the Master Course

{CODENATION}

Backend Development

Intro to Databases

{CODENATION}

Learning Objectives

To know what a database is.

To create a remote database using MongoDB.

To be able to use Environmental Variables to protect important data.

What is a database?

A database is quite simply a computerised system that stores information.

{CODENATION}

What is a database?

Databases can hold structured or non-structured data.

Think of a database like a filing cabinet, it could hold information on employees, all with the same structured information in alphabetical order.

Or it could hold the mail you've received in a year, all with different information with no real order to it.

In order to interact with a database, we need to use something called a Database Management System or DBMS.

What is a DBMS?

A DBMS is essentially a way for us to manipulate the contents of a database.

A DBMS will provide a variety of ways to manage the data in our database.

Local vs Cloud-Hosted?

We have two main options with databases, we can either host them on our Local Machine or use a Cloud Hosted option.

Local-Hosted Database

A local database is stored on our personal computer and only accessible via a direct connection to our hardware.

The speed and efficiency of this database is governed by the quality of our hardware.

Cloud Hosted Database

A cloud hosted database is available through the internet, it is hosted on a collection of servers elsewhere, and isn’t limited by the speed of our hardware, rather it is governed by the speed of both our internet and the cloud provider.

There are many different DBMS's to choose from.

We are going to use MongoDB.

MongoDB is a NoSQL database, which allows us to store structured or non-structured data.

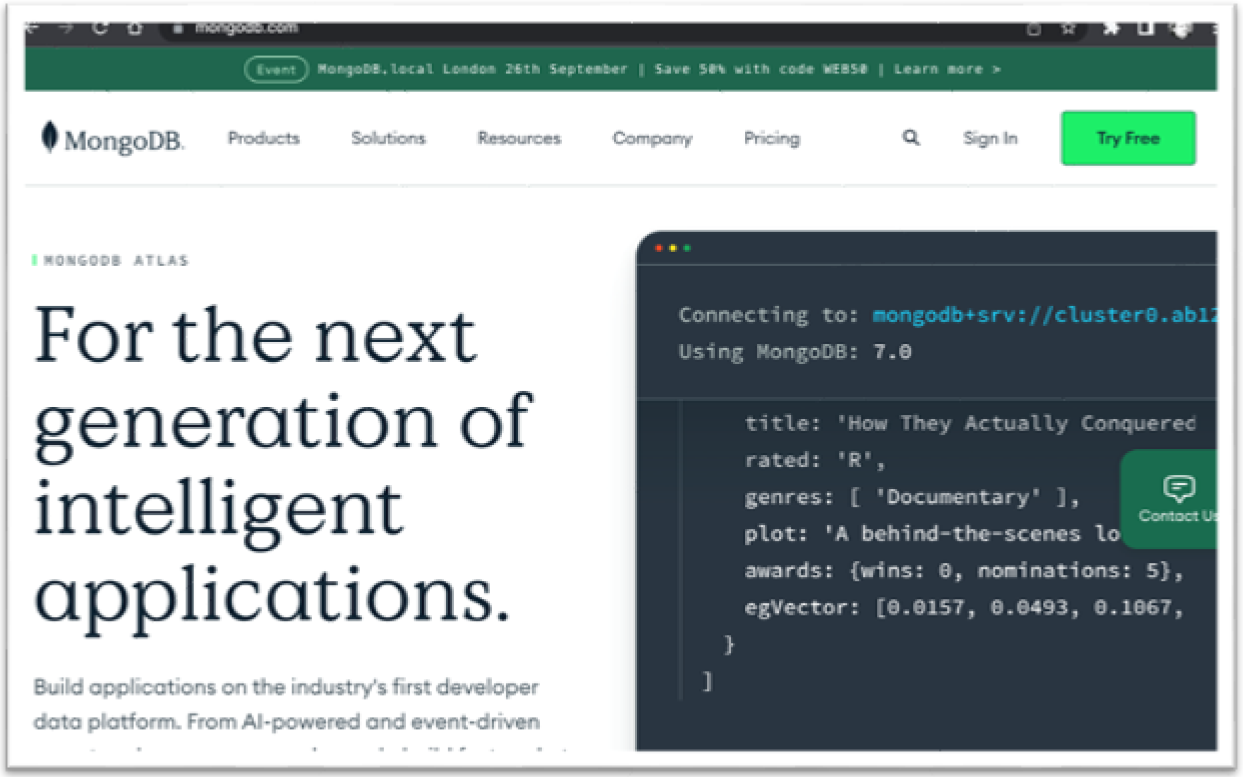
Quick Recap from yesterday…

What were the **CRUD** operations?

How does **CRUD** relate to the **HTTP Verbs** used in a server?

{CODENATION}

https://www.mongodb.com



For the next generation of intelligent applications.

Build applications on the industry's first developer data platform. From AI-powered and event-driven

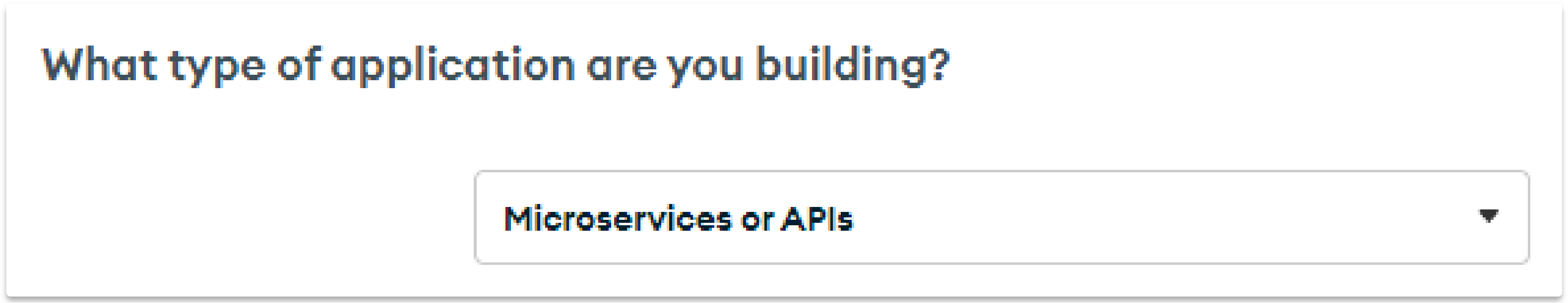
```
Connecting to: mongodb+srv://cluster0.ab12
Using MongoDB: 7.0

    title: 'How They Actually Conquered
    rated: 'R',
    genres: [ 'Documentary' ],
    plot: 'A behind-the-scenes lo
    awards: {wins: 0, nominations: 5},
    egVector: [0.0157, 0.0493, 0.1067,
  }
]
```

Please visit this site and setup an account.
It's FREE!

mongoDB®

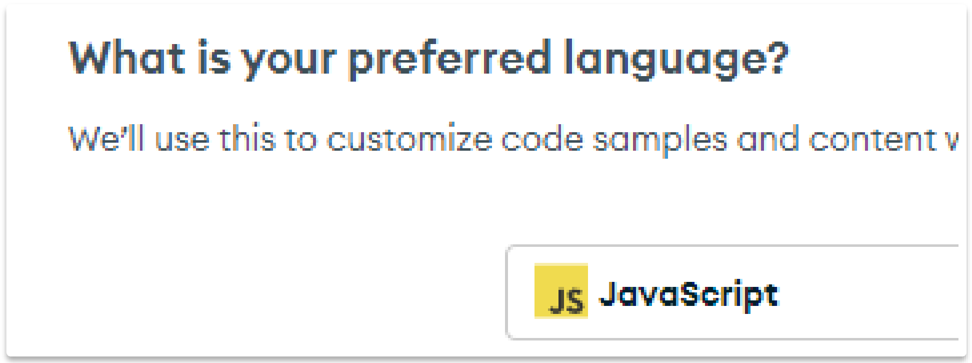
https://www.mongodb.com

Here are some of the options to select from the signup process…



What type of application are you building?

