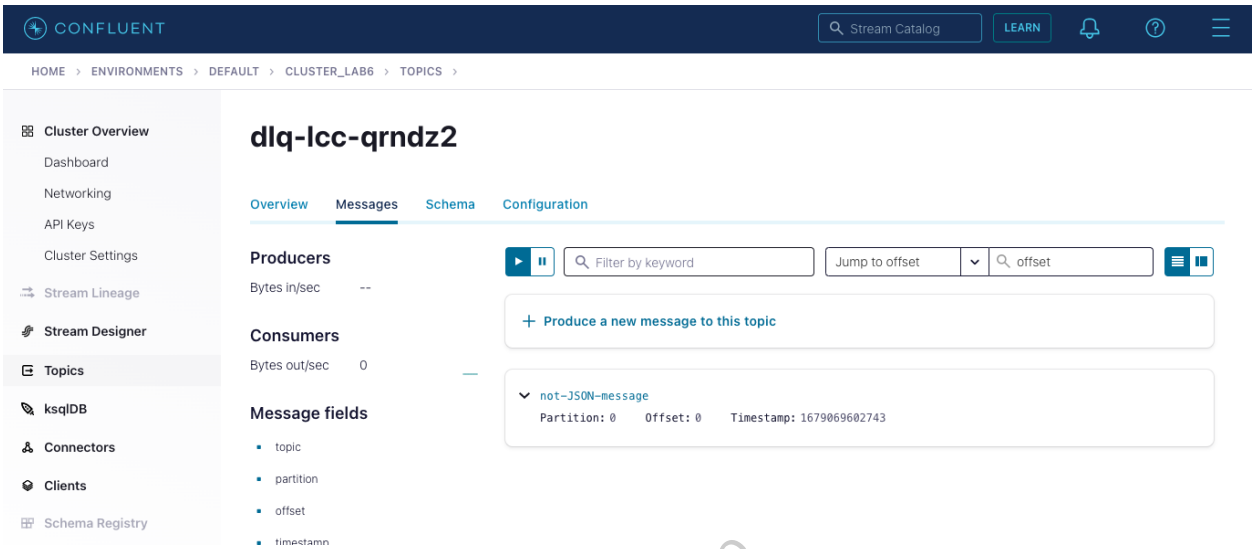
Leave the **Messages** window open for the rest of the lab.

Now, let's produce a message that is not JSON format to force a deserialization error in the MongoDB Atlas sink connector. Remember that this sink connector was configured with a JsonDeserializer:

52. Go to the HiveMQ websocket client and click on **Connect** if the client is not yet connected.
53. In the **Publish** section, make sure the topic is **testtopic/lab6**. Then, type in the Message box: **`not-JSON-message`**.

54. Click on **Publish**.

55. Come back to Confluent Cloud in the **Messages** window, you should see a new message in this DLQ topic.



56. Click on the message to inspect the Key, Value and Header:

    - Key: **testtopic/lab6**
    - Value: **not-JSON-message**
    - Header: Metadata of the message and information about the error

hitesh@datacouch.io

Clean up

57. Delete the two connectors:

    - In the left sidebar inside your cluster, click on **Connectors**.
    - Select the **MongoDbAtlasSinkConnector_Lab6** connector.
    - Click on **Settings**.
    - At the bottom of the window, click on **Delete connector**. Type the connector name and click **Confirm**.
    - Repeat the same steps to delete the **MqttSourceConnector_Lab6** connector.

Conclusion

In this lab you have learned how to create fully-managed connectors in Confluent Cloud. You

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

have created a MQTT Source connector to get data from an online MQTT Broker. Then, you have created a MongoDB Atlas Sink connector to send JSON data from a Confluent Cloud topic to a MongoDB collection.

You have also learned what the Egress IP addresses are and how to use them to allow Confluent Cloud to access MongoDB. Finally, you have worked with Dead Letter Queues for sink connectors.

STOP HERE. THIS IS THE END OF THE EXERCISE.

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Lab 07 ksqlDB

a. Using ksqlDB in Confluent Cloud for stream processing

The goal of this lab is to set up a kqslDB cluster in Confluent Cloud following best practices and to run ksqlDB queries to perform real time stream processing.

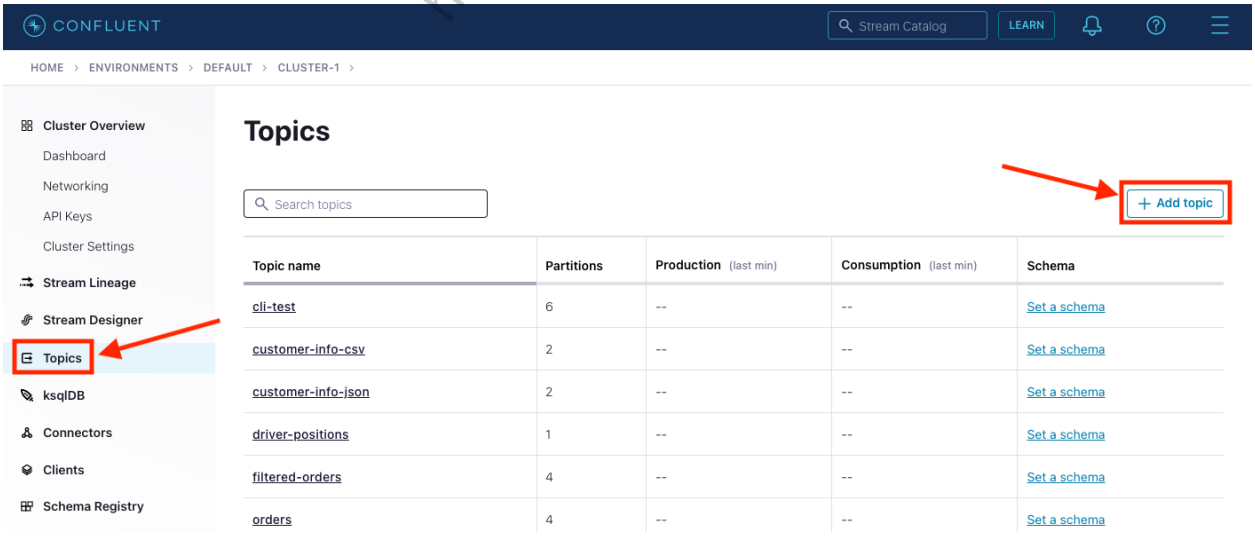
For this lab, you will use data based on a Gaming Platform (gaming player profiles, gaming player activity and gaming rooms).

Prerequisites

hitesh@datacouch.io

Creation of four topics

1. In Confluent Cloud website, go to your cluster in the **default** environment.
2. Navigate to the cluster you created in Lab 1, which should be inside the Environment called **default**.
3. In the left pane, click on **Topics**.



4. Create 3 topics with **1 partition** using the following topic names:
    - **gaming_rooms**
    - **gaming_players**

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

- **gaming_player_activity**

Creation of ksqlDB cluster

5. In the left pane, click on **ksqlDB**.
6. Then, click on **Create cluster myself**.
7. Select **Granular access** and choose **Create a new one**:
   - New service account name: **sa-ksqldb-lab7**
   - Description: **Service Account for the ksqlDB cluster of Lab7**
   - **Check the box** for "Adding all required ACLs" and click on **Continue**
8. Allow access to the `gaming` topics, by using the prefix option. Click on **Add access by prefix**. Then:
   - Topic prefix: **gaming**
   - Allow: **Read**, **Write**, **Alter** and **Create**
   - Click on **Continue**
9. Set the name of the ksqlDB cluster to **ksqlDB_cluster_lab7**.
10. Click on **Launch cluster!**.

hitesh@datacouch.io

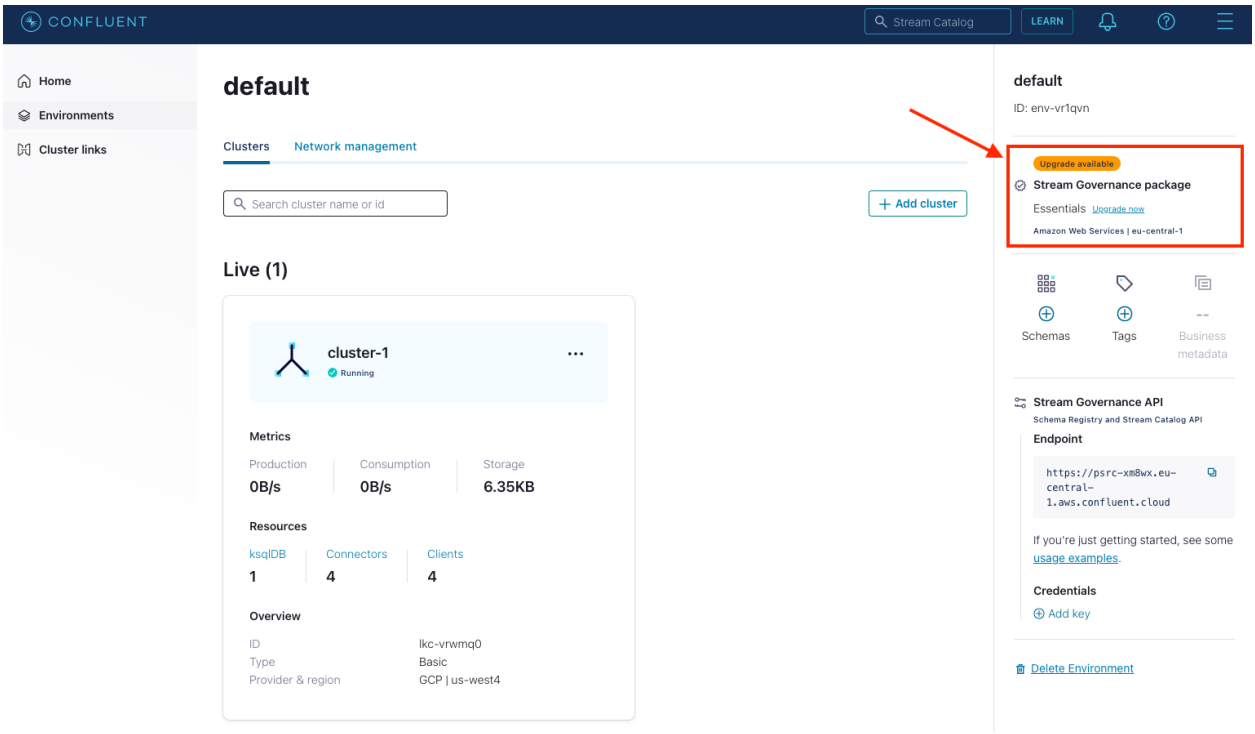
Service Account access to Schema Registry

The provisioning of the ksqlDB cluster will take a few minutes. In the meantime, configure the new Service Account **sa-ksqldb-lab7** with the required privileges to access Schema Registry.

11. First, make sure you have Schema Registry enabled in the **default** Environment.

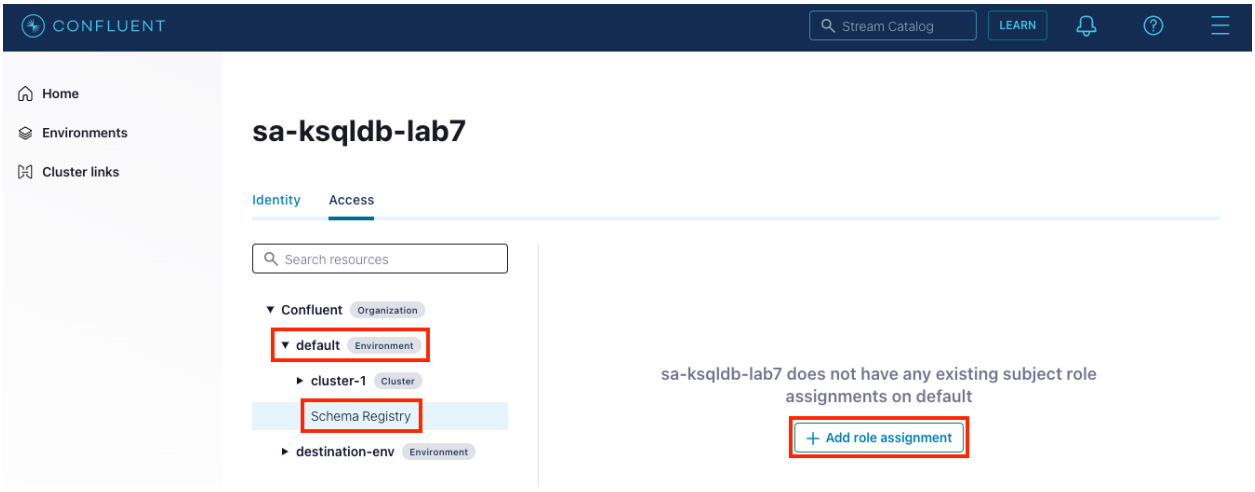
You can use either the Stream Governance Essential or Advanced package.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



12. Go to **Main Menu** (three lines icon in the upper-right corner) and click on **Accounts & access**.
13. In this window, select the **Service accounts** option in the **Accounts** tab.
14. Click on the Service Account: **sa-ksqldb-lab7**
15. Now, click on the **Access** tab.
16. Drill down in the tree structure to select **Schema Registry** in the **default** environment.

hitesh@datacouch.io



17. Click on **Add role assignment**. Then, choose the following options:
    - **All schema subjects**
    - Role: **DeveloperWrite**

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

- Click **Save**

DeveloperWrite role can write in the subject, but also read in the subject. Check access for different roles in this link: RBAC for Confluent Cloud Schema Registry

Generate data to the new topics

To produce data to the topics, you are going to use Datagen Source connectors, one per topic.

18. Login to your Confluent Cloud account using the Confluent CLI:

```
$ confluent login
Enter your Confluent Cloud credentials:
Email: <enter email>
Password: <enter password>
```

19. Run the following command to create a new Service Account for the Datagen connectors:

```
$ confluent iam service-account create sa-gaming-connectors
--description "Service Account for the Datagen connectors in Lab7"
+-------------+---------------------------------+
| ID          | <service-account-ID>            |
| Name        | sa-gaming-connectors            |
| Description | Service Account for the         |
|             | Datagen connectors in Lab7      |
+-------------+---------------------------------+
```

Write down the **Service Account ID**, you will need it in the next steps. You will use this Service Account for all the Datagen Source connectors.

20. Let's apply the required permissions using ACLs, so the Service Account can write to the topics . Replace the highlighted text with your Service Account ID and then run the command:

hitesh@datacouch.io

```
$ confluent kafka acl create --allow --service-account <service-
account-ID> --operations write --topic "gaming" --prefix
         Principal          | Permission | Operation | Resource Type
| Resource Name | Pattern Type
----------------------------+------------+-----------+
---------------+---------------+--------------
  User:<service-account-ID> | ALLOW      | WRITE     | TOPIC
| gaming        | PREFIXED
```

Make sure your cluster in the **default** environment is selected as the `Current` cluster in your Confluent CLI.

Use:

```
confluent environment list
confluent environment use <env-id>
confluent kafka cluster list
confluent kafka cluster use <cluster-id>
```

21. Now, go back to Confluent Cloud website and go to your cluster in the **default** environment.
22. In the left sidebar, click on **ksqlDB**.
23. Click on the **ksqlDB_cluster_lab7**. Its status should be `Up`, if it's still `Provisioning`, wait a few more minutes.
24. In the Editor box, paste the following command to create the Datagen Source connectors. Replace the highlighted areas with the Service Account ID of **sa-gaming-connectors**:

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
CREATE SOURCE CONNECTOR Datagen_Gaming_Rooms WITH(
    'connector.class'='DatagenSource',
    'kafka.auth.mode'='SERVICE_ACCOUNT',
    'kafka.service.account.id'='<mark><service-account-
ID></mark>',
    'kafka.topic'='gaming_rooms',
    'output.data.format'='JSON_SR',
    'quickstart'='GAMING_GAMES',
    'max.interval'='1000',
    'tasks.max'='1');

CREATE SOURCE CONNECTOR Datagen_Gaming_Players WITH(
    'connector.class'='DatagenSource',
    'kafka.auth.mode'='SERVICE_ACCOUNT',
    'kafka.service.account.id'='<mark><service-account-
ID></mark>',
    'kafka.topic'='gaming_players',
    'output.data.format'='JSON_SR',
    'quickstart'='GAMING_PLAYERS',
    'max.interval'='1000',
    'tasks.max'='1');

CREATE SOURCE CONNECTOR Datagen_Gaming_Player_Activity WITH(
    'connector.class'='DatagenSource',
    'kafka.auth.mode'='SERVICE_ACCOUNT',
    'kafka.service.account.id'='<mark><service-account-
ID></mark>',
    'kafka.topic'='gaming_player_activity',
    'output.data.format'='JSON_SR',
    'quickstart'='GAMING_PLAYER_ACTIVITY',
    'max.interval'='1000',
    'tasks.max'='1');
```

https://datazoom.io

25. Click **Run query** and then **Continue** in the Review Connectors pop-up window.

26. In **Connectors**, check that the three connectors have been correctly created.

Stream Processing with ksqlDB

27. First, let's create two TABLES from the topics `gaming_rooms` and `gaming_players`, and one STREAM from topic `gaming_player_activity`. Run this query in the Editor box under **ksqlDB**:

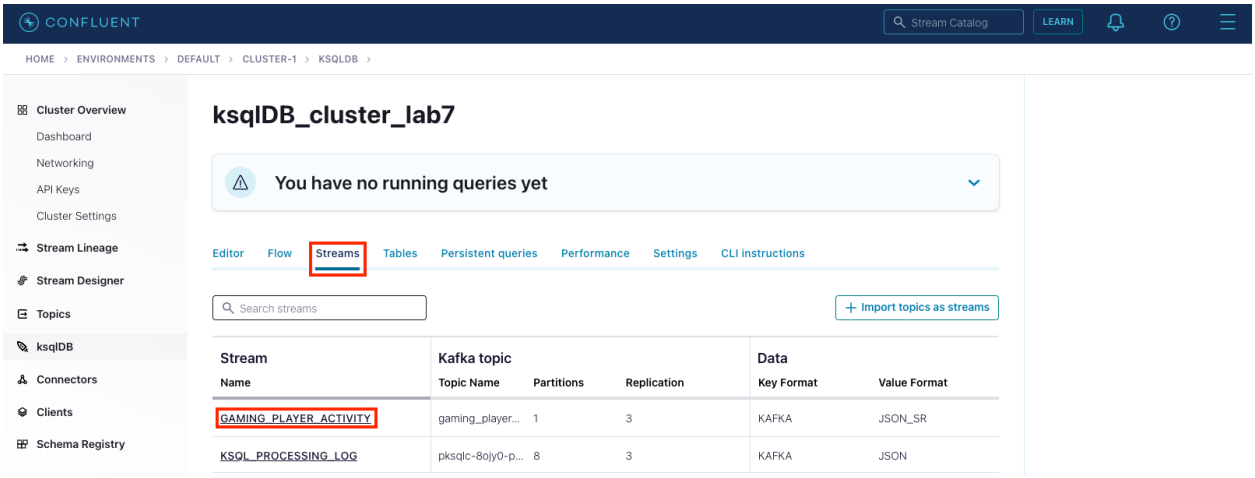
© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
CREATE TABLE gaming_rooms (
      game_room_id VARCHAR PRIMARY KEY
    ) WITH (
      KAFKA_TOPIC = 'gaming_rooms',
      VALUE_FORMAT = 'JSON_SR'
    );

CREATE TABLE gaming_players (
      id VARCHAR PRIMARY KEY
    ) WITH (
      KAFKA_TOPIC = 'gaming_players',
      VALUE_FORMAT = 'JSON_SR'
    );

CREATE STREAM gaming_player_activity (
      player_id VARCHAR,
      game_room_id VARCHAR,
      points INTEGER
    ) WITH (
      KAFKA_TOPIC = 'gaming_player_activity',
      VALUE_FORMAT = 'JSON_SR'
    );
```

hitesh@datacouch.io

The column information for the TABLES is automatically loaded from Schema Registry. However, for the STREAM **`gaming_player_activity`**, we have explicitly defined the column/format since we want to cast the **`player_id`** and **`game_room_id`** to VARCHAR format.

