

# Backend Pre Requisite Setup

## BE1.1

setting up mongoDB with mongoDB atlas and replit

ex01: create your mongoDB atlas account

challenge

Let's start by heading over to the [MongoDB Atlas website](https://www.mongodb.com/atlas/database) and creating a free account.

This will be our central hub for managing our MongoDB clusters. Click "Try Free".

The screenshot shows the MongoDB Atlas website homepage. The URL in the browser bar is [mongodb.com/atlas/database](https://www.mongodb.com/atlas/database). The page features a dark header with the MongoDB logo, navigation links for Products, Solutions, Resources, Company, and Pricing, and a search bar. A prominent green button labeled "Try Free" is visible. Below the header, a navigation bar includes "Database" (selected), "Overview", "Features", "Deep Dive", "Resources", and "Explore", along with a "View all platform details" dropdown. The main content area has a dark background with the word "ATLAS" in small green capital letters at the top. The central text reads "Database. Deploy a multi-cloud database." in large, bold, white font. Below this, a description in smaller white text states: "The most advanced cloud database service on the market, with unmatched data distribution and mobility across AWS, Azure, and Google Cloud, built-in automation for resource and workload optimization, and so much more." At the bottom, there are two buttons: a green "Try Free" button and a "Contact sales" button with a right-pointing arrow.

ex02: set up your "neoG" cluster

challenge

1. Once you're logged in to MongoDB Atlas, click on "Build a Cluster" to create a new MongoDB cluster.
2. Choose your preferred cloud provider and region for the cluster. Let's name the cluster "neoG" for our class.

The screenshot shows the MongoDB Atlas interface for creating a new cluster. At the top, the URL is [cloud.mongodb.com/v2/64d79c5b5de0fc16b41daba6#/clusters/starterTemplates](https://cloud.mongodb.com/v2/64d79c5b5de0fc16b41daba6#/clusters/starterTemplates). Below the header, the MongoDB logo is displayed. The main section is titled "Deploy your database". It features three cluster templates:

- M10      \$0.08/hour**  
For production applications with sophisticated workload requirements.  

STORAGE 10 GB	RAM 2 GB	vCPU 2 vCPUs
------------------	-------------	-----------------

**+ Backups, elastic scalability, & more.**
- SERVERLESS      \$0.10/1M reads**  
For application development and testing, or workloads with variable traffic.  

STORAGE Up to 1TB	RAM Auto-scale	vCPU Auto-scale
----------------------	-------------------	--------------------

**+ Backups, elastic scalability, & more.**
- M0      FREE**  
For learning and exploring MongoDB in a cloud environment.  

STORAGE 512 MB	RAM Shared	vCPU Shared
-------------------	---------------	----------------

A "Compare Features" button is located below the cluster options. At the bottom of the page, there is a "FREE" section with a "Create" button, a note about the M0 cluster being ideal for experimentation, and a link to access advanced configuration.

FREE

**Create**

**Free forever!** Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

I'll deploy my database later

[Access Advanced Configuration](#)

**Provider**

**Region** ★ Recommended region ⓘ

**Mumbai (ap-south-1) ★**

**Name**  
You cannot change the name once the cluster is created.

**neoG**

**Tag (optional)**  
Create your first tag to categorize and label your...

**FREE**

**Create**

**Free forever!** Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

I'll deploy my database later

## ex03: connect to the "neoG" cluster

1. Click on "Connect" to set up the connection to your "neoG" cluster.

**Atlas** **Akanksha's ...** **Access Manager** **Billing** **All Clusters** **Get Help** **Akanksha**

**neoGProject** **Data Services** **App Services** **Charts**

**Overview** **AKANKSHA'S ORG - 2021-10-03 > NEOGBEPROJECT**

**Database Deployments**

**neoG** **Connect** **View Monitoring** **Browse Collections** **...** **FREE** **SHARED**

**Visualize Your Data**

Build dashboards and charts, and embed them in your apps with MongoDB Charts.