Microservices or APIs



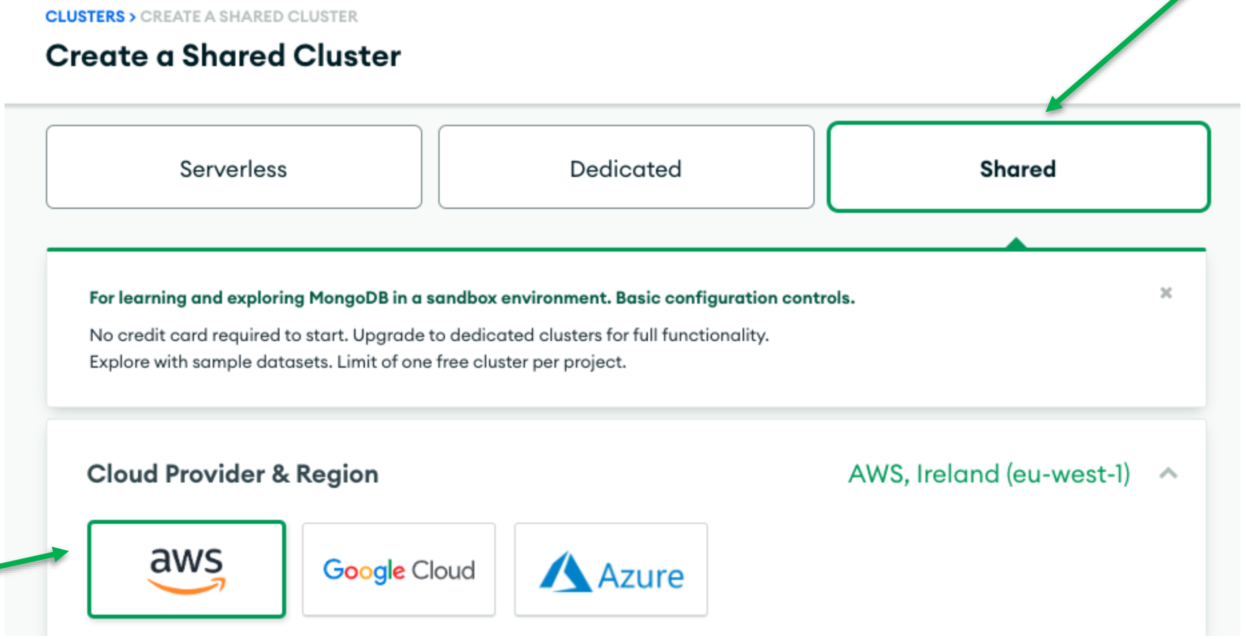
What is your preferred language?

We'll use this to customize code samples and content v

JavaScript

{CODENATION}

Select Shared cluster


CLUSTERS > CREATE A SHARED CLUSTER

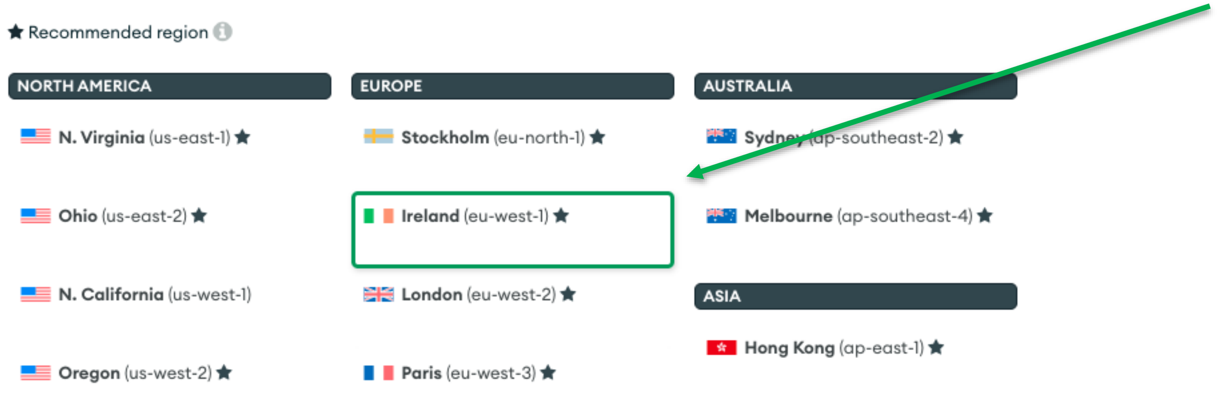
Create a Shared Cluster

For learning and exploring MongoDB in a sandbox environment. Basic configuration controls.

No credit card required to start. Upgrade to dedicated clusters for full functionality.
Explore with sample datasets. Limit of one free cluster per project.

Cloud Provider & Region: AWS, Ireland (eu-west-1)

Select AWS for the cloud provider

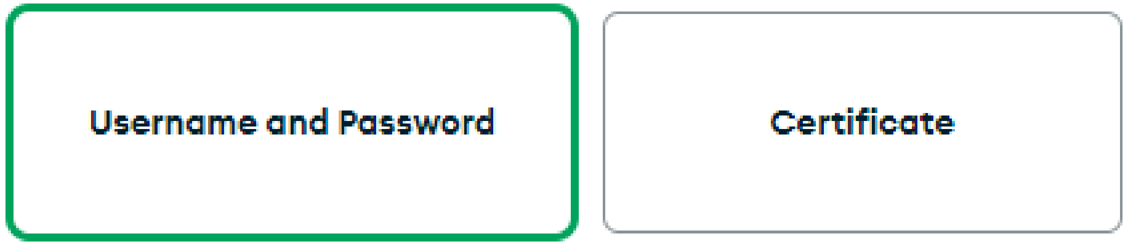


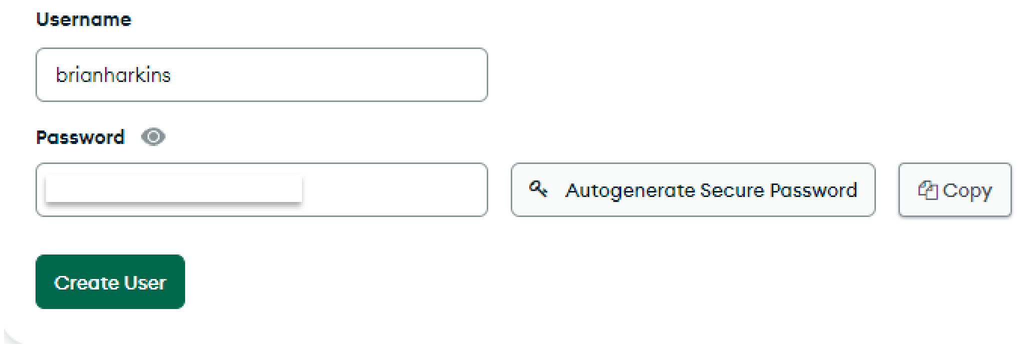
★ Recommended region

NORTH AMERICA	EUROPE	AUSTRALIA
N. Virginia (us-east-1) ★	Stockholm (eu-north-1) ★	Sydney (ap-southeast-2) ★
Ohio (us-east-2) ★	Ireland (eu-west-1) ★	Melbourne (ap-southeast-4) ★
N. California (us-west-1)	London (eu-west-2) ★	**ASIA**
Oregon (us-west-2) ★	Paris (eu-west-3) ★	Hong Kong (ap-east-1) ★

Select Ireland as our nearest loction

Select this to authenticate!



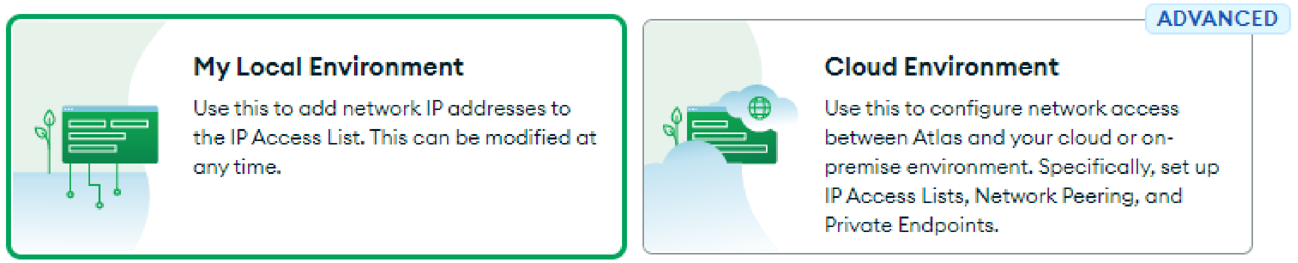

Username

brianharkins

Password

Write your Username & Password down safely!

Select ' My Local Environment'



My Local Environment

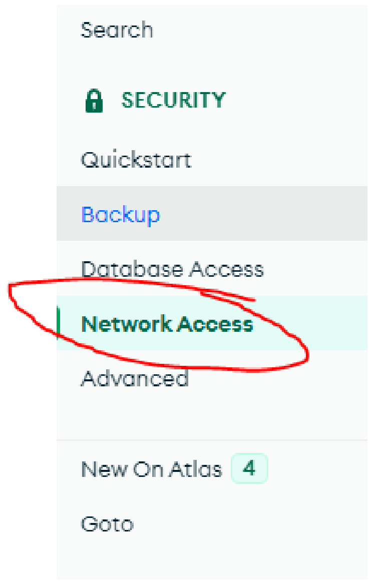
Use this to add network IP addresses to the IP Access List. This can be modified at any time.