28. Inspect the created TABLES and STREAM. Click on the tab **Streams** or **Tables**, then click on one of them to view additional information; like, the topic related to this Stream/Table or its Schema.



© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

29. Now, you need to run a ksql query to enrich the stream `gaming_player_activity` with data from the tables `gaming_players` and `gaming_rooms`.

    The stream `gaming_player_activity` contains information about player's current points in a specific game room:

    - `player_id`
    - `game_room_id`
    - `points`

    The table `gaming_players` contains information about each player:

    - `id` (which is the `player_id` in VARCHAR format)
    - `player_name`
    - `ip`

    The table `gaming_rooms` contains information about each game room:

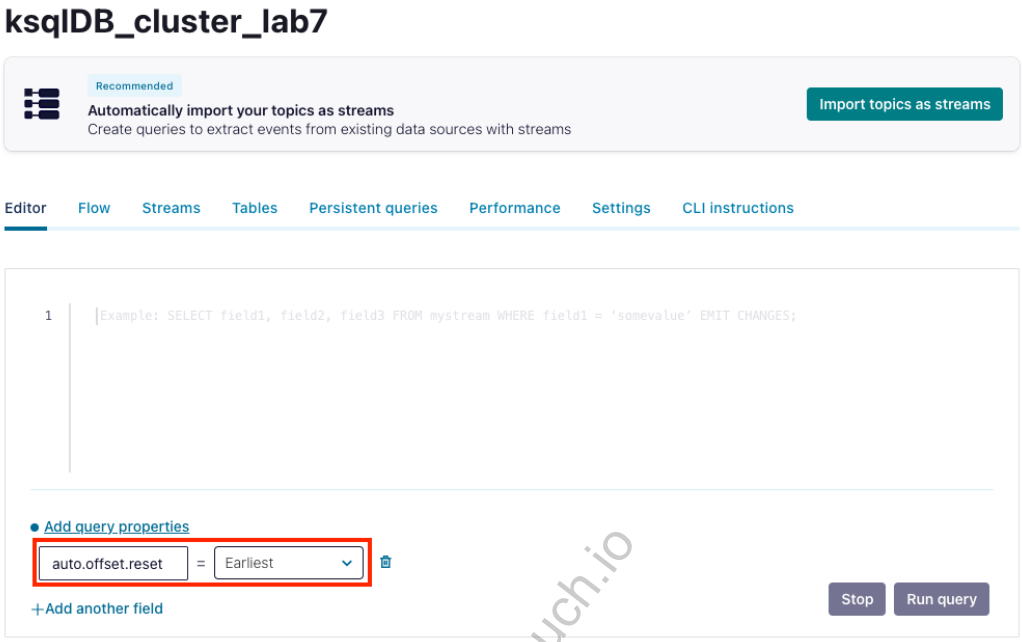
    - `game_room_id`
    - `room_name`
    - `created_date`

    Write a query to JOIN the stream `gaming_player_activity` with the two tables `gaming_players` and `gaming_rooms`. The result should be a new stream called `gaming_player_room_name_activity` with the following columns:

    - `game_room_id`
    - `player_id`
    - `points`
    - `player_name`
    - `room_name`

hitesh@datatouch.io

Before you run the queries, change the **auto.offset.reset** property to **Earliest**.



nitesh@datacouch.io

SOLUTION

```
CREATE STREAM gaming_player_room_name_activity WITH (
    KAFKA_TOPIC = 'gaming_player_room_name_activity',
    PARTITIONS = 1
  ) AS
  SELECT
    a.player_id AS player_id,
    a.game_room_id AS game_room_id,
    a.points,
    p.player_name,
    r.room_name
  FROM gaming_player_activity a
    INNER JOIN gaming_players p ON a.player_id = p.id
    INNER JOIN gaming_rooms r ON a.game_room_id =
r.game_room_id;
```

30. After running the query, go to the tab **Flow** and click on the **gaming_player_room_name_activity** stream to view its messages.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

31. Now, you need to know how many players have played each game based on its room_name, not the room_id. To get the result you may need to use `COUNT_DISTINCT()` and `GROUP BY()`.

    The result of the query should be a TABLE called `gaming_total_players_per_game` with the following columns:

    - `room_name`
    - `total_players`

hitesh@datacouch.io

    What is the total number of players per game?

SOLUTION

```
CREATE TABLE gaming_total_players_per_game WITH (
    KAFKA_TOPIC = 'gaming_total_players_per_game',
    PARTITIONS = 1
  ) AS
  SELECT
    room_name,
    COUNT_DISTINCT(player_id) AS total_players
  FROM gaming_player_room_name_activity
    GROUP BY room_name;
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

32. The last two queries have been PUSH queries, as they are persistently running. Let's now run a PULL query to the table `gaming_total_players_per_game`.

    Get the total number of players for the game `Arcade -- Expert`.

    SOLUTION

```
SELECT
  *
FROM gaming_total_players_per_game
  WHERE room_name = 'Arcade -- Expert';
```

33. Finally, create a TABLE indicating the number of active players per game in each minute. To get the result you may need to use `COUNT_DISTINCT()`, `GROUP BY()` and `WINDOW TUMBLING()`.

    The result of the query should be a TABLE called `gaming_active_players_per_minute` with at least the following columns:

    - `room_name`
    - `active_players`

hitesh@datatouch.io

    SOLUTION

```
CREATE TABLE gaming_active_players_per_minute WITH (
    KAFKA_TOPIC = 'gaming_active_players_per_minute',
    PARTITIONS = 1
  ) AS
  SELECT
    room_name,
    TIMESTAMPTOSTRING(WINDOWSTART, 'yyyy-MM-dd HH:mm') AS
window_start,
    TIMESTAMPTOSTRING(WINDOWEND, 'yyyy-MM-dd HH:mm') AS
window_end,
    COUNT_DISTINCT(player_id) AS active_players
  FROM gaming_player_room_name_activity
    WINDOW TUMBLING (SIZE 1 MINUTE)
    GROUP BY room_name;
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Clean up

34. Delete the ksqlDB cluster:
    - In the left sidebar, click **ksqlDB**.
    - Click on **Delete** in the `ksqlDB_cluster_lab7` row.
35. Delete the three connectors:
    - In the left sidebar, click on **Connectors**.
    - Select the **DATAGEN_GAMING_ROOMS** connector.
    - Click on **Settings**.
    - Scroll down, click on **Delete connector**. Type the connector name and click **Confirm**.
    - Repeat the same steps to delete the **DATAGEN_GAMING_PLAYERS** and **DATAGEN_GAMING_PLAYER_ACTIVITY** connector.

Conclusion

In this lab you have learned how to set up a ksqlDB cluster in Confluent Cloud following best practices. You have created a three Datagen Source connectors using ksqlDB. You have also created two tables and one stream with data from Kafka topics. Finally, you have run Push and Pull queries to process your data in real time using joins, windows and aggregations.

nitesh@datacouch.io

STOP HERE. THIS IS THE END OF THE EXERCISE.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Lab 09 Automation in Confluent Cloud using Terraform

a. Deploy a Terraform Confluent Provider

The goal of this lab is to:

- Install Terraform.
- Configure a Terraform provider in Confluent Cloud.
- Test the Terraform provider in Confluent Cloud.
- Destroy the Terraform provider.

hitesh@datacouch.io

Install Terraform

1. Ensure that your system is up to date and you have installed the `gnupg` and `software-properties-common` packages. You will use these packages to verify HashiCorp's GPG signature and install HashiCorp's Debian package repository.

   Password: `training`

   ```
   $ sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
   ```

2. Install the HashiCorp GPG key.

   ```
   $ wget -O- https://apt.releases.hashicorp.com/gpg | gpg --dearmor | sudo tee /usr/share/keyrings/hashicorp-archive-keyring.gpg
   ```

3. Verify the key's fingerprint.

   ```
   $ gpg --no-default-keyring \
        --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg \
        --fingerprint
   ```

   The gpg command will report the key fingerprint.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Refer to the Official Packaging Guide for the latest public signing key. You can also verify the key on Security at HashiCorp under Linux Package Checksum Verification.

4. Add the official HashiCorp repository to your system. The `lsb_release -cs` command finds the distribution release codename for your current system, such as buster, groovy, or sid.

```
$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-
keyring.gpg] \
        https://apt.releases.hashicorp.com $(lsb_release -cs) main" |
\
  sudo tee /etc/apt/sources.list.d/hashicorp.list
```

5. Download the package information from HashiCorp.

```
$ sudo apt update
```

6. Install Terraform from the new repository.

```
$ sudo apt-get install terraform
```

hitesh@datacouch.io

Now that you have added the HashiCorp repository, you can install Vault, Consul, Nomad and Packer with the same command.

Verify the Installation

7. Verify that the installation worked by opening a new terminal session and listing Terraform's available subcommands.

```
$ terraform -help
```

8. Add any subcommand to `terraform -help` to learn more about what it does and available options.

```
$ terraform -help plan
```

If you get an error that Terraform could not be found, your **PATH** environment variable was not set up properly. Please go back and ensure that your **PATH** variable contains the directory where Terraform was installed.

Create a Cloud API Key

9. Go to Confluent Cloud website and open the administration menu (▯) in the upper-right corner of the Confluent Cloud user interface.
10. Click **Cloud API keys** in the ADMINISTRATION section.
11. Click **Create key**. Then, follow these instructions:
    a. Access control: Select **Granular access**, then click **Next**.
    b. Service account: Select **Create a new one**. Then, type name and description:
       - Name: **sa-terraform**
       - Description: **Service Account for Terraform**
    c. Get your API key: Type **`Terraform API key`** in the **Description** and click on **Download and continue** to store the API key/secret locally.

nitesh@datacouch.io

You will need this API key and secret **to use the Confluent Terraform Provider**.

Add the *OrganizationAdmin* role to the *sa-terraform* Service Account

12. Open the administration menu (▯) in the upper-right corner of the Confluent Cloud user interface.
13. Click **Accounts & access** in the ADMINISTRATION section.
14. Now, click **Service accounts** button in the Accounts tab.
15. Select the Service Account **sa-terraform**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

16. In the Service Account's details page, click the **Access** tab.
17. Here in the tree view, click on the top level of the tree view which should be the `Organization` level.
18. On the right side, assign a new role. Select the `OrganizationAdmin` role and click **Save**.
19. When you return to **Accounts & access**, you can view the **Access** for the organization, and also see that the Service Account you created has the `OrganizationAdmin` role binding.

Create Resources on Confluent Cloud via Terraform

20. In the Terminal, create the terraform directory and go to that directory:

```
$ mkdir ~/confluent-ccl/lab-terraform && cd ~/confluent-ccl/lab-terraform
```

21. Clone the Confluent GitHub repository which contains a lot of different Confluent Cloud deployments for Terraform:

```
$ git clone https://github.com/confluentinc/terraform-provider-confluent.git
```

hitesh@datacouch.io

22. Change into configurations subdirectory and list its content:

```
$ cd terraform-provider-confluent/examples/configurations && ls -l
```

The **configurations** directory has a subdirectory for each of the following configurations:

Configuration	Description
`basic-kafka-acls`	Basic Kafka cluster with authorization using ACLs
`basic-kafka-acls-with-alias`	Basic Kafka cluster with authorization using ACLs
`standard-kafka-acls`	Standard Kafka cluster with authorization using ACLs
`standard-kafka-rbac`	Standard Kafka cluster with authorization using RBAC
`dedicated-public-kafka-acls`	Dedicated Kafka cluster that is accessible over the public internet with authorization using ACLs

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Configuration	Description
`dedicated-public-kafka-rbac`	Dedicated Kafka cluster that is accessible over the public internet with authorization using RBAC
`dedicated-privatelink-aws-kafka-acls`	Dedicated Kafka cluster on AWS that is accessible via PrivateLink connections with authorization using ACLs
`dedicated-privatelink-aws-kafka-rbac`	Dedicated Kafka cluster on AWS that is accessible via PrivateLink connections with authorization using RBAC
`dedicated-privatelink-azure-kafka-rbac`	Dedicated Kafka cluster on Azure that is accessible via PrivateLink connections with authorization using RBAC
`dedicated-privatelink-azure-kafka-acls`	Dedicated Kafka cluster on Azure that is accessible via PrivateLink connections with authorization using ACLs
`dedicated-private-service-connect-gcp-kafka-acls`	Dedicated Kafka cluster on GCP that is accessible via Private Service Connect connections with authorization using ACLs
`dedicated-private-service-connect-gcp-kafka-rbac`	Dedicated Kafka cluster on GCP that is accessible via Private Service Connect connections with authorization using RBAC
`dedicated-vnet-peering-azure-kafka-acls`	Dedicated Kafka cluster on Azure that is accessible via VPC Peering connections with authorization using ACLs
`dedicated-vnet-peering-azure-kafka-rbac`	Dedicated Kafka cluster on Azure that is accessible via VPC Peering connections with authorization using RBAC
`dedicated-vpc-peering-aws-kafka-acls`	Dedicated Kafka cluster on AWS that is accessible via VPC Peering connections with authorization using ACLs
`dedicated-vpc-peering-aws-kafka-rbac`	Dedicated Kafka cluster on AWS that is accessible via VPC Peering connections with authorization using RBAC
`dedicated-vpc-peering-gcp-kafka-acls`	Dedicated Kafka cluster on GCP that is accessible via VPC Peering connections with authorization using ACLs
`dedicated-vpc-peering-gcp-kafka-rbac`	Dedicated Kafka cluster on GCP that is accessible via VPC Peering connections with authorization using RBAC
`dedicated-transit-gateway-attachment-aws-kafka-acls`	Dedicated Kafka cluster on AWS that is accessible via Transit Gateway Endpoint with authorization using ACLs
`dedicated-transit-gateway-attachment-aws-kafka-rbac`	Dedicated Kafka cluster on AWS that is accessible via Transit Gateway Endpoint with authorization using RBAC

hitesh@dataculus.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Basic Kafka cluster with authorization using RBAC configuration is not supported, because both DeveloperRead and DeveloperWrite roles are not available for Basic Kafka clusters.

23. Select the target configuration **basic-kafka-acls**, change into its directory and list its content:

```
$ cd basic-kafka-acls && ls -l
total 20
-rw-r--r-- 1 training users  231 Jul 27 00:27 README.md
-rw-r--r-- 1 training users 7888 Jul 27 00:27 main.tf
-rw-r--r-- 1 training users 3289 Jul 27 00:27 outputs.tf
-rw-r--r-- 1 training users  268 Jul 27 00:27 variables.tf
```

24. Open the **main.tf** file to view the changes you will apply to your Confluent Cloud organization. Try to understand what resources will be created based on what you have learned in class. Run:

```
$ code main.tf
```

hitesh@datacouch.io

25. Initialize Terraform by running the following command:

```
$ terraform init
```

26. Use the saved Cloud API Key of the **sa-terraform** service account from Step 11 to set values to the **confluent_cloud_api_key** and **confluent_cloud_api_secret** input variables using environment variables:

```
$ export TF_VAR_confluent_cloud_api_key="<cloud_api_key>"
```

```
$ export TF_VAR_confluent_cloud_api_secret="<cloud_api_secret>"
```

27. Ensure the configuration is syntactically valid and internally consistent:

```
$ terraform validate
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

28. View the execution plan before applying the changes:

```
$ terraform plan
```

29. Apply the configuration:

```
$ terraform apply
```

30. You have now created infrastructure using Terraform. Go to Confluent Cloud website to see the resources you provisioned.

Run a Test

31. Run the following command to print out generated Confluent CLI code out commands with the correct resource IDs injected:

```
$ terraform output resource-ids
```

Your output should resemble:

hitesh@datacouch.io

```
<<EOT
Environment ID:   env-zgvzo0
Kafka Cluster ID: lkc-230k72
Kafka topic name: orders

Service Accounts and their Kafka API Keys (API Keys inherit the
permissions granted to the owner):
app-manager:                    sa-dw0d91
app-manager's Kafka API Key:    "FAOE6KXGLKSLQLFQ"
app-manager's Kafka API Secret:
"9Z0M+hxhuygG9jeCIBifiVC45Tm341ZYuR8CE0P8O7vnYGz5qsR1zZ4JP3lam7cq"

app-producer:                   sa-687v38
app-producer's Kafka API Key:   "KZUHJEOJLY24IIFG"
app-producer's Kafka API Secret:
"pqzRws/w3Dh9Vcd7fV8yYsJa3cDDoKxedcDhjyGUwcRJHZwkFD2Er0qZTMf6m/Yn"

app-consumer:                   sa-561pwq
app-consumer's Kafka API Key:   "6OAQNM2TIYPA36T3"
app-consumer's Kafka API Secret:
"AkEKxQpMPZm8W1OPeT1lixzil3Oz6AW4c2u7vuOCd896wxsrppIFdITKLeVWrzVG"

In order to use the Confluent CLI v2 to produce and consume messages
from topic 'orders' using Kafka API Keys
of app-producer and app-consumer service accounts
run the following commands:

# 1. Log in to Confluent Cloud
$ confluent login

# 2. Produce key-value records to topic 'orders' by using app-
producer's Kafka API Key
$ confluent kafka topic produce orders --environment env-zgvzo0
--cluster lkc-230k72 --api-key "KZUHJEOJLY24IIFG" --api-secret
"pqzRws/w3Dh9Vcd7fV8yYsJa3cDDoKxedcDhjyGUwcRJHZwkFD2Er0qZTMf6m/Yn"
# Enter a few records and then press 'Ctrl-C' when you're done.
# Sample records:
# {"number":1,"date":18500,"shipping_address":"899 W Evelyn Ave,
Mountain View, CA 94041, USA","cost":15.00}
# {"number":2,"date":18501,"shipping_address":"1 Bedford St, London
WC2E 9HG, United Kingdom","cost":5.00}
# {"number":3,"date":18502,"shipping_address":"3307 Northland Dr
Suite 400, Austin, TX 78731, USA","cost":10.00}

# 3. Consume records from topic 'orders' by using app-consumer's
Kafka API Key
$ confluent kafka topic consume orders --from-beginning --environment
env-zgvzo0 --cluster lkc-230k72 --api-key "6OAQNM2TIYPA36T3" --api
-secret
"AkEKxQpMPZm8W1OPeT1lixzil3Oz6AW4c2u7vuOCd896wxsrppIFdITKLeVWrzVG"
# When you are done, press 'Ctrl-C'.

