

TD2

Database with MongoDB



<https://www.jordansablon.fr/enseignement>



https://github.com/jsablonEnseignement/td2_sources

- Déploiement d'une base de données MongoDB via Atlas
- Initialisation de la collection
- Insertion des données via MongoDB Compass
- Récupération du code source du projet & installation des dépendances
- Connecter l'application à la base de données
- Création du modèle
- Mise à jour des services et contrôleurs
- Utilisation des variables d'environnement

Déploiement d'une base de données MongoDB via Atlas

Créez un compte sur [Atlas](#)

MongoDB Atlas

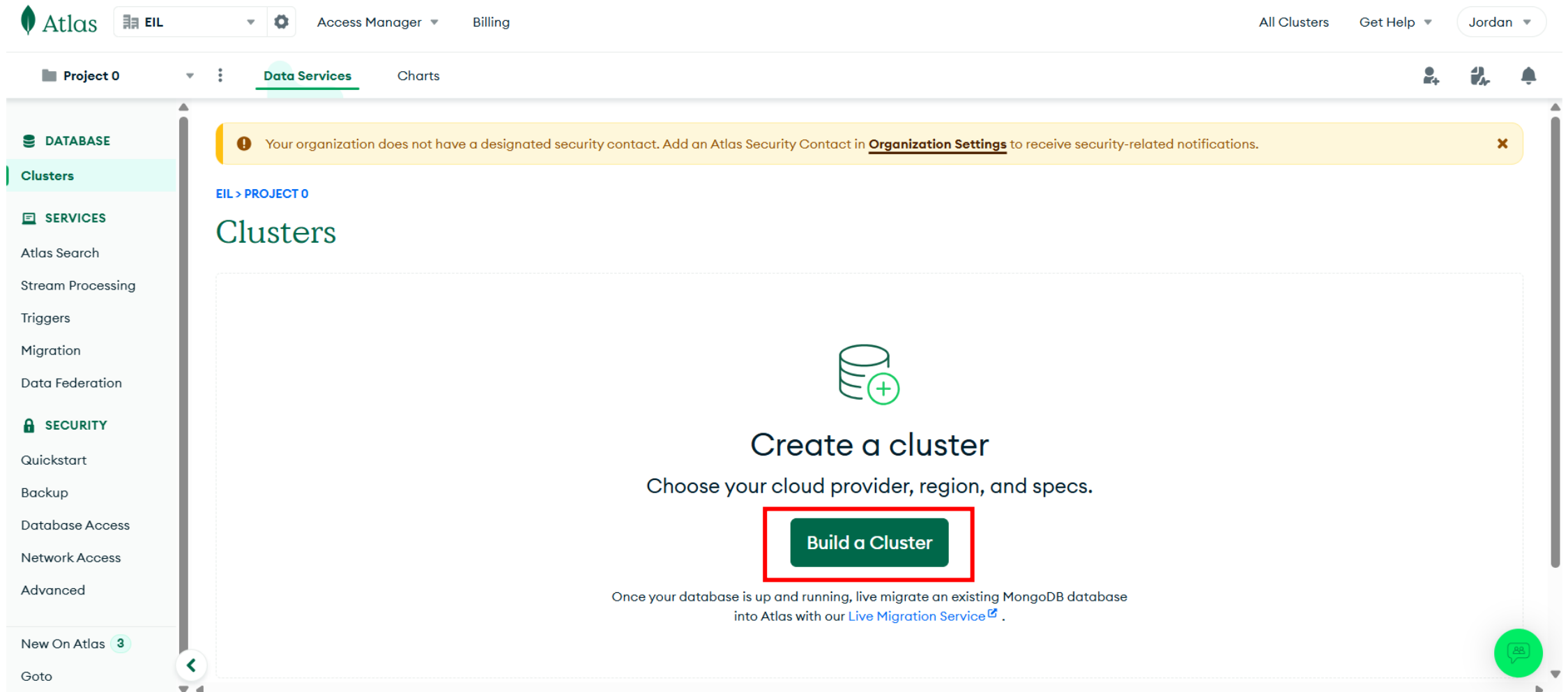
- ✓ **Work with your data as code**
Documents in MongoDB map directly to objects in your programming language. Modify your schema as your apps grow over time.
- ✓ **Focus on building, not managing**
Let MongoDB Atlas take care of the infrastructure operations you need for performance at scale, from always-on security to point-in-time recovery.
- ✓ **Simplify your data dependencies**
Leverage application data for full-text search, real-time analytics, rich visualizations and more with a single API and minimal data movement.

Sign up

See what Atlas is capable of for free

Déploiement d'une base de données MongoDB via Atlas

Sélectionnez « Build a Database » pour créer votre cluster de base de données



Sélectionnez le cluster « M0 » qui est gratuit

Deploy your cluster

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

☐ **M10** **\$0.59/hour**

Dedicated cluster for development environments and low-traffic applications.

STORAGE	RAM	vCPU
40 GB	2 GB	2 vCPUs

☐ **Flex** **From \$0.011/hour**
Up to \$30/month

For application development and testing, with on-demand burst capacity for unpredictable traffic.

STORAGE	RAM	vCPU
5 GB	Shared	Shared

☒ **Free**

For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

✓ **Free forever!** Your free cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Configurations

Name

Quick setup

☒ Automate security setup ⓘ

I'll do this later

Go to Advanced Configuration

Create Deployment



Déploiement d'une base de données MongoDB via Atlas

Descendre sur la page et créer un utilisateur pour vous connecter à votre base de données

Atlas EIL Access Manager Billing All Clusters Get Help Jordan

Project 0 Data Services Charts

✓ How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

1 **Username and Password** Certificate

Create a database user using a username and password. Users will be given the *read and write to any database privilege* by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password. You can manage existing users via the [Database Access Page](#).

2 **Username**
jsabloneilco

Password 🔑
..... 🔍 Autogenerate Secure Password 📋 Copy

3 **Create User**

Déploiement d'une base de données MongoDB via Atlas

Descendre sur la page et sélectionnez ensuite « My Local Environment »

Atlas EIL Access Manager Billing All Clusters Get Help Jordan

Project 0 Data Services Charts

Where would you like to connect from?

Enable access for any network(s) that need to read and write data to your cluster.

My Local Environment

Use this to add network IP addresses to the IP Access List. This can be modified at any time.

Cloud Environment

Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

ADVANCED

Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the [Network Access Page](#).

IP Address	Description
Enter IP Address	Enter description
Add My Current IP Address	
Add Entry	

IP Access List	Description
----------------	-------------

Déploiement d'une base de données MongoDB via Atlas

Descendre sur la page et entrez l'adresse IP 0.0.0.0/0 et cliquez sur « Add Entry »

