

CIENCIA DE DATOS CON PYTHON: RECOLECCIÓN, ALMACENAMIENTO Y PROCESO



Fuente: adobestock/175199100



Autor:

Amaury Giovanni Méndez Aguirre

ÍNDICE

Introducción.....	3
Bases de datos no relacionales NoSQL con MongoDB	4
Crear un proyecto en MongoDB	6
Crear una base de Datos en MongoDB.....	13
Conclusiones	17
Introducción.....	18
Códigos en Python para MongoDB.....	19
Agregar datos a una base de Datos en MongoDB	20
Consultas simples	23
Conclusiones	26
Procesos - pasos	27
Bibliografía.....	30

INTRODUCCIÓN

Las bases de datos son un concepto esencial en la computación moderna y su desarrollo tanto industrial como científico, porque permiten no solo almacenar los datos, sino poder compartirlos, realizar consultas, actualizarlos o eliminarlos de ser necesario, en cualquier situación. Piense en una entidad financiera que debe mantener actualizadas las transacciones bancarias de sus clientes; existe un modelo particular para tratar este tema, denominado Bases de Datos Relacionales, o bases de datos SQL, donde los datos deben ser uniformes, donde los datos son almacenados en tablas y éstas deben ser definidas antes de empezar a almacenar los datos. Pero también piense en datos que no son uniformes y que se pueden estar actualizando todo el tiempo, como los datos de las redes sociales o datos de los sensores en una industria, en donde intentar guardar una relación por medio de tablas ya no es una opción frente al rendimiento de las consultas para acceder a éstos. Es aquí donde entran las bases de datos no relacionales conocidas como NoSQL.



Bases de datos no relacionales NoSQL con MongoDB



Estas bases de datos están orientadas a ser documentos y lo que se podría interpretar como una fila en una tabla de datos relacionales, en las bases de datos NoSQL son documentos. Una fila es un documento, usualmente en formato JSON (Caballero, 2019).

La característica fundamental en una base de datos NoSQL es que no todos los documentos de una colección tienen que tener la misma estructura, que es lo que sucede hoy en día con los datos de las redes sociales y la ciencia de datos. Como ejemplo, recuerde que datos desde un archivo CSV o EXCEL tienen en común su estructura de tabla donde cada fila debe tener los mismos datos por sus columnas. Pero en NoSQL esto ya no es una restricción.

En MongoDB, las colecciones son como las tablas, los documentos en formato JSON son como las filas, las columnas son las claves del formato JSON y los valores son los datos en sí, los contenidos de una celda en el caso de las tablas.



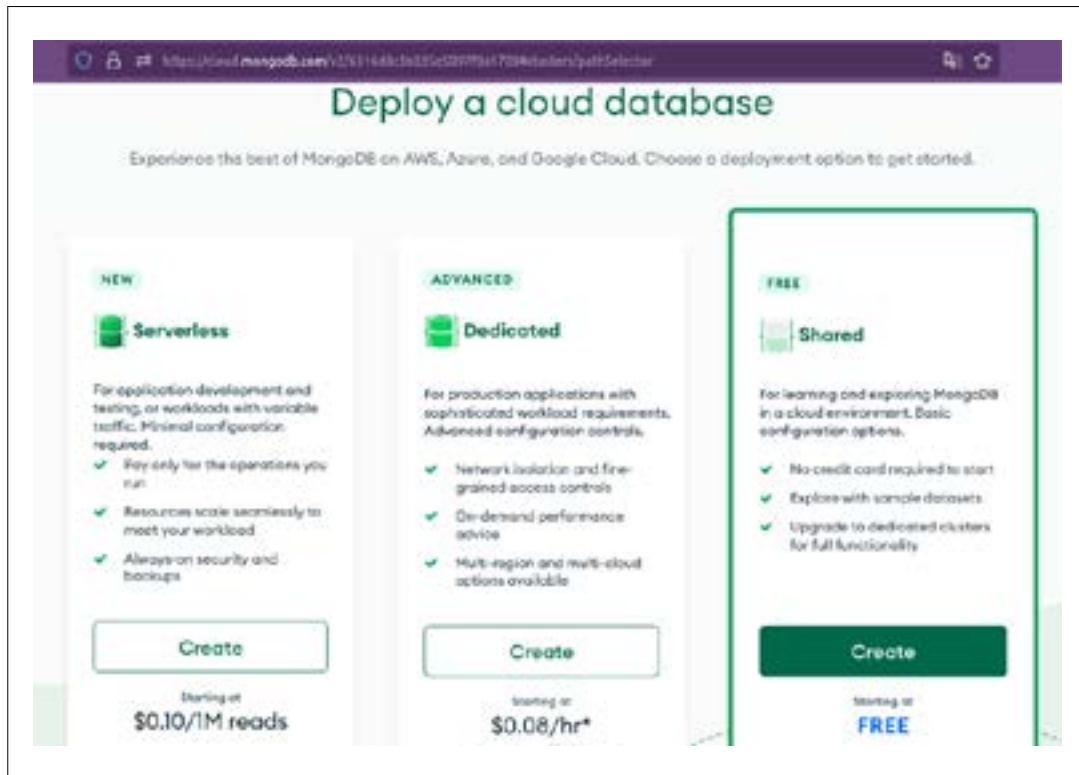
Visitar página

Para tener acceso a un sistema gestor de bases de datos SGBD como MongoDB vamos a utilizar el SGBD Atlas. Como primer paso, se debe hacer el registro de forma gratuita desde la siguiente url:

<https://www.mongodb.com/cloud/atlas/register>

Una vez finalizado el proceso de registro y loguearse por primera vez, aparecerán las siguientes opciones:

Figura 1



Fuente: propia

Observa que se puede seleccionar entre tres servicios: **Serverless**, **Dedicated** y **Shared**. Por ahora, seleccionaremos el servicio **Shared**

Luego, iniciaremos a crear nuestro primer proyecto.

Crear un proyecto en MongoDB

Una vez que tengamos nuestra cuenta creada en MongoDB, vamos a crear nuestro primer proyecto, donde alojaremos nuestra base de datos. Este paso creará un **cluster** para almacenar nuestras bases de datos



Serverless:

Servicio de la base de datos para el desarrollo de aplicaciones y pruebas de tráfico pesadas

Dedicated:

Servicio de la base de datos para aplicaciones reales que estarán en producción

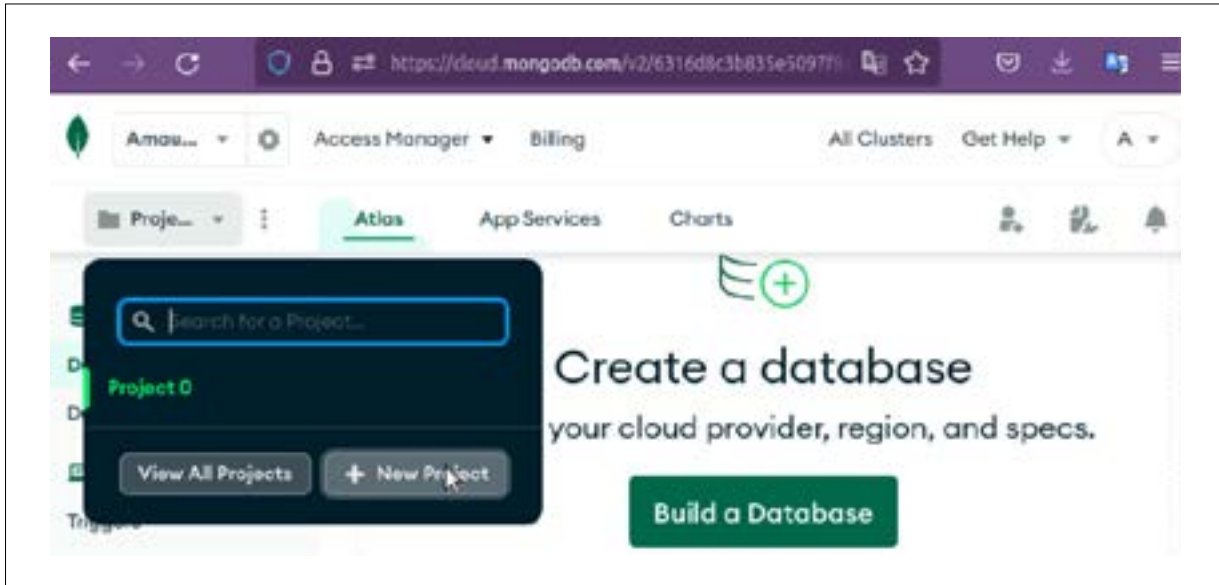
Shared:

Servicio de la base de datos para el aprendizaje de su sistema gestor en un entorno cerrado

Cluster:

Es un conjunto dedicado de servidores para distribuir las capacidades de nuestras bases de datos, permitiéndonos la conexión remota a ésta

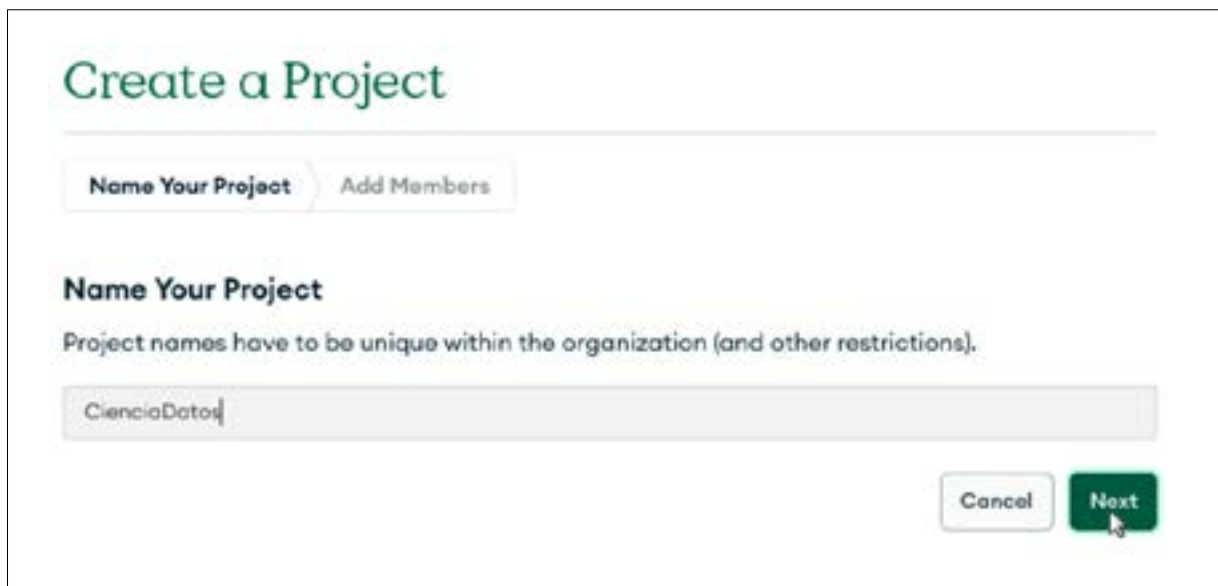
Figura 2



Fuente: propia

Así que damos clic en **New Project**

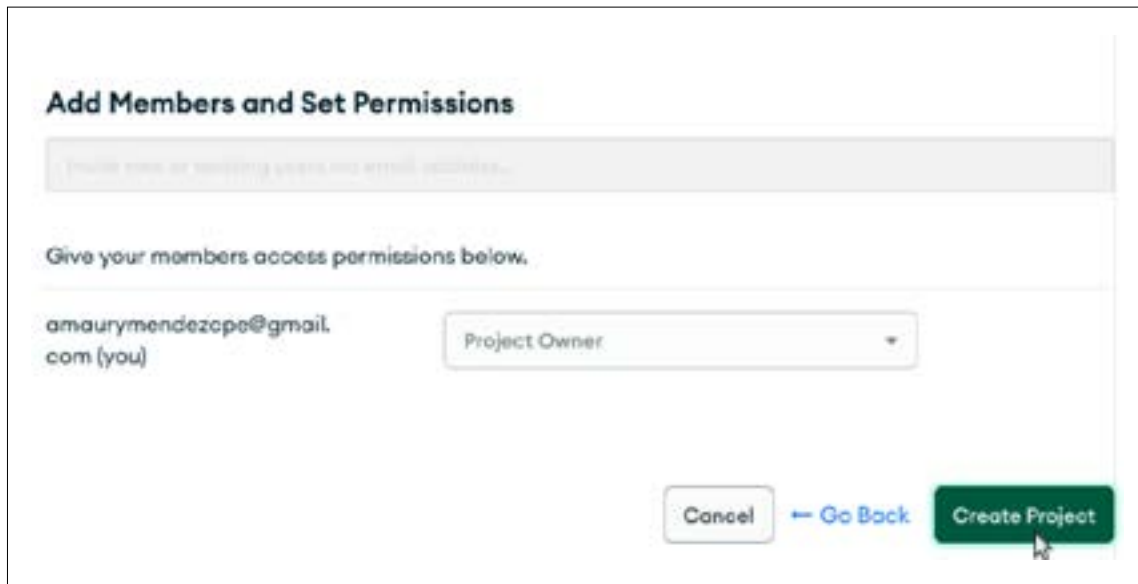
Figura 3



Fuente: propia

Daremos un nombre a nuestro proyecto, por ejemplo: **CienciaDatos**

Figura 4

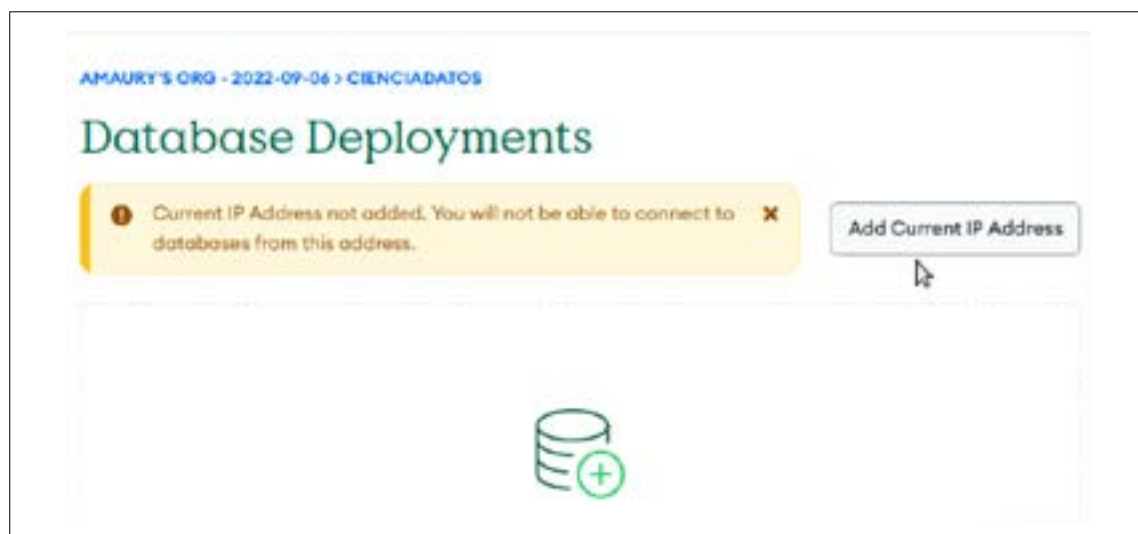


The screenshot shows a web interface titled "Add Members and Set Permissions". At the top, there is a search bar with the placeholder text "Type email or existing group and email addresses...". Below this, a message states "Give your members access permissions below." A table lists members, with one entry showing the email "amaurymendezcpe@gmail.com (you)" and a dropdown menu set to "Project Owner". At the bottom right, there are three buttons: "Cancel", "Go Back" (with a left arrow), and "Create Project" (in green). A mouse cursor is pointing at the "Create Project" button.