EOT
```

hitesh@datacouch.io

32. Follow the printed out instructions that you got to test your new resources.

Clean-Up

33. Run the following command to destroy all the resources you created:

```
$ terraform destroy
```

This command destroys all the resources specified in your Terraform state. `terraform destroy` doesn't destroy resources running elsewhere that aren't managed by the current Terraform project.

In this exercsie, we created and destroyed an entire Confluent Cloud deployment.

34. Go to Confluent Cloud website to verify the resources have been destroyed to avoid unexpected charges.

hitesh@datacouch.io

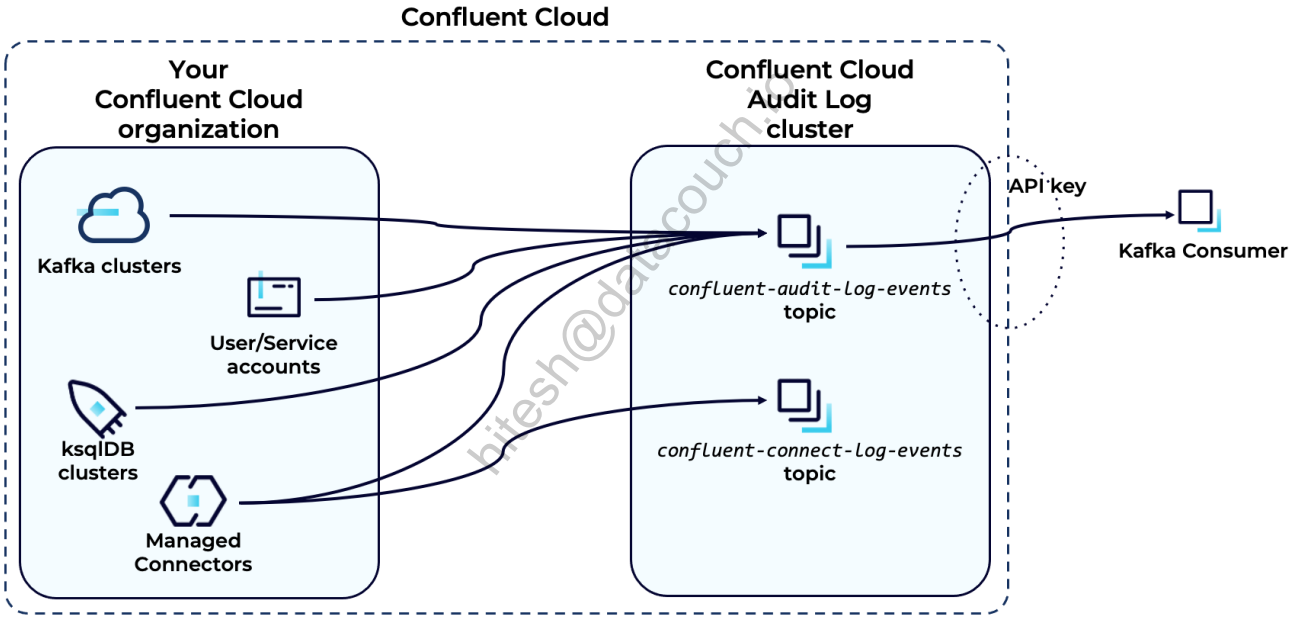
STOP HERE. THIS IS THE END OF THE EXERCISE.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Lab 10 Confluent Cloud Audit Logs & Metrics

a. Confluent Cloud Audit Logs & Metrics

The goal of this lab is to get familiar with the Audit Logs and Metrics API of Confluent Cloud. You will consume Audit Log messages using the Confluent CLI and Kafka consumers. In the second part of the lab, you will perform multiple requests using the Metrics API to get the available metrics or the number of requests per principal. All the tasks that you will perform will follow recommended best practices.



bitesh@datacouch.io

Prerequisite - Create a Standard Cluster

1. Go to Confluent Cloud website and log in into your account.
2. Browse to your **default** environment and **create a new cluster**:
   a. Choose a **Standard cluster** by clicking on **Begin configuration**.
   b. Choose your preferred **cloud provider**, **region**, and **availability**.
   c. Click on **Skip payment**.
   d. Give your cluster a **name** and review **Configuration & cost**, **Usage limits** and **Uptime SLA**. Then, select **Launch cluster**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Generate Audit Log Events

Create a Topic

Once the cluster is up and running, create a topic.

3. Go into the new Standard cluster, click on Topics and then click on **Create topic**.
4. Apply the following Topic configuration:
    - Topic name: **test-topic**
    - Partitions: **6**
5. Then, click on **Create with defaults**.

hitesh@datacouch.io

6. Click on **Skip** for the "Define a data contract" window.

Delete the Topic

7. Inside the **test-topic**, click on the **Configuration** tab.
8. Now, click on **Delete topic** and then confirm the deletion.

Inspect the Audit Log cluster

9. In the **Virtual Machine**, open a new Terminal window and go to the lab directory:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ cd ~/confluent-ccl/lab-audit-logs
```

10. Log in to your Confluent Cloud organization using the Confluent CLI:

```
$ confluent login
```

11. Run the following command to view information about the Audit Log topic:

```
$ confluent audit-log describe
+-----------------+------------------------------+
| Cluster         | <cluster-ID>                 |
| Environment     | <environment-ID>             |
| Service Account | <service-account-ID>         |
| Topic Name      | confluent-audit-log-events   |
+-----------------+------------------------------+
```

Response Sample

```
+-----------------+------------------------------+
| Cluster         | lkc-d6071                    |
| Environment     | env-w8q9m                    |
| Service Account | sa-yom2mj                    |
| Topic Name      | confluent-audit-log-events   |
+-----------------+------------------------------+
```

hitesh@datacouch.io

Description of the response:

- **Cluster**: cluster ID of the Audit Log cluster
- **Environment**: environment ID to which the Audit Log cluster belongs
- **Service Account**: service account ID that is automatically created for you to be used to read from your audit log topic. This service account is properly configured to have just the access to read the audit log topic.
- **Topic Name**: topic name where audit log messages are written

12. Select the correct environment to which the Audit Log cluster belongs:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ confluent environment use <environment-ID>
Now using "<environment-ID>" as the default (active) environment.
```

13. Select the Audit Log cluster as your `Current` cluster:

```
$ confluent kafka cluster use <cluster-ID>
Set Kafka cluster "<cluster-ID>" as the active cluster for
environment "<environment-ID>".
```

14. Now, you can describe the Audit Log cluster using its `<cluster-ID>` to view more details, e.g. cluster endpoint:

```
$ confluent kafka cluster describe <cluster-ID>
+----------------------
+-------------------------------------------------------+
| Current               | true
|
| ID                    | <cluster-ID>
|
| Name                  | _confluent_audit_log_cluster
|
| Type                  | STANDARD
|
| Ingress Limit (MB/s) |
250 |
| Egress Limit (MB/s)  |
750 |
| Storage               | Infinite
|
| Provider              | aws
|
| Region                | us-west-2
|
| Availability          | single-zone
|
| Status                | UP
|
| Endpoint              | SASL_SSL://<host>:9092
|
| REST Endpoint         | https://<host>:443
|
| Topic Count           |
2 |
+----------------------
+-------------------------------------------------------+
```

hitesh@datacouch.io

Response Sample

```
+----------------------
+-------------------------------------------------------------+
| Current                | true
|
| ID                     | lkc-d6071
|
| Name                   | _confluent_audit_log_cluster
|
| Type                   | STANDARD
|
| Ingress Limit (MB/s) |
250 |
| Egress Limit (MB/s)  |
750 |
| Storage                | Infinite
|
| Provider               | aws
|
| Region                 | us-west-2
|
| Availability           | single-zone
|
| Status                 | UP
|
| Endpoint               | SASL_SSL://pkc-r50y0.us-west-
2.aws.confluent.cloud:9092 |
| REST Endpoint          | https://pkc-r50y0.us-west-
2.aws.confluent.cloud:443    |
| Topic Count            |
2 |
+----------------------
+-------------------------------------------------------------+
```

hitesh@datacouch.io

Write down the bootstrap-server for the Audit Log cluster, which is the highlighted part **<host>:9092**.
You will need it later to configure the Audit Log consumer.

Create API Key and API Secret to access the Audit Log cluster

To create a new API key/secret for the Audit Log cluster, it **must** be owned by the Service Account shown in Step 11.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

15. Create a new API key/secret for your audit log cluster. Run the following command:

```
$ confluent api-key create --resource <cluster-ID> --service-account
<service-account-ID>
It may take a couple of minutes for the API key to be ready.
Save the API key and secret. The secret is not retrievable later.
+------------
+-------------------------------------------------------------------+
| API Key    | <audit-log-api-key>
|
| API Secret | <audit-log-api-secret>
|
+------------
+-------------------------------------------------------------------+
```

Note that this new API Key is owned by `<service-account-ID>`, since you have used the flag `--service-account`, which indicates which service account will own the new API Key.

Remember that there is a limit of 2 API Keys per Audit Log cluster.

hitesh@datacouch.io

(OPTIONAL) Consume Audit Logs using the Confluent CLI

16. If you want to directly consume the Audit Log messages with the Confluent CLI, then you must tell your CLI which API Key to use for the Audit Log cluster. To do so, run:

```
$ confluent api-key use <audit-log-api-key> --resource <cluster-ID>
Set API Key "<audit-log-api-key>" as the active API key for
"<cluster-ID>".
```

17. Consume Audit Log messages from the `confluent-audit-log-events` topic:

```
$ confluent kafka topic consume -b confluent-audit-log-events
{"datacontenttype":"application/json","data":{"serviceName":"crn://co
nfluent.cloud/","methodName":"mds.Authorize","resourceName":"crn://co
nfluent.cloud/organization=78f34236-056b-4631-a77f-
00f47e028fd8/environment=env-n1wpd/schema-registry=lsrc-
vmvqp","authenticationInfo":{"principal":"User:u-
41wr56"},"authorizationInfo":{"granted":true,"operation":"AccessWithT
oken","resourceType":"SchemaRegistry","resourceName":"schema-
registry-
cluster","patternType":"LITERAL","rbacAuthorization":{"role":"Organiz
ationAdmin","scope":{"outerScope":["organization=78f34236-056b-4631-
a77f-00f47e028fd8"]}}},"request":{"correlation_id":"-
1"},"requestMetadata":{}},"subject":"crn://confluent.cloud/organizati
on=78f34236-056b-4631-a77f-00f47e028fd8/environment=env-n1wpd/schema-
registry=lsrc-vmvqp","specversion":"1.0","id":"dc1561b6-29c8-4e55-
95d7-94777b8c6ca2","source":"crn://confluent.cloud/","time":"2023-07-
07T02:21:00.160837188Z","type":"io.confluent.kafka.server/authorizati
on"}
% Headers: [content-type="application/cloudevents+json; charset=UTF-
8"]
{"datacontenttype":"application/json","data":{"serviceName":"crn://co
nfluent.cloud/","methodName":"mds.Authorize","resourceName":"crn://co
nfluent.cloud/organization=78f34236-056b-4631-a77f-
00f47e028fd8/environment=env-w5xww5/schema-registry=lsrc-
xrqddx","authenticationInfo":{"principal":"User:u-
41wr56"},"authorizationInfo":{"granted":true,"operation":"AccessWithT
oken","resourceType":"SchemaRegistry","resourceName":"schema-
registry-
cluster","patternType":"LITERAL","rbacAuthorization":{"role":"Organiz
ationAdmin","scope":{"outerScope":["organization=78f34236-056b-4631-
a77f-00f47e028fd8"]}}},"request":{"correlation_id":"-
1"},"requestMetadata":{}},"subject":"crn://confluent.cloud/organizati
on=78f34236-056b-4631-a77f-00f47e028fd8/environment=env-
w5xww5/schema-registry=lsrc-
xrqddx","specversion":"1.0","id":"ed6389a8-e24f-4b88-b27b-
f5ff67ce503b","source":"crn://confluent.cloud/","time":"2023-07-
07T02:21:00.160859331Z","type":"io.confluent.kafka.server/authorizati
on"}
...
```

18. Press **Ctrl+C** to stop the consumer.

19. Export all Audit Log messages to a local file **audit-log-events.json**:

```
$ confluent kafka topic consume -b confluent-audit-log-events >
audit-log-events-cli.json
```

20. Open a new Terminal window and filter the Audit Logs using the tool **grep** to get just the events related to **kafka.CreateTopics**:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ cd ~/confluent-ccl/lab-audit-logs && cat audit-log-events-cli.json
| grep 'kafka.CreateTopics' | jq
{
  "datacontenttype": "application/json",
  "data": {
    "serviceName": "crn://confluent.cloud/",
    "methodName": "kafka.CreateTopics",
    "cloudResources": [
      {
        "scope": {
          "resources": [
            {
              "type": "ORGANIZATION",
              "resourceId": "78f34236-056b-4631-a77f-00f47e028fd8"
            },
            {
              "type": "ENVIRONMENT",
              "resourceId": "t37335"
            },
            {
              "type": "CLOUD_CLUSTER",
              "resourceId": "lkc-dg1p7z"
            },
            {
              "type": "KAFKA_CLUSTER",
              "resourceId": "lkc-dg1p7z"
            }
          ]
        },
        "resource": {
          "type": "TOPIC",
          "resourceId": "test-topic"
        }
      }
    ],
    "authenticationInfo": {
      "principal": {
        "confluentUser": {
          "resourceId": "u-41wr56"
        }
      },
      "result": "SUCCESS",
      "identity": "crn://confluent.cloud/organization=78f34236-056b-
4631-a77f-00f47e028fd8/identity-provider=Confluent/identity=u-41wr56"
    },
    "authorizationInfo": {
      "result": "ALLOW"
    },
    "requestMetadata": {
      "connectionId": "168872691651700064",
      "requestId": [
        "168872691653200018"
      ],
      "clientId": "proxy:81458"
```

hitesh@datacouch.io

```
    },
    "request": {
      "accessType": "MODIFICATION",
      "data": {
        "name": "test-topic",
        "numPartitions": 6,
        "replicationFactor": 3,
        "assignments": [],
        "configs": [
          {
            "name": "compression.type",
            "value": "producer"
          },
          {
            "name": "leader.replication.throttled.replicas",
            "value": ""
          },
          {
            "name": "message.downconversion.enable",
            "value": "true"
          },
          {
            "name": "min.insync.replicas",
            "value": "2"
          },
          {
            "name": "segment.jitter.ms",
            "value": "0"
          },
          {
            "name": "cleanup.policy",
            "value": "delete"
          },
          {
            "name": "flush.ms",
            "value": "9223372036854775807"
          },
          {
            "name": "follower.replication.throttled.replicas",
            "value": ""
          },
          {
            "name": "default.replication.factor",
            "value": "3"
          },
          {
            "name": "retention.ms",
            "value": "604800000"
          },
          {
            "name": "segment.bytes",
            "value": "104857600"
          },
          {
```

hitesh@datacouch.io

```
      "name": "flush.messages",
      "value": "9223372036854775807"
    },
    {
      "name": "message.format.version",
      "value": "3.0-IV1"
    },
    {
      "name": "num.partitions",
      "value": "6"
    },
    {
      "name": "file.delete.delay.ms",
      "value": "60000"
    },
    {
      "name": "max.compaction.lag.ms",
      "value": "9223372036854775807"
    },
    {
      "name": "max.message.bytes",
      "value": "2097164"
    },
    {
      "name": "min.compaction.lag.ms",
      "value": "0"
    },
    {
      "name": "message.timestamp.type",
      "value": "CreateTime"
    },
    {
      "name": "preallocate",
      "value": "false"
    },
    {
      "name": "index.interval.bytes",
      "value": "4096"
    },
    {
      "name": "min.cleanable.dirty.ratio",
      "value": "0.5"
    },
    {
      "name": "unclean.leader.election.enable",
      "value": "false"
    },
    {
      "name": "delete.retention.ms",
      "value": "86400000"
    },
    {
      "name": "retention.bytes",
      "value": "-1"
```


nitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
            },
            {
              "name": "segment.ms",
              "value": "604800000"
            },
            {
              "name": "message.timestamp.difference.max.ms",
              "value": "9223372036854775807"
            },
            {
              "name": "segment.index.bytes",
              "value": "10485760"
            }
          ],
          "validateOnly": false
        }
      },
      "result": {
        "status": "SUCCESS",
        "data": {
          "message": null,
          "errorCode": 0,
          "errorType": "NONE"
        }
      },
      "resourceName": "crn://confluent.cloud/organization=78f34236-
056b-4631-a77f-00f47e028fd8/environment=t37335/cloud-cluster=lkc-
dg1p7z/kafka=lkc-dg1p7z/topic=test-topic"
    },
    "subject": "crn://confluent.cloud/organization=78f34236-056b-4631-
a77f-00f47e028fd8/environment=t37335/cloud-cluster=lkc-
dg1p7z/kafka=lkc-dg1p7z/topic=test-topic",
    "specversion": "1.0",
    "id": "1b6711e1-e651-47a7-8992-1f98042ec14e",
    "source": "crn://confluent.cloud/",
    "time": "2023-07-07T10:48:37.155096747Z",
    "type": "io.confluent.kafka.server/request"
}
...
```


heri@datarouch.io

21. Go back to the Terminal window where the Audit Log Consumer CLI is running and press `Ctrl+C` to stop it.

Consume Audit Logs using a Java Consumer

Before you start the Java Consumer, you must configure it so that it can access the Audit Log cluster.

22. Run this command to open the consumer configuration file:

```
$ cd ~/confluent-ccl/lab-audit-logs/audit-log-consumer && code
consumer.config
```

23. Replace the following properties in the `consumer.config` file with the previous highlighted values:
    - Replace `{{ BOOTSTRAP_SERVERS }}` with `<host>:9092`
    - Replace `{{ CONSUMER_API_KEY }}` with `<audit-log-api-key>`
    - Replace `{{ CONSUMER_API_SECRET }}` with `<audit-log-api-secret>`
24. Press `Ctrl+S` to save the changes of the `consumer.config` file.
25. In the Terminal window, build the Java Consumer application:

```
$ ./gradlew build
Starting a Gradle Daemon, 1 incompatible Daemon could not be reused,
use --status for details

BUILD SUCCESSFUL in 6s
6 actionable tasks: 6 executed
```

hitesh@datacouch.io

26. Start the Consumer by running the following command:

This Java Consumer, in addition to printing the Audit Log Events to the Terminal, also writes them to the `audit-log-events-consumer.json` file.

