le cnam

# Intéraction avec une base Cassandra

Travaux Pratiques

**CNAM Paris**
nicolas.travers (at) cnam.fr

# Chapitre

# Table des matières

# Chapitre 1

# Guide d'utilisation de Cassandra

Cassandra est un serveur de Bases de Données NoSQL basé sur une table de hachage distribuée (DHT) inspiré de DynamoDB (Amazon). Il a été créé par Facebook et maintenant est maintenu par Apache.

## 1.1 Installation de Cassandra

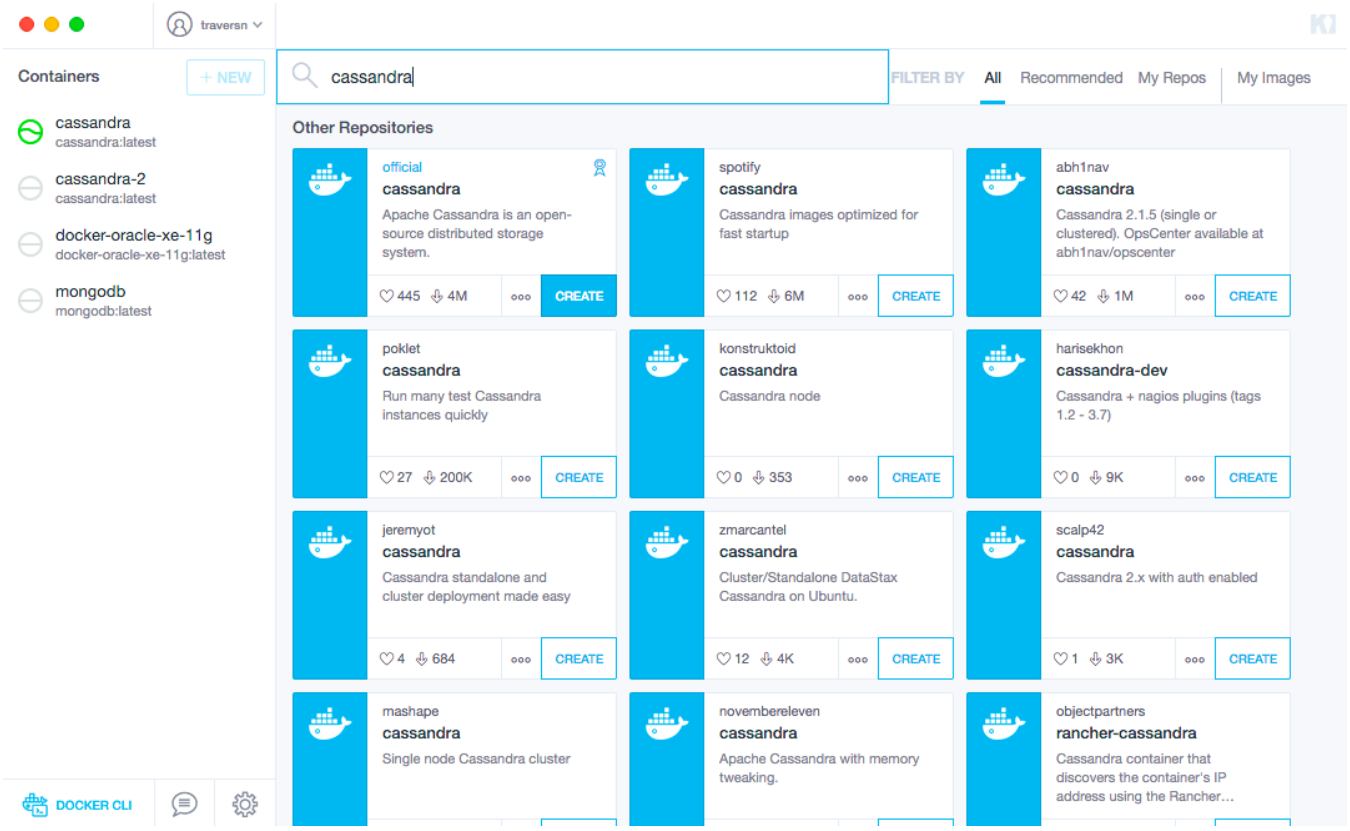
Dans le cadre des Travaux Pratiques sur Cassandra, nous utiliserons "Docker" pour nous permettre de simplifier son utilisation sur différents systèmes. Tout d'abord, il faut se référer au guide d'installation de Docker et de l'outil "Kitematic".