Fuente: propia

En cada paso a continuación, simplemente iremos avanzando y creando lo necesario para tener nuestra propia base de datos en la nube. Aquí, podremos agregar más usuarios a nuestro proyecto, pero por ahora, dejaremos solo el acceso a nuestro propio usuario, así que daremos clic en **Create Project**.

Figura 5

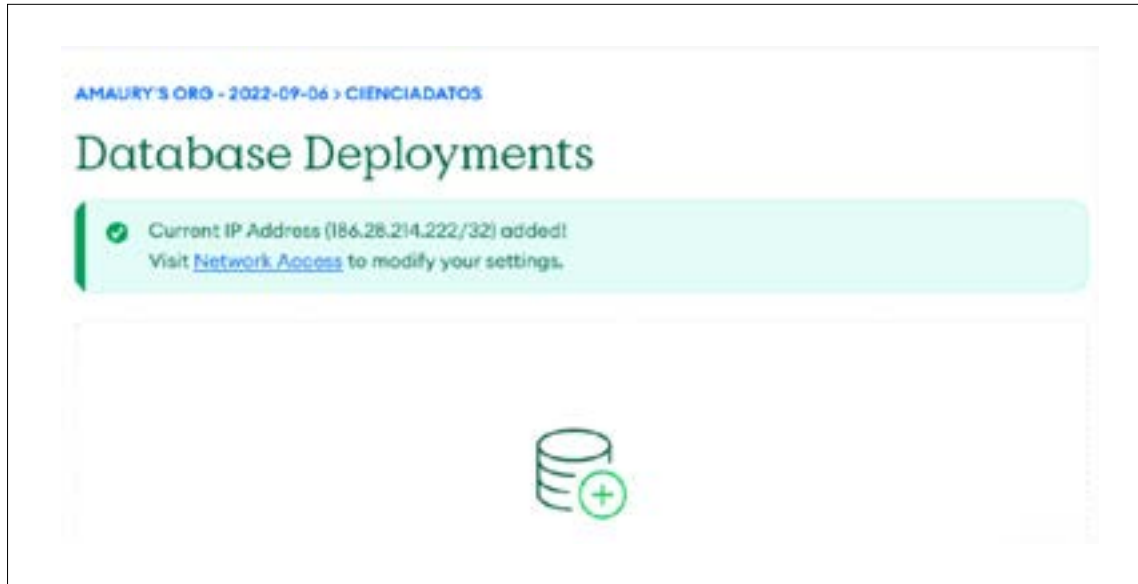


The screenshot shows a web interface titled "Database Deployments". At the top, there is a breadcrumb trail: "AMAURY'S ORG - 2022-09-06 > CIENCIA DATOS". Below this, a large green heading reads "Database Deployments". A yellow warning box contains the message: "Current IP Address not added. You will not be able to connect to databases from this address." To the right of the warning box is a button labeled "Add Current IP Address". Below the warning box, there is a large empty rectangular area. At the bottom center of this area is a green icon of a database cylinder with a plus sign. A mouse cursor is pointing at the "Add Current IP Address" button.

Fuente: propia

Nuestro proyecto necesita de algunos permisos especiales, así que damos clic en **Add Current IP Address**

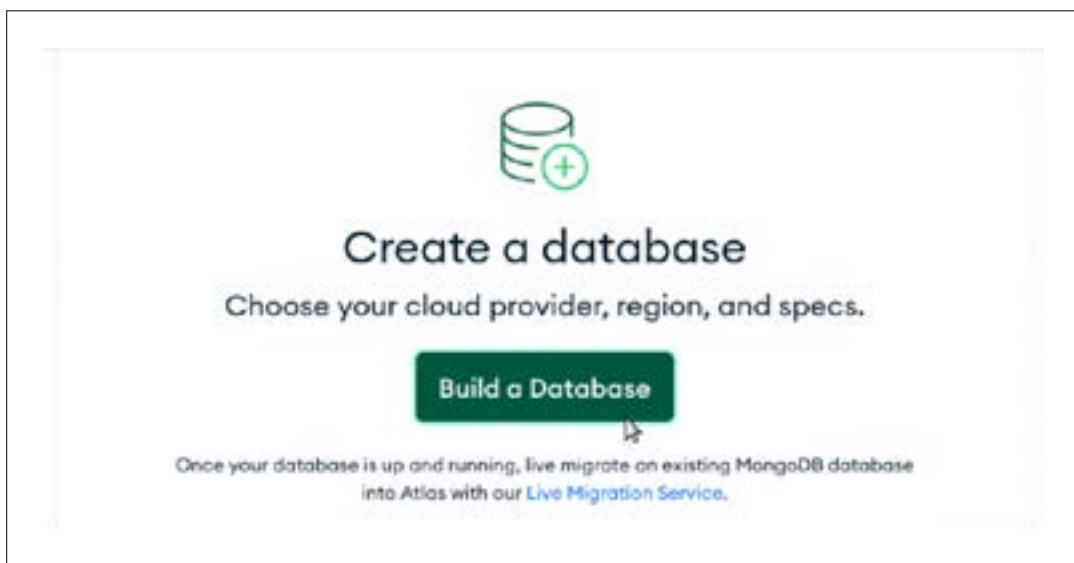
Figura 6



Fuente: propia

¡Ya podremos conectarnos a nuestra propia base de datos!

