Installation guide

Advanced Topics on NoSQL databases

A4 - S8

ESILV
nicolas.travers (at) devinci.fr

Chapter 1 Cassandra

Cassandra is a NoSQL database server based on a distributed hash-table (DHT), inspired from DynamoDB (Amazon). It has been designed by Facebook and now as an Apache license, supported by DataStaX.

1.1 Installation

To install it for the practice work, you have 2 different strategies: Virtualizatin (easier), software installation (harder).

⚠It is highly recommended that you install it with **Docker** in order that you do not loose time with installation.

1.1.1 Cassandra container with Docker

Remind that you have already followed a course on docker.
You can install it here: `https://www.docker.com/get-started`

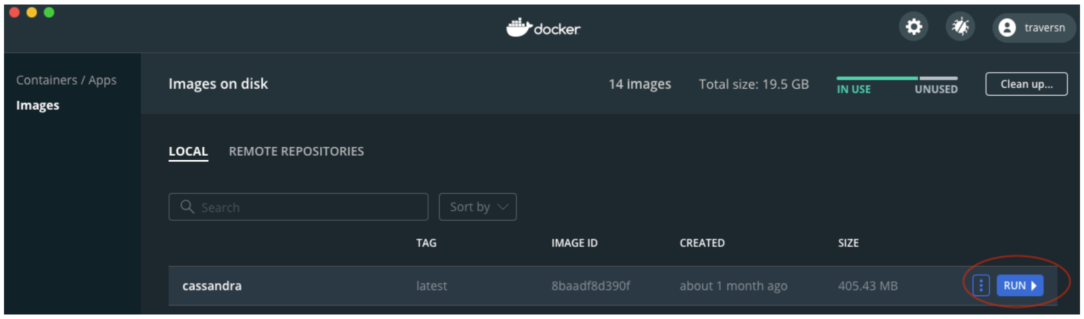
1.1.1 Launch *Docker machine* (desktop version is recommended)

1.1.2 Open *command line* (console) and pull last cassandra image:

```
docker pull cassandra
```

1.1.3 Open *Docker dashboard*

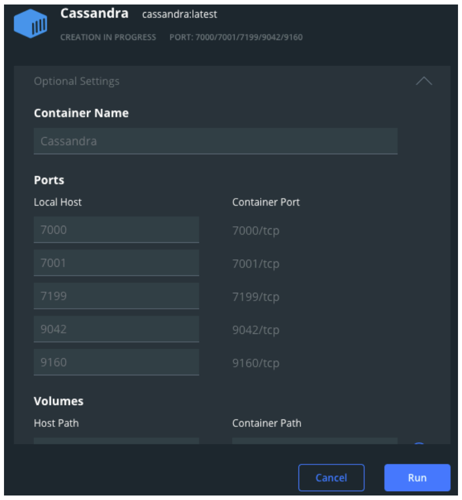
- Show downloaded images and run an instance



Images on disk   14 images   Total size: 19.5 GB   IN USE   UNUSED

	TAG	IMAGE ID	CREATED	SIZE
cassandra	latest	8baadf8d390f	about 1 month ago	405.43 MB

- Modify the optional settings to define port redirection:



Cassandra cassandra:latest
CREATION IN PROGRESS PORT: 7000/7001/7199/9042/9160

Optional Settings

Container Name: Cassandra

Ports

Local Host	Container Port
7000	7000/tcp
7001	7001/tcp
7199	7199/tcp
9042	9042/tcp
9160	9160/tcp

Volumes

Host Path   Container Path

- Run the container, the tab "Containers/Apps" will open with a green box for Cassandra. It's running!