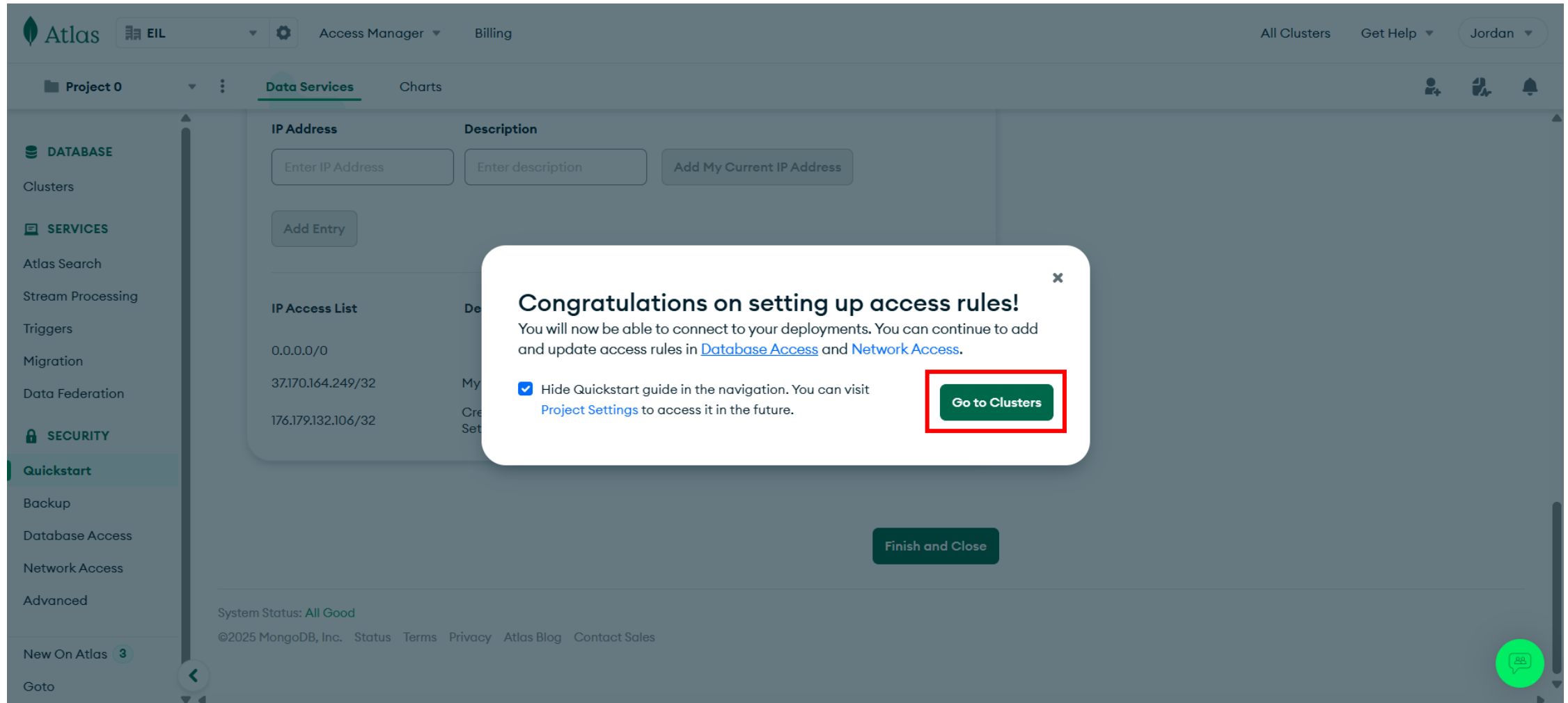
The screenshot shows the MongoDB Atlas interface for managing IP Access Lists. The left sidebar contains navigation links for DATABASE, SERVICES, and SECURITY. The main content area is titled 'Add entries to your IP Access List' and includes a form with fields for 'IP Address' and 'Description', and an 'Add My Current IP Address' button. Below the form is a table of existing IP Access List entries. A red box labeled '1' highlights the 'IP Address' input field, and another red box labeled '2' highlights the 'Add Entry' button.

IP Access List

IP Address	Description	
0.0.0.0/0		EDIT REMOVE
37.170.164.249/32	My IP Address	EDIT REMOVE
176.179.132.106/32	Created as part of the Auto Setup process	EDIT REMOVE

Déploiement d'une base de données MongoDB via Atlas

Cliquez sur « Finish and Close » : votre cluster de base de données est désormais créé



Initialisation de la collection

Depuis le menu « Clusters », cliquez sur « Browse Collections » pour afficher vos collections

The screenshot shows the MongoDB Atlas interface. At the top, the 'Atlas' logo is on the left, and 'EIL' is selected in a dropdown menu. To the right are links for 'Access Manager', 'Billing', 'All Clusters', 'Get Help', and a user profile 'Jordan'. Below this, a navigation bar shows 'Project 0' and 'Data Services' (which is highlighted). The left sidebar contains a 'DATABASE' section with 'Clusters' selected, and a 'SERVICES' section with options like 'Atlas Search', 'Stream Processing', 'Triggers', 'Migration', and 'Data Federation'. Below these are 'SECURITY' options like 'Backup', 'Database Access', 'Network Access', and 'Advanced'. At the bottom of the sidebar are 'New On Atlas' (with a '3' badge) and a 'Goto' button. The main content area has a blue banner at the top stating 'We are deploying your changes (current action: creating a plan)'. Below this is a yellow warning banner: 'Your organization does not have a designated security contact. Add an Atlas Security Contact in [Organization Settings](#) to receive security-related notifications.' The main heading is 'Clusters' with a sub-header 'EIL > PROJECT 0'. A search bar 'Find a database deployment...' is on the left, and 'Edit Config' and 'Create' buttons are on the right. A blue banner says 'Loading your sample dataset...'. Below this, for 'Cluster0', there are buttons for 'Connect', 'View Monitoring', 'Browse Collections' (highlighted with a red box), and a three-dot menu. To the right of these buttons is a 'FREE' badge. At the bottom, there's a 'Visualize Your Data' section with a description and an 'Explore Charts' button. To the right of this are four metrics: 'Connections' (0), 'In/Out' (100.0 B/s), and 'Data Size' (0.00 B / 512.00 MB (0%)).

Initialisation de la collection

Supprimez la base de données d'exemple

The screenshot shows the MongoDB Atlas web interface. At the top, the 'Atlas' logo is on the left, and 'EIL', 'Access Manager', and 'Billing' are in the center. On the right, there are links for 'All Clusters', 'Get Help', and a user profile 'Jordan'. Below the top bar, the 'Project 0' dropdown is on the left, and 'Data Services' and 'Charts' are in the center. The left sidebar contains a 'DATABASE' section with 'Clusters' selected, and a 'SERVICES' section with 'Atlas Search', 'Stream Processing', 'Triggers', 'Migration', 'Data Federation', and 'SECURITY' with 'Backup', 'Database Access', 'Network Access', 'Advanced', and 'Goto'. The main content area shows the 'Cluster0' overview with tabs for 'Overview', 'Real Time', 'Metrics', 'Collections', 'Atlas Search', 'Query Insights', 'Performance Advisor', 'Online Archive', 'Cmd Line Tools', and 'Info'. The 'Collections' tab is active, showing 'DATABASES: 1' and 'COLLECTIONS: 6'. A '+ Create Database' button and a 'Search Namespaces' input are at the top. The 'sample_mflix' namespace is expanded, showing a list of collections: 'comments', 'embedded_movies', 'movies', 'sessions', 'theaters', and 'users'. A red box highlights the trash icon next to 'sample_mflix'. The 'sample_mflix.comments' collection is selected, showing its details: 'STORAGE SIZE: 6.46MB', 'LOGICAL DATA SIZE: 11.14MB', 'TOTAL DOCUMENTS: 41079', and 'INDEXES TOTAL SIZE: 1.94MB'. Below this are tabs for 'Find', 'Indexes', 'Schema Anti-Patterns', 'Aggregation', and 'Search Indexes'. A 'Generate queries from natural language in Compass' link is present. An 'INSERT DOCUMENT' button is on the right. A 'Filter' input with a placeholder 'Type a query: { field: 'value' }' and 'Reset', 'Apply', and 'Options' buttons is shown. The 'QUERY RESULTS: 1-20 OF MANY' section displays a single document with fields: '_id', 'name', 'email', 'movie_id', and 'text'. The 'text' field contains a Latin sentence: 'Eius veritatis vero facilis quaerat fuga temporibus. Praesentium exped...'. A green chat bubble icon is in the bottom right corner.

Initialisation de la collection

Validez la suppression en saisissant le nom de la base de données puis cliquez sur « Drop »

The screenshot shows the MongoDB Atlas interface. A modal dialog titled "Drop Database" is open, prompting the user to confirm the deletion of the database "sample_mflix". The input field contains the text "sample_mflix" and is highlighted with a red box and the number "1". The "Drop" button is highlighted with a red box and the number "2". The background interface shows the "sample_mflix" database selected, with a list of collections including "comments", "embedded_movies", "movies", "sessions", "theaters", and "users". The "comments" collection is currently selected, showing its storage size, logical data size, total documents, and indexes total size. A query filter is visible at the bottom, and the query results are displayed below it.

Initialisation de la collection

Cliquez sur « Add My Own Data »

The screenshot shows the Atlas web interface. At the top, there's a header with the Atlas logo, a dropdown menu for 'EIL', and links for 'Access Manager' and 'Billing'. On the right, there are links for 'All Clusters', 'Get Help', and a user profile 'Jordan'. Below the header, there's a navigation bar with 'Project 0', 'Data Services' (highlighted), and 'Charts'. The left sidebar contains a 'DATABASE' section with 'Clusters' (highlighted) and 'SERVICES' (Atlas Search, Stream Processing, Triggers, Migration, Data Federation), and a 'SECURITY' section (Backup, Database Access, Network Access, Advanced). The main content area shows 'Cluster0' with tabs for 'Overview', 'Real Time', 'Metrics', 'Collections' (highlighted), 'Atlas Search', 'Query Insights', 'Performance Advisor', 'Online Archive', 'Cmd Line Tools', and 'Info'. Below the tabs, it says 'DATABASES: 0 COLLECTIONS: 0'. There are two buttons: 'VISUALIZE YOUR DATA' and 'REFRESH'. The main content area features a large icon of a document with a magnifying glass and the text 'Explore Your Data'. Below this, there's a list of actions: '- Find: run queries and interact with documents', '- Indexes: build and manage indexes', '- Aggregation: test aggregation pipelines', and '- Search: build search indexes'. At the bottom, there are two buttons: 'Load a Sample Dataset' and 'Add My Own Data' (highlighted with a red box). A link 'Learn more in Docs and Tutorials' is at the bottom right.

