

# ACTIVIDAD 3 - COCKROACHDB

Cristóbal Suárez Abad

ADMINISTRACIÓN DE SISTEMAS GESTORES DE BASES DE DATOS - 2º ASIR

## Índice

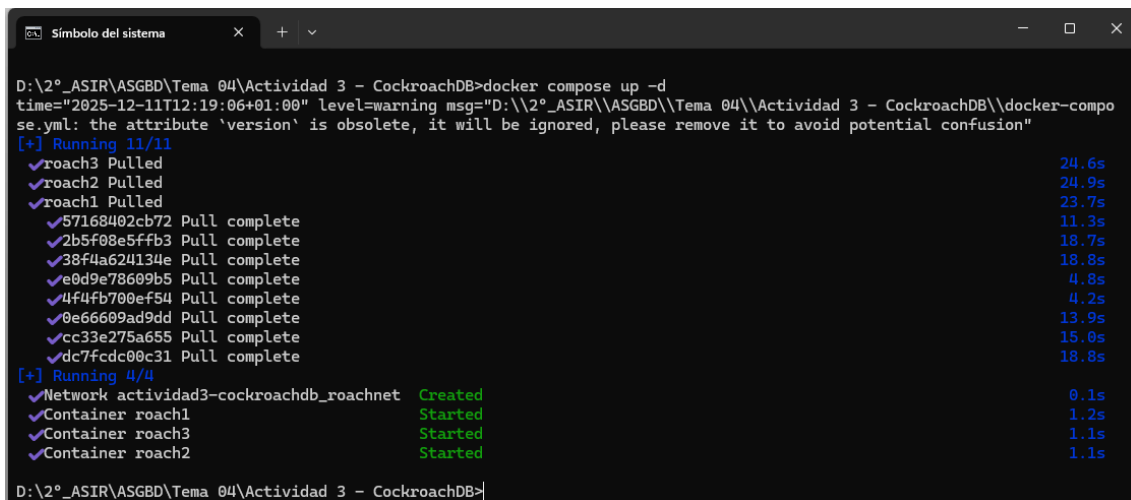
1. El escenario (Docker Compose) .....	2
2. Visualización de la Homogeneidad .....	3
3. Generación de Carga (Sharding Automático) .....	4
4. La Auditoría (Comprobar el criterio C).....	5

# Taller: Distribución de Datos en CockroachDB

## 1. El escenario (Docker Compose)

Vamos a levantar 3 nodos usando el docker compose adjunto. Fíjate que la configuración es idéntica para los tres (homogénea), solo cambia el nombre. El primer nodo expone el puerto 8080 para que podamos ver la consola web.

1. Levanta el clúster con el documento docker compose



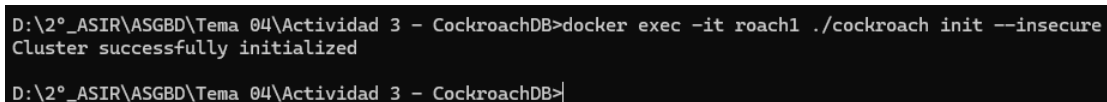
```
D:\2°_ASIR\ASGBD\Tema 04\Actividad 3 - CockroachDB>docker compose up -d
time="2025-12-11T12:19:06+01:00" level=warning msg="D:\\2°_ASIR\\ASGBD\\Tema 04\\Actividad 3 - CockroachDB\\docker-compo
se.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 11/11
  ✓ roach3 Pulled                                24.6s
  ✓ roach2 Pulled                                24.9s
  ✓ roach1 Pulled                                23.7s
  ✓ 57168402cb72 Pull complete                    11.3s
  ✓ 2b5f08e5ffb3 Pull complete                    18.7s
  ✓ 38f4a624134e Pull complete                    18.8s
  ✓ e0d9e78609b5 Pull complete                     4.8s
  ✓ 4f4fb700ef54 Pull complete                     4.2s
  ✓ 0e66609ad9dd Pull complete                    13.9s
  ✓ cc33e275a655 Pull complete                    15.0s
  ✓ dc7fcdc00c31 Pull complete                    18.8s
[+] Running 4/4
  ✓ Network actividad3-cockroachdb_roachnet Created 0.1s
  ✓ Container roach1 Started                       1.2s
  ✓ Container roach3 Started                       1.1s
  ✓ Container roach2 Started                       1.1s
D:\2°_ASIR\ASGBD\Tema 04\Actividad 3 - CockroachDB>
```

2. Inicializa **el clúster** (paso obligatorio para que los nodos se "casen"):

Bash

```
docker exec -it roach1 ./cockroach init --insecure
```

*Si dice "Cluster successfully initialized", ya tienes una BDD distribuida funcionando.*

