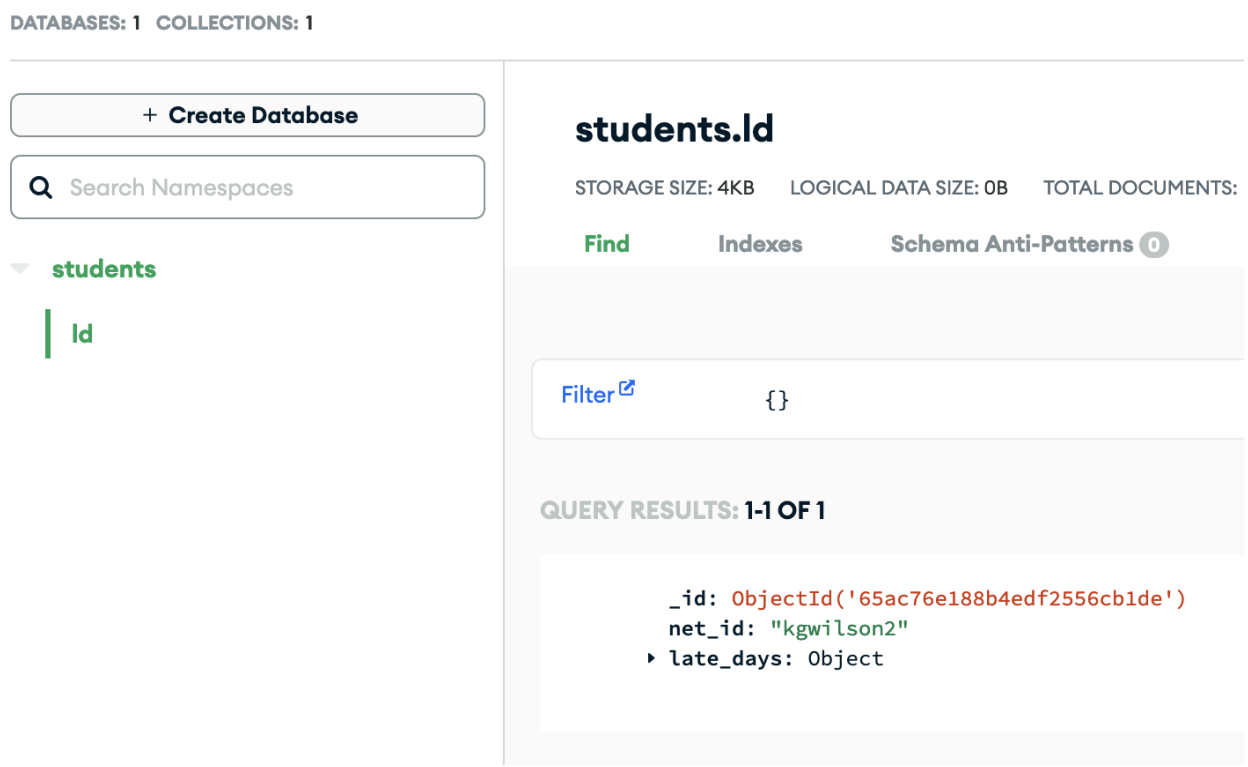
Accounts Info

Google account: REDACTED FOR SECURITY
Password: REDACTED FOR SECURITY
Recovery account: REDACTED FOR SECURITY
MongoDB Atlas pwd: REDACTED FOR SECURITY
MongoDB user: REDACTED FOR SECURITY
MongoDB user pwd: REDACTED FOR SECURITY

MongoDB Introduction

Please note that the system runs on the M0 (FREE) tier. Check here for limitations, but most importantly “Atlas free tier allow 500 connections and a max of 500 collections (100 databases) in total. It also allows only 100 crud requests per second at max.” The storage size for M0 clusters is 512MB but one semester’s worth of data only takes up about 50KB so we’re fine there. Also note that the IP access list is set to any so that GradeScope’s servers have access.

MongoDB Atlas is a fully managed, cloud-based database service that simplifies the deployment, scaling, and maintenance of MongoDB databases. We use it to track the late days of our students.



Terms: A **cluster** in MongoDB is a group of interconnected servers that work together to store and manage data; a **database** is a container for collections and documents, providing a logical organizational unit for data; a **collection** is a grouping of MongoDB documents, similar to a table in a relational database; and a **document** is a JSON-like data structure containing key-value pairs, the basic unit of data in MongoDB.

