

Topic: Document Databases

Description

Document databases allow users to store data in text-based documents. Binary objects and non-textual data (e.g., images, video, etc.) can be wrapped in MongoDB documents. However, they do not take advantage of the powerful retrieval methods built in to the document databases. It is most useful to use a searchable, semi-structured form such as JSON, BSON, or XML. With semi-structured data, the database can be searched for documents with specific properties.



Like most NoSQL database systems, document databases are used when consistency is not as important as availability and constant updates. Advantages of document databases are that content schemas are flexible, documents are easy to store, applications are easy to write, and the documents are independent units. The last point is important to having the database replicate and shard the data – for distributed, parallel processing of independent documents.

A document database can be used for a wide range of applications (e.g., mobile apps and websites). They are often used in cloud applications because they can be housed on inexpensive hardware and replicated easily. You might imagine a forum being hosted on a document database. Each entry could be formatted in JSON/XML to have an author, subject, date and text of post. Queries could search in parallel for the posts of a particular author, postings with certain title or for specific keywords in the text.

Learning objectives

- become familiar with document databases
- experience manipulating data from a shell interface (command line)
- learn how to execute queries in a document database
- (future) learn how to use a programming API to interact with a document database

Components

For this assignment, you will be installing MongoDB Compass (GUI) software on your personal computers. This will connect to MongoDB Atlas (cloud storage) for basic data storage and retrieval via the mongo shell.

Background Resources

<http://nosql-database.org>
<http://www.mongodb.com/document-databases>

Setup

To do these exercises, you will need to install MongoDB Compass on your computer to connect to the cloud DB.

Step #0: Setup a MongoDB cloud instance

Visit <https://www.mongodb.com> and use the Try Free option to create a new account for MongoDB Atlas. Alternatively, you could install MongoDB on your own machine and not connect to the cloud, but this requires more installation.

A screenshot of a web browser displaying the MongoDB website at https://www.mongodb.com. The page features a dark header with the MongoDB logo and navigation links for Products, Solutions, Resources, Company, Pricing, a search bar, Sign In, and a prominent green 'Try Free' button. The main content area contains promotional text about MongoDB Atlas and a large 'Try Free' button.

Fill in the form to get a free account. NEVER type in your credit card details. You do not need to give any credit card data

to use the free tier.

The form consists of several input fields:

- Your Company (optional)
- How are you using MongoDB? (dropdown menu)
- Your Work Email
- First Name
- Last Name
- Password (with a note: 8 characters minimum)
- A checkbox for agreeing to terms of service and privacy policy.
- A green "Get started free" button.
- Text at the bottom: "Already have an account? [Sign in.](#)"

Use your new account email and password to login to <https://cloud.mongodb.com>.

Step #0.1: Create a new project with any project name (e.g., CISLab) and create the project with default settings.

The screenshot shows the MongoDB cloud interface under the 'Projects' tab. On the left is a sidebar with links like 'Alerts', 'Activity Feed', 'Settings', etc. The main area has a search bar and tabs for 'Project Name', 'Clusters', 'Users', 'Teams', 'Alerts', and 'Actions'. A prominent green 'New Project' button is located in the top right corner.

Create a Project

This is the 'Name Your Project' step of the wizard. It shows a progress bar with 'Name Your Project' and 'Add Members'. A text input field contains 'CISLab'. To the right are 'Cancel' and 'Next' buttons.

Create a Project

This is the 'Add Members and Set Permissions' step. It includes a 'Create Project' button at the top right. Below it, there's a section for adding members via email and a dropdown for selecting a member ('leidijon@gsu.edu (you)'). On the right, a sidebar titled 'Project Member Permissions' lists five roles with their descriptions:

- Project Owner: Has full administration access
- Project Cluster Manager: Can update clusters
- Project Data Access Admin: Can access and modify a cluster's data and indexes, and kill operations
- Project Data Access Read/Write: Can access a cluster's data and indexes, and modify data
- Project Data Access Read Only: Can access a cluster's data and indexes
- Project Read Only: May only modify personal preferences

Step #0.2: Build a new database which will be hosted in the MongoDB cloud. **Make sure to only select the free/shared DB option.** Let's just use the default options – simply click the *Create* button. You could have selected whether you want to use Amazon (aws), GoogleCloud, or Microsoft cloud (Azure) in Advanced Configuration Options.

