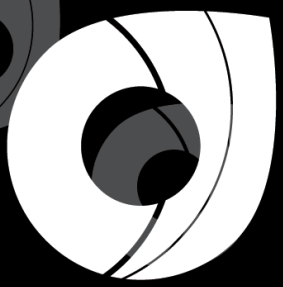
Corndel DevOps Engineering Programme

in association with Softwire

Module 9: Project Exercise



Corndel Digital.

DB1P

Module 9

Project Exercise Brief

In this exercise we will finally say farewell to Trello. While it has been a useful tool so far, it is limiting: we have to follow Trello's data model; use their API and generally be restricted to what they make available. It is time for us to remove Trello and replace it with something more under our control.

You will create a MongoDB cluster and configure your application to use it instead of Trello. We'll be using an open-source NoSQL database called **MongoDB**. The way MongoDB stores data is *comparable* to Trello (see table below). This should make the transition from Trello to MongoDB easier than transition to other databases might be.

Trello	MongoDB
`Board`	`Database`
`List`	`Collection`
`Card`	`JSON object`

Part 1: Run a MongoDB database

Step 1: MongoDB Atlas

For this exercise we are going to use a service called MongoDB Atlas. This will let us create a MongoDB cluster that our application can use. There is a free tier, which is suitable for our purposes. If you choose the "I'm learning MongoDB" option at sign-up then the set-up instructions are very intuitive. Start the sign-up process here and refer to the below for guidance.

Cloud & Region

Atlas can be backed by AWS, Azure or Google Cloud. It doesn't matter which cloud you choose; however the free-tier isn't available in all locations. Make sure you choose a location near you with a free-tier available. The the UK, we suggest one of the following:

Cloud	Free-Tier Region	Location
`AWS`	`eu-west-1`	Ireland
`Google Cloud`	`europe-west1`	Belgium
`Azure`	`northeurope`	Ireland

> *You can only have one instance of the free* `M0 Sandbox` *cluster per account. If you're already using that allowance for something else you may need to use a paid tier or create a new account.*

Security

When creating a database user you will be offered two authentication methods: `username and password` and `X.509 certificate`. You should choose the former. Make a note of the username and password, you'll need these to connect to the database.

Atlas restricts access to its clusters based on IP address. Make sure you add your local IP address when prompted (there's a button to do this automatically) so you can access the cluster from your local machine.

Getting Connected

Finally, Atlas will provide some basic guides on how to get connected to your cluster. The "Connect your application" option will provide sample code for connecting your application in multiple languages. You can also, optionally, explore the other options to help get a better understanding of how to interact with MongoDB.

> *It can take a few minutes for your cluster to spin up. If you got through the set-up quickly you may need to wait for it to finish initialising.*

Step 2: Installing Pymongo

Python support for MongoDB comes in the form of PyMongo. You can add this dependency to your project with `poetry` (or if your project uses `pip` to manage dependencies, you can use `pip install` instead of `poetry add`). Connecting to MongoDB Atlas has an additional dependency, so we also need to add `pymongo[srv]`:

```
poetry add pymongo pymongo[srv]
```

Once installed you should be able to interact with MongoDB from the Python shell in Visual Studio Code.

- Open the VSC terminal.
- Type `python` to open the Python shell
- Type `import pymongo`

If this does not produce an error then you have successfully installed PyMongo.

Step 3: Connecting to Atlas

Assuming you've completed steps 1 and 2 (and allowed sufficient time for your cluster to spin up) you are now ready to try connecting to your Atlas cluster. You will need some details from the set-up you did in step 1.

Parameter	Where to find it?
`USER_NAME`	Created in step 1. A list is visible in the "Database Access" menu under the "Security" heading.
`PASSWORD`	Created in step 1. If lost you'll have to to change the password in the "Database Access" menu.
`MONGO_URL`	Visible in the "Connect" menu of your cluster. Look for a URL that ends `.mongodb.net`
`DEFAULT_DATABASE`	The name of the default database: you choose what this is.

Once you have these you can run the following lines of code in the Python shell. The result should be something similar to the last line. If you run into problems check Atlas' troubleshooting Documentation.

```
    >>> import pymongo
    >>> client = pymongo.MongoClient("mongodb+srv://
<USER_NAME>:<PASSWORD>@<MONGO_URL>/<DEFAULT_DATABASE>?w=majority")
    >>> client.list_database_names()
    ['admin', 'local']
```

Assuming you've connected successfully you should take some time to explore the functionality of MongoDB. The pymongo tutorial is a good place to start (hint: by this point, you've already done "Prerequisites" and "Making a Connection with MongoClient"). Try out the commands in the tutorial to build up your understanding of MongoDB and PyMongo. Make sure you are able to do the following:

- Add an item to a collection
- Read the contents of a collection - including filtering by properties
- Remove an item from a collection

Part 2: Replacing Trello

Now that you've done the above, we:

- know we have a working MongoDB Atlas cluster
- know how to talk to it from Python (i.e. add items, read items, etc.)

It is time to tell our application how to use MongoDB instead of Trello.

Step 1: Updating the code

We've chosen MongoDB as it has some parallels with Trello's data model, e.g. you can treat MongoDB's **collections** in a very similar manner to Trello's **lists**. This should make it relatively straightforward to copy the data access code you've written for Trello and rewrite it to target MongoDB instead.

Not everything is totally like for like; you will have to make some changes. For example, MongoDB doesn't automatically set a `dateLastActivity` property like Trello does, you'll have to keep track of this separately. There may be other issues, depending on how you implemented your app.

This will likely be the longest part of the exercise. Make small, incremental changes and test often. Good luck.

Step 2: Update CI/CD

Once you've finished the code changes you'll need to update the CI/CD pipeline. At the very least you will need to provide the new environment variables required to connect to Atlas. (See Part 1, Step 3).

As Atlas only allows a single free cluster we will reuse the same cluster for the deployed environment. We wouldn't usually do this, but don't worry, we'll be fixing this in a future module.

> *Atlas is IP restricted by default. Change your cluster's `Network Access` settings to allow access from anywhere. This is **not** recommended best practice, but an acceptable risk given this is a tutorial exercise.*

Step 3: Tidy up

Last of all we need to tidy up. The code relating to Trello is no longer needed so it should be removed. It'll still be present in the git history if we ever need to go back to it.

Go over your new data access code to make sure it is **clean**.

Before you finish make sure your local build is still working; the CI/CD pipeline is still green; and that deployed environment is still available.

Stretch Goals

(Stretch goal) Part 3: Use docker for local dev

In Part 2, Step 2 we acknowledged we'd be using the same cluster in both of our environments. This isn't great.

One way around this is to use docker to run an instance of MongoDB for local dev. There are MongoDB images available on Docker Hub.

You can tell docker compose to create a MongoDB container using the `image` keyword. (If you haven't set up docker compose yet, this is a good opportunity to do so.)

(Stretch goal) Part 4: Migrate your existing data

While you now have a nice, new, shiny MongoDB database it doesn't have any of your old To-dos in it. Write some code to automate the migration of your data from Trello to MongoDB. You could either write this as a separate script or include it in your application code.

However you decide to approach the problem your solution should be idempotent and robust.

(Stretch goal) Part 5: IP Restrictions

In Part 2, Step 2 we turned off the IP Restrictions on Atlas Cluster so we could access it from our Heroku app. To fix this you will need to use an add-on to provide a static outbound IP address for your app. You can then restrict incoming connections to just that IP address in Atlas.

> *While some Heroku add-ons have free tiers, they are limited. You might want only want to implement your solution to Part 5 temporarily to avoid a broken app or having to pay.*