

|  |
| --- |
| ESILV |
| Développement d'application pour le Cloud |
| Enseignant : N. Travers |

|  |
| --- |
| BEN MABROUK Salem  FOUAD Badre-Addine  GAZON Félicien  **31/01/2018** |

Table des matières

[Le Jeu de données 2](#_Toc505193453)

[Installation de l’application 3](#_Toc505193454)

[Dénormalisation & Transformation 3](#_Toc505193455)

[Le format final 3](#_Toc505193456)

[Interrogations de la base 5](#_Toc505193457)

[Vues 6](#_Toc505193458)

[Vue Administration 6](#_Toc505193459)

[Réplicas sets 6](#_Toc505193460)

[Etape 1 : Création de trois répertoires réplicas set (facteur de réplication = 3) 6](#_Toc505193461)

[Etape 2 : Ouvrir Mongod dans les trois replicas (port 27017 , 27018, 27019 et 27020) 7](#_Toc505193462)

[Etape 3 : Connection au port 27018 (premier replica set), initialiser et ajouter avec les autres replicas set 7](#_Toc505193463)

[Etape 4 : récupérer le status des réplicas 8](#_Toc505193464)

[Etape 5 : Test de la tolérance aux pannes 9](#_Toc505193465)

[Etape 6 : tout est ok, réactivons le serveur du port 27018 9](#_Toc505193466)

[Etape 7 : ajout d’un arbitre 9](#_Toc505193467)

[Le sharding 10](#_Toc505193468)

[Etape 1 : création des répertoires de configuration et shards 10](#_Toc505193469)

[Etape 2 : Comment distribuer les données dans les différents serveurs. 10](#_Toc505193470)

[Aperçu de l’application : 12](#_Toc505193471)

# Structure du projet

C:.

| CloudAppMongo.py

| Rapport\_Cloud.docx

| README.md

|

+---build

|

+---Data

|

+---Queries

| Queries\_used\_in\_project.js

| Query\_to\_execute.js

|

\---transform

| 1\_SQLtoCSV.R

| 2\_CSVtoMongo.py

| 3\_ActionsExtended.js

| 3\_MatchesExtended.js

|

+---CSV

| Actions.csv

| Matches.csv

| Players.csv

| Teams.csv

|

\---JSON

premierleague.actionsExtended.json

premierleague.matchesExtended.json

**build** : contains exe file.

**Data** : contains servers data.

**Queries** : contains queries.

**Transform** : needed for installation

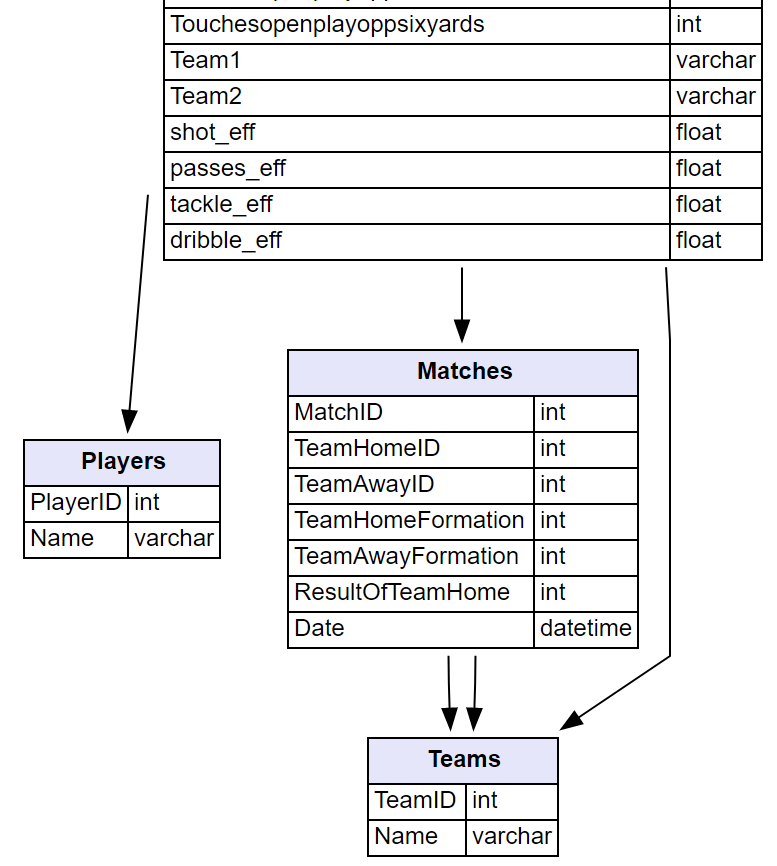
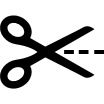
**CSV** : Original dataset

