**Lab: Data Enrichment and PII Masking with Bloblang**

**Goal:** Use a Bloblang pipeline to consume raw, nested JSON events from a Redpanda Cloud topic, enrich them with a timestamp, remap the structure, and mask Personally Identifiable Information (PII) before publishing to a new topic.

# Purpose of the Lab

This lab provides a hands-on introduction to Bloblang, the powerful transformation language built into Redpanda Connect. You will tackle a common data engineering task: taking messy source data and transforming it into a well-structured, compliant format. By mapping fields, extracting nested values, adding a new timestamp, and using a built-in function to hash an email address, you will gain a practical understanding of how to build data cleaning and enrichment pipelines.

# Prerequisites

* A Redpanda Cloud account with a running cluster.
* The rpk CLI and rpk connect binary installed on your local machine.
* An rpk profile configured to connect to your cloud cluster (e.g., rpk-cloud).

# 

# Project Layout

You will create a new directory for this lab.

|  |
| --- |
| rp-bloblang-lab1/ ├── raw\_events.jsonl ├── pii-masking-pipeline.yaml └── .env |

# **Part 1: Setting up the Environment and Data**

## **Step 1: Get Cloud Credentials**

If you don't already have them, create a new user (e.g., bloblang-user) in the **Security -> Users** tab of the Redpanda Cloud UI. Go to the **ACLs** tab and grant this user **Allow All** permissions for the lab. Save the **Username**, **Password**, and your cluster's **Broker Address** from the **Kafka API** tab.

## **Step 2: Prepare the Project**

1. **Create the project directory:**

|  |
| --- |
| mkdir rp-bloblang-lab1 cd rp-bloblang-lab1 |

1. **Create a sample data file named raw\_events.jsonl:** This file contains nested data and a sensitive email field.

|  |
| --- |
| {"event\_id": "evt-001", "details": {"user": {"email": "alice@example.com", "id": 101}}, "payload": {"action": "login", "success": true}} {"event\_id": "evt-002", "details": {"user": {"email": "bob@corp.com", "id": 102}}, "payload": {"action": "logout", "success": true}} |

1. **Create the topics on Redpanda Cloud:**

|  |
| --- |
| rpk topic create raw-events --profile rpk-cloud rpk topic create enriched-events --profile rpk-cloud |

1. **Create the Environment File (.env):**

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

Replace the placeholders with the credentials for your bloblang-user.

1. **Produce the raw data to the raw-events topic:**

|  |
| --- |
| rpk topic produce raw-events --profile rpk-cloud < raw\_events.jsonl |

# 

# **Part 2: Building and Running the Transformation Pipeline**

## **Step 3: Create the Bloblang Pipeline Configuration**

Create a file named pii-masking-pipeline.yaml. This connector will read from the raw topic, apply the Bloblang mapping, and write to the enriched topic.

|  |
| --- |
| # pii-masking-pipeline.yaml input:  kafka:  addresses: [ "${REDPANDA\_BROKERS}" ]  topics: [ "raw-events" ]  consumer\_group: "enrichment-group"  start\_from\_oldest: true  tls: { "enabled": true }  sasl:  mechanism: "SCRAM-SHA-256"  user: "${REDPANDA\_USER}"  password: "${REDPANDA\_PASS}"  pipeline:  processors:  - bloblang: |  root = {  "correlation\_id": this.event\_id,  "user\_id": this.details.user.id,  "user\_email\_hash": this.details.user.email.hash("sha256"),  "user\_action": this.payload.action,  "status": if this.payload.success { "SUCCESS" } else { "FAILURE" },  "processing\_time": now().string()  }  output:  kafka:  addresses: [ "${REDPANDA\_BROKERS}" ]  topic: "enriched-events"  tls: { "enabled": true }  sasl:  mechanism: "SCRAM-SHA-256"  user: "${REDPANDA\_USER}"  password: "${REDPANDA\_PASS}" |

**What's included in the Bloblang**

* root = {}: We are creating a brand new, clean JSON object.
* user\_id: this.details.user.id: This shows how to "un-nest" a deeply nested field.
* user\_email\_hash: this.details.user.email.hash("sha256"): This extracts the email and applies a SHA256 hash to it, masking the PII.
* status: if this.payload.success { "SUCCESS" } else { "FAILURE" }: A simple conditional mapping to transform a boolean into a more readable string.
* processing\_time: now().string(): Enriches the message with the current timestamp.

