



CRUD OPERATING WITH PYTHON AND MONGODB™

MR. KONATE

TEAM

TRAORE
CHEICK
OUMAR TIDIANE

SORO KAYAZIN
LEVI



SOMMAIRES

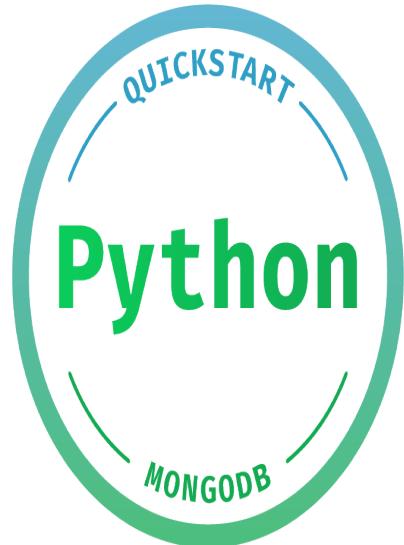
I-) Opérations MongoDB de base en Python



Opérations MongoDB de base en Python

Évaluez ce démarrage rapide

Comme Python ? Vous voulez démarrer avec MongoDB ? Bienvenue dans ce guide de démarrage rapide ! Je vais vous montrer comment configurer une base de données Atlas avec quelques exemples de données à explorer. Ensuite, vous créerez des données et apprendrez à les lire, les mettre à jour et les supprimer.



Conditions préalables

Vous aurez besoin des éléments suivants installés sur votre ordinateur pour suivre ce didacticiel :

- Une version à jour de Python 3. J'ai écrit le code de ce tutoriel en Python 3.8, mais il devrait fonctionner correctement dans la version 3.6+.
- Un éditeur de code de votre choix. je recommande soit **PyCharmName** ou la gratuité **Code VS** avec l'officiel **Extension Python**

Démarrer un cluster MongoDB sur Atlas

Maintenant que votre environnement local est configuré, il est temps de créer une base de données MongoDB avec laquelle travailler et de charger des exemples de données que vous pouvez explorer et modifier.

Vous pouvez créer une base de données sur votre machine de développement, mais il est plus facile de démarrer sur le service hébergé Atlas sans avoir à apprendre à configurer un cluster MongoDB.

Démarrer avec un cluster M0 sur [Atlas](#) aujourd'hui. C'est gratuit pour toujours et c'est le moyen le plus simple d'essayer les étapes de cette série de blogs.

Vous devrez créer un nouveau cluster et le charger avec des exemples de données.

Si vous ne voulez pas regarder la vidéo, les étapes sont les suivantes :

Cliquez sur "Commencer gratuitement".

Entrez vos coordonnées et acceptez les conditions d'utilisation.

Créez un cluster Starter .

- Sélectionnez le même fournisseur de cloud auquel vous êtes habitué ou laissez-le tel quel. Choisissez une région qui vous convient.
- Vous pouvez modifier le nom du cluster si vous le souhaitez. J'ai appelé le mien "PythonQuickstart".

Il faudra quelques minutes pour que votre cluster soit provisionné, donc pendant que vous attendez, vous pouvez passer à l'étape suivante.

Il faudra quelques minutes pour que votre cluster soit provisionné, donc pendant que vous attendez, vous pouvez passer à l'étape suivante.

Configurez votre environnement

Vous devez configurer un virtualenv Python qui contiendra les bibliothèques que vous installez lors de ce démarrage rapide. Il existe plusieurs façons de configurer virtualenvs,

mais pour simplifier les choses, nous utiliserons celle incluse avec Python. Tout d'abord, créez un répertoire pour contenir votre code et votre virtualenv. Ouvrez votre terminal, `cd`dans ce répertoire, puis exécutez la commande suivante :

```
# Note:  
# On Debian & Ubuntu systems you'll first need to install  
virtualenv with:  
# sudo apt install python3-venv  
python3 -m venv venv
```

La commande ci-dessus créera un virtualenv dans un répertoire appelé `venv`. Pour activer le nouveau virtualenv, exécutez l'une des commandes suivantes, selon votre système :

```
# Run the following on OSX & Linux:  
source venv/bin/activate
```

```
# Run the following on Windows:  
.\\venv\\Scripts\\activate
```

Pour écrire des programmes Python qui se connectent à votre base de données MongoDB (ne vous inquiétez pas - vous le configurerez dans un instant !), vous devrez installer un pilote Python - une bibliothèque qui sait comment parler à MongoDB. En Python, vous avez deux choix ! Le pilote recommandé est **pymongo**

- c'est ce que je vais couvrir dans ce démarrage rapide. Cependant, si vous souhaitez écrire des programmes *asynchrones* avec MongoDB, vous devrez utiliser une bibliothèque appelée **Moteur**, qui est également entièrement pris en charge par MongoDB.

Pour installer PyMongo, exécutez la commande suivante :
python -m pip install pymongo[srv]==3.10.1

Pour ce didacticiel, nous utiliserons également une bibliothèque appelée **python-dotenv** pour charger la configuration. Exécutez également la commande ci-dessous pour l'installer :

python -m pip install python-dotenv==0.13.0

Configurer votre instance MongoDB

J'espère que votre cluster MongoDB devrait avoir fini de démarrer maintenant et qu'il fonctionne probablement depuis quelques minutes.

Les instructions suivantes étaient correctes au moment de la rédaction, mais peuvent changer, car nous améliorons constamment l'interface utilisateur d'Atlas :

Dans l'interface Web d'Atlas, vous devriez voir un bouton vert en bas à gauche de l'écran, indiquant "Commencer". Si vous cliquez dessus, une liste de contrôle des étapes à suivre pour configurer votre base de données s'affichera. Cliquez sur chacun des éléments de la liste (y compris l'élément facultatif "Charger des exemples de données"), et cela vous aidera à travers les étapes de configuration.

Créer un utilisateur

En suivant les étapes "Commencer", créez un utilisateur avec "Accès en lecture et en écriture à n'importe quelle base de données". Vous pouvez lui donner un nom d'utilisateur et un mot de passe de votre choix - prenez-en une copie, vous en aurez besoin dans une minute. Utilisez le bouton "générer automatiquement un mot de passe sécurisé" pour vous assurer que vous disposez d'un long mot de passe aléatoire qui peut également être collé en toute sécurité dans votre chaîne de connexion ultérieurement.

Autoriser une adresse IP

Lors du déploiement d'une application avec des données sensibles, vous ne devez autoriser que l'adresse IP des serveurs qui doivent se connecter à votre base de données. Pour autoriser l'adresse IP de votre machine de développement, sélectionnez "Accès réseau", cliquez sur le bouton "Ajouter une adresse IP", puis cliquez sur "Ajouter l'adresse IP actuelle" et cliquez sur "Confirmer".

**Mrs. KONATE
(web programming
Teacher)**

Connectez-vous à votre base de données

La dernière étape de la liste de contrôle "Get Started" est "Se connecter à votre cluster". Sélectionnez "Connecter votre application" et sélectionnez "Python" avec une version "3.6 ou ultérieure".

Assurez-vous que l'étape 2 a "Chaîne de connexion uniquement" en surbrillance et appuyez sur le bouton "Copier" pour copier l'URL dans votre table de montage. Enregistrez-le au même endroit où vous avez enregistré votre nom d'utilisateur et votre mot de passe. Notez que l'URL a <password> comme espace réservé pour votre mot de passe. Vous devez coller votre mot de passe ici, en remplaçant tout l'espace réservé, y compris les caractères '<' et '>'.

Il est maintenant temps d'écrire du code Python pour se connecter à votre base de données MongoDB !

Dans votre éditeur de code, créez un fichier Python dans votre répertoire de projet appelé basic_operations.py. Entrez le code suivant :

La dernière étape de la liste de contrôle "Get Started" est "Se connecter à votre cluster".

Sélectionnez "Connecter votre application" et sélectionnez "Python" avec une version "3.6 ou ultérieure".

Assurez-vous que l'étape 2 a "Chaîne de connexion uniquement" en surbrillance et appuyez sur le bouton "Copier" pour copier l'URL dans votre table de montage. Enregistrez-le au même endroit où vous avez enregistré votre nom d'utilisateur et votre mot de passe. Notez que l'URL a <password> comme espace réservé pour votre mot de passe. Vous devez coller votre mot de passe ici, en remplaçant tout l'espace réservé, y compris les caractères '<' et '>'.

