**Lab: The Complete Python Workflow (File Ingest & Real-time Stream)**

**Goal:** Use the kafka-python client library to programmatically manage a Redpanda Cloud topic and build a complete data pipeline that first ingests data from a file and then streams it back to your console in real-time.

# Purpose of the Lab

This lab is designed for developers who want to integrate Redpanda Cloud directly into their applications. You will write and run a series of Python scripts to perform the full lifecycle of a data pipeline: creating a topic, producing enriched data from a file, and creating a long-running consumer to process messages in real-time. This lab provides a practical, code-first understanding of how to use any standard Kafka client to build powerful, cloud-native streaming applications on Redpanda.

# **Prerequisites**

* A Redpanda Cloud account with a running cluster.
* Python is installed on your local machine.

# **Project Layout**

You will create a single directory for this lab.

|  |
| --- |
| rp-python-cloud-lab/ ├── users.jsonl ├── producer.py ├── consumer.py ├── manual\_producer.py └── .env |

# 

# **Part 1: Ingesting Data from a File**

## **Step 1: Get Cloud Credentials and Create a User**

1. **Navigate to Security:** In theRedpanda Cloud UI, click on the **Security** tab.
2. **Create a User:** Go to the **Users** sub-tab. Click **Create user**. Give the user a name (e.g., python-user) and save the generated **username** and **password**.
3. **Go to ACLs:** Click on the **ACLs** sub-tab.
4. **Grant Permissions:** Click **Create ACL**. Select the python-user. For the lab, select the **Allow All** preset, then click **Create**.
5. **Get Broker Address:** Go to the **Clusters** page, select your cluster. On the **Overview** page, click the **Kafka API** tab and copy the **Brokers** address.

## Step 2: Prepare the Project and Python Environment

1. **Create the project directory:**

|  |
| --- |
| mkdir rp-python-cloud-lab cd rp-python-cloud-lab |

1. **Create a sample data file named users.jsonl:**

**users.jsonl**

|  |
| --- |
| {"id":1,"email":"alice@example.com","country":"USA"} {"id":2,"email":"bob@example.com","country":"CA"} {"id":3,"email":"carol@example.com","country":"UK"} |

1. **Set up the Python virtual environment:**

|  |
| --- |
| python3 -m venv .env source .env/bin/activate pip install kafka-python python-dotenv certifi |

1. **Create the Environment File (.env):** This file will securely hold your secrets. **In a real project, you would add .env to your .gitignore file.**

|  |
| --- |
| # .env REDPANDA\_BROKERS="<YOUR\_BROKERS\_URL>" REDPANDA\_USER="<YOUR\_USERNAME>" REDPANDA\_PASS="<YOUR\_PASSWORD>" |

Replace the placeholders with the credentials you saved in Step 1.

## **Step 3: Create the Ingestion (Producer) Script**

Create a file named producer.py. This script will now load its configuration from the .env file and use certifi for SSL verification.