```
$ ./gradlew run --console plain
> Task :compileJava
> Task :processResources UP-TO-DATE
> Task :classes

> Task :run
Starting Java Consumer.
{
  "datacontenttype": "application/json",
  "data": {
    "requestMetadata": {"requestId": ["1da379c9-cc70-4734-a62c-
f50e38ba8656"]},
    "authenticationInfo": {
      "principal": {"confluentServiceAccount": {"resourceId": "sa-
y2ogr6"}},
      "result": "SUCCESS",
      "credentials": {
        "mechanism": "HTTP_BASIC",
        "idSecretCredentials": {"credentialId": "ZUFBVLQDHLMYF3EK"}
      }
    },
    "cloudResources": [{
      "resource": {
        "resourceId": "lsrc-2383y",
        "type": "SCHEMA_REGISTRY"
      },
      "scope": {"resources": [
        {
          "resourceId": "78f34236-056b-4631-a77f-00f47e028fd8",
          "type": "ORGANIZATION"
        },
        {
          "resourceId": "t37335",
          "type": "ENVIRONMENT"
        }
      ]}
    }],
    "methodName": "schema-registry.Authentication",
    "resourceName": "crn://confluent.cloud/organization=78f34236-
056b-4631-a77f-00f47e028fd8/environment=t37335/schema-registry=lsrc-
2383y",
    "serviceName": "crn://confluent.cloud/"
  },
  "subject": "crn://confluent.cloud/organization=78f34236-056b-4631-
a77f-00f47e028fd8/environment=t37335/schema-registry=lsrc-2383y",
  "specversion": "1.0",
  "id": "e94d0bf7-3852-4502-af46-2a8cecfcf9d7",
  "source": "crn://confluent.cloud/",
  "time": "2023-07-10T16:53:34.612741Z",
  "type": "io.confluent.sg.server/authentication"
}
...
```

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

27. Open a new Terminal window and filter the Audit Logs from the **audit-log-events-consumer.json** file. This time filter out the logs related to **kafka.DeleteTopics**:

```
$ cd ~/confluent-ccl/lab-audit-logs && cat audit-log-events-
consumer.json | grep 'kafka.DeleteTopics' | jq
{
  "datacontenttype": "application/json",
  "data": {
    "serviceName": "crn://confluent.cloud/organization=78f34236-056b-
4631-a77f-00f47e028fd8/environment=t37335/cloud-cluster=lkc-
dg1p7z/kafka=lkc-dg1p7z",
    "methodName": "kafka.DeleteTopics",
    "resourceName": "crn://confluent.cloud/organization=78f34236-
056b-4631-a77f-00f47e028fd8/environment=t37335/cloud-cluster=lkc-
dg1p7z/kafka=lkc-dg1p7z/topic=test-topic",
    "authenticationInfo": {
      "principal": "User:81458",
      "principalResourceId": "u-41wr56",
      "identity": "crn://confluent.cloud/organization=78f34236-056b-
4631-a77f-00f47e028fd8/identity-provider=Confluent/identity=u-41wr56"
    },
    "authorizationInfo": {
      "granted": true,
      "operation": "Delete",
      "resourceType": "Topic",
      "resourceName": "test-topic",
      "patternType": "LITERAL",
      "rbacAuthorization": {
        "role": "OrganizationAdmin",
        "scope": {
          "outerScope": [
            "organization=78f34236-056b-4631-a77f-00f47e028fd8"
          ]
        }
      }
    },
    "request": {
      "correlation_id": "5",
      "client_id": "proxy:81458"
    },
    "requestMetadata": {
      "request_id": "168901326134400016",
      "connection_id": "168901326129100064"
    }
  },
  "subject": "crn://confluent.cloud/organization=78f34236-056b-4631-
a77f-00f47e028fd8/environment=t37335/cloud-cluster=lkc-
dg1p7z/kafka=lkc-dg1p7z/topic=test-topic",
  "specversion": "1.0",
  "id": "906b8154-30e6-47d5-b940-eb2c5c5449b2",
  "source": "crn://confluent.cloud/organization=78f34236-056b-4631-
a77f-00f47e028fd8/environment=t37335/cloud-cluster=lkc-
```

nitesh@datacouch.io

```
dg1p7z/kafka=lkc-dg1p7z",
  "time": "2023-07-10T18:21:01.344572948Z",
  "type": "io.confluent.kafka.server/authorization"
}
...
```

28. Go back to the Terminal window where the Java Consumer is running and press `Ctrl+C` to stop it.

Using the Metrics API

So far, you have been working with the Audit Logs to experience how you can view the events using the Consumer CLI and a Java Consumer.

Now, let's learn how to use the Metrics API to view important Confluent Cloud metrics.

Create a Cloud API Key to Access the Metrics API

Remember you must use a Cloud API Key to communicate with the Metrics API.

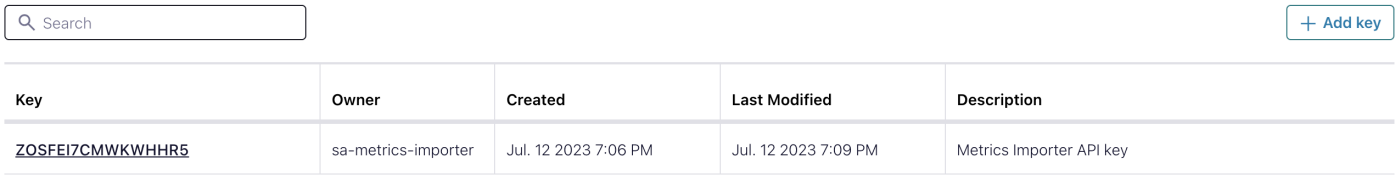
https://@datacouch.io

29. Go back to Confluent Cloud website and open the administration menu (▯) in the upper-right corner of the Confluent Cloud user interface.
30. Click **Cloud API keys** in the ADMINISTRATION section.
31. Click **Create key** or **Add key**. Then, follow these instructions:
    a. Access control: Select **Granular access**, then click **Next**.
    b. Service account: Select **Create a new one**. Then, type name and description:
        - Name: **sa-metrics-importer**
        - Description: **Service Account for a third-party monitoring tool to import Confluent Cloud metrics**
    c. Get your API key: Type `Metrics Importer API key` in the **Description** and click on **Download and continue** to store the API key/secret locally.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

You will need this API key and secret later in the lab, so be sure to safely store this information.

Cloud API keys



Add the *MetricsViewer* role to the new Service Account

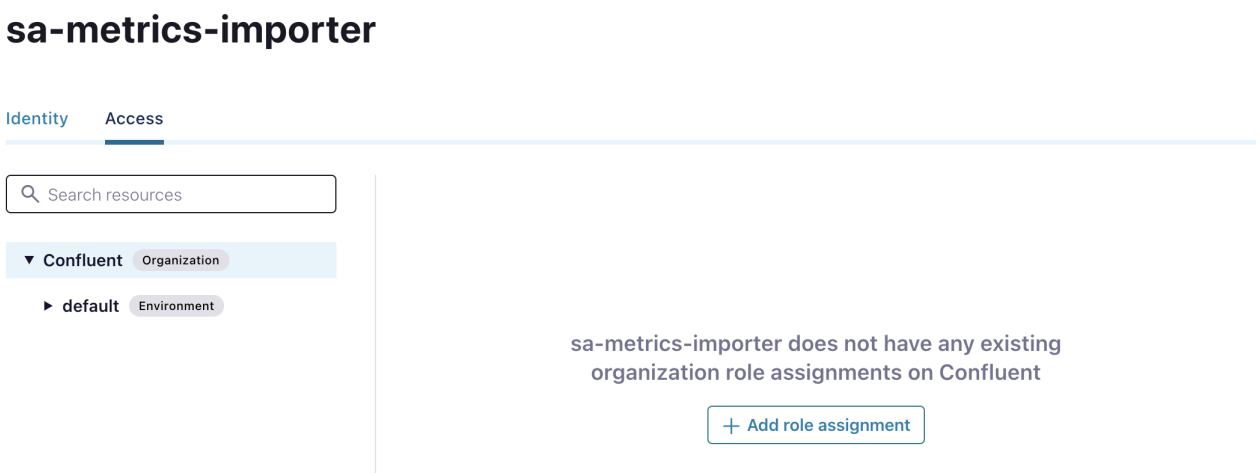
The `MetricsViewer` role provides AUTHORIZATION to the Metrics API for clusters in an organization. This role also enables service accounts to import metrics into third-party metrics platforms.

The previously created Service Account will authenticate to the Confluent Cloud using the Cloud API Key. However, it also needs its requests to be authorized, which can be achieved by adding the `MetricsViewer` role to the Service Account.

hitesh@catacouch.io

32. Open the administration menu (▯) in the upper-right corner of the Confluent Cloud user interface.
33. Click **Accounts & access** in the ADMINISTRATION section.
34. Now, click **Service accounts** button in the Accounts tab.
35. Select the Service Account **sa-metrics-importer**.
36. In the Service Account's details page, click the **Access** tab.
37. Here in the tree view, you can select the resource where you want the Service Account to have the `MetricsViewer` role. In this lab, you want the Service Account to view metrics of the entire organization. So, click on the top level of the tree view which should be the `Organization` level.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



38. Click **+ Add role assignment**.
39. Select the `MetricsViewer` tile and click **Save**.
40. When you return to **Accounts & access**, you can view the **Access** for the organization, and also see that the Service Account you created has the `MetricsViewer` role binding.

nitesh@datacouch.io

Discover Available Resources and Metrics using the Metrics API

41. Go back to the VM and open a Terminal window.
42. Now, create two Environment Variables for your Cloud API Key and Cloud API Secret, as you will use them several times. Run the following commands replacing `<cloud-api-key>` and `<cloud-api-secret>` with the credentials you obtained in Step 29.

```
$ export CLOUD_API_KEY=<cloud-api-key>
```

```
$ export CLOUD_API_SECRET=<cloud-api-secret>
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

43. Install the tool HTTPie (tool for making HTTP requests and interacting with APIs and web services). Run this script.
    Password = **training**

```
$ sudo ~/confluent-ccl/lab-audit-logs/install-httpie.sh
```

44. Discover available resources of the Metrics API by running this command:

```
$ http
'https://api.telemetry.confluent.cloud/v2/metrics/cloud/descriptors/r
esources' --auth "$CLOUD_API_KEY:$CLOUD_API_SECRET"

kafka
connector
ksql
schema_registry
```

45. Now, discover available metrics of Kafka clusters by running this command:

```
$ http
'https://api.telemetry.confluent.cloud/v2/metrics/cloud/descriptors/m
etrics?resource_type=kafka' --auth "$CLOUD_API_KEY:$CLOUD_API_SECRET"
```

    You should get a list of all metrics you can get for the Kafka brokers.

46. Here, discover available metrics of Connectors by running this command:

```
$ http
'https://api.telemetry.confluent.cloud/v2/metrics/cloud/descriptors/m
etrics?resource_type=connector' --auth
"$CLOUD_API_KEY:$CLOUD_API_SECRET"
```

    You should get a list of all metrics you can get for the connectors.

hitesh@datacouch.io

Using Queries with the Metrics API

The Confluent Cloud Metrics API has an expressive query language that allows users to flexibly filter and group timeseries data.

In this section you will create a query to get the number of requests per principal using the

Metrics API.

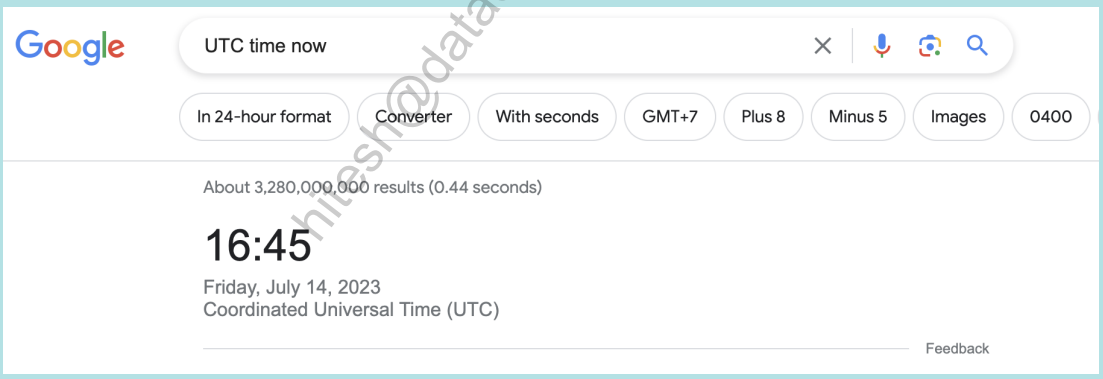
47. To perform a query you need to create the JSON payload of the **POST** request.
    In this example, you will use a JSON template file **~/confluent-ccl/lab-audit-logs/query.json**
    Open this file in VS Code by running this command:

```
$ cd ~/confluent-ccl/lab-audit-logs && code query.json
```

48. Modify the two highlighted values below in the **query.json** file.
    - Change **lkc-XXXXX** with your Kafka cluster ID
    - Change the date **2023-06-25** and the hours **00** with the current UTC time subtracting two hours

Timestamp Change Example

- Search in Google the current UTC Time:



- The timestamp in this example is: **2023-07-14T14:00:00**

hitesh@datacouch.io

```
{
    "aggregations": [
        {
            "metric": "io.confluent.kafka.server/request_count"
        }
    ],
    "filter": {
        "field": "resource.kafka.id",
        "op": "EQ",
        "value": "<mark>lkc-XXXXX</mark>"
    },
    "granularity": "PT1H",
    "group_by": [
        "metric.principal_id"
    ],
    "intervals": [
        "<mark>2023-06-25</mark>T<mark>00</mark>:00:00Z/PT1H"
    ]
}
```

- Press **Ctrl+S** to save the changes.

49. Take a look at the keys and values in the **query.json** file.

    - **aggregations** → **… .kafka.server/request_count**
        - **request_count** is a metric of the resource Kafka cluster (**kafka.server**)
    - **filter** → **resource.kafka.id** = **lkc-XXXXX**
    - **granularity** → **PT1H** which means "give metrics every 1 hour"
    - **group_by** → **metric.principal_id**. You want the number of requests **per** principal
    - **intervals** → **2023-06-25T00:00:00Z/PT1H**. It is the time range of observation
        - **/PT1H** specifies that the time interval is 1 hour from the timestamp provided

50. Run the following command to send the query to Confluent Cloud:

hitesh@datacouch.io

```
$ http 'https://api.telemetry.confluent.cloud/v2/metrics/cloud/query'
--auth "$CLOUD_API_KEY:$CLOUD_API_SECRET" < query.json
HTTP/1.1 200 OK
Connection: keep-alive
Content-Length: 177
content-type: application/json
date: Mon, 14 Jul 2023 16:47:07 GMT
server: envoy
x-envoy-upstream-service-time: 402

{
    "data": [
        {
            "metric.principal_id": "sa-z6owm3",
            "timestamp": "2023-07-14T14:00:00Z",
            "value": 204.0
        },
        {
            "metric.principal_id": "u-818y9r",
            "timestamp": "2023-07-14T14:00:00Z",
            "value": 33.0
        }
    ]
}
```

51. Feel free to modify the `granularity`, e.g. `PT15M`; and the `intervals`, e.g. `/PT2H`. So, you can view how the out changes.
52. You can see more examples of queries in this page (examples) if you want to test any other metric.

hitesh@datacouch.io

Clean up

53. Delete the Standard cluster you created in Step 2.

Conclusion

In this lab you have learned how the Audit Logs work in Confluent Cloud. Now, you know where Audit Logs are stored and how you can consume them using the Confluent CLI and Kafka consumers. You have also created an API Key to get access to the Audit Log cluster.

In the second part of this Lab, you have worked with the Metrics API. First, you have learned how to configure the authentication and authorization of a Service Account using a Cloud API Key and the `MetricsViewer` role. You have also performed multiple requests using the Metrics API to get the available metrics or the number of requests per principal.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

STOP HERE. THIS IS THE END OF THE EXERCISE.

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Lab 11 Real Use Case

a. Real Use Case in Confluent Cloud

The goal of this exercise is to set up an entire streaming architecture on Confluent Cloud for a vehicle monitoring application.

Objectives

- Use Kafka API Keys, Schema Registry API Keys, ACLs and RBAC to grant granular access to Confluent Cloud resources
- Create Service Accounts for components that programmatically access Confluent Cloud
- Create a fully-managed Postgres Source connector to import tables to Confluent Cloud using SMTs
- Configure Kafka producers to write events to Confluent Cloud topics
- Create a ksqlDB cluster to run queries for transforming and enriching the data
- Use Schema Linking to sync schemas between two Schema Registry clusters
- Use Cluster Linking to mirror a topic between two Kafka clusters

hitesh@datatouch.io

Step 1 - Create a new Environment, Kafka cluster, source Topics and ksqlDB cluster

Create a new Environment and enable Stream Governance

1. In Confluent Cloud website, go to **Environments**.
2. Click on **Add cloud environment** and name the new environment as **IoT-team**. Click, **Create**.
3. In the Stream Governance Packages, choose Essentials or Advanced package (either option works for this exercise). Click on **Begin configuration**.
4. Choose your preferred cloud provider and region. Then, click on **Enable**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Create a new Kafka cluster

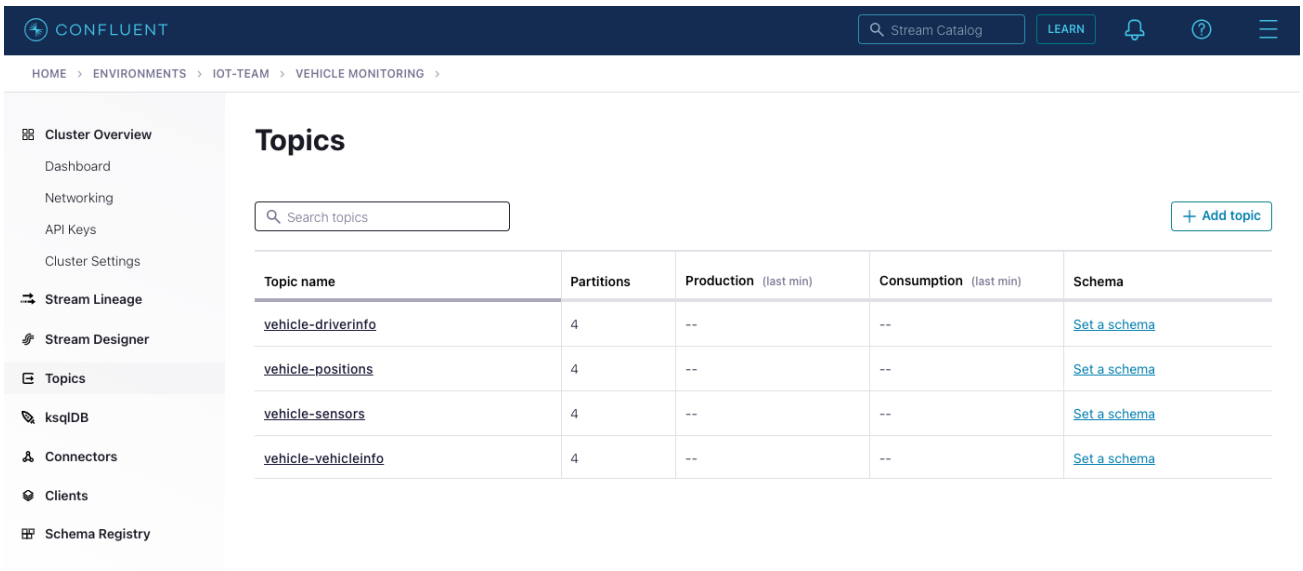
5. Inside the **IoT-team** environment, click on **Create cluster on my own**.
6. Choose an **AWS Basic cluster** in one of the following regions:
   - **AWS Oregon (us-west-2)**
   - **AWS Frankfurt (eu-central-1)**
   - **AWS Mumbai (ap-south-1)**
7. Choose a **Single zone** availability and click **Continue**.
8. Name the new cluster as **Vehicle Monitoring**. Then, click on **Launch cluster**.