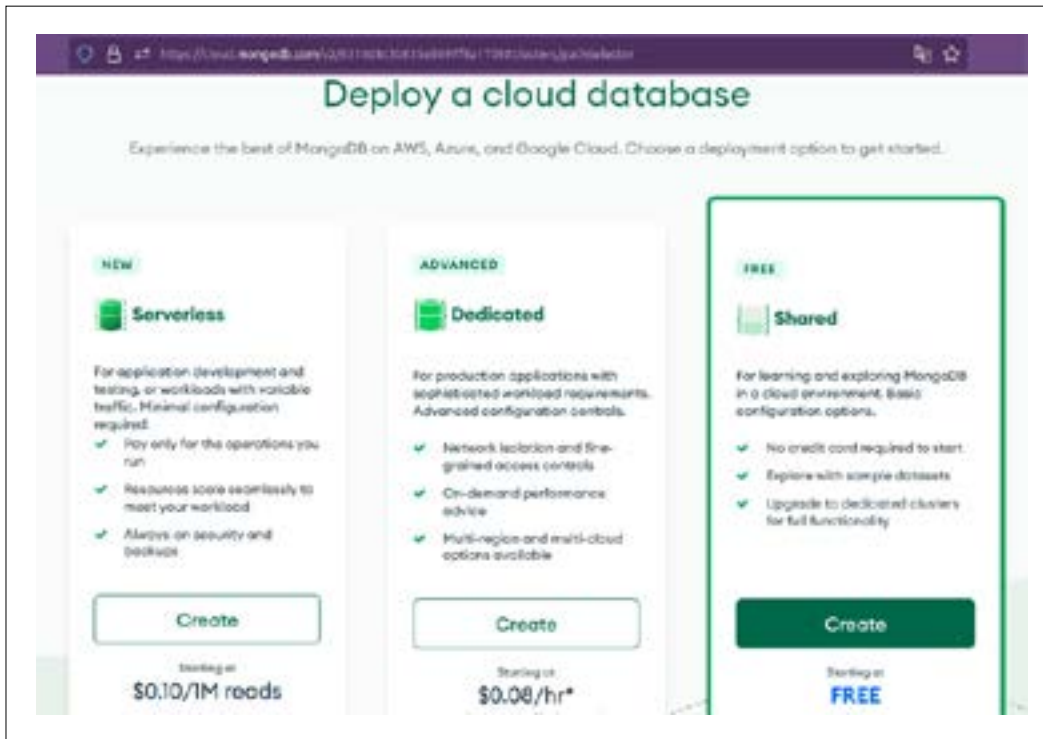
Figura 7



Fuente: propia

Ahora podremos crear nuestra base de datos sin ningún problema

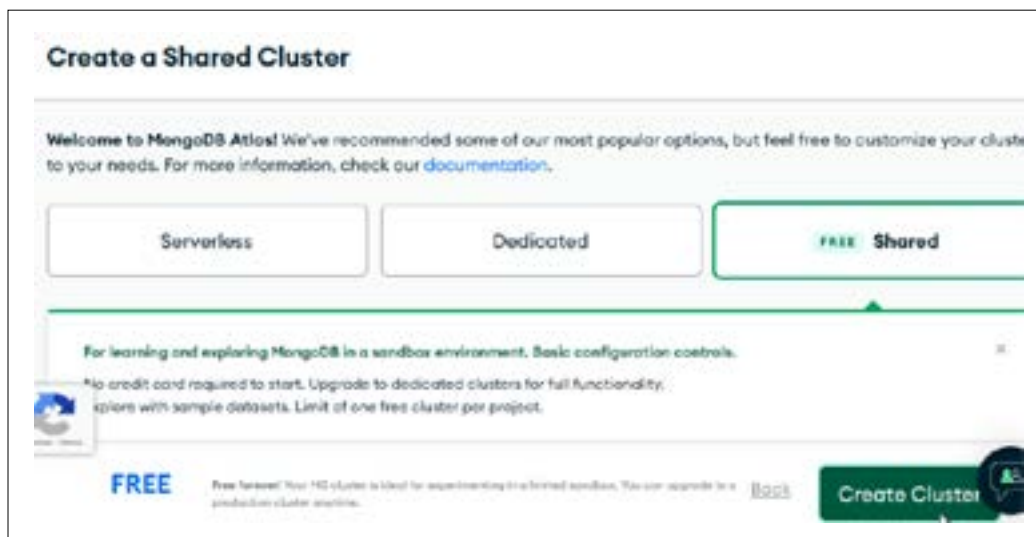
Figura 8



Fuente: propia

Por ahora, siempre que se nos pida esta información, simplemente seleccionaremos la opción **FREE**

Figura 9



Fuente: propia

Clic en Create **Cluster**.

Figura 10

1 How would you like to authenticate your connection?

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

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.

Username

ciencia

Password

Autogenerate Secure Password

Fuente: propia

Creamos nuestro usuario y contraseña con la cual accederemos a nuestra base de datos. En este ejercicio el usuario es la palabra **ciencia**, y la contraseña es la palabra **datos**

Figura 11

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.

Username

ciencia

Password

Autogenerate Secure Password

Copy

Create User

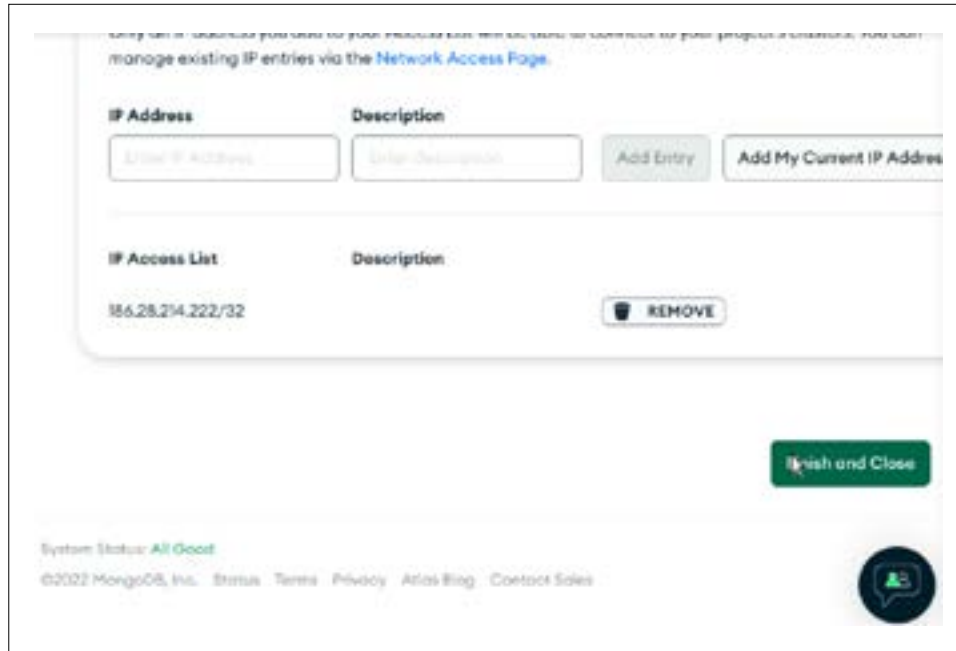
Where would you like to connect from?