1.1.4 Now you can connect the database with a UI (here *TabePlus* in Section 1.2)

A more complete guide for Docker with OpsCenter:
`https://devinci-online.brightspace.com/d2l/le/lessons/43258/topics/137860`

1.1.2 Native Cassandra on your laptop

⚠*Remind that Cassandra is initially planned for a cloud deployment and not on a laptop. That is why you should have some issues during this type of installation.*

Online tutorials

You can find some online tutorials:

- Linux: `http://accel-archives.intra-mart.jp/2014-winter/document/iap/public_en/imbox/cassandra_administrator_guide/texts/start-stop/index.html`
- Datastax guide: `https://devinci-online.brightspace.com/d2l/le/lessons/43258/topics/137842`
- Online guide for windows:
  `http://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-windows-the-easy-way`
- Online guide for Mac:
  - With DataStax:
    `http://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-windows-the-easy-way`
  - With Python "*pip install*" :
    `https://dbglory.wordpress.com/2015/02/22/installing-cassandra-on-mac-os-x/`
  - With Homebrew "*brew install cassandra*":
    `https://code2bits.com/how-to-install-cassandra-on-macos-using-homebrew/`

Server Installation

The **JDK 1.8** Java version is necessary to make Cassandra working:
`http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html`
Of course, you need to set up your environment with the `JAVA_HOME`:
`https://docs.oracle.com/cd/E19182-01/820-7851/inst_cli_jdk_javahome_t/`

On Windows

On Windows, Datastax has decided not to up-to-date the last version since 2016. So you have a stable version for Windows (if you have PowerShell). You need to download the server: `https://cassandra.apache.org/download/`
A detailed guide can be found here:
`http://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-windows-the-easy-way`

To install and launch the server:

1.1.1 Launch the MSI file,

1.1.2 Install Cassandra under `C:\Program Files\DataStax-DDC\` ;

1.1.3 To launch the server: `C:\Program Files\DataStax-DDC\apache-cassandra\bin\cassandra.bat`
⚠Do not forget the access rights and the JAVA_HOME.

Since a database is a **server** (not a UI) you need to launch it each time you need it, and never switch it off (until end of use). So, do not close the console.