Create Topics

9. Inside the **Vehicle Monitoring** cluster, go to **Topics** and create the following topics:
   a. *Topic name*: **vehicle-positions** — *Partitions*: **4**
   b. *Topic name*: **vehicle-sensors** — *Partitions*: **4**
   c. *Topic name*: **vehicle-vehicleinfo** — *Partitions*: **4**
   d. *Topic name*: **vehicle-driverinfo** — *Partitions*: **4**

hitesh@datacouch.io

Skip the definition of the Schema for the topics



© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Create ksqlDB cluster

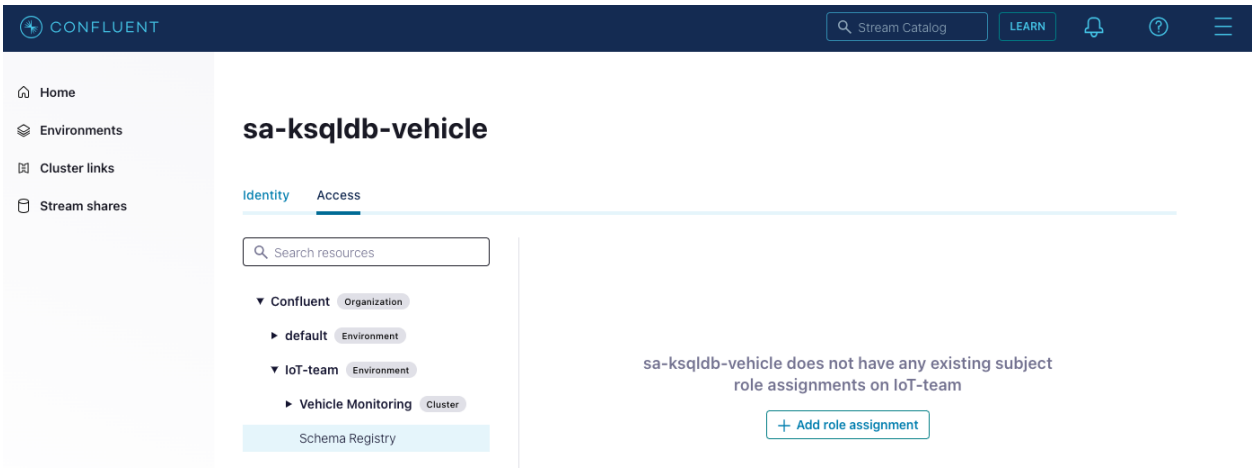
10. In the left pane of the **Vehicle Monitoring** cluster, click on **ksqlDB**.
11. Then, click on **Create cluster myself**.
12. Select **Granular access** and choose **Create a new one**:
    - New service account name: **sa-ksqldb-vehicle**
    - Description: **Service Account for the ksqlDB cluster for the Vehicle Monitoring application**
    - **Check the box** for "Adding all required ACLs" and click on **Continue**
13. Allow access to the topics that start by `vehicle-` using the prefix option. Click on **Add access by prefix**. Then:
    - Topic prefix: **vehicle-**
    - Allow: **Read**, **Write**, **Alter** and **Create**
    - Click on **Continue**
14. Set the name of the ksqlDB cluster to **vehicle_ksqlDB_cluster**.
15. Click on **Launch cluster!**.

hitesh@datacouch.io

Allow ksqlDB to access Schema Registry

16. Go to **Main Menu** (three lines icon in the upper-right corner) and click on **Accounts & access**.
17. In this window, select the **Service accounts** option in the **Accounts** tab.
18. Click on the Service Account: **sa-ksqldb-vehicle**
19. Now, click on the **Access** tab.
20. Drill down in the tree structure to select **Schema Registry** in the **IoT-team** environment.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



21. Click on **Add role assignment**. Then, choose the following options:
    - **Prefix rule** → Enter prefix: **vehicle-**
    - Role: **DeveloperWrite**
    - Click **Save**
22. Click again on **Add role assignment**. You also need to allow the ksqlDB application to manage the subjects in Schema Registry for their internal topics. Choose the following options:
    - **Prefix rule** → Enter prefix: **_confluent-ksql-pksqlc-**
    - Role: **DeveloperWrite**
    - Click **Save**

hitesh@datacouch.io

DeveloperWrite role can write in the subject, but also read in the subject. Check access for different roles in this link: [RBAC for Confluent Cloud Schema Registry]

Step 2 - Start Postgres database and create fully-managed Postgres Source connector

Start Postgres database

23. Go to your VM.
24. Open a Terminal window and run this command to change the directory:

```
$ cd ~/confluent-ccl/lab-real-use-case
```

25. First run the **enable-ssh.sh** shell script to enable SSH access to your VM. Use the password: ***training***

```
$ ./enable-ssh.sh
[sudo] password for training:
Generating public/private rsa key pair.
Created directory '/home/training/.ssh'.
Your identification has been saved in /home/training/.ssh/id_rsa
Your public key has been saved in /home/training/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:5MNyvkKd9mRdqXgEauQZDyhnRI5LylvWzJlAdhayS6U id_rsa
The key's randomart image is:
+---[RSA 4096]----+
|    .o.          |
|.  o=oo          |
|o.oEo   .        |
|oo... +          |
|++ o=++oS        |
|=.Oo=B*o+.       |
|.*+@ +..         |
|.=  *..          |
| .   .o          |
+----[SHA256]-----+
Done!

PUBLIC IP ADDRESS: 18.156.33.165
```

hitesh@datacouch.io

Keep the ***Public IP address*** of your VM as you will need it later to set up the Postgres connector.

26. Run the following command to start the Postgres database which will be deployed in a Docker container:

```
$ docker-compose up -d postgres
[+] Running 2/2
 ⠿ Network lab-real-use-case_default  Created
0.0s
 ⠿ Container postgres                 Started
0.4s
```

Inspect Postgres Tables

27. Inspect the contents of the tables by first connecting to the Postgres database:

```
$ psql -h postgres -U postgres
psql (12.15 (Ubuntu 12.15-0ubuntu0.20.04.1), server 11.2)
Type "help" for help.

postgres=#
```

28. At the postgres prompt use a SQL select statement to view the contents of the **driverinfo** table:

```
postgres=# select * from driverinfo;
id | driverid | firstname |  lastname   |    dob     |      phone
|            email            |         timestamp
----+----------+-----------+-------------+------------+
----------------+-------------------------+------+
----------------------------
  1 | 1        | Markku    | Lehtonen    | 1985-03-12 | +358-
123456789  | markku.lehtonen@email.com   | 2023-06-21 18:49:27.496685
  2 | 2        | Anna      | Virtanen    | 1992-07-28 | +358-
987654321  | anna.virtanen@email.com     | 2023-06-21 18:49:27.497574
...
```

nitesh@datatouch.io

29. Type the letter **q** to exit the view of the table.

30. Repeat the same operation to view the content of the vehicleinfo table:

```
postgres=# select * from vehicleinfo;
 id | vehicleid |     make      |    model    |  regdate   |  plate
|         timestamp
----+-----------+---------------+-------------+------------+
---------+----------------------------
  1 | 1         | MAN           | Lion's City | 2015-06-19 | ABC-123
| 2023-06-21 18:49:27.527198
  2 | 2         | MAN           | Lion's City | 2015-06-19 | DEF-456
| 2023-06-21 18:49:27.528405
...
```

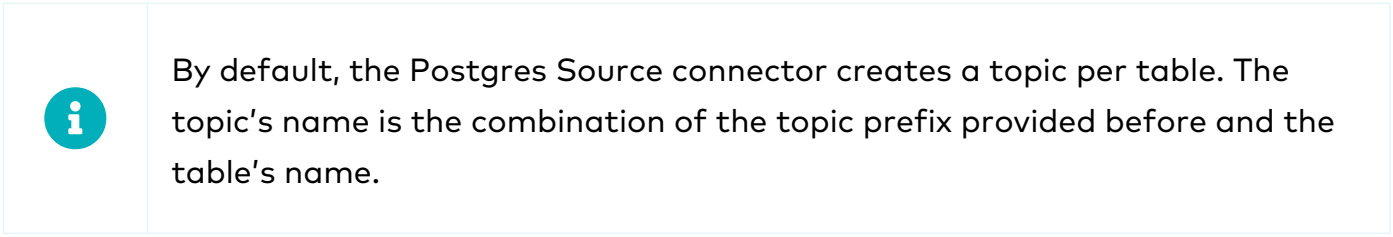
31. Type the letter **q** to exit the view of the table.

32. Press **Ctrl+D** to exit the psql client.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Create Postgres Source connector

33. In Confluent Cloud website, go to the **Vehicle Monitoring** cluster.
34. In the left pane, click on **Connectors**.
35. In the Search connectors box, search for **postgres**.
36. Click on the **Postgres Source** connector.
37. In *Topic selection*, define a topic prefix: `vehicle-` and click **Continue**.

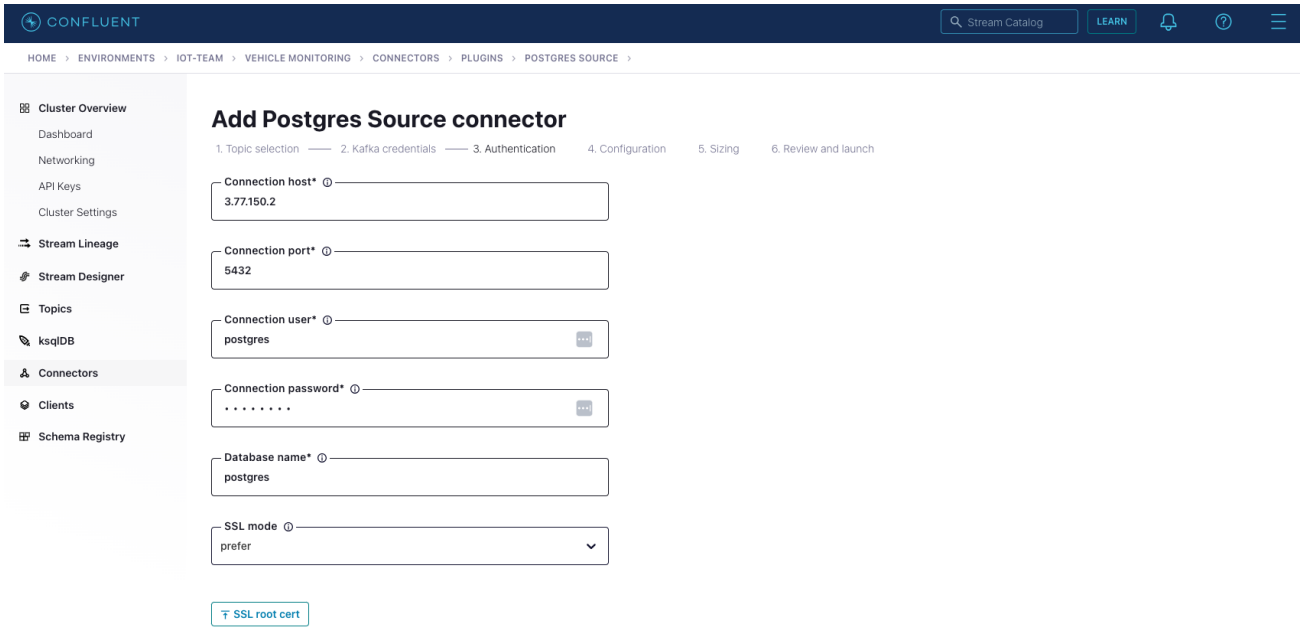


By default, the Postgres Source connector creates a topic per table. The topic's name is the combination of the topic prefix provided before and the table's name.

38. In *Kafka Credentials*, select **Granular access** and choose **Create a new one**:
    - New service account name: **sa-postgres-source-connector**
    - Description: **Service Account for the Postgres Source connector for the vehicle application**
    - **Check the box** for "Adding all required ACLs" and click on **Continue**
39. In *Authentication*, use the following settings and then click on **Continue**:
    - Connection host: **Use the IP address you got in step 25**
    - Connection port: **5432**
    - Connection user: **postgres**
    - Connection password: **postgres**
    - Database name: **postgres**
    - SSL mode: **prefer**

hiesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



40. In *Configuration*, select the following settings and then click on **Continue**:

    - Output record value format: **AVRO**
    - Table names: **driverinfo, vehicleinfo** (Use the comma to separate the two table names)
    - Table types: **TABLE**
    - Database timezone: **UTC**

        a. Click on **Show advanced configuration**:

            - Mode: **timestamp+incrementing**
            - Numeric Mapping: **best_fit**
            - Timestamp column name: **ts**
            - Incrementing column name: **id**

        b. Click on **Add SMT** to set the message's key.

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

By default, Postgres Source connector produces the messages to the topic without key.
In this application, we want the messages of the:

- `vehicle-driverinfo` topic to have as key the `driverid` value
- `vehicle-vehicleinfo` topic to have as key the `vehicleid` value

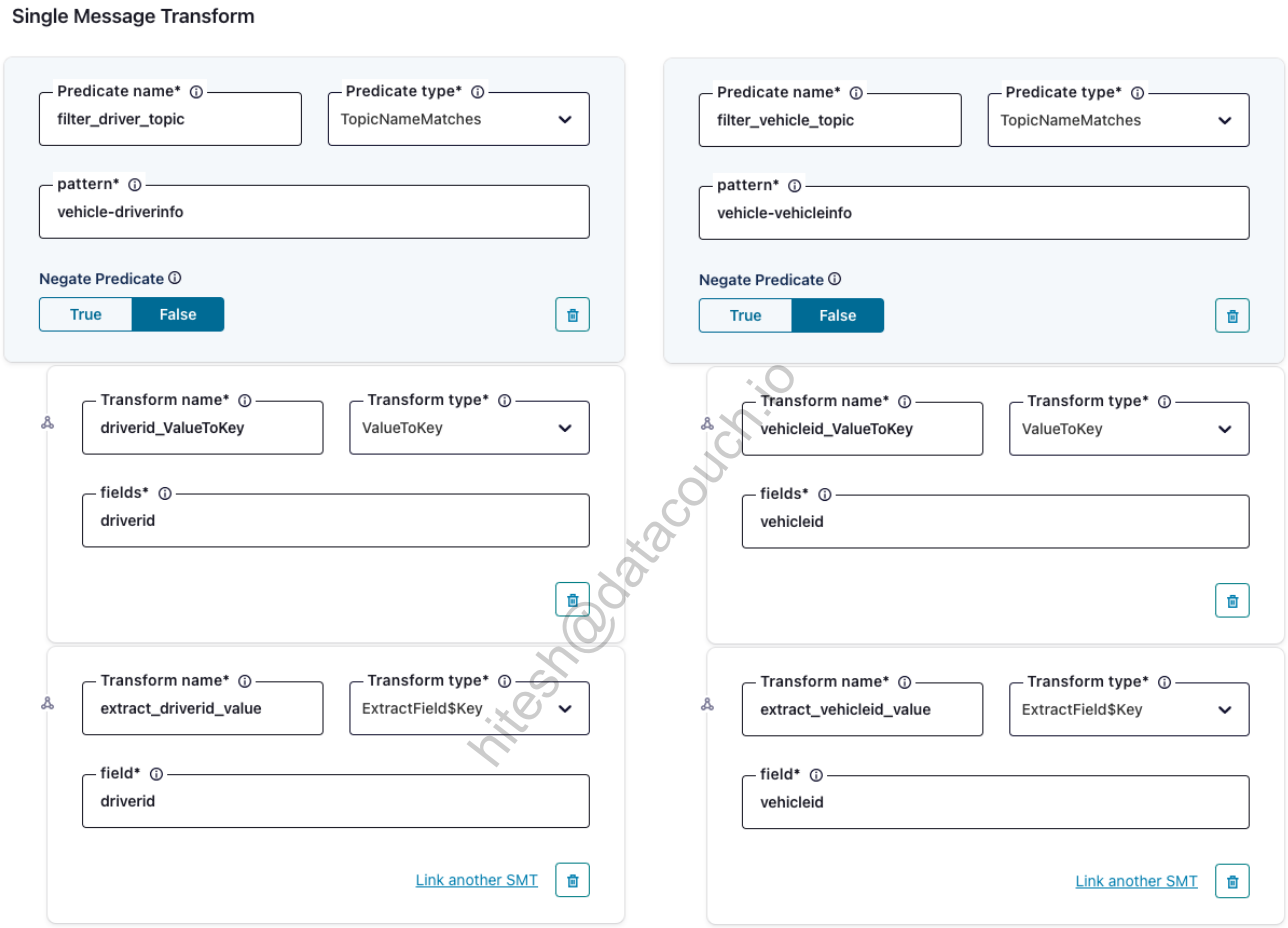
- First SMT:
  - Transform name: **driverid_ValueToKey**
  - Transform type: **ValueToKey**
  - fields: **driverid**
- Click on **Add Predicate**:
  - Predicate name: **filter_driver_topic**
  - Predicate type: **TopicNameMatches**
  - pattern: **vehicle-driverinfo**
- Click on **Link another SMT** under the First SMT:
  - Transform name: **extract_driverid_value**
  - Transform type: **ExtractField$Key**
  - field: **driverid**
- Click on **Add new SMT**:
  - Transform name: **vehicleid_ValueToKey**
  - Transform type: **ValueToKey**
  - fields: **vehicleid**
- Click on **Add Predicate**:
  - Predicate name: **filter_vehicle_topic**
  - Predicate type: **TopicNameMatches**
  - pattern: **vehicle-vehicleinfo**

htest@datatouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

- Click on **Link another SMT** under the **vehicleid_ValueToKey** SMT:
  - Transform name: **extract_vehicleid_value**
  - Transform type: **ExtractField$Key**
  - field: **vehicleid**

- Finally, click on **Continue**.



