MongoDB and Atlas Cloud

Note: *These instructions are meant to get anyone up and running with the technologies we are using in class. There may be other ways to complete these tasks, and other applications you can use, and you are welcome to do so as long as you understand how they work.*

MongoDB is a NoSQL database, and the primary Database Management System that we are using in this course.
This page explains how to install MongoDB, as well as how to access your local Mongo database. It also explains how to create and access a database using MongoDB Atlas, a cloud database solution, and how to connect it to your web app on Heroku.

Installing MongoDB

Go to **https://docs.mongodb.com/manual/installation/#tutorials (https://docs.mongodb.com/manual/installation/#tutorials)** to install MongoDB on your machine. This allows you to create a local database and is essential when developing your app locally (and for running our lecture examples).

Creating and Running a Local Mongo Database Server

You can quickly create and run a Mongo database server once you have MongoDB installed on your machine.

1. Decide where you want to store your database's data files (likely in the same directory as your web app).
2. Create a folder to store your databases's data:

   ```
   $ mkdir mongo-data
   ```

3. Start up the database server:

   ```
   $ mongod --dbpath mongo-data
   ```

   Your terminal window will run the database server on your localhost port 27017 until you end the process. You can now access it as necessary.
4. Add `mongo-data/` to your `.gitignore` file - you should not commit your local database to your git repo. If you ever want to start your database server with an empty database, simply delete the `mongo-data` folder and make a new empty one.

Connecting to your local Mongo Database in NodeJS

Follow the lecture code examples to see how to connect to your local database with Node. You can connect directly using something like MongoClient from the mongodb package, or through Mongoose if you're using models.

Either way, you will have to name the database within your database server you want to connect to. For example, below is the URI to connect to your localhost database server, and to access the StudentAPI database, which will store your collections and documents for a students API.

```
const mongoURI = 'mongodb://localhost:27017/StudentAPI'
```

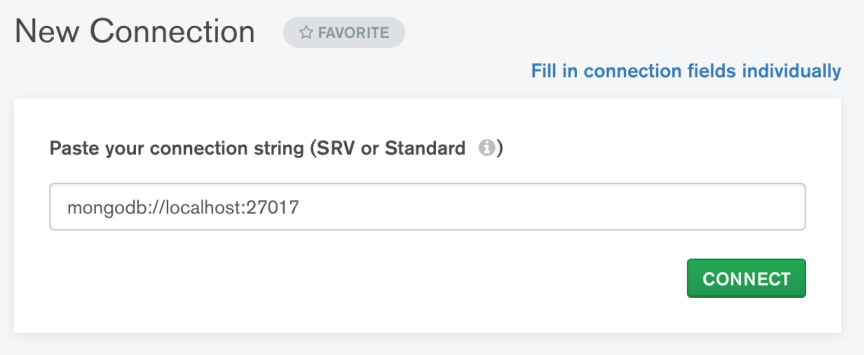
MongoDB Compass

MongoDB Compass is an easy to use GUI for accessing your Mongo database (local and cloud). Go to **https://www.mongodb.com/download-center/compass (https://www.mongodb.com/download-center/compass)** and install MongoDB Compass

Setting up MongoDB Compass for your local database server

Connecting MongoDB Compass to your local database server is simple:

1. Make sure your local Mongo database server is running (for example, by running `mongod --dbpath <database_folder_path>`).
2. Insert the following connection string for a New Connection. It should be `mongodb://localhost:27017`, as shown below:



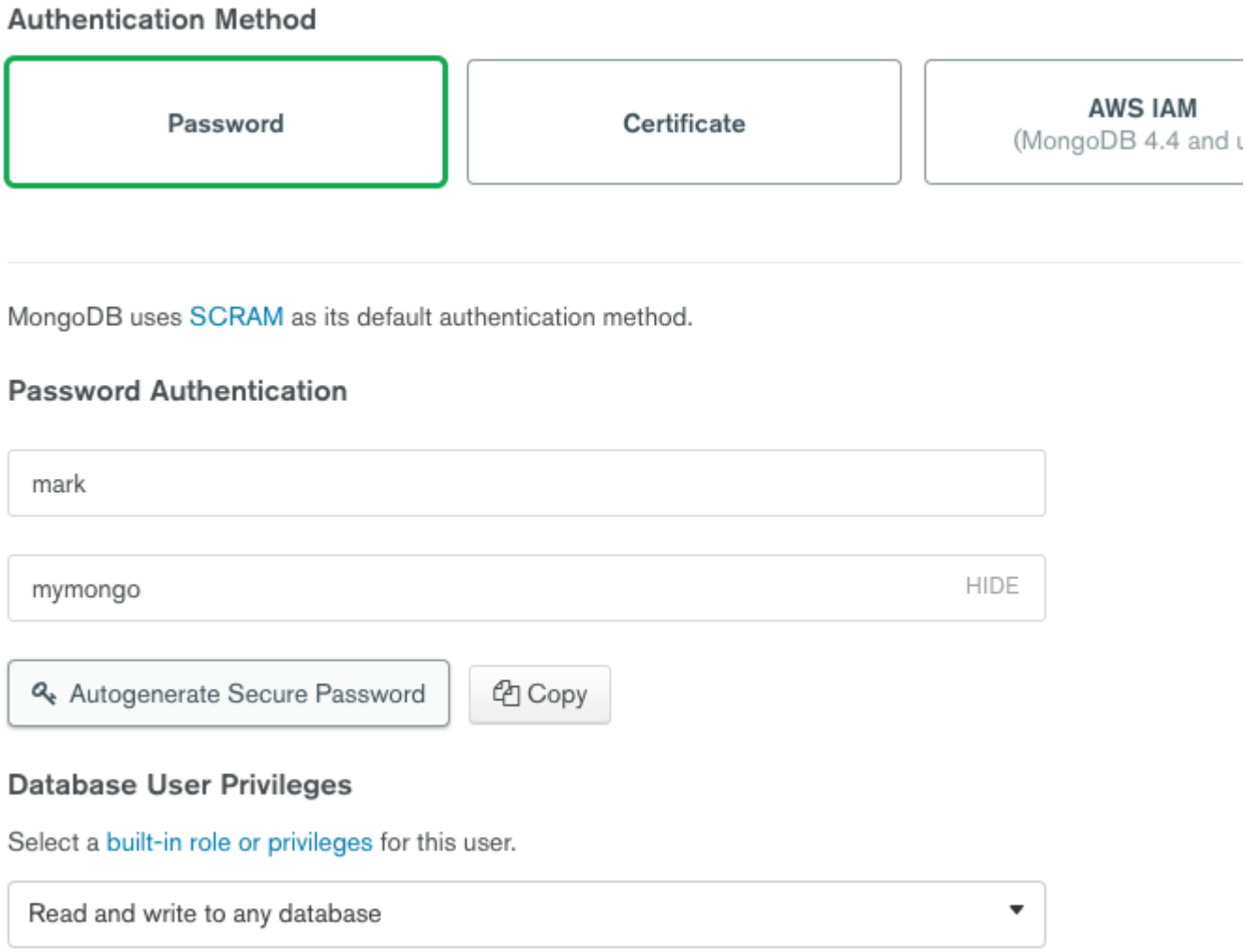
3. Click 'Connect' to connect to your database server. You will be able to see all databases and their collections on the left-hand side. If you make changes to documents outside of Compass (for instance, by making a request to your API), you may have to hit 'Refresh' to see the changes in your database.

Cloud Database with MongoDB Atlas

MongoDB Atlas allows you to create and access a cloud database, which you can use as your production database for your deployed web app.

Setting up MongoDB Atlas

1. Go to **https://cloud.mongodb.com/ (https://cloud.mongodb.com/)** and setup an account. When you are asked which cluster you want to create, select the free 'Starter Cluster'. You can then accept all the default settings for the cloud provider and click 'Create Cluster'.
2. Your cluster will now take some time to set up. In the meantime, you should follow two of the steps from the Getting Started guide that pops up on the Atlas dashboard:
    1. **Create a Database user,** which at the bare minimum you need one of to allow access to read and write to your database, with password authentication. It can look something like this:



    Note that this username/password is **not** the same as the one for your Atlas account. It is for accessing your database outside of Atlas.
    2. **Allow access** from outside servers (such as your web server). For your purposes, you can simply click on 'Allow Access From Anywhere'.