Il est maintenant temps d'écrire du code Python pour se connecter à votre base de données MongoDB !

Dans votre éditeur de code, créez un fichier Python dans votre répertoire de projet appelé basic_operations.py. Entrez le code suivant :

**Mrs. KONATE
(web programming
Teacher)**

La dernière étape de la liste de contrôle "Get Started" est "Se connecter à votre cluster".

Sélectionnez "Connecter votre application" et sélectionnez "Python" avec une version "3.6 ou ultérieure".

Assurez-vous que l'étape 2 a "Chaîne de connexion uniquement" en surbrillance et appuyez sur le bouton "Copier" pour copier l'URL dans votre table de montage. Enregistrez-le au même endroit où vous avez enregistré votre nom d'utilisateur et votre mot de passe. Notez que l'URL a <password> comme espace réservé pour votre mot de passe. Vous devez coller votre mot de passe ici, en remplaçant tout l'espace réservé, y compris les caractères '<' et '>'.

Il est maintenant temps d'écrire du code Python pour se connecter à votre base de données MongoDB !

Dans votre éditeur de code, créez un fichier Python dans votre répertoire de projet appelé basic_operations.py. Entrez le code suivant :

```
1 import datetime # Cela sera nécessaire plus tard
2 importer le système d' exploitation
3
4 depuis dotenv importer load_dotenv
5 de pymongo importer MongoClient
6
7 # Charger la configuration à partir d'un fichier .env :
8 load_dotenv ()
9 MONGODB_URI = os . environ [ 'MONGODB_URI' ]
10
11 # Connectez-vous à votre cluster MongoDB :
12 client = MongoClient ( MONGODB_URI )
13
14 # Lister toutes les bases de données du cluster :
15 pour db_info dans le client . list_database_names () :
16     imprimer ( db_info )
```

Pour exécuter ceci, vous devrez définir la variable d'environnement MONGODB_URI sur la chaîne de connexion que vous avez obtenue ci-dessus. Vous pouvez le faire de deux manières. Tu peux:

- Exécutez une commande `export`(ou `set`sous Windows) pour définir la variable d'environnement chaque fois que vous configurez votre session.
- Enregistrez l'URI dans un fichier de configuration qui ne doit *jamais* être ajouté au contrôle de révision.

Je vais vous montrer comment adopter la deuxième approche. N'oubliez pas qu'il est très important de ne pas publier accidentellement vos informations d'identification sur git ou ailleurs, alors ajoutez `.env`-les à votre `.gitignore`fichier si vous utilisez git. La `python-dotenv`bibliothèque charge la configuration à partir d'un fichier dans le répertoire courant appelé `.env`. Créez un `.env`fichier dans le même répertoire que votre code et collez la configuration ci-dessous, en remplaçant l'URI d'espace réservé par votre propre URI MongoDB.

```
1 #Unix : _  
2 export MONGODB_URI =  
'mongodb+srv://yourusername:yourpasswordgoeshere@pythonquickstar  
t-123ab.mongodb.net/test?retryWrites=true&w=majority'
```

L'URI contient votre nom d'utilisateur et votre mot de passe (donc gardez-le en sécurité !) et le nom d'hôte d'un serveur DNS qui fournira des informations à PyMongo sur votre cluster. Une fois que PyMongo a récupéré les détails de votre cluster, il se connectera au serveur MongoDB principal et commencera à faire des requêtes.

Maintenant, si vous exécutez le script Python, vous devriez voir une sortie semblable à la suivante :

SEMBLABLE À LA SUITE .

```
1 $ python basic_operations . py
2 sample_airbnb
3 sample_analytics
4 sample_geospatial
5 sample_mflix
6 échantillons_fournitures
7 échantillon_formation
8 sample_weatherdata
9 twitter_analytics
10 administrateur
11 local
```

Vous venez de connecter votre programme Python à MongoDB et de lister les bases de données de votre cluster ! Si vous ne voyez pas cette liste, vous n'avez peut-être pas réussi à charger les exemples de données dans votre cluster ; Vous voudrez peut-être revenir en arrière jusqu'à ce que l'exécution de cette commande affiche la liste ci-dessus.

Dans le code ci-dessus, vous avez utilisé la `list_database_names` méthode pour répertorier les noms de bases de données dans le cluster. L' `MongoClient` instance peut également être utilisée comme mappage (comme un `dict`) pour obtenir une référence à une base de données spécifique. Voici du code pour jeter un œil aux collections à l'intérieur de la base de `sample_mflix` données. Collez-le à la fin de votre fichier Python :

```
1 # Récupère une référence à la base de données 'sample_mflix'
2 db = client [ 'sample_mflix' ]
3
4 # Lister toutes les collections dans 'sample_mflix' :
5 collections = db . list_collection_names ()
6 pour la collecte dans les collections :
7     impression ( collection )
```

L'exécution de ce morceau de code devrait produire ce qui suit :

```
1 $ python basic_operations . py
2 films
3 séances
4 commentaires
5 utilisateurs
6 théâtres
```

Une base de données se comporte également comme un mappage de collections à l'intérieur de cette base de données. Une collection est un ensemble de documents, de la même manière qu'une table contient des lignes dans une base de données relationnelle traditionnelle. Le code suivant recherche un seul document dans la `movies` collection :

```
1 # Import the ` pprint` function to print nested data:  
2 from pprint import pprint  
3  
4 # Get a reference to the 'movies' collection:  
5 movies = db['movies']  
6  
7 # Get the document with the title 'Blacksmith Scene':  
8 pprint(movies.find_one({'title': 'Blacksmith Scene'}))
```

Lorsque vous exécutez le code ci-dessus, il recherche un document appelé "Scène de forgeron" dans la collection "films". Ça ressemble un peu à ça :

```
{'_id': ObjectId('573a1390f29313caabcd4135'),  
'awards': {'nominations': 0, 'text': '1 win.', 'wins': 1},  
'cast': ['Charles Kayser', 'John Ott'],  
'countries': ['USA'],  
'directors': ['William K.L. Dickson'],  
'fullplot': 'A stationary camera looks at a large anvil with a blacksmith '  
                  'behind it and one on either side. The smith in the middle draws  
'
```

```

'a heated metal rod from the fire, places it on the anvil, and '
'all three begin a rhythmic hammering. After several blows, the
'
'metal goes back in the fire. One smith pulls out a bottle of '
'beer, and they each take a swig. Then, out comes the glowing '
'metal and the hammering resumes.',

'genres': ['Short'],
'imdb': {'id': 5, 'rating': 6.2, 'votes': 1189},
'lastupdated': '2015-08-26 00:03:50.133000000',
'num_mflix_comments': 1,
'plot': 'Three men hammer on an anvil and pass a bottle of beer
around.',
'rated': 'UNRATED',
'released': datetime.datetime(1893, 5, 9, 0, 0),
'runtim': 1,
'title': 'Blacksmith Scene',
'tomatoes': {'lastUpdated': datetime.datetime(2015, 6, 28, 18, 34, 9),
             'viewer': {'meter': 32, 'numReviews': 184, 'rating': 3.0}},
'type': 'movie',
'year': 1893}

```

C'est un film d'une minute tourné en 1893 - c'est comme une vidéo YouTube d'il y a près de 130 ans ! Les données ci-dessus constituent un seul document. Il stocke les données dans des champs accessibles par nom, et vous devriez pouvoir voir que le `title` champ contient la même valeur que celle que nous avons recherchée dans notre appel à `find_one` dans le code ci-dessus. La structure de chaque document d'une collection peut être différente les unes des

autres, mais il est généralement recommandé de suivre la même structure ou une structure similaire pour tous les documents d'une même collection.

Une diversion rapide sur BSON

MongoDB est souvent décrit comme une base de données JSON, mais il y a des preuves dans le document ci-dessus

qu'il ne stocke pas JSON. Un document MongoDB se compose de données stockées sous tous les types que JSON peut stocker, y compris les booléens, les entiers, les flottants, les chaînes, les tableaux et les objets (nous les appelons sous-documents). Cependant, si vous regardez les champs `_id` et `released`, ce sont des types que JSON ne peut pas stocker. En fait, MongoDB stocke les données dans un format binaire appelé BSON, qui inclut également le `ObjectId` type ainsi que les types natifs pour les nombres décimaux, les données binaires et les horodatages (qui sont convertis par PyMongo en `datetime` type natif de Python.)

