JOMHACK
INSPIRE INNOVATION

PYTHON
BOOTCAMP

www.jomhack.com

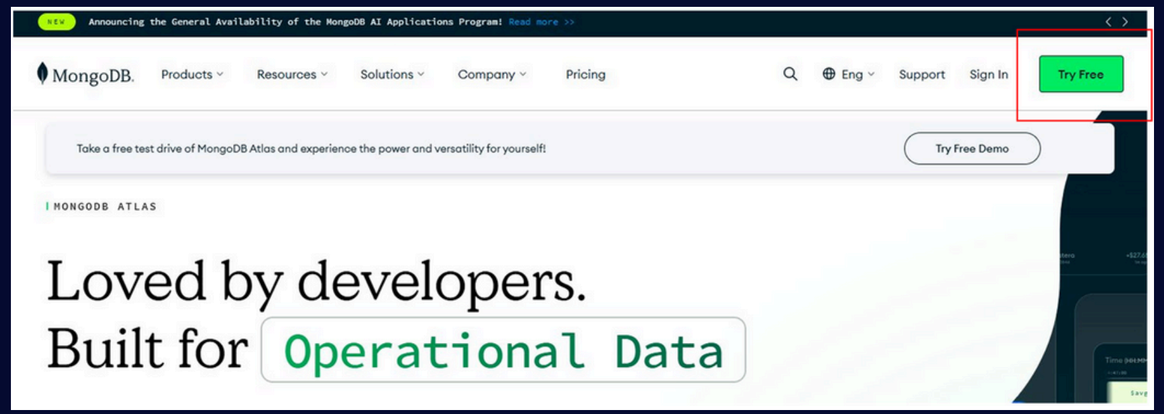
DATABASE (MONGO)

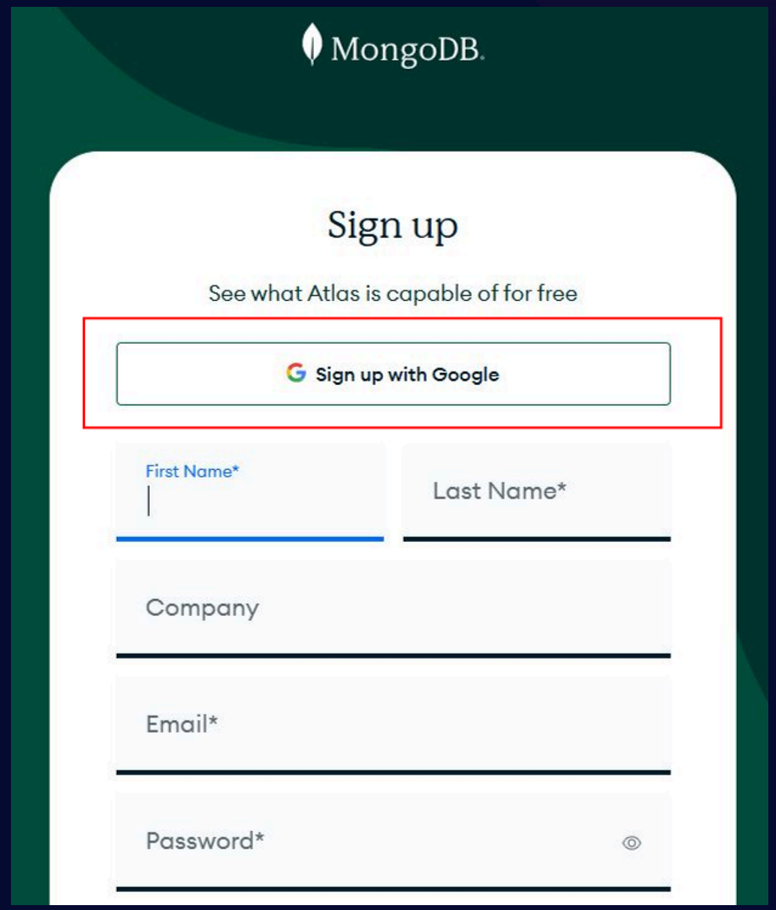
MongoDB Atlas:

- https://www.mongodb.com
- Cloud database service for MongoDB
- 512mb free tier
- pip install pymongo

DATABASE (MONGO)

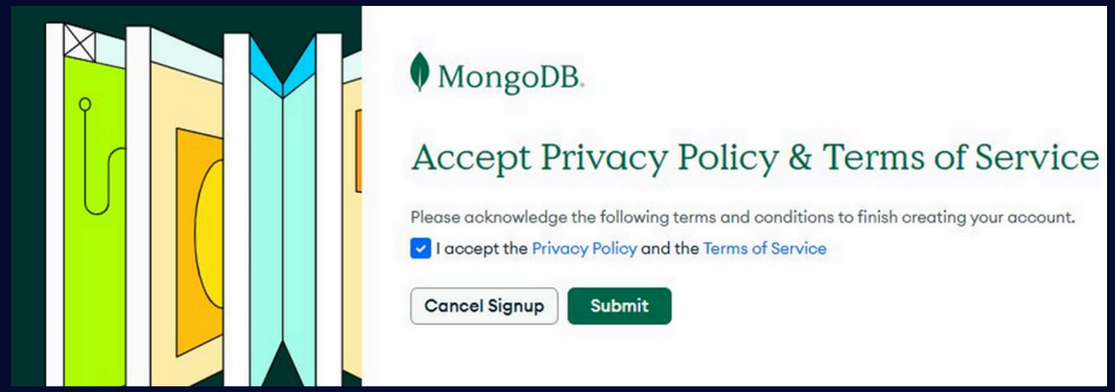
MongoDB Setup:

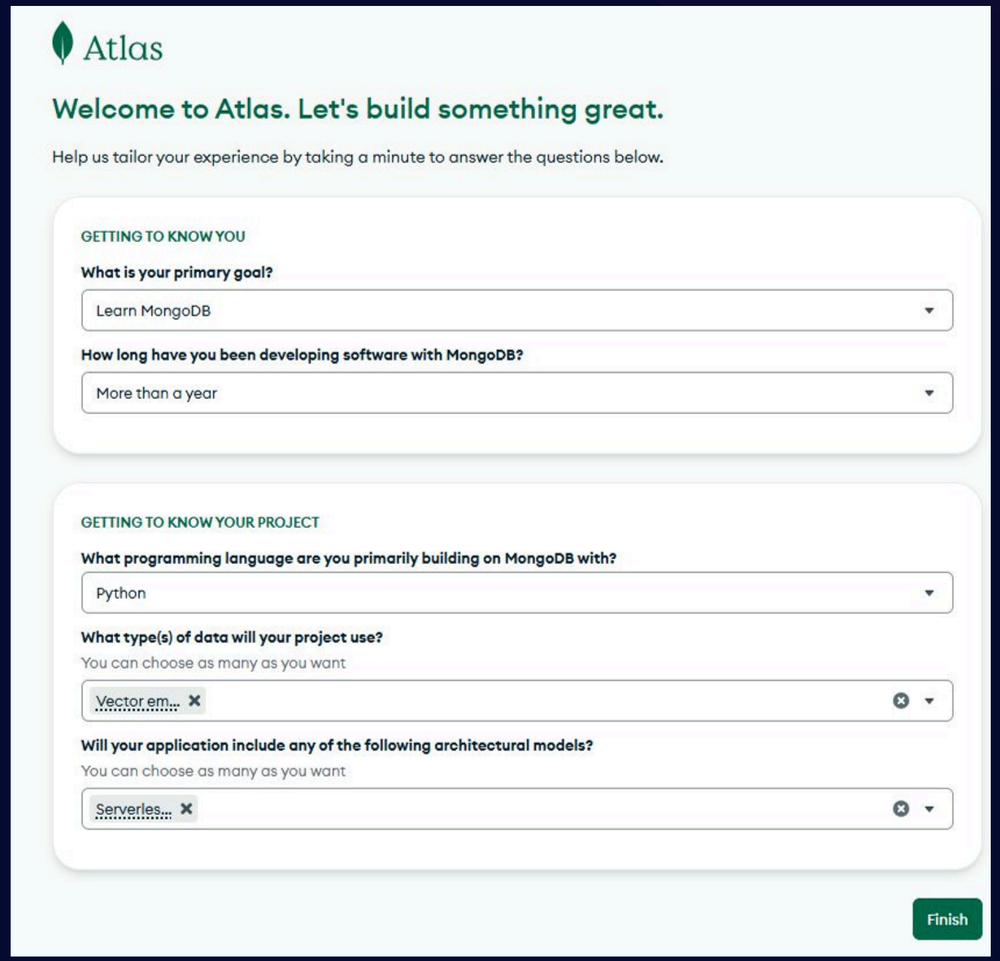




DATABASE (MONGO)

