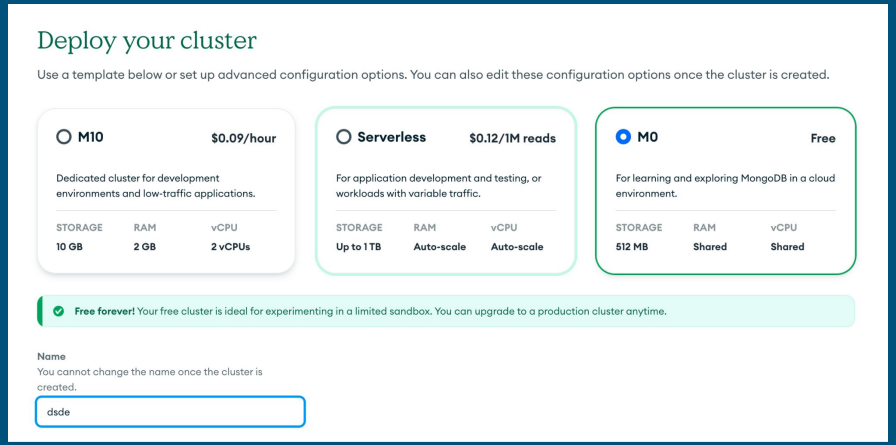
MongoDB

CRUD Operations tutorial
Colab [Link]
Credit: TA.Theerapat

0. Cloud Setup [Link]

1. Sign up
2. Create project
3. Create Cluster (Free tier)
4. Get a connection string
5. Configure Network access



0. Get the connection string

Connect to dsde

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 Python

Version 3.12 or later

2. Install your driver

Run the following on the command line

Note: Use appropriate Python 3 executable

```
python -m pip install "pymongo[srv]"
```

View MongoDB Python Driver installation instructions.

3. Add your connection string into your application code

dsde is provisioning...

3. Add your connection string into your application code

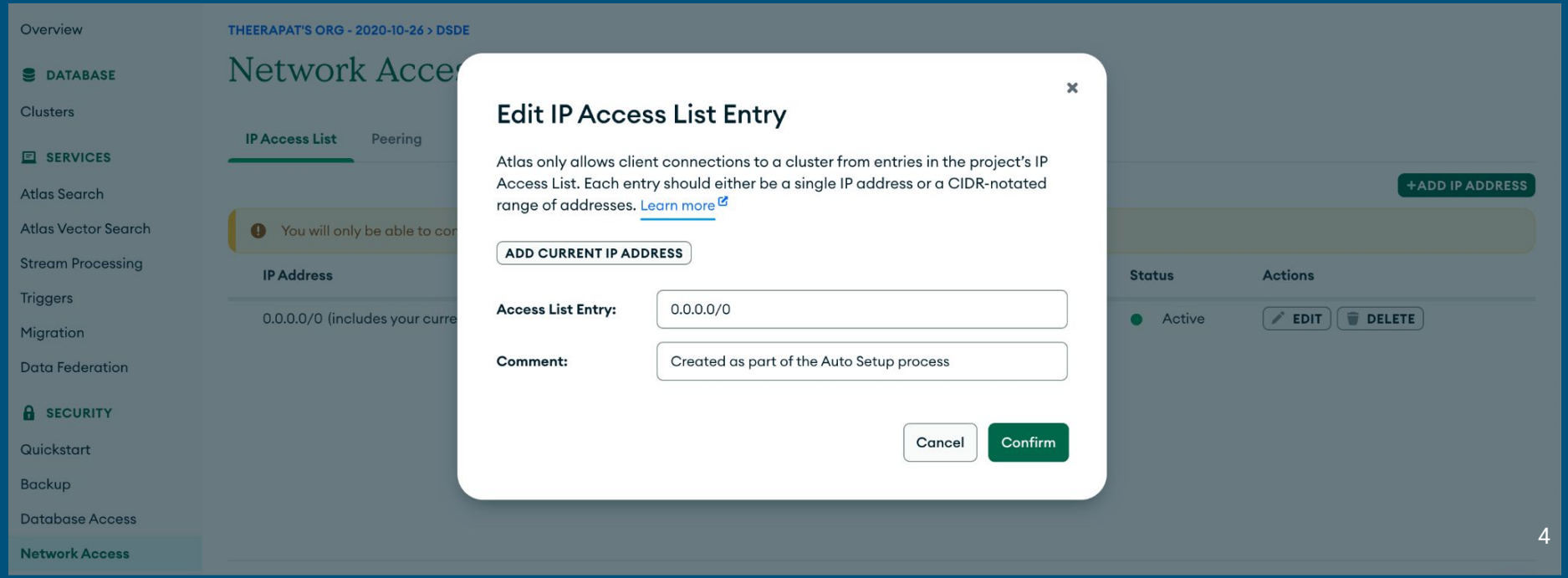
Use this connection string in your application

View full code sample — Show Password

```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi

uri = "mongodb+srv:/[redacted].mongodb.net/?retryWrites=t

# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi('1'))

# Send a ping to confirm a successful connection
try:
    client.admin.command('ping')
    print("Pinged your deployment. You successfully connected to MongoDB!")
except Exception as e:
    print(e)
```

0. Config Network Access

Config Access List Entry to 0.0.0.0/0 which will allow all ip address to connect [For **Colab**].



