**✅ Part 1: Multi-node Cassandra Cluster on Single Machine (Docker)**

**🔧 Prerequisites:**

* Install **Docker Desktop** on Windows
* Install **Docker Compose**

**📦 Step 1: Create a**docker-compose.yml

**Create a folder (e.g., cassandra-cluster) and add this docker-compose.yml:**

version: '3'

services:

cassandra-seed:

image: cassandra:4.1

container\_name: cassandra-seed

environment:

- CASSANDRA\_CLUSTER\_NAME=2025GRPXX

- CASSANDRA\_NUM\_TOKENS=256

- CASSANDRA\_SEEDS=cassandra-seed

- CASSANDRA\_DC=DC1

ports:

- "9042:9042"

networks:

cassandra\_net:

cassandra-node1:

image: cassandra:4.1

container\_name: cassandra-node1

environment:

- CASSANDRA\_CLUSTER\_NAME=2025GRPXX

- CASSANDRA\_SEEDS=cassandra-seed

- CASSANDRA\_DC=DC1

depends\_on:

- cassandra-seed

networks:

cassandra\_net:

cassandra-node2:

image: cassandra:4.1

container\_name: cassandra-node2

environment:

- CASSANDRA\_CLUSTER\_NAME=2025GRPXX

- CASSANDRA\_SEEDS=cassandra-seed

- CASSANDRA\_DC=DC1

depends\_on:

- cassandra-seed

networks:

cassandra\_net:

networks:

cassandra\_net:

driver: bridge

**Step 2: Start the Cluster**

Open terminal in the project directory and run:

Bash:

docker-compose up -d

**Step 3: Verify Cluster**

Enter any container (e.g., seed node):

Bash:

docker exec -it cassandra-seed cqlsh

SELECT cluster\_name FROM system.local;

nodetool status

**✅ Part 2: Install DataStax OpsCenter in Docker**

You can also run OpsCenter in Docker:

Bash:

docker run -d \

--name opscenter \

-p 8888:8888 \

-e OPSCENTER\_ACCEPT\_EULA=true \

datastax/dse-opscenter

**✅ Part 3: Weather Station Use Case**

Once the cluster is running, you can:

1. Connect to any node via cqlsh
2. Create keyspace and table (as shown before)
3. Insert sensor data manually or simulate with a Python script

**🧪 Bonus: Simulate Sensor Data in Python**

Python:

from cassandra.cluster import Cluster

from datetime import datetime, timedelta

import random

import time

cluster = Cluster(['localhost']) # or 'cassandra-seed'

session = cluster.connect('weather\_data')

station\_ids = ['WS001', 'WS002', 'WS003']

while True:

for sid in station\_ids:

temp = round(random.uniform(20.0, 40.0), 2)

session.execute("""

INSERT INTO temperature\_readings (station\_id, reading\_time, temperature)

VALUES (%s, toTimestamp(now()), %s)

""", (sid, temp))

print("Inserted data...")

time.sleep(300) # every 5 minutes

**🛠 Step 1: Create Keyspace**

Sql:

CREATE KEYSPACE weather\_data WITH REPLICATION = { 'class': 'SimpleStrategy', 'replication\_factor': 2 };

replication\_factor = 2 is good for a 3-node test cluster.

**🛠 Step 2: Use the Keyspace**

Sql:

USE weather\_data;

**🛠 Step 3: Create Table**

Sql:

CREATE TABLE temperature\_readings ( station\_id text, reading\_time timestamp, temperature float, PRIMARY KEY (station\_id, reading\_time) ) WITH CLUSTERING ORDER BY (reading\_time DESC);

This table:

* Stores data **per weather station**
* Keeps readings **in reverse chronological order**

**🧪 Step 4: Insert Sample Data**

Sql:

INSERT INTO temperature\_readings (station\_id, reading\_time, temperature) VALUES ('WS001', toTimestamp(now()), 32.5);

INSERT INTO temperature\_readings (station\_id, reading\_time, temperature) VALUES ('WS001', toTimestamp(now()), 33.2);

INSERT INTO temperature\_readings (station\_id, reading\_time, temperature) VALUES ('WS002', toTimestamp(now()), 29.1);

**Step 5: Query the Data**

Sql:

SELECT \* FROM temperature\_readings WHERE station\_id = 'WS001';