# **Lab: Docker-Compose Dev Cluster (Core)**

**Goal:** Stand up a local **single-node Redpanda + Console** using Docker Compose with **persistent volumes**, **custom network**, and **healthchecks**, then confirm cluster status in Console.

## **Purpose of the Lab**

Give learners a fast, repeatable way to spin up a **local Redpanda + Console** environment using **Docker Compose**, so they can prototype and debug pipelines without touching shared clusters. By defining services, volumes, networks, and healthchecks in a single docker-compose.yml, participants practice **infrastructure-as-code for development**, validate broker readiness in **Console**, and confirm end-to-end messaging locally. The outcome is a reproducible dev sandbox that supports rapid iteration, safer experimentation, and clear handoffs between teammates.

## **Prerequisites**

* Docker Desktop / Engine (4+ GB RAM available to Docker)
* Docker Compose v2
* macOS/Windows: make sure **localhost:8080** and **9092** are free

## **Project layout**

|  |
| --- |
| redpanda-dc-lab/  docker-compose.yml |

Optional: create this folder anywhere (e.g., in your home directory or a dev workspace).

## **Compose file**

Create **docker-compose.yml** with the following:

**docker-compose.yml**

|  |
| --- |
| version: "3.8"  services:  redpanda:  image: redpandadata/redpanda:latest  container\_name: redpanda  command:  - redpanda  - start  - --overprovisioned  - --smp=1  - --memory=1G  - --reserve-memory=0M  - --node-id=0  - --check=false  # Two Kafka listeners: internal (containers) + external (host)  - --kafka-addr=PLAINTEXT://0.0.0.0:9092,OUTSIDE://0.0.0.0:19092  - --advertise-kafka-addr=PLAINTEXT://redpanda:9092,OUTSIDE://localhost:19092  # RPC (node-to-node / client metadata)  - --rpc-addr=0.0.0.0:33145  - --advertise-rpc-addr=redpanda:33145  # NOTE: no --http-addr (not supported on your build; admin API uses default 9644)  ports:  - "9092:9092" # for other containers via redpanda:9092  - "19092:19092" # for host apps via localhost:19092  - "9644:9644" # admin API (default bind)  volumes:  - redpanda-data:/var/lib/redpanda/data  healthcheck:  test: ["CMD-SHELL", "curl -sf http://localhost:9644/v1/status/ready >/dev/null"]  interval: 5s  timeout: 3s  retries: 30  start\_period: 10s  networks:  - redpanda-net   console:  image: redpandadata/console:latest  container\_name: redpanda-console  depends\_on:  redpanda:  condition: service\_healthy  environment:  KAFKA\_BROKERS: redpanda:9092 # talk to broker on the compose network  ports:  - "8080:8080"  networks:  - redpanda-net  volumes:  redpanda-data:  networks:  redpanda-net: |

**What’s included**

* **Persistent volume** redpanda-data so your topics & messages survive up/down.
* **Custom network** redpanda-net so services talk by name (redpanda:9092).
* **Healthcheck** hitting http://localhost:9644/v1/status/ready inside the container so Console waits until the broker is ready.

## **Bring it up**

From the redpanda-dc-lab/ folder:

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

**Expected output (truncated):**

|  |
| --- |
| [+] Running 2/2  ✔ Container redpanda Started  ✔ Container redpanda-console Started |

Check status:

|  |
| --- |
| docker compose ps |

**Expected:**

|  |
| --- |
| NAME COMMAND STATE PORTS redpanda "redpanda start ..." Up 0.0.0.0:9092->9092/tcp, 0.0.0.0:9644->9644/tcp redpanda-console "/app/console" Up 0.0.0.0:8080->8080/tcp |

(Optional) Inspect health:

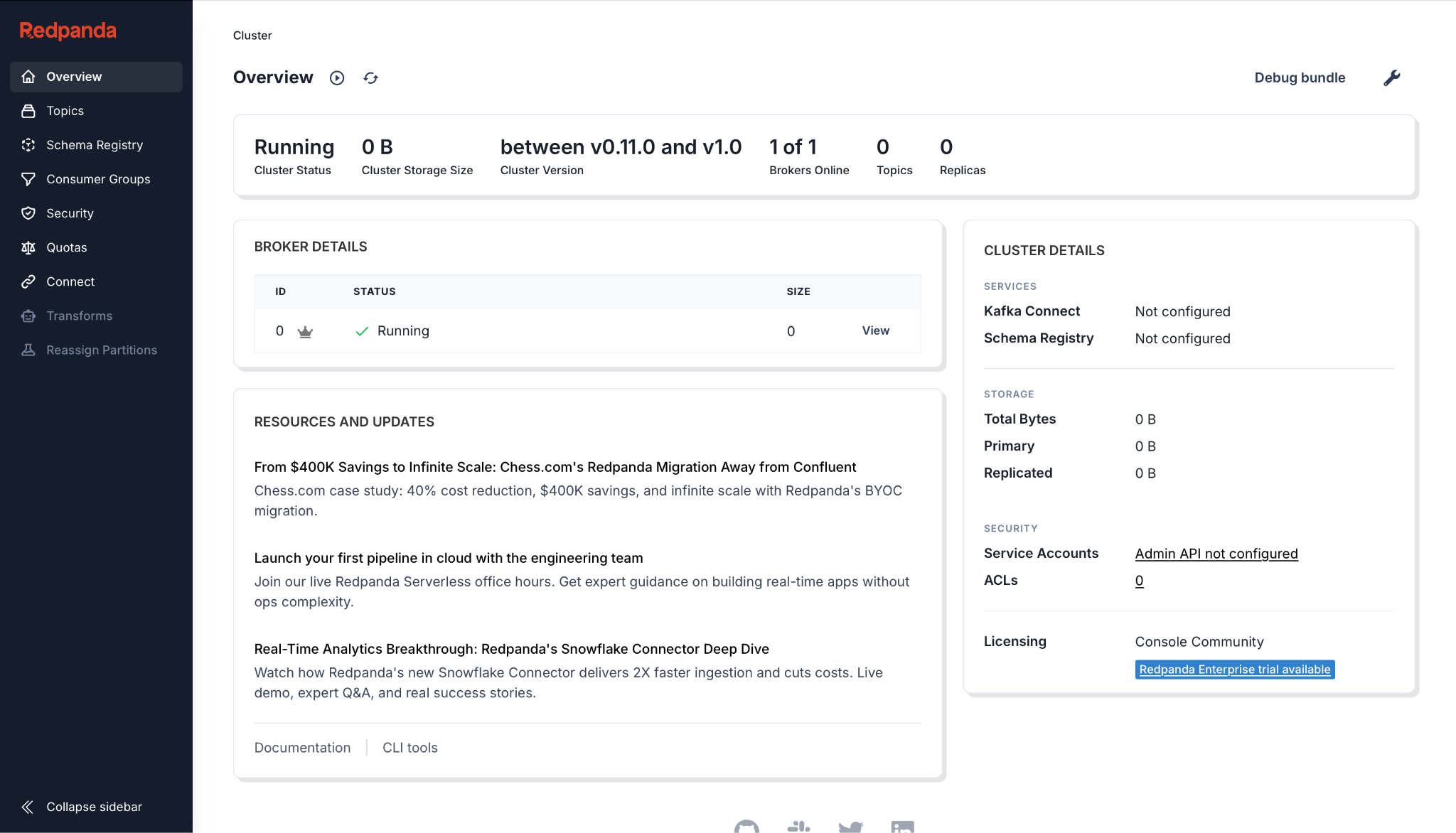
|  |
| --- |
| docker inspect --format '{{json .State.Health }}' redpanda | jq |

**Expected (truncated):**

|  |
| --- |
| {  "Status": "healthy",  "Log": [  {"Output": "", "ExitCode": 0}  ] } |

## **Verify in Console (UI)**

Open [**http://localhost:8080**](http://localhost:8080/) in your browser.



You should see **Redpanda Console**:

* **Brokers/Cluster** visible on the landing page
* **Topics** list (empty on first boot)
* **Consumer Groups** (empty on first boot)
* **Configuration** and **Metrics** tabs available

If the page doesn’t load, re-check that Docker Desktop is running and that port **8080** isn’t used by another app.

## **Quick functional smoke test with rpk (inside the container)**

Use the rpk CLI shipped in the Redpanda container.

**Create a topic:**

|  |
| --- |
| docker exec -it redpanda rpk topic create demo -p 3 -r 1 --brokers redpanda:9092 |

**Expected:**

|  |
| --- |
| TOPIC STATUS  demo OK |

**Produce a couple of messages (Ctrl+D to finish):**

|  |
| --- |
| docker exec -it redpanda rpk topic produce demo --brokers redpanda:9092 |

