**Lab 2: Conditional Logic and Filtering with Bloblang**

**Goal:** Use a Bloblang pipeline to process a stream of mixed transaction events, filter out irrelevant messages, and route high-value transactions to a dedicated alerting topic based on conditional logic.

# **Purpose of the Lab**

This lab explores one of the most powerful features of Bloblang: conditional logic for filtering and routing. You will build an intelligent pipeline that acts as a dynamic filter. By inspecting the content of incoming messages and using an if/else statement, the pipeline will selectively process only the messages that meet specific criteria (e.g., high-value sales). This lab teaches a critical skill for building efficient, real-time systems that can react to business events, reduce noise, and route important information to the correct downstream systems.

# **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-lab2/ ├── transactions.jsonl ├── high-value-filter.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 and grant it **Allow All** permissions in the **ACLs** tab. Save the **Username**, **Password**, and your cluster's **Broker Address**.

## Step 2: Prepare the Project

1. **Create the project directory:**

|  |
| --- |
| mkdir rp-bloblang-lab2 cd rp-bloblang-lab2 |

1. **Create a sample data file named transactions.jsonl:** This file contains a mix of transaction types and values.

**transactions.jsonl**

|  |
| --- |
| {"tx\_id": "txn\_1001", "type": "SALE", "amount": 250.75, "currency": "USD"} {"tx\_id": "txn\_1002", "type": "REFUND", "amount": 50.00, "currency": "USD"} {"tx\_id": "txn\_1003", "type": "SALE", "amount": 99.99, "currency": "USD"} {"tx\_id": "txn\_1004", "type": "AUTH", "amount": 1500.00, "currency": "USD"} {"tx\_id": "txn\_1005", "type": "SALE", "amount": 1200.00, "currency": "USD"} |

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

|  |
| --- |
| rpk topic create all-transactions --profile rpk-cloud rpk topic create high-value-sales-alerts --profile rpk-cloud |

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

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

Populate the file with your credentials.

1. **Produce the raw data to the all-transactions topic:**

|  |
| --- |
| rpk topic produce all-transactions --profile rpk-cloud < transactions.jsonl |

# 

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

## 

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

Create a file named high-value-filter.yaml. This version uses a more robust Bloblang mapping.

**high-value-filter.yaml**

|  |
| --- |
| # high-value-filter.yaml input:  kafka:  addresses: [ "${REDPANDA\_BROKERS}" ]  topics: [ "all-transactions" ]  consumer\_group: "filtering-group"  start\_from\_oldest: true  tls: { "enabled": true }  sasl:  mechanism: "SCRAM-SHA-256"  user: "${REDPANDA\_USER}"  password: "${REDPANDA\_PASS}"  pipeline:  processors:  - bloblang: |  root = if this.type == "SALE" && this.amount >= 100.00 {  {  "alert\_type": "HIGH\_VALUE\_SALE",  "transaction\_id": this.tx\_id,  "sale\_amount": this.amount,  "alert\_timestamp": now().string()  }  } else {  deleted()  }  output:  kafka:  addresses: [ "${REDPANDA\_BROKERS}" ]  topic: "high-value-sales-alerts"  tls: { "enabled": true }  sasl:  mechanism: "SCRAM-SHA-256"  user: "${REDPANDA\_USER}"  password: "${REDPANDA\_PASS}" |

**What's included in the Bloblang**

* root = if ...: This is a more idiomatic Bloblang pattern. The entire if/else block is an expression, and its result is assigned to root.
* { ... }: If the condition is met, the expression returns a new JSON object.
* else { deleted() }: If the condition is false, the expression returns a signal to delete the message.

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

Execute the connector. It will start, process the five existing messages, and then wait for more, occupying your current terminal.

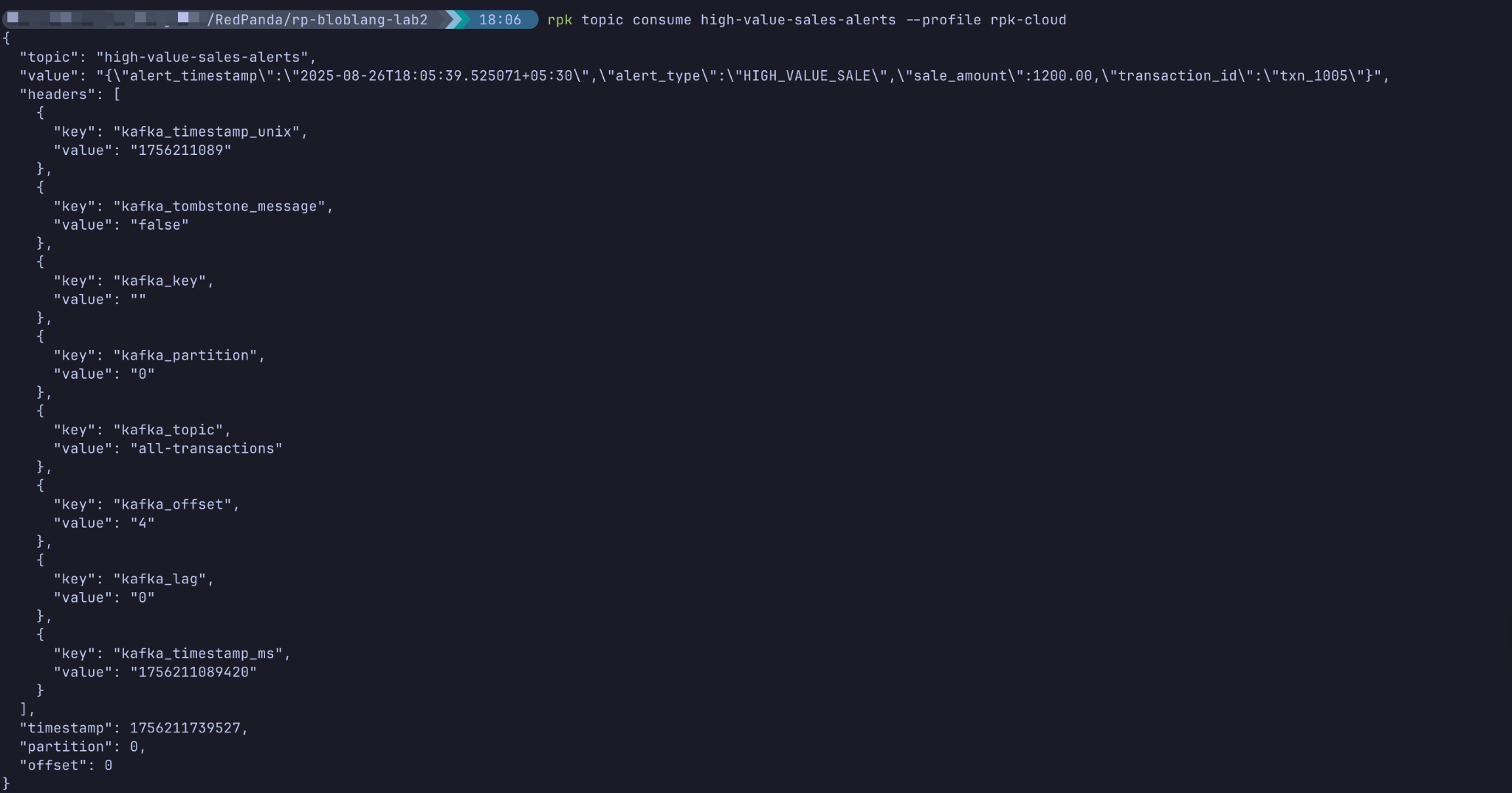
|  |
| --- |
| rpk connect run --env-file .env ./high-value-filter.yaml |

## Step 5: Verify the Filtering

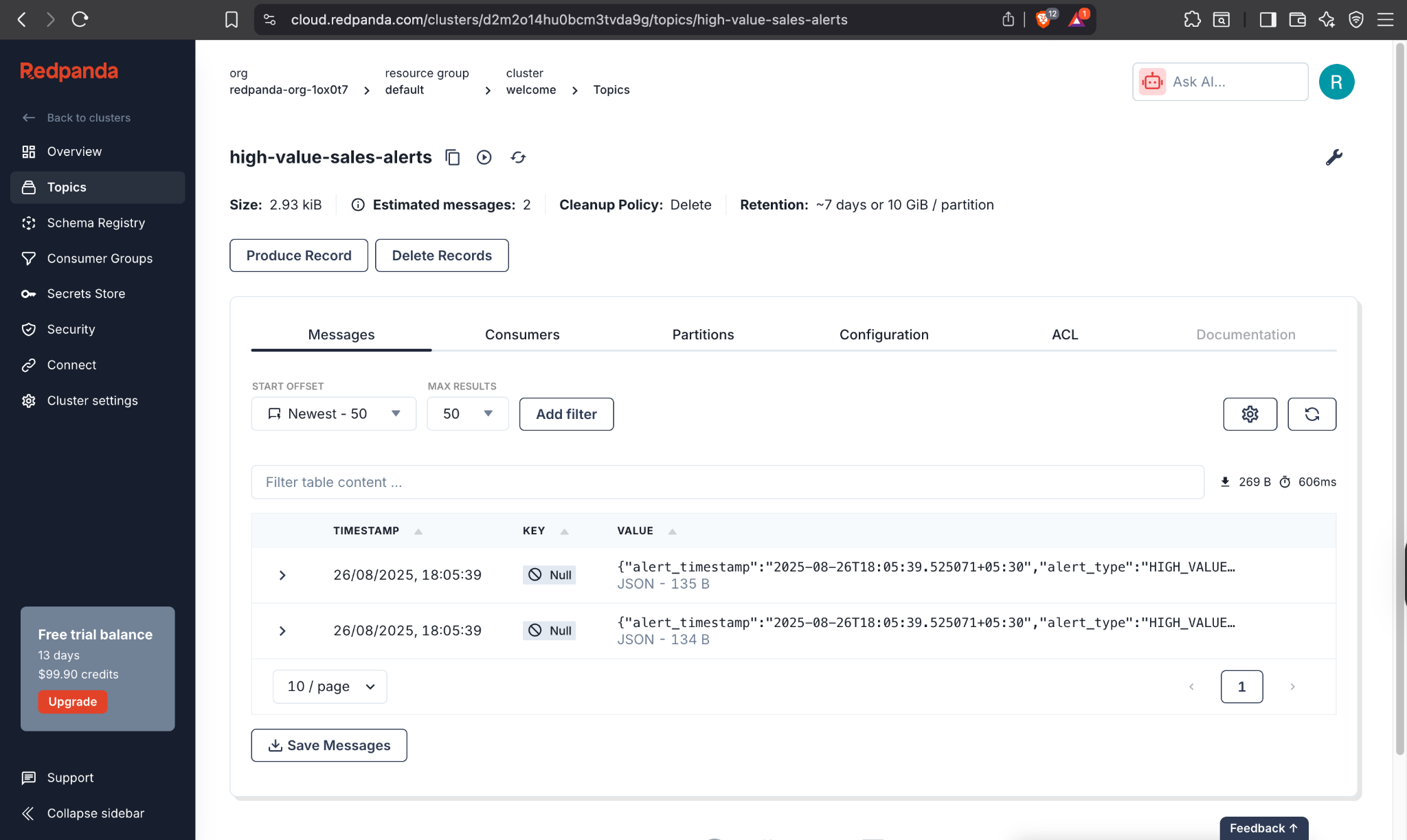
1. **Open a new terminal window** and navigate to the same project directory.
2. **Verify with rpk (CLI):** Consume from the high-value-sales-alerts topic.

|  |
| --- |
| rpk topic consume high-value-sales-alerts --profile rpk-cloud |

**Expected output:** You will **only** see the two transactions that were sales of $100 or more. The refund, the low-value sale, and the authorization have been filtered out.



1. **Verify in Cloud UI:** Navigate to the high-value-sales-alerts topic in the Redpanda Cloud UI.

**

# **Cleanup**

1. Stop the connector by pressing Ctrl+C in the first terminal.
2. Delete the topics:

|  |
| --- |
| rpk topic delete all-transactions --profile rpk-cloud rpk topic delete high-value-sales-alerts --profile rpk-cloud |