## **Step 4: Run the Pipeline**

Execute the connector. Since this is a streaming pipeline, it will process the existing messages and then wait for more, occupying your current terminal window.

|  |
| --- |
| rpk connect run --env-file .env ./pii-masking-pipeline.yaml |

**Expected output:** The connector will start and run without errors. You can stop it with Ctrl+C after verification.

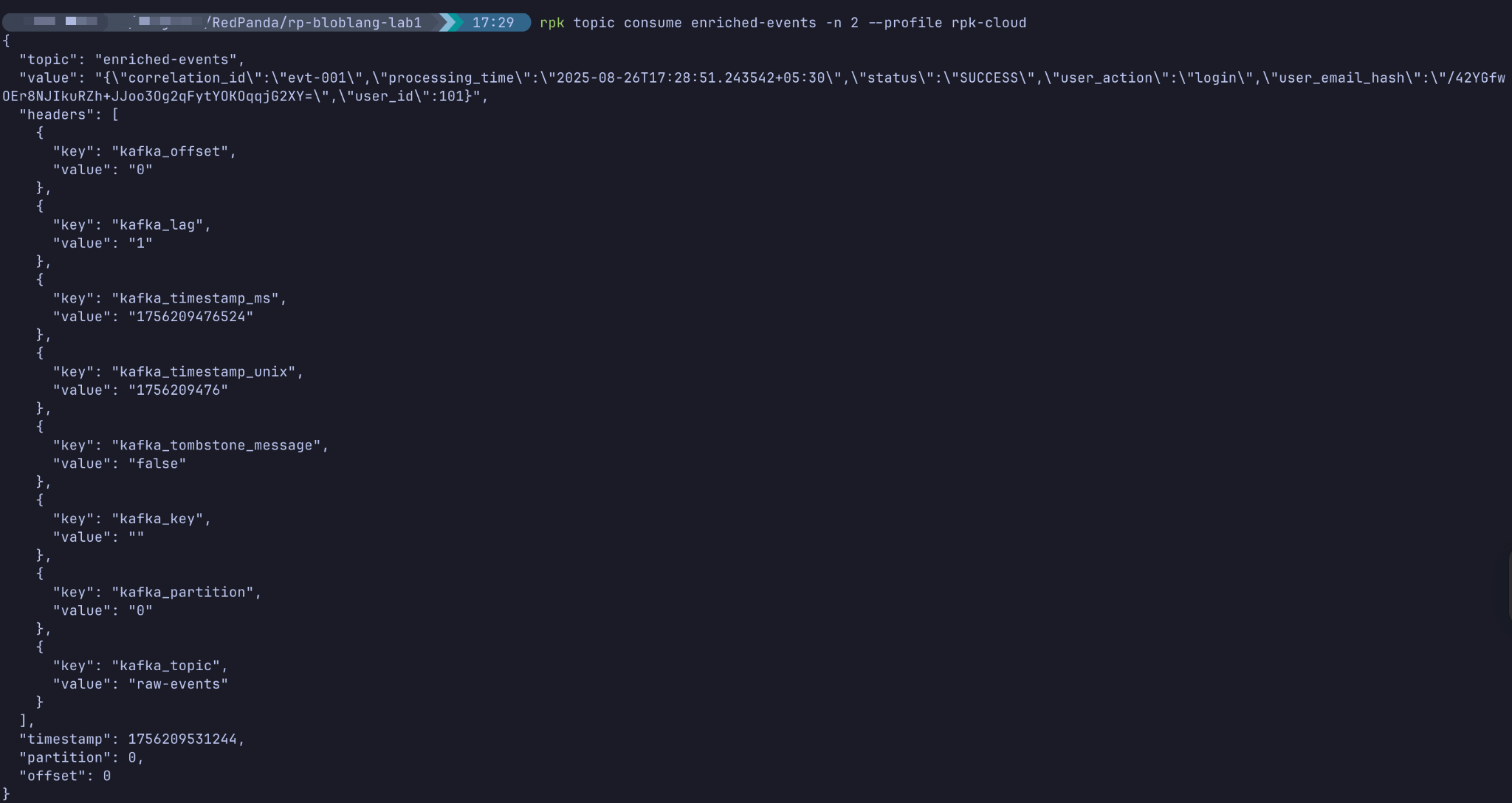
## **Step 5: Verify the Transformation**

1. **Open a new terminal window** and navigate to the same project directory.