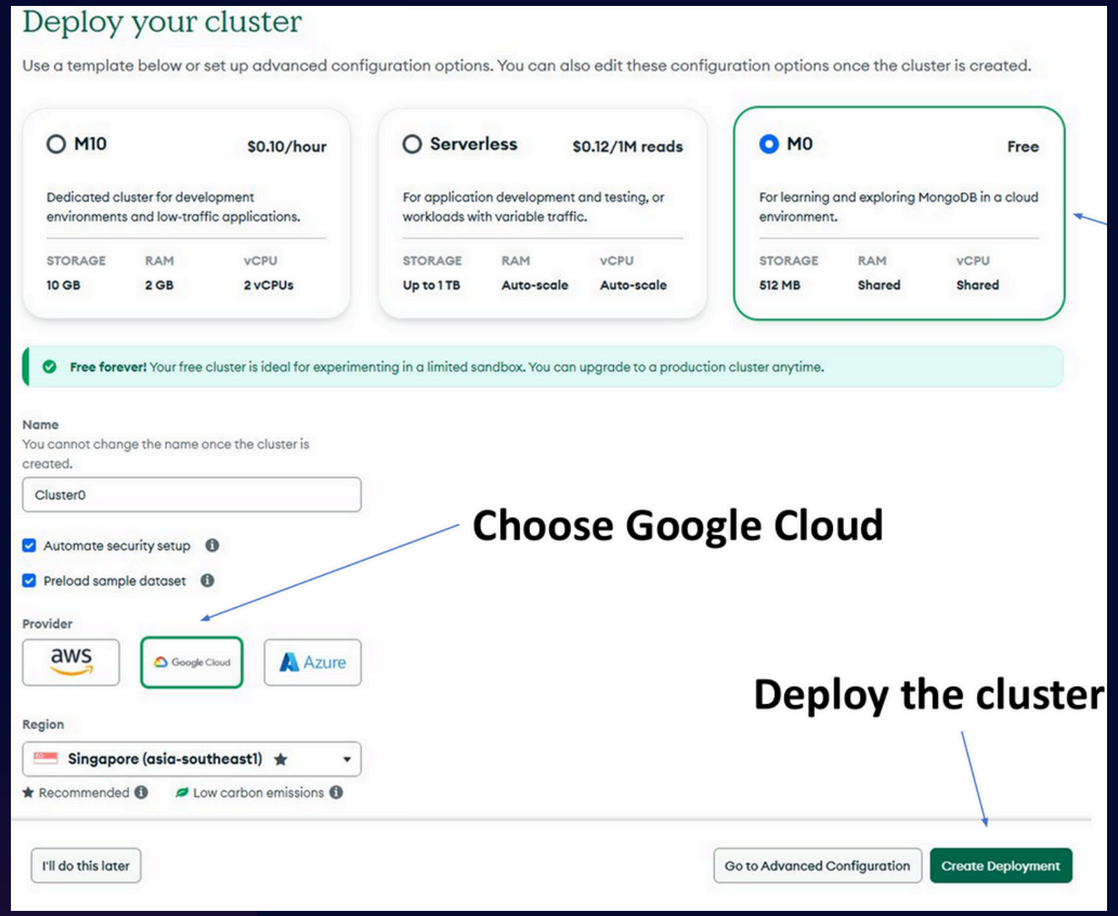
MongoDB Setup:

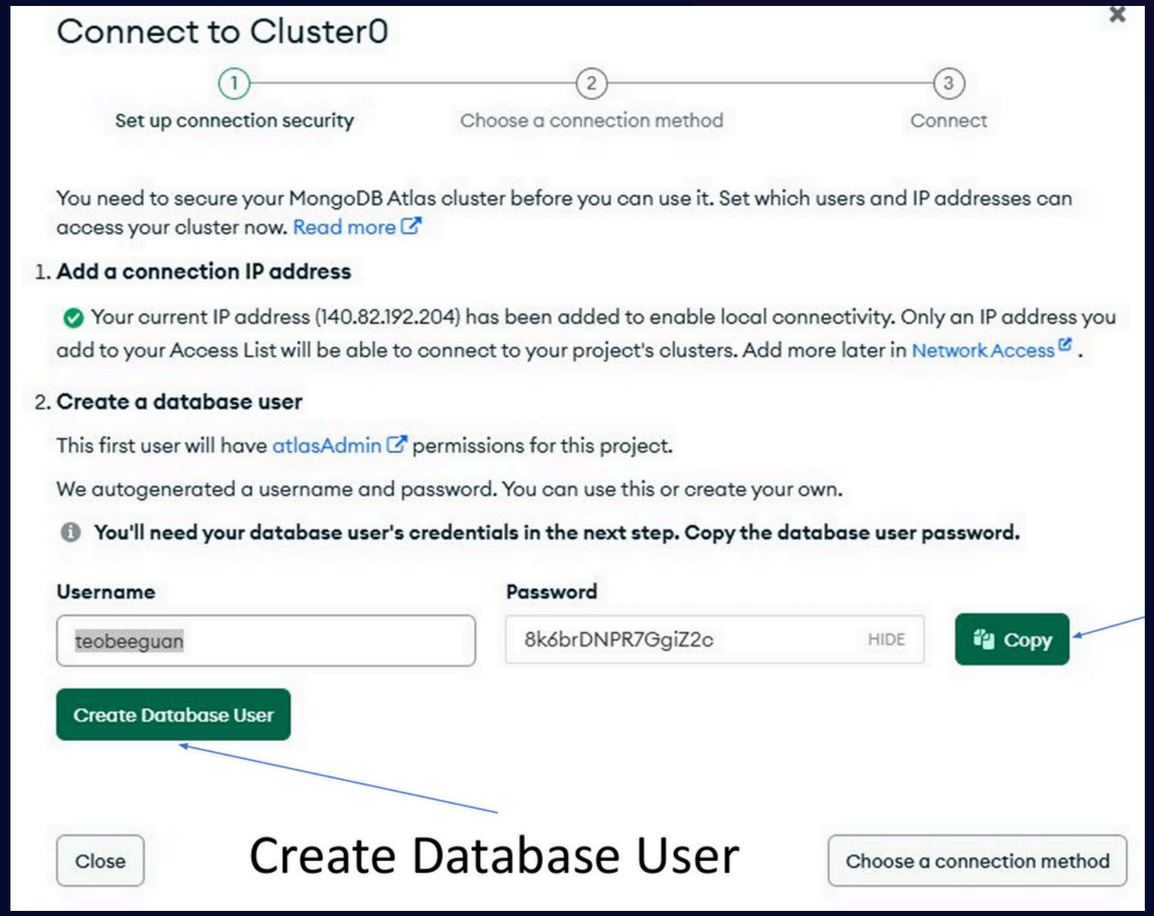




DATABASE (MONGO)