**producer.py**

|  |
| --- |
| # producer.py import json, time, os, certifi from kafka import KafkaProducer, KafkaAdminClient from kafka.admin import NewTopic from kafka.errors import KafkaError, TopicAlreadyExistsError from dotenv import load\_dotenv  # Load environment variables from .env file load\_dotenv()  # --- Configuration --- BROKER\_URL = os.getenv("REDPANDA\_BROKERS") USERNAME = os.getenv("REDPANDA\_USER") PASSWORD = os.getenv("REDPANDA\_PASS") TOPIC\_NAME = "user-signups-python" SOURCE\_FILE = "users.jsonl" # --- End Configuration ---  # Check if credentials are loaded if not all([BROKER\_URL, USERNAME, PASSWORD]):  raise ValueError("REDPANDA\_BROKERS, REDPANDA\_USER, and REDPANDA\_PASS must be set in the .env file.")  try:  # Step 1: Create the topic using an admin client  print("Connecting to Redpanda Cloud to create topic...")  admin = KafkaAdminClient(  bootstrap\_servers=BROKER\_URL, security\_protocol="SASL\_SSL",  sasl\_mechanism="SCRAM-SHA-256", sasl\_plain\_username=USERNAME, sasl\_plain\_password=PASSWORD,  ssl\_cafile=certifi.where(), # Use certifi for SSL verification  api\_version=(2, 0, 2) # Specify API version to avoid protocol errors  )  try:  admin.create\_topics(new\_topics=[NewTopic(name=TOPIC\_NAME, num\_partitions=1, replication\_factor=3)])  print(f"Topic '{TOPIC\_NAME}' created.")  except TopicAlreadyExistsError:  print(f"Topic '{TOPIC\_NAME}' already exists.")  finally:  admin.close()   # Step 2: Create the producer  producer = KafkaProducer(  bootstrap\_servers=BROKER\_URL, security\_protocol="SASL\_SSL",  sasl\_mechanism="SCRAM-SHA-256", sasl\_plain\_username=USERNAME, sasl\_plain\_password=PASSWORD,  value\_serializer=lambda v: json.dumps(v).encode('utf-8'),  ssl\_cafile=certifi.where(), # Use certifi for SSL verification  api\_version=(2, 0, 2) # Specify API version  )   # Step 3: Read the file and produce messages  print(f"Producing messages from '{SOURCE\_FILE}'...")  with open(SOURCE\_FILE, 'r') as f:  for line in f:  user\_data = json.loads(line)  user\_data['ingest\_time'] = time.strftime('%Y-%m-%dT%H:%M:%SZ', time.gmtime())  future = producer.send(TOPIC\_NAME, value=user\_data)  metadata = future.get(timeout=10)  print(f"Sent: {user\_data}")    producer.flush()  producer.close()  print("Producer finished and closed.")  except Exception as e:  print(f"An error occurred: {e}") |