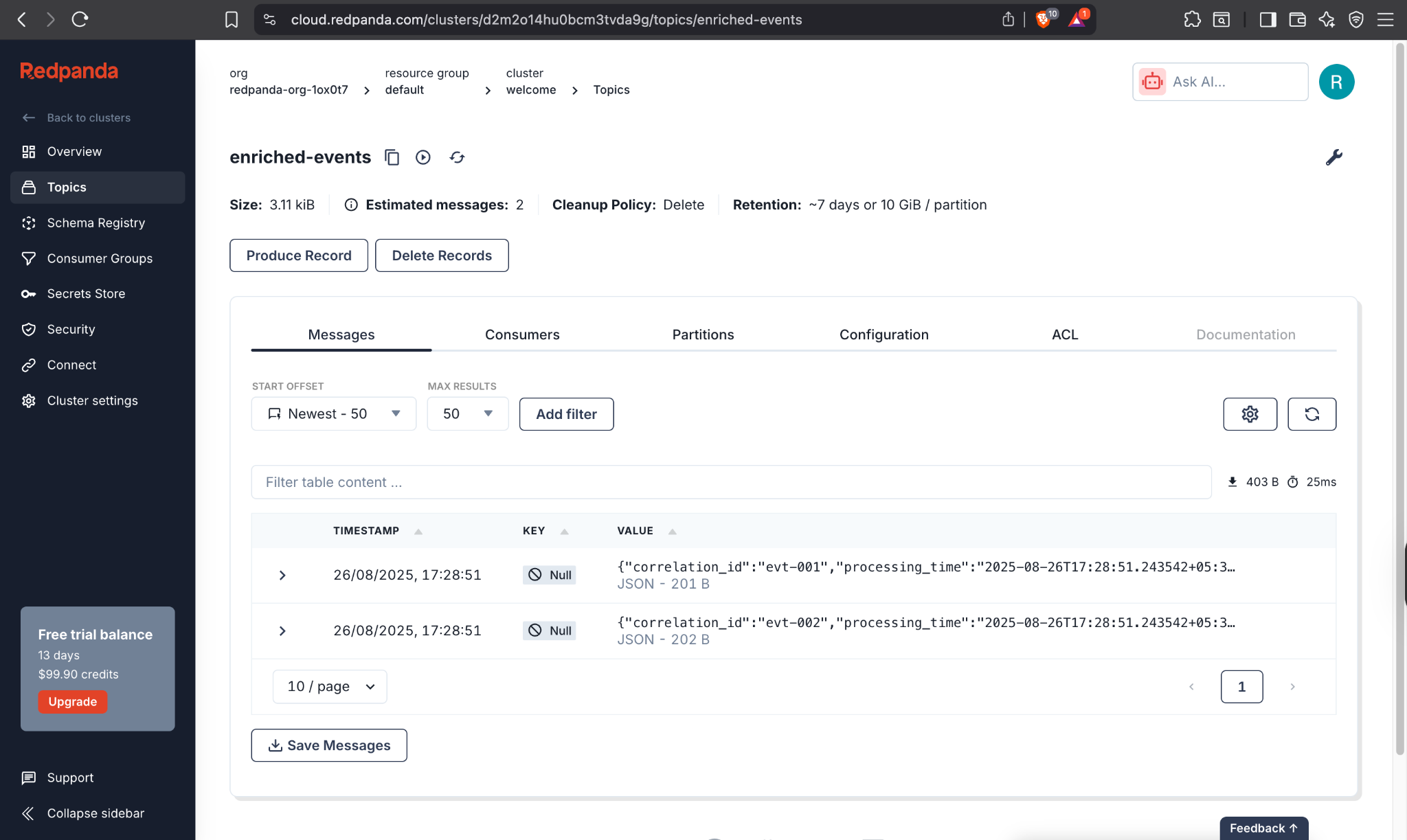
**Verify with rpk (CLI):** Consume from the enriched-events topic to see the result.

|  |
| --- |
| rpk topic consume enriched-events -n 2 --profile rpk-cloud |

**Expected output:** You will see two completely restructured JSON messages. The email is hashed, fields are renamed, and the new timestamp is present.



1. **Verify in Cloud UI:** Navigate to the enriched-events topic in the Redpanda Cloud UI.



# Cleanup

|  |
| --- |
| rpk topic delete raw-events --profile rpk-cloud rpk topic delete enriched-events --profile rpk-cloud |