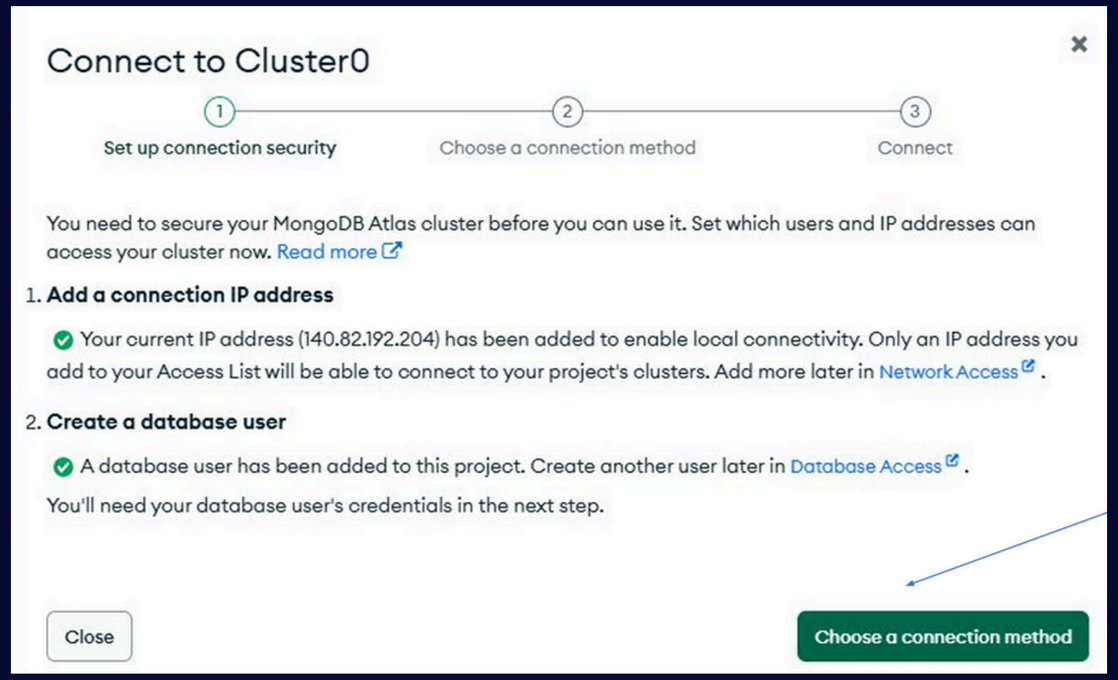
MongoDB Setup:

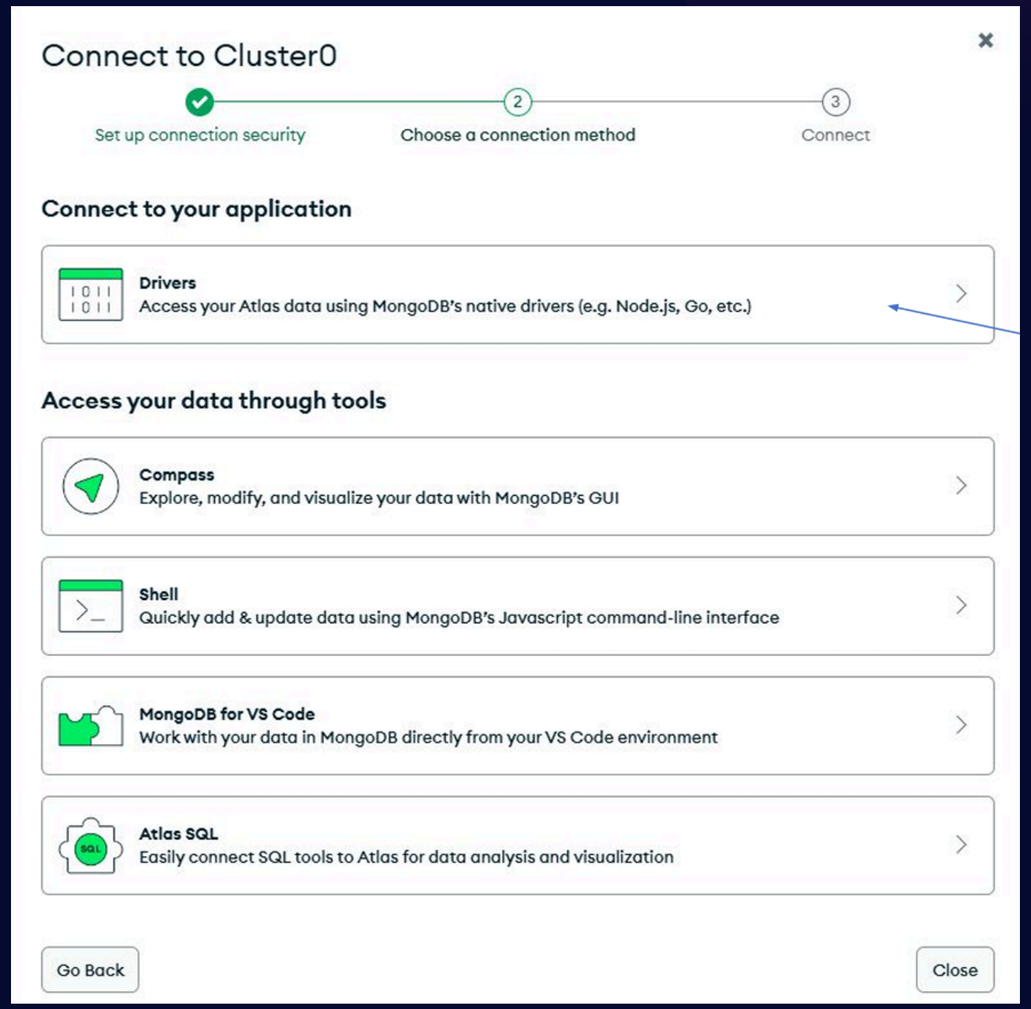




DATABASE (MONGO)

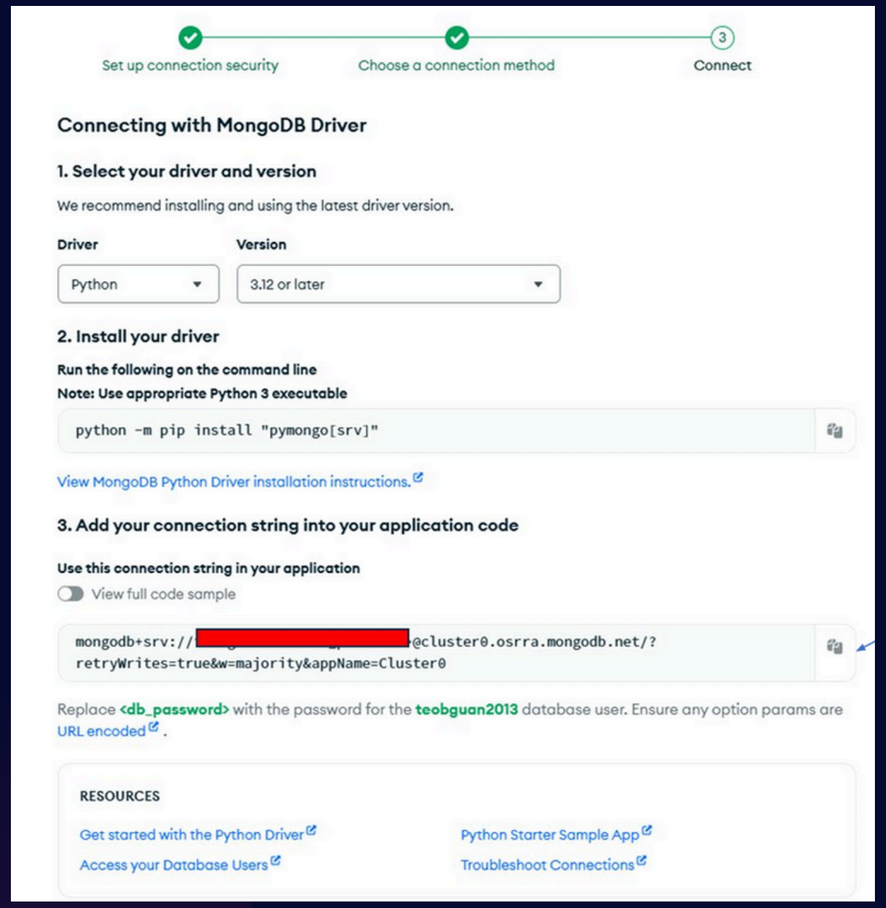
MongoDB Setup:





DATABASE (MONGO)

MongoDB Setup:



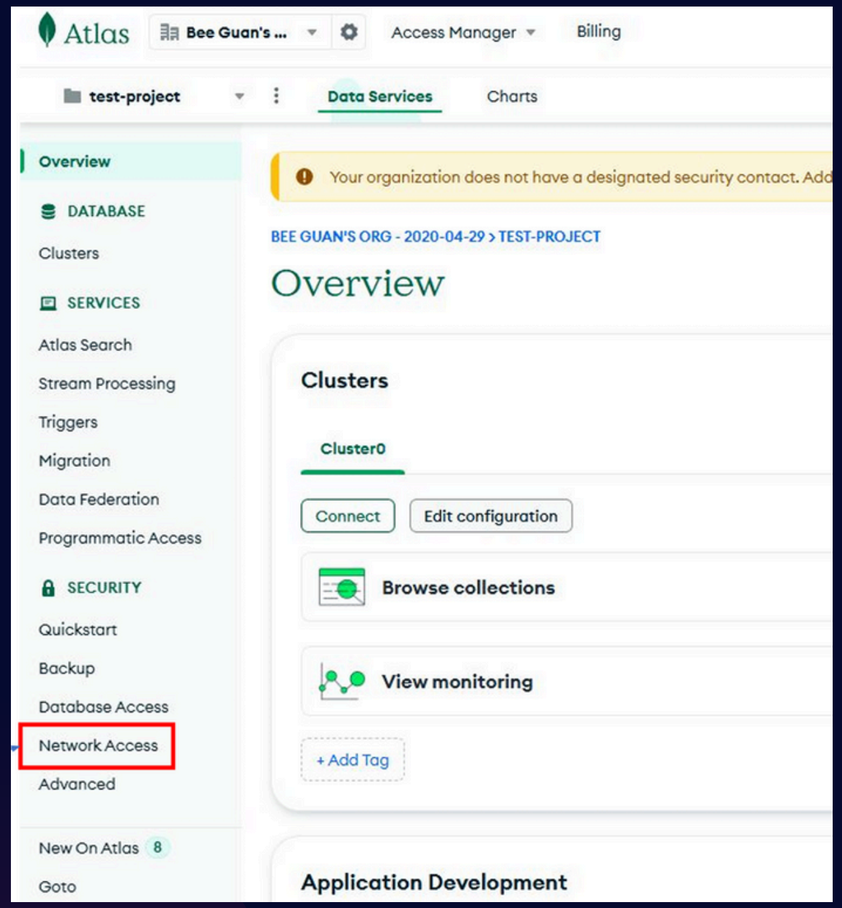
.env

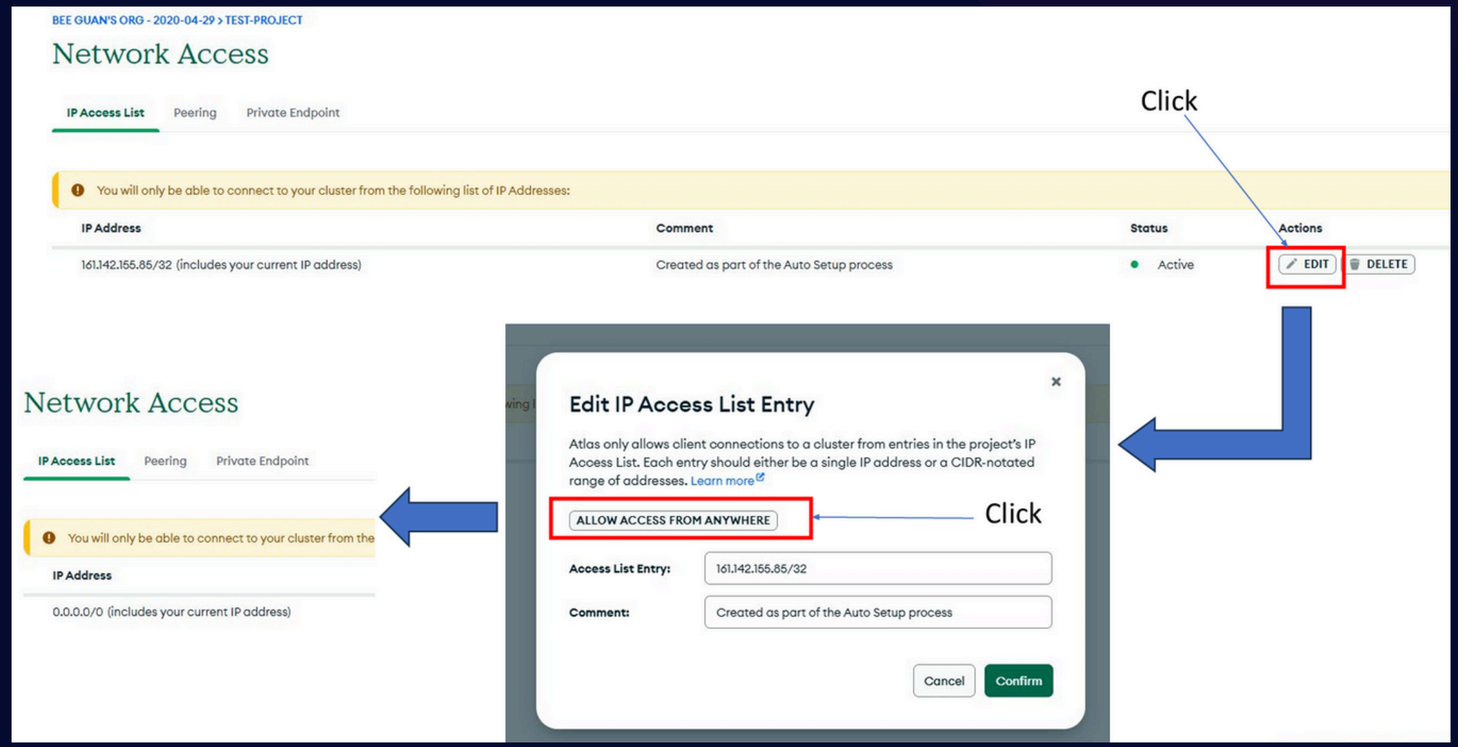
Import to Postman

```
1  MONGODB_ATLAS_CLUSTER_URI=mongodb+srv://
2
```

DATABASE (MONGO)

