Neo4j Migration Guide: AuraDB Free → Self-Hosted Community Edition

Project Information

Field	Value
Project	Weavink - NFC Business Card Platform
Author	Leo (CTO, Weavink)
Date	November 30, 2025
Server	Hetzner CX43 (8 vCPU, 16GB RAM, 160GB SSD)
Deployment Platform	Coolify
Neo4j Version	2025.10.1 Community Edition

Table of Contents

---

1. Executive Summary

What We Did

Migrated Neo4j graph database from **AuraDB Free** (Neo4j's managed cloud service) to a **self-hosted Neo4j**

Community Edition running in Docker on our Hetzner VPS via Coolify.

Final Result

- **110 Contacts**, **50 Tags**, **190 Events**, **12 Companies** successfully migrated
- **5-7ms query latency** (self-hosted) vs **30-70ms** (AuraDB) - **6-14x faster!**
- **€0/month** vs potential paid tier when exceeding free limits
- Full control over data and configuration

Key Lesson

AuraDB backups use Enterprise Edition format and CANNOT be restored to Community Edition. You must export data using Cypher/APOC instead.

---

2. Why Self-Host Neo4j?

AuraDB Free Tier Limitations

Aspect	AuraDB Free	Self-Hosted
Nodes Limit	200,000	Unlimited (disk-limited)
Relationships	400,000	Unlimited
Cost	€0 (with limits)	€0 (included in VPS)
Latency	~30-70ms	~5ms (localhost)
Backups	1 snapshot only	Configure yourself
Data Location	Google Cloud	Your server (GDPR)

When to Migrate

- Approaching AuraDB limits (200K nodes)
- Need sub-10ms query latency
- Want full data sovereignty
- Running other services on same VPS anyway

---

3. Infrastructure Setup

Server Specifications

```
Provider: Hetzner
Model: CX43
vCPU: 8
```

```
RAM: 16GB
Storage: 160GB SSD
Location: Falkenstein, Germany (EU)
Cost: €8.99/month
```

Neo4j Resource Allocation (Current Production Config)

```
Page Cache: 4GB (caches data for fast reads)
Heap Initial: 2GB (query processing)
Heap Max: 2GB (query processing)
Total Reserved: ~6GB RAM
```

Docker Compose Configuration (Coolify)

```yaml
services:
  neo4j:
    image: 'neo4j:community'
    restart: unless-stopped
    environment:
      - NEO4J_AUTH=neo4j/YourSecurePassword123!
      - NEO4J_server_memory_pagecache_size=4G
      - NEO4J_server_memory_heap_initial__size=2G
      - NEO4J_server_memory_heap_max__size=2G
    volumes:
      - 'neo4j-data:/data'
      - 'neo4j-logs:/logs'
    healthcheck:
      test:
        - CMD
        - wget
        - '-q'
        - '--spider'
        - 'http://localhost:7474'
      interval: 30s
      timeout: 10s
      retries: 3
volumes:
  neo4j-data: null
  neo4j-logs: null
```

Ports (Internal Only)

- **7474**: HTTP Browser (internal)

- **7687**: Bolt protocol (app connections)

> ⚠️ **Security**: Keep Neo4j internal to Docker network. No public exposure needed - your app connects via internal network.

---

4. Performance Benchmarks

Test Environment

- **AuraDB**: Free tier, cloud-hosted
- **Self-Hosted**: Hetzner CX43, 4GB page cache, 2GB heap
- **Data**: 110 Contacts, 50 Tags, 190 Events, 12 Companies (~362 nodes)

Results (Warm Cache)

Query	AuraDB	Self-Hosted	Improvement
`MATCH (c:Contact)-[:WORKS_AT]->(comp:Company)`	69-72ms	**5ms**	**14x faster**
`MATCH (c:Contact) RETURN count(c)`	22-23ms	**3-4ms**	**6x faster**
`MATCH (c:Contact)-[:SIMILAR_TO]->(other:Contact)`	31-33ms	**5ms**	**6x faster**

Cold vs Warm Cache

State	Query Time	Notes
Cold (after restart)	~480ms	First query loads data into page cache
Warming	~12ms	Subsequent queries
Warm	**2-5ms**	Fully cached, production performance

Understanding Neo4j Memory

Component	Location	Purpose
Data on disk	Docker volume	Permanent storage (~542MB)
Page Cache	RAM (4GB)	Caches disk data for fast reads
Heap	RAM (2GB)	Query processing and operations

The flow: `Disk → Page Cache (RAM) → Query Results`

Once your data is in the page cache, all queries run from RAM!