Créer un document dans une collection

La `movies` collection contient beaucoup de données - 23539 documents, mais elle ne contient que des films jusqu'en 2015. L'un de mes films préférés, "Parasite", qui a remporté

un Oscar, est sorti en 2019, il n'est donc pas dans la base de données ! Vous pouvez corriger cette omission flagrante avec le code ci-dessous :

```
2     insert_result = films . insert_one ({
3         "title" : "Parasite" ,
4         "année" : 2020 ,
5         "complot" : "Une famille pauvre, les Kim, s'est frayé un
6             chemin pour devenir les serviteurs d'une famille riche, les
7             Parks. "
8             "Mais leur vie facile se complique lorsque leur tromperie
9             est menacée d'exposition." ,
10            "publié" : dateheure ( 2020 , 2 , 7 , 0 , 0 , 0 ),
11        })
12
13
14 # Enregistrez l'insert_id du document que vous venez de créer :
15 parasite_id = insert_result . inséré_id
16 print ( "_id du document inséré : {parasite_id}" . format (
17     parasite_id = parasite_id ))
```

Si vous insérez plusieurs documents en une seule fois, il peut être beaucoup plus efficace d'utiliser

la `insert_many` méthode, qui prend un tableau de documents à insérer. (Si vous ne faites que charger des documents dans votre base de données à partir de fichiers JSON stockés, vous devriez jeter un œil à [mongoimport](#).

Lire des documents d'une collection

L'exécution du code ci-dessus insérera le document dans la collection et imprimera son ID, ce qui est utile, mais pas grand-chose à regarder. Vous pouvez récupérer le document pour prouver qu'il a été inséré, avec le code suivant :

```
1 import bson # <- Placez cette ligne près du début du fichier
  si vous préférez.
2
3 # Recherchez le document que vous venez de créer dans la
  collection :
4 print ( films . find_one ( { '_id' : bson . ObjectId (
  parasite_id )}))
```

Le code ci-dessus recherchera un seul document qui correspond à la requête (dans ce cas, il recherche un fichier spécifique `_id`).

Si vous souhaitez rechercher *tous* les documents correspondant à une requête, vous devez utiliser la `find` méthode, qui renvoie un fichier `Cursor`. Un curseur chargera les données par lots, donc si vous essayez d'interroger toutes les données de votre collection, il commencera à produire des documents immédiatement - il ne charge pas toute la collection en mémoire sur votre ordinateur ! Vous pouvez parcourir les documents renvoyés dans un Cursor avec une `for` boucle. La requête suivante devrait imprimer un ou plusieurs documents - si vous avez exécuté votre script plusieurs fois, vous aurez inséré un document pour ce film à chaque fois que vous avez exécuté votre script ! (Ne vous souciez pas de les nettoyer - je vous montrerai comment faire cela dans un instant.)

```
1 # Recherchez les documents que vous avez créés dans la
  collection :
2 pour doc dans les films . trouver ( { "title" : "Parasite" } ):
3     pprint ( doc )
```

Supprimer des documents de la collection

Il est maintenant temps de nettoyer après vous-même ! Le code suivant supprimera tous les documents correspondants de la collection - en utilisant la même requête large qu'auparavant - tous les documents avec un `title` "Parasite":

```
1  films . delete_many (  
2      { "titre" : "Parasite" ,}  
3  )
```

Encore une fois, PyMongo a une `delete_one` méthode équivalente qui ne supprimera que le premier document correspondant trouvé par la base de données, au lieu de supprimer *tous* les documents correspondants.

II-)Connecter MongoDB Atlas Avec MongoDB Compass

***Mrs. KONATE*
(web programming
Teacher)**



Si votre application utilise MongoDB, MongoDB Atlas est la meilleure méthode pour l'héberger. Nous pouvons facilement déployer une base de données et la gérer dans MongoDB Atlas. Il fournit un outil polyvalent MongoDB Compass pour interagir avec la base de données à l'aide de l'interface graphique.

Ici, nous discuterons des étapes pour connecter MongoDB Atlas à MongoDB Compass.

Mrs. KONATE
**(web programming
Teacher)**

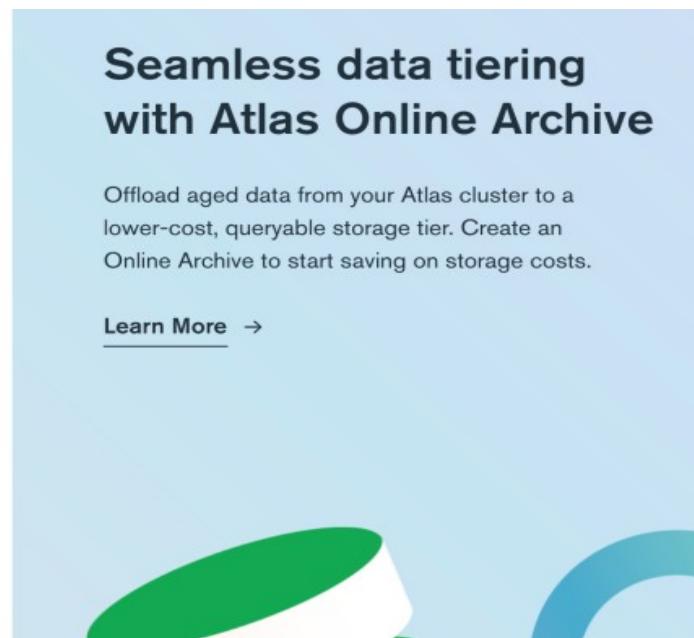
Create an account on MongoDB Atlas

Dans un premier temps, nous allons créer une base de données cloud sur MongoDB Atlas. Il offre le moyen le plus simple de déployer, d'exploiter et de faire évoluer MongoDB dans le Cloud.

Discutons-en étape par étape avec des captures d'écran et des explications.

Nous avons besoin d'un compte MongoDB pour créer une base de données. Alors, connectez-vous à la page MongoDB et créez un compte. Ou nous pouvons simplement créer un compte et nous connecter en utilisant notre compte Google.

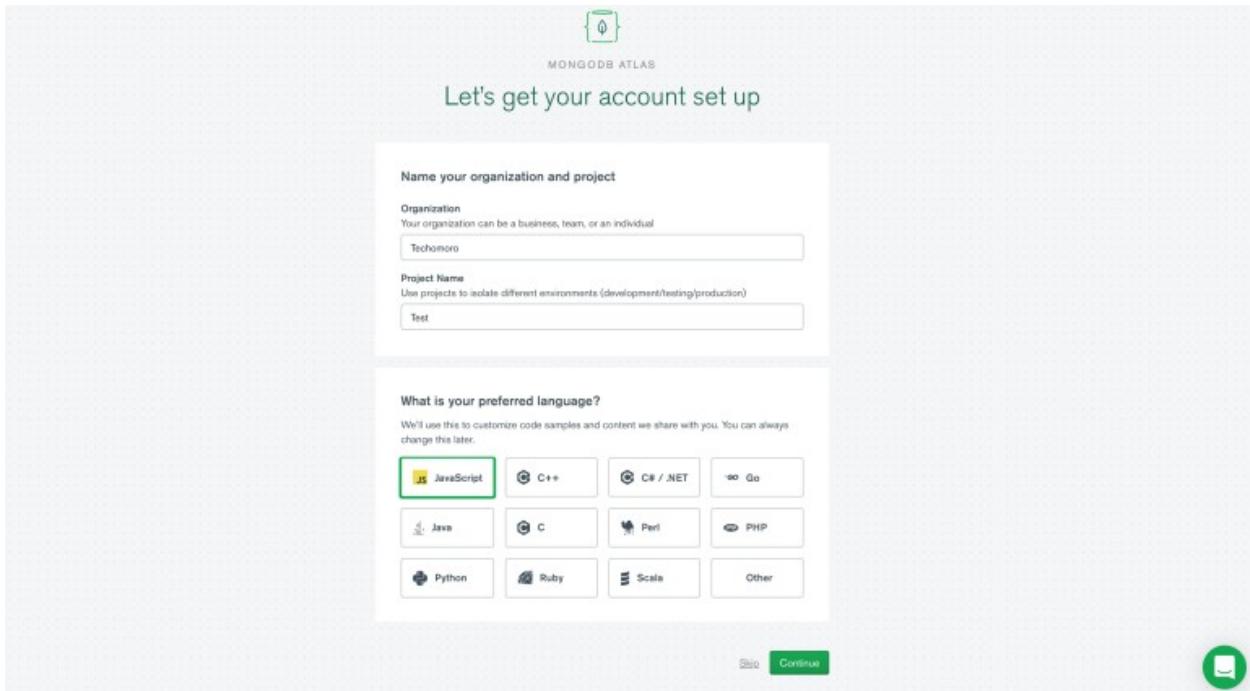
The image shows the MongoDB login page. It features the MongoDB logo at the top left. Below it, the text "Log in to your account" is centered. There are two main sign-in options: a blue "Log in with Google" button with a white "G" icon, and a "Email Address" input field below it. A horizontal line with the word "or" in the center separates these options. At the bottom left is a "Next" button, and at the bottom right is the text "Don't have an account? [Sign Up](#)".