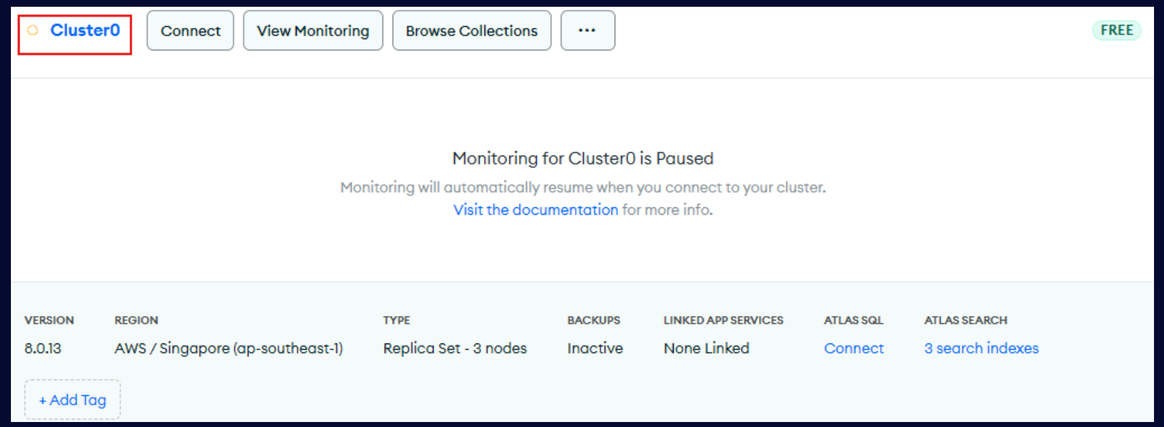
MongoDB Setup:

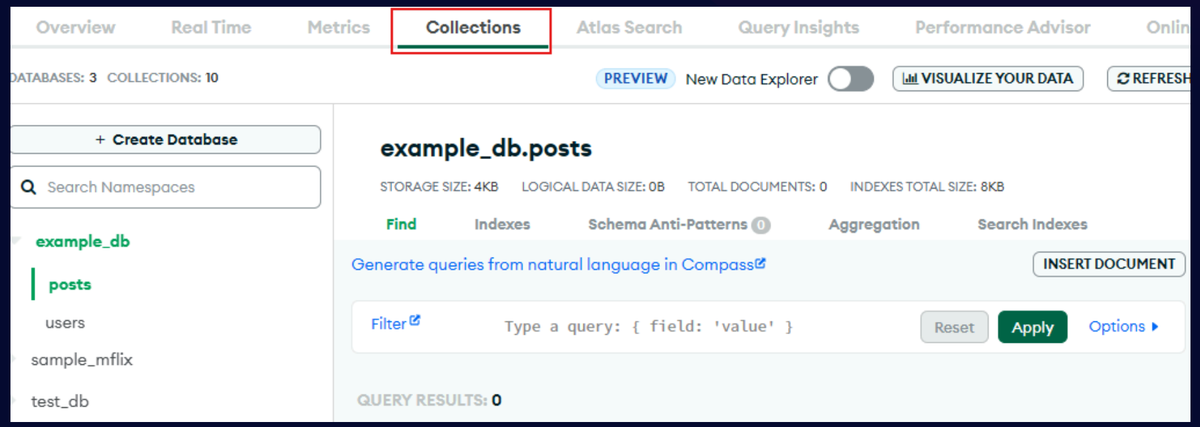




DATABASE (MONGO)

MongoDB Setup:





DATABASE (MONGO)

Initialize:

- pip install pymongo
- pip install python-dotenv

```python
from pymongo import MongoClient
from datetime import datetime
from bson.objectid import ObjectId
from dotenv import load_dotenv
import os

load_dotenv()  # Load environment variables from .env file

mongo_uri = os.getenv('MONGODB_ATLAS_CLUSTER_URI')

class DatabaseManager:
    def __init__(self, db_name='example_db', connection_string=mongo_uri):
        self.client = MongoClient(connection_string)
        self.db = self.client[db_name]
        self.users_collection = self.db.users
        self.posts_collection = self.db.posts
        self.init_database()

    def init_database(self):
        """Initialize database with collections and indexes"""
        # Create unique index on email for users
        self.users_collection.create_index("email", unique=True)
        # Create index on user_id for posts for better query performance
        self.posts_collection.create_index("user_id")
```

DATABASE (MONGO)

Create Function:

```python
    def create_user(self, name, email, age):
        """Create a new user"""
        try:
            user_doc = {
                "name": name,
                "email": email,
                "age": age,
                "created_at": datetime.now()
            }
            result = self.users_collection.insert_one(user_doc)
            return str(result.inserted_id)
        except Exception as e:
            print(f"Error: {e}")
            return None
```

Create Function:

```python
    def create_post(self, user_id, title, content):
        """Create a new post"""
        try:
            # Convert string user_id to ObjectId if it's a valid ObjectId
            if ObjectId.is_valid(user_id):
                user_object_id = ObjectId(user_id)
            else:
                user_object_id = user_id

            post_doc = {
                "user_id": user_object_id,
                "title": title,
                "content": content,
                "created_at": datetime.now()
            }
            result = self.posts_collection.insert_one(post_doc)
            return str(result.inserted_id)
        except Exception as e:
            print(f"Error creating post: {e}")
            return None
```