ADVANCED

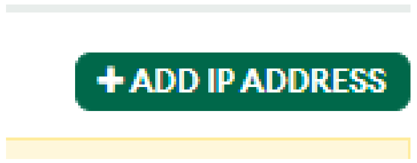
Cloud Environment

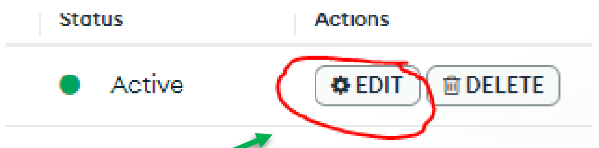
Use this to configure network access between Atlas and your cloud or on-premise environment. Specifically, set up IP Access Lists, Network Peering, and Private Endpoints.

Let's setup your network access...

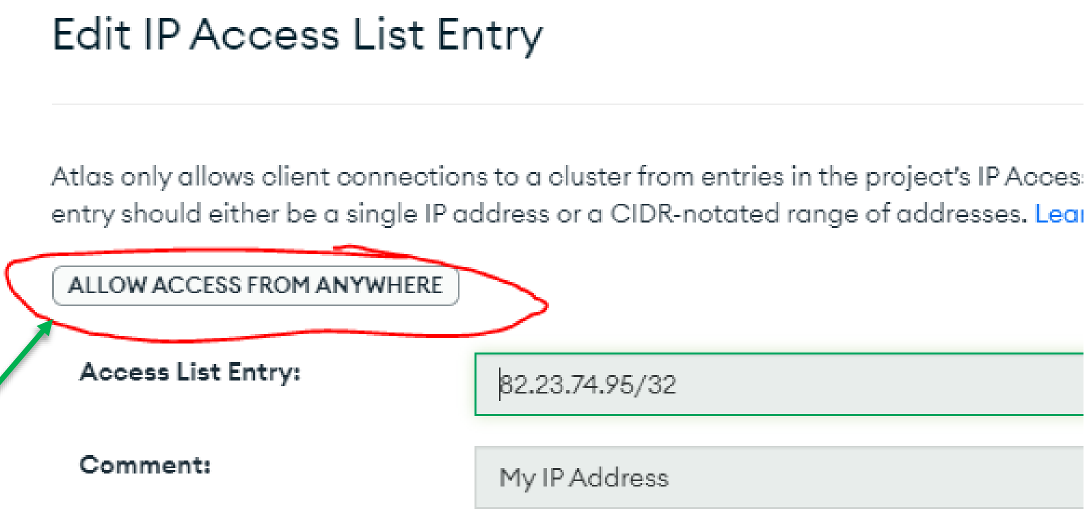


Click on the 'Add IP Address' button





Click EDIT


Edit IP Access List Entry

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

ALLOW ACCESS FROM ANYWHERE

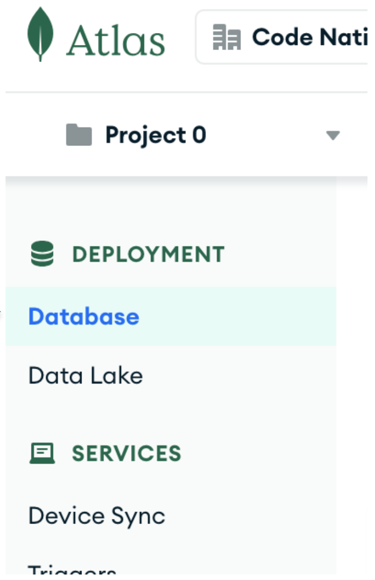
Access List Entry: 82.23.74.95/32

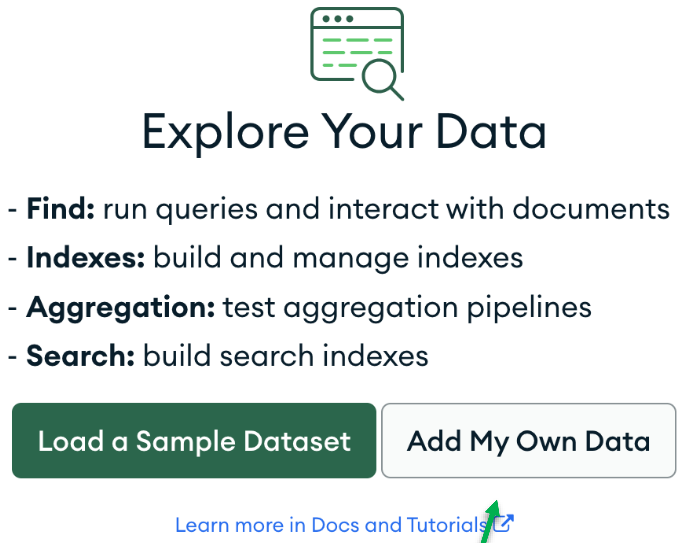
Comment: My IP Address

Click 'Allow Access from Anywhere

If you boot your PC then the likelihood is that you will get a different IP address.

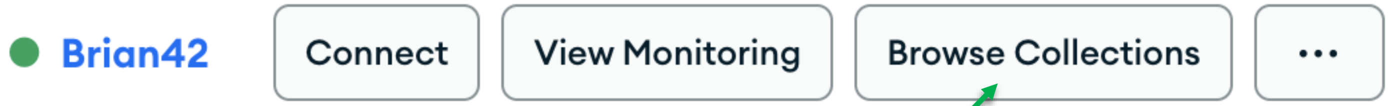
Select 'Database'




Explore Your Data

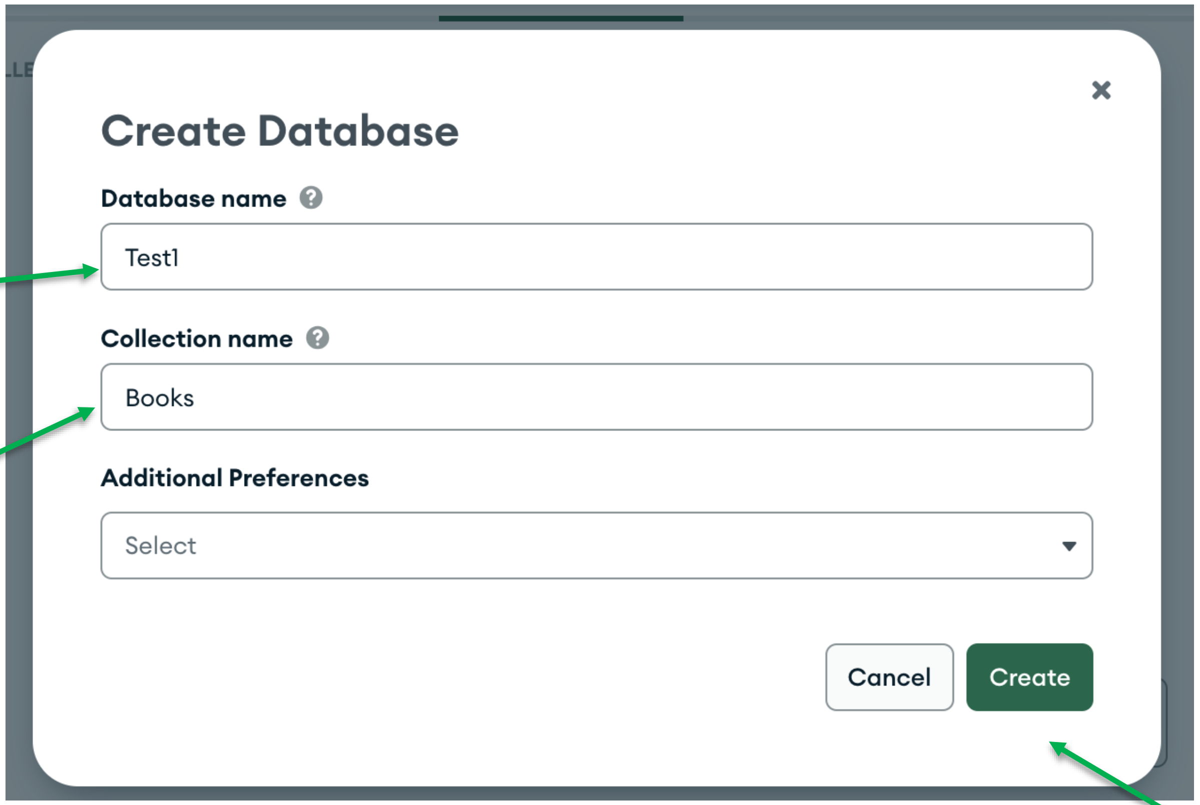
- **Find:** run queries and interact with documents
- **Indexes:** build and manage indexes
- **Aggregation:** test aggregation pipelines
- **Search:** build search indexes

Then click on 'Add My Own Data'



Click on 'Browse Collections'

Collections are tables of data but there aren't any here at the moment.