D'autres guides sont disponibles pour l'installation de Cassandra :

- Téléchargement de Cassandra : `https://cassandra.apache.org/download/`
- Guide complet windows : `http://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-wind`
- Guide complet Mac :
  - Avec DataStax : `http://www.datastax.com/2012/01/getting-started-with-apache-cassandra-on-windows`
  - Avec l'installeur Python "*pip install*" : `https://dbglory.wordpress.com/2015/02/22/installing-cassandra-o`

### 1.1.1 Docker : Image Cassandra

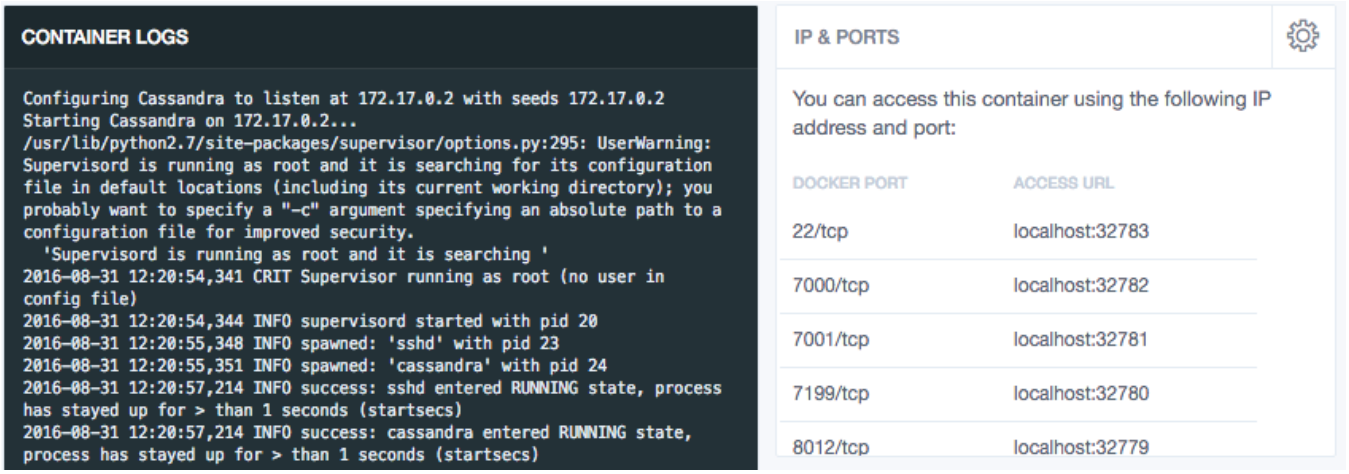
Une Docker mis en route et l'interface Kitematic ouverte, nous pouvons télécharger l'image "**cassandra :latest**" comme sur l'image ci-dessous :




Une fois téléchargée et lancée, la fenêtre d'administration affichera les informations comme dans l'image ci-dessous. Vous pourrez constater les redirections de ports ('local à l'image' → 'local à votre machine')

- 22 → 32783 : Accès SSH

— 7000 → 32782 : communication inter-noeuds
— 7001 → 32781 : communication TLS Inter-noeuds
— 7199 → 32780 : JMX (8080 pre Cassandra 0.8.xx)
— 8012 → 32779 : "Hadoop Job Tracker"




### 1.1.2 Installation Windows

Pour une installation sous Windows, il faut télécharger le serveur : `https://cassandra.apache.org/download/`
Un guide complet à suivre pour l'installation est disponible ici : `http://www.datastax.com/2012/01/getting-started-with-apac`

Il faut savoir que vous installez un serveur, il faudra donc le lancer avant toute utilisation de Cassandra. Pour cela, une console va s'ouvrir à ne surtout pas fermer.