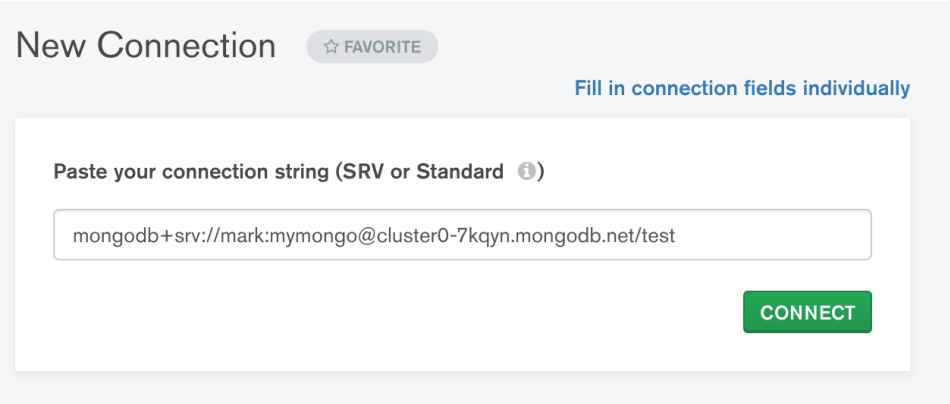
    Next, you can connect to your cluster.

Connecting to your cloud database with MongoDB Compass

Follow these instructions to connect to your database using MongoDB Compass.

1. **Find your connection string on Atlas.** Click on 'Connect' on your Atlas cluster:

   In the pop-up window, select 'Connect with MongoDB Compass'.
   Select **I have MongoDB Compass.**
   This will provide you with a connection string, for example:
   `mongodb+srv://mark:mymongo@cluster0-7kqyn.mongodb.net/test`
   The first part of the string contains the username and password that is used to access your database (the one you set up before). The rest is the location of your database. You will now use this string to connect to your database.
2. Open **MongoDB Compass** and enter your connection string.
3. Insert your connection string for a New Connection. Make sure you replace the password with your password.



4. Once you've done the above steps, click 'Connect'. You should now be able to access your Atlas cloud database on MongoDB Compass in the same way you access your local database.

Linking up your cloud database to your deployed Heroku app

Follow these instructions to link your Atlas database to your deployed app on Heroku. This assumes that you have already successfully **deployed your app on Heroku (https://q.utoronto.ca/courses/250451/pages/deploying-your-app-on-heroku)** .

1. Find where you are connecting to your Mongo database within your Node app. This may be a general connection through MongoClient, or it might be through Mongoose (which is the most likely option if you are making data models the way we do in lecture).
   Remember that you can have multiple databases in your database server. In the example below, we are connecting the **StudentAPI** database on our local database server. We will do the same for the cloud database later on.
   Either way, you must setup your MongoDB URI variable that you use to connect to the database to use a Node environment variable like so:

```
const mongoURI = process.env.MONGODB_URI || 'mongodb://localhost:27017/StudentAPI'
```

This will set the URI variable to the localhost address if developing locally, or to the one on the `MONGODB_URI` environment variable, which we will set to be your cloud database.

2. **Find your connection string on Atlas.** Click on 'Connect' on your Atlas cluster:

In the pop-up window, select '**Connect your Application**'.
Select **Node.js** as the driver, and **version 3.0 or later** for the version.
You will be given a connection string that can be used for your app.
Replace the password with your database user password, and the database name `/test` in the string with the name of the database you want to use, for example, **StudentAPI**.

`mongodb+srv://mark:mymongo@cluster0-7kqyn.mongodb.net/StudentAPI?retryWrites=true&w=majority`

This will be the URI for your database.

3. In the terminal, set the `MONGODB_URI` environment variable on Heroku to be the URI from above (making sure to put the string in single quotes):

```
$ heroku config:set MONGODB_URI='mongodb+srv://mark:mymongo@cluster0-7kqyn.mongodb.net/StudentAPI?retryWrites=true&w=majority'
```

Note: Make sure you have the correct database username and password in the URI (in this case the username is 'mark' and the password is 'mymongo'. Don't leave it as <password> from the original string Atlas gives you).

4. Heroku should then restart your app, and your deployed app will now access your Atlas cluster when making database changes.