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

Fuente: propia

Clic en **Create User**

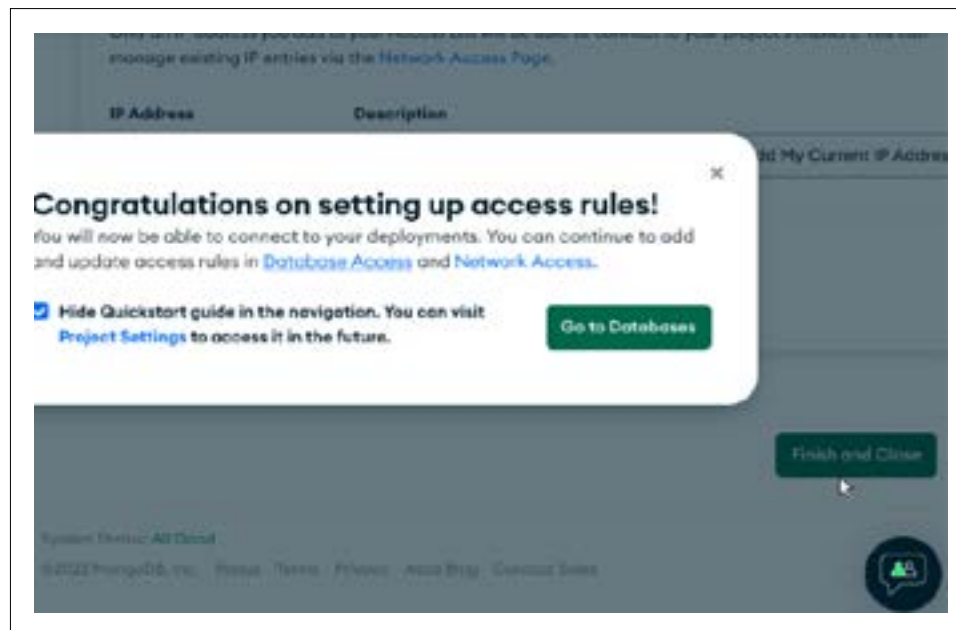
Figura 12



Fuente: propia

Clic en **Finish and Close.**

Figura 13



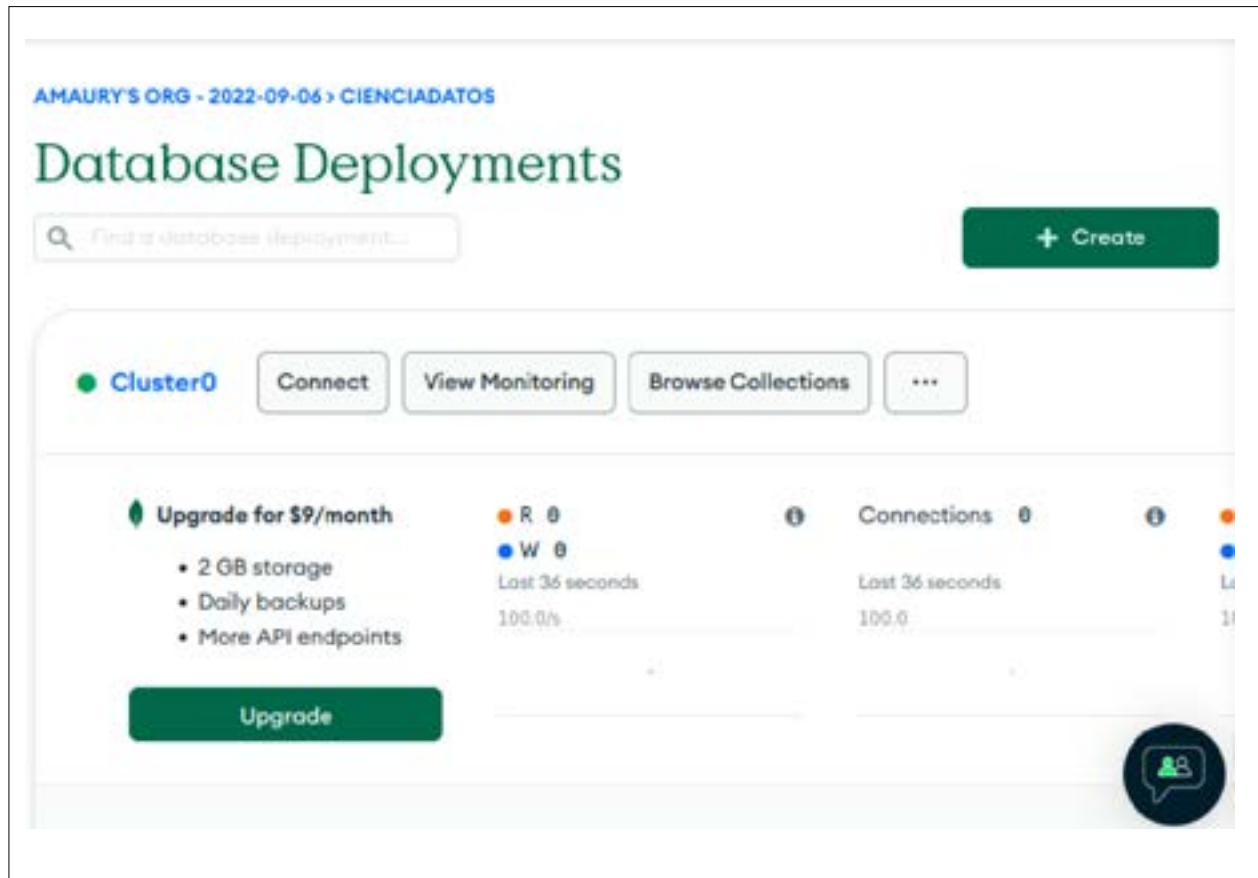
Fuente: propia

Ahora podremos dar clic en el botón **Go to Databases** para ir definiendo cómo acceder a nuestra base de datos por medio de Python

Crear una base de Datos en MongoDB

Cuando ya tenemos un proyecto en MongoDB, un usuario y una contraseña para acceder al proyecto, veremos algo similar a la imagen presentada a continuación y será momento de definir el nombre de la base de datos y el nombre de la colección o colecciones que tendrá nuestra base de datos

Figura 14

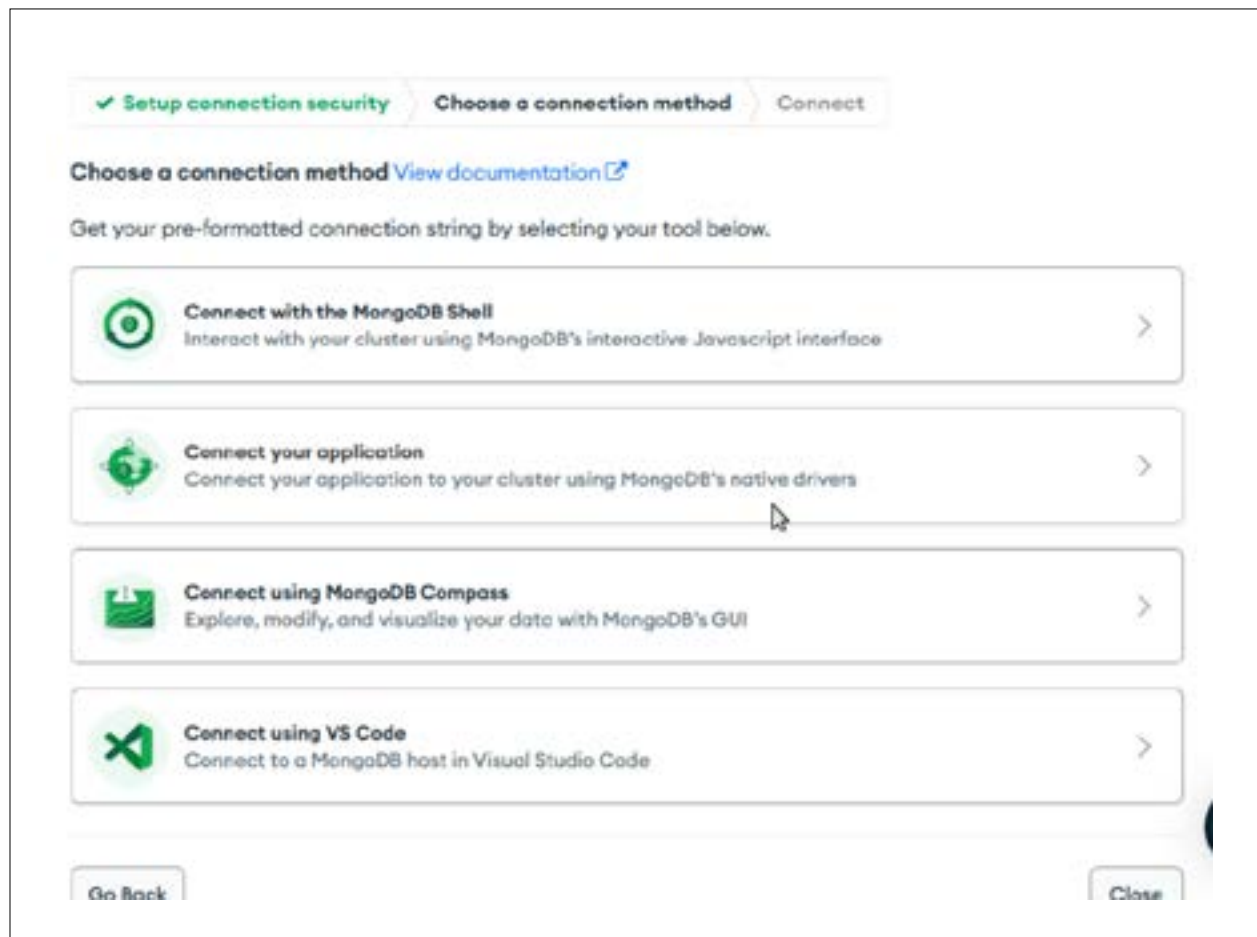


Fuente: propia

Aquí, vamos a dar clic en el botón **Connect**, el cual nos mostrará nuestro código para acceder a nuestra base de datos

```
    # Determine which of the main  
20 boardRow = int(block / 3);  
21 good = True  
22 for b in range(boardRow * 3, (b + 3) * 3):  
23     if b != block:  
24         if num in board[b][row]:  
25             good = False
```

Figura 15



Fuente: propia

Nuestro código se encuentra en la opción **Connect your application**

Figura 16

Connect to Cluster0

✓ Setup connection security ✓ Choose a connection method Connect

1 Select your driver and version

DRIVER	VERSION
Python	3.6 or later

2 Add your connection string into your application code

☒ Include full driver code example

```
client = pymongo.MongoClient("mongodb+srv://ciencia:
<password>@cluster0.ijw85oy.mongodb.net/?retryWrites=true&w=majority")
db = client.test
```

Replace <password> with the password for the ciencia user. Ensure any option params are URL encoded.