Mrs. KONATE

(web programming Teacher)

Setup our MongoDB account



Après avoir créé un compte, nous verrons une page pour entrer les détails de notre organisation et de notre projet. Nous aurons également la possibilité de choisir notre langage de programmation préféré. Comme nous utilisons le backend Node/Express.js, nous pouvons choisir JavaScript dans cette catégorie.

Create a cluster

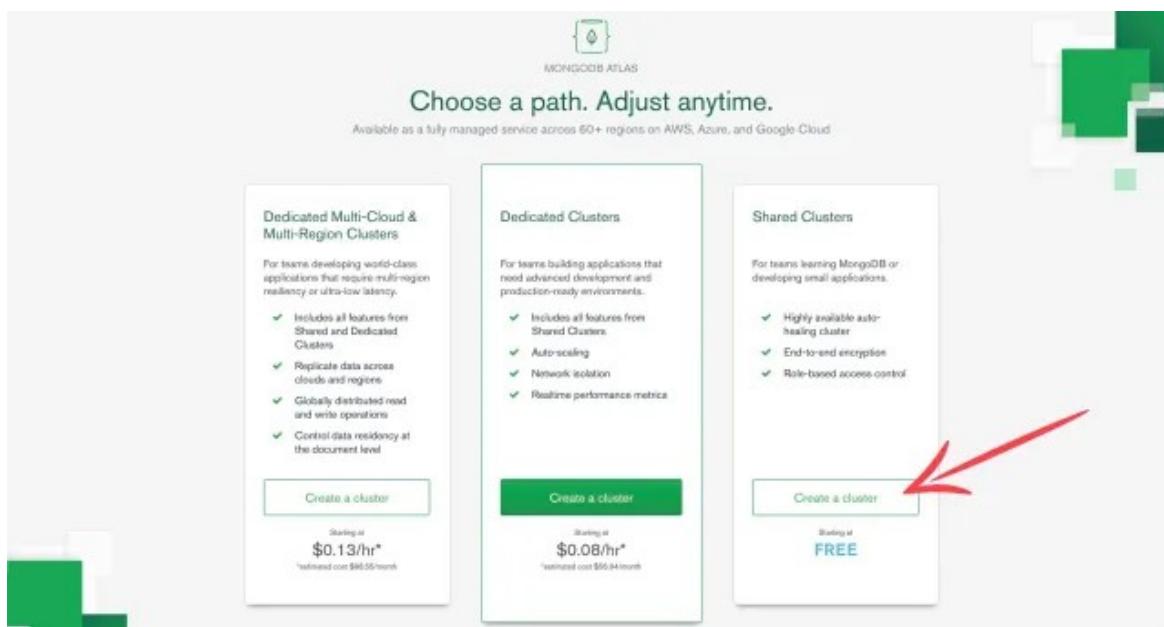
Le mot cluster peut vous dérouter. Il s'agit en fait d'un serveur sur lequel MongoDB stocke nos données. Choisir le bon plan avant de créer un cluster est important. Parlons d'abord des plans.

Cluster dédié multi-cloud et multi-régions - Si nous développons un projet pour des utilisateurs mondiaux, choisir ce plan est excellent. Parce qu'il stocke les données dans des serveurs dédiés situés dans plusieurs régions. Cela augmentera la vitesse de notre application.

Cluster dédié - Un cluster dédié est un serveur unique uniquement dédié à notre organisation/projet. Il peut fournir des performances avancées.

Cluster partagé - Les clusters partagés sont dédiés à un certain nombre d'utilisateurs. Cela affectera donc les performances de notre application aux heures de pointe.

Parce que nous sommes en mode développement, **Shared Cluster** nous convient. C'est **gratuit**.



Cluster dédié multi-cloud et multi-régions - Si nous développons un projet pour des utilisateurs mondiaux, choisir ce plan est excellent. Parce qu'il stocke les données dans des serveurs dédiés situés dans plusieurs régions. Cela augmentera la vitesse de notre application.

Cluster dédié - Un cluster dédié est un serveur unique uniquement dédié à notre organisation/projet. Il peut fournir des performances avancées.

Cluster partagé - Les clusters partagés sont dédiés à un certain nombre d'utilisateurs. Cela affectera donc les performances de notre application aux heures de pointe.

Parce que nous sommes en mode développement, **Shared Cluster** nous convient. C'est **gratuit**.

**Mrs. KONATE
(web programming
Teacher)**

CLUSTERS > CREATE A SHARED CLUSTER

Create a Shared Cluster

Welcome to MongoDB Atlas! We've recommended some of our most popular options, but feel free to customize your cluster to your needs. For more information, check our documentation.

Cloud Provider & Region
AWS, N. Virginia (us-east-1) ▾

 AWS
 Google Cloud
 Azure

★ Recommended region ⓘ

NORTH AMERICA	ASIA	EUROPE
 N. Virginia (us-east-1) ★	 Singapore (ap-southeast-1) ★	 Frankfurt (eu-central-1) ★
 Oregon (us-west-2) ★	 Mumbai (ap-south-1)	 Ireland (eu-west-1) ★
AUSTRALIA		
 Sydney (ap-southeast-2) ★		

Cluster Tier
M0 Sandbox (Shared RAM, 512 MB Storage) >

Encrypted

Additional Settings
MongoDB 4.4, No Backup >

Cluster Name
Cluster0 >

FREE
Free forever! This M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Back
 Create Cluster



Mrs. KONATE
(web programming
Teacher)