Atlas EIL Access Manager Billing All Clusters Get Help Jordan

Project 0 Data Services Charts

DATABASE Clusters SERVICES Atlas Search Stream Processing Triggers Migration Data Federation SECURITY Backup Database Access Network Access Advanced Goto

EIL > PROJECT 0 > DATABASES Cluster0

Overview Real Time Metrics Collections Atlas Search Query Insights Performance Advisor Online Archive Cmd Line Tools Info

DATABASES: 0 COLLECTIONS: 0 VISUALIZE YOUR DATA REFRESH

Explore Your Data

- **Find:** run queries and interact with documents
- **Indexes:** build and manage indexes
- **Aggregation:** test aggregation pipelines
- **Search:** build search indexes

Load a Sample Dataset Add My Own Data

Learn more in Docs and Tutorials

Initialisation de la collection

Renseignez les champs puis cliquez sur « Create » : votre collection est créée

The screenshot shows the Atlas 'Create Database' dialog box. The background interface includes the Atlas logo, 'EIL' project name, 'Access Manager', 'Billing', 'All Clusters', 'Get Help', and 'Jordan' user. The left sidebar lists 'DATABASE' (Clusters, SERVICES, SECURITY) and 'Goto'. The main area shows 'Cluster0' with 'Overview' and 'Real Time' tabs, and 'DATABASES: 0 COLLECTIONS: 0'. The dialog box has a title 'Create Database' and a close button. It contains three fields: 'Database name' with the value 'eilco_web' (annotated with a red box and '1'), 'Collection name' with the value 'students' (annotated with a red box and '2'), and 'Additional Preferences' with a 'Select' dropdown. At the bottom right of the dialog are 'Cancel' and 'Create' buttons (the 'Create' button is annotated with a red box and '3'). Below the dialog, there is a search bar with the text '- Search: build search indexes' and two buttons: 'Load a Sample Dataset' and 'Add My Own Data'.

Insertion des données via MongoDB Compass

Depuis le menu « Clusters », cliquez sur « Connect » pour récupérer l'URL de connexion

The screenshot shows the MongoDB Atlas interface. At the top, there's a navigation bar with the Atlas logo, a dropdown menu set to 'EIL', and links for 'Access Manager' and 'Billing'. On the right, it says 'All Clusters', 'Get Help', and a user profile 'Jordan'. Below this, a breadcrumb trail shows 'Project 0' > 'Data Services' > 'Charts'. The left sidebar has a 'DATABASE' section with 'Clusters' selected, and a 'SECURITY' section with various options like 'Backup', 'Database Access', etc. The main content area is titled 'Clusters' and features a search bar, an 'Edit Config' button, and a '+ Create' button. A yellow warning banner at the top states: 'Your organization does not have a designated security contact. Add an Atlas Security Contact in [Organization Settings](#) to receive security-related notifications.' Below this, a green success banner says: 'Sample dataset successfully loaded. Access it in [Collections](#) or by connecting with the MongoDB Shell.' The 'Cluster0' entry is shown with a red box around the 'Connect' button. Other buttons for 'View Monitoring' and 'Browse Collections' are also visible. At the bottom, there's a 'Visualize Your Data' section with various metrics and charts.

Atlas EIL Access Manager Billing All Clusters Get Help Jordan

Project 0 Data Services Charts

DATABASE

- Clusters
- SERVICES
 - Atlas Search
 - Stream Processing
 - Triggers
 - Migration
 - Data Federation
- SECURITY
 - Backup
 - Database Access
 - Network Access
 - Advanced
- New On Atlas 3
- Goto

Clusters

Find a database deployment...

Edit Config + Create

✓ Sample dataset successfully loaded. Access it in [Collections](#) or by connecting with the MongoDB Shell.

Cluster0 **Connect** View Monitoring Browse Collections ... FREE

Visualize Your Data

Build dashboards and charts, and embed them in your apps with MongoDB Charts.

Dismiss Explore Charts

R 0 W Last 56 seconds 100.0/s

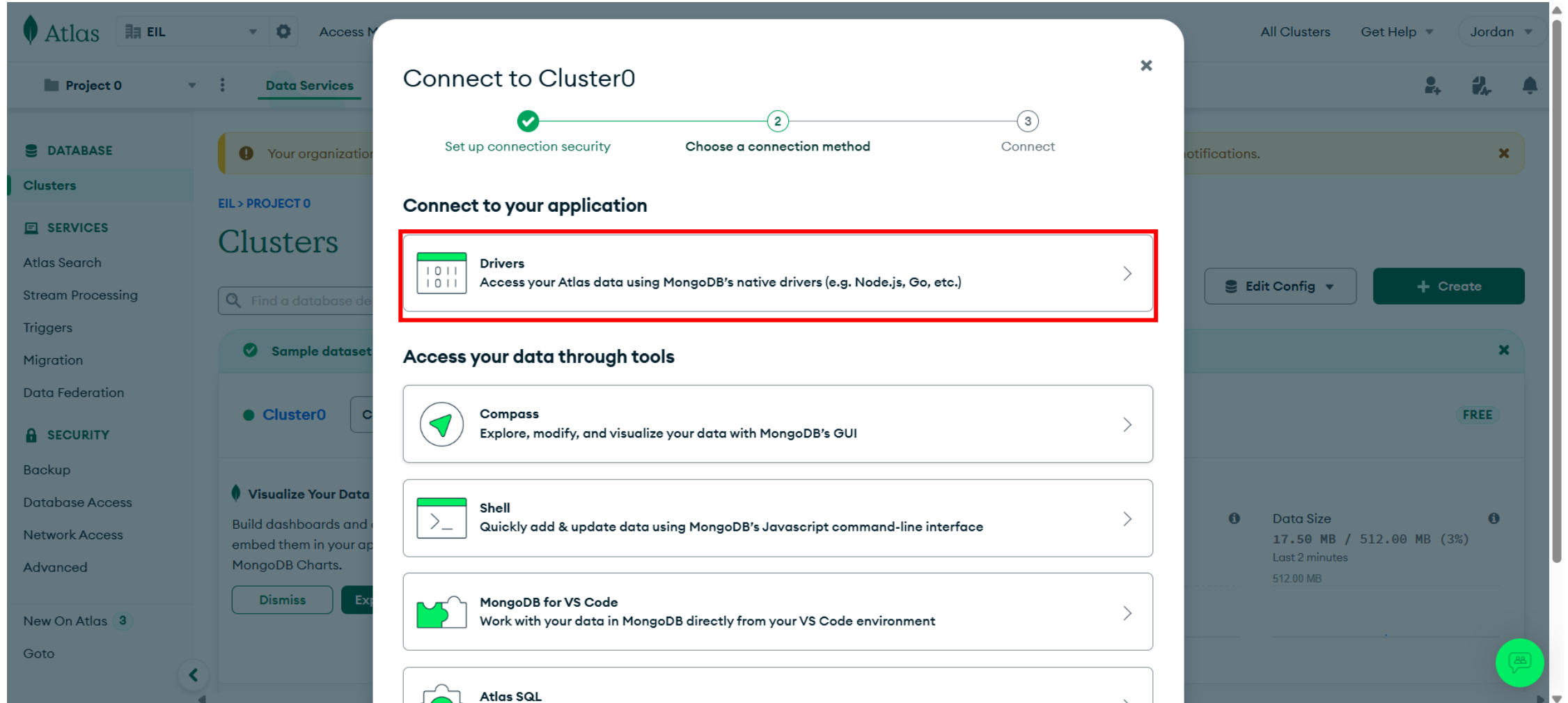
Connections 3.0 Last 56 seconds 3.0

In 41.98 KB/s Out 1.41 KB/s Last 56 seconds 41.98 KB/s

Data Size 17.50 MB / 512.00 MB (3%) Last 56 seconds 512.00 MB

Insertion des données via MongoDB Compass

Récupération de l'URL de connexion



Insertion des données via MongoDB Compass

Récupération de l'URL de connexion

Connect to Cluster0

Set up connection security ✓ Choose a connection method ✓ **3 Connect**

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

Driver	Version
Node.js	6.7 or later

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

[View MongoDB Node.js Driver installation instructions.](#)