The screenshot shows the MongoDB Atlas deployment options page. At the top right is the MongoDB logo and the text "MONGODB ATLAS". Below it is the heading "Deploy a cloud database" and a sub-instruction: "Experience the best of MongoDB on AWS, Azure, and Google Cloud. Choose a deployment option to get started." There are three main deployment options shown in boxes:

- Serverless** (Preview): For serverless applications. Starting at \$0.30/1M reads.
- Dedicated**: For production applications. Starting at \$0.08/hr*.
- Shared** (Free): For learning and exploring MongoDB in a cloud environment. No credit card required to start.

Each box has a "Create" button. A dashed arrow labeled "Advanced Configuration Options" points from the bottom right towards the "Shared" box.

Step #0.3: Deploy a new cluster. Again, simply use the default options to create a free Shared Cluster0 with no backup.

The screenshot shows the "Create a Shared Cluster" form. At the top left is the breadcrumb "CLUSTERS > CREATE A SHARED CLUSTER". The title "Create a Shared Cluster" is centered above the form. The "Shared" tab is selected, indicated by a green border and the word "FREE".

The form includes the following sections:

- Cloud Provider & Region**: Set to "AWS, N. Virginia (us-east-1)". Options include AWS, Google Cloud, and Azure.
- Region Selection**: A large grid of regions with icons and names. The "N. Virginia (us-east-1)" region is highlighted with a green border.
- Cluster Tier**: Set to "M0 Sandbox (Shared RAM, 512 MB Storage) Encrypted".
- Additional Settings**: Set to "MongoDB 5.0, No Backup".
- Cluster Name**: Set to "Cluster0".

After a few minutes, your cluster will be created on the Amazon cloud and you will now be able to connect to your database to store and retrieve data. Let's setup a username (e.g., myAdminUser) and a password for this cluster so that Compass (on your local computer) can connect and login to your DB. Click *Create User*.

Step #0.4: Configure users and access permissions.

The screenshot shows the MongoDB Atlas security setup interface. On the left, there's a sidebar with various options like Database, Data Lake, PREVIEW, DATA SERVICES, Triggers, Data API, Data Federation, Atlas Search, SECURITY (which is highlighted in green), and Quickstart. Under Quickstart, there are links for Database Access, Network Access, and Advanced. A 'New On Atlas' button is also present.

Security Quickstart

To access data stored in Atlas, you'll need to create users and set up network security controls. [Learn more about security setup](#)

1 How would you like to authenticate your connection?

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



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

myAdminUser

Password

.....

Autogenerate Secure Password

Copy

Create User

Click on “Add My Current IP Address”. This will allow only applications (such as MongoDB Compass) installed on your computer at your location to login to the DB.

2 Where would you like to connect from?

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

My Local Environment
Use this to add network IP addresses to the IP Access List. This can be modified at any time.

Cloud Environment
Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters.

IP Address	Description	
Enter IP Address	Enter description	Add Entry
Add My Current IP Address		

