# Distributed Systems
## Cassandra Tutorial

Cristian Klein, Ewnetu Bayuh, Francisco Hernandez-Rodriguez
2013-12-12

This tutorial will allow you to acquire hands-on experience with Cassandra, building on the concepts introduced during the lecture part. The exercises will help you understand the trade-offs between different replication strategies and consistency levels.

At the core of the tutorial, we shall set up a Cassandra cluster. For your convenience, this tutorial has been designed to emulate a Cassandra cluster of 3 nodes on a single physical machine. This works by binding each Cassandra node to a different "localhost" IP address: Node 1 binds on 127.0.1.1, node 2 on 127.0.1.2 and node 3 on 127.0.1.3. Scripts are given to you for starting and stopping a node, so that you can focus on the rest of the tutorial.

You will encounter some questions in this tutorial. Answering them is optional, however, is highlight advised to make sure you properly understood the tutorial. In case you encounter difficulties, do not hesitate to ask any teaching assistant for help.

Note that the tutorial has been designed and tested for Linux. The tutorial is divided in three parts: setting up, Cassandra Query Language (CQL), understanding consistency levels.

# 1 Setting up

**NOTE:** The tutorial requires 30 MB of disk space. Make sure you have enough quota available in your home directory.

We recommend you to use 4 terminal for this tutorial: 1 for the client (TermA) and 1 for each of the 3 nodes (TermB, TermC, TermD).

1. Download the tutorial kit located at:
   `http://www8.cs.umu.se/~cklein/teaching/cassandra-tutorial.tar.gz`
2. Decompress it in a convenient location, for example, your home directory.
   `tar -xzvf cassandra-tutorial.tar.gz`
   and change the current directory of **all** open terminals to it:
   `cd cassandra-tutorial`
3. In separate terminals, start each node:
   TermB: `./start.sh 1`
   TermC: `./start.sh 2`

TermD: `./start.sh 3`
Note that, a node can be killed by typing CTRL+C in its terminal.

4. Check the status of the nodes:
   TermA: `./apache-cassandra-2.0.1/bin/nodetool status`
   The output of the command should indicate that all nodes are up and normal (UN).

# 2 Cassandra Query Language

In this part of the tutorial, we focus on the Cassandra Query Language (CQL). All nodes are supposed to be up and all commands are issued in TermA.

1. Cassandra requires Java 7. Therefore, we first have to make sure the `JAVA_HOME` environment variable point to the right directory as follows:

   ```
   setenv JAVA_HOME /usr/lib/jvm/java-7-openjdk-amd64
   ```

2. Start CQL shell and connect it to the first node:
   `./apache-cassandra-2.0.1/bin/cqlsh 127.0.1.1`

3. Create a new keyspace with replication factor 3:

   ```
   CREATE KEYSPACE mykeyspace
       WITH REPLICATION = { 'class' : 'SimpleStrategy',
       'replication_factor' : 3 };
   ```

4. Use it as the active keyspace:

   ```
   USE mykeyspace;
   ```

5. Create a new table:

   ```
   CREATE TABLE users (
       user_id int PRIMARY KEY,
       firstName text,
       lastName text
   );
   ```

6. Insert some data into the table. Use the following examples, but be creative and write your own commands:

   ```
   INSERT INTO users (user_id, firstName, lastName)
       VALUES (1, 'John', 'Smith');
   INSERT INTO users (user_id, firstName, lastName)
       VALUES (2, 'Jane', 'Smith');
   INSERT INTO users (user_id, firstName, lastName)
       VALUES (3, 'Lars', 'Larsson');
   ```

7. Update some data, using the following statement as example (again, be creative):

   ```
   UPDATE users SET firstName='Karl' where user_id = 1;
   ```

8. To retrieve the data you entered so far, type:

   ```
   SELECT * FROM users;
   ```

9. Now try different queries on your data:

   ```
   SELECT * FROM users WHERE user_id = 2;
   SELECT * FROM users WHERE lastName = 'Smith';
   ```

   **Q1.** Did all queries succeed? What was the query that gave the error? What error did you get?

10. To allow queries for other fields than the primary key, you need to add indexes as follows:

    ```
    CREATE INDEX ON users(firstName);
    ```

    Add indexes for both non-primary fields, first name and last name, and retry the queries you entered during the previous step.

# 3 Consistency Levels

In this part of the tutorial, we shall highlight the interaction between consistency levels and fault tolerance.

1. Set consistency level to 3 in CQL shell, as follows:

   ```
   CONSISTENCY THREE;
   ```

2. Let us assume that node 3 becomes inaccessible, either due to machine failure or network partitioning. Kill node 3 by pressing CTRL+C in TermD.

3. Run a few read (SELECT) and write (INSERT and UPDATE) queries from the previous steps.

   **Q2.** Did the queries succeed? What error did you get?

4. Set consistency level to 2:

   ```
   CONSISTENCY TWO;
   ```

5. Run a few read and write queries. Make sure you remember the changes that you have made (e.g., I added a new user "Jenny" and "Jake").

   **Q3.** Did the queries succeed?

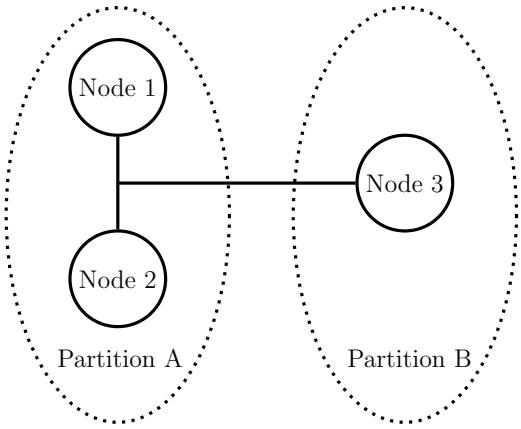


Figure 1: Diagram of network failure

Let us study how Cassandra behaves with respect to network partitions. Assume that in the previous steps, node 3 became inaccessible due to network partitioning, as shown in Figure 1. Since we issued commands to node 1, we essentially saw what happened in partition A. Now we shall look at the database from partition B's perspective.

1. Exit CQLSH in TermA by typing `QUIT` and kill all the nodes by pressing CTRL+C in TermB and TermC.
2. Start node 3 in TermD and connect to it using the following command (note that the IP address is different than the previous invocation of CQLSH):

   ```
   ./apache-cassandra-2.0.1/bin/cqlsh 127.0.1.3
   ```

   Note that since we restarted CQLSH, the consistency level is now ONE, the default.
3. Select the right keyspace (using the `USE` command) and list all users.

   **Q4.** Are all the users you previously inserted present? Does that database look like what you saw in partition A?
4. Start node 1 in TermB and list all users again.

   **Q5.** How about now? Are all users listed? What happened?

Let us now study how write conflicts are dealt with by Cassandra.

1. Kill node 1.
2. Insert a new user with the ID 100. Note that, only node 3 is aware of this new user.
3. Exit CQLSH and kill node 3.
4. Start node 1 and connect CQLSH to it.
5. Insert a new user with the ID 100, but with a **different** first or last name.

6. Start node 3 and list all users.

   **Q6.** What do you observe? Which version of the user with ID 100 is retained?

Feel free to do more experiments by yourself. You can combine reading and writing to the database, setting different consistency levels, starting and killing Cassandra nodes, and observing the results.

**Answer Sheet** Name: _____