1. Install required library

```
!pip install 'pymongo[srv]'
```

2.1 Copy/Paste a client connection code

3. Add your connection string into your application code

Use this connection string in your application

View full code sample Show Password

```
from pymongo.mongo_client import MongoClient
from pymongo.server_api import ServerApi

uri = "mongodb+srv://[redacted].mongodb.net/?retryWrites=t

# Create a new client and connect to the server
client = MongoClient(uri, server_api=ServerApi('1'))

# Send a ping to confirm a successful connection
try:
    client.admin.command('ping')
    print("Pinged your deployment. You successfully connected to MongoDB!")
except Exception as e:
    print(e)
```

2.2 Create Books collection

```
db = client['dsde']
books_collection = db['books']
```

Python Library for mongodb field validating https://docs.pydantic.dev/1.10/

3. Create Book Class

```python
class Book(BaseModel):
    id: str = Field(default_factory=lambda: str(uuid.uuid4()), alias="_id")
    title: str = Field(...)
    author: str = Field(...)
    synopsis: str = Field(...)
```

```python
class BookUpdate(BaseModel):
    title: Optional[str] = None
    author: Optional[str] = None
    synopsis: Optional[str] = None
```

4. Create a Book

```python
def create_book(book: Book):
    book_data = book.dict(by_alias=True)
    result = books_collection.insert_one(book_data)
    return str(result.inserted_id)
```

```python
new_book = Book(title="Don Quixote",
                author="Miguel de Cervantes",
                synopsis="A novel about a man who reads too many chivalric romances...")
book_id = create_book(new_book)
```

You can check the result in the cloud.

Atlas | theerapat's ... | Access Manager | Billing | All Clusters | Get Help | thethee

dsde | Data Services | Charts

Overview

DATABASE

Clusters

SERVICES

Atlas Search

Atlas Vector Search

Stream Processing

Triggers

Migration

Data Federation

SECURITY

DATABASES: 1 COLLECTIONS: 1

VISUALIZE YOUR DATA | REFRESH

+ Create Database

Search Namespaces

dsde

books

dsde.books

STORAGE SIZE: 20KB LOGICAL DATA SIZE: 362B TOTAL DOCUMENTS: 2 INDEXES TOTAL SIZE: 20KB

Find | Indexes | Schema Anti-Patterns 0 | Aggregation | Search Indexes

INSERT DOCUMENT

Filter | Type a query: { field: 'value' } | Reset | Apply | Options

QUERY RESULTS: 1-1 OF 1

```
_id: "7ca404c1-6f88-4ad0-bdf1-d5b3e2ce2516"
title : "Don Quixote"
author : "Miguel de Cervantes"
synopsis : "A novel about a man who reads too many chivalric romances..."
```

5. Get a Book

```python
def get_book_by_id(book_id: str):
    book_data = books_collection.find_one({"_id": book_id})
    if book_data:
        return Book(**book_data)
    return None
```

```python
book = get_book_by_id(book_id)
print(book)
```

6. Update a Book

```python
def update_book(book_id: str, book_update: BookUpdate):
    update_data = {k: v for k, v in book_update.dict(
        exclude_unset=True).items() if v is not None}
    result = books_collection.update_one(
        {"_id": book_id}, {"$set": update_data})
    return result.modified_count
```

```python
book_update = BookUpdate(synopsis="Updated synopsis of Don Quixote")
updated_count = update_book(book_id, book_update)
print(f"Number of documents updated: {updated_count}")
```

```python
book = get_book_by_id(book_id)
print(book.synopsis)
```

7. Delete a Book

```python
def delete_book(book_id: str):
    result = books_collection.delete_one({"_id": book_id})
    return result.deleted_count
```

```python
deleted_count = delete_book(book_id)
print(f"Number of documents deleted: {deleted_count}")
```

8. Formatting MongoDB Input/Output

```python
import json

with open('/content/book_list.json', 'r') as file:
    books = json.load(file)

result = books_collection.insert_many(books)
print(f"Inserted {len(result.inserted_ids)} books into MongoDB.")
```

Read the json file

then insert into collection.

8. Format output into DataFrame

```
book_list = list(books_collection.find())
```

```
df = pd.DataFrame(book_list)
df
```

	_id	title	author	synopsis
0	51b4bc79-f8b9-40ba-b5de-cecf04404268	Don Quixote	Miguel de Cervantes	A novel about a man who reads too many chivalr...
1	832e2878-44cc-4f70-833e-609ac631f4f8	Pride and Prejudice	Jane Austen	A story about manners, upbringing, and marriag...
2	c7c60768-866f-4c5b-8538-5ae87f9feac4	Moby Dick	Herman Melville	A quest for vengeance against the white whale,...
3	e493ee7e-b052-48a7-92fe-71de578ecd35	War and Peace	Leo Tolstoy	A chronicle of Napoleon's invasion of Russia.
4	9c11ad52-11f5-4a69-a6ae-285c13254a36	The Great Gatsby	F. Scott Fitzgerald	A critique of the American Dream in the Roarin...

Service on AWS: Amazon DocumentDB [link]

aws

About AWS Contact Us Support English My Account Sign In Create an AWS Account

Amazon Q Products Solutions Pricing Documentation Learn Partner Network AWS Marketplace Customer Enablement Events Explore More

Amazon DocumentDB Overview Features Pricing Resources Solutions FAQs Customers Migrate with AWS DMS

How it works

Amazon DocumentDB (with MongoDB compatibility) is a fully managed native JSON document database that makes it easy and cost effective to operate critical document workloads at virtually any scale without managing infrastructure. Amazon DocumentDB simplifies your architecture by providing built-in security best practices, continuous backups, and native integrations with other AWS services. You can enhance your applications with generative artificial intelligence (AI) and machine learning (ML) capabilities using vector search for Amazon DocumentDB and integration with Amazon SageMaker Canvas.