Finish and Close

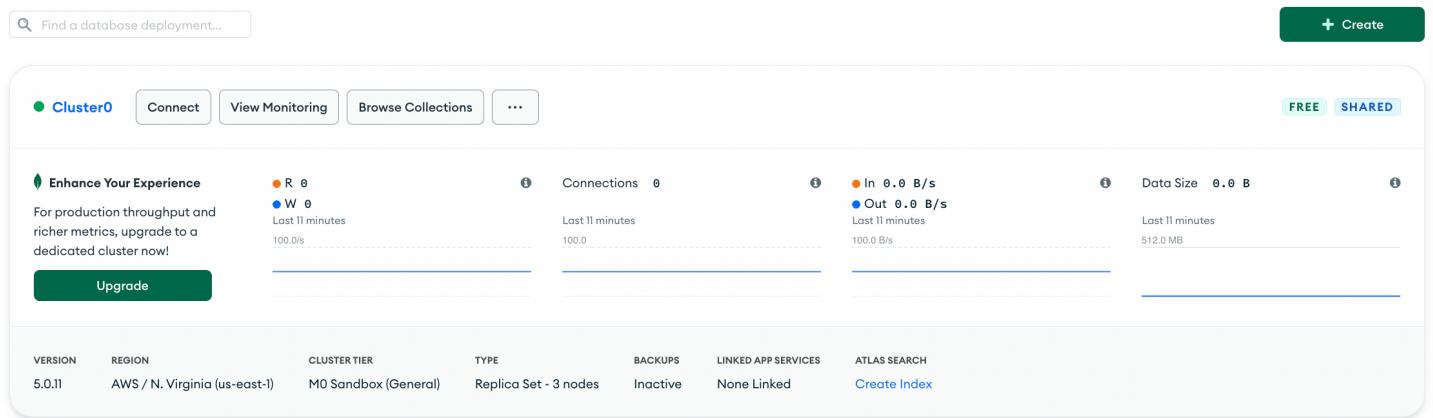
Click “Finish and Close” after adding your IP address.

At this point, you could develop a application/program/website to store data on this cluster in the cloud. Common programming languages such as Java, Python, and R have APIs and libraries that allow you to connect to the DB. You could also login to the DB via a command line tool to run commands. We will simply choose to install and connect to the DB using *MongoDB Compass* tool on your computer to interact with your DB.

Step #1.0: Downloading MongoDB Atlas from the cloud to your machine.

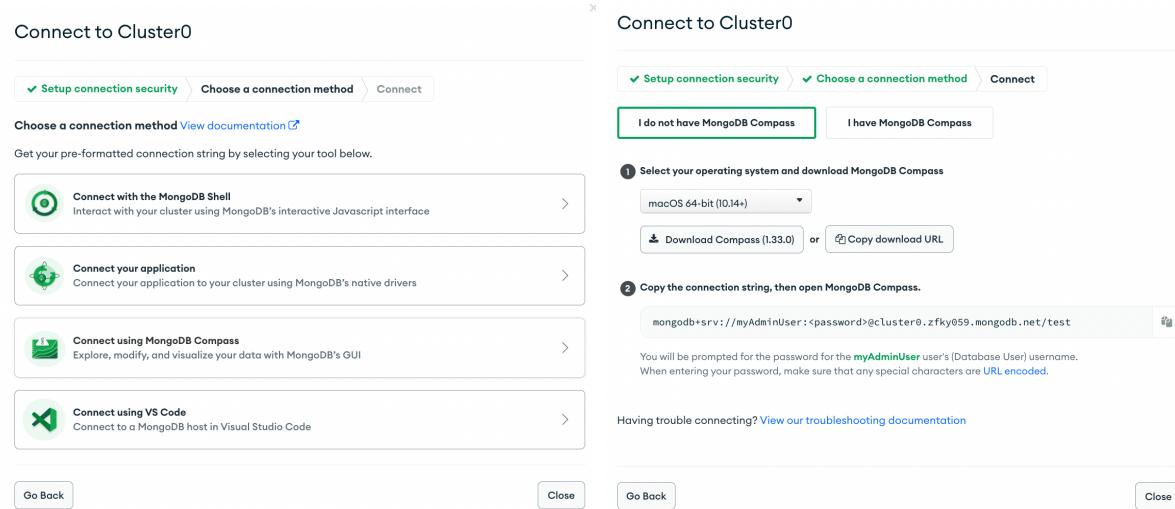
For this step, you will have to download and install MongoDB Compass on your own computer. See the instructor in office hours or after class if you have difficulties with this step. From MongoDB Atlas, menus, click “Connect” within your Cluster0 instance.