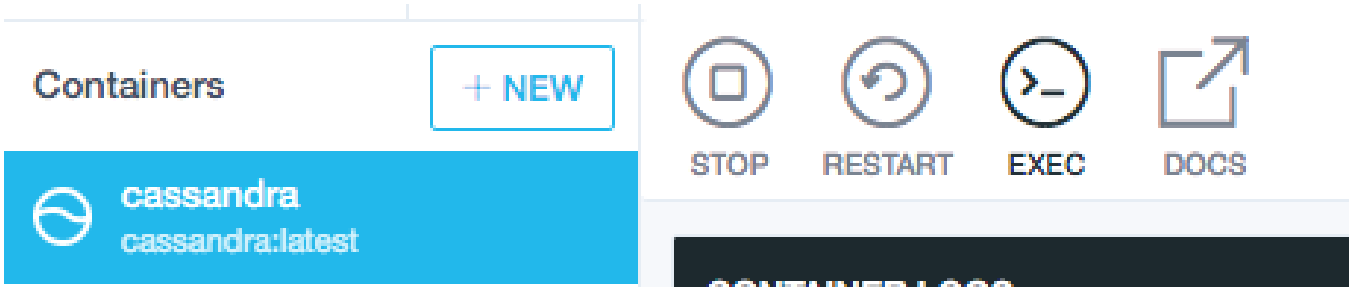
La console pour CQLSH (voir plus bas) devra être ouverte séparément.

## 1.2 Client pour Cassandra

Lorsque le serveur Cassandra tourne, vous pouvez vous y connecter avec une interface client.

### 1.2.1 Ouvrir une console sur le container Cassandra

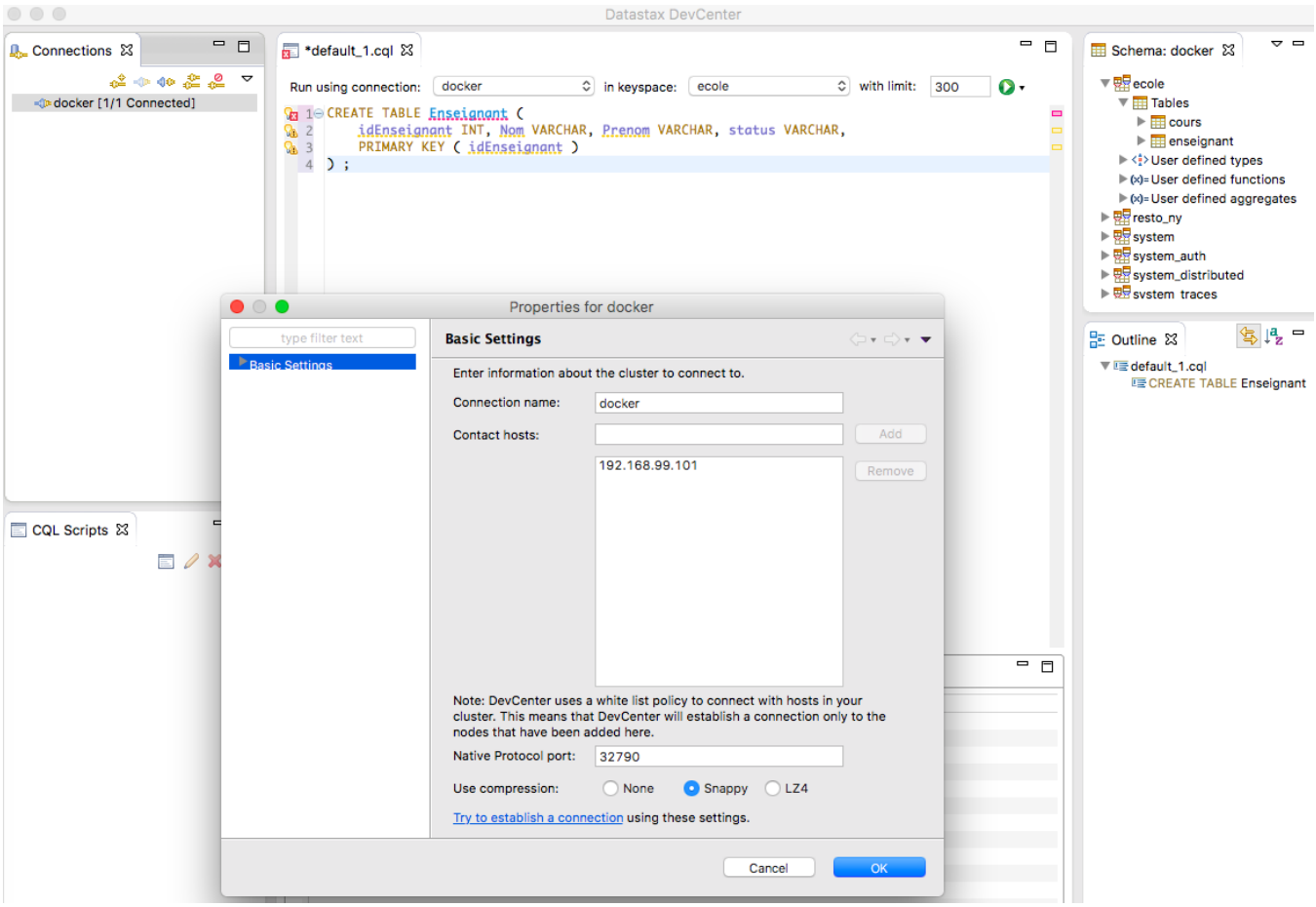
Pour ouvrir une console en vous connectant à Cassandra, il vous suffit sous KITEMATIC, de cliquer (après avoir sélectionné votre contener Cassandra) sur "**EXEC**" comme ci-dessous :