**JSON** : Final dataset

# Le Jeu de données

Etant tous les 3 fans de foot, nous avons choisi un jeu de données en rapport avec le Football. C’est un dataset contenant un lot de détails la saison 2011/2012 de la Premier League anglaise. Nous pouvons y trouver la liste des joueurs, les équipes, les matchs et les actions de chaque joueur lorsqu’ils ont joué.

C’est un jeu de données de 11.3MB utilisé pour faire de la classification. Elle contient 4 tables, 10716 lignes pour 217 colonnes.



# Installation de l’application

Les étapes d’installation sont détaillées dans le Readme.md.

# Dénormalisation & Transformation

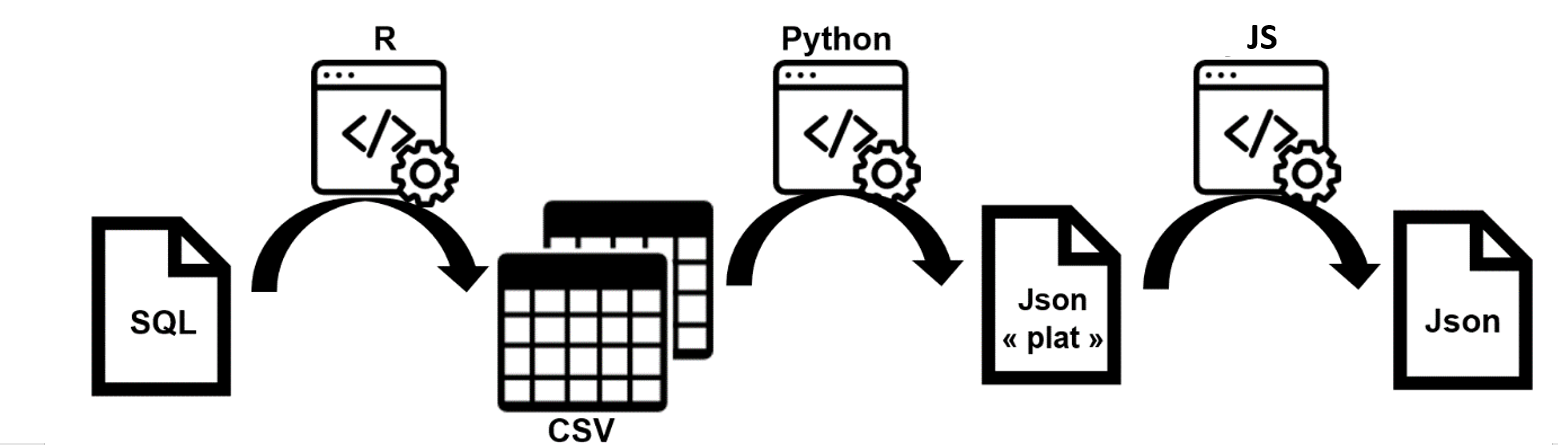
Afin de travailler cette donnée sous MongoDB, nous devons l’obtenir sous format Json. Plusieurs étapes sont requises avant d’arriver à cette fin.

En téléchargeant le dataset, nous obtenons un fichier sql. Nous installons alors les données sur MySQL.

Ensuite, un script codé en R permet d’obtenir 1 fichier CSV / table. Ainsi, nous avons maintenant 4 fichiers CSV.

