Data Management and Visualization

Politecnico di Torino

NoSQL with PyMongo – Practice 4

The practice purpose is to become familiar with **MongoDB** tools. You will be required to experience both

A. *(legacy) the visual application Compass, provided by MongoDB,*
B. the MongoDB python APIs (pymongo). In this second case, we have already set up a Colab notebook to avoid local configuration problems.

The python side is highly recommended, since it is broader. Ask the teacher assistant if you have problems with this path.

1. Problem specifications

The database contains Car Sharing information divided into two main collections: Bookings and Parkings. The most relevant information for each collection is shown in Table 1 (Parkings) and 2 (Bookings).

Name	Type	Description
_id	objectid	Document identifier
address	string	Parking address of the vehicle
city	string	City location of the vehicle
engineType	string	Identifier of the engine type of the vehicle
exterior	string	String describing the external condition of the vehicle during the parking
final_date	date	Date and hour of the end of the parking period
fuel	int32	Fuel level (0-100) during the parking period
init_date	date	Date and hour of the beginning of the parking period
interior	string	String describing the internal condition of the vehicle during the parking
loc	coordinates	Coordinates of the parking location
plate	int32	Identifier of the vehicle's plate
smartphoneRequired	Boolean	Boolean value denoting if the smartphone is required to
		start/finish the parking
vendor	string	Company owner of the vehicle
vin	string	Identifier of the chassis of the vehicle

Table 1: **Parkings** database info.

Name	**Type**	**Description**
_id	objectid	Document identifier
car_name	string	Vehicle’s model
city	string	City location where the vehicle has been booked
distance	int32	Distance covered during the vehicle renting
driving	object	**distance** int32 Distance covered during the vehicle renting (in meters) **duration** int32 Duration of the renting (in seconds)
engineType	string	Identifier of the engine type of the vehicle
exterior	string	String describing the external condition of the vehicle during the renting
final_address	string	Address of the final position of the renting period
final_date	date	Date and hour of the end of the renting period
final_fuel	int32	Fuel level (0-100) at the end of the renting period
init_address	int32	Address of the starting position of the renting period
init_date	date	Date and hour of the beginning of the renting period
init_fuel	int32	Fuel level (0-100) at the beginning of the renting period
interior	string	String describing the internal condition of the vehicle during the renting
plate	int32	Identifier of the vehicle’s plate
smartphoneRequired	Boolean	Boolean value denoting if the smartphone is required to start/finish the parking
vendor	string	Company owner of the vehicle
walking	object	**distance** int32 Walk distance to reach the vehicle (in meters). **duration** int32 Duration of the walking trip to reach the vehicle (in seconds).

Table 2: **Bookings** database info.

2. Database Connection (remote database)

A: Use, Install and configure Compass

You can download MongoDB Compass at https://www.mongodb.com/try/download/compass.

It allows you also to perform analysis on a local cluster, but we will skip it for now.

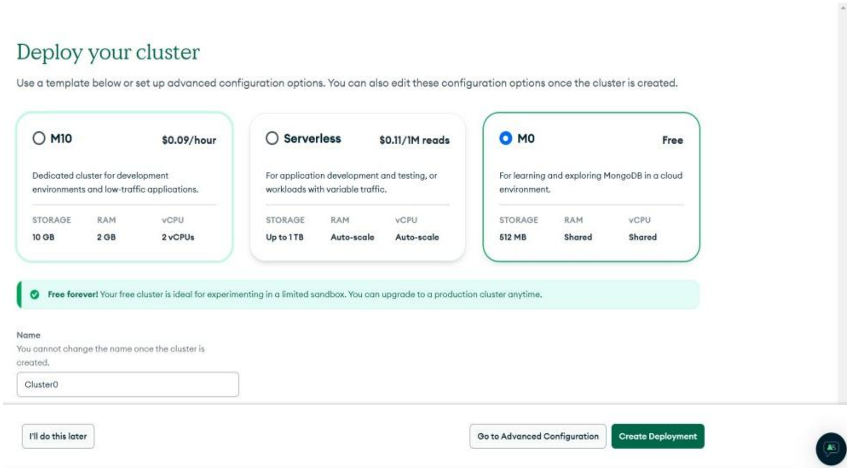
B: Work with the Colab notebook

You can find the notebook with all the instructions at the following link.
https://colab.research.google.com/drive/1nu3785xgJeSe3EMxk1xCh5JIVvSuBxNn#scrollTo=tQ4uBOR-q1ms

Remember to go to **File** > **Save a copy on Drive** before make any edit on the file. You can also download the file as a Jupyter Notebook to run it locally if you want.

Now, you need to create a free cluster to upload the database (if you do not have already one) with the following steps:

1. create a MongoDB account (https://www.mongodb.com/cloud/atlas/register)
2. Select **"Build a Cluster"** option (free cluster). The default settings are the ones set to get the account completely free.



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.