## 

## Step 4: Run the Ingestion Script

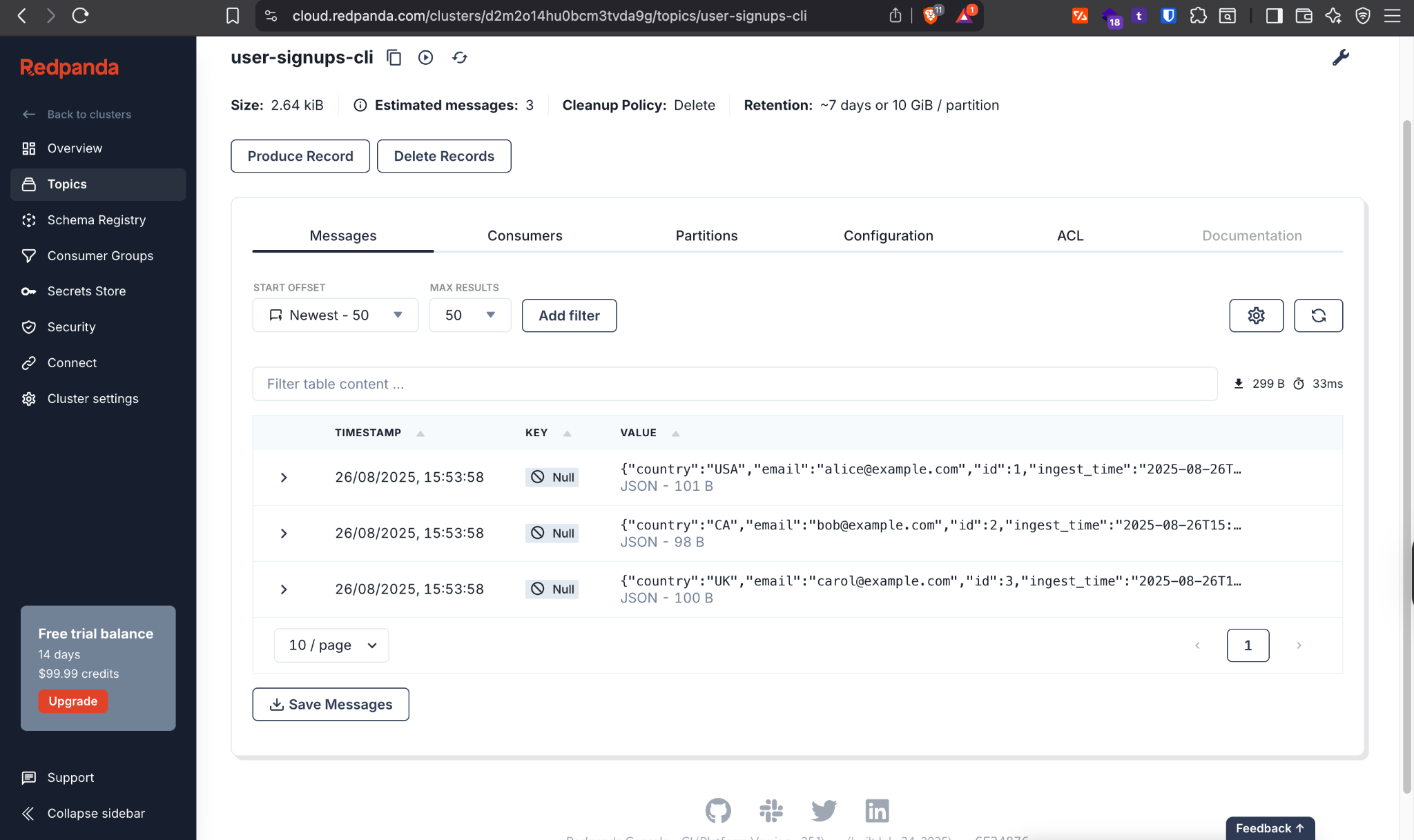
Execute the script from your terminal. It will automatically load the credentials from your .env file.

|  |
| --- |
| python producer.py |

**Expected output:** The script will print messages as it creates the topic and sends each of the three user records.

## **Step 5: Verify the Ingestion**

1. **Verify with rpk (CLI):** If you have an rpk profile configured, you can use it to quickly check the topic.  
   rpk topic consume user-signups-python -n 3 --profile rpk-cloud
2. **Verify in Cloud UI:** Navigate to your topic's **Messages** tab in the Redpanda Cloud UI to see the records.



# **Part 2: Streaming Data in Real-time**

## **Step 6: Create the Streaming (Consumer) Script**

Create a second file named consumer.py. This script will also load its configuration from the .env file and use certifi.

**consumer.py**

|  |
| --- |
| # consumer.py import json, os, certifi from kafka import KafkaConsumer from kafka.errors import KafkaError from dotenv import load\_dotenv  # Load environment variables from .env file load\_dotenv()  # --- Configuration --- BROKER\_URL = os.getenv("REDPANDA\_BROKERS") USERNAME = os.getenv("REDPANDA\_USER") PASSWORD = os.getenv("REDPANDA\_PASS") TOPIC\_NAME = "user-signups-python" # --- End Configuration ---  print("Starting Python consumer...") print("Press Ctrl+C to stop.")  try:  consumer = KafkaConsumer(  TOPIC\_NAME, bootstrap\_servers=BROKER\_URL,  security\_protocol="SASL\_SSL", sasl\_mechanism="SCRAM-SHA-256",  sasl\_plain\_username=USERNAME, sasl\_plain\_password=PASSWORD,  group\_id="python-stream-consumer-group", auto\_offset\_reset="earliest",  value\_deserializer=lambda v: json.loads(v.decode('utf-8')),  ssl\_cafile=certifi.where(), # Use certifi for SSL verification  api\_version=(2, 0, 2) # Specify API version  )  for message in consumer:  print(f"Received: {message.value}") except KeyboardInterrupt:  print("Consumer stopped by user.") finally:  if 'consumer' in locals() and consumer:  consumer.close()  print("Consumer closed.") |