Database Deployments



The screenshot shows the MongoDB Atlas Cluster Overview dashboard. At the top, there's a search bar with placeholder text "Find a database deployment..." and a green "Create" button. Below the search bar are navigation buttons: "Cluster" (highlighted with a green dot), "Connect", "View Monitoring", "Browse Collections", and "...". On the right, there are "FREE" and "SHARED" buttons. The main area displays cluster metrics: "Enhance Your Experience" (upgrade to a dedicated cluster), "Connections" (0), "Data Size" (0.0 B), and "Throughput" (0 R/s, 0 W/s). Below the metrics is a table with cluster details: VERSION (5.0.11), REGION (AWS / N. Virginia (us-east-1)), CLUSTER TIER (M0 Sandbox (General)), TYPE (Replica Set - 3 nodes), BACKUPS (Inactive), LINKED APP SERVICES (None Linked), and ATLAS SEARCH (Create Index).

From the Connect popup window, select “Connect using MongoDB Compass”, “Download Compass”, and then install MongoDB compass on your personal computer.



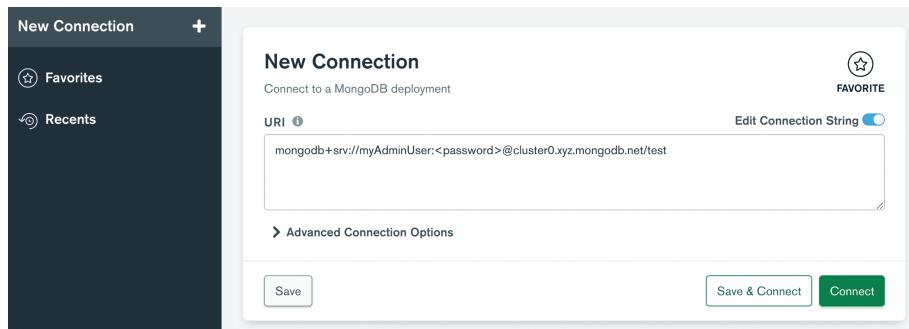
The screenshot shows two side-by-side "Connect to Cluster0" popups. The left one is titled "Choose a connection method" and lists options: "Connect with the MongoDB Shell", "Connect your application", "Connect using MongoDB Compass" (selected), and "Connect using VS Code". The right one is titled "Choose a connection method" and shows "I do not have MongoDB Compass" and "I have MongoDB Compass" buttons. It also includes steps for selecting an operating system (macOS 64-bit) and downloading MongoDB Compass (v1.33.0), or copying the download URL. Both popups have "Go Back" and "Close" buttons at the bottom.

Question 1 to turn-in: what is your DB connection string? 5% of your grade.

e.g., `mongodb+srv://myAdminUser:<password>@cluster0.crjd1.mongodb.net/test`

Step #1.1: Connecting MongoDB Compass on your personal computer to the MongoDB Atlas Cloud.

In these steps, we will practice interacting with the database using the GUI interface. There are other ways to interact with the database, e.g., via commands from a command shell or within a Python or Java program. After downloading and installing Compass, run the new application. Use your connection string to *connect* to your Atlas Cloud cluster. In this connection string, replace the default “<password>” with your cluster’s actual password, e.g., “myAdminUser:someGreatPassword”.



The screenshot shows the "New Connection" dialog in MongoDB Compass. It has a sidebar with "Favorites" and "Recents" sections. The main area is titled "New Connection" with a sub-instruction "Connect to a MongoDB deployment". It shows a URI input field containing `mongodb+srv://myAdminUser:<password>@cluster0.crjd1.mongodb.net/test`. There's a "FAVORITE" button with a star icon. Below the URI is an "Edit Connection String" toggle switch. At the bottom are "Save" and "Connect" buttons.

Now, let's use the Compass GUI to create a new DB. Note that the actual DB and documents will be hosted in the Atlas cloud. Compass is simply an interface to add, change, search, and delete the documents currently in the cloud. Create a new document DB (*db*) and the first collection for it to contain (*students*). Click Create Database.