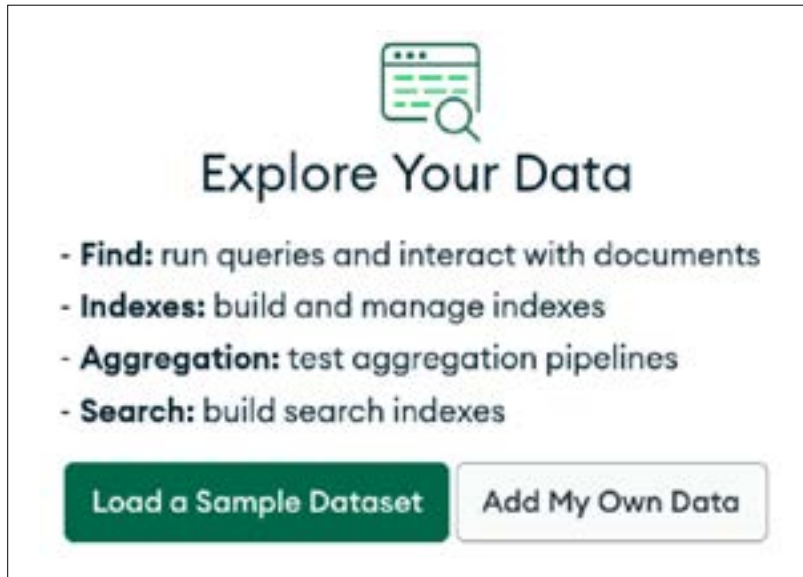
Having trouble connecting? [View our troubleshooting documentation](#)

Fuente: propia

Al copiar nuestro código, nos aseguramos que en la sección 1 donde dice **Select your driver and version**, esté seleccionada la opción **Python** y por ejemplo, la **versión 3.6 or later**

Volviendo a la imagen de la figura 14, si damos clic en el botón **Browse Collections**, nos llevará a la opción **Collections** para dar un nombre a nuestra base de datos, y un nombre a nuestra primera colección de datos

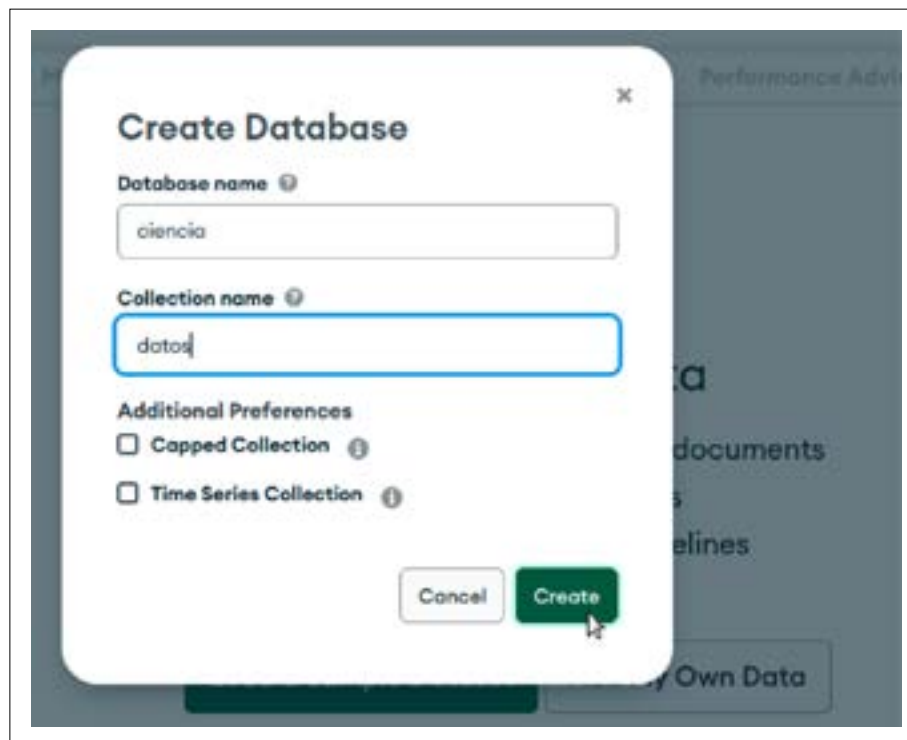
Figura 17



Fuente: propia

Daremos clic en el botón **Add My Own Data** y pondremos como nombre la palabra **ciencia** y para la colección, el nombre **datos**.

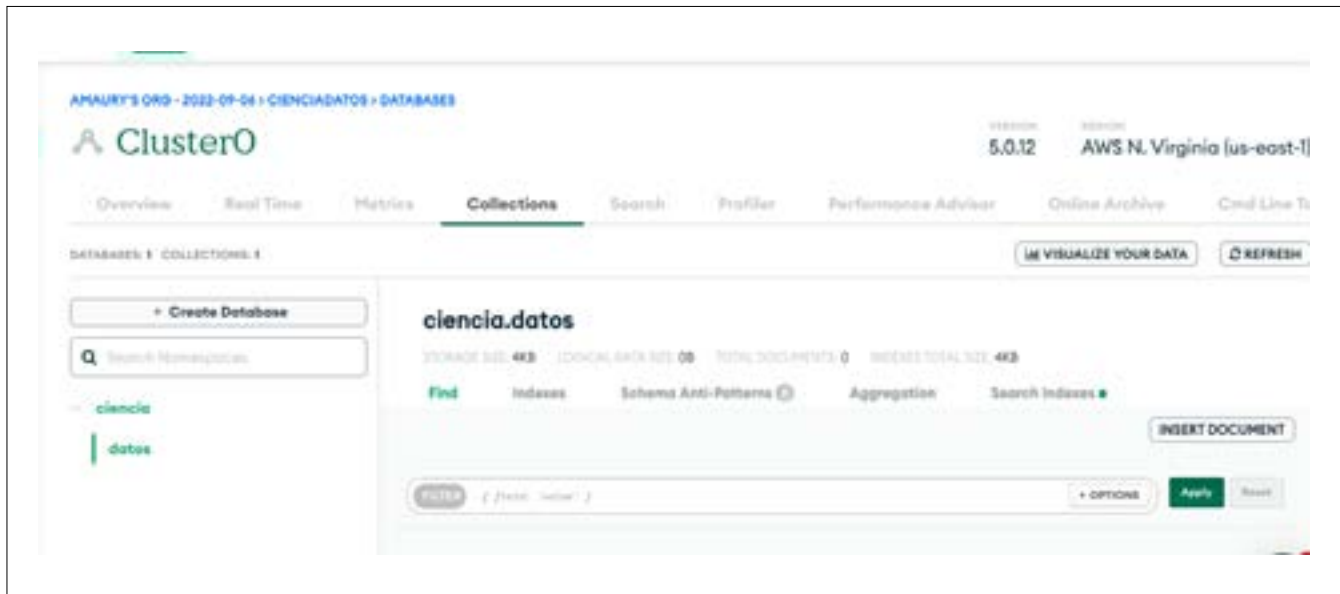
Figura 18



Fuente: propia

Da clic en **Create**

Figura 19



Fuente: propia

Ahora tendrás algo similar en tu pantalla. Es hora de tener datos y simplemente usaremos el siguiente código para cargar datos y posteriormente recuperarlos usando Python

Conclusiones

Aunque también se pueden crear las bases de datos con código, se recomienda hacerlo desde el sitio web por facilidad, crear el usuario o usuarios que tendrán acceso a ésta y sus privilegios. Inicialmente bastará con utilizar una base de datos compartida (shared FREE), pero si requiere implementarlo en un proyecto real, debe inclinarse por usar un servicio dedicado (dedicated) que es el aconsejado para producción

INTRODUCCIÓN

Se debe diferenciar en este proceso que crear una base de datos y hacer operaciones con ellas son dos cosas aparte. Para la primera no es necesario desarrollar códigos, pero para la segunda, lo conveniente es implementar códigos de Python de bibliotecas específicas como pymongo.