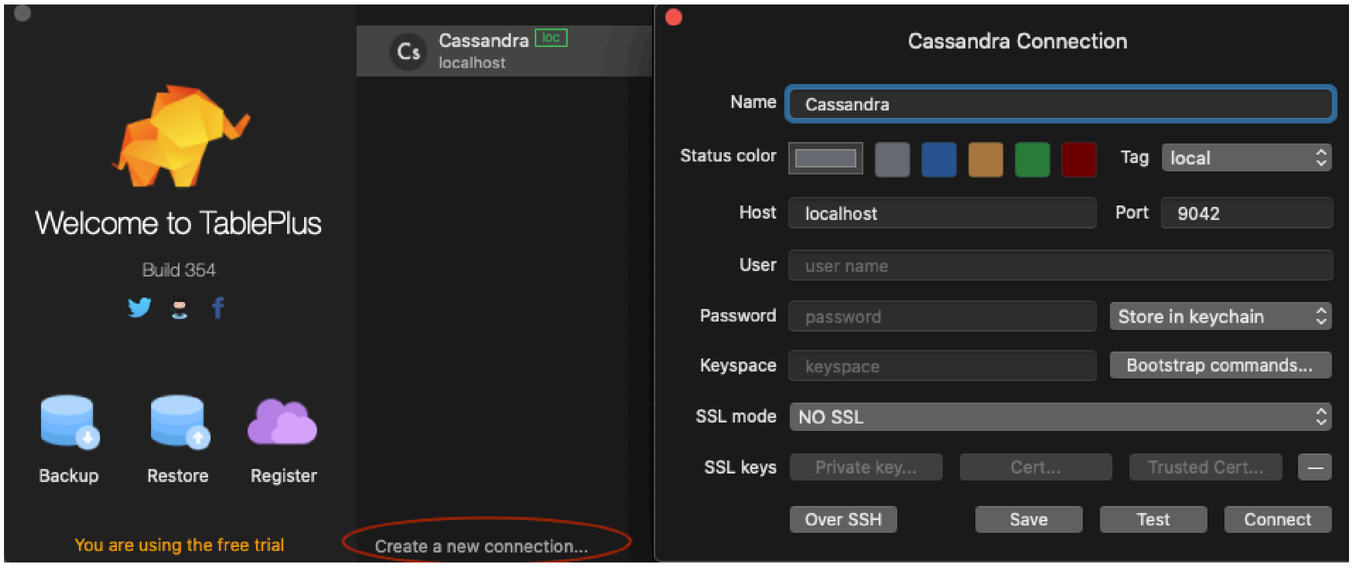
For linux

Cassandra binaries and instructions: `https://cassandra.apache.org/download/`

1.2 User Interface for Cassandra: TablePlus

To access the cassandra server, we need a UI. `TablePlus` is a new software to connect to several types of databases (including Cassandra): `https://tableplus.com/download`

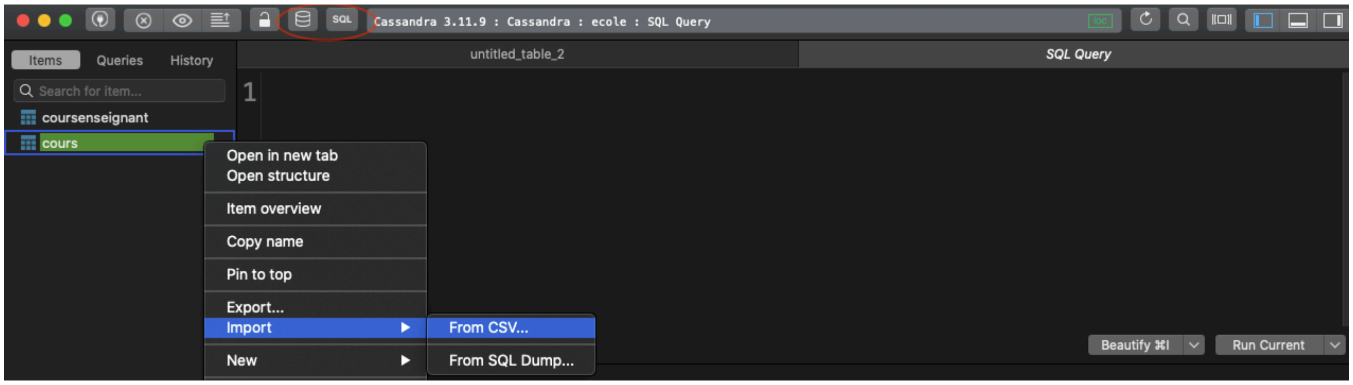
Once installed, create a connection to the server. The default port is **9042**.



Name	Cassandra
Status color	
Tag	local
Host	localhost
Port	9042
User	user name
Password	password
	Store in keychain
Keyspace	keyspace
	Bootstrap commands...
SSL mode	NO SSL
SSL keys	Private key... Cert... Trusted Cert...

You can select or create a `Keyspace` (database icon), insert rows in the database, and execute CQL queries. The SQL icon allows to manipulate the SQL console and execute queries.

You can import CSV data (not JSON documents) in a created table, like in the following:


Cassandra 3.11.9 : Cassandra : ecole : SQL Query

1.3 JSON data import for your dataset (report step)

As you know, you have to import your dataset in Cassandra for your report (and explain how you did it). I know that it's a complicate step since Cassandra has been built to import CSV files or SQL scripts. Here are steps you should follow (not mandatory):