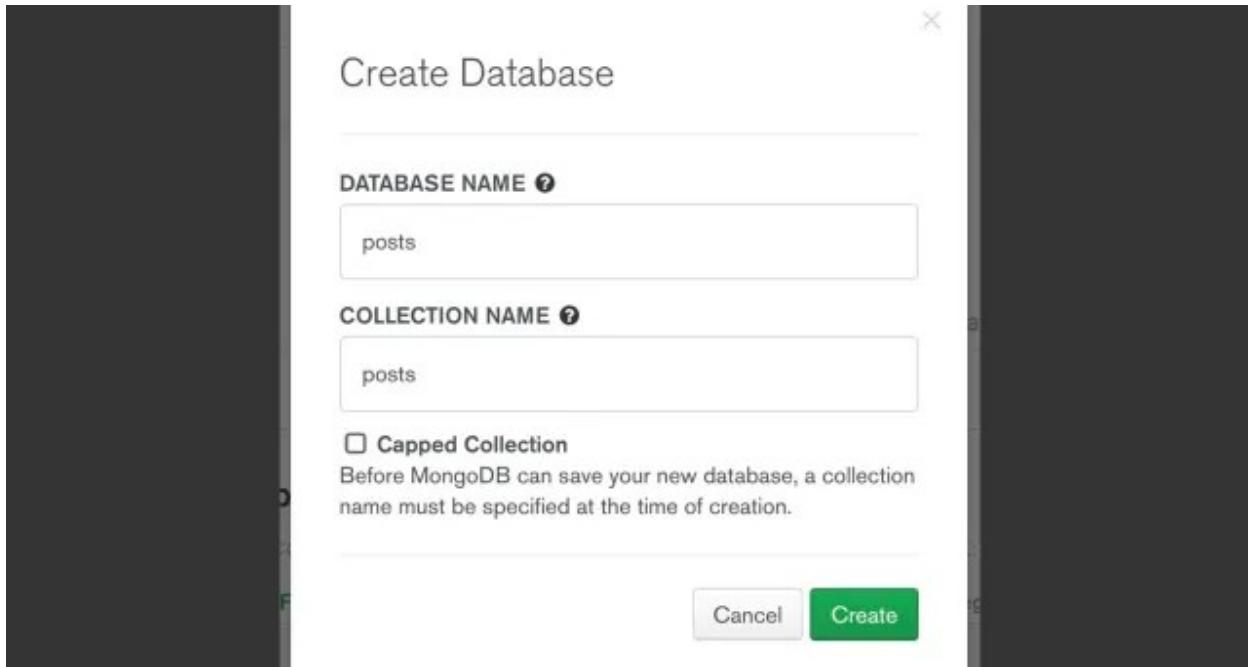
Create a collection

Le cluster est donc créé et nous devons maintenant créer une **collection**. Nous avons expliqué qu'un cluster est un serveur pour stocker nos bases de données de projet.

Une collection est simplement l'ensemble des bases de données associées à chaque projet que nous allons créer. Un cluster peut contenir plusieurs collections.

Donc à partir du cluster que nous avons créé, cliquez sur l' onglet **COLLECTIONS**.

The screenshot shows the MongoDB Atlas Cluster Overview page for a cluster named 'Cluster0'. A red arrow points to the 'COLLECTIONS' tab in the navigation bar at the bottom left of the cluster card. The cluster card displays information such as 'This is a Shared Tier Cluster', 'Logical Size 0.0 B', and 'Connections 0'. Other tabs visible include 'CONNECT', 'METRICS', and '...'. The top navigation bar includes links for Access Manager, Support, Billing, and a 'Create' button. The left sidebar shows 'TECHMODR > TEST' and lists 'Clusters' (Cluster0), 'Sandbox', 'REGION AWS / N. Virginia (us-east-1)', 'TYPE Replica Set - 3 nodes', and 'LINKED REALM APP'.



Nous aurons la possibilité de charger un exemple de jeu de données et nos propres données . Choisissez d' ajouter mes propres données parmi les options. Cela ouvrira un modal pour créer une base de données. Je donne des articles comme nom de base de données et des articles et des noms de collection.

Mrs. KONATE
(web programming Teacher)

Connect to the cluster

Nous pouvons maintenant rechercher les étapes pour connecter cette base de données à notre application backend. Pour cela, nous avons besoin d'une **chaîne de connexion**.

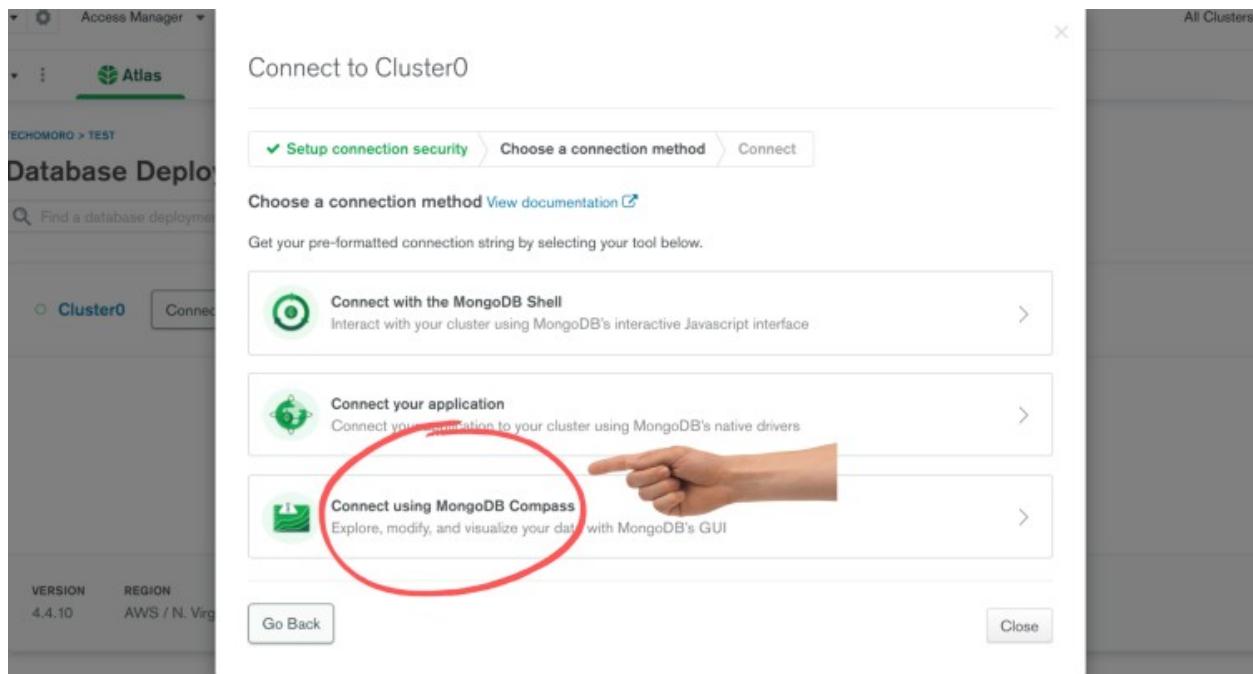
Avant d'obtenir une chaîne de connexion, nous devons effectuer certaines étapes.

En haut à droite, nous pouvons voir un onglet **Cmd Line Tools**. Cliquez dessus, et il nous montrera une page comme ci-dessous.

Cliquez sur le bouton **Connecter les instructions**.

The screenshot shows the MongoDB Cluster Overview page. At the top, there's a navigation bar with 'TECHOMORO > TEST > CLUSTERS'. Below it, the cluster details are shown: 'Cluster0', 'VERSION 4.4.5', 'REGION AWS N. Virginia (us-east-1)', and 'CLUS MO'. A horizontal menu bar includes 'Overview', 'Real Time', 'Metrics', 'Collections', 'Search', 'Profiler', 'Performance Advisor', 'Online Archive', and 'Cmd Line Tools' (which is underlined). Underneath, there's a section titled 'Connect To Your Cluster' with a note about connecting via MongoShell, URI, or Compass. A green button labeled 'Connect Instructions' is highlighted with a red arrow pointing to it. Below this, there's another section titled 'Manage Your Cluster From the Command Line Interface' with a note about provisioning and managing clusters with the MongoDB CLI. A green button labeled 'Install MongoDB CLI' is visible. At the bottom, there's a section for 'Binary Import and Export Tools' with a note about replacing placeholder variables.

Il affichera une vue comme ci-dessous, où nous devrions choisir l'option Connexion votre application .