**R: 0** **W: 0** **Last 7 minutes** **Connections: 0** **Last 7 minutes** **In: 15.4 B/s** **Out: 39.1 B/s** **Last 7 minutes** **Data Size** **0.0 B / 512.0 MB (0%)** **Last 7 minutes**

**VERSION** **REGION** **CLUSTER TIER** **TYPE** **BACKUPS** **LINKED APP SERVICES** **ATLAS SQL** **ATLAS SEARCH**

6.0.9 AWS / Mumbai (ap-south-1) M0 Sandbox (General) Replica Set - 3 nodes Inactive Unable to load application data Connect Create Index

**+ Add Tag**

# ex04: create your "neoGStudent" database user

## challenge

1. Create a database user with the username "neoGStudent" and grant the necessary privileges for the database you'll be working with in our class.

The screenshot shows the MongoDB Atlas Data Services Quickstart interface. On the left, a sidebar lists various project components: Overview, Deployment (Database, Data Lake), Services (Device Sync, Triggers, Data API, Data Federation, Search, Stream Processing), and Security (Quickstart, Backup, Database Access, Network Access, Advanced). The 'Quickstart' section is currently active. At the top, there's a navigation bar with 'Atlas', 'Akanksha's ...', 'Access Manager', 'Billing', 'All Clusters', and 'Get Help'. The main content area is titled 'Security Quickstart' and contains instructions for creating a database user. It highlights the 'Username and Password' method, which is highlighted with a green border. A note states: '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. You can manage existing users via the [Database Access Page](#)'. Below this, there are fields for 'Username' (with placeholder 'Enter username') and 'Password' (with placeholder 'Enter password'). An 'Autogenerate Secure Password' button and a 'Copy' button are also present. A green notification bar at the bottom says 'Your cluster has finished provisioning.' with a close button.

Atlas Akanksha's ... Access Manager Billing All Clusters

**neoGBEProject** Data Services App Services Charts

## 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](#)

✓ How would you like to authenticate your connection?

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

**Username and Password** **Certificate**

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. You can manage existing users via the [Database Access Page](#).

**Username:** neoGStudent **Password:**  [Autogenerate Secure Password](#) [Copy](#)

**Create User**

## 2. Finally Finish and Close.

Atlas Akanksha's ... Access Manager Billing All Clusters Get

**neoGBEProject** Data Services App Services Charts

**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. You can manage existing IP entries via the [Network Access Page](#).

IP Address	Description
<input type="text" value="Enter IP Address"/>	<input type="text" value="Enter description"/>
<a href="#">Add My Current IP Address</a>	
<a href="#">Add Entry</a>	

**IP Access List** Description

49.206.130.159/32	My IP Address	<a href="#">EDIT</a>	<a href="#">REMOVE</a>
-------------------	---------------	----------------------	------------------------

**Finish and Close**

System Status: All Good  
©2023 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

## 3. Add your IP address to the IP Whitelist. For simplicity during our class, you can also allow access from anywhere.

1. On the left-hand navigation menu, under Security, click the "Network Access" link.
2. Click the "Add IP Address" button. The "Add IP Whitelist Entry" modal will appear.
3. To make your database accessible from anywhere, click "Allow Access From Anywhere".
4. Click Confirm.