DATABASE (MONGO)

Read Function:

```python
    def get_all_users(self):
        """Get all users"""
        try:
            users = list(self.users_collection.find())
            # Convert ObjectId to string for display
            for user in users:
                user['_id'] = str(user['_id'])
            return users
        except Exception as e:
            print(f"Error fetching users: {e}")
            return []
```

Read Function:

```python
    def get_user_posts(self, user_id):
        """Get posts by user"""
        try:
            # Convert string user_id to ObjectId if it's a valid ObjectId
            if ObjectId.is_valid(user_id):
                user_object_id = ObjectId(user_id)
            else:
                user_object_id = user_id

            posts = list(self.posts_collection.find(
                {"user_id": user_object_id}
            ).sort("created_at", -1))

            # Convert ObjectId to string for display
            for post in posts:
                post['_id'] = str(post['_id'])
                post['user_id'] = str(post['user_id'])

            return posts
        except Exception as e:
            print(f"Error fetching posts: {e}")
            return []
```

DATABASE (MONGO)

Delete Function:

```python
    def delete_user(self, user_id):
        """Delete user and their posts"""
        try:
            # Convert string user_id to ObjectId if it's a valid ObjectId
            if ObjectId.is_valid(user_id):
                user_object_id = ObjectId(user_id)
            else:
                user_object_id = user_id

            # Delete user's posts first
            self.posts_collection.delete_many({"user_id": user_object_id})

            # Delete the user
            result = self.users_collection.delete_one({"_id": user_object_id})
            return result.deleted_count > 0
        except Exception as e:
            print(f"Error deleting user: {e}")
            return False
```

Close DB Function:

```python
    def close_connection(self):
        """Close the MongoDB connection"""
        self.client.close()
```

DATABASE (MONGO)

Run on Terminal Function:

```python
def display_menu():
    """Display the main menu"""
    print("\n" + "="*40)
    print("          DATABASE MANAGER")
    print("="*40)
    print("1. Create User")
    print("2. View All Users")
    print("3. Create Post")
    print("4. View User Posts")
    print("5. Delete User")
    print("6. Exit")
    print("-"*40)
```

```python
def main():
    """Main interactive CLI function"""
    try:
        db = DatabaseManager()
        print("✓ Connected to MongoDB successfully!")
    except Exception as e:
        print(f"✗ Failed to connect to MongoDB: {e}")
        print("Make sure MongoDB is running on localhost:27017")
        return
```

DATABASE (MONGO)

Run on Terminal Function:

```python
    while True:
        display_menu()
        choice = input("Enter your choice (1-6): ").strip()

        if choice == '1':
            print("\n--- Create New User ---")
            name = input("Enter name: ").strip()
            email = input("Enter email: ").strip()
            try:
                age = int(input("Enter age: ").strip())
                user_id = db.create_user(name, email, age)
                if user_id:
                    print(f"✓ User created successfully! ID: {user_id}")
                else:
                    print("✗ Failed to create user")
            except ValueError:
                print("✗ Invalid age. Please enter a number.")

        elif choice == '2':
            print("\n--- All Users ---")
            users = db.get_all_users()
            if users:
                for user in users:
                    print(f"ID: {user['_id']} | Name: {user['name']} | Email: {user['email']} | Age: {user['age']}")
            else:
                print("No users found.")
```

DATABASE (MONGO)

Run on Terminal Function:

```
171         elif choice == '3':
172             print("\n--- Create New Post ---")
173             user_id = input("Enter user ID: ").strip()
174             title = input("Enter post title: ").strip()
175             content = input("Enter post content: ").strip()
176             post_id = db.create_post(user_id, title, content)
177             if post_id:
178                 print(f"✓ Post created successfully! ID: {post_id}")
179             else:
180                 print("✗ Failed to create post")
181 
182         elif choice == '4':
183             print("\n--- View User Posts ---")
184             user_id = input("Enter user ID: ").strip()
185             posts = db.get_user_posts(user_id)
186             if posts:
187                 for post in posts:
188                     print(f"\nPost ID: {post['_id']}")
189                     print(f"Title: {post['title']}")
190                     print(f"Content: {post['content']}")
191                     print(f"Created: {post['created_at']}")
192                     print("-" * 30)
193             else:
194                 print("No posts found for this user.")
```

```
196         elif choice == '5':
197             print("\n--- Delete User ---")
198             user_id = input("Enter user ID to delete: ").strip()
199             confirm = input(f"Are you sure you want to delete user {user_id}? (y/N): ").strip().lower()
200             if confirm == 'y':
201                 if db.delete_user(user_id):
202                     print("✓ User deleted successfully!")
203                 else:
204                     print("✗ User not found or deletion failed.")
205             else:
206                 print("Deletion cancelled.")
207 
208         elif choice == '6':
209             print("\nClosing database connection...")
210             db.close_connection()
211             print("Goodbye! 👋")
212             break
213 
214         else:
215             print("✗ Invalid choice. Please enter 1-6.")
216 
217         input("\nPress Enter to continue...")
218 
219 if __name__ == "__main__":
220     main()
```