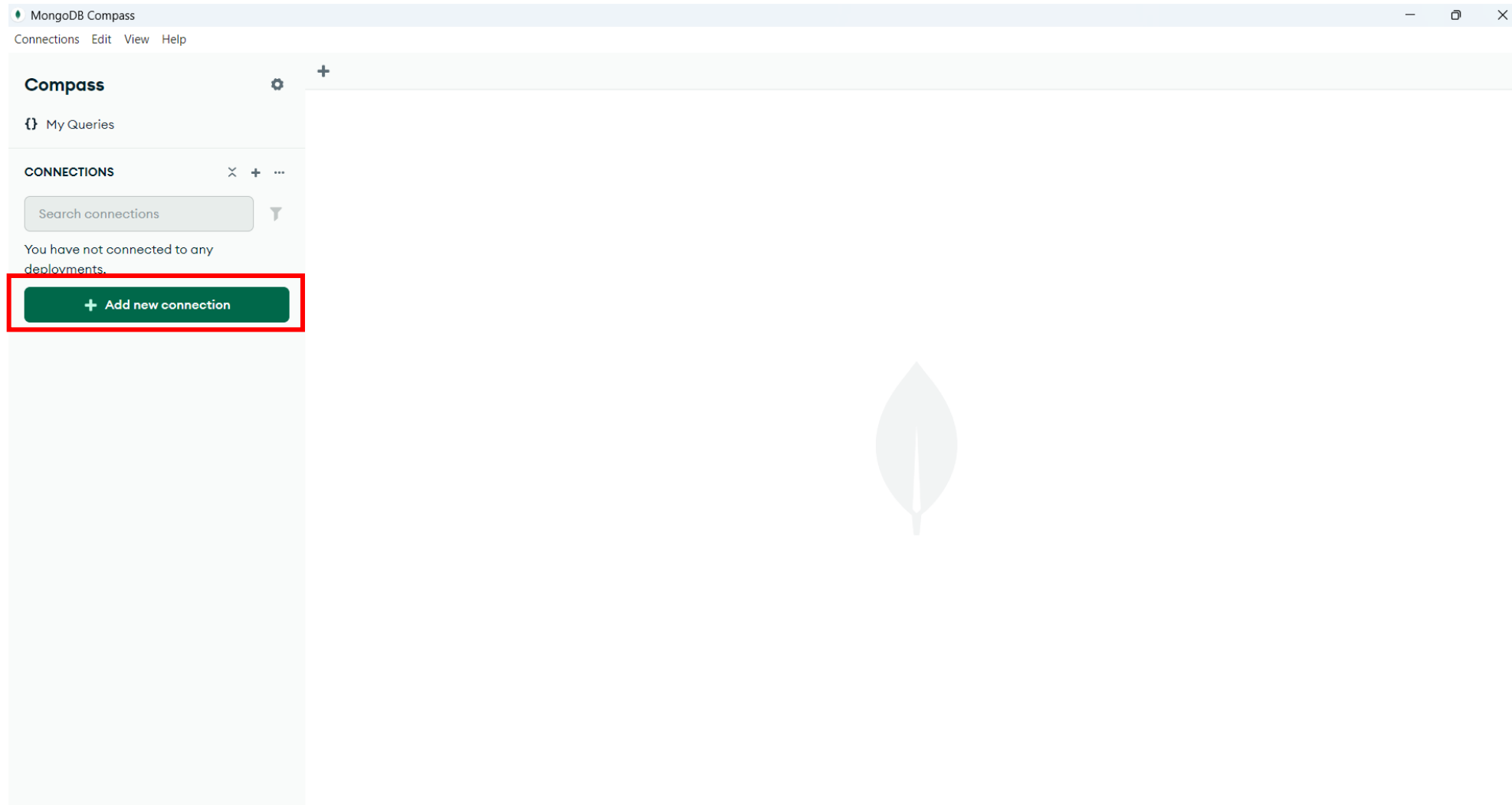
3. Add your connection string into your application code

Use this connection string in your application

☐ View full code sample

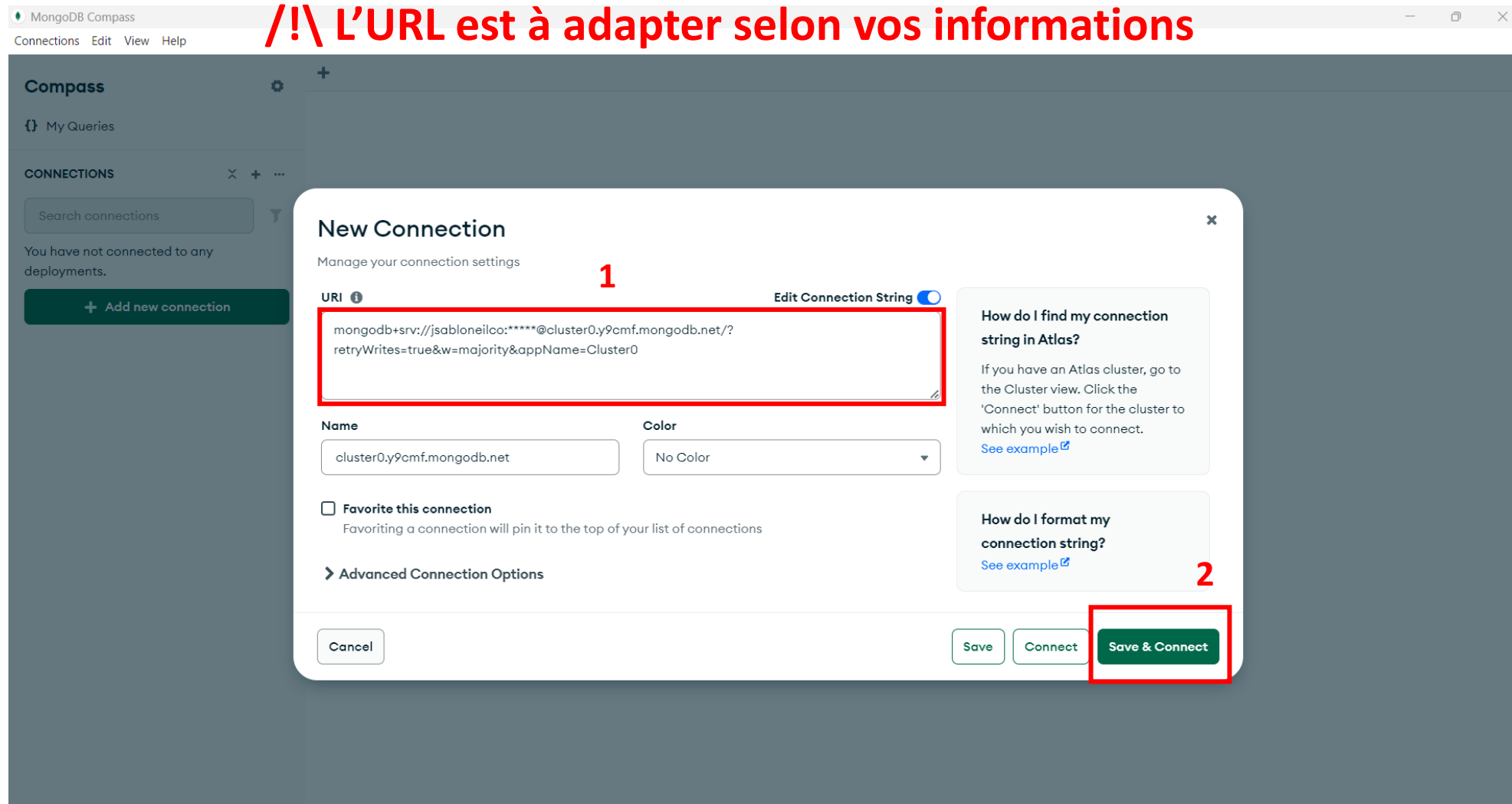
```
mongodb+srv://jsabloneilco:<db_password>@cluster0.y9cmf.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
```