The screenshot shows the Compass GUI interface. On the left, there's a sidebar with 'HOSTS', 'CLUSTER', 'EDITION', and a 'My Queries' section. The main area shows 'admin' and 'local' sections with 'Storage size' and 'Collections' counts. A 'Create database' dialog is open on the right, prompting for 'Database Name' (set to 'db') and 'Collection Name' (set to 'students'). Under 'Advanced Collection Options', three options are listed: 'Capped Collection' (unchecked), 'Use Custom Collation' (unchecked), and 'Time-Series' (unchecked). At the bottom of the dialog are 'Cancel' and 'Create Database' buttons.

Navigate to the *db.students* collection on the left menu. From here, we can manually add data, upload a large dataset from a file, run queries, modify documents, etc. We could try doing this by the GUI (*ADD DATA*) or by the command line shell (the shell is located at the bottom of your screen, `>_mongosh`).

The screenshot shows the Compass GUI with the 'Local' database selected. The left sidebar lists 'HOSTS', 'CLUSTER', 'EDITION', and a 'My Queries' section. The main area shows the 'db.students' collection details: 0 documents, 0B total size, 0B avg size, and 1 index, 4.0KB total size, 4.0KB avg size. The 'Documents' tab is active. Below it, there's a search bar with a filter ('{ field: 'value' }'), 'OPTIONS', 'FIND', 'RESET', and a 'REFRESH' button. A message says 'Displaying documents 0 - 0 of N/A'. At the bottom, there's a green 'Import Data' button with a file icon. The bottom of the screen shows a terminal prompt: '>_MONGOSH'.

Step #2.0: Try CRUD commands

Data is organized within (mostly similar) documents, with multiple independent documents stored within a collection within a larger database. We will use the command line to add several documents to this database. The shell client is an avenue for you to manually enter a CRUD command to MongoDB and view the output results. See the help/documentation: <https://docs.mongodb.com/manual>

Step #2.1: Verification of your cloud connection

Let's try to add a new doc to the *students* collection inside the *db* database, be careful with smart quotes and regular quotes “” versus “”. First, use the command line to navigate through your database and collections. In practice, you might have multiple DBs and collections for different applications, datasets, etc.

```
> show dbs           displays available dbs  
> db                shows current db  
> use __             switch the current db  
> show collections  displays collections
```

```
> MONGOSH  
> show dbs  
< db      8.19 kB  
admin    377 kB  
local    6.55 GB  
> db  
< test  
> use db  
< 'switched to db db'  
> show collections  
< students  
Atlas atlas-fmtu9o-shard-0 [primary] db>
```

Step #2.2: Create several documents

Try to insert your first student document from the command line. Note the success message that is returned from the cloud server (<) with your document's auto-generated ID.

```
> db.students.insertOne({"gnumber":"G00000001"});
```

The console will provide output acknowledging the new document and its new unique identifier.

```
> db.students.insertOne({"gnumber":"G00000001"});  
< { acknowledged: true,  
     insertedId: ObjectId("6137a683b4486e2c5076c52d") }  
Atlas atlas-fmtu9o-shard-0 [primary] db>
```

Create another new document in the *students* collection. Note an arbitrary unique ObjectId was generated.

```
> db.students.insertOne(  
{  
  "gnum": "G0001234",  
  "firstname": "Jonathan",  
  "middlename": null,  
  "lastname": "Smith",  
  "hobbies": ["SCUBA", "Falconry", "Gardening", "Outdoorsing", 0, null],  
  "age": 25,  
  "gpa": 3.9,  
  "gradstudent": true,  
  "currentlyenrolled": false,  
  "cars": [  
    { "make": "Ford", "model": ["F150 (Red)", "F150 Crew Cab (Black)"] },  
    { "make": "Chevy", "model": ["Equinox (Silver)"] }  
  ],  
  "x": [{ "key1": "val1"}, {"key2": "val2", "key3": "val3"}]  
)
```

The unique ObjectId is generated by MongoDB for each new document based on its creation timestamp, client machine id, client process id, and counter, e.g., ObjectId("6137a77cb4486e2c5076c52e").