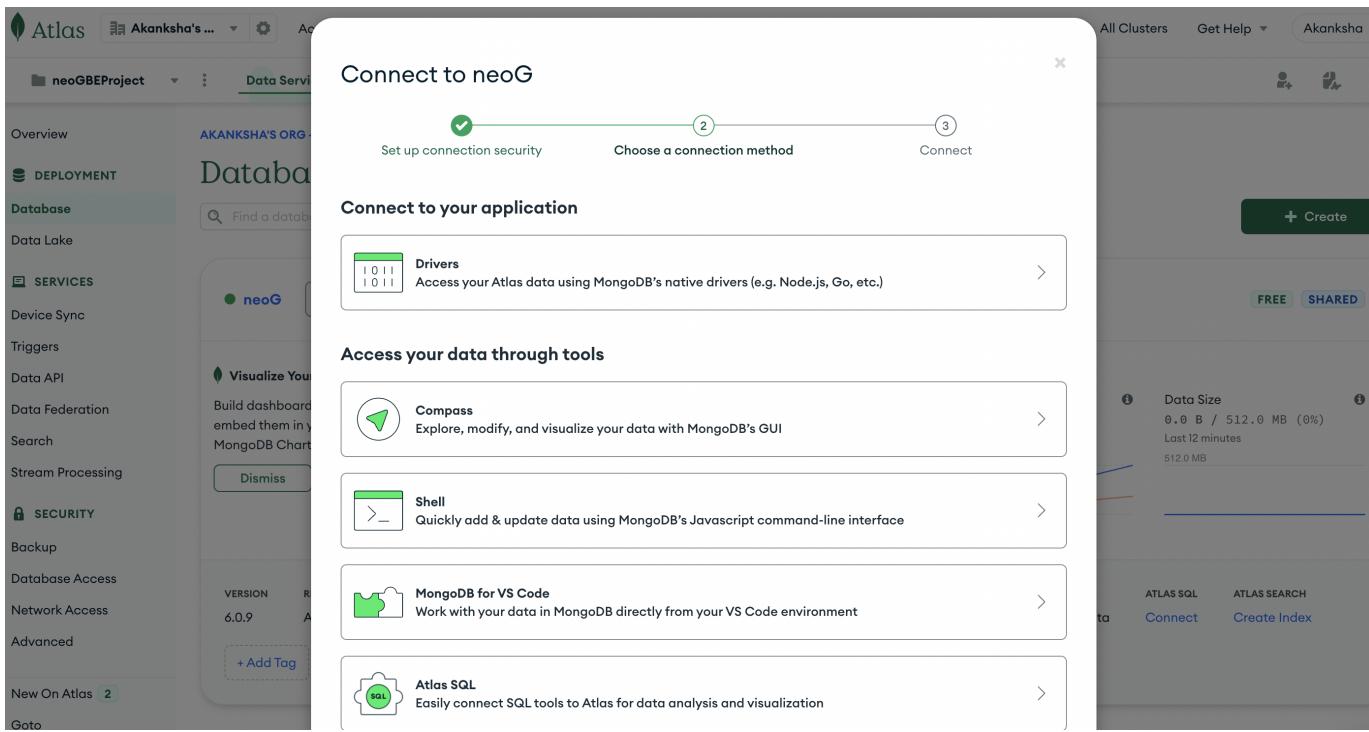
The screenshot shows the Neo4j Project interface. The left sidebar has 'Network Access' selected. The main area shows the 'IP Access List' tab of the 'Network Access' section. It displays a single entry: '49.206.130.159/32 (includes your current IP address)'. There is a note above the table stating: 'You will only be able to connect to your cluster from the following list of IP Addresses:'. A green button at the top right says '+ ADD IP ADDRESS'.

The screenshot shows the 'Add IP Access List Entry' modal. It contains fields for 'Access List Entry' (set to 0.0.0.0/0), 'Comment' (Optional comment describing this entry), and a toggle switch for 'This entry is temporary and will be deleted in' (set to 6 hours). At the bottom are 'Cancel' and 'Confirm' buttons.