Nous avons ensuite développé un code en Python qui transforme chaque fichier CSV en une suite de documents Json. Ces documents étant dits « plats », ils ne sont pas adaptés à MongoBD. Nous avons alors agrégé ces documents grâce à un autre code Python.

Schéma général :

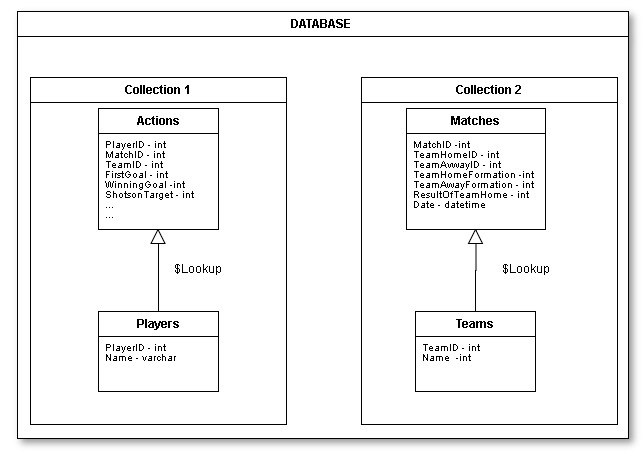


Vous pouvez trouver :

* Le code R dans le dossier transform/1\_SQLtoCSV.R
* Le code python dans transform/2\_CSVtoMongo.py
* Les codes JavaScript dans transform/3\_ActionsExtended.js et transform/3\_MatchesExtended.js

# Le format final

Des 4 tables initiales, nous avons gardé 2 collections. Nous avons intégré Player dans Actions et Team dans Match.



Voici la différence entre le format original et le final :

|  |
| --- |
| **Match** |
| **Avant** |
|  |
| **Après** |
| **{**  "\_id" **:** ObjectId**(**"5a2058753dc65d0c247629c3"**),**  "MatchID" **:** "50"**,**  "Date" **:** "2012-04-07 00:00:00"**,**  "TeamHome" **:** **{**  "TeamID" **:** "14"**,**  "Name" **:** "Liverpool"**,**  "TeamHomeFormation" **:** "6"**,**  "ResultOfTeamHome" **:** NumberInt**(**0**)**  **},**  "TeamAway" **:** **{**  "TeamID" **:** "7"**,**  "Name" **:** "Aston Villa"**,**  "TeamAwayFormation" **:** "2"  **}**  **}** |

|  |
| --- |
| **Actions** |
| **Avant** |
|  |
| **Après** |
| **{**  "\_id" **:** ObjectId**(**"5a20586a3dc65d0c2475fefa"**),**  "MatchID" **:** "150"**,**  "FirstGoal" **:** "0"**,**  "WinningGoal" **:** "0"**,**  "ShotsonTargetincgoals" **:** "0"**,**  "SavesMade" **:** "0"**,**  "Starts" **:** "1"**,**  "SubstituteOn" **:** "0"**,**  "SubstituteOff" **:** "0"**,**  …………………………………………… Json coupé ………………………………………………………………  "tackle\_eff" **:** "0.5"**,**  "dribble\_eff" **:** "1"**,**  "Player" **:** **{**  "PlayerID" **:** "3"**,**  "Name" **:** "Giggs Ryan"  **},**  "TeamAgainst" **:** **{**  "TeamID" **:** "8"**,**  "Name" **:** "Chelsea"**,**  "isHome" **:** true  **},**  "TeamWith" **:** **{**  "TeamID" **:** "1"**,**  "Name" **:** "Manchester United"**,**  "isHome" **:** false  **},**  "SummaryMatch" **:** **{**  "MatchResult" **:** "won"**,**  "goals" **:** NumberInt**(**0**),**  "TimePlayed" **:** "90"**,**  "Assists" **:** "1"**,**  "YellowCards" **:** "0"**,**  "RedCards" **:** "0"**,**  "passes\_eff" **:** "0.78125"**,**  "PositionID" **:** "4"**,**  "TotalSuccessfulPassesAll" **:** "50"**,**  "TotalUnsuccessfulPassesAll" **:** "14"  **}**  **}** |