Une console sera alors ouverte et vous pourrez alors lancer la commande "CQLSH" vous permettant d'effectuer des requêtes sur la base. Le TP est prêt à tourner.

### 1.2.2 DevCenter

DEVCENTER est un client pour Cassandra développé par DataStax. Vous pouvez l'utiliser quelquesoit le serveur installé, du moment que le port "9042" (ou équivalent redirigé sous Docker) soit ouvert.




Une fois le serveur lancé, vous pouvez donc démarrer DevCenter. Il vous suffit de créer une connexion comme ci-dessus.

Cette interface permettra de créer un Keyspace, puis une fois sélectionné (onglet keyspace) vous pourrez créer des column family et exécuter des requêtes CQL sur votre schéma.

# Chapitre 2

# Interrogation de Cassandra

Dans ce chapitre, nous allons étudier la création d'une base de données (appelée KEYSPACE), puis son interrogation. Nous allons réutiliser notre base de données relationnelle vue dans les Travaux Pratiques précédents pour l'intégrer dans un environnement NoSQL. Nous allons pouvoir constater les différences avec une base de données relationnelle standard, en particulier sur les problèmes de jointures. Puis nous regarderons la création d'un cluster de 4 noeuds.

## 2.1 Création de la base de données

### 2.1.1 Keyspace

Avant d'interroger la base de données, il nous la créer. Pour commencer :

```
CREATE KEYSPACE IF NOT EXISTS ecole WITH REPLICATION = { 'class' : 'SimpleStrategy', 'replication_factor': 3 };
```

Nous créons ainsi une base de données ECOLE pour laquelle le facteur de réplication est mis à 3 pour gérer les problèmes de reprise sur panne.

Vous pouvez maintenant sélectionner la base de données pour vos prochaines requêtes (à effectuer après chaque démarrage de CQLSH) :

```
USE ecole;
```

### 2.1.2 Column Family

Nous pouvons maintenant créer les tables (ou *Column Family* sous Cassandra) COURS et ENSEIGNANT à partir de notre schéma :

```
CREATE TABLE Cours (
        idCours INT, Intitule VARCHAR, Responsable INT, Niveau VARCHAR, nbHeuresMax INT, Coeff INT,
        PRIMARY KEY ( idCours )
) ;
CREATE INDEX fk_Enseignement_Enseignant_idx ON Cours ( Responsable ) ;
```

```
CREATE TABLE Enseignant (
        idEnseignant INT, Nom VARCHAR, Prenom VARCHAR, status VARCHAR,
        PRIMARY KEY ( idEnseignant )
) ;
```

Pour vérifier si les tables ont bien été créées.

```
DESC ecole;
```

Nous pourrons voir le schéma des deux tables mais également des informations relatives au stockage dans Cassandra.

Nous pouvons maintenant y rajouter les données correspondantes :

```
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (1,'Introduction aux Bases de Donnees',1,'M1',30,3);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (2,'Immeubles de Grandes Hauteurs',4,'M1',30,2);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (3,'Production et distribution de biens et de ser',5,'M1',30,2);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (4,'Bases de Donnees Avancees',1,'M2',30,5);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (5,'Architecture des Systemes Materiel',6,'M2',8,1);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (6,'IT Business / Introduction',7,'M2',20,3);
INSERT INTO Cours (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES (7,'IT Business / Strategie et Management',8,'M2',10,1);
```

```
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (1,'Travers','Nicolas','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (2,'Mourier','Pascale','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (3,'Boisson','Francois','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (4,'Mathieu','Eric','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (5,'Chu','Chengbin','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (6,'Boutin','Philippe','Titulaire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (7,'Escribe','Julien','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (8,'Znaty','David','Vacataire');
INSERT INTO Enseignant (idEnseignant,Nom,Prenom,status) VALUES (9,'Abal-Kassim','Cheik Ahamed','Vacataire');
```