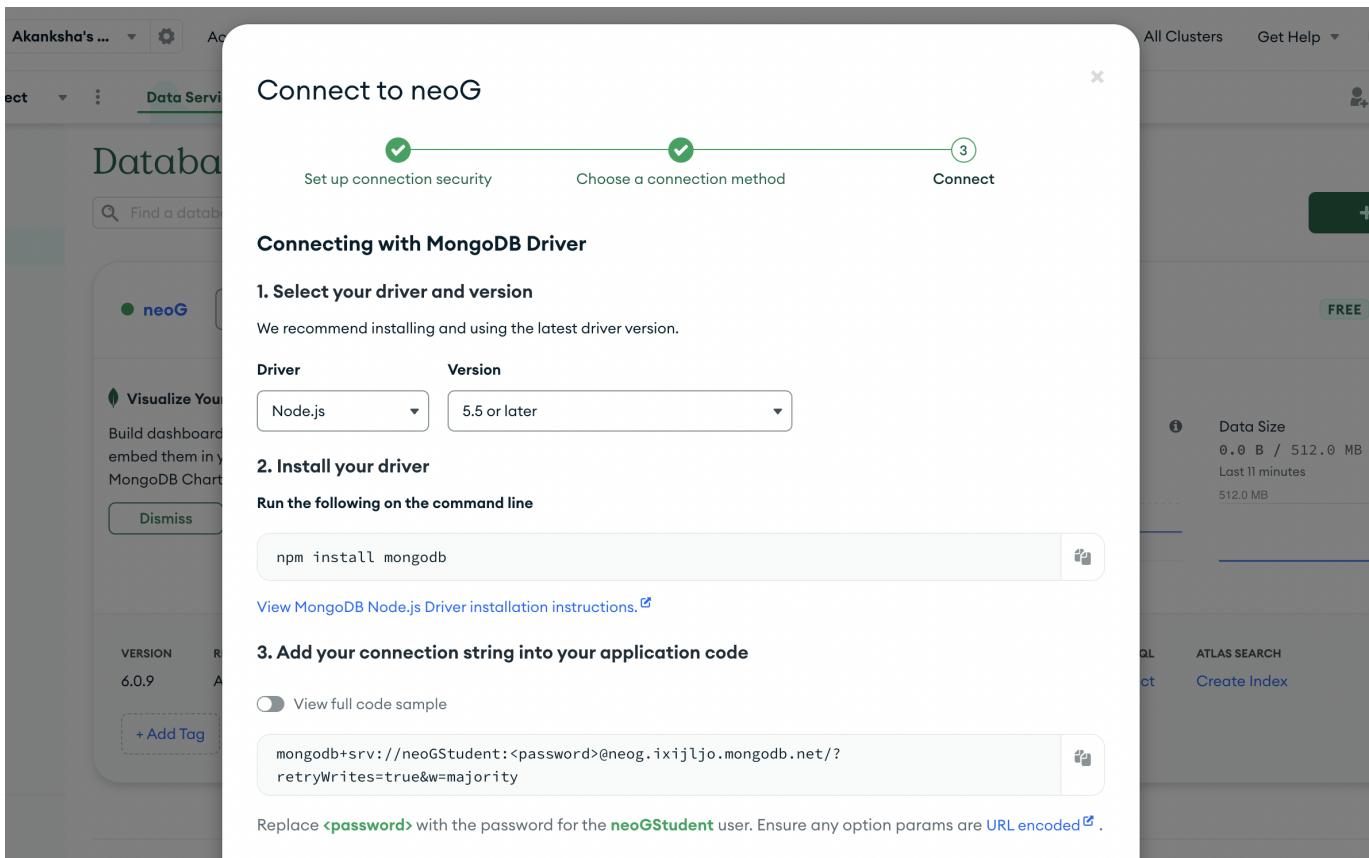
## ex05: get your connection string

### challenge

1. Click on "Connect" button again and select "Connect your application."



2. Copy the connection string. This is your golden ticket to connecting to your "neoG" MongoDB cluster from your Replit project. Your connection string is the one starting from "mongodb+srv://neoGStudent:....."