# Interrogations de la base

4 types d’interrogations fréquentes :

* Le nombre de but par joueur
* Le nombre de cleansheet par gardien
* Le nombre de but concédé par joueur
* Le nombre de pénalty concédé par joueur

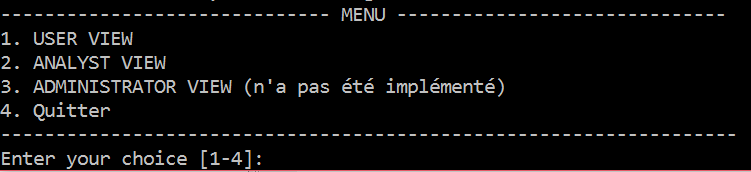
3 types d’interrogation lourdes :

* V/N/D par équipe
* Top n des équipes qu’ont le plus gagné de matches
* Comparaison entre l'équipe X et les autres équipes du top n

Utiliser des requêtes avec explain ! Bien entendu, il faudra extrapoler l’évolution de la taille de la base de données pour un passage à l’échelle.

# Vues

Nous avons fait le choix de diviser nos différentes vues dans une application console.

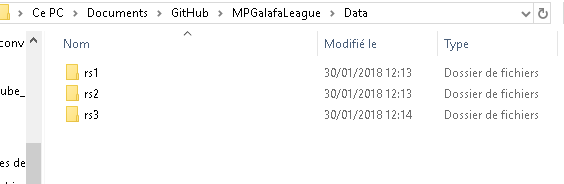


Pour chaque vue, les requêtes vues dans la partie [Interrogations de la base](#_Interrogations_de_la_1) y sont affichées et attendent d’être exécutées.