4d	0a	c9	75	bb	30	77	32	66	f3	9f	e8
0	1	2	3	4	5	6	7	8	9	10	11
time	mid	pid	inc								

Each MongoDB document organizes data into separate key-value pairs (`"gnum": "G0001234"`), arrays (`"hobbies": ["SCUBA", "Falconry", "Gardening", "Outdoorsing", 0, null]`), and nested sub-documents (`{"key1": "val1"}`).

Question 2 to turn-in: what is your second document's ObjectId? 5% of your grade.

e.g., { acknowledged: true, insertedId: ObjectId("6137a77cb4486e2c5076c52e") }

Step #2.3: Read

MongoDB queries were originally aimed at finding a set of independent documents without joining any independent documents together on a server. Thus, the documents could be distributed on multiple resources and queried efficiently in parallel. Later, the `$lookup` command was added to allow joins (e.g., left joins/uncorrelated subqueries). The `find` command will list all the documents in a collection, specific documents, or subsets of a full documents.

Try the following three queries. Remember to use your second document's ObjectId instead of mine.

Find all documents

```
> db.students.find()
```

Find a specific document by id

```
> db.students.find({ "_id" : ObjectId("63164be0e9b02764d2761d16") })
```

Find a document and only return two fields (by marking attributes as either 1=returned or 0=hidden in the query result)

```
> db.students.find({ "_id" : ObjectId("63164c38e9b02764d2761d17") }, {firstname:1, age:1})
```

Boolean clauses can also be used to specify conditions to check against every document to filter out undesired documents. This is done by selecting a field, conditional operator, and value. E.g., `population : { $lte : 2000000 }` filters out documents that do not have a population less than or equal to 2000000.

These clauses can be logically joined as Boolean conjunctions (X AND Y must both be true). Commas between clauses indicate that the following three conditions must be true to retrieve a document.

```
> db.students.find(
  {
    "gradstudent":true,
    age: {$lt: 26},
    lastname : {$in: ["Smith", "Brown", "Jones"]}
  }
)
```

These clauses can also be logically joined as disjunctions (either X OR Y must be true). The three indices of the array hold three different clauses, where one or more of the clauses must be true to add the document to the query result.

```
> db.students.find(
  {
    $or: [
      {"gradstudent":true},
      {age: {$lt: 26}},
      {lastname : {$in: ["Smith", "Brown", "Jones"]}}
    ]
  }
)
```

Boolean expressions for queries can also be used in the same query. As you may expect, query clauses can contain operators such as greater than, less than, in, etc. See more details on queries options at:

<https://docs.mongodb.com/manual/reference/operator/query>

Arrays values can be queried to find exact matches for the full array:

```
> db.students.find( { hobbies:  
  ["SCUBA","Falconry","Gardening","Outdoorsing",0,null]  
} )
```

Arrays values can be queried to find partial matches for the array, either at any position or in a specific index position.

```
> db.students.find( { hobbies: "SCUBA" } )          (search within all positions of the array)  
> db.students.find( { "hobbies.2": "Gardening" } )   (Gardening search only at index #2 in the array)  
> db.students.find( { "hobbies.3": "Gardening" } )   (no results? Gardening is not stored at index #3)
```

Step #2.4: *Update*

Documents can be modified or replaced using update, updateOne (the first document that matches the query), updateMany, or replaceOne. Let's add skills to our student's document. modifiedCount reports how many documents were changed. Use your own ObjectId.

```
> db.students.updateMany({  
  _id :ObjectId("63164c38e9b02764d2761d17")},  
  {$set:{'skills':  
    {db:"MongoDB", version:5, nosql:"document"}  
  }}  
})
```

Did the document get modified, check?> db.students.find({ "_id" : ObjectId("63164c38e9b02764d2761d17") })

Updates require criteria (i.e., a filter) that is used to locate the documents to change and the updated data. \$set specifies which portions of the document to change.

```
> db.students.updateOne(  
{ "_id":ObjectId("63164c38e9b02764d2761d17")},  
{$set:{'middlename':'B'}})
```

updateMany changes all documents that match the query (e.g., a GPA range).

```
> db.students.updateMany(  
{gpa:{$gt:3.782}},  
{$set:{'status':'Cum Laude'}}  
)
```

Verify the document has changed *skills*, *middlename*, and *status* values.

```
> db.students.find({ "_id" : ObjectId("63164c38e9b02764d2761d17") })
```

Step #2.5: *Delete*

Deleting documents (via delete, deleteOne, and deleteMany) follows the same query logic as the find and update commands. The operation also reports how many documents were deleted.

```
> db.students.insertOne({hello:"world"})  
> db.students.deleteMany({hello:"world"})
```

Step #2.6: *GUI Interactions*

If you refresh your GUI (above your console), you will see these two documents.