Create Database

Database name: Test1

Collection name: Books

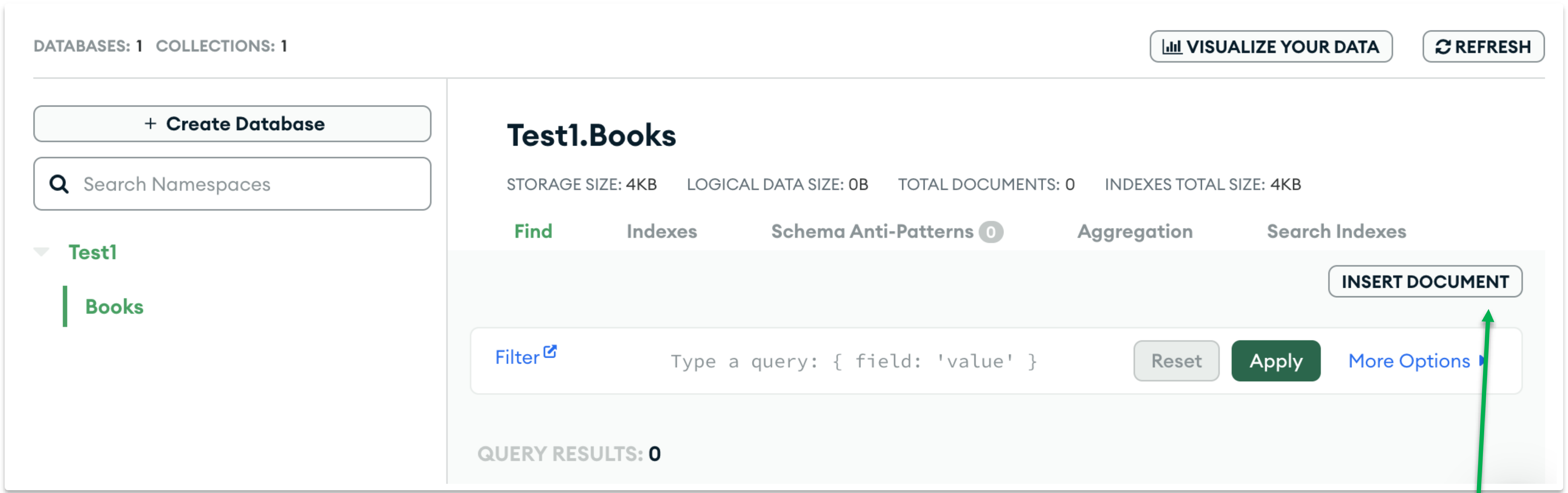
Additional Preferences: Select

Enter a name, e.g., Test1

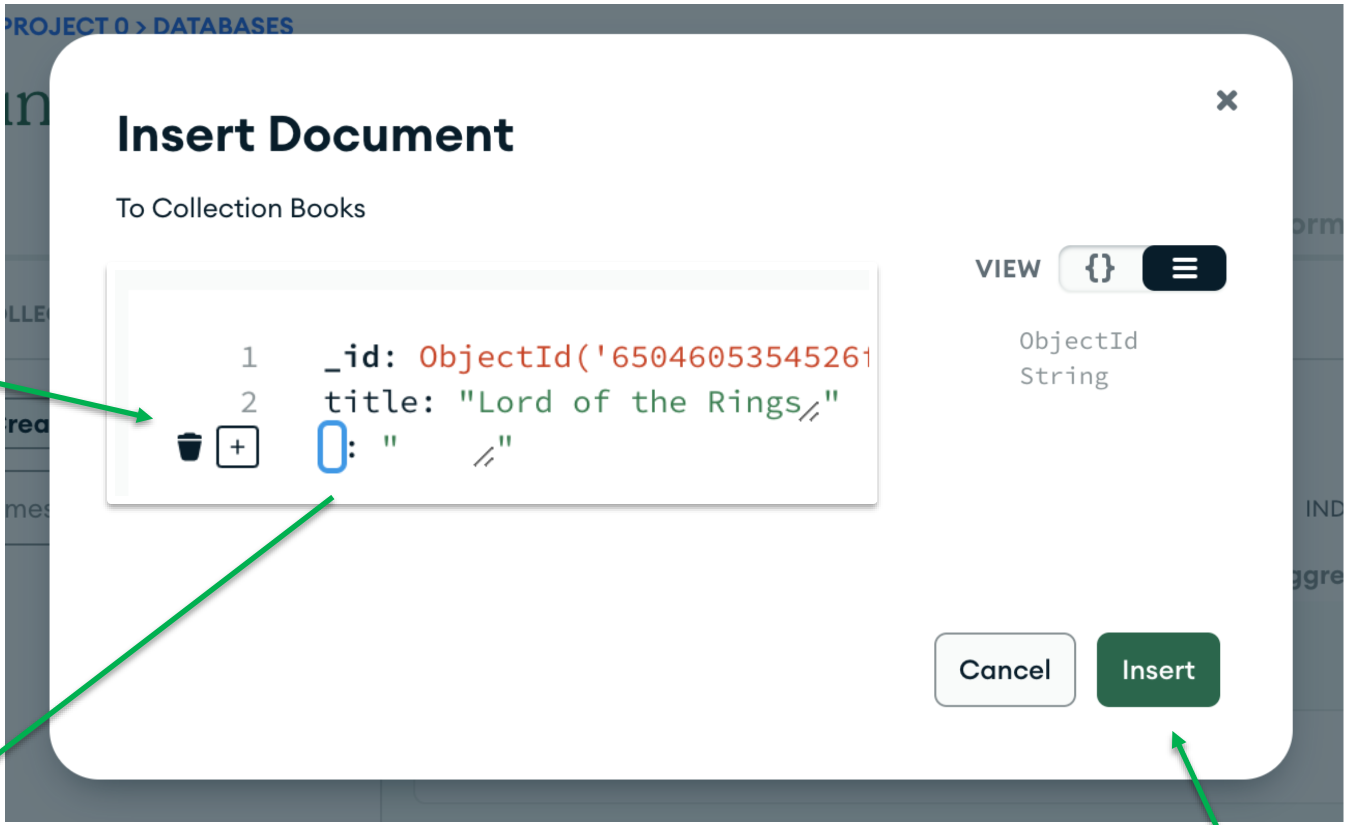
Enter a collection name, e.g., Books

Then click on 'Create'

This will help us to create a new Database with collection.



Click on the 'INSERT DOCUMENT' button.


PROJECT 0 > DATABASES

Insert Document

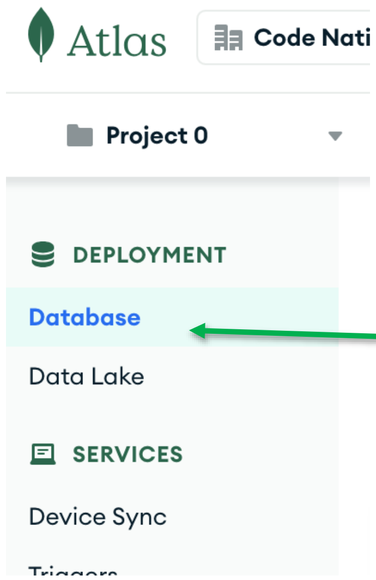
To Collection Books

```
_id: ObjectId('65046053545261
title: "Lord of the Rings"
: ""
```