Códigos en Python para MongoDB



Agregar datos a una base de Datos en MongoDB

El primer paso será instalar dos librerías especiales, **pymongo** y **dnspython**

```
!pip install pymongo dnspython
```



pymongo:

Biblioteca de Python con todos los módulos para realizar las conexiones a las bases de datos MongoDB

dnspython:

Biblioteca de Python que resuelve las conexiones de red con nombres de dominio

Figura 20

```
Módulo 2.ipynb
[1]: !pip install pymongo dnspython

Collecting pymongo
  Using cached pymongo-4.2.0-cp38-cp38-manylinux
Requirement already satisfied: dnspython in /usr
(1.16.0)
Installing collected packages: pymongo
Successfully installed pymongo-4.2.0
```

Fuente: propia

```
    # Determine which of the main  
20 boardRow = int(block / 3);  
21 good = True  
22 for b in range(boardRow * 3, (b  
23 if b != block:  
24 if num in board[b][row]:  
25 good = False
```

Ahora nuestro código para conectarnos a nuestra propia base de datos

```
from pymongo import MongoClient  
  
import json  
  
#creamos nuestra colección  
  
#revisa cambiar la palabra <password> por la contraseña  
datos  
  
client = MongoClient("mongodb+srv://ciencia:datos@cluster0.  
ijw85oy.mongodb.net/?retryWrites=true&w=majority")  
  
#conectar a la colección datos  
  
coleccion = client.ciencia.datos  
  
#cargar el archivo JSON  
  
with open("notas.json",encoding="utf-8") as archivo:  
  
    datosJson = json.load(archivo)  
  
#insertar los datos en la base de datos  
  
coleccion.insert_many(datosJson)
```

Figura 21

```
from pymongo import MongoClient
import json

#creamos nuestra colección
#revisa cambiar la palabra <password> por la contraseña datos
client = MongoClient("mongodb+srv://ciencia:datos@cluster0.ijw85oy.mongodb.net")

#conectar a la colección datos
coleccion = client.ciencia.datos

#cargar el archivo JSON
with open("notas.json", encoding="utf-8") as archivo:
    datosJson = json.load(archivo)

#insertar los datos en la base de datos
coleccion.insert_many(datosJson)
```

Fuente: propia

Al correr este código, recuerda cambiar la palabra **<password>** por la contraseña adecuada. Observa que al código original se le ha hecho un cambio al llamar la clase **.MongoClient()** suprimiendo la palabra **pymongo**, ya que en la primera línea estamos indicando que importamos directamente este método. Como los documentos son los registros o filas para nuestra base de datos, es importante que éstos estén en formato JSON o que estén en tipo de diccionario. Usaremos en este ejemplo el archivo **notas.json** para cargar los datos. Finalmente, con el método **.insert_many()** podremos cargar todos los datos del documento JSON



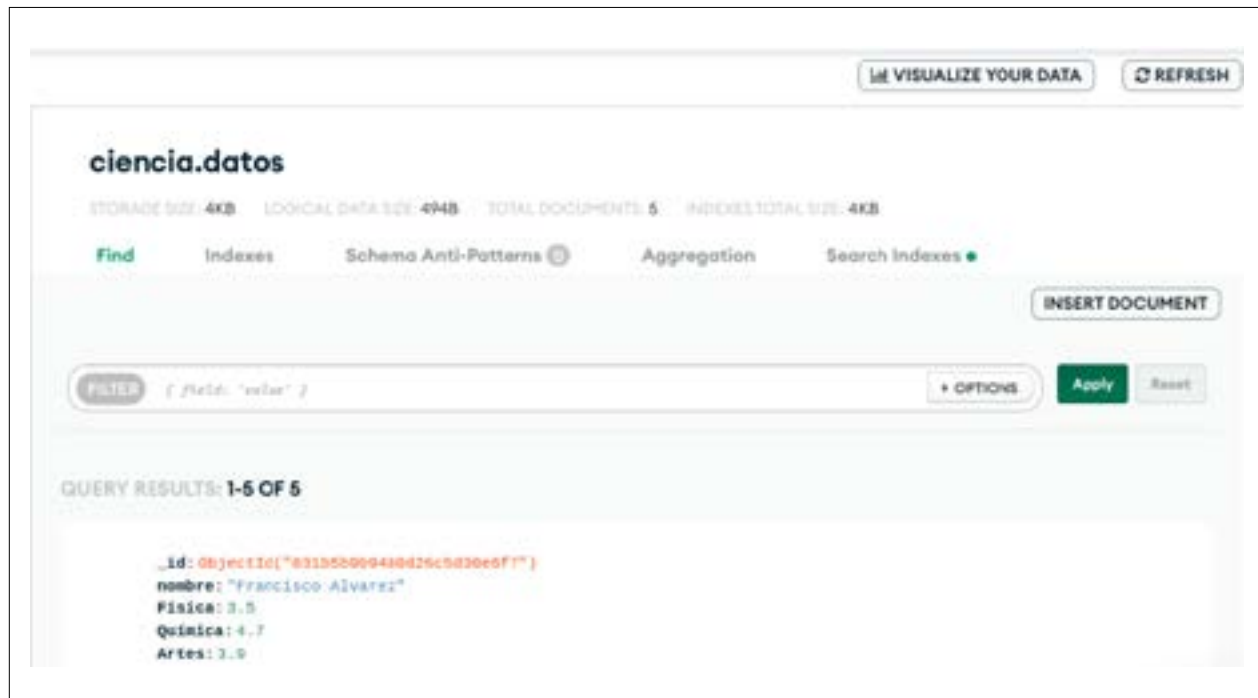
MongoClient:

Clase que permite la conexión a la base de datos MongoDB

insert_many:

Método de la clase MongoClient que permite la inserción de varios registros a la vez, cargados principalmente de objetos o documentos con formato JSON

Figura 22



Fuente: propia

Si vamos nuevamente a nuestra pantalla de Mongo Atlas, y damos clic en el botón **REFRESH**, observarás que los datos ya se encuentran en la nube

Consultas simples

Para realizar consultas, es importante primero que existan datos, luego de ésto, se puede mantener la conexión o simplemente iniciar una nueva, el código a continuación puede ser una forma de lograr nuevamente la conexión para recuperar los datos almacenados

```
from pymongo import MongoClient
import json

#creamos nuestra colección
#revisa cambiar la palabra <password> por la contraseña datos
client = MongoClient("mongodb+srv://ciencia:datos@cluster0.
ijw85oy.mongodb.net/?retryWrites=true&w=majority")

#conectar a la colección datos
coleccion = client.ciencia.datos

#recuperar todos los datos
datos = coleccion.find()

for d in datos:
    print(d)
```

Figura 23

```
from pymongo import MongoClient
import json

#creamos nuestra colección
#revisa cambiar la palabra <password> por la contraseña datos
client = MongoClient("mongodb+srv://ciencia:datos@cluster0.ijw85oy.mongodb.net/?retryWrites=true&w=majority")

#conectar a la colección datos
coleccion = client.ciencia.datos

#recuperar todos los datos
datos = coleccion.find()

for d in datos:
    print(d)
```

```
{'_id': ObjectId('631b5b9b9480d26c5d30e6f7'), 'nombre': 'Francisco Alvarez',
'Física': 3.5, 'Química': 4.7, 'Artes': 3.9}
{'_id': ObjectId('631b5b9b9480d26c5d30e6f8'), 'nombre': 'Juan Gonzalez', 'Fís
```