Try using the GUI's **FILTER** and **FIND** query options to search for one of your documents. Note that you can review the results in either a list, JSON document file, or tabular form

You can manually edit or delete document, e.g., changing SCUBA to SCUBA DIVING, by double clicking a key-value.

At this point, you can also go back to your web browser, browse your new *db.student* collection, and query your dataset in the cloud via the Atlas interface. Atlas will provide similar results to Compass.

The screenshot shows the MongoDB Compass interface with the 'db.students' collection selected. The collection has 603B documents and 2 indexes. A query result shows two documents. The first document is a simple object with an '_id'. The second document is a more complex student profile with fields like 'firstname', 'lastname', 'age', 'gpa', 'gradstudent', 'cars', and 'x'.

Step #3: Use MongoDB CRUD commands to store and manipulate a custom dataset

Refer to these MongoDB examples. Keep a copy of all of your commands to turn in. Execute the commands and record the output from each step.

1. Download a simple, small dataset of JSON documents of your choice. As an example:

U.S. Government data - https://catalog.data.gov/dataset?res_format=JSON

2. Add at least five SEPARATE documents based on a subset of your dataset – 10% grade.

Note to yourself: how is this different than inserting one document that contains five sub-documents?

Hint: DO NOT insert one big document that contains five nested sub-documents! Each document should get its own document ID.

3. Generate a query that retrieves a document based on its unique id – 20% grade.
4. Generate a query that utilizes an ‘and’ condition – 20% grade.
5. Generate a query that utilizes an ‘or’ condition – 20% grade.
6. Update one document by adding a new field based on an updateOne or updateMany statement – 10% grade.
7. Delete one or more documents – 10% grade.

TURN IN: A script (text file) that contains your answers from question 1 and 2 as well as all of your commands from steps 3.2-3.7 along with their output.

You are now welcome to **terminate** your Cluster0 and **delete** your CISLab project.

The screenshot shows the MongoDB Atlas Cluster0 dashboard. It indicates a shared tier cluster with three shards: shard-00-00.zf..., shard-00-01.zf..., and shard-00-02.zf... (SECONDARY). The dashboard also shows operations (R: 0.03, W: 0) and connections (20).

Organization

Projects

Alerts 0

Activity Feed

Settings

Integrations

Access Manager

Billing

Support

Live Migration

Projects

New Project

Find a project... 

Project Name	Database Deployments	Users	Teams	Alerts	Delete Project
CISLab	1 Deployment	1 User	0 Teams	0 Alerts	 
lab1	1 Deployment	1 User	0 Teams	0 Alerts	 

Done for the day!

Want to learn more about MongoDB for your own benefit?

Optional Step #1: Aggregation and Analysis – 0% of grade

Try out:

```
db.students.aggregate([
  {$match:{ which documents? }},
  {$group:{ how to group?, what to calculate} }
])
```

As an example, count all of the students with an interest in SCUBA DIVING for Professor Leidig's club.

Optional Step #2: Embedded Programming API – 0% of grade

Note: MongoDB has great APIs for programming and connectivity.

Most applications and projects require the use of programs to interact with the API instead of the shell client. The API defines how you can send requests (e.g., to add a document or run a query) from a program you are writing to the DB. As an example of an application that would benefit from an API, imagine a web-based program that allows customers to order products and check on these orders.

Download and install the MongoDB libraries/packages required by a programming language such as Java, Python, etc.
Develop a program that inserts, queries, updates, and deletes documents from database collection.