|  |
| --- |
| > hello from compose > second line |

**Consume them back:**

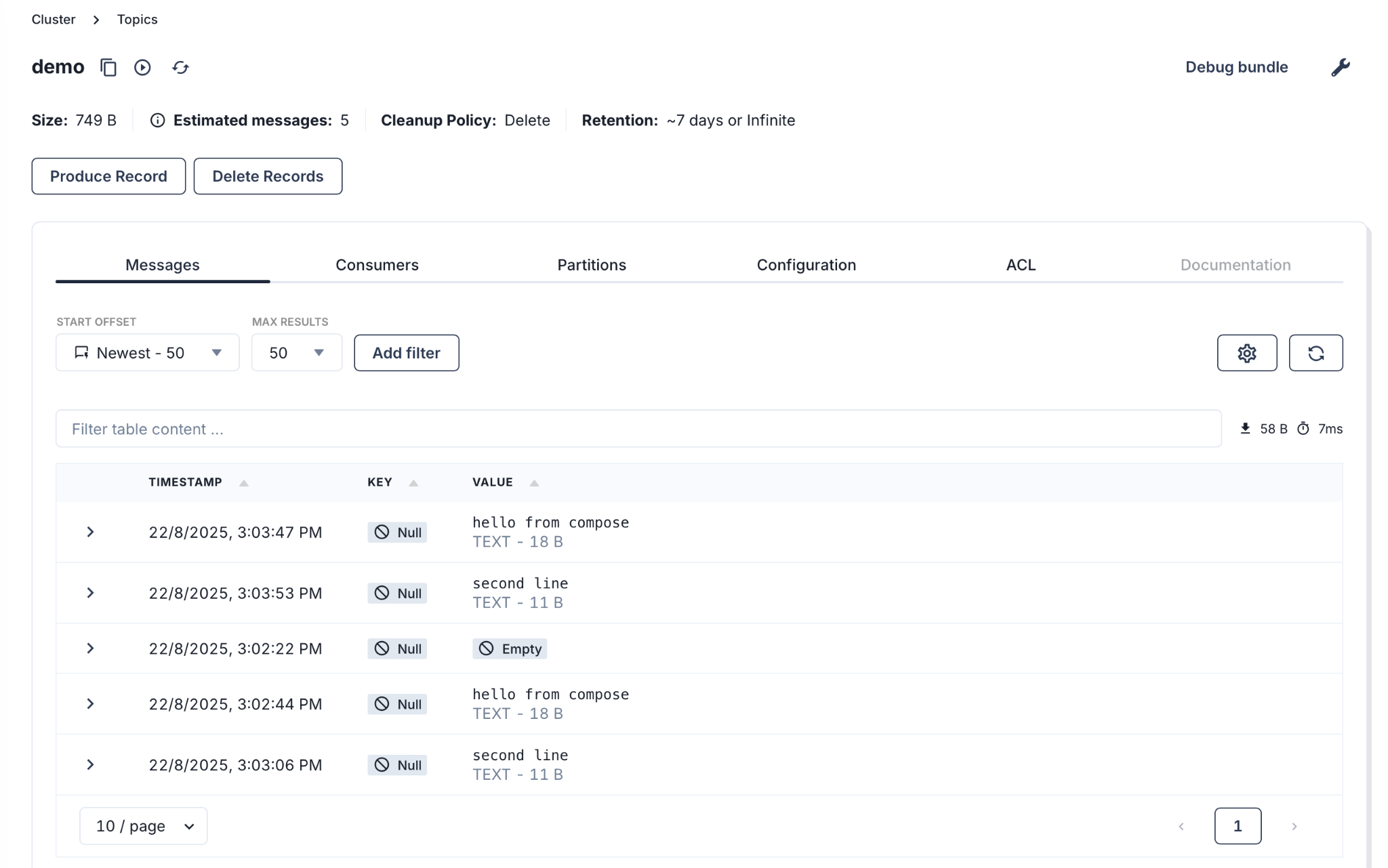
|  |
| --- |
| docker exec -it redpanda rpk topic consume demo --brokers redpanda:9092 |

**Expected:**

|  |
| --- |
| {  "message": "hello from compose" } {  "message": "second line" } |

**Now check Console:**

* Navigate to **Topics → demo → Messages** and confirm the two records are visible.



## **What to show learners (confirmation checklist)**

* Both containers **Up** in docker compose ps
* Console reachable at [**http://localhost:8080**](http://localhost:8080/)
* Created **topic demo** via rpk
* Produced & consumed messages successfully
* Messages visible in Console “Messages” tab

## **Troubleshooting**

**Port already in use**

Change host ports in docker-compose.yml:

|  |
| --- |
| ports:  - "19092:9092"  - "18080:8080" |

* Then access [**http://localhost:18080**](http://localhost:18080/) and point clients to localhost:19092.
* **Console shows “cannot connect to broker”**Ensure KAFKA\_BROKERS: redpanda:9092 is set (Console must use the **container name** on the compose network, not localhost).

**Redpanda never becomes healthy**

Run logs:

|  |
| --- |
| docker logs redpanda --tail=200 |

* Make sure your machine has enough CPU/RAM and Docker has at least ~2 CPUs / 3–4 GB RAM allocated.

**Apple Silicon (M1/M2) quirks**

The official images support multi-arch; if you hit issues, update Docker Desktop and pull latest images:

|  |
| --- |
| docker compose pull && docker compose up -d --force-recreate |

## Cleanup

|  |
| --- |
| docker compose down # keep data: # volume persists for next run # remove all: docker compose down -v |

## **Extension ideas (next labs hook)**

* Add **Schema Registry** & a **Connect** container to test local connectors.
* Toggle **retention / cleanup policy** and validate in Console.
* Mount a **.env** file and compose **profiles** for optional app services.