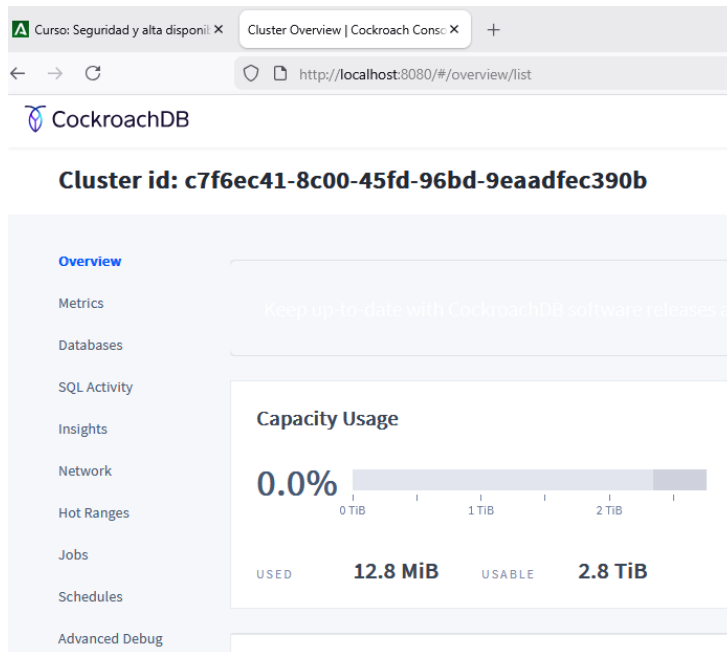


```
D:\2°_ASIR\ASGBD\Tema 04\Actividad 3 - CockroachDB>docker exec -it roach1 ./cockroach init --insecure
Cluster successfully initialized
D:\2°_ASIR\ASGBD\Tema 04\Actividad 3 - CockroachDB>
```

## 2. Visualización de la Homogeneidad

Ahora viene la parte "guau" de CockroachDB.

1. Abre tu navegador y ve a: <http://localhost:8080>



2. Verás el **Overview**.

- o Comprueba que tienes 3 nodos en el cluster.

Node List ▼

Nodes (3)

nodes ▼	Uptime ▼	Replicas ▼
roach1:26257 (n1)	a minute	63
roach2:26257 (n2)	a minute	51
roach3:26257 (n3)	a minute	51

### 3. Generación de Carga (Sharding Automático)

Para evaluar que la base de datos es **distribuida**, necesitamos ver cómo reparte los datos. Si insertamos 3 filas, se quedarán en un solo nodo. Necesitamos miles.

CockroachDB trae una herramienta interna llamada workload que genera datos de prueba (simulando una empresa de transporte tipo Uber llamada "Movr").

- Ejecuta este comando para crear la base de datos y llenarla de datos iniciales:

```
docker exec -it roach1 ./cockroach workload init movr
"postgresql://root@roach1:26257?sslmode=disable"
```

```
D:\2°_ASIR\ASGBD\Tema 04\Actividad 3 - CockroachDB>docker exec -it roach1 ./cockroach workload init movr "postgresql://root@roach1:26257?sslmode=disable"
I251211 11:21:45.427255 1 workload/cli/run.go:624 [-] 1 random seed: 10865267595032214388
I251211 11:21:45.454884 1 ccl/workloadccl/fixture.go:315 [-] 2 starting import of 6 tables
I251211 11:21:46.044118 70 ccl/workloadccl/fixture.go:492 [-] 3 imported 72 KiB in vehicle_location_histories table (1000 rows, 0 index entries, took 441.275202ms, 0.16 MiB/s)
I251211 11:21:46.126289 71 ccl/workloadccl/fixture.go:492 [-] 4 imported 218 KiB in promo_codes table (1000 rows, 0 index entries, took 523.324322ms, 0.41 MiB/s)
I251211 11:21:46.179188 68 ccl/workloadccl/fixture.go:492 [-] 5 imported 3.2 KiB in vehicles table (15 rows, 15 index entries, took 576.404035ms, 0.01 MiB/s)
I251211 11:21:46.242425 72 ccl/workloadccl/fixture.go:492 [-] 6 imported 412 B in user_promo_codes table (5 rows, 0 index entries, took 639.625673ms, 0.00 MiB/s)
I251211 11:21:46.290191 67 ccl/workloadccl/fixture.go:492 [-] 7 imported 4.8 KiB in users table (50 rows, 0 index entries, took 687.300949ms, 0.01 MiB/s)
I251211 11:21:46.385479 69 ccl/workloadccl/fixture.go:492 [-] 8 imported 153 KiB in rides table (500 rows, 1000 index entries, took 782.567016ms, 0.19 MiB/s)
I251211 11:21:46.385751 1 ccl/workloadccl/fixture.go:323 [-] 9 imported 451 KiB bytes in 6 tables (took 930.790355ms, 0.47 MiB/s)
I251211 11:21:46.412657 1 workload/workloadsddl/workloadsddl.go:148 [-] 10 starting 8 splits
I251211 11:21:46.599461 1 workload/workloadsddl/workloadsddl.go:148 [-] 11 starting 8 splits
I251211 11:21:46.771615 1 workload/workloadsddl/workloadsddl.go:148 [-] 12 starting 8 splits
D:\2°_ASIR\ASGBD\Tema 04\Actividad 3 - CockroachDB>
```