Pour vérifier le contenu de la table :

```
SELECT * FROM Cours;
```

## 2.2 Interrogation des Column Family

Pour interroger la base de données, la syntaxe CQL (pour *Cassandra Query Language*[1]) est fortement inspirée de SQL. Pour la suite des exercices, exprimer en CQL la requête demandée :

*2.2.1* Liste tous les cours ;

*2.2.2* Liste des intitulés de cours ;

*2.2.3* Nom de l'enseignant n°4 ;

*2.2.4* Intitulé des cours du responsable n° 1 ;

*2.2.5* Intitulé des cours dont le nombre d'heures maximum est égal à 30 ;

*2.2.6* Intitulé des cours dont le responsable '1' et dont le niveau est 'M1' ;
Utiliser 'ALLOW FILTERING'.

*2.2.7* Intitulé des cours dont les responsables ont une valeur inférieure à 5 ;

*2.2.8* Intitulé des cours dont l'identifiant est inférieure à 5 ;
Utiliser la fonction 'TOKEN()'.

*2.2.9* Compter le nombre de lignes retournées par la requête précédente ;
Utiliser 'COUNT(*)'

### 2.2.1 CQL Avancé

*2.2.1* La requête ci-dessous ne fonctionne pas, trouver une solution pour la faire fonctionner ;

```
select nom, prenom from enseignant where status='Vacataire';
```

*2.2.2* Créer un deuxième index sur cours.niveau. Exécutez de nouveau la requête 2.2.6. Regarder la trace générée par la requête (DevCenter : onglet 'Query Trace' à côté des résultats - Console : TRACING ON, avant la requête). Quel index a-t-il été utilisé pour cette requête ?

*2.2.3* Donner les intitulés des cours dont le statut du responsable est 'Vacataire' ;

*2.2.4* La jointure n'est pas possible avec CQL (à cause du stockage par hachage distribué). Nous allons créer une solution alternative en fusionnant les deux tables. Pour cela, il va nous allons choisir d'intégrer la table 'Enseignant' dans la table 'Cours', nous appellerons cette table '*coursEnseignant*'. Trois possibilités sont disponibles pour ce faire : *SET, LIST, MAP, SubType*. Choisissez la solution qui permettra de faire un filtrage sur le statut de l'enseignant.

*2.2.5* Insérer les données suivantes :

```
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (1,'Introduction aux Bases de Donnees',{'idenseignant':'1','nom':'Travers','prenom':'Nicolas','status':'Vacataire'},'M1',30,3);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (2,'Immeubles de Grandes Hauteurs',{'idenseignant':'4','nom':'Mathieu','prenom':'Eric','status':'Titulaire'},'M1',30,2);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (3,'Production et distribution de biens et de ser',{'idenseignant':'5','nom':'Chu','prenom':'Chengbin','status':'Titulaire'},'M1',30,2);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (4,'Bases de Donnees Avancees',{'idenseignant':'1','nom':'Travers','prenom':'Nicolas','status':'Vacataire'},'M2',30,5);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (5,'Architecture des Systemes Materiel',{'idenseignant':'6','nom':'Boutin','prenom':'Philippe','status':'Titulaire'},'M2',8,1);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (6,'IT Business / Introduction',{'idenseignant':'7','nom':'Escribe','prenom':'Julien','status':'Vacataire'},'M2',20,3);
INSERT INTO CoursEnseignant (idCours,Intitule,Responsable,Niveau,nbHeuresMax,Coeff) VALUES
 (7,'IT Business / Strategie et Management',{'idenseignant':'8','nom':'Znaty','prenom':'David','status':'Vacataire'},'M2',10,1);
```

1. Vous trouverez la syntaxe complète ici : `https://cassandra.apache.org/doc/latest/cql/dml.html#select`

*2.2.6* Créer un index sur le Niveau de la table CoursEnseignant ;

*2.2.7* Donner les noms des responsables des cours (table CoursEnseignant) de niveau 'M1' ;

*2.2.8* Donner l'intitulé des cours dont le responsable est vacataire ;

*2.2.9* Nous allons inverser la fusion des tables en créant une table 'EnseignantCours' avec un sous-type 'Cours'. Pour pouvoir rajouter un ensemble de valeurs pour les cours d'un responsable, on peut utiliser : set, map ou list. Nous utiliserons le type map.