Here's the basics of MongoDB:

```
# Import necessary modules from the PyMongo library
from pymongo.mongo_client import MongoClient

# MongoDB connection URI containing authentication details and cluster
information.
uri = "REDACTED FOR SECURITY"

# Create a new client and connect to the MongoDB server
client = MongoClient(uri)

# The name of the database and collection you want to access in Cluster0
database_name = "students"
collection_name = "ld"

try:
      # Select the specific database and collection
      db = client[database_name]
      collection = db[collection_name]

      # Perform a find_one operation to retrieve a document from the
collection with a net_id property of "kgwilson2"
      document = collection.find_one({"net_id" : "kgwilson2"})
      print("Found document:", document)
      # if found: {"_id": ObjectId(65ac76e188b4edf2556cb1de), "net_id" :
"kgwilson2", "late_days" : {"2" : 3, "4" : 1}}
      # if not found: None

except Exception as e:
      print(e)
```

Get Late Days Assignment:

We use a GradeScope assignment that students can submit anything to in order to get back the late days they've used so far into the semester. Using GradeScope allows us to verify student net_ids and ensure privacy of student data.

The actual assignment is a Docker image like any other, and it's quite simple. However, it does use a system that's a bit different from the other assignments in order to be more light-weight (since it doesn't handle any user files). The README.md file in there does a good job of explaining what each file does.

There should be no maintenance to do here except copying the GradeScope assignment for each new semester and uploading the same .zip file. The .zip file will be in the project design repo in the Late_Days folder. Changes will need to be made if the cluster, user, user password, database name (students), or collection name (ld) are changed on MongoDB Atlas.

Project Late Days Update on Submission

Whenever a student makes a late submission on GradeScope, this is the relevant late day code that gets run:

```
def update_late_days(netid, proj_num, ld_used):
    """
    Creates a connection to the MongoDB cluster and updates
    the student's late days with a backoff strategy.
    With a maxPoolSize of 399, the maximum number of concurrent
    connections that can be established to the MongoDB cluster
    is limited to 399. This ensures that the number of connections
    does not exceed the limit of 500 and does not generate a warning
    (which occurs at 80% of max connections). This helps prevent connection
    errors or performance degradation. When the function no longer
    needs the connection, it closes it, returning it to the pool.
    Short timeout (5 sec) helps reduce load on the cluster too.

    Parameters:
        netid (str): student whose late days should be updated
        proj_num (int): index of project to update, i.e. 3 for P3
        ld_used (int): number of late days used for project

    Returns:
        late_days (dict): late days dictionary where keys are project
                numbers as ints and values are late days used as ints
                (note that late days can be recorded past 12 total.)
                {
                    '3': 1,
                    '7': 3
```

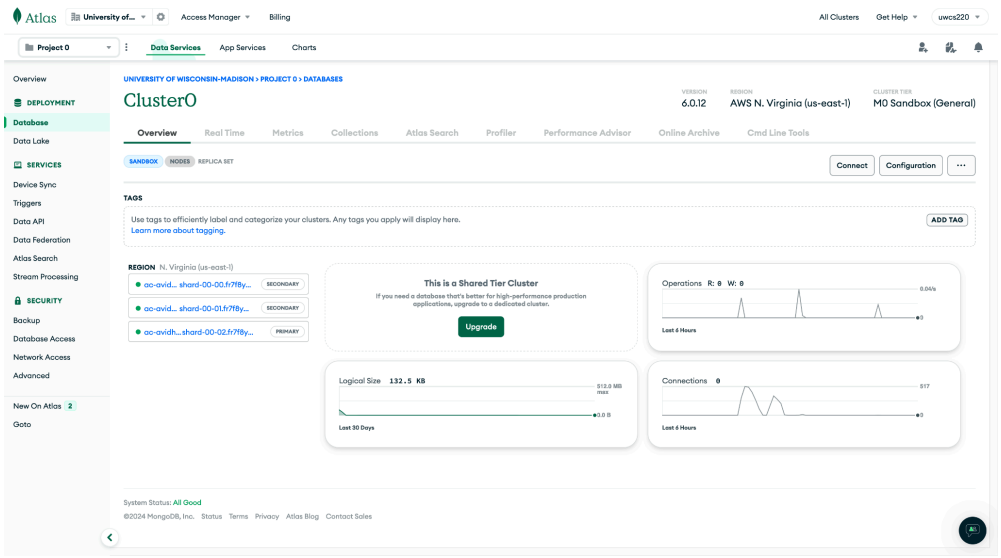
```
            }
        None: if operation fails
    """

    max_attempts = 15
    filter_query = {"net_id": netid}
    update_query = {
        "$set": {f"late_days.{proj_num}": ld_used}
    }
    options = {
        "upsert": True,
        "return_document": ReturnDocument.AFTER
    }
    for _ in range(max_attempts):
        time.sleep(random.uniform(1,7))
        try:
            client = MongoClient(
                URI,
                ssl=True,
                tls=True,
                tlsAllowInvalidCertificates=True,
                maxPoolSize=399,
                connectTimeoutMS=5000,
                socketTimeoutMS=5000,
                serverSelectionTimeoutMS=5000,
            )
            collection = client[DB_NAME][COLLECTION_NAME]
            result = collection.find_one_and_update(
                filter_query,
                update_query,
                **options
            )
            return result.get("late_days") if result else None
        except Exception as e:
            print(e)
            if client:
                client.close()
            continue
    print('Failed to connect to MongoDB Cluster and update late days.')
    return None
```