Click here and enter some fields like the ones we have used earlier…

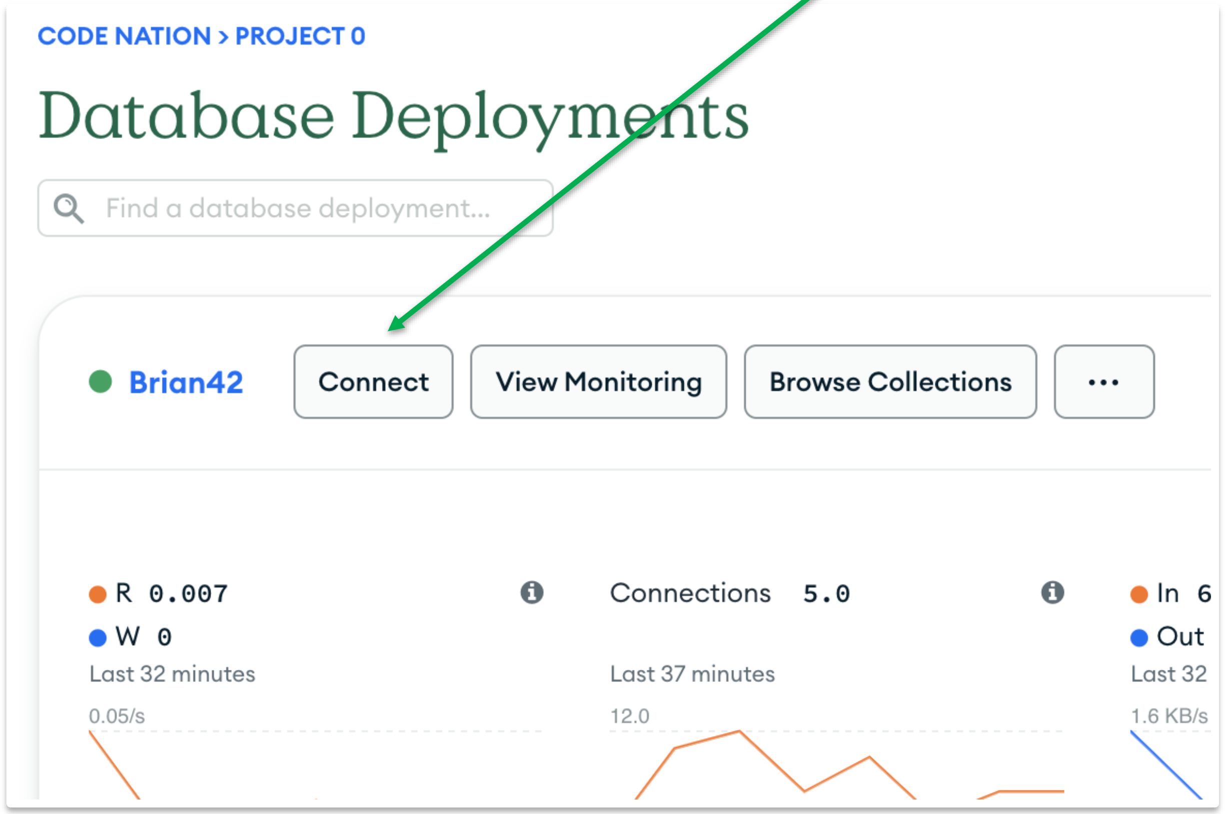
```
_id: ObjectId('6504605354526f05149e5b88')
title: "Lord of the Rings"
author: "J.R.R. Tolkein"
genre: "Fantasy"
```

When ready, click on the 'INSERT' button.

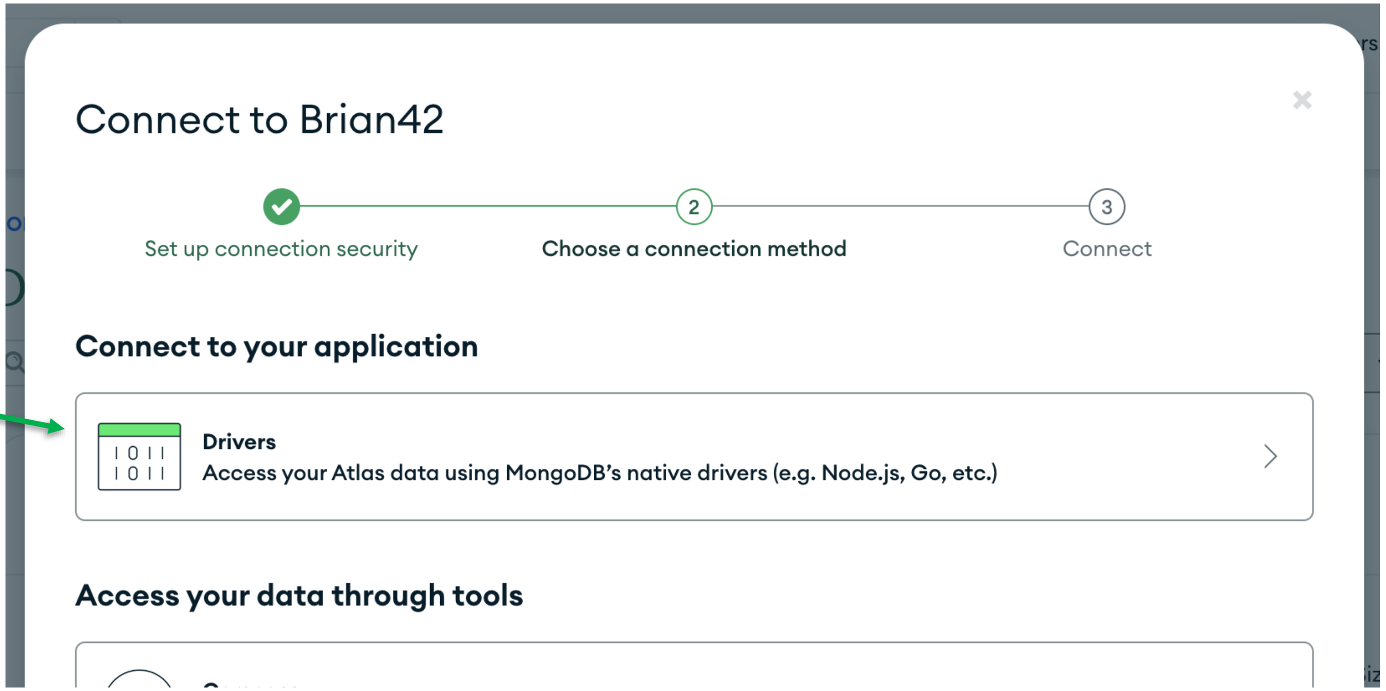


Select ‘Database’, again.

Press on the ‘Connect’ button.



Select ‘Drivers’.



Connect to Brian42

Set up connection security

Choose a connection method

Connect

Connect to your application

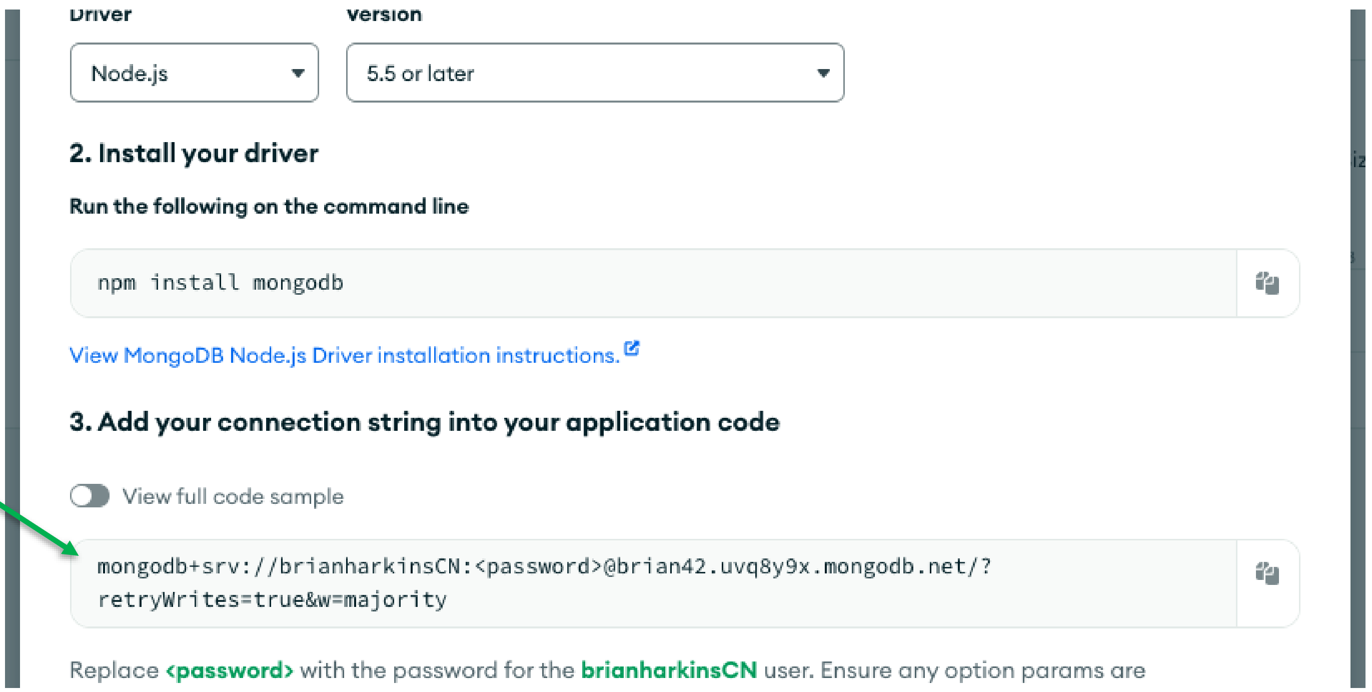
Drivers

Access your Atlas data using MongoDB’s native drivers (e.g. Node.js, Go, etc.)