## **Step 7: Run the Streaming Consumer**

In your terminal, run the consumer script.

|  |
| --- |
| python consumer.py |

**Expected output:** The script will start, print the three messages that are already in the topic, and then wait.

## **Step 8: Test the Real-time Stream**

1. **Open a new terminal window** and navigate to the same project directory. Activate the virtual environment (source .env/bin/activate).
2. Create a small, separate producer script named manual\_producer.py to send a single message.

**manual\_producer.p**

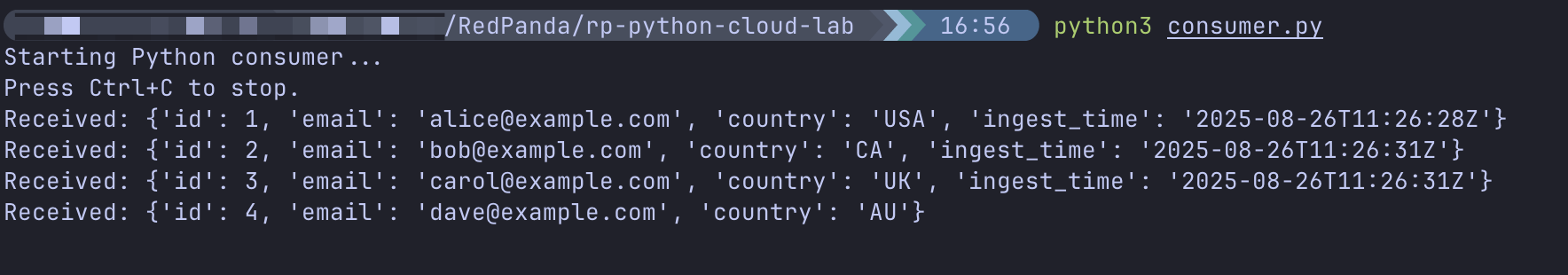
|  |
| --- |
| # manual\_producer.py import json, os, certifi from kafka import KafkaProducer from dotenv import load\_dotenv  # Load environment variables from .env file load\_dotenv()  # --- Configuration --- BROKER\_URL = os.getenv("REDPANDA\_BROKERS") USERNAME = os.getenv("REDPANDA\_USER") PASSWORD = os.getenv("REDPANDA\_PASS") TOPIC\_NAME = "user-signups-python" # --- End Configuration ---  producer = KafkaProducer(  bootstrap\_servers=BROKER\_URL, security\_protocol="SASL\_SSL",  sasl\_mechanism="SCRAM-SHA-256", sasl\_plain\_username=USERNAME, sasl\_plain\_password=PASSWORD,  value\_serializer=lambda v: json.dumps(v).encode('utf-8'),  ssl\_cafile=certifi.where(), # Use certifi for SSL verification  api\_version=(2, 0, 2) # Specify API version )  new\_user = {"id":4,"email":"dave@example.com","country":"AU"} print(f"Sending manual message: {new\_user}") producer.send(TOPIC\_NAME, value=new\_user) producer.flush() producer.close() print("Done.") |