- Create a schema
  - New TABLE in cassandra which corresponds to your dataset (look at your JSON documents and integrate each feature)
  - Choose well your nest types: SET, LIST, MAP, TYPE, even TUPLE (will be appreciated)
  - You can try several tables to try several types of queries
- Take the first line
  - 1st JSON document, add ”INSERT INTO xxx” with your JSON document.
  - Then execute the query in CQLSH (TablePlus or other)
  - If it works, go to the next item
  - If not, change your schema
  - If the schema is “impossible”, you have to change a little bit your JSON document
    - change quotes (keys for different types)
    - modify text content (some are not proper to a type)
    - brackets or braces (sometimes braces are not necessary)
  - This step is considered to be *data cleaning*, which is very common in Data Engineering (let say always necessary)
  - You have to **detail** in the report all the data cleaning steps you have done
- Create your own CQL script:
  - The script modifies automatically the whole dataset as you did in the previous step
  - Ctrl+F/Ctrl+R in a file reading software is not accepted
  - This script will generate a new file with ”INSERT INTO” queries
- Execute the CQL script:
  - Either directly in CQLSH (with Docker, you must copy the file in the container with ”docker cp”)
  - Create a script which connects to Cassandra and executes each line of the script
  - port 9042 - verify that your container's port is available outside
  - This solution will be appreciated
- Test your queries (simple, complex & hard)
  You can change your schema if you find that your queries are not proper

Of course, this is not the best solution nor the only one. However, it is the simplest one.

Chapter 2 MongoDB

The MongoDB[1] database server can be installed in a local or distributed environment.
You can download it here: `http://www.mongodb.org/downloads/` (Find some tips on Moodle or online)

2.1 Installation

2.1.1 MongoDB with Docker

It's possible also with Docker (see Section 1.1.1):

```
docker pull mongodb
```

with the port 27017.

2.1.2 Windows

Download and install MongoDB[2].

```
REM Open a first shell cmd
REM Get the architecture of the OS (32 or 64 bits)
wmic os get osarchitecture

REM Extract the archive on ``C:\Program...'' (for example)
REM Create the data folder
md \data
md \data\db

REM Launch server: (ne pas fermer la fenêtre)
C:\mongodb\bin\mongod.exe
REM Or
C:\mongodb\bin\mongod.exe --dbpath "your own data folder"
```

2.1.3 Linux and MacOS

Download and install MongoDB[2].

```
#Open a first shell cmd:
tar zxvf mongodb-xxx.tgz          #xxx:version
mv mongodb-xxx /your-own-folder
ln -s mongodb-xxx mongodb         #create a symbolic link
mkdir ~/data                      #create the data folder
mkdir ~/data/db

#launch server:
bin/mongod --dbpath ~/data/db
```

[1] `http://www.mongodb.org/`
[2] Documentation : `http://docs.mongodb.org/manual/contents/`

2.2 Import datafile

You can import data with the UI *Studio3t* or *mongocompass*.

Otherwise, you can do it in your command line environment:

```
mongoimport --host localhost:27017 --db DBLP --collection publis  dblp.json
```

where:

- localhost:27017 is the mongodb server address
- DBLP: is the database name
- publis: is the collection name
- dblp.son: is the file name to import from

2.3 User Interface

You can use different UI for MongoDB interactions:

- `mongo` shell. The easiest one, but it's a shell...

  ```
  #Launch a second shell cmd:
  bin/mongo
  ```

  ⚠Do not forget that you have the "command line" and the "mongo shell".

- `Studio3t` `https://studio3t.com/`. Very powerful, can integrate datasets import, schema extraction. But a license has to be asked, anyway *a student license can be used*.
- `Robo3t` `https://robomongo.org/download`. Quite simple to use, free. But not dataset import (see previous section).
- `MongoDB Compass` `https://www.mongodb.com/products/compass`. Good solution, many improvement from MongoDB for this UI. But less options like in Studio3t. Free.

Chapter 3 Elasticsearch

The `elasticsearch` software is a column oriented NoSQL solution, aiming at doing similarity queries with an intuitive query language dedicated to text content.

To store and to query it, we can use the REST API with the port 9200.

3.1 Install

3.1.1 elasticsearch/kibana with Docker

As for cassandra (Section 1.1.1), we can download an image with Elasticsearch and Kibana. For this, on your command line:

```
docker pull nshou/elasticsearch-kibana:latest
```

When you instanciate the container, modify the external port "9200" and "5601" to work properly.