Access your data through tools

This will allow us to access our data through Node.js

In part 3, we can see our connection string with username and password placeholders.


Driver: Node.js

Version: 5.5 or later

2. Install your driver

Run the following on the command line

```
npm install mongodb
```

View MongoDB Node.js Driver installation instructions.

3. Add your connection string into your application code

```
mongodb+srv://brianharkinsCN:<password>@brian42.uvq8y9x.mongodb.net/?
retryWrites=true&w=majority
```

Replace **<password>** with the password for the **brianharkinsCN** user. Ensure any option params are

You will need this connection string in your server code, later on …

Well Done!

New Server

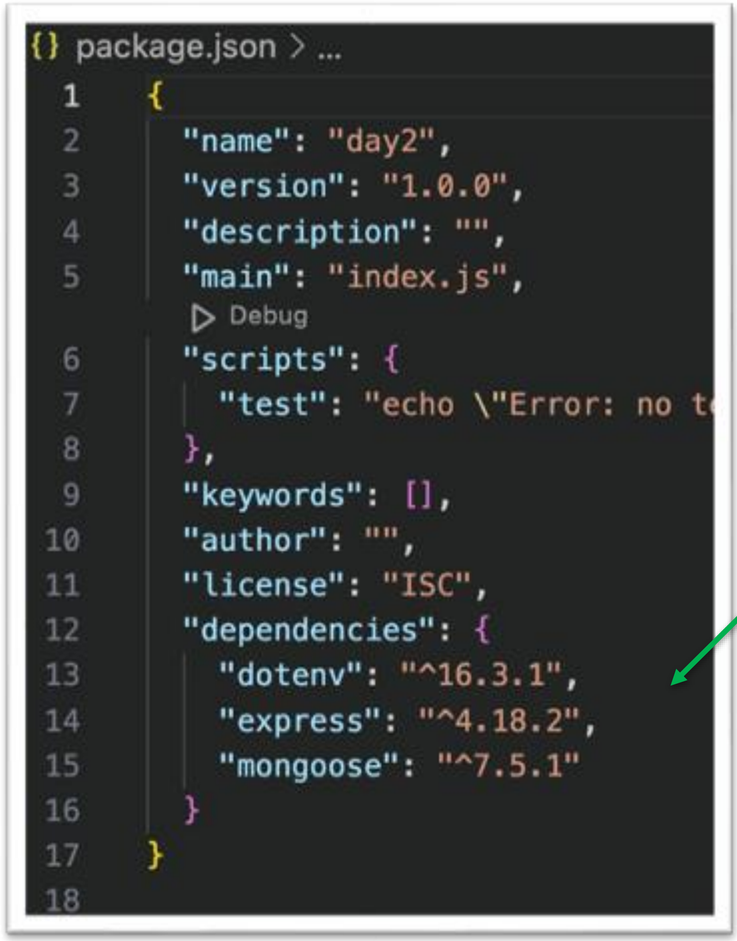
{CODENATION}

Let’s setup a new server…

1. Create a new project folder
2. Setup the new project for npm (npm init –y)
3. Install Express..js. (npm install express)
4. Install Mongoose. (npm install mongoose)
5. Install Environmental Variables (npm install dotenv)

We will delve into 4 & 5 soon ….

{CODENATION}



{} package.json > ...

```
{
  "name": "day2",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no t
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "mongoose": "^7.5.1"
  }
}
```

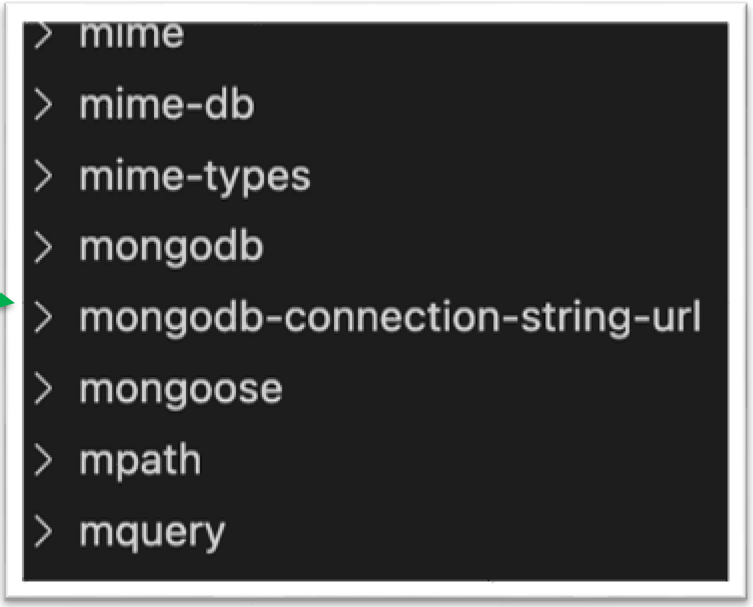
In the package.json file, we now have the express, mongoose and dotenv dependencies.

{CODENATION}

mongoose

In the node_modules folder, we can see the mongoDb folders.

We didn't have to install MongoDb separately.



```
> mime
> mime-db
> mime-types
> mongodb
> mongodb-connection-string-url
> mongoose
> mpath
> mquery
```

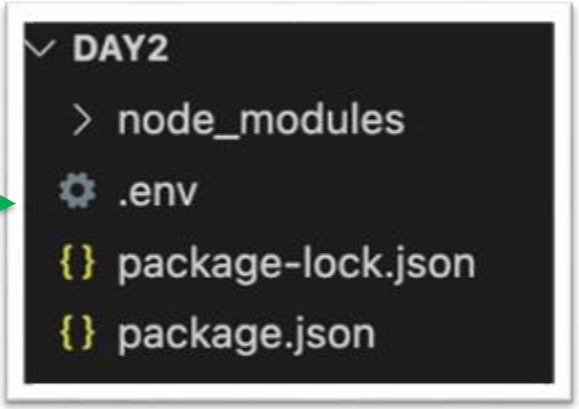
Environmental Variables

The dotenv package is a safe way to keep passwords, API keys, and other sensitive data out of your code – especially when using source control like GitHub.com.

It allows you to create environment variables in a . env file instead of putting them in your code.

Environmental Variables

Add a new file called **.env**

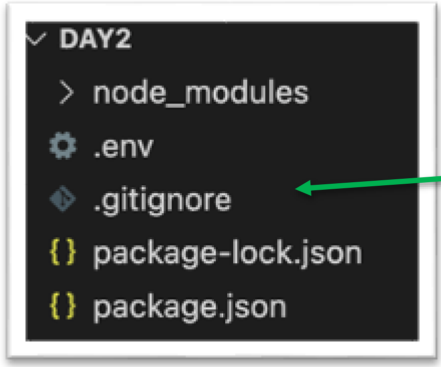


```
DAY2
  > node_modules
    .env
    package-lock.json
    package.json
```

This is the place where we store what we call Environment Variables. Things like userId, passwords, any specific variable that we might want to keep secret.

We will be putting things in here later.

.gitignore folder

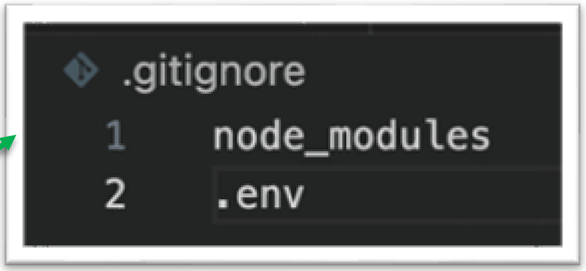


```
DAY2
> node_modules
.env
.gitignore
{} package-lock.json
{} package.json
```

Add a new file called **.gitignore**

As well as the important .env file, we don't want the **node_modules** folder to get pushed up to GitHub as it is too big.