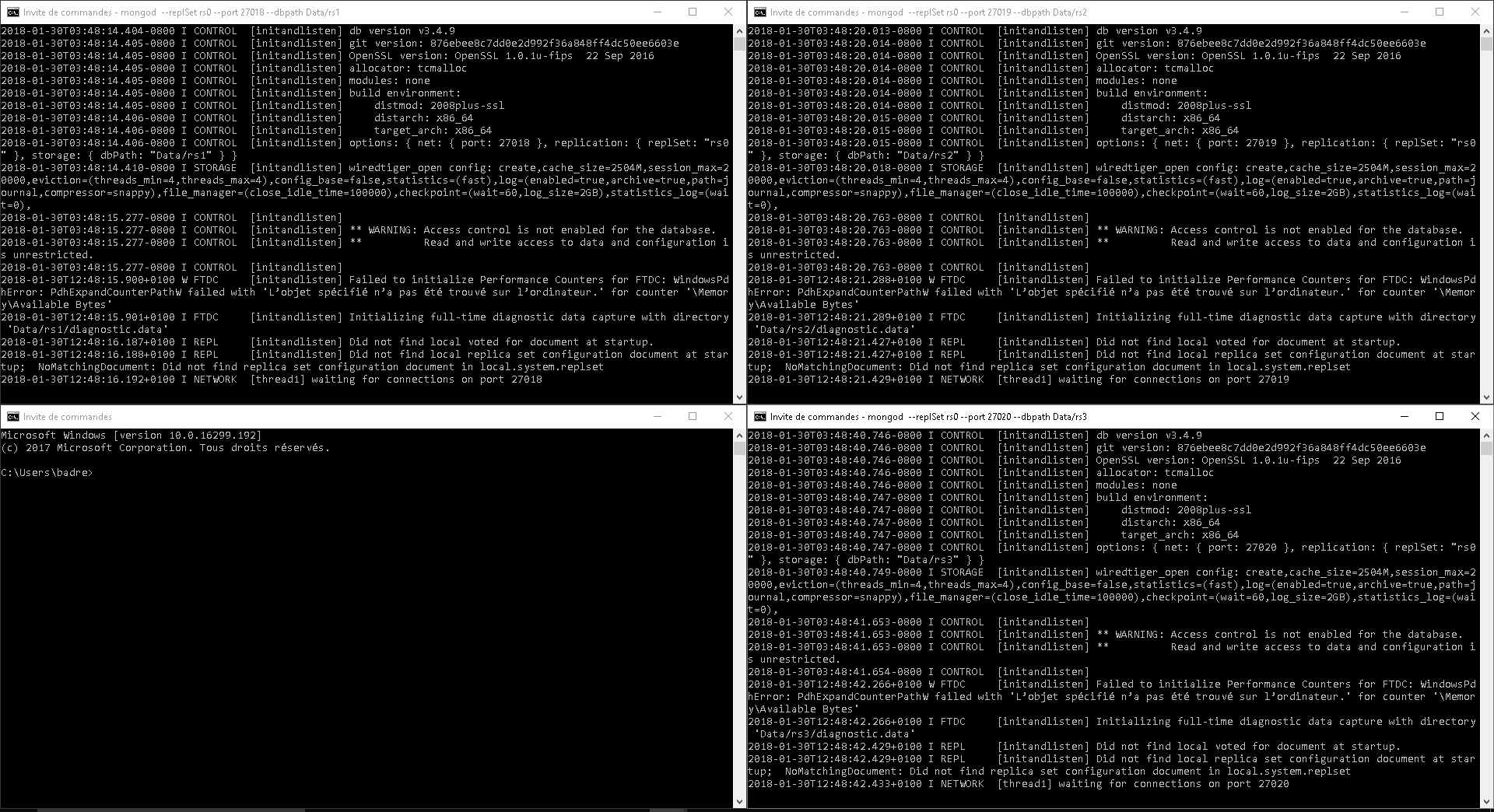
## Vue Administration

### Réplicas sets

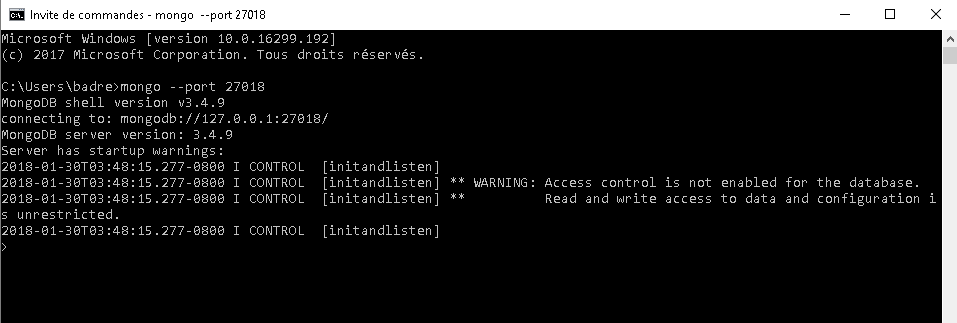
#### Etape 1 : Création de trois répertoires réplicas set (facteur de réplication = 3)