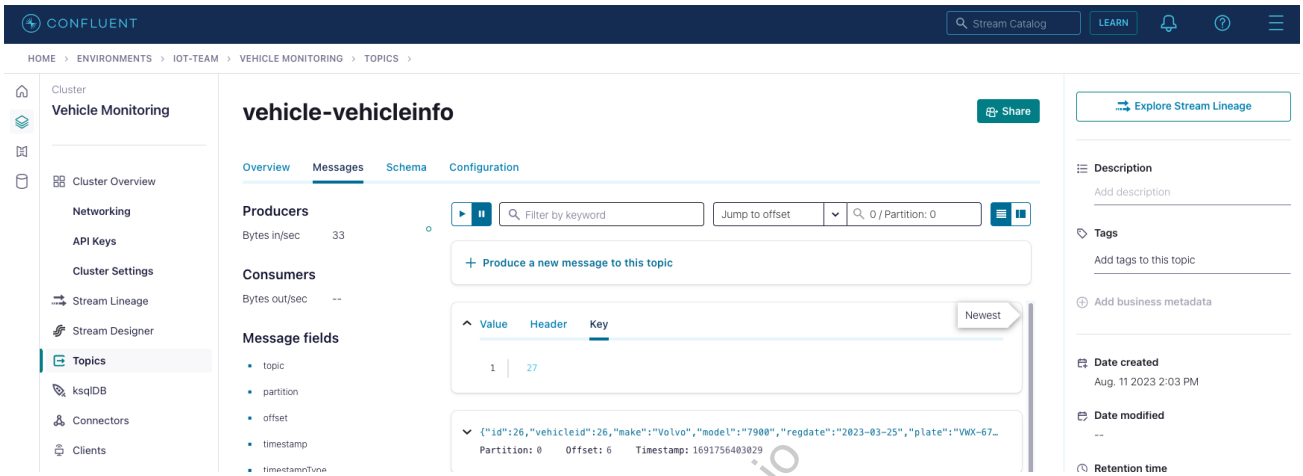
41. In *Sizing*, leave the number of tasks to 1 and click on **Continue**.
42. In *Review and launch*, set the name of the connector to **PostgresSourceConnector_Vehicle** and click on **Continue**.

Inspect Topics

43. Once the Postgres Source connector is running, go to **Topics**.
44. Click on **vehicle-driverinfo** topic.
45. Click on the **Messages** tab.

hitesh@datacouch.io

46. Type **0** in the offset box, then select: **0 / Partition: 0**
47. Click on one of the messages that appear in the window to view its value.
48. Click on the tab **Key** of the message to view its key, which should be the `driverid`.
49. Repeat the previous steps to inspect the topic **vehicle-vehicleinfo**.



Step 3 - Create the *producer.config* file and start the vehicle producers

hitesh@datacouch.io

All vehicle producers will use the same *producer.config* file to write data to Confluent Cloud. In this exercise there will be 8 vehicles in operation, each vehicle will produce two types of messages:

- Vehicle position information to the `vehicle-positions` topic
- Vehicle sensor information to the `vehicle-sensors` topic

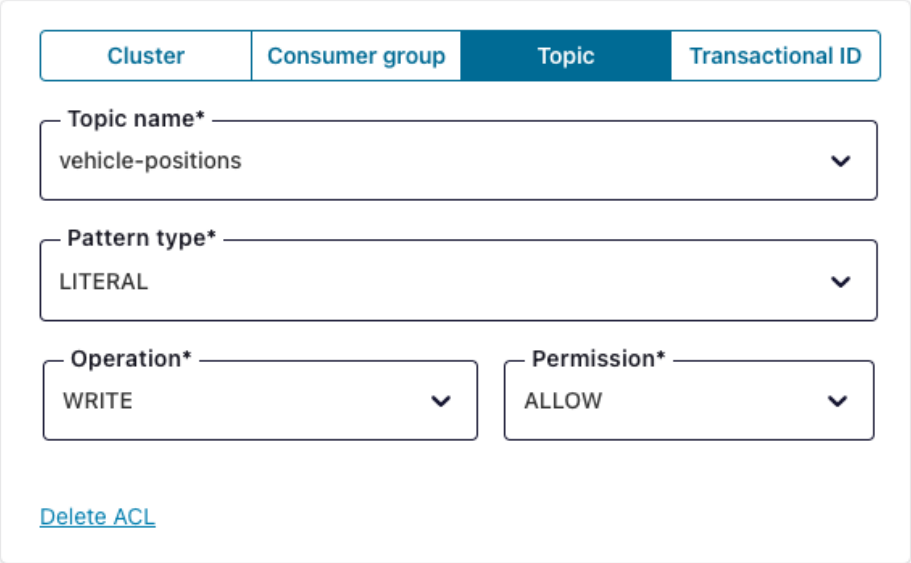
Create the producer Service Account and Kafka API Key

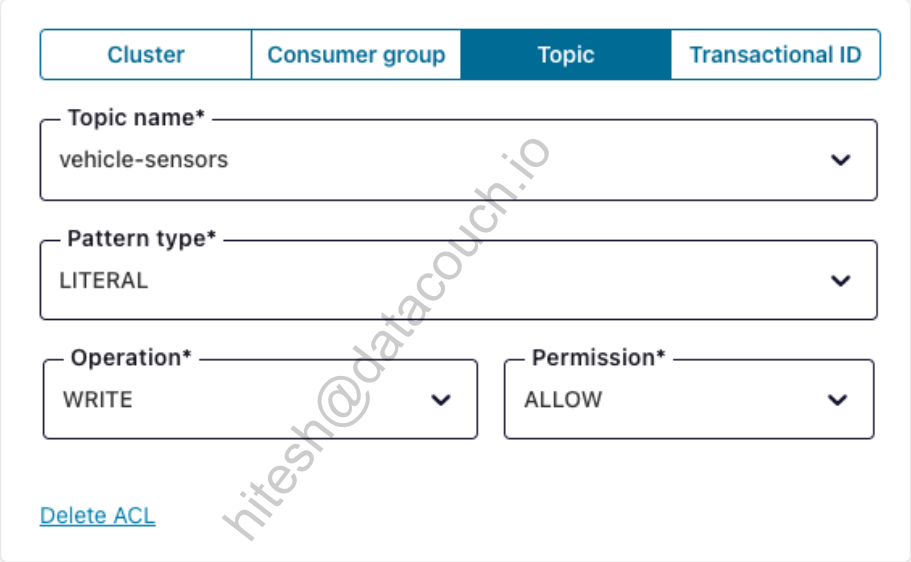
50. In Confluent Cloud website, go to the **Vehicle Monitoring** cluster.
51. In the left pane, click on **API Keys**.
52. Click on **Add key**.
53. Choose **Granular access** and click **Next**.
54. Now, select **Create a new one**:
    - New service account name: **sa-vehicle-producers**

- Description: **Service Account for the vehicle producers**
- Click on **Next**.

55. To allow the producers to write data to `vehicle-positions` and `vehicle-sensors`, we need to create two ACLs:
    - First ACL:
        i. Resource: `Topic`
        ii. Topic name: `vehicle-positions`
        iii. Pattern type: `LITERAL`
        iv. Operation: `WRITE`
        v. Permission: `ALLOW`
    - Click **Add ACLs** to create a second ACL:
        i. Resource: `Topic`
        ii. Topic name: `vehicle-sensors`
        iii. Pattern type: `LITERAL`
        iv. Operation: `WRITE`
        v. Permission: `ALLOW`
    - Once you complete all the fields with these values, click **Next**.

hitesh@datacouch.io





hitesh@datacouch.io

56. Give a **Description** to your new Kafka cluster API key/secret pair to be able to recognize it in the future. Then, click on **Download and continue** to store your API key/secret locally.

Create the Schema Registry API Key

Let's create a Schema Registry API Key owned by the Service Account **sa-vehicle-producers**. To do that, you need to use the Confluent CLI.

57. Go to your VM and open a Terminal window.
58. Login to your Confluent Cloud account using the Confluent CLI:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ confluent login
Enter your Confluent Cloud credentials:
Email: <enter email>
Password: <enter password>
```

59. Select the **IoT-team** environment in the Confluent CLI by listing the environments and then using the **IoT-team** environment ID:

```
$ confluent environment list
  Current |     ID     |   Name
----------+------------+-----------
  *       | <env-ID-1> | default
          | <env-ID-2> | IoT-team
```

```
$ confluent environment use <env-ID-2>
Now using "<env-ID-2>" as the default (active) environment.
```

60. Now, describe the Schema Registry cluster in the **IoT-team** environment to view its Cluster ID. Run this command:

hitesh@datacouch.io

```
$ confluent schema-registry cluster describe
+-------------------------
+-------------------------------------------------+
| Name                    | Stream Governance Package
|
| Cluster                 | <schema-registry-ID-1>
|
| Endpoint URL            | <schema-registry-URL-1>
|
| Used Schemas            |
2 |
| Available Schemas       |
19998 |
| Free Schemas Limit      |
20000 |
| Global Compatibility    | <Requires API Key>
|
| Mode                    | <Requires API Key>
|
| Service Provider        | aws
|
| Service Provider Region | eu-central-1
|
| Package                 | advanced
|
+-------------------------
+-------------------------------------------------+
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

61. You also need to know the Service Account ID of **sa-vehicle-producers**. With the Confluent CLI, you can list all your Service Accounts running the following command:

```
$ confluent iam service-account list
     ID     |               Name                |          Description
------------+-----------------------------------+
-----------------------------------
  <sa-ID-1> | sa-ksqldb-vehicle                 | Service Account for
the
            |                                   | ksqlDB cluster for the
Vehicle
            |                                   | Monitoring application
  <sa-ID-2> | sa-vehicle-producers              | Service Account for
the
            |                                   | vehicle producers
  <sa-ID-3> | sa-postgres-source-connector      | Service Account for
the
            |                                   | Postgres Source
connector for
            |                                   | the vehicle
application
```

62. Create the Schema Registry API Key owned by **sa-vehicle-producers**. Run this command using your Schema Registry ID and Service Account ID:

```
$ confluent api-key create --service-account <sa-ID-2> --resource
<schema-registry-ID-1> --description "API Key for the vehicle
producers"
It may take a couple of minutes for the API key to be ready.
Save the API key and secret. The secret is not retrievable later.
+------------
+-----------------------------------------------------------------+
| API Key    | <API-Key>
|
| API Secret | <API-Secret>
|
+------------
+-----------------------------------------------------------------+
```

Copy and paste the API Key/Secret as you will need it in the next section to create the *producer.config* file.

hitesh@dataroutch.io

Apply *DeveloperWrite* role to operate in Schema Registry

63. Here, you are going to use the Confluent CLI to apply a role-binding. Run the following command to apply the *DeveloperWrite* role to the Service Account **sa-vehicle-producers**, so it will be able to write and read schemas in subjects that start with `vehicle-`:

```
$ confluent iam rbac role-binding create --principal User:<sa-ID-2>
--role DeveloperWrite --environment <env-ID-2> --schema-registry
-cluster <schema-registry-ID-1> --resource "Subject:vehicle-*"
+---------------+----------------+
| Principal     | User:<sa-ID-2> |
| Role          | DeveloperWrite |
| Resource Type | Subject        |
| Name          | vehicle-*      |
| Pattern Type  | LITERAL        |
+---------------+----------------+
```

Create the *producer.config* file

64. In the VM, open the template file `producer.config` in Visual Studio Code:

```
$ code ~/confluent-ccl/lab-real-use-case/producer.config
```

65. Replace the placeholders with the following values:

    - **{{ BOOTSTRAP_SERVERS }}** is found in text file downloaded for the Kafka API Key in Step 56
    - **{{ CLUSTER_API_KEY }}** is found in text file downloaded for the Kafka API Key in Step 56
    - **{{ CLUSTER_API_SECRET }}** is found in text file downloaded for the Kafka API Key in Step 56
    - **{{ SCHEMA_REGISTRY_URL }}** is found in the output in Step 60
    - **{{ SR_API_KEY }}** is found in the output in Step 62
    - **{{ SR_API_SECRET }}** is found in the output in Step 62

nitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

`producer.config` *sample*

```
# Required connection configs for Kafka producer, consumer, and admin
bootstrap.servers=pkc-zajg0.eu-central-1.aws.confluent.cloud:9092
security.protocol=SASL_SSL
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginMod
ule required username='PFH5EA07UFHLWSLA'
password='80LuD1J+zMtVVw8e4XF7BoIOc1Ng71y3FTG8Zz0In72e5Bu8YfaXaNIJLYw
QDVlp';
sasl.mechanism=PLAIN

# Required connection configs for Confluent Cloud Schema Registry
schema.registry.url=https://psrc-v9kpz.eu-central-
1.aws.confluent.cloud
basic.auth.credentials.source=USER_INFO
basic.auth.user.info=DRAK7PNSP7POJGTA:VBgjGSX8Zf2alDCk2xhhY1lidgVOw8P
211/IocNajlDkpYPeDxqyklZ+ZmI2joft
```

66. Press **Ctrl+S** to save the file and close Visual Studio Code.

Start the vehicle producers

67. In the Terminal window, make sure your working directory is the `~/confluent-ccl/lab-real-use-case/`. Run:

```
$ cd ~/confluent-ccl/lab-real-use-case
```

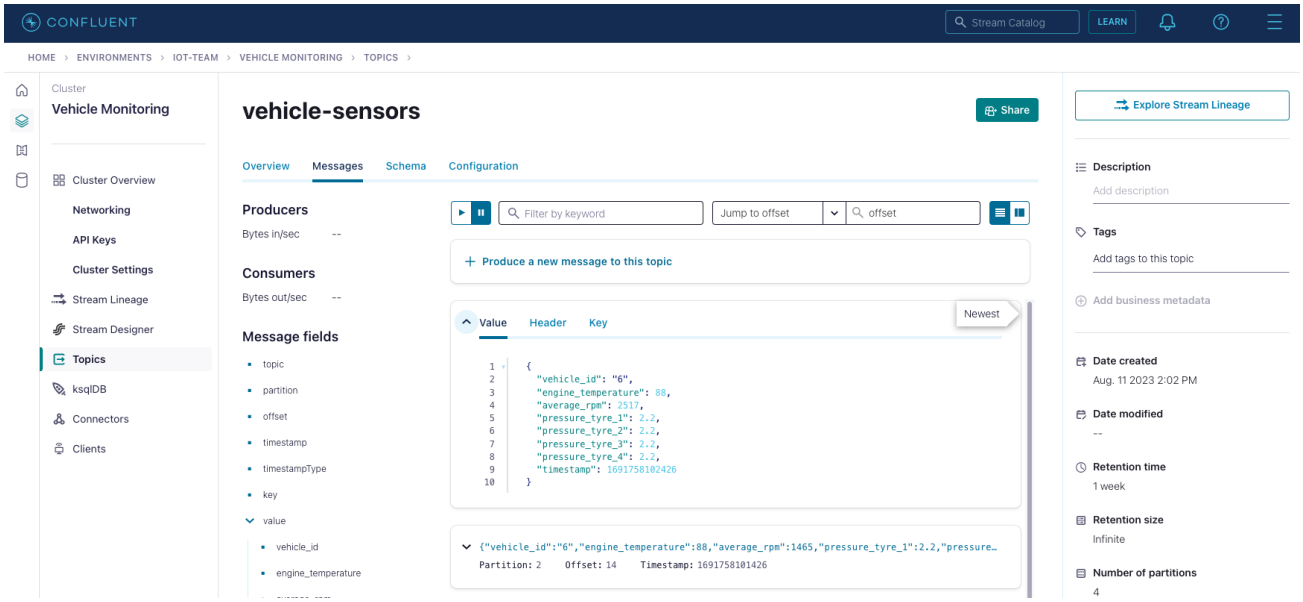
68. Use docker-compose to start your producers. Run the following command:

hitesh@datacouch.io

```
$ docker-compose up -d vehicle1 vehicle2 vehicle3 vehicle4 vehicle5
vehicle6 vehicle7 vehicle8
[+] Running 8/8
 ⠿ Container vehicle6  Started
1.3s
 ⠿ Container vehicle1  Started
0.8s
 ⠿ Container vehicle5  Started
1.5s
 ⠿ Container vehicle8  Started
1.6s
 ⠿ Container vehicle3  Started
1.7s
 ⠿ Container vehicle2  Started
1.3s
 ⠿ Container vehicle7  Started
1.1s
 ⠿ Container vehicle4  Started
1.5s
```

Inspect Topics

69. In Confluent Cloud website, go to **Topics** in the cluster **Vehicle Monitoring**.
70. Click on **vehicle-sensors** topic.
71. Click on the **Messages** tab and then you should view the messages that are currently being produced.
72. Check that topic **vehicle-positions** is also receiving messages.



hitesh@datatouch.io

Step 4 - Use ksqlDB to transform and enrich the data

Create the ksqlDB API Key

You can use the ksqlDB web-based editor to build and run queries. However, you will use the REST API to connect to the ksqlDB cluster.

73. First, login to your Confluent Cloud account using the Confluent CLI:

```
$ confluent login
Enter your Confluent Cloud credentials:
Email: <enter email>
Password: <enter password>
```

74. List out the ksqlDB clusters:

```
$ confluent ksql cluster list
      ID      |          Name          | Topic Prefix | ... |
Endpoint     |   Status
-------------+------------------------+--------------+-----+
------------------+------------
  <ksqlDB-ID> | vehicle_ksqlDB_cluster | pksqlc-7g9j2 | ... |
<ksqlDB_Endpoint> | PROVISIONED
```

whitesh@datacouch.io

Make a note of the **<ksqlDB-ID>** and **<ksqlDB_Endpoint>**, as you will use them later.

75. Run the following command to create a ksqlDB API Key:

```
$ confluent api-key create --resource <ksqlDB-ID>
It may take a couple of minutes for the API key to be ready.
Save the API key and secret. The secret is not retrievable later.
+------------
+-------------------------------------------------------------------+
| API Key    | <ksqlDB-API-Key>
|
| API Secret | <ksqlDB-API-Secret>
|
+------------
+-------------------------------------------------------------------+
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

76. Now, create three Environment variables to store this information. Run the following commands replacing with your own values:

```
$ export KSQLDB_API_KEY=<ksqlDB-API-Key>
```

```
$ export KSQLDB_API_SECRET=<ksqlDB-API-Secret>
```

```
$ export KSQLDB_ENDPOINT=<ksqlDB_Endpoint>
```

Run ksqlDB queries using the REST API

77. First, `cd` to the lab directory. Run this:

```
$ cd ~/confluent-ccl/lab-real-use-case
```

78. Run the following three commands using `curl` to run multiple KSQL queries written in JSON files:

    - First command:

```
$ curl -u $KSQLDB_API_KEY:$KSQLDB_API_SECRET $KSQLDB_ENDPOINT/ksql
-d @queries-1.json | jq
```

    - Second command:

```
$ curl -u $KSQLDB_API_KEY:$KSQLDB_API_SECRET $KSQLDB_ENDPOINT/ksql
-d @queries-2.json | jq
```

    - Third command:

```
$ curl -u $KSQLDB_API_KEY:$KSQLDB_API_SECRET $KSQLDB_ENDPOINT/ksql
-d @queries-3.json | jq
```

hitesh@datacoutch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

If you are curious, open Visual Code to view the queries you've just executed. Run:

```
$ code .
```

Inspect the Streams and Tables

The queries executed before were used to create Stream and Tables to get the following outputs:

- A Table showing who is the current driver's name for each vehicle
- A Table indicating how many seconds a driver has exceeded the urban speed limit (80 km/h)
- A Stream containing real-time messages with all the available information about each vehicle, which will be used later for the "Data Analysis" team.

hitesh@dancouch.io

79. In Confluent Cloud website, go to the **Vehicle Monitoring** cluster.
80. In the left pane, click on **ksqlDB** and then on the **vehicle_ksqlDB_cluster**.
81. In the Editor box, run the following Pull query to view who is the current driver of VehiculeKey "23":

```
<strong>SELECT
  plate,
  firstname,
  lastname
FROM vehicle_info_driver_info_table
  WHERE vehiclekey = '23';</strong>
```

Result:

```
{
  "PLATE": "NOP-789",
  "FIRSTNAME": "Hanna",
  "LASTNAME": "Lindholm"
}
```

82. Now, check for how many seconds drivers have been exceeding the speed limit by running

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

this command:

Change the `auto.offset.reset` property to `Earliest` before running the query.