- Ahora, pon el clúster a trabajar duro durante 5 minutos para ver cómo mueve datos:

Bash

```
docker exec -it roach1 ./cockroach workload run movr --duration=5m
"postgresql://root@roach1:26257?sslmode=disable"
```

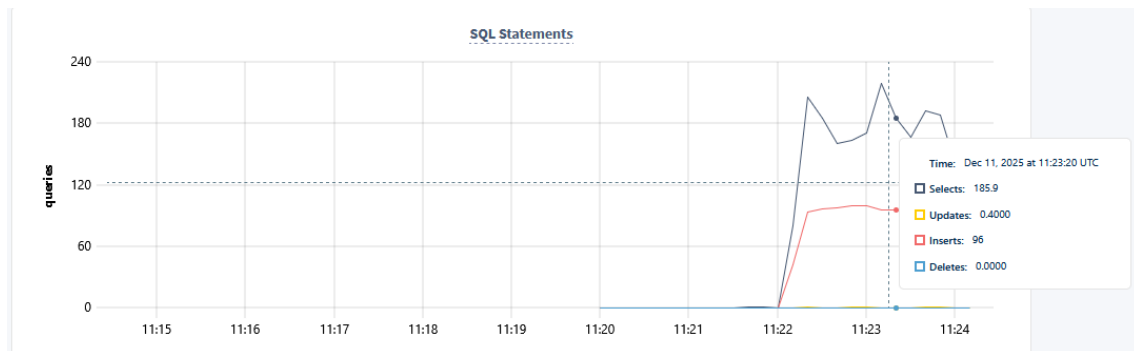
```
D:\2°_ASIR\ASGBD\Tema 04\Actividad 3 - CockroachDB>docker exec -it roach1 ./cockroach workload run movr --duration=5m "postgresql://root@roach1:26257?sslmode=disable"
I251211 11:22:12.529032 1 workload/cli/run.go:624 [-] 1 random seed: 16247298314140650057
I251211 11:22:12.529383 1 workload/cli/run.go:431 [-] 2 creating load generator...
I251211 11:22:12.529941 1 workload/cli/run.go:462 [-] 3 creating load generator... done (took 558.229µs)
   elapsed__errors__ops/sec(inst)__ops/sec(cum)__p50(ms)__p95(ms)__p99(ms)__pMax(ms)
   1.0s      0          5.0          5.0      8.1      8.9      8.9      8.9 addUser
   1.0s      0          1.0          1.0     10.5     10.5     10.5     10.5 addVehicle
   1.0s      0          1.0          1.0     18.9     18.9     18.9     18.9 createPromoCode
   1.0s      0          1.0          1.0      8.1      8.1      8.1      8.1 endRide
   1.0s      0       285.5       285.7      0.7      1.3      1.9     15.7 readVehicles
   1.0s      0          8.0          8.0     12.1     30.4     30.4     30.4 startRide
   1.0s      0         16.0         16.0     35.7     58.7     67.1     67.1 updateActiveRides
   2.0s      0          6.0          5.5      7.9     10.5     10.5     10.5 addUser
   2.0s      0          0.0          0.5      0.0      0.0      0.0      0.0 addVehicle
   2.0s      0          0.0          0.5      0.0      0.0      0.0      0.0 createPromoCode
   2.0s      0          1.0          1.0     10.5     10.5     10.5     10.5 endRide
   2.0s      0     116.1     201.0      0.7      1.0      1.4      1.5 readVehicles
   2.0s      0          4.0          6.0     12.1     12.6     12.6     12.6 startRide
   2.0s      0         12.0         14.0     67.1     75.5     88.1     88.1 updateActiveRides
```