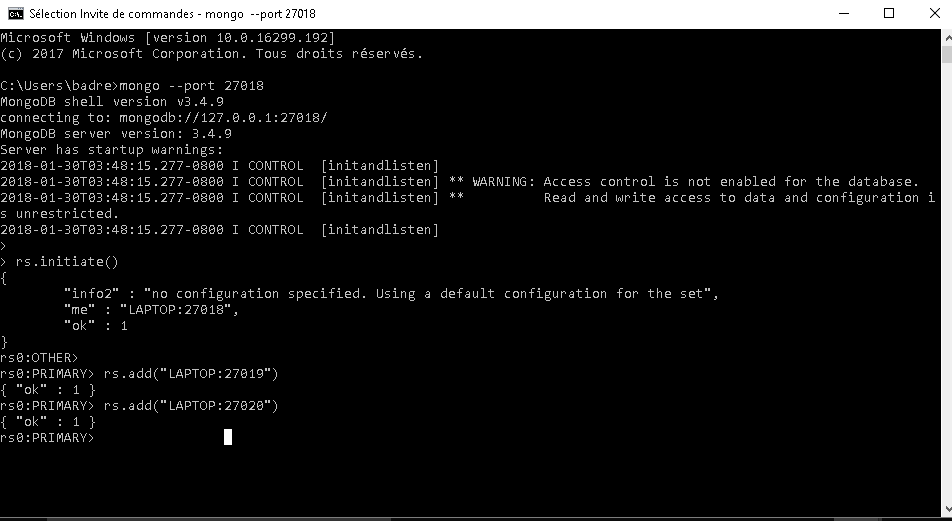


#### Etape 2 : Ouvrir Mongod dans les trois replicas (port 27017 , 27018, 27019 et 27020)



#### Etape 3 : Connection au port 27018 (premier replica set), initialiser et ajouter avec les autres replicas set





#### Etape 4 : récupérer le status des réplicas



#### Etape 5 : Test de la tolérance aux pannes



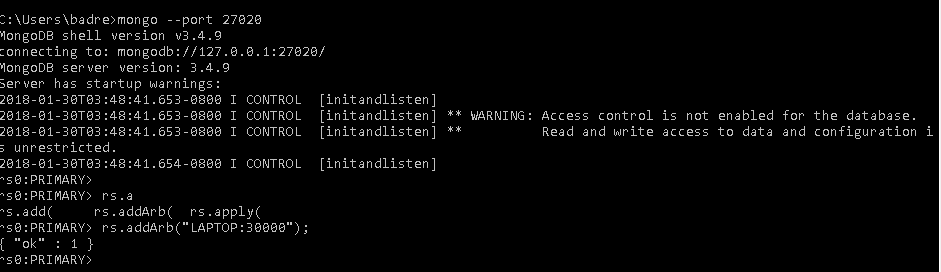
Désactivation du port 27018, observation des status.

Nous remarquons que le serveur connecté au port 27020 est devenu primary

#### Etape 6 : tout est ok, réactivons le serveur du port 27018

Le serveur du port 27018 reste secondary car le primary a été attribué au serveur du port 27020

#### Etape 7 : ajout d’un arbitre





Nos données sont maintenant disponibles en permanence. Notre système est tolérant aux pannes

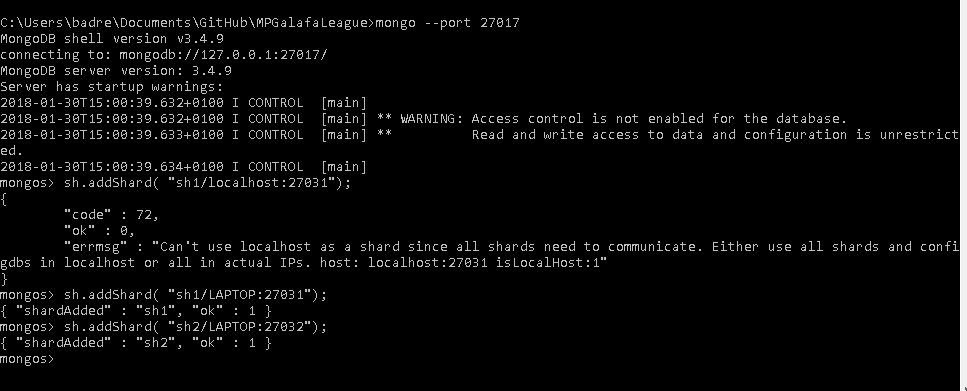
### Le sharding

#### Etape 1 : création des répertoires de configuration et shards

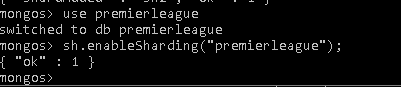


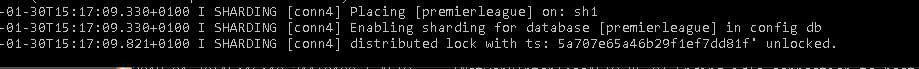
#### Etape 2 : Comment distribuer les données dans les différents serveurs.

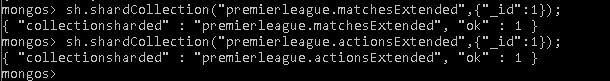
Création de shard et mise en place d’un serveur de routage



Activation du sharding :







Récuperation du status des shards :



## Aperçu de l’application :

Vous pouvez lancer l’application sans installation python préalable.