```
<strong>SELECT
  *
FROM vehicle_driver_speed_exceeded_count
EMIT CHANGES;</strong>
```

Result:

```
{"DRIVER_ID":"3","FIRSTNAME":"Mikko","LASTNAME":"Nieminen","COUNT":30
}
{"DRIVER_ID":"18","FIRSTNAME":"Laura","LASTNAME":"Saari","COUNT":102}
{"DRIVER_ID":"21","FIRSTNAME":"Juha","LASTNAME":"Aaltonen","COUNT":91
}
{"DRIVER_ID":"17","FIRSTNAME":"Ville","LASTNAME":"Turunen","COUNT":35
}
...
```

hitesh@datacouch.io

83. Click on the **Stop** button to stop running the query.

84. Run the following query to see how the events look like in the Stream that contains all available information about each vehicle:

```
<strong>SELECT
  *
FROM vehicle_all_data_stream
  LIMIT 1;</strong>
```

Result:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
{
  "VEHICLEKEY": "17",
  "VEHICLE_ID": "17",
  "DRIVER_ID": "9",
  "LATITUDE": 60.19506,
  "LONGITUDE": 24.955608,
  "SPEED": 4.17,
  "ACCELERATION": -0.62,
  "ODOMETER": 16176,
  "ENGINE_TEMPERATURE": 96,
  "AVERAGE_RPM": 2200,
  "PRESSURE_TYRE_1": 2.4,
  "PRESSURE_TYRE_2": 2.4,
  "PRESSURE_TYRE_3": 2.4,
  "PRESSURE_TYRE_4": 2.4,
  "MAKE": "Mercedes-Benz",
  "MODEL": "Citaro",
  "PLATE": "VWX-901",
  "REGISTRATION_DATE": "2019-05-17",
  "TIMESTAMP": 1687477683468
}
```

hitesh@datatouch.io

Step 5 - Share the data with the "Data Analysis" team

In our "fake" company, the data analysis team is located in another region and they also have their own Confluent Cloud Environment and Kafka clusters to perform their data analysis.

The objective of this section is to sync the topic **`vehicle-all-data`** and its Schema Registry subject **`vehicle-all-data-value`** to the data analysis team clusters.

Thus, let's create a new Environment and Kafka cluster:

85. Create a new Environment called **Data-Analysis-team**.
86. Regarding the Stream Governance Packages, choose Essentials or Advanced package (either option works for this exercise). Click on **Begin configuration** and choose your preferred cloud provider and region. Then, click on **Enable**.
87. Create a new **DEDICATED** cluster in the **Data-Analysis-team** environment. Click on **Create cluster on my own**. Then, follow these instructions:
    a. Cluster type: Go to the **Dedicated** cluster and select **1 CKU** by dragging the slider. Then click on **Begin Configuration**.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

b. Region/zones: Choose your preferred Cloud Provider and Region. Select **Single zone**. Click on **Continue**.

c. Networking: Select **Internet**, then click on **Continue**.

d. Security: Select **Automatic**, then click on **Continue**.

e. Set payment: Click on **Skip payment**.

f. Review and launch: Set the cluster name to **Vehicle Analysis**. Review and click on **Launch cluster**.

Step 5a - Sync the Schema using Schema Linking

Since the Schema Linking Exporter is located in the Source cluster, the exporter will need the following information to access the Destination Schema Registry:

- *Destination SR_URL*
- *Destination SR_API_KEY*
- *Destination SR_API_SECRET*

hitesh@datacouch.io

Create Schema Registry API Key

At this point, you need to create two things:

- A **Service Account** representing the Schema Registry Exporter of the **IoT-team** environment
- An **API Key/Secret** owned by the new Service Account to access the Schema Registry cluster of the **Data-Analysis-team** environment

88. Go to your VM and open a Terminal window.
89. Login to your Confluent Cloud account using the Confluent CLI, if you are not already logged in:

```
$ confluent login
Enter your Confluent Cloud credentials:
Email: <enter email>
Password: <enter password>
```

90. Select the **Data-Analysis-team** environment in the Confluent CLI by listing the

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

environments and then using the **Data-Analysis-team** environment ID:

```
$ confluent environment list
  Current |     ID     |        Name
----------+------------+---------------------
          | <env-ID-3> | Data-Analysis-team
          | <env-ID-1> | default
  *       | <env-ID-2> | IoT-team
```

```
$ confluent environment use <env-ID-3>
Now using "<env-ID-3>" as the default (active) environment.
```

91. Now, describe the Schema Registry cluster in the **Data-Analysis-team** environment to view its Cluster ID. Run this command:

```
$ confluent schema-registry cluster describe
+------------------------
+-----------------------------------------------+
| Name                     | Stream Governance Package
|
| Cluster                  | <schema-registry-ID-2>
|
| Endpoint URL             | <schema-registry-URL-2>
|
| Used Schemas             |
0 |
| Available Schemas        |
1000 |
| Free Schemas Limit       |
1000 |
| Global Compatibility     | <Requires API Key>
|
| Mode                     | <Requires API Key>
|
| Service Provider         | gcp
|
| Service Provider Region | us-central1
|
| Package                  | essentials
|
+------------------------
+-----------------------------------------------------+
```

hitesh@datacouch.io

92. Let's create the new Service Account to represent the Exporter:

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

```
$ confluent iam service-account create sa-iot-exporter --description
"Service Account for the Schema Registry Exporter of the IoT team"
+-------------+---------------------------------+
| ID          | <sa-ID-4>                       |
| Name        | sa-iot-exporter                 |
| Description | Service Account for the Schema  |
|             | Registry Exporter of the IoT    |
|             | team                            |
+-------------+---------------------------------+
```

93. Create the Schema Registry API Key owned by **sa-iot-exporter**. Run this command using your Schema Registry ID and Service Account ID:

```
$ confluent api-key create --service-account <sa-ID-4> --resource
<schema-registry-ID-2> --description "API Key for the IoT team
Exporter"
It may take a couple of minutes for the API key to be ready.
Save the API key and secret. The secret is not retrievable later.
+------------
+-------------------------------------------------------------------+
| API Key    | <schema-registry-API-Key-dest>
|
| API Secret | <schema-registry-API-Secret-dest>
|
+------------
+-------------------------------------------------------------------+
```

hitesh@datacouch.io

94. Open the configuration file `dest-SR.config` in VS Code:

```
$ code ~/confluent-ccl/lab-real-use-case/dest-SR.config
```

- Replace `<schema-registry-URL>` with the Schema Registry Endpoint of Step 91
- Replace `<schema-registry-API-Key>` with the API Key you have just generated
- Replace `<schema-registry-API-Secret>` with the API Secret you have just generated

```
schema.registry.url=<schema-registry-URL>
basic.auth.credentials.source=USER_INFO
basic.auth.user.info=<schema-registry-API-Key>:<schema-registry-
API-Secret>
```

Press `Ctrl+S` to save the changes.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Sample

```
schema.registry.url=https://psrc-x77pq.us-
central1.gcp.confluent.cloud
basic.auth.credentials.source=USER_INFO
basic.auth.user.info=QYZSOSFMQNTTTW3C:inUhDqXiy4AaOLx5TML1loPoR
vodFxdAH9g6YR2RBL05hkQaMYNAuAxHpeueGteq
```

95. You also need to provide authorization to the new Service Account to allow the Exporter to manage the subjects in the Destination Schema Registry.
Run the following command to apply the *ResourceOwner* role to the Service Account **sa-iot-exporter**:

```
$ confluent iam rbac role-binding create --principal User:<sa-ID-4>
--role ResourceOwner --environment <env-ID-3> --schema-registry
-cluster <schema-registry-ID-2> --resource "Subject:*"
+---------------+----------------+
| Principal     | User:<sa-ID-4> |
| Role          | ResourceOwner  |
| Resource Type | Subject        |
| Name          | *              |
| Pattern Type  | LITERAL        |
+---------------+----------------+
```

hitesh@datacouch.io

Start the Schema Registry Exporter in the *IoT team* Environment

96. Use the Confluent CLI to create the exporter called `vehicle-data-exporter` to sync the subject `vehicle-all-data-value`. Run this command:

Note that the `<env-ID-2>` is the ID of the **IoT-team** environment.

```
$ confluent schema-registry exporter create vehicle-data-exporter
--environment <env-ID-2> --subjects "vehicle-all-data-value"
--context-type NONE --config-file dest-SR.config
Created schema exporter "vehicle-data-exporter".
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

If you run the previous command and you are asked to introduce an API Key/Secret. Then, follow these instructions.

- Run the following command to create an API Key and to store it in your Confluent CLI:

Note that the **<schema-registry-ID-1>** is the ID of the **IoT-team** Schema Registry cluster in the Step 60.

```
$ confluent api-key create --resource <schema-registry-ID-1>
--description "API Key for me"
It may take a couple of minutes for the API key to be ready.
Save the API key and secret. The secret is not retrievable
later.
+------------
+-------------------------------------------------------------
----+
| API Key    | <schema-registry-API-Key-source>
|
| API Secret | <schema-registry-API-Secret-source>
|
+------------
+-------------------------------------------------------------
----+
```

nitesh@datacouch.io

- Use the Confluent CLI to create the exporter called **vehicle-data-exporter** to sync the subject **vehicle-all-data-value**. Run this command:

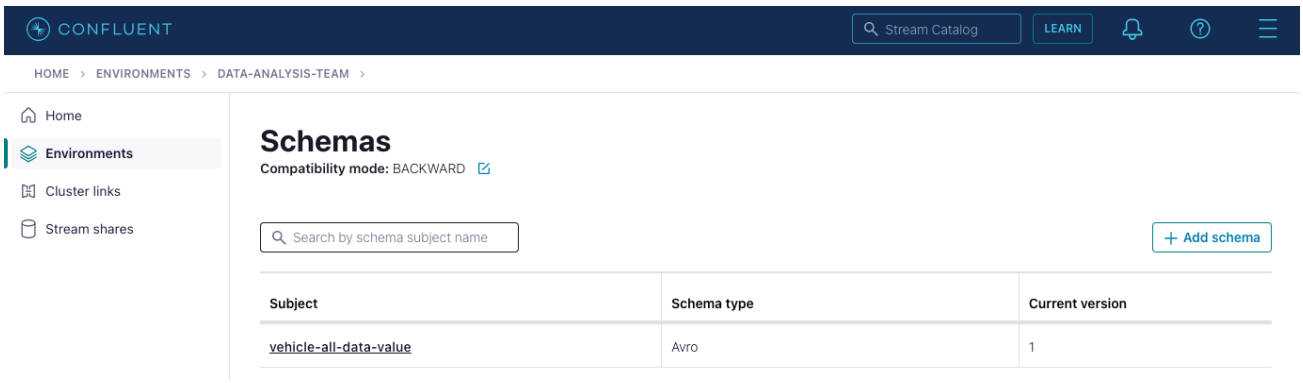
Note that the **<env-ID-2>** is the ID of the **IoT-team** environment.