Fuente: propia



El método `.find()` será el encargado de traer toda la información a nuestro programa. Ahora, tendremos nuestros datos en un objeto y podremos realizar el filtrado por medio de sus claves. Si queremos recuperar los datos de cada documento por su clave Física, simplemente podremos hacer lo siguiente



find:

Método de la clase MongoClient que permite recuperar todos los datos de una base de datos y los carga como objeto pymongo. Pueden ser trabajados como diccionarios

Figura 24

```
#conectar a la colección datos
coleccion = client.ciencia.datos

#recuperar todos los datos
datos = coleccion.find()

for d in datos:
    print(d["Física"])

3.5
4.1
3.2
4.5
3.9
```

Fuente: propia

De esta forma, ya tendremos los datos recuperados. Por tanto, conviene conocer con anterioridad la estructura de los datos almacenados, pues su recuperación se realizará por medio de las claves como si de un objeto tipo diccionario de Python se tratase

Conclusiones

Python ha simplificado en gran medida el acceso a las bases de datos y en este caso, a una base de datos NoSQL. Recuerde que la mejor manera es trabajar con los datos tipo diccionario o JSON para tener compatibilidad con la base de datos MongoDB. Métodos como `.insert_many()` o `.find()` de la biblioteca pymongo facilitan al programador la mayoría de tareas relacionadas con bases de datos. De esta manera, al recuperar los datos, podrá trabajarlos con índices de clave como sucede con los diccionarios para realizar las consultas que se requieran.



Lectura recomendada

Te invito a realizar las siguientes lecturas complementarias:

- ¿Qué es MongoDB?
MongoDB
- Welcome to the MongoDB Documentation
MongoDB



.....

Procesos - pasos

@

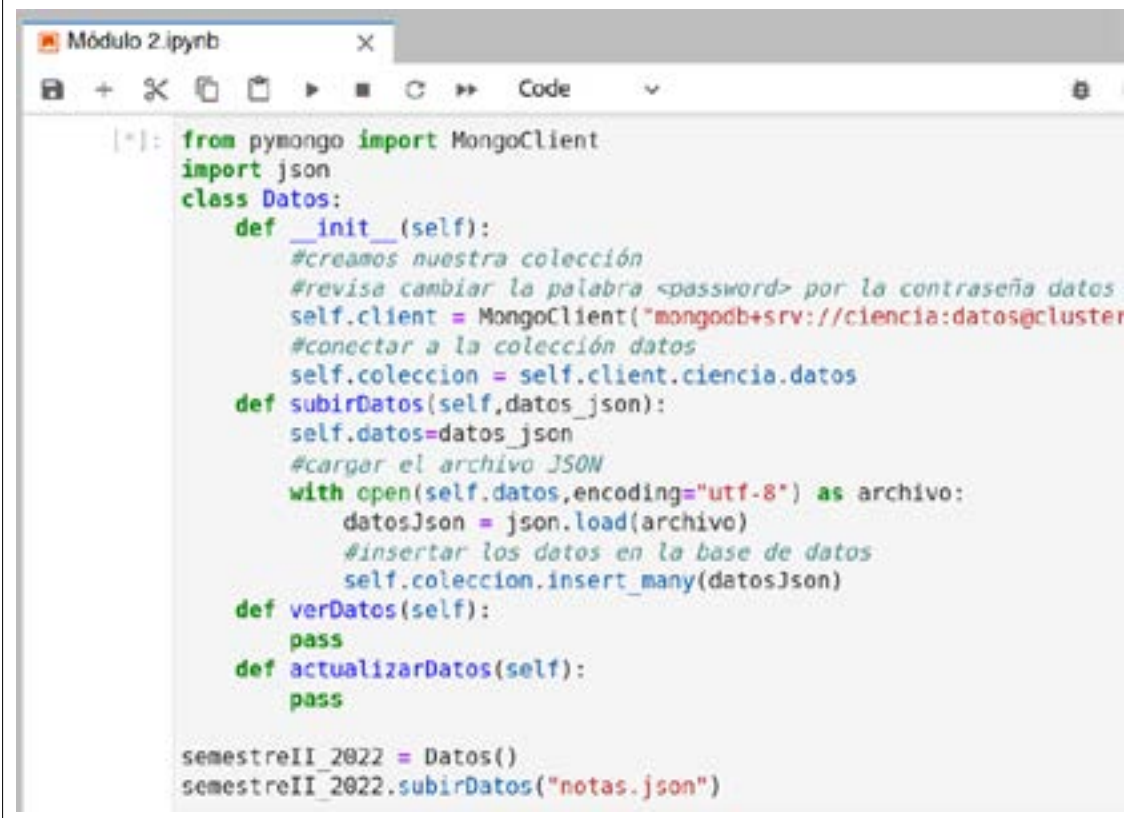


Ahora es momento de crear un programa que permita subir los datos sin inconvenientes y para ésto se usará la programación orientada a objetos. Puedes realizar sugerencias a Saray en el código que ella está desarrollando

```
from pymongo import MongoClient
import json
class Datos:
    def __init__(self):
        #creamos nuestra colección
        #revisa cambiar la palabra <password> por la contraseña
        datos

        self.client = MongoClient("mongodb+srv://ciencia:datos@cluster0.ijw85oy.mongodb.net/?retryWrites=true&w=majority")
        #conectar a la colección datos
        self.coleccion = self.client.ciencia.datos
    def subirDatos(self,datos_json):
        self.datos=datos_json
        #cargar el archivo JSON
        with open(self.datos,encoding="utf-8") as archivo:
            datosJson = json.load(archivo)
            #insertar los datos en la base de datos
            self.coleccion.insert_many(datosJson)
    def verDatos(self):
        pass
    def actualizarDatos(self):
        pass
semestreII_2022 = Datos()
semestreII_2022.subirDatos("notas.json")
```


Figura 1



```
[*]: from pymongo import MongoClient
import json
class Datos:
    def __init__(self):
        #creamos nuestra colección
        #revisa cambiar la palabra <password> por la contraseña datos
        self.client = MongoClient("mongodb+srv://ciencia:datos@cluster
        #conectar a la colección datos
        self.coleccion = self.client.ciencia.datos
    def subirDatos(self,datos_json):
        self.datos=datos_json
        #cargar el archivo JSON
        with open(self.datos,encoding="utf-8") as archivo:
            datosJson = json.load(archivo)
            #insertar los datos en la base de datos
            self.coleccion.insert_many(datosJson)
    def verDatos(self):
        pass
    def actualizarDatos(self):
        pass

semestreII_2022 = Datos()
semestreII_2022.subirDatos("notas.json")
```

¿Qué requisitos se deben tener en cuenta para que el código funcione completamente? Por ejemplo: la base de datos debe estar creada con anterioridad, los datos en los archivos json deben estar en algún orden específico, etc.

¿Qué código implementarías en el método **verDatos(self)**?

Saray ha pensado hacer algo como lo siguiente:

```
def verDatos(self):
    #recuperar todos los datos
    datos = self.coleccion.find()
    for d in datos:
        print(d)
```


BIBLIOGRAFÍA

García, J. (2018). Ciencia De Datos. Técnicas Analíticas Y Aprendizaje Estadístico. Un Enfoque Práctico - Jesús García.pdf [6ng22yvvd2lv]. Idoc. pub. Recuperado de <https://idoc.pub/documents/idocpub-6ng22yvvd2lv>.

Python Tutorial. W3schools.com. (2022). Retrieved 31 August 2022, from <https://www.w3schools.com/python/default.asp>.