The relevant pieces here to understand are that a backoff system with a random start is used in order to reduce the number of simultaneous connections. This is important because when re-running an autograder with many late submissions, it might be possible (though highly unlikely) that this function could be running simultaneously in more than 500 places. In addition, short timeouts (5 seconds) are used to reduce the load on the cluster. Finally, a maxPoolSize of 399 is used to keep concurrent connections lower than 400 because MongoDB generates an alert whenever a free cluster goes above 400 connections and 500 is the max. The function returns the late days dictionary of the student with the given net_id.

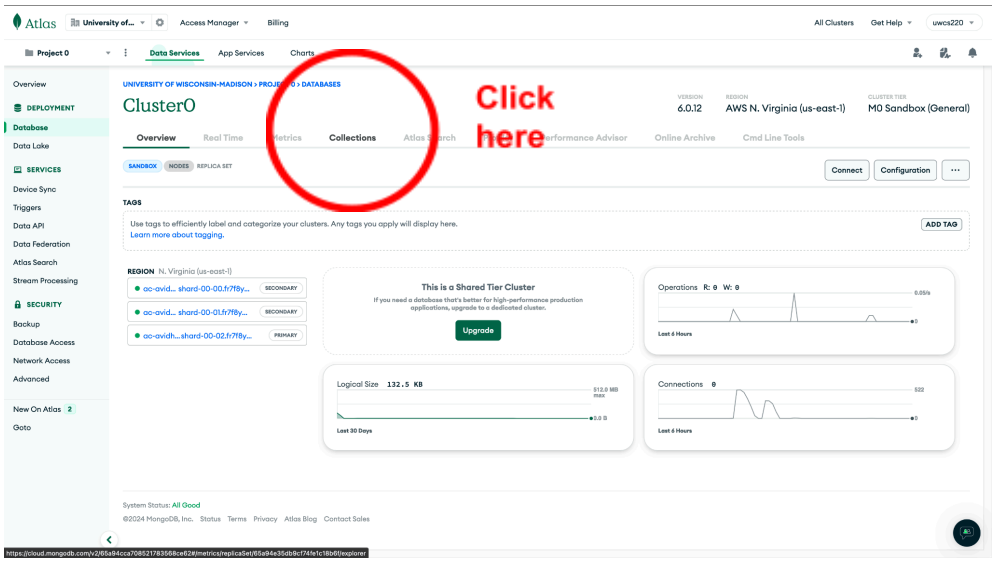
Access and Modifications

Although I've done my best to ensure that the system will run without any issues for the rest of time, I'll include some instructions here about how to access the cluster, check on it, and make necessary modifications.