Créer le type 'Cours' et la table 'EnseignantCours'

*2.2.10* Insérer les données suivantes :

```
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (1,'Travers','Nicolas','Vacataire',
 {1:{idcours:1,intitule:'Introduction aux Bases de Donnees',niveau:'M1',nbHeuresMax:30,Coeff:3},
 4:{idcours:4,intitule:'Bases de Donnees Avancees',niveau:'M2',nbHeuresMax:30,coeff:5}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (2,'Mourier','Pascale','Titulaire', {});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (3,'Boisson','Francois','Vacataire', {});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (4,'Mathieu','Eric','Titulaire',
 {4:{idcours:2,intitule:'Immeubles de Grandes Hauteurs',niveau:'M1',nbHeuresMax:30,coeff:2}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (5,'Chu','Chengbin','Titulaire', {});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (6,'Boutin','Philippe','Titulaire',
 {5:{idcours:5,intitule:'Architecture des Systemes Materiel',niveau:'M2',nbHeuresMax:8,coeff:1}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (7,'Escribe','Julien','Vacataire',
 {6:{idcours:6,intitule:'IT Business / Introduction',niveau:'M2',nbHeuresMax:20,coeff:3}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (8,'Znaty','David','Vacataire',
 {7:{idcours:7,intitule:'IT Business / Strategie et Management',niveau:'M2',nbHeuresMax:10,coeff:1}});
INSERT INTO EnseignantCours (idEnseignant,Nom,Prenom,status,cours) VALUES (9,'Abal-Kassim','Cheik Ahamed','Vacataire', {});
```

*2.2.11* Créer un index sur le status de la table EnseignantCours ;

*2.2.12* Donner les intitulés des cours dont le responsable est Vacataire ;

## 2.3 Map/Reduce

Nous allons tester la fonction '*Average*' donnée en exemple dans le cours. Toutefois, il faut auparavant activer le paramètre '*enable_user_defined_functions*' pour activer les fonctions utilisateurs (donc Map/Reduce).

Pour ce faire :

- *Docker* :
  - Sous Kitematic, cliquez sur 'exec'
  - Editer le fichier de configuration : `vi /etc/cassandra/default.conf/cassandra.yaml`
  - Chercher la chaine '`/user_define`'
  - Modifier (pour éditer sous `i` : 'i')le paramètre en le passant à '`true`' (bien mettre un espace entre ' :' et 'true')
  - Enregistrer : "`:wq`"
  - Redémarrer Cassandra (attention au changement de port éventuel)
- *Installation locale*
  - Editer le fichier '`cassandra.yaml`' dans le répertoire de configuration de cassandra
  - Chercher la chaine '`user_define`'
  - Modifier le paramètre en le passant à '`true`' (bien mettre un espace entre ' :' et 'true')
  - Sauvegarder le fichier de configuration
  - Redémarrer Cassandra

*2.3.1* Créer la fonction *Map*

```
CREATE OR REPLACE FUNCTION avgState ( state tuple<int,bigint>, val int )
        CALLED ON NULL INPUT RETURNS tuple<int,bigint> LANGUAGE java
        AS 'if (val !=null) { state.setInt(0, state.getInt(0)+1);
                state.setLong(1, state.getLong(1)+val.intValue()); }
                return state;';
```

*2.3.2* Créer la fonction *Reduce*

```
CREATE OR REPLACE FUNCTION avgFinal ( state tuple<int,bigint> )
        CALLED ON NULL INPUT RETURNS double LANGUAGE java
        AS 'double r = 0;
                if (state.getInt(0) == 0) return null;
                r = state.getLong(1);
                r/= state.getInt(0);
                return Double.valueOf(r);';
```

*2.3.3* Créer la fonction d'agrégat utilisateur (*UDA*)

```
CREATE AGGREGATE IF NOT EXISTS average ( int )
        SFUNC avgState STYPE tuple<int,bigint>
        FINALFUNC avgFinal INITCOND (0,0);
```

*2.3.4* Calculer la moyenne des '*nbHeuresMax*' pour la table 'coursEnseignant'

*2.3.5* La même mais pour des responsables 'Vacataire'

### 2.3.1 Bonus

*2.3.1* Créer une fonction Map/Reduce pour faire l'équivalent d'un "*GROUP BY + COUNT*" sur du texte pour la requête suivante :

```
SELECT countGroup(niveau) from CoursEnseignant;
```

Le paramètre du 'state' doit être un '*map<text, int>*'.