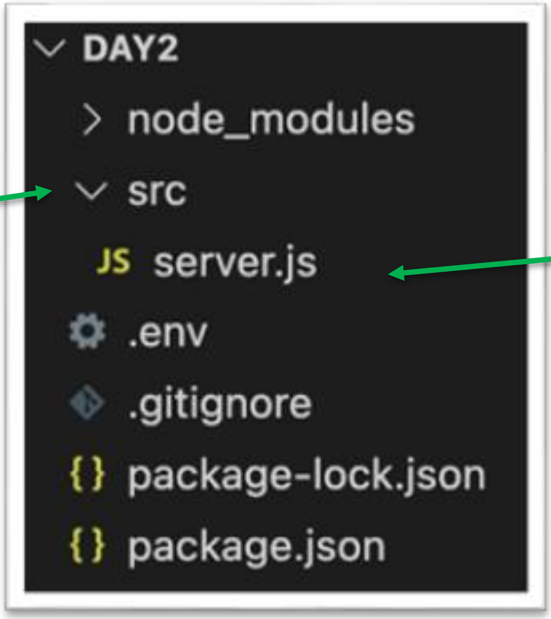
Add node_modules and .env to the .gitignore file.



```
.gitignore
node_modules
.env
```

Mongoose in our server

Mongoose is a Node. js-based Object Data Modeling (ODM) library for MongoDB. It allows us to talk to a database server somewhere else.

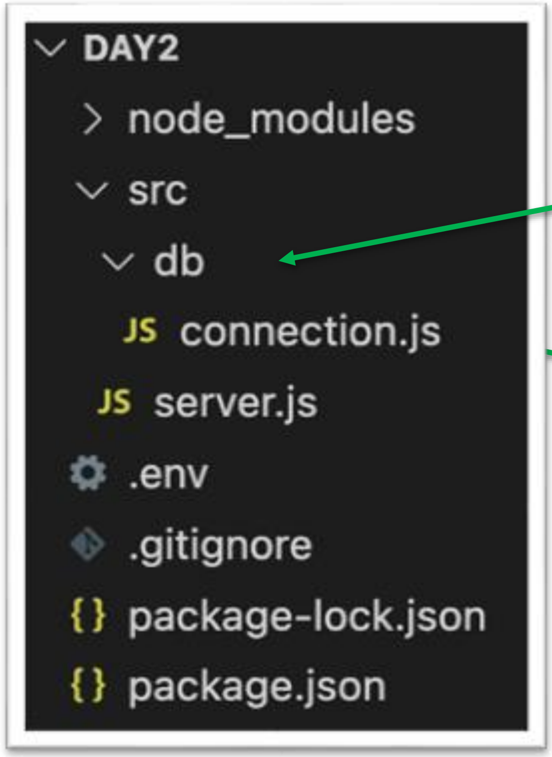


```
DAY2
  node_modules
  src
    server.js
  .env
  .gitignore
  package-lock.json
  package.json
```

Add a new folder called 'src'

Inside src, add a new file called server.js

Mongoose in our server



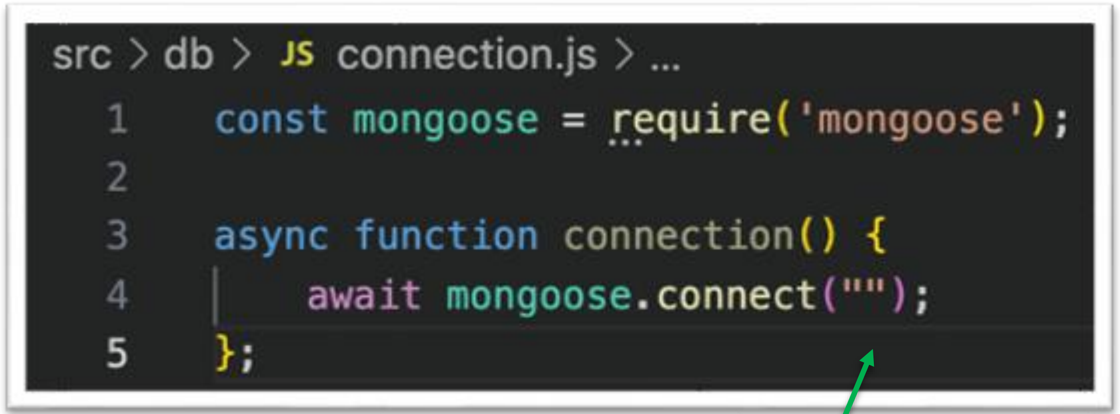
```
DAY2
  node_modules
  src
    db
      connection.js
    server.js
  .env
  .gitignore
  package-lock.json
  package.json
```

Add a new folder called ‘db’

Inside db, add a new file called ‘connection.js’

Mongoose in our server

In the new connection.js file, add the following code to prepare our connection to the remote database …

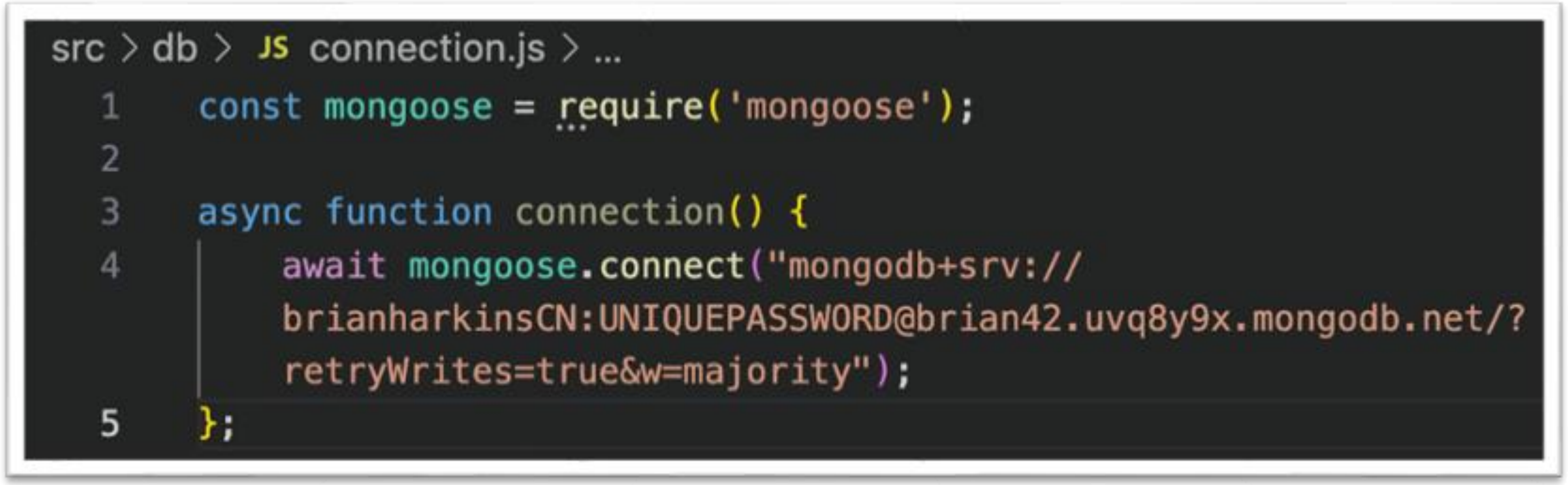


```
src > db > JS connection.js > ...
const mongoose = require('mongoose');

async function connection() {
    await mongoose.connect("");
};
```

We don't know how long it will take to connect with the database and so we will have to make it an asynchronous function.

We will paste our connection string from MongoDB in here…

Mongoose in our server



src > db > JS connection.js > ...

```javascript
const mongoose = require('mongoose');

async function connection() {
    await mongoose.connect("mongodb+srv://
    brianharkinsCN:UNIQUEPASSWORD@brian42.uvq8y9x.mongodb.net/?
    retryWrites=true&w=majority");
};
```

Use the MongoDb website connectionString and paste it into here. Change username and <password>. Your info can be found in the MongoDB Atlas website.

If there was an error in our connection string, or code, how could we check for errors?