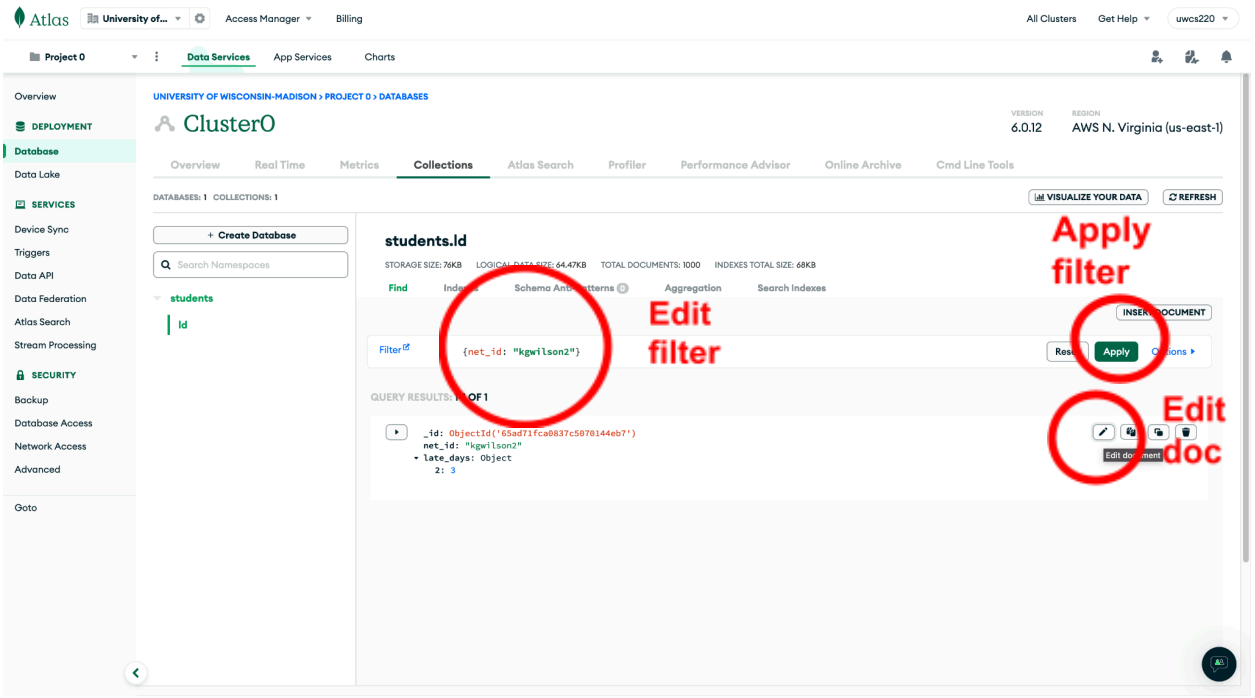
You can login to the database using this link https://account.mongodb.com/account/login. This will bring you to this nice GUI.



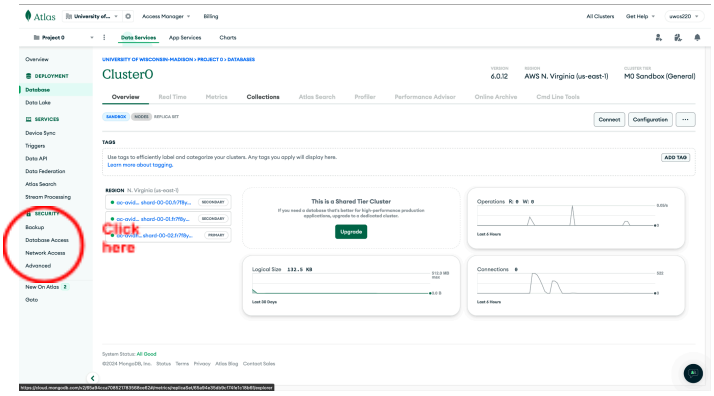
To view the data in the students database, click on the collections tab:



Once you're in the collections tab, you can see the storage size, the total number of documents in the collection, and other useful info. You can also find the data for individual students using the filter field. The screenshot below shows how this can be used to find the data for a given student. The screenshot also shows how you can edit the document in the case that a mistake has been made for this student.



In the case that database access needs to be updated, click on the Database Access tab in the left panel



Then, click on the edit button to change the info for this user. Keep in mind that this user needs to have admin privileges, and that if you change the user's name or password, you will need to change the URI connection string following this format:

```
URI =
"mongodb+srv://<user>:<password>@cluster0.fr7f8yc.mongodb.net/?retryWrites=
true&w=majority"
```