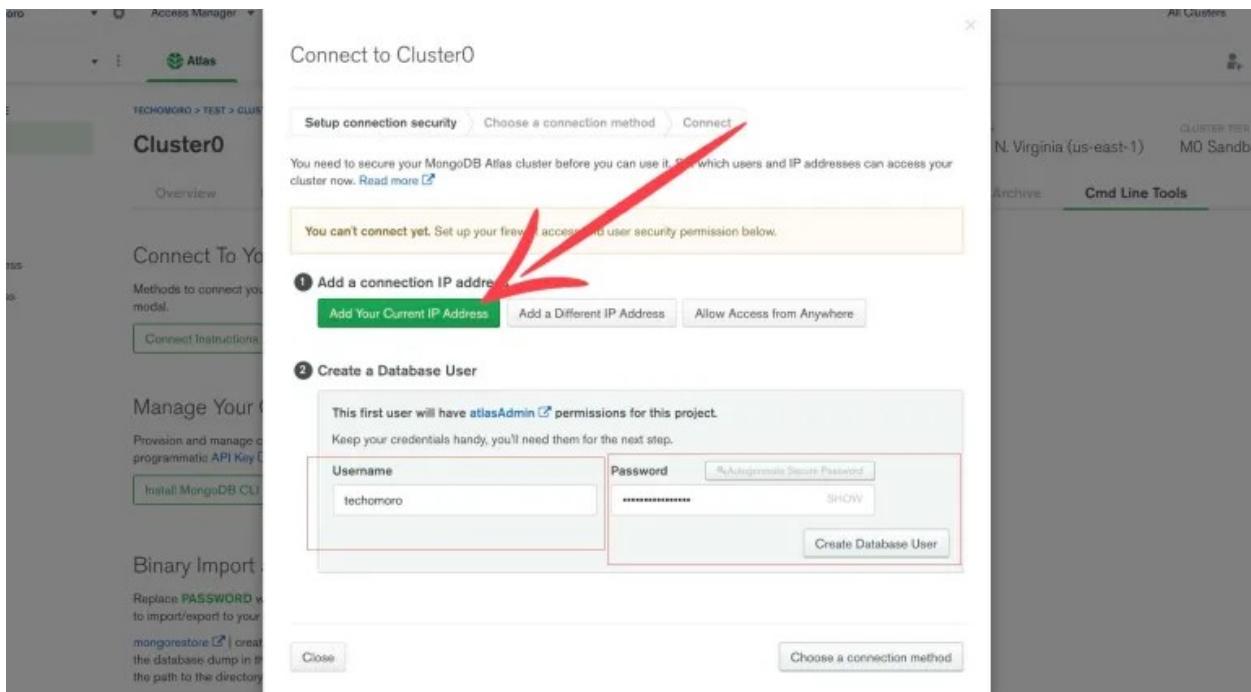


Pour configurer la connexion en toute sécurité, nous pouvons définir une **adresse IP** du serveur sur lequel notre backend s'exécute. Ainsi, la demande à la base de données ne peut se faire qu'à partir du serveur.

Mais dans notre cas, nous développons notre backend localement. Alors, laissez-nous choisir, **Ajoutez votre adresse IP actuelle**. Pour que nous puissions utiliser la base de données localement.

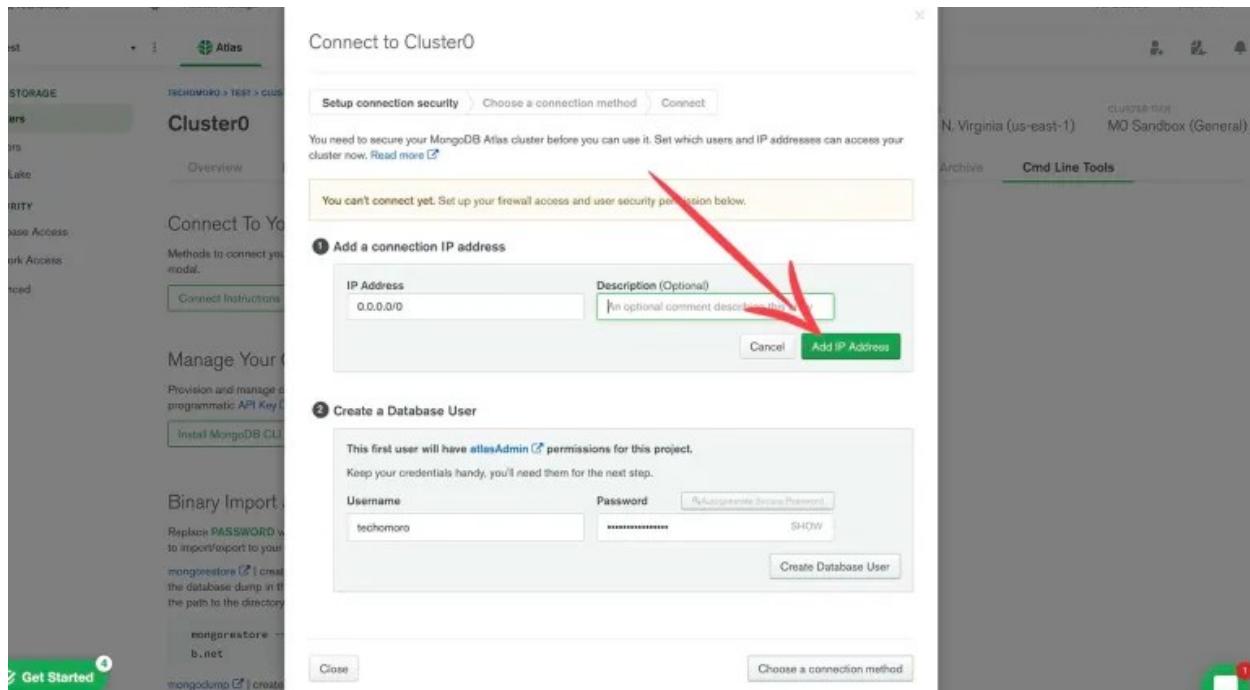
Mrs. KONATE
(web programming
Teacher)

Notez que, si plusieurs développeurs accèdent à la base de données à partir d'adresses IP différentes, il est préférable de choisir l'option **Autoriser l'accès depuis n'importe où**.

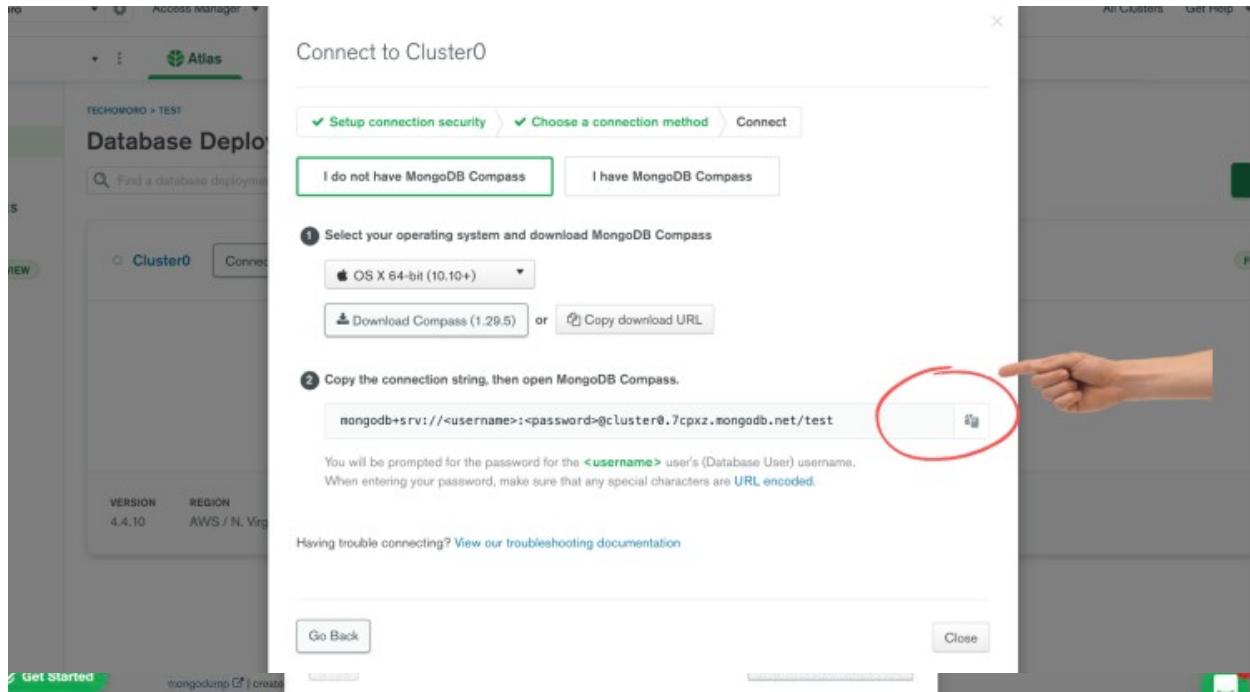


Mrs. KONATE
(web programming
Teacher)

Nous pouvons maintenant créer un nom d'utilisateur et un mot de passe pour la base de données . Ces données seront utilisées dans notre chaîne de connexion pour maintenir un accès sécurisé.



Sur l'écran suivant, nous obtiendrons une chaîne de connexion qui peut être utilisée pour se connecter à MongoDB Compass.