3.1.2 Standard installation

This application developped in Java can be found here: `http://elastic.co`

Once unziped, you can find the following folder hierarchy:

- `bin`: executables,
- `config`: configuration files,
- `plugins`: extensions,
- `logs`: journal files used in case of problems.

Configuration

You can edit the `config/elasticsearch.yml` file:

- `cluster.name`: a common name for all the nodes,
- `node.name`: a single node name for this server,
- `index.number_of_shards`: number of servers (default 1),
- `index.number_of_replicas`: number of replicates to enable fault tolerance (default 0).

Launch

In the elasticsearch directory, launch: `./bin/elasticsearch`
To test the connection: `curl -XGET 'localhost:9200'`
To shutdown the server: `curl -XPOST 'localhost:9200/_shutdown'`

Kibana

If you want to use a User Interface with elasticsearch (which is not required for this practice work), you can install Kibana from elastic.co. However, it can be quite long to download and param. Especially to set the kibana plugin into elasticsearch. Follow the tutorial online if you really want it.

3.2 Queries

3.2.1 REST structure

The REST query must contain:

- An index name (i.e., the database)
- A type name (i.e., the collection/table)
- Optionnaly an id (i.e., the docId)
- An operator: _count, _search

```
curl -XGET 'localhost:9200/tests/test/1'
```

Corresponds to the index 'tests', the type 'test' and the first docId.

```
curl -XGET 'localhost:9200/tests/test/_count'
```

Counts the number of documents in the test's type of the tests' index.

3.2.2 Matching

The service `_search` allow to query an index type is some simple and complex parameters.

- The simple matching can be done with a sequence of words like:

  ```
  curl 'localhost:9200/tests/test/_search?q=alien'
  ```

- The simple matching can be done with a sequence of words on a specific field like:

  ```
  curl 'localhost:9200/tests/test/_search?q=title:alien'
  ```

- With combination of fields with negations

  ```
  curl 'localhost:9200/tests/test/_search?q=title:alien%20-year:1992'
  ```

- The matching query (with score for results) must be a valid JSon document beginning by:

  ```
  curl -XGET http://localhost:9200/tests/test/_search -H 'Content-Type: application/json' -d '
  {
    "query" : ...
  }'
  ```

- Where "query" can be a simple match for a field:

  ```
  {
    "query" : {
      "match" : { "<<<<attr>>>>" : <<<<value>>>>}
    }
  }
  ```

- Or range queries:

```
{
  "query" : {
    "range" : { "<<<<attr>>>>" : <<<<comparison>>>>}
  }
}
```

- You can have '*should*' and '*must*' operations in order to include possibilities of presency for text. It is combined with boolean ('*bool*' operator) queries:

```
{"query":{
  "bool": {
    "should": { "match": { "<<<<attr>>>>" : <<<<value>>>> }},
    "must":{ "range": { "<<<<attr>>>>" : <<<<comparison>>>>}},
    "must_not":[{"match":{"<<<<attr>>>>" : <<<<value>>>>}},
                {"match":{"<<<<attr>>>>" : <<<<value>>>>}}, ...]
}}}
```

- With filtering queries ('*filter*' operator), no score computed for this filtering part

```
{
  "query": {
    "filtered": {
      "query": {
        "match": {"<<<<attr>>>>" : <<<<value>>>>}
      },
      "filter": {
        "range": {"<<<<attr>>>>" : <<<<comparison>>>>}
}}}}
```

The complete DSL[1] language is available here: `https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl.html`

3.2.3 INSERT

The POST query on the 'index' and 'type' is necessary to add a new JSon document which is automatically indexed. All the document must have the same structure to be correctly indexed and queried.

```
curl -XPOST 'http://localhost:9200/tests/test2/' -H 'Content-type: application/json'
       -d "{'test':1, 'title' : 'my new elastic document'}"
```

[1]Domain Specific Language