ex06: create your replit account

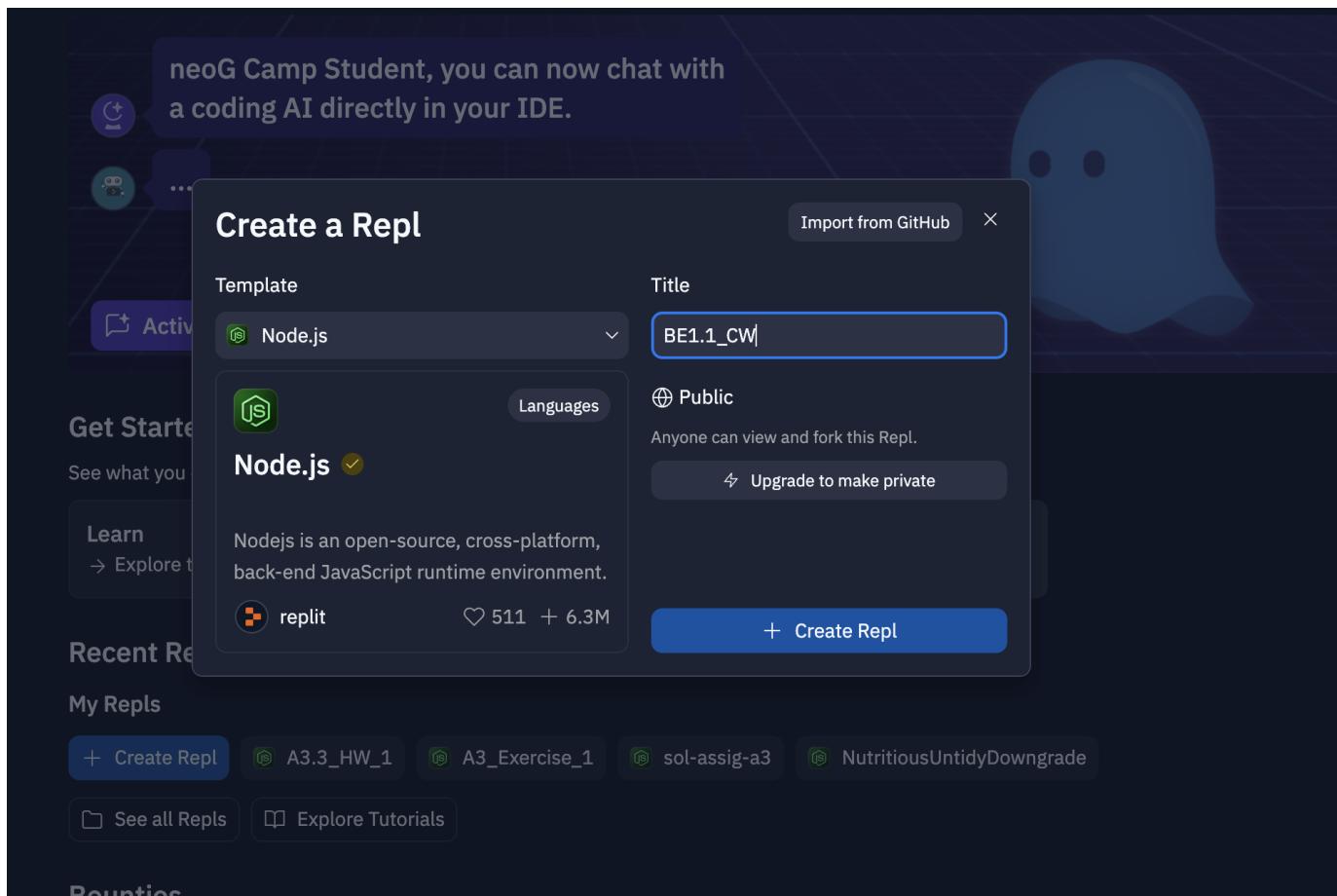
## challenge

1. Let's now create a free account on [Replit](#), which is where we'll be writing our Node.js code.

## ex07: create a new replit project with node.js

### challenge

1. Set up a new Replit project and choose Node.js as the programming language you'll be using in our class.



## ex08: keep your connection details safe

### challenge

1. Go to the Secrets tab on the left sidebar in Replit.

The screenshot shows the