{CODENATION}

Try Catch Block

```
Try {
   //…
} catch (e) {
   //…
}
```

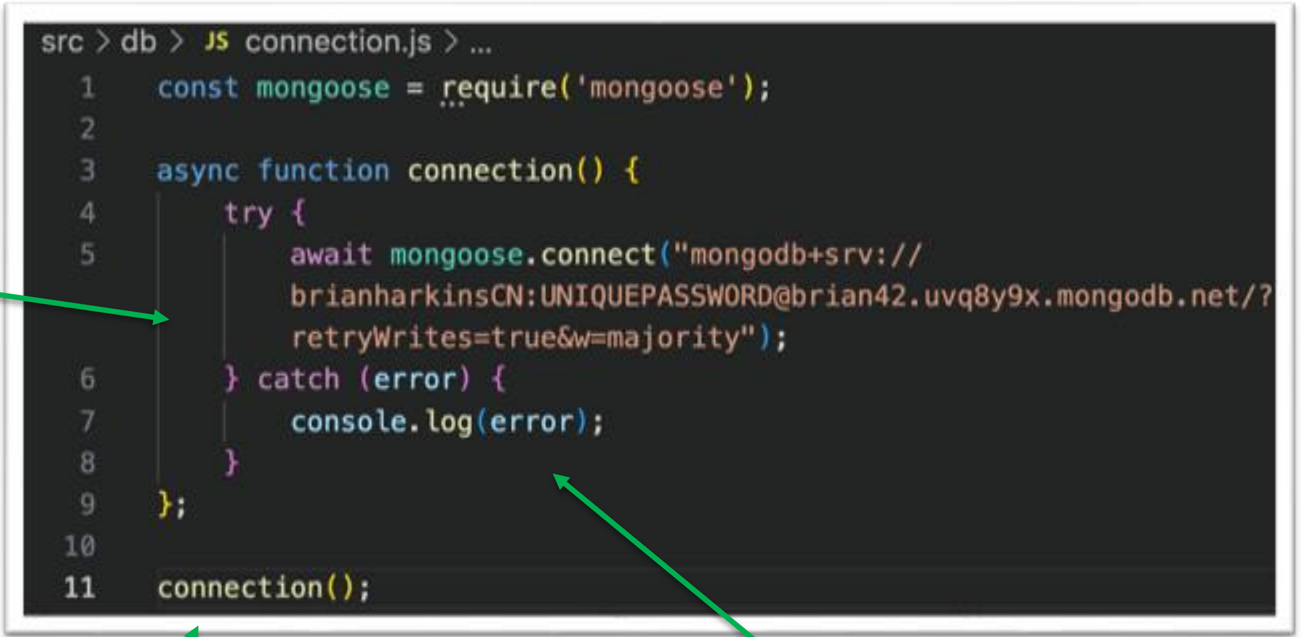
A try statement lets you test a block of code for errors.

A catch statement lets you handle that error.

Let's apply this to our connection.js code…

Mongoose in our server

Try this code



src > db > JS connection.js > ...

```
const mongoose = require('mongoose');

async function connection() {
    try {
        await mongoose.connect("mongodb+srv://
        brianharkinsCN:UNIQUEPASSWORD@brian42.uvq8y9x.mongodb.net/?
        retryWrites=true&w=majority");
    } catch (error) {
        console.log(error);
    }
};

connection();
```

Note: If you forget your password, then you can go to the MongoDb.com page:

Security → Database Access → EDIT

and change the password.

Call the function here.

Catch and report the error

CODE ALONG

CODENATION

Let’s test it …

We normally run src/server.js but we will independently test this function by running the following line in Terminal:

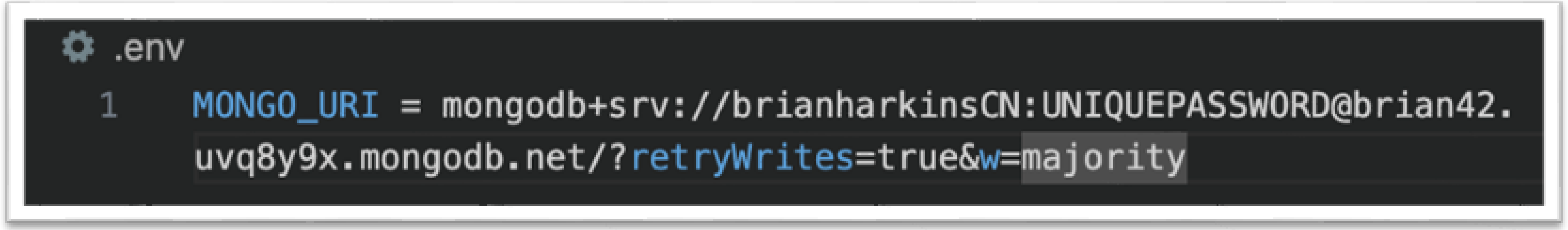
```
node src/db/connection.js
```

{CODENATION}

Environmental Variables

This connection string in our code is too revealing. It was good to test it but now we want to put this whole string into the **.env** file.

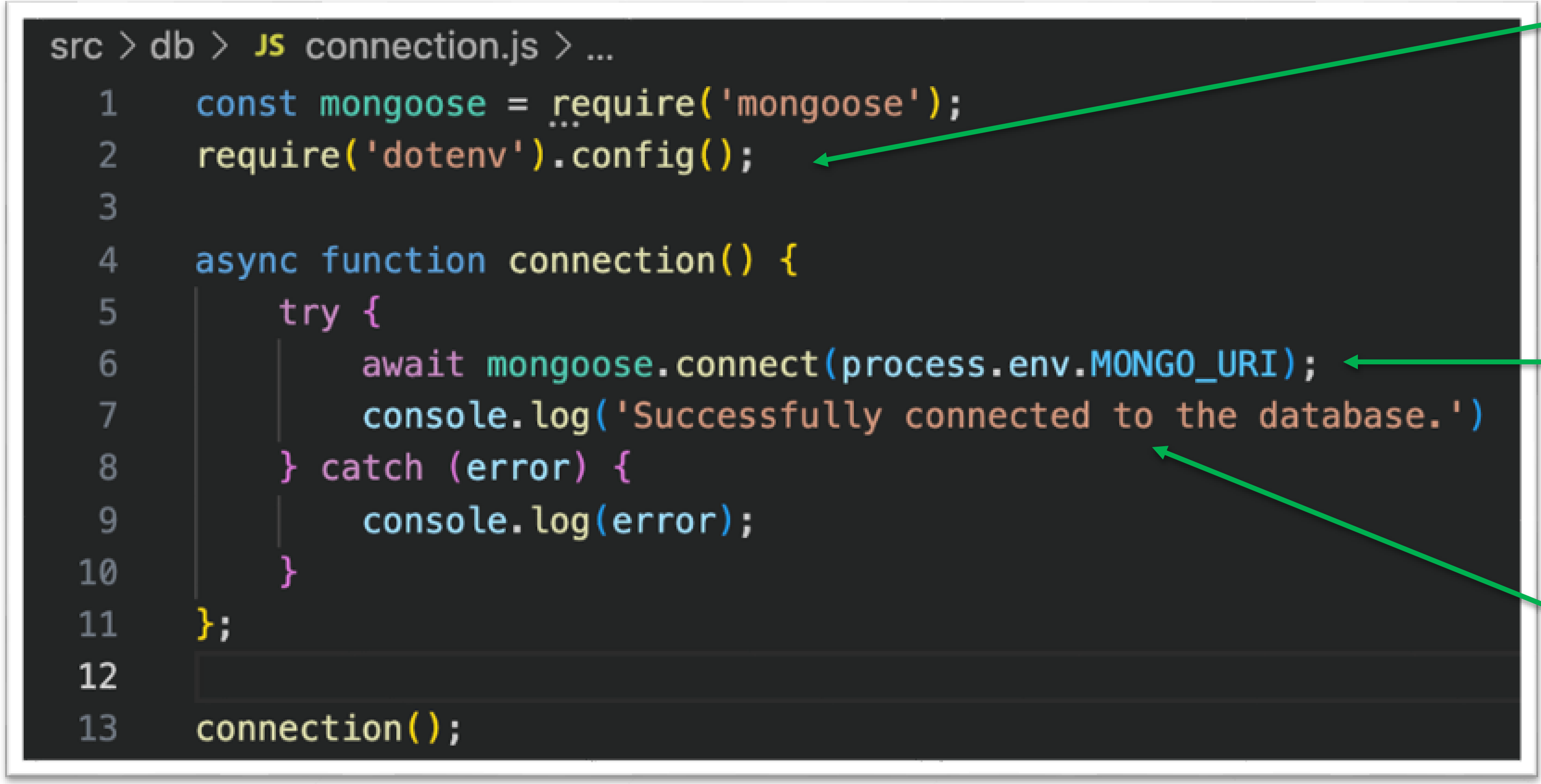
In our **.env** file create a variable to hold this string …



```
.env
MONGO_URI = mongodb+srv://brianharkinsCN:UNIQUEPASSWORD@brian42.
uvq8y9x.mongodb.net/?retryWrites=true&w=majority
```

We can now reference this MONGO_URI in our code …

Environmental Variables



src > db > JS connection.js > ...

```javascript
const mongoose = require('mongoose');
require('dotenv').config();

async function connection() {
    try {
        await mongoose.connect(process.env.MONGO_URI);
        console.log('Successfully connected to the database.')
    } catch (error) {
        console.log(error);
    }
};

connection();
```

Prepare dotenv for use

Reference the new env. var.

Add a console log.

Essentially, we have injected the .env file variable into this connection

We have successfully connected to our own database.

Next Steps…

Work with our database, etc

{CODENATION}

Learning Objectives

To know what a database is.

To create a remote database using MongoDB.

To be able to use Environmental Variables to protect important data.