1. Run this manual producer script.

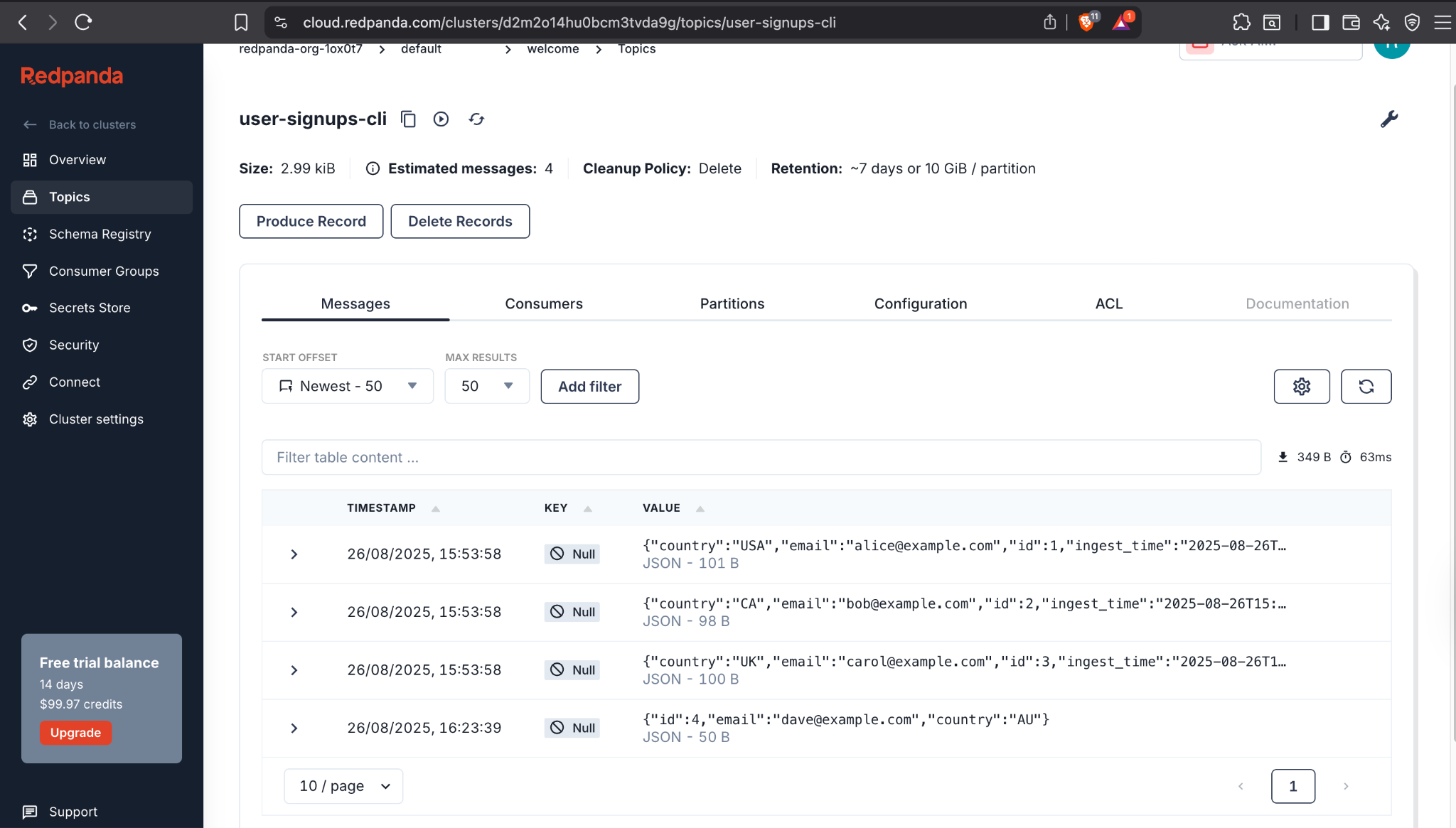
|  |
| --- |
| python manual\_producer.py |

## Step 9: Verify the Stream

1. **Verify in Terminal:** Switch back to your first terminal where the consumer.py script is running. You should see the new message for "dave" appear almost instantly.



1. **Verify in Cloud UI:** Refresh the **Messages** tab in the Redpanda Cloud UI. You will now see the fourth record for "dave".

**

# 

# **Cleanup**

1. Stop the consumer script by pressing Ctrl+C.
2. You can delete the topic from the Redpanda Cloud UI or with rpk.
3. Deactivate the virtual environment.

|  |
| --- |
| deactivate |