---

5. The Migration Attempt That Failed

What We Tried First

1. **Downloaded AuraDB Backup**
    - Went to AuraDB Console → Instance → Snapshots
    - Downloaded `.backup` file (~100KB)
    - File: `neo4j-2025-11-30T03-58-03-9077f1b7.backup`
2. **Attempted Restore to Community Edition**

```bash
# Copied backup to server
scp ~/Downloads/neo4j-*.backup root@159.69.215.143:/root/

# Copied into container
docker cp /root/neo4j-*.backup neo4j-container:/var/lib/neo4j/backups/

# Fixed permissions
docker exec neo4j-container chown -R neo4j:neo4j /var/lib/neo4j/backups

# Attempted restore
docker exec -u neo4j neo4j-container neo4j-admin database load \
  --from-path=/var/lib/neo4j/backups \
  --overwrite-destination=true neo4j
```

The Error

```
Files: 63/63, data: 100.0%
Done: 63 files, 1.924MiB processed in 0.099 seconds.
Failed to load database 'neo4j': Block format detected for database neo4j
but unavailable in this edition.
Load failed for databases: 'neo4j'
```

Root Cause

AuraDB (even the Free tier) runs on Neo4j Enterprise Edition internally. When you export a backup, it uses the Enterprise "block format" which is **incompatible with Community Edition**.

This is NOT documented clearly by Neo4j. The backup downloads successfully, the restore runs to 100%, but fails at the final step.

What This Corrupted

The failed restore attempt left the database in a broken state:

- The `neo4j` database became unavailable
- Only the `system` database remained

- Community Edition cannot create new databases (`CREATE DATABASE` is Enterprise-only)

---

6. The Solution That Worked

The Correct Approach: APOC Cypher Export

Instead of using the binary backup, we exported all data as **Cypher statements** using APOC (A Package Of Components), which is available in AuraDB.

Why This Works

- Cypher is plain text, edition-agnostic
- Creates `CREATE` statements for all nodes and relationships
- Can be run on any Neo4j edition
- Preserves all properties and relationships

---

7. Step-by-Step Migration Guide

Phase 1: Export from AuraDB

Step 1.1: Check Your Data Structure

Connect to AuraDB via Neo4j Browser and run:

```cypher
-- List all node labels
CALL db.labels() YIELD label RETURN label

-- List all relationship types
CALL db.relationshipTypes() YIELD relationshipType RETURN relationshipType
```

Our results:

- Labels: Contact, Company, Tag, Event
- Relationships: WORKS_AT, HAS_TAG, SIMILAR_TO, KNOWS, ATTENDS, MATCHED_AT

Step 1.2: Export Using APOC

Run this query in AuraDB Browser:

```cypher
```

```
CALL apoc.export.cypher.all(null, {format: "plain", stream: true})
YIELD cypherStatements
RETURN cypherStatements
```

Step 1.3: Download as CSV

1. Click the **Export** button in Neo4j Browser
2. Choose **CSV** format
3. Save as `neo4j_export.csv`

The CSV contains all your data as executable Cypher statements.

Phase 2: Deploy Neo4j Community Edition

Step 2.1: Create Neo4j Service in Coolify

1. Go to **Coolify** → Your Project → **+ Add Resource**
2. Select **Docker Compose** (NOT Dockerfile)
3. Paste the docker-compose configuration from Section 3
4. Click **Save** and **Deploy**

Step 2.2: Verify Deployment

```bash
# SSH to server
ssh root@159.69.215.143

# Check container is running
docker ps | grep neo4j

# Check logs
docker logs neo4j-container-name --tail 20
```

You should see:

```
INFO  Bolt enabled on 0.0.0.0:7687.
INFO  HTTP enabled on 0.0.0.0:7474.
INFO  Started.
```

Phase 3: Prepare Import File

Step 3.1: Transfer CSV to Server

```bash
# From your local machine
scp ~/Downloads/neo4j_export.csv root@159.69.215.143:/root/
```

Step 3.2: Convert CSV to Cypher Script

The CSV has a header and quoted content. Clean it up:

```bash
# SSH to server
ssh root@159.69.215.143

# Convert CSV to clean Cypher file
tail -n +2 /root/neo4j_export.csv | sed 's/^"//;s/"$//' | sed 's/""/"/g' > /root/neo4j_import.cypher

# Verify the file looks correct
head -5 /root/neo4j_import.cypher
```

You should see clean Cypher statements:

```cypher
CREATE RANGE INDEX FOR (n:Contact) ON (n.userId);
CREATE CONSTRAINT company_name FOR (node:Company) REQUIRE (node.name, node.userId) IS UNIQUE;
...
```

Phase 4: Import Data

Step 4.1: Copy Import File to Container

```bash
docker cp /root/neo4j_import.cypher neo4j-container-name:/var/lib/neo4j/
```

Step 4.2: Run the Import

```bash
```

```
docker exec -it neo4j-container-name cypher-shell \
  -u neo4j \
  -p 'YourSecurePassword123!' \
  -f /var/lib/neo4j/neo4j_import.cypher
```