La chaîne de connexion copiée sera
dans un format comme ci-dessous.

```
mongodb+srv://<nom d'utilisateur> :<mot de  
passe>@cluster0.7cpzx.mongodb.net/test
```

Mrs. KONATE
(web programming
Teacher)

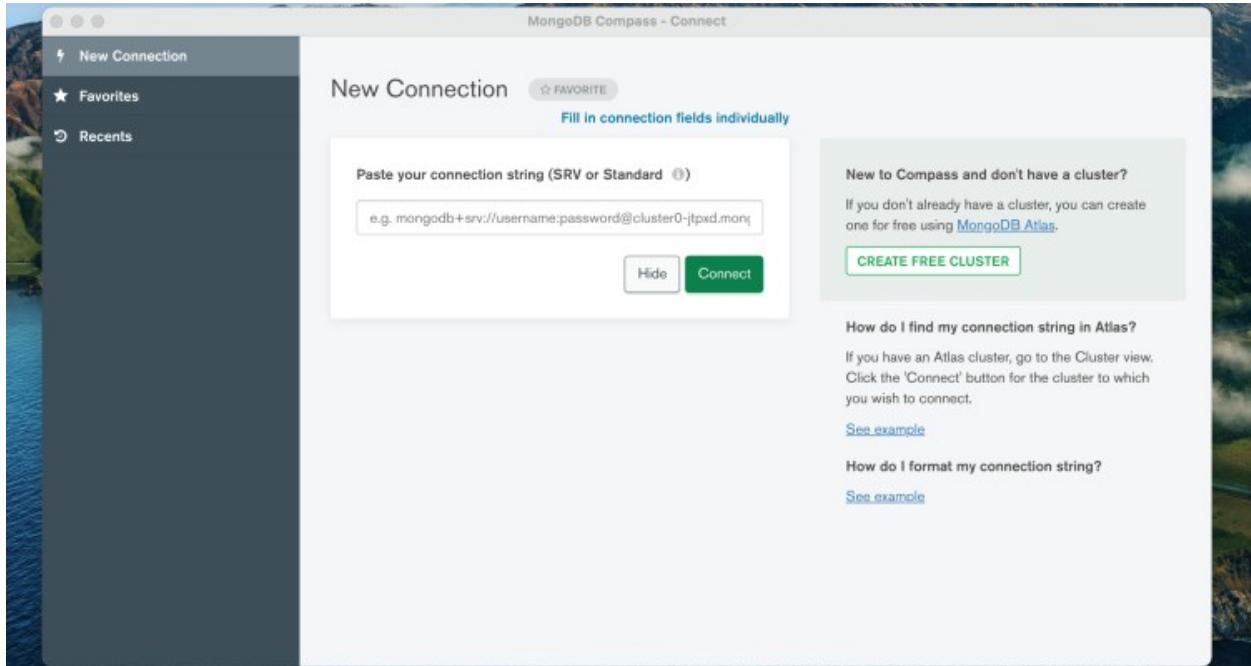
Download and Install MongoDB Compass

Maintenant, téléchargeons et installons MongoDB Compass sur notre système. Je fournis l'URL du site officiel pour télécharger la boussole MongoDB.

<https://www.mongodb.com/try/download/compass>

The screenshot shows the MongoDB download page. At the top, there's a dropdown menu set to 'MongoDB Shell'. Below it, the 'MongoDB Compass' section is visible. It contains a brief description of Compass, information about its three versions (Full, Read-only, and Isolated), and a note about network connections. To the right, the 'Available Downloads' sidebar is open, showing the selected version (1.29.5 Stable), platform (OS X 64-bit), and package type (dmg). A large green 'Download' button is prominently displayed, along with a 'Copy Link' option.

Après l'avoir installé sur notre système, nous pouvons voir une fenêtre ci-dessous. Ici, nous pouvons coller la chaîne de connexion que nous avons copiée à l'étape précédente.



Mrs. KONATE
(web programming
Teacher)

Configurer une nouvelle connexion avec la chaîne de connexion

Créons maintenant une nouvelle connexion avec la chaîne de connexion que nous avons déjà copiée à l'étape précédente.

Collez la chaîne de connexion dans le champ de saisie de MongoDB Compass.

Avant de coller la chaîne de connexion, nous devons remplacer le nom d'utilisateur et le mot de passe par celui que nous avons créé auparavant.

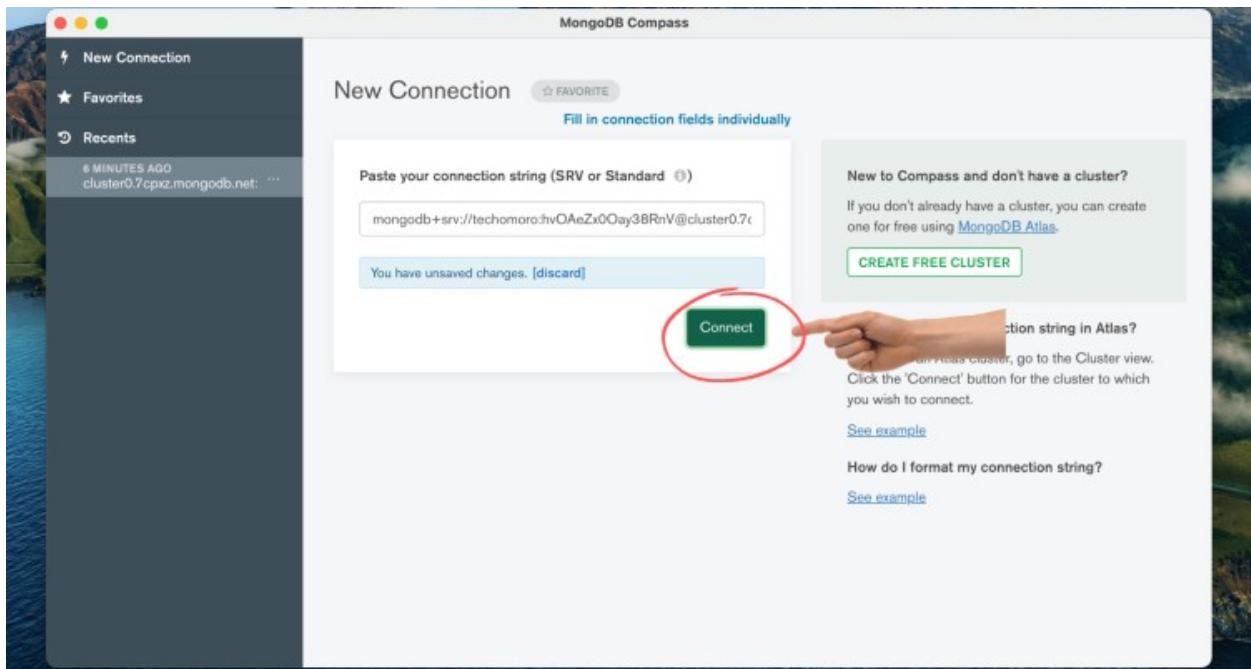
Dans notre cas, le nom d'utilisateur est techomoro et le mot de passe est une chaîne aléatoire. Ainsi, la chaîne de connexion sera la même que ci-dessous.

votre nom de famille : techomoro

mot de passe : uWfMzw2oilahWIS5

mongodb+srv://techomoro :

uWfMzw2oilahWIS5@cluster0.7cpxz.mongodb.net/test



Après avoir créé une connexion, la fenêtre sera la même que ci-dessous. Il se connectera au test de la base de données et nous pourrons voir la collection d'utilisateurs à l'intérieur. C'est une collection factice que j'ai faite à des fins de test.

The screenshot shows the MongoDB Compass interface. On the left, the sidebar displays the 'Local' connection with 7 DBs and 8 Collections. The 'test' database is selected, showing its collections: 'users', 'posts', 'nextjs-mongodb-atlas-demo', 'nextjs-api-demo', 'local', 'config', 'admin', and '_MONGOSH'. The main panel is titled 'Collections' and shows a table for the 'users' collection. The table has columns: Collection Name, Documents, Avg. Document Size, Total Document Size, Num. Indexes, Total Index Size, and Properties. One row is present for the 'users' collection, showing 1 document, an average size of 69.0 B, a total size of 69.0 B, 1 index, and a total index size of 20.5 KB.