```
$ confluent schema-registry exporter create vehicle-data-
exporter --environment <env-ID-2> --subjects "vehicle-all-data-
value" --context-type NONE --config-file dest-SR.config
Enter your Schema Registry API key: <schema-registry-API-Key-
source>
Enter your Schema Registry API secret: <schema-registry-API-
Secret-source>
Created schema exporter "vehicle-data-exporter".
```

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.

Check *vehicle-all-data-value* subject was created

98. In Confluent Cloud website, go to the **Data-Analysis-team** environment.
99. In the right pane, click on **View & manage** under **Schemas**.
100. You should see the synced subject **vehicle-all-data-value**.



Step 5a - Mirror the *vehicle-all-data* topic using Cluster Linking

Create the Cluster Link using the Confluent Cloud UI

Here, you are going to set up the Cluster Link using the Confluent Cloud UI:

101. In the left pane, click on **Cluster links**.
102. Click on **Create cluster link**.
103. In the *Source cluster* configuration, use these settings and then click **Continue**:
    - Source cluster: **Confluent Cloud (in my org)**
    - Environment: **IoT-team**
    - Source cluster: **Vehicle Monitoring**
    - Security access: **Granular access**
        a. Add first ACL:
            - Resource: `Topic`
            - Topic name: `vehicle-all-data`
            - Pattern type: `LITERAL`
            - Operation: `DESCRIBE_CONFIGS`
            - Permission: `ALLOW`

hitesh@datascouch.io

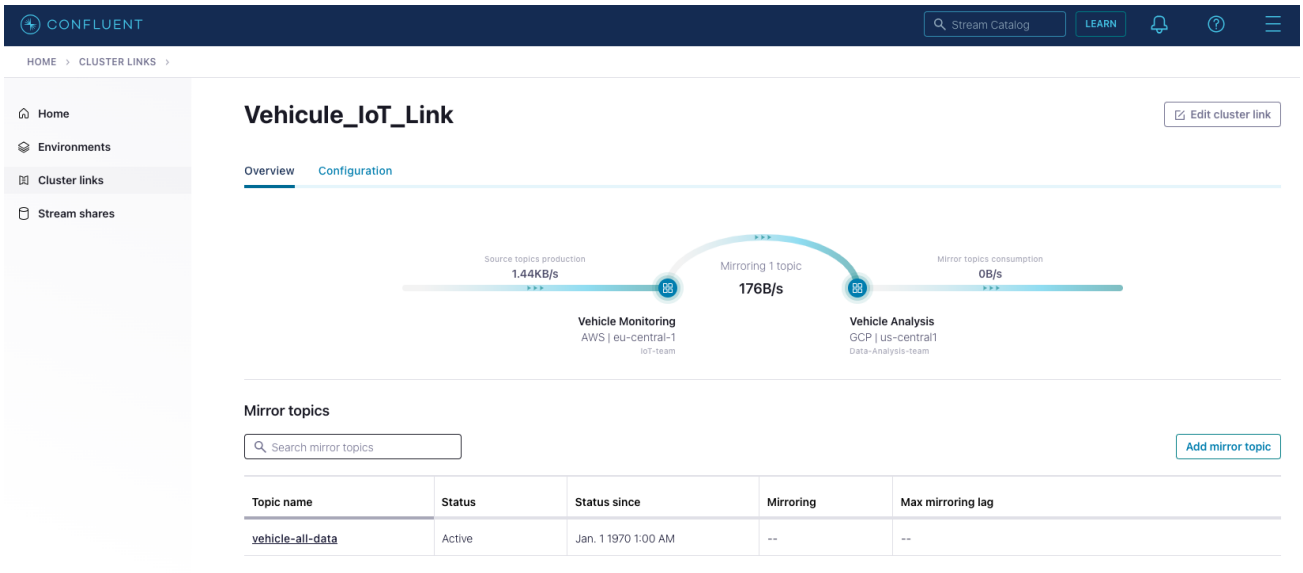
b. Add second ACL:

- Resource: `Topic`
- Topic name: `vehicle-all-data`
- Pattern type: `LITERAL`
- Operation: `READ`
- Permission: `ALLOW`

104. In the *Destination cluster* configuration, use these settings and then click **Continue**:
    - Environment: **Data-Analysis-team**
    - Destination cluster: **Vehicle Analysis**
105. In the *Configurations* section, use these settings and then click **Continue**:
    - Auto-create mirror topics: **Off**
    - Add prefix to mirror topics: **Off**
    - Sync consumer offsets: **Off**
    - Sync access control lists (ACLs): **Off**
106. In the *Review & launch* section:
    - Cluster link name: **Vehicule_IoT_Link**
107. Click on **Launch cluster link**.
108. Now that you have the Cluster Link running, click on **Add mirror topic**.
109. Select the topic: `vehicle-all-data`. Then, click on **Add**.

hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



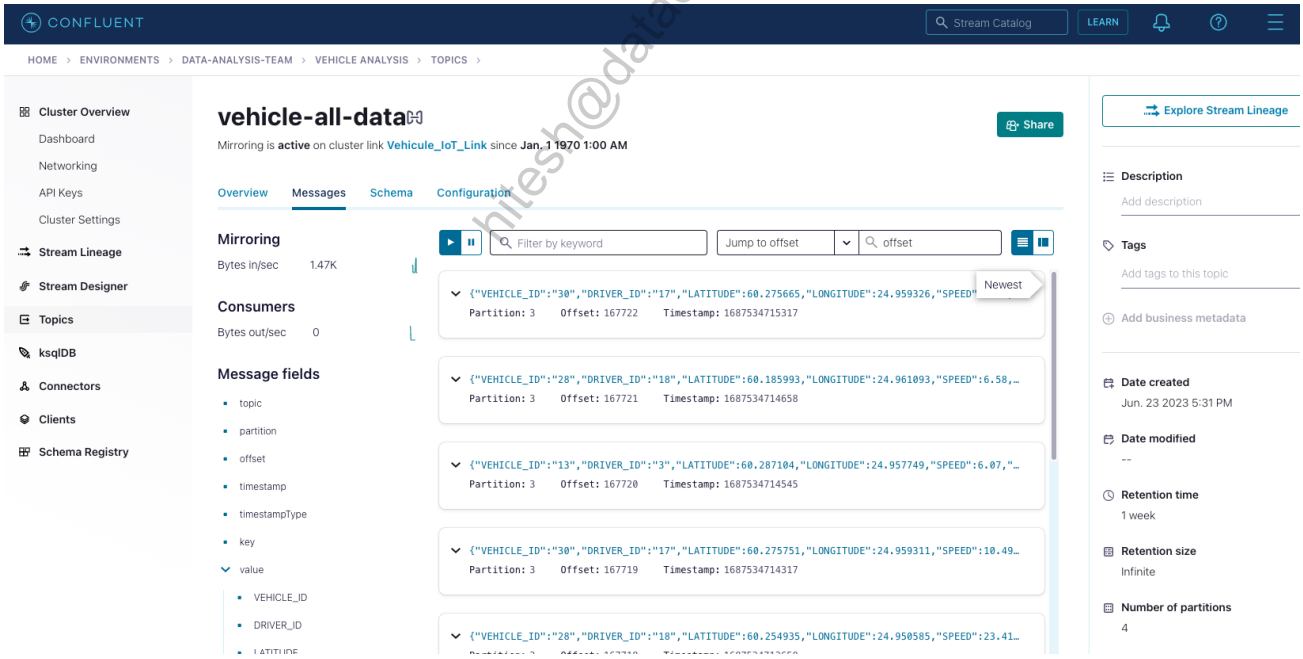
Inspect the mirror topic

110. Go to the **Vehicle Analysis** cluster under the **Data-Analysis-team** environment.

111. Click on **Topics** and select the mirror topic **vehicle-all-data**.

112. Click on the tab **Messages** to view the messages that are currently mirrored.

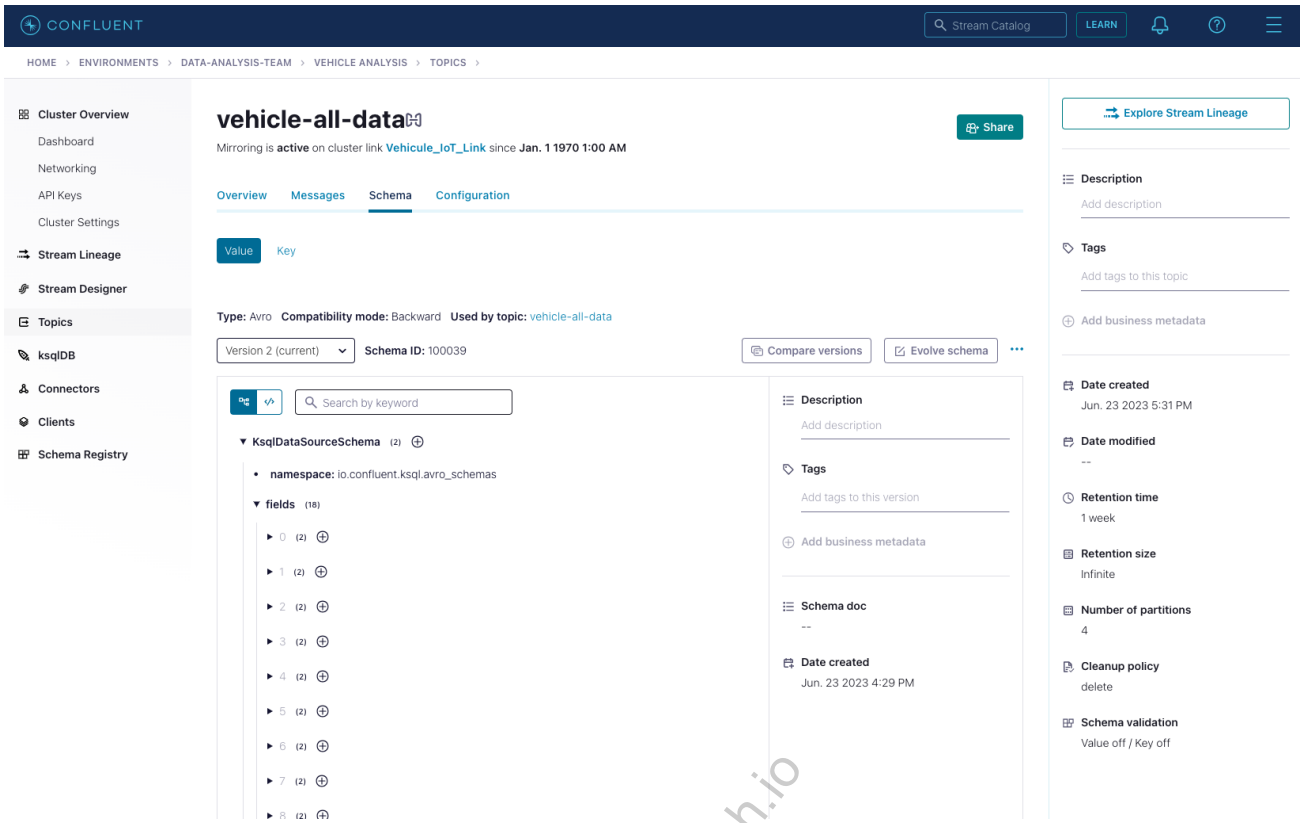
hitesh@datacouch.io



Make sure that the docker 'vehicle' containers are still running.

113. In the **Schema** tab, check that the schemas are correctly implemented from the synced subject **vehicle-all-data-value** by Schema Linking.

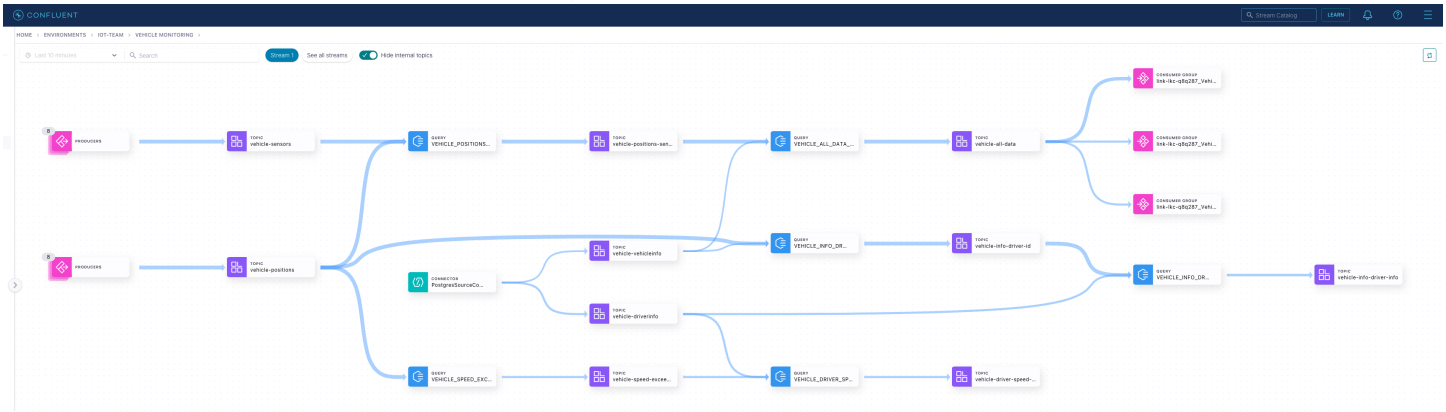
© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



hitesh@datacouch.io

Stream Lineage view

If you go to **Stream Lineage** in the **Vehicule Monitoring** cluster, you should see a bird's eye view of all components you have created during this exercise. Please, feel free to explore the different components.

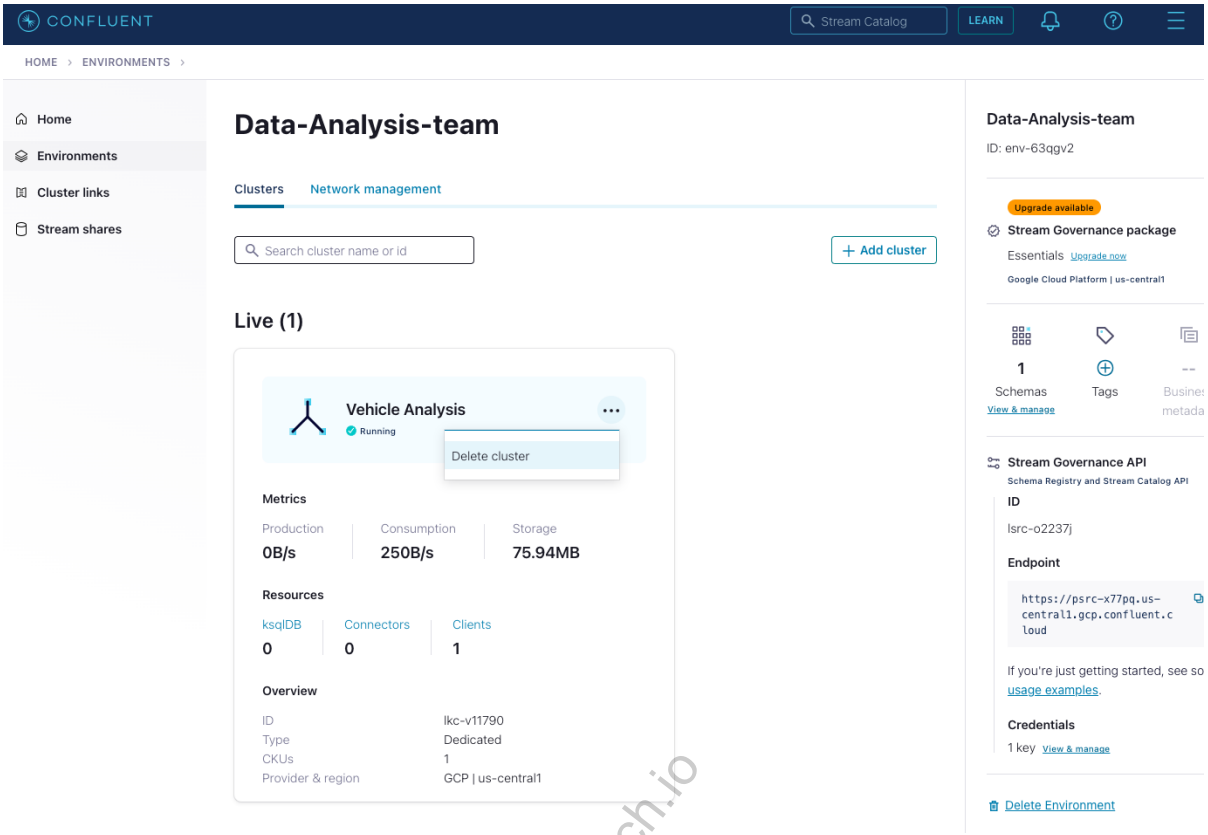


Clean up

114. Delete the DEDICATED cluster **Vehicule Analysis**:

    - Inside the **Data-Analysis-team** environment, click on the **three dots** on the **Vehicule Analysis** cluster.

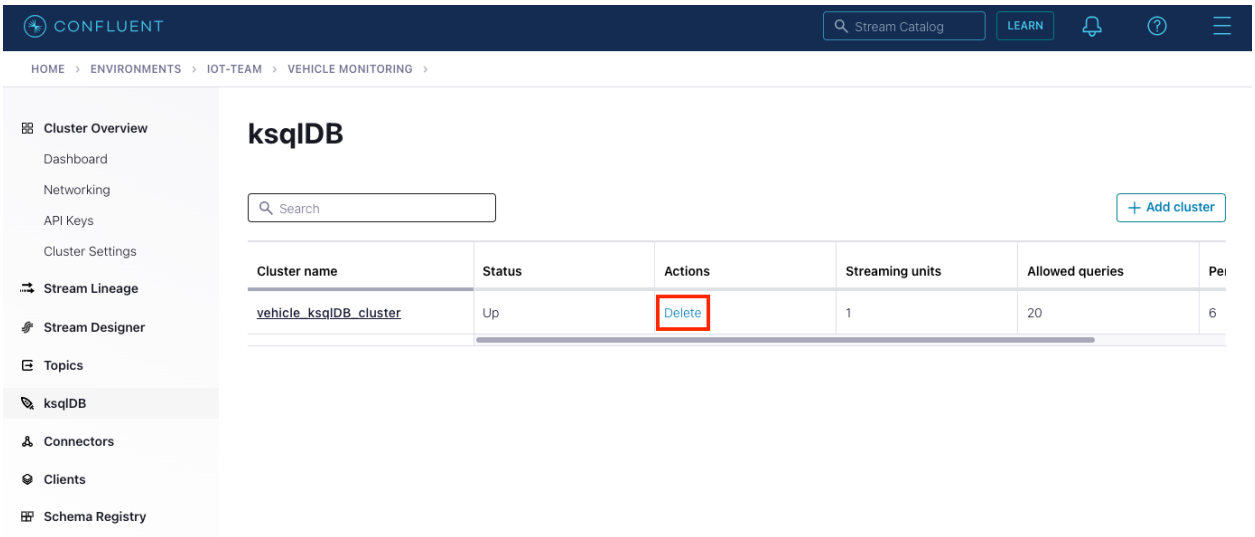
© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



- Click on **Delete cluster** and confirm.

115. Inside the **Data-Analysis-team** environment, click on **Delete Environment** and then confirm.

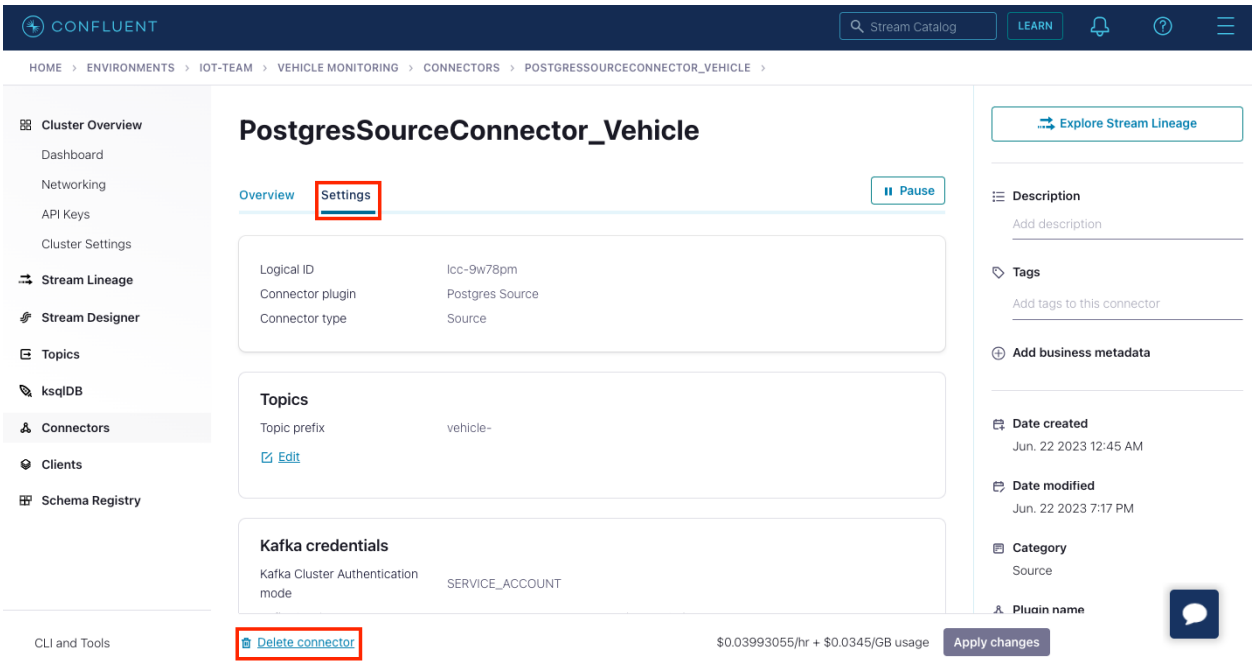
116. Go to the **IoT-team** environment and delete the ksqlDB cluster in the **Vehicle Monitoring** cluster.



117. Delete the Postgres connector in the **Vehicle Monitoring** cluster.

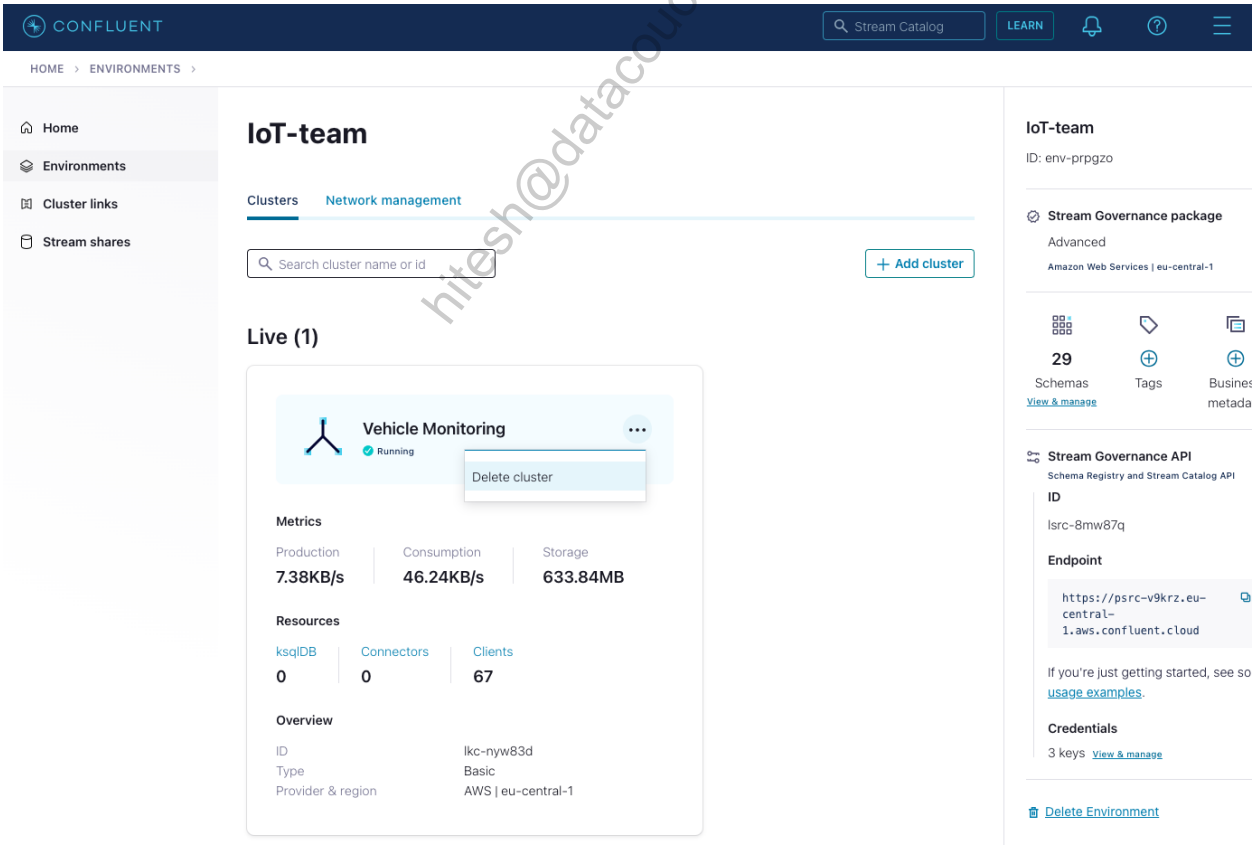
hitesh@datacouch.io

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.



118. Inside the **IoT-team** environment, click on the **three dots** on the **Vehicule Monitoring** cluster. Click on **Delete cluster** and confirm.

hitesh@datacouch.io



119. Now delete the **IoT-team** environment, click on **Delete Environment** on the right pane and then confirm.

Conclusion

In this exercise you have created an end-to-end data streaming architecture on Confluent Cloud. To do so, you have:

- Used Kafka API Keys, Schema Registry API Keys, ACLs and RBAC to grant granular access to Confluent Cloud resources
- Created Service Accounts for components that programmatically access Confluent Cloud
- Created a fully-managed Postgres Source connector to import tables to Confluent Cloud using SMTs
- Configured Kafka producers to write events to Confluent Cloud topics
- Created a ksqlDB cluster to run queries for transforming and enriching the data
- Used Schema Linking to sync schemas between two Schema Registry clusters
- Used Cluster Linking to mirror a topic between two Kafka clusters

hitesh@datacouch.io

STOP HERE. THIS IS THE END OF THE EXERCISE.

© 2014-2021 Confluent, Inc. Do not reproduce without prior written consent.