**TU Dublin TU856/TU857/TU858**

**Advanced Databases**

Replication in CouchDB

Cassandra DB Setup

[Introduction 1](#_Toc178154732)

[Step 0 Only for those working in Codespaces 2](#_Toc178154733)

[Step 1 Create a Cassandra Cluster 2](#_Toc178154734)

[Step 2 Check the status of the cluster 3](#_Toc178154735)

[Step 3 Connect to a node. 4](#_Toc178154736)

[Step 4 Create a Keyspace. 5](#_Toc178154737)

[Step 5 Create a Table and Insert Some Data 6](#_Toc178154738)

# Introduction

In this lab you are going to:

* Create a small Cassandra cluster using docker containers.
* Working with CQLSH
  + Create a keyspace.
    - Create a table
    - Insert some data
    - Query this data

## Step 0 Only for those working in Codespaces

**If you are working in Codespaces** change your machine type to be 4 core with 16GB of RAM before adding the Cassandra Cluster.

A screenshot of a computer

Description automatically generated

## Step 1 Create a Cassandra Cluster

* You are going to create a cluster with 3 nodes.
  + We are mimicking a server cluster with servers on notional racks.
* Download the docker compose file **docker-cassandra-compose.yml** from Brightspace.
* Open and examine it in an editor:
  + It is configuring three Cassandra services – cassandra1, cassandra2, cassandra-3
  + You will see that as part of the service configuration you set the MAX\_HEAP\_SIZE and HEAP\_NEWSIZE parameters.
    - These control the Java Virtual Machine (JVM) memory allocation for Cassandra, which is written in Java. These parameters help optimize memory usage and performance.
    - The heap is the portion of memory where Java objects are allocated.
    - MAX\_HEAP\_SIZE- sets the maximum amount of memory (heap size) that the JVM will allocate for the Cassandra process. By setting MAX\_HEAP\_SIZE to 512M, you are limiting Cassandra to use no more than 512 MB of memory for storing objects, such as data structures, buffers, etc
    - HEAP\_NEWSIZE - sets the size of the "young generation" or "new generation" heap space ( where short-lived objects are stored (e.g., temporary objects)) within the total heap space.
* Change the password under the environment section for Cassandra1 to be whatever you want it to be.
* Bring the containers up (use the docker compose up or restart command) and you should get three Cassandra containers running.
* docker ps -a or view in Docker Desktop to verify

## Step 2 Check the status of the cluster

* **Before you try to connect – make sure all three nodes are up and running**
* You can check the status of your Cassandra containers using the **nodetool status command** (an Apache Cassandra command).
* You can check the status of your node by executing the following command in the command window:

docker exec -it yourcontainername nodetool status

e.g.

docker exec -it cassandra1 nodetool status

A computer screen shot of a computer code

Description automatically generated

Here you can see that the three nodes are up and running on IP addresses 172.20.0.2, 172.20.0.3 and 172.20.0.4 and all are **up and normal (status UN).**

**Note: This may take a couple of minutes even though they are appearing in docker ps or Docker Desktop.**

**Node Status Indicators:**

Each node is represented with two key indicators:

**Status**: This shows whether the node is currently up or down.

U: The node is Up and running.

D: The node is Down, meaning it is not reachable or functioning.

**State**: This shows whether the node is part of the cluster or not.

N: The node is Normal, meaning it is a functioning member of the cluster and holding data.

L: The node is Leaving the cluster (it is in the process of decommissioning or being removed).

J: The node is Joining the cluster (it is in the process of starting up and joining the cluster for the first time).

M: The node is Moving data, meaning it's either moving data to or from another node in the cluster (usually due to a rebalancing or a topology change).

**Tokens** represent the range of the hash space (data partition space) that a node is responsible for in the cluster. Cassandra uses consistent hashing to distribute data across multiple nodes. Each node is assigned one or more tokens, which define the portion of the data that node is responsible for.

The **Owns** column shows the **percentage of the token space** that the node is responsible for. Represents the effective ownership of the data based on the token ranges assigned to the node. It indicates how much of the total data (key space) in the cluster that particular node owns.

## Step 3 Connect to a node.

* **Important!** Before you start this step make sure your all your Cassandra nodes are up and normal.
* Launch CQLsh – in the terminal enter the following command:

docker exec -it yourFIRSTcontainername bash -c 'cqlsh'

e.g. docker exec -it cassandra1 bash -c 'cqlsh'

* You should get a cqlsh> prompt. For example:

A black screen with white text

Description automatically generated

**Note1**: If you are working on your own machine you can connect using DBeaver:

* Create a New Connection (NoSQL and Apache Cassandra)
  + If working on the community edition you will need to download and install the driver.
* Username:cassandra Password: your password
* It will connect by default to the system keyspace – you can create a new one via the client.
* To launch a console to enter CQL commands – Open New SQL console.

**Note2:** If you are working in Codespaces the Database client extension will work for you also <https://marketplace.visualstudio.com/items?itemName=cweijan.vscode-database-client2>

## Step 4 Create a Keyspace.

* In Cassandra data is stored in *tables*, whose schema defines the layout of the data in the table.
* Tables are located in *keyspaces*.
  + A keyspace defines options that apply to all the keyspace’s tables
* To create a keyspace
  + At the cqlsh prompt (or SQL Script window in DBeaver)
  + Create a keyspace which uses a simple replication strategy and a replication factor of 2 (we will talk about this in the lecture later):

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

* Note: If you are cutting and pasting into cqlsh you need to right-click)
  + Since we have a single data center and a single rack, we adopt the simple strategy.
  + A replication factor of 2 means there are two copies of all data.

## Step 5 Create a Table and Insert Some Data

* Use the Keyspace - at the cqlsh prompt:

Use advanceddb;

**Note**: The prompt should now reflect that the keyspace is being used.

* Create a table within this keyspace:

CREATE TABLE employee (emp\_id int PRIMARY KEY, name text, city text);

* This will create a table called employee in the keyspace advanceddb and will use the emp\_id column for partitioning.
* To verify it worked describe the table:

DESC employee;

A close-up of a computer screen

Description automatically generated

* Insert some data:

INSERT INTO employee (emp\_id , name , city ) VALUES (1, 'John' , 'Dublin' ) ;

INSERT INTO employee (emp\_id , name , city ) VALUES (2, 'Mimi' , 'Cork' ) ;

INSERT INTO employee (emp\_id , name , city ) VALUES (3, 'Annie' , 'Wexford' ) ;

* Verify the data by executing a select statement:

SELECT \* FROM employee;

* Open a new terminal and execute the following nodetool command on your first Cassandra node:
* E.g.

docker exec -it YOURcontainername nodetool tablestats advanceddb.employee

e.g.

docker exec -it cassandra1 nodetool tablestats advanceddb.employee

This will provide statistics about the employee table. Details of what the output provides is detailed here: <https://docs.datastax.com/en/cassandra-oss/3.x/cassandra/tools/toolsTablestats.html>

Key pieces of information:

* Memtable data size – how much data is stored in memtable
* Local read latency - time in milliseconds to complete the most recent request to read the table
* Local write latency – time in milliseconds to complete the most recent write request
* Index summary off heap memory used – amount of off heap memory used to store indexes for this table
* Bloom filter space used – how much space is used to store the Bloom filter data
* Bloom filter false positives ratio - fraction of all bloom filter checks resulting in a false positive from the most recent read.
  + We will talk about the Bloom filter during the lecture.