Collection Name	Documents	Avg. Document Size	Total Document Size	Num. Indexes	Total Index Size	Properties
users	1	69.0 B	69.0 B	1	20.5 KB	

Mrs. KONATE
(web
programming Teacher)

ABOUT THE PROJECT

**CREER UN FORMULAIRE PERMETTANT
D'EFFECTUER DES OPERATIONS DE
CRUD (Create, Read, Update and
Delete)**

Source Code

#Readme.md

```
# python-mongodb-CRUD-example
```

```
Python MongoDB CRUD example
```

Installation

- Clone the project using git clone
- Move to the project using cd command
- \$ docker-compose up --build
- Web access >> 0.0.0.0:5000

#app.py

```
from flask import Flask, render_template, request,  
redirect, url_for  
import pymongo  
from bson.objectid import ObjectId
```

```
app = Flask(__name__)
client = pymongo.MongoClient("mongodb://mongo:27017")
db = client["mydb"]
fnf_coll = db["fnf"]

@app.route("/")
def home():
    return render_template('index.html',
members=fnf_coll.find())

@app.route("/fnf/create")
def create():
    return render_template('create.html')

@app.route("/fnf/save", methods=['POST'])
def save():
    name = request.form['name']
    relation = request.form['relation']
    phone = request.form['phone']
    email = request.form['email']

    data = {"name": name, "relation": relation, "phone": phone, "email": email}
```

```
fnf_coll.insert_one(data)
return redirect(url_for('home'))


@app.route("/fnf/edit/<string:id>")
def edit(id):
    result = fnf_coll.find_one({"_id": ObjectId(id)})
    return render_template('edit.html', member=result)


@app.route("/fnf/update", methods=['POST'])
def update():
    id = request.form['id']
    name = request.form['name']
    relation = request.form['relation']
    phone = request.form['phone']
    email = request.form['email']

    data = {"name": name, "relation": relation, "phone": phone, "email": email}

    fnf_coll.update_one({'_id': ObjectId(id)}, {"$set": data}, upsert=False)
    return redirect(url_for('home'))


@app.route("/fnf/delete/<string:id>", methods=['GET', 'DELETE'])
def delete(id):
```

```
def delete(id):
    if request.method == 'DELETE':
        if request.json:
            params = request.json
        else:
            params = request.form

        id = params.get('id')

    fnf_coll.remove({"_id": ObjectId(id)})
    return redirect(url_for('home'))

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=5000, debug=True)
```

#requirement.txt

```
click==6.7
Flask==1.0.2
Flask-PyMongo==0.5.2
itsdangerous==0.24
Jinja2==2.10
MarkupSafe==1.0
pymongo==3.6.1
Werkzeug==0.14.1
```

DockerFile

```
# Use official Python
FROM python:3.6

# Set working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at
# /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r
requirements.txt

# Make port 80 available to the world outside this
# container
EXPOSE 5000

# Define environment variable
ENV NAME flask_mongo

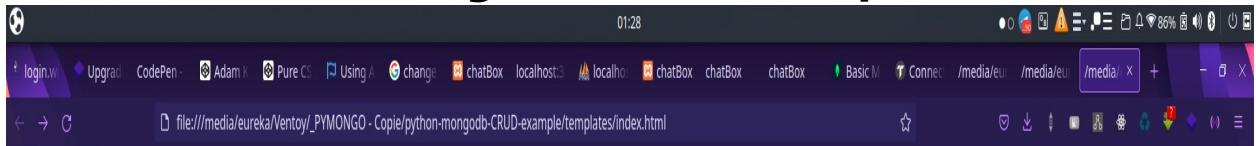
CMD ["python", "app.py"]
```

```
# docker-compose.yml

version: '3'
services:
  app:
    build: .
    volumes:
      - .:/app/
    ports:
      - "5000:5000"
    depends_on:
      - mongo
    links:
      - mongo
  mongo:
    image: mongo:latest
    container_name: "mongo"
    environment:
      - MONGO_DATA_DIR=/data/db
      - MONGO_LOG_DIR=/dev/null
    volumes:
      - ./data/db:/data/db
    ports:
      - 27017:27017
```

Screenshots of the program

Source Code of Program Read & Update & Delete



My Friends and Family Information

My Friends and Family Information				
Create New				
{% if members %}{% for member in members %}{% endfor %}{% endif %}				
Name	Relation	Phone	Email	Action
{% member.name %}	{% member.relation %}	{% member.phone %}	{% member.email %}	Edit Delete

Figure 1: Source Code of Program Create

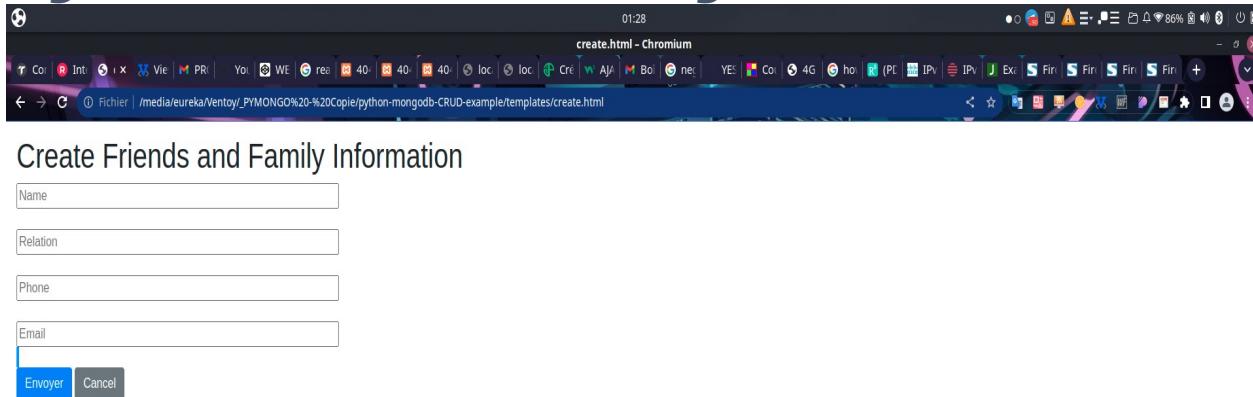
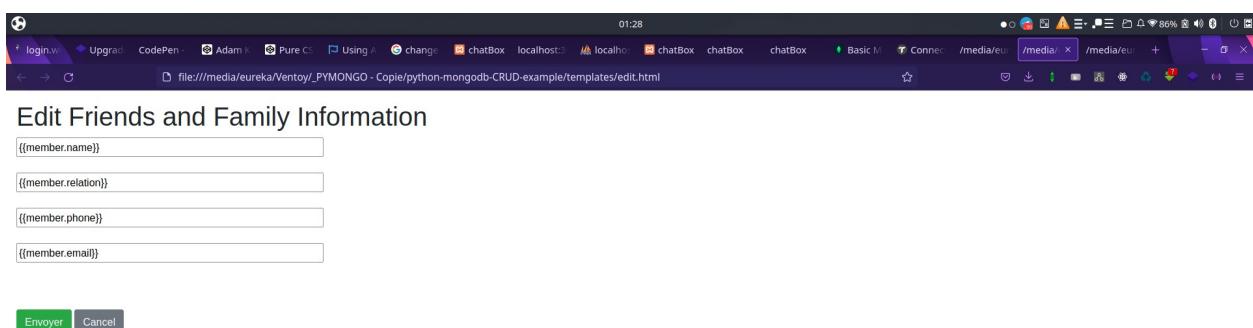


Figure 2: Source Code of Program Update



System Build

Operating System :	Windows 10 Pro
Version :	20H2
OS Build :	19041.572
Processor :	Intel® Core™ i5 vPro
@2.30GHz	
RAM :	8.00 GB
GPU :	Intel® HD Graphics 620
Python Version :	3.8.6
MongoDB Version :	x.x
Text Editor Used :	Sublime Text 3.2.2 Build 3211

Mrs. KONATE
(web programming Teacher)

Bibliography

- 1. YouTube**
- 2. Codemy.com YouTube channel**
- 3. Stack overflow**
- 4. GitHub**
- 5. www.w3schools.com**
- 6. Edureka live YouTube channel**

Mrs. KONATE
(web programming
Teacher)