Step 4.3: Verify Import

```bash
# Count all nodes by type
docker exec -it neo4j-container-name cypher-shell \
  -u neo4j -p 'YourSecurePassword123!' \
  "MATCH (n) RETURN labels(n) AS type, count(*) AS count"

# Sample some data
docker exec -it neo4j-container-name cypher-shell \
  -u neo4j -p 'YourSecurePassword123!' \
  "MATCH (c:Contact) RETURN c.name, c.company LIMIT 5"
```

Expected output:

```
+--------------------+
| type        | count |
+--------------------+
| ["Contact"] | 110   |
| ["Tag"]     | 50    |
| ["Event"]   | 190   |
| ["Company"] | 12    |
+--------------------+
```

---

8. Post-Migration Configuration

Update Weavink Environment Variables

In Coolify, update your Weavink app's environment variables:

Before (AuraDB):

```env
NEO4J_URI=neo4j+s://xxxx.databases.neo4j.io
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=your-aura-password
```

After (Self-hosted):

```env
NEO4J_URI=bolt://neo4j-ws4www84wss8scck0g0cow04:7687
NEO4J_USERNAME=neo4j
NEO4J_PASSWORD=YourSecurePassword123!
```

> **Note**: Use `bolt://` not `neo4j+s://` for local connections. The container name is the hostname within Docker's network.

Redeploy Weavink

After updating environment variables, redeploy the Weavink application in Coolify.

Verify Memory Configuration

After any configuration change, verify settings:

```bash
docker exec neo4j-ws4www84wss8scck0g0cow04 cat /var/lib/neo4j/conf/neo4j.conf | grep memory
```

Expected output:

```
server.memory.pagecache.size=4G
server.memory.heap.max_size=2G
server.memory.heap.initial_size=2G
```

---

9. Troubleshooting Reference

Problem: "Block format detected but unavailable in this edition"

Cause: Trying to restore Enterprise backup to Community Edition

Solution: Use APOC Cypher export instead of binary backup

Problem: "Unable to get a routing table for database 'neo4j'"

Cause: Database corrupted or not created

Solution: Reset the data volume:

```bash
```

```
# Stop container
docker stop neo4j-container-name

# Remove container (so volume can be deleted)
docker rm neo4j-container-name

# Find volume name
docker volume ls | grep neo4j

# Remove volume
docker volume rm volume-name-here

# Redeploy from Coolify
```

Problem: "CREATE DATABASE is not supported in community edition"

Cause: Community Edition only supports one database

Solution: Don't try to create databases. Use the default `neo4j` database.

Problem: "Connection refused" when connecting to cypher-shell

Cause: Neo4j still starting up

Solution: Wait 20-30 seconds after container start:

```bash
sleep 30
docker logs neo4j-container-name --tail 10
# Should see "Started." before connecting
```

Problem: Permission errors with neo4j user

Cause: Running commands as root but Neo4j runs as `neo4j` user

Solution: Use `-u neo4j` flag or fix permissions:

```bash
```

```
# Run as neo4j user
docker exec -u neo4j neo4j-container-name neo4j-admin ...

# Or fix permissions
docker exec neo4j-container-name chown -R neo4j:neo4j /var/lib/neo4j/
```

Problem: Slow first query after restart

Cause: Cold cache - data needs to load from disk to page cache

Solution: This is normal. First query may take 400-500ms. Subsequent queries will be 2-5ms once cache is warm. Your app's regular queries will keep the cache warm.

---

10. Maintenance Commands

Daily Operations

```bash
# Check container status
docker ps | grep neo4j

# View recent logs
docker logs neo4j-ws4www84wss8scck0g0cow04 --tail 50

# Check database health
docker exec -it neo4j-ws4www84wss8scck0g0cow04 cypher-shell \
  -u neo4j -p 'YourSecurePassword123!' \
  "CALL dbms.components() YIELD name, versions RETURN name, versions"
```

Check Disk Usage

```bash
# Check data volume size
docker system df -v | grep neo4j

# Or check inside container
docker exec neo4j-ws4www84wss8scck0g0cow04 du -sh /data
```

Backup (Manual)

```bash
```

```
# Create a Cypher export (same method we used to migrate)
docker exec -it neo4j-ws4www84wss8scck0g0cow04 cypher-shell \
  -u neo4j -p 'YourSecurePassword123!' \
  "CALL apoc.export.cypher.all('/var/lib/neo4j/backup.cypher', {})"

# Copy backup out of container
docker cp neo4j-ws4www84wss8scck0g0cow04:/var/lib/neo4j/backup.cypher /root/backups/
```

> **Note**: APOC may not be installed in Community Edition by default. You may need to add the APOC plugin to your Docker configuration.