Démarrer le logiciel MongoDB Compass et se connecter à la base de données



Insertion des données via MongoDB Compass

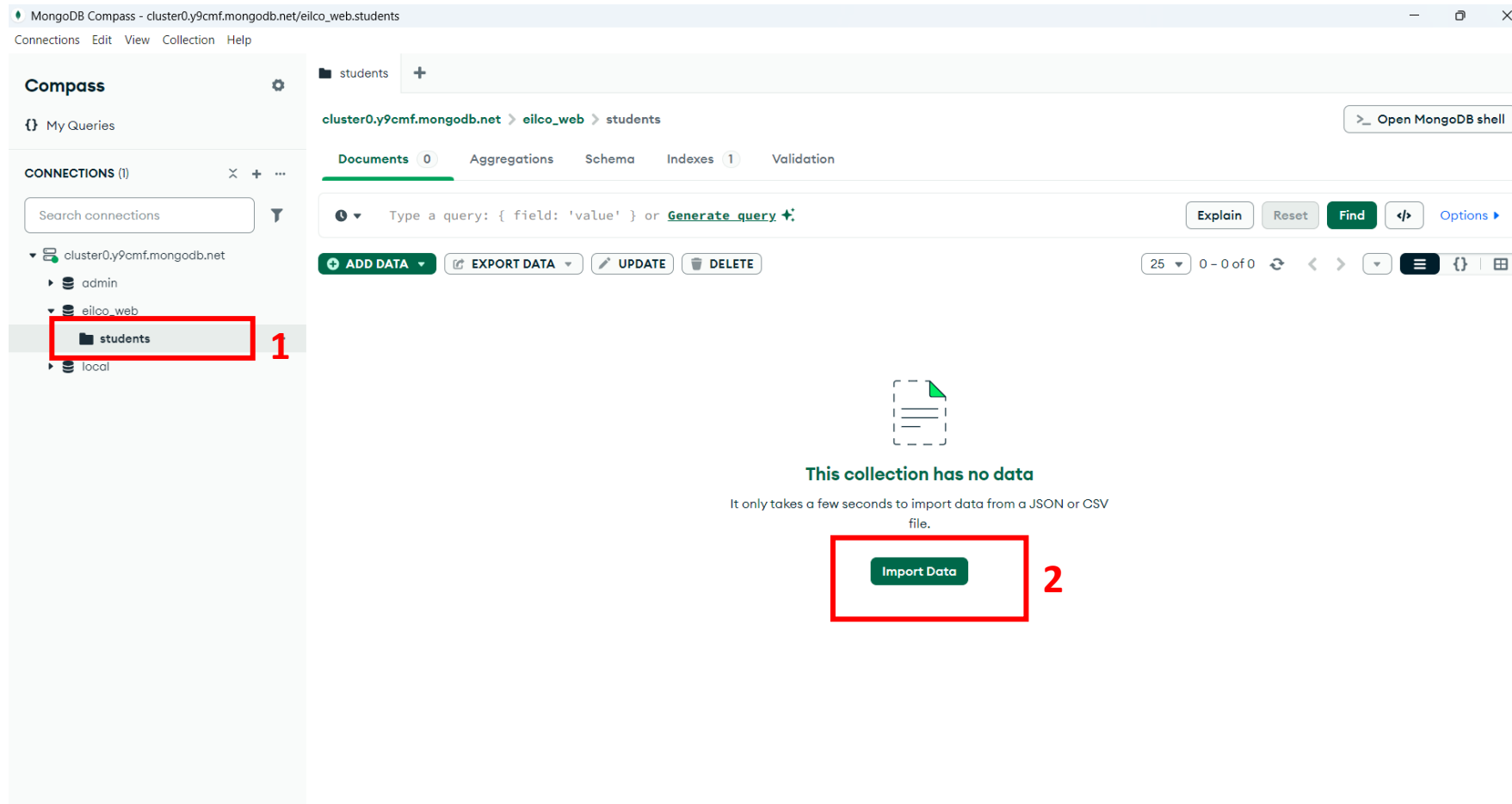
Démarrer le logiciel MongoDB Compass et se connecter à la base de données



Insertion des données via MongoDB Compass

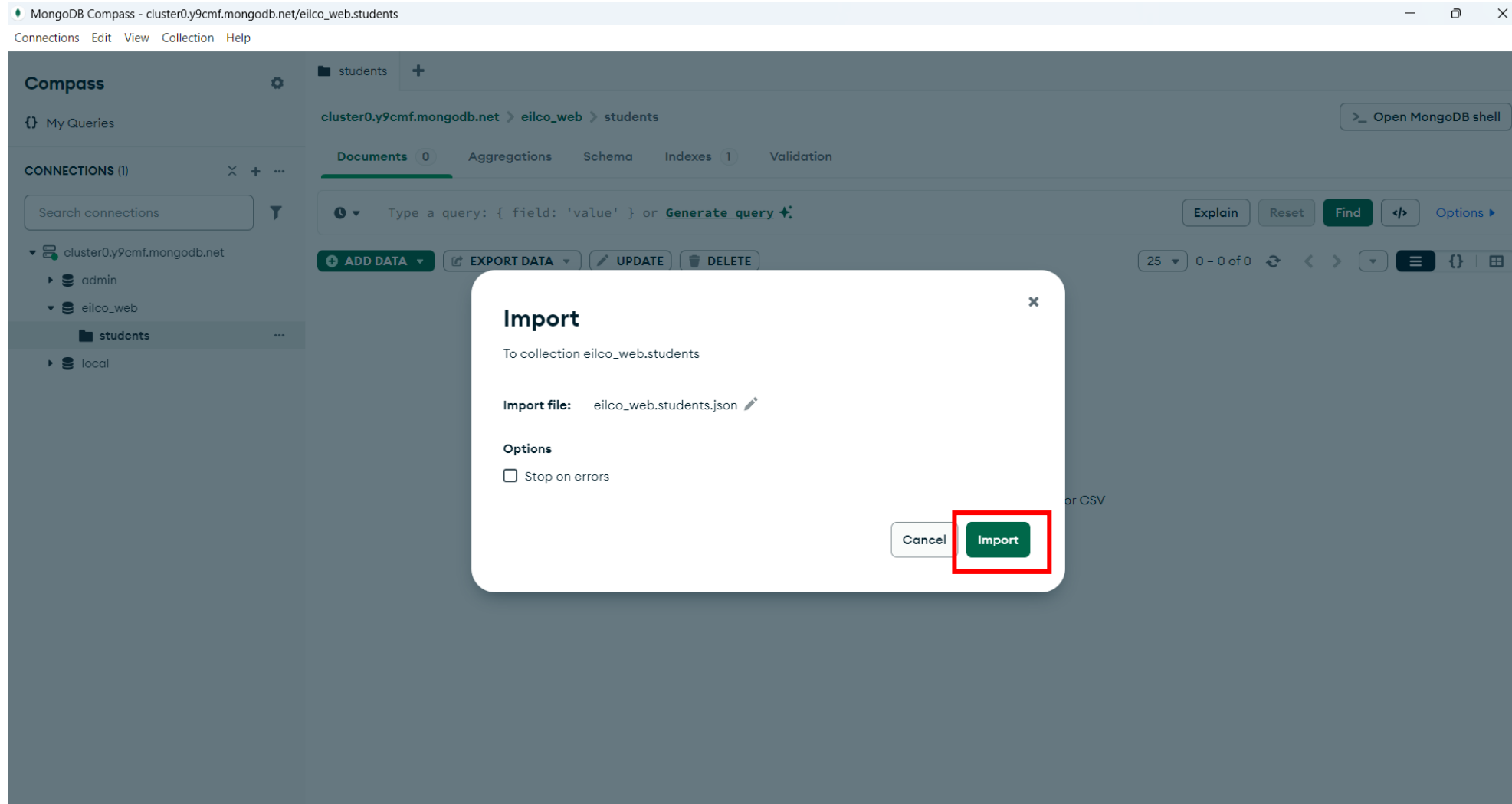
Téléchargez ce fichier de données : [eilco_web.students.json](#)

Sélectionnez la collection students puis « Import Data »