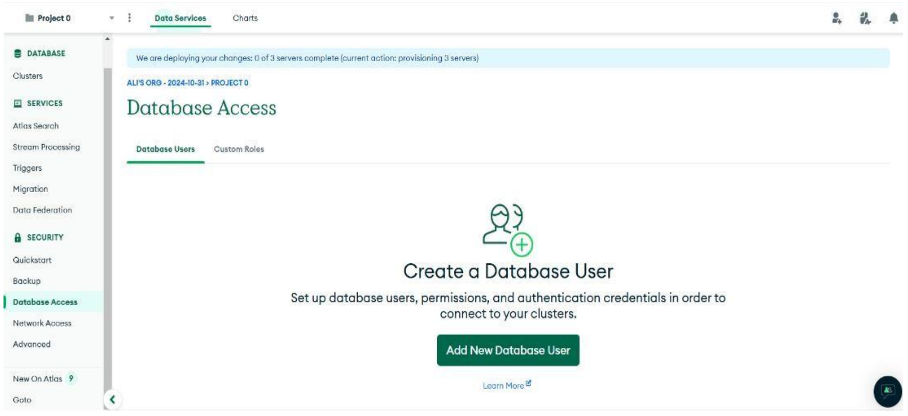
○ M10	$0.09/hour	Dedicated cluster for development environments and low-traffic applications.	STORAGE 10 GB	RAM 2 GB
○ Serverless	$0.11/1M reads	For application development and testing, or workloads with variable traffic.	STORAGE Up to 1 TB	RAM Auto-scale
● M0	Free	For learning and exploring MongoDB in a cloud environment.	STORAGE 512 MB	RAM Shared

Option	vCPU
M10	2 vCPUs
Serverless	Auto-scale
M0	Shared

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

Name
You cannot change the name once the cluster is created.
Cluster0

3. give a name to your cluster (default: cluster0)
4. Close the connection options (we will return here later)
5. Under **Database Access**, click on **"Add New Database User"**
   a. select authentication method with password
   b. In the section **"Database User Privileges"**, Select the role **"Read and write to any database"** in the **"Built-in Role"** sub-section.
   c. fill in the form and click on **"Add User"**.



We are deploying your changes: 0 of 3 servers complete (current action: provisioning 3 servers)

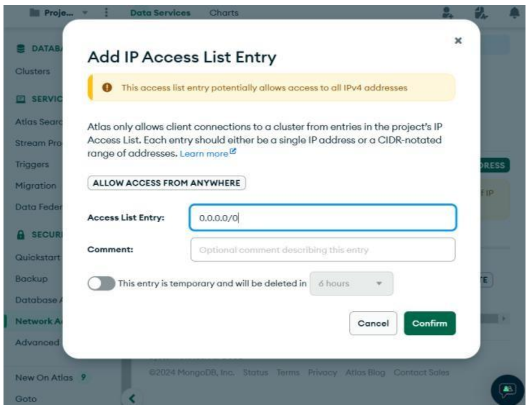
ALI'S ORG - 2024-10-31 > PROJECT 0

Database Access

Create a Database User

Set up database users, permissions, and authentication credentials in order to connect to your clusters.

6. Configure remote access by clicking on **"Network Access"**
   a. click on **"Add IP Address"**
   b. enter in the Access List Entry field `0.0.0.0/0`


Add IP Access List Entry

This access list entry potentially allows access to all IPv4 addresses

Atlas only allows client connections to a cluster from entries in the project's IP Access List. Each entry should either be a single IP address or a CIDR-notated range of addresses. Learn more

Access List Entry: 0.0.0.0/0

Comment: Optional comment describing this entry

This entry is temporary and will be deleted in 6 hours

7. Get string connection from MongoDB Atlas server
    a. Under the **"Database"** page from the side menu, select **"Clusters"**. Now you can see the cluster that you've created
    b. click **"Connect"** near the newly created cluster and select "Drivers"
    c. copy the connection string, similar to:

```
mongodb+srv://<user_db_name>:<db_password>@cluster0.kudqw.mongodb.net/?retryWrites
=true&w=majority&appName=<cluster_name>
```

- **[Python Notebook]** Paste the connection string into the appropriate field in the notebook and **Run** the cell.
- **[MongoDB Compass]** On the left panel, go to the **"Connections"** section
    - Click on **"add connection"** (+ sign)
    - Paste the connection string and click "Connect"

8. Finally, download the following JSON files locally from the following link.

You can either create a Database, 2 collections, and add the data via Compass, or **follow the instructions on the notebook** to create the database 'lab4' with pymongo and the set of collections.

3. Analyze the database using visualizations

1. (Bookings) Identify the most common percentage(s) of fuel level at the beginning of the renting period.
2. (Bookings) Identify the most common percentage(s) of fuel level at the end of the renting period.
3. (Parkings) Identify the time range(s) with most parking requests (init date).
4. (Parkings) Identify the time range(s) with most booking requests (end parking).
5. (Parkings) Visualize on the map the vehicles having the fuel level lower than 5%.

4. Querying the database

1. (Parkings) Find the plates and the parking addresses of the vehicles that begin the booking (end parking) after 2017-09-30 at 6AM.
   Hint: it is possible to use the function Date("<YYYY-mm-ddTHH:MM:ss>")
2. (Parkings) Find the addresses and the level of fuel of the vehicles that during the parking period had at least 70% of fuel level. Order the results according to descending value of fuel level.
3. (Parkings) Find the plate, the engine type and fuel level for 'car2go' vehicles (vendor) with good internal and external conditions.
4. (Bookings) For the renting that required a walking distance greater than 15 Km (to reach the vehicle), find the hour and the fuel level at the beginning of the renting period. Order results according to decreasing initial fuel level.

5. Data Aggregation

1. (Bookings) Group documents according to their fuel level at the end of the renting. For each group, select the average fuel level at the beginning of the renting period.
2. (Bookings) Select the average driving distance for each vendor. On average, for which vendor the users cover longer distances?

Bonus Queries

1. (Parkings) Find the vehicles parked less than a mile far from Piazza San Carlo (coordinates: 7.683016, 45.067764).
   Hint: use the operators $geoWithin and $centerSphere.
2. (Parkings) Repeat the query at the previous step using the coordinates of a place of personal interest in Turin (e.g. Politecnico di Torino) using Open Street Maps to find the exact coordinates (www.openstreetmap.org, inverse the coordinates order).