## 4. La Auditoría (Comprobar el criterio C)

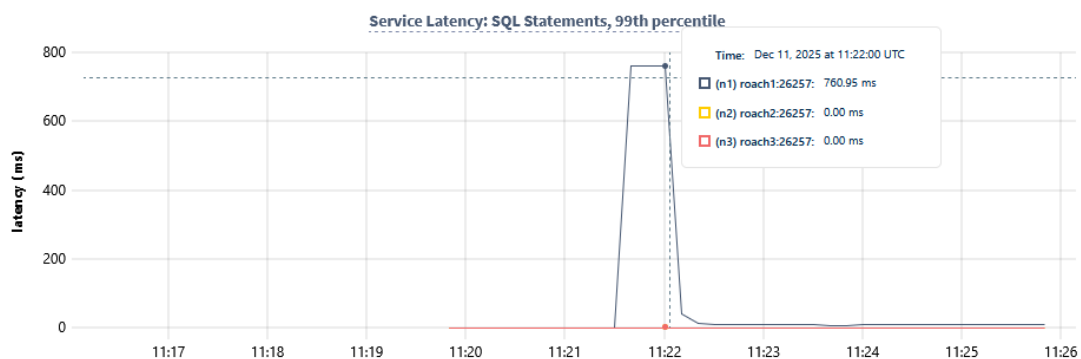
Mientras el comando anterior se está ejecutando (verás líneas de texto pasando rápido en la terminal con estadísticas), vuelve al navegador (<http://localhost:8080>).

### Pasos:

1. Ve a la pestaña "**Metrics**" (o "Overview").
2. Fíjate en las gráficas de **SQL Queries** o **Service Latency**. ¿Qué es lo que ves?
  - o Explica que ves en cada uno de los gráficos



En la gráfica se ven la cantidad de consultas que se realizan, dividiendo según "selects", "updates", "inserts" y "deletes".

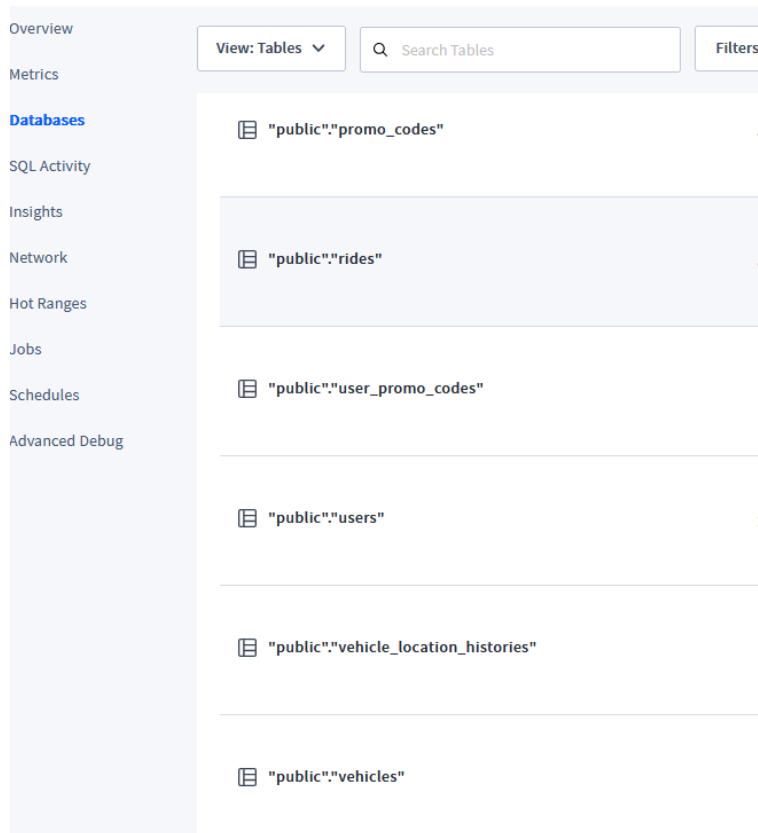


Muestra la latencia con la que este nodo ha ejecutado las consultas requeridas. En este caso un 99%.

### Service Latency: SQL Statements, 99th percentile

Over the last minute, this node executed 99% of SQL statements within this time. This time only includes SELECT, INSERT, UPDATE and DELETE statements and does not include network latency between the node and client.

3. Ve a la pestaña "**Databases**" (a la izquierda) -> Baja hasta ver las tablas (users, vehicles, etc.).



- Fíjate en la columna "**Ranges**" (Rangos). ¿Qué significa?

Son los “trozos” o “shards” en los que se dividen las tablas que permiten distribuir las de manera eficiente. Se vinculan los trozos con “key-values”<sup>1</sup>.

<sup>1</sup> <https://www.cockroachlabs.com/glossary/distributed-db/range-shard/>

4. Sigue:

- Haz clic en la pestaña "**Overview**" otra vez.
- Mira la sección "**Replicas per Node**".

### Nodes (3)

nodes ▾	Uptime ▾	Replicas ▾
roach1:26257 (n1)	14 minutes	101
roach2:26257 (n2)	14 minutes	101
roach3:26257 (n3)	14 minutes	101

- Deberías ver que los 3 nodos tienen un número similar de réplicas (ej. 15, 14, 16).

Si, tienen 101.

- *Conclusión:* El sistema ha distribuido homogéneamente los datos entre los 3 servidores sin que nosotros le dijéramos nada.