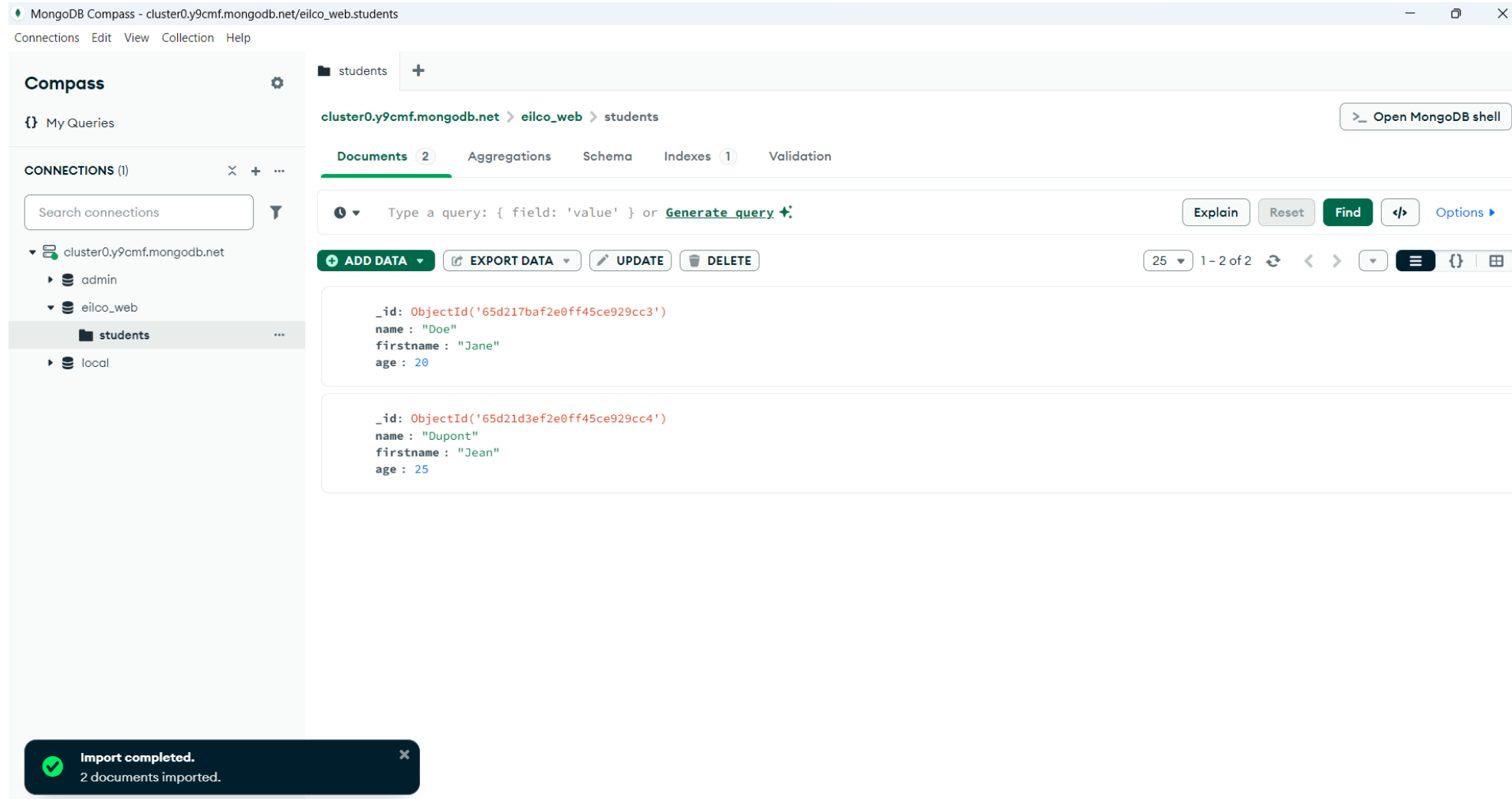
Insertion des données via MongoDB Compass

Sélectionner le fichier `eilco_web.students.json` puis cliquer sur « Import »



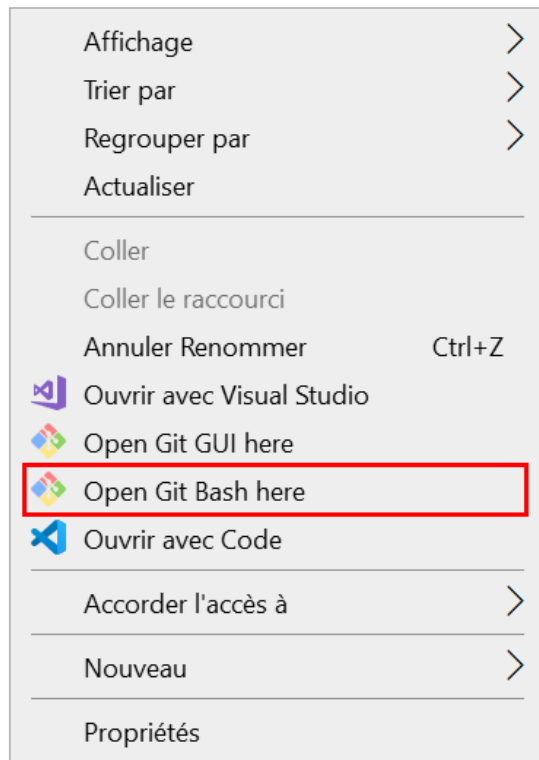
Insertion des données via MongoDB Compass

Les données ont bien été importées



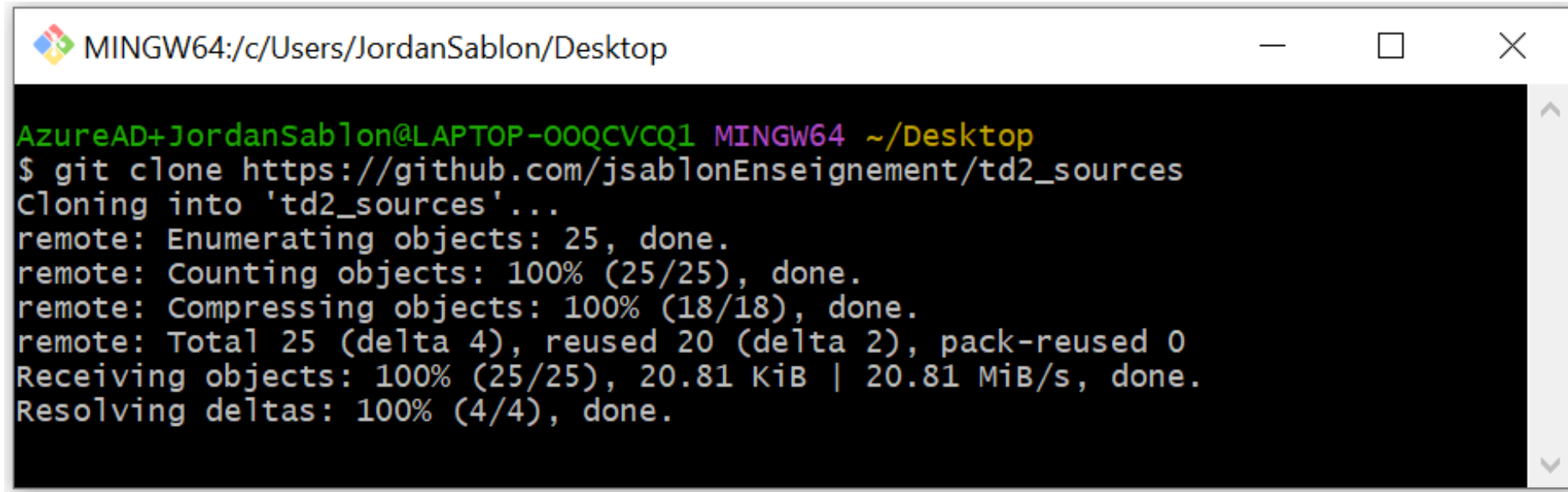
Récupération du code source du projet & installation des dépendances

1. Se positionner à l'emplacement de votre choix (où se trouvera le futur dossier du projet)
2. Ouvrir un invité de commande à l'emplacement précédent :