Query Statistics

```bash
# Count all nodes
docker exec -it neo4j-ws4www84wss8scck0g0cow04 cypher-shell \
  -u neo4j -p 'YourSecurePassword123!' \
  "MATCH (n) RETURN count(n) AS totalNodes"

# Count all relationships
docker exec -it neo4j-ws4www84wss8scck0g0cow04 cypher-shell \
  -u neo4j -p 'YourSecurePassword123!' \
  "MATCH ()-[r]->() RETURN count(r) AS totalRelationships"

# Database size (approximate)
docker exec neo4j-ws4www84wss8scck0g0cow04 du -sh /data
```

Restart Neo4j

```bash
# Via Docker
docker restart neo4j-ws4www84wss8scck0g0cow04

# Via Coolify
# Go to Coolify UI → Project → Neo4j → Restart
```

---

11. Scaling Guide

Current Resource Usage

Resource	Used	Allocated
Disk	~542MB	160GB

Resource	Used	Allocated
Page Cache	~542MB	4GB
Heap	Variable	2GB

Capacity Planning

Users	Est. Nodes	Est. Disk	Fits in 4GB Cache?
1	~362	~542MB	✅ Yes
5-6	~2,000	~1-1.5GB	✅ Yes
50	~18,000	~5-8GB	⚠️ Partially
200	~72,000	~20-30GB	❌ Need more RAM

Memory Configuration Recommendations

User Count	Page Cache	Heap	Total RAM
1-50	4GB	2GB	~6GB
50-100	6GB	2GB	~8GB
100-200	8GB	3GB	~11GB

How to Change Memory

Update docker-compose in Coolify:

```yaml
environment:
  - NEO4J_server_memory_pagecache_size=6G   # Increase for more data caching
  - NEO4J_server_memory_heap_initial__size=2G
  - NEO4J_server_memory_heap_max__size=2G
```

Then **Save** and **Redeploy**.

When to Upgrade Server

If your data exceeds 10GB and you need fast queries, consider:

- Hetzner CX43 → CX53 (16GB → 32GB RAM, ~€18/month)
- Or optimize data model to reduce node/relationship count

---

12. Lessons Learned

1. AuraDB Backups ≠ Community Compatible

The binary `.backup` files from AuraDB use Enterprise Edition format. **Always use Cypher/APOC export for cross-edition migration.**

2. Test Before Production

We should have tested the restore on a local Docker instance before attempting on production server.

3. Community Edition Limitations

- Single database only (no `CREATE DATABASE`)
- No clustering
- Single `neo4j` user
- Some APOC procedures unavailable

4. Container Naming in Coolify

Coolify generates container names like `neo4j-ws4www84wss8scck0g0cow04`. Use `docker ps | grep neo4j` to find it.

5. Connection String Differences

- AuraDB: `neo4j+s://` (encrypted)
- Self-hosted local: `bolt://` (internal network, no encryption needed)

6. Failed Imports Corrupt Data

A failed import can leave the database in a broken state. Only solution is to delete the data volume and start fresh.

7. Cache Warmup is Normal

First query after restart may take 400-500ms. This is normal - Neo4j is loading data into the page cache. Regular app traffic keeps the cache warm.

8. Self-Hosted is Much Faster

With proper configuration, self-hosted Neo4j is **6-14x faster** than AuraDB due to:

- Zero network latency (localhost vs internet round-trip)
- Dedicated resources (no multi-tenant overhead)
- Data fits entirely in page cache

Quick Reference Card

Container Name

```bash
docker ps | grep neo4j
# Current: neo4j-ws4www84wss8scck0g0cow04
```

Connect to Database

```bash
docker exec -it neo4j-ws4www84wss8scck0g0cow04 cypher-shell -u neo4j -p 'YourSecurePassword123!'
```

Connection String for App

```
bolt://neo4j-ws4www84wss8scck0g0cow04:7687
```

Server Details

```
IP: 159.69.215.143
SSH: ssh root@159.69.215.143
```

Check Memory Config

```bash
docker exec neo4j-ws4www84wss8scck0g0cow04 cat /var/lib/neo4j/conf/neo4j.conf | grep memory
```

Check Disk Usage

```bash
docker system df -v | grep neo4j
```

Useful Cypher Queries

```cypher
```

```
-- Count all nodes
MATCH (n) RETURN labels(n), count(*)

-- List all indexes
SHOW INDEXES

-- List all constraints
SHOW CONSTRAINTS

-- Test query performance (run twice for warm cache)
MATCH (c:Contact)-[:SIMILAR_TO]->(other:Contact) RETURN c.name, other.name LIMIT 20;
```

Document History

Date	Version	Changes
2025-11-30	1.0	Initial migration and documentation
2025-11-30	1.1	Updated memory config (4GB page cache, 2GB heap), added performance benchmarks, scaling guide

---

Document created after successful migration from AuraDB Free to self-hosted Neo4j Community Edition on Hetzner VPS via Coolify.