**Lab: Docker-based File-to-File Pipeline**

**Goal:** Use a single Docker Compose file to stand up a complete streaming pipeline that reads data from a local file, sends it through a Redpanda topic, and writes the transformed data to a new local file, all within a containerized environment.

# **Purpose of the Lab**

This lab demonstrates a complete, self-contained data pipeline orchestrated entirely with Docker Compose. It is designed for developers who want to understand how to build and test complex, multi-stage data flows in a local, reproducible environment. By defining the broker, console, and multiple Redpanda Connect instances as services, you will see how to chain connectors together to create a data flow that reads from a source, enriches the data, and writes to a sink—a foundational pattern in stream processing.

# **Prerequisites**

* Docker and Docker Compose installed on your local machine.

# **Project Layout**

You will create a single directory for this lab which will contain all the necessary files.

|  |
| --- |
| rp-docker-pipeline-lab/ ├── input\_data.jsonl ├── producer-config.yaml ├── consumer-config.yaml └── docker-compose.yml |

# **Part 1: Setting up the Project and Configuration**

## Step 1: Prepare the Project Directory

1. **Create the project directory:**

|  |
| --- |
| mkdir rp-docker-pipeline-lab cd rp-docker-pipeline-lab |

1. **Create the input data file named input\_data.jsonl:**

**input\_data.jsonl**

|  |
| --- |
| {"log\_id": "err-001", "level": "error", "message": "Service unavailable"} {"log\_id": "inf-002", "level": "info", "message": "User logged in"} {"log\_id": "warn-003", "level": "warning", "message": "Disk space is low"} |

## **Step 2: Create the Redpanda Connect Configurations**

1. **Create the Producer Configuration:** Create a file named producer-config.yaml. This connector reads the local file and produces its contents to a Redpanda topic.

**producer-config.yaml**

|  |
| --- |
| # producer-config.yaml input:  file:  paths: ["/data/input\_data.jsonl"]  codec: "lines" output:  kafka:  addresses: ["redpanda:9092"]  topic: "logs-topic" |

1. **Create the Consumer Configuration:** Create a file named consumer-config.yaml. This connector consumes from the Redpanda topic, enriches the data with Bloblang, and writes the result to a new file.

**consumer-config.yaml**

|  |
| --- |
| # consumer-config.yaml input:  kafka:  addresses: ["redpanda:9092"]  topics: ["logs-topic"]  consumer\_group: "file-writer-group"  start\_from\_oldest: true pipeline:  processors:  - bloblang: |  root = this  root.processed\_at = now().string()  root.level = this.level.uppercase() output:  file:  path: "/data/output\_data.jsonl"  codec: "lines" |

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

## Step 3: Create the Complete Docker Compose File

Create the docker-compose.yml file. This single file defines the Redpanda broker, the Redpanda Console, and both Redpanda Connect instances with more specific container names.

**docker-compose.yml**

|  |
| --- |
| services:  redpanda:  image: docker.redpanda.com/redpandadata/redpanda:latest  container\_name: file-pipeline-broker  command:  - redpanda  - start  - --smp=1  - --memory=1G  - --overprovisioned  - --node-id=0  - --kafka-addr=PLAINTEXT://0.0.0.0:9092  - --advertise-kafka-addr=PLAINTEXT://redpanda:9092  ports:  - "9092:9092"  - "9644:9644"  healthcheck:  test: ["CMD-SHELL", "rpk cluster health | grep -E 'Healthy:.+true' || exit 1"]  interval: 5s  timeout: 3s  retries: 5  networks:  - redpanda\_net   redpanda-console:  image: docker.redpanda.com/redpandadata/console:latest  container\_name: file-pipeline-console  depends\_on:  redpanda:  condition: service\_healthy  environment:  KAFKA\_BROKERS: "redpanda:9092"  ports:  - "8080:8080"  networks:  - redpanda\_net   redpanda-connect-producer:  image: docker.redpanda.com/redpandadata/connect:latest  container\_name: file-pipeline-producer  depends\_on:  redpanda:  condition: service\_healthy  volumes:  - ./:/data  command: ["run", "/data/producer-config.yaml"]  networks:  - redpanda\_net   redpanda-connect-consumer:  image: docker.redpanda.com/redpandadata/connect:latest  container\_name: file-pipeline-consumer  depends\_on:  redpanda:  condition: service\_healthy  volumes:  - ./:/data  command: ["run", "/data/consumer-config.yaml"]  networks:  - redpanda\_net  networks:  redpanda\_net: |

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

Execute the entire pipeline with a single command. The -d flag runs the containers in "detached" mode, meaning they will run in the background.

|  |
| --- |
| docker compose up -d |

**Expected output:** Docker Compose will start all services in the background and return you to your command prompt.

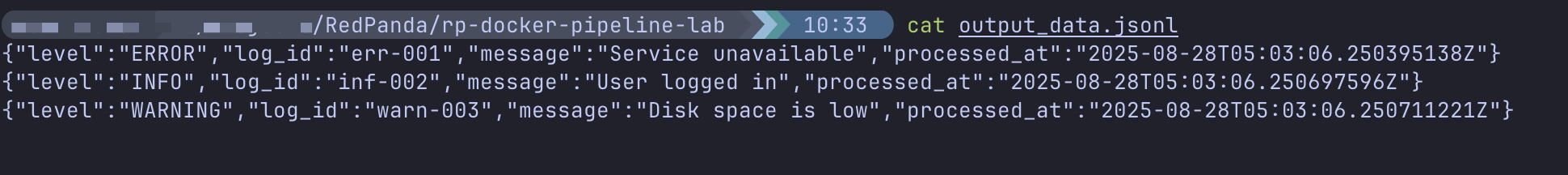
## **Step 5: Verify the Results**

1. **Check the container status:** Run docker compose ps to see the state of your services. After a few moments, you should see that the file-pipeline-producer has exited (since its job is done), while the other services are still running.  
   docker compose ps
2. **Open a new terminal window** and navigate to the same project directory.
3. **Check the output file:** Examine the contents of the newly created output\_data.jsonl file.

|  |
| --- |
| cat output\_data.jsonl |

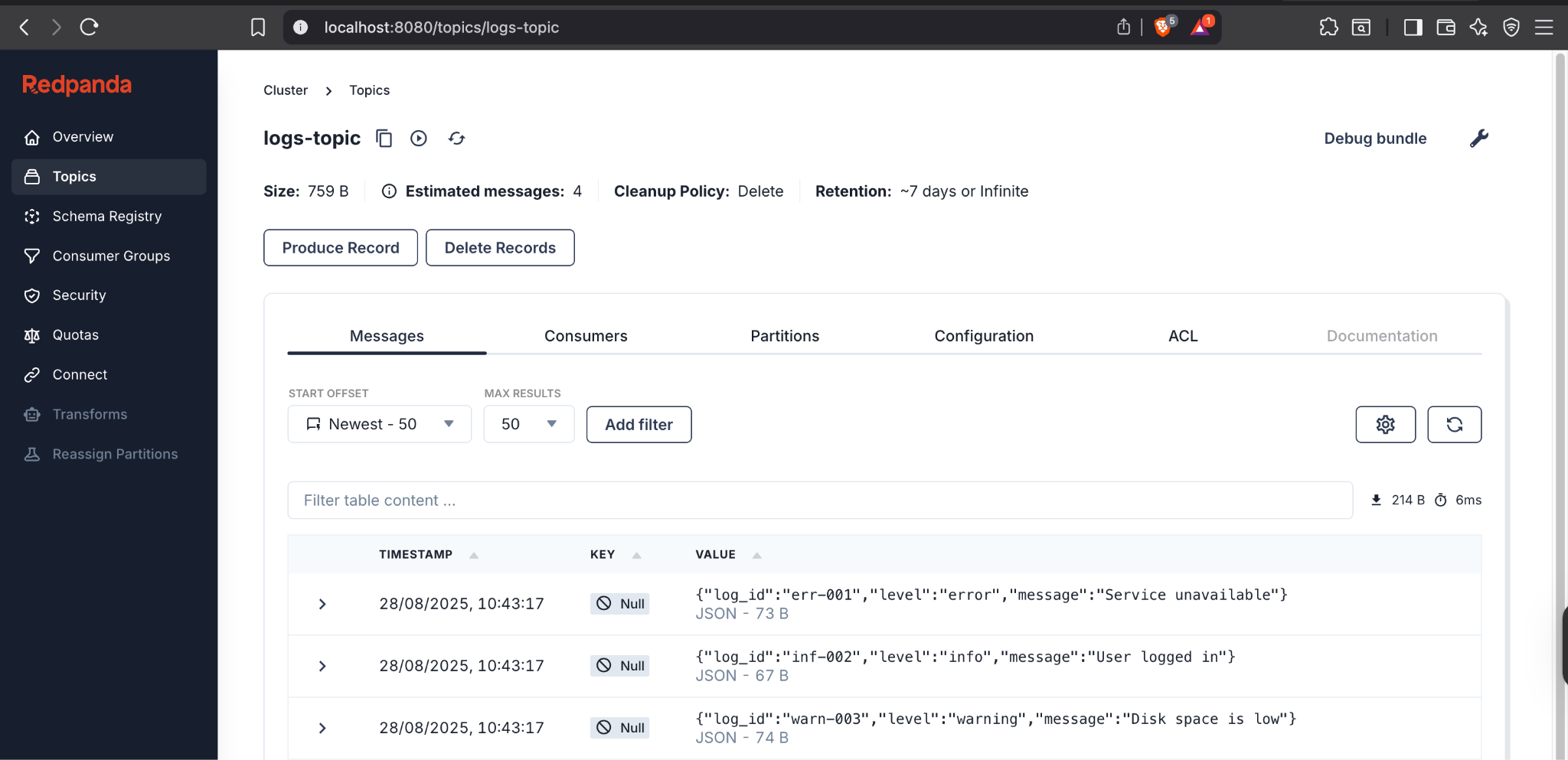
**Expected output:** You should see the three log messages, now transformed. The level is in uppercase, and each message has a new processed\_at timestamp.

|  |
| --- |
| {"level":"ERROR","log\_id":"err-001","message":"Service unavailable","processed\_at":"2025-08-28T05:00:00Z"} {"level":"INFO","log\_id":"inf-002","message":"User logged in","processed\_at":"2025-08-28T05:00:00Z"} {"level":"WARNING","log\_id":"warn-003","message":"Disk space is low","processed\_at":"2025-08-28T05:00:00Z"} |



1. **Verify in Redpanda Console (UI):**

* Open your web browser and navigate to the Redpanda Console, which is typically running at http://localhost:8080.
* Go to the **Topics** page and click on logs-topic.
* You should see the three original, untransformed log messages that were sent by the producer.



# Cleanup

1. Stop and remove the containers and network:

|  |
| --- |
| docker compose down |