Récupération du code source du projet & installation des dépendances

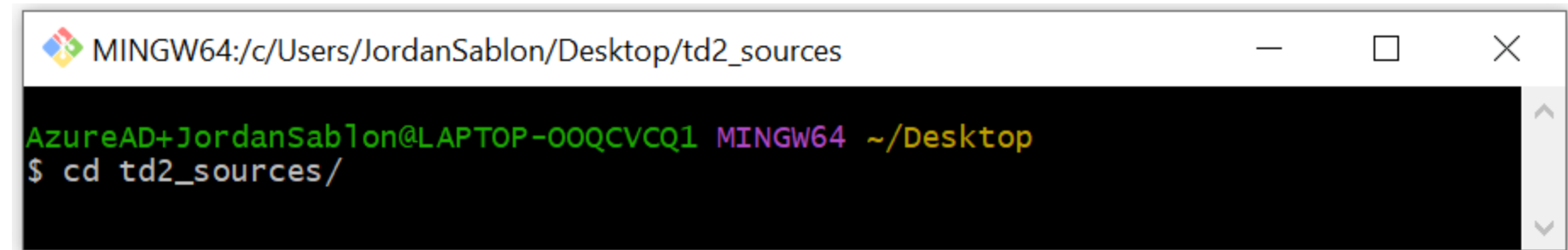
3. Cloner le répertoire Git : https://github.com/jsablonEnseignement/td2_sources



```
MINGW64:/c/Users/JordanSablon/Desktop

AzureAD+JordanSablon@LAPTOP-00QCVCQ1 MINGW64 ~/Desktop
$ git clone https://github.com/jsablonEnseignement/td2_sources
Cloning into 'td2_sources'...
remote: Enumerating objects: 25, done.
remote: Counting objects: 100% (25/25), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 25 (delta 4), reused 20 (delta 2), pack-reused 0
Receiving objects: 100% (25/25), 20.81 KiB | 20.81 MiB/s, done.
Resolving deltas: 100% (4/4), done.
```

4. Se rendre dans le dossier précédemment cloné :

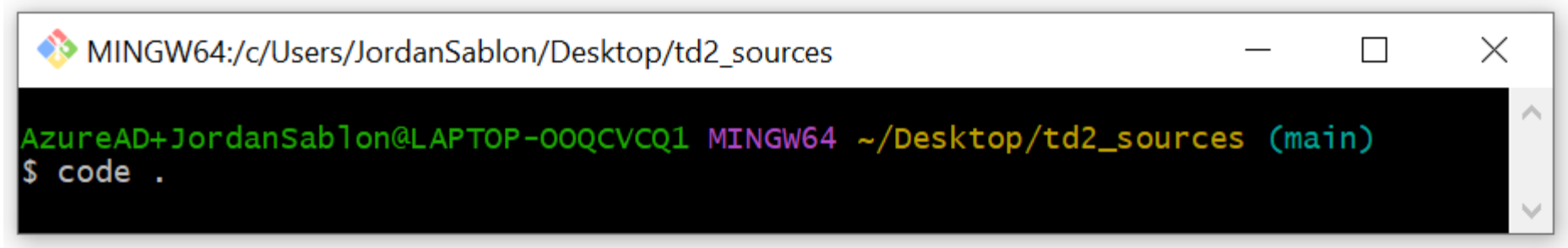


```
MINGW64:/c/Users/JordanSablon/Desktop/td2_sources

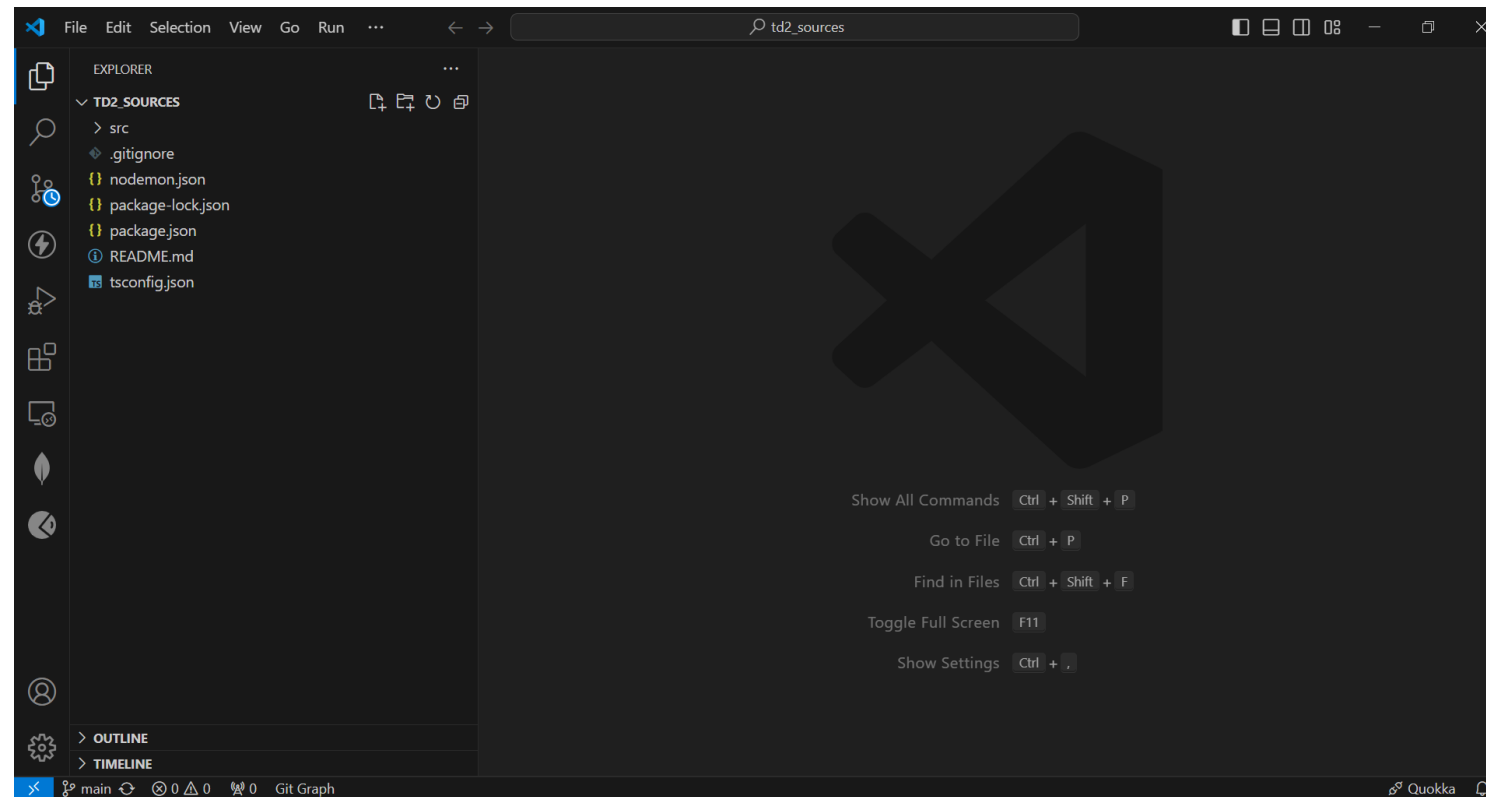
AzureAD+JordanSablon@LAPTOP-00QCVCQ1 MINGW64 ~/Desktop
$ cd td2_sources/
```

Récupération du code source du projet & installation des dépendances

5. Ouvrir le projet dans VSCode :

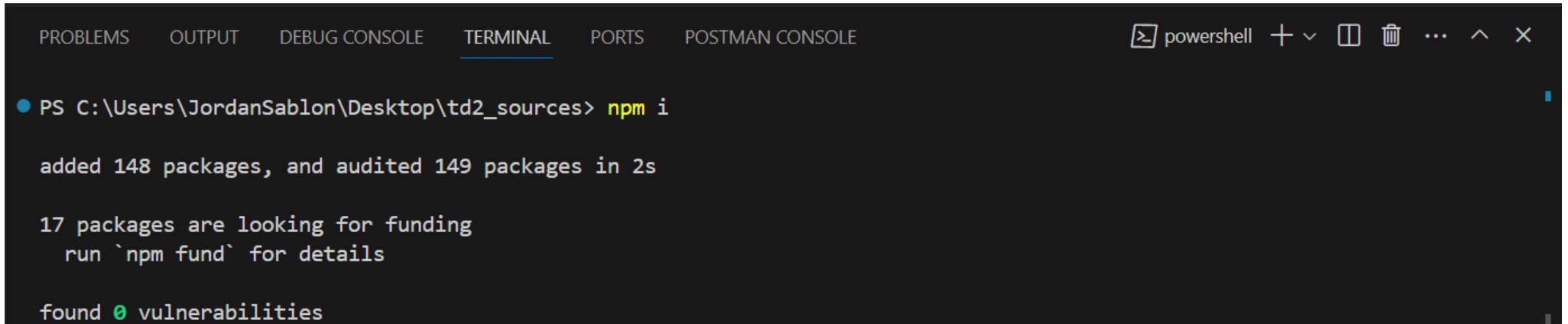


```
MINGW64:/c/Users/JordanSablon/Desktop/td2_sources  
AzureAD+JordanSablon@LAPTOP-OOQCVCQ1 MINGW64 ~/Desktop/td2_sources (main)  
$ code .
```



Récupération du code source du projet & installation des dépendances

Depuis le terminal de VSCode, exécuter la commande suivante :



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  POSTMAN CONSOLE  powershell + - [ ] [X] ... ^ X

● PS C:\Users\JordanSablon\Desktop\td2_sources> npm i

added 148 packages, and audited 149 packages in 2s

17 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
```

Les dépendances nécessaires au bon fonctionnement de l'application viennent d'être installées

Remplacez la chaine de connexion à la base de données dans le fichier **index.ts**

```
mongoose.connect(  
  "mongodb+srv://<username>:<password>@<domain>.mongodb.net/<db_name>?retryWrites=true&w=majority);  
const db = mongoose.connection;
```

/!\ L'URL est à adapter selon vos informations

Démarrage du serveur

```
PS C:\Users\JordanSablon\Documents\dev\eilco_web\backend> npm start

> backend@1.0.0 start
> npx tsc && nodemon

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): src\**\*
[nodemon] watching extensions: ts
[nodemon] starting `ts-node-esm src/index.ts`
Server running at http://127.0.0.1:5000/
Connected successfully
```

Créez un dossier **models** dans le dossier **/src**

Créez un fichier **students.model.ts** dans le dossier **/models**

Implémentez le schéma représentant la collection **students** donné à la diapo suivante

Création du modèle

```
import { model, Schema, Model, Types, Document } from "mongoose";

const StudentSchema: Schema = new Schema(
  {
    name: {
      type: String,
      required: true,
    },
    firstname: {
      type: String,
      required: true,
    },
    age: {
      type: Number,
      default: 0,
    },
  },
  { versionKey: false }
);

export interface IStudent {
  _id: Types.ObjectId;
  name: String;
  firstname: String;
  age: Number;
}

export interface Student extends Omit<IStudent, "_id">, Document {}

export const StudentModel = model<Student, Model<Student>>>(
  "Student",
  StudentSchema,
  "students"
);
```

Récupération de tous les étudiants

Dans le fichier **students.service.ts**, importer le modèle ainsi que l'interface précédemment créés :

```
import { Student, StudentsModel } from "../models/students.model";
```

Modifier la fonction *getStudents* :

```
const getStudents = async () => {  
  return await StudentsModel.find();  
};
```

- ① On applique une fonction `find()` sur le modèle **StudentsModel** afin de récupérer tous les étudiants dans la collection **students** de notre base de données
- ① On utilise l'instruction *await* afin de forcer l'attendre du retour de la base de données pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Récupération de tous les étudiants

Dans le fichier **students.controller.ts**, modifier la fonction *getStudents* :

```
export const getStudents = async (req: any, res: any) => {  
  const students = await StudentsService.getStudents();  
  return res.status(200).json(students);  
};
```

- ① On utilise l'instruction *await* afin de forcer l'attendre l'exécution de notre service pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Récupération d'un étudiant par son id

Dans le fichier **students.service.ts**, modifier la fonction *getStudent* :

```
const getStudent = async (id: string) => {  
  return await StudentsModel.findById(id);  
};
```

- ① On applique la fonction `findById()`, avec comme paramètre l'id de l'étudiant, sur le modèle **StudentsModel** afin de récupérer l'étudiant correspondant dans la collection **students** de notre base de données
- ① On utilise l'instruction *await* afin de forcer l'attendre du retour de la base de données pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Récupération d'un étudiant par son id

Dans le fichier **students.controller.ts**, modifier la fonction *getStudent* :

```
export const getStudent = async (req: any, res: any) => {  
  const { id } = req.params;  
  const student = await StudentsService.getStudent(id);  
  return res.status(200).json(student);  
};
```

- ① On utilise l'instruction *await* afin de forcer l'attendre l'exécution de notre service pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Création d'un étudiant

Dans le fichier **students.service.ts**, modifier la fonction *createStudent* :

```
const createStudent = async (studentToCreate: Student) => {  
  const newStudent = new StudentsModel(studentToCreate);  
  await newStudent.save();  
  return getStudents();  
};
```

- ① On crée une nouvelle instance du modèle **StudentsModel** avec les données de notre étudiant puis on applique la fonction *save()* afin d'insérer l'étudiant en base de données
- ① On utilise l'instruction *await* afin de forcer l'attendre du retour de la base de données pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Création d'un étudiant

Dans le fichier **students.controller.ts**, modifier la fonction *createStudent* :

```
export const createStudent = async (req: any, res: any) => {  
  const studentToCreate = req.body;  
  const students = await StudentsService.createStudent(studentToCreate);  
  return res.status(200).json(students);  
};
```

- ① On utilise l'instruction *await* afin de forcer l'attendre l'exécution de notre service pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Mise à jour d'un étudiant

Dans le fichier **students.service.ts**, modifier la fonction *updateStudent* :

```
const updateStudent = (id: string, studentToUpdate: Student) => {  
  await Student.updateOne(  
    {  
      _id: new Types.ObjectId(id),  
    },  
    studentToUpdate  
  );  
  return getStudents();  
};
```

- ① On applique une fonction `updateOne()` sur le modèle **StudentsModel** afin de mettre à jour l'étudiant concerné dans la collection **students** de notre base de données
- ① On utilise l'instruction *await* afin de forcer l'attendre du retour de la base de données pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Mise à jour d'un étudiant

Dans le fichier **students.controller.ts**, modifier la fonction *updateStudent* :

```
export const updateStudent = async (req: any, res: any) => {  
  const { id } = req.params;  
  const studentUpdated = await StudentsService.updateStudent(  
    id,  
    req.body  
  );  
  return res.status(200).json(studentUpdated);  
};
```

- ① On utilise l'instruction *await* afin de forcer l'attendre l'exécution de notre service pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Suppression d'un étudiant

Dans le fichier **students.service.ts**, modifier la fonction *deleteStudent* :

```
const deleteStudent = async (id: string) => {  
  await StudentsModel.deleteOne({ _id: new Types.ObjectId(id) });  
};
```

- ① On applique une fonction `deleteOne()` sur le modèle **StudentsModel** afin de supprimer l'étudiant correspondant de la collection **students** de notre base de données
- ① On utilise l'instruction *await* afin de forcer l'attendre du retour de la base de données pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)
- ① On fait le choix de ne pas retourner de données

Suppression d'un étudiant

Dans le fichier **students.controller.ts**, modifier la fonction *deleteStudent* :

```
export const deleteStudent = async (req: any, res: any) => {  
  const { id } = req.params;  
  await StudentsService.deleteStudent(id);  
  return res  
    .status(200)  
    .json(`Student with id ${id} successfully deleted`);  
};
```

- ① On utilise l'instruction *await* afin de forcer l'attendre l'exécution de notre service pour retourner les étudiants. On précise alors que notre fonction est asynchrone (*async*)

Objectifs :

- éviter d'envoyer des informations sensibles sur votre repo git
- spécifier une configuration variable selon les environnements tout en conservant un code générique

Installez la dépendance **dotenv**

```
npm install dotenv
```

Importez **dotenv** dans le fichier **index.ts**

```
import dotenv from "dotenv";
```

Ajoutez cette ligne de code en début de fichier (après les imports)

```
dotenv.config();
```

Créez un fichier **.env** à la racine du projet (au même niveau que le **package.json**) qui contiendra les informations suivantes :

```
DB_USERNAME=your_username  
DB_PASSWORD=your_password  
DB_DOMAIN=your_domain.mongodb.net  
DB_NAME=your_db_name
```

/!\ Adapter les valeurs de droite ci-dessus en fonction de vos informations de connexion à la base de données

Modifiez votre ligne de connexion à la base de données dans le fichier **index.ts** pour utiliser les variables ci-dessus au lieu des valeurs :

```
mongoose.connect(  
  `mongodb+srv://${process.env.DB_USERNAME}:${process.env.DB_PASSWORD}@${process.env.DB_DOMAIN}/${process.env.DB_NAME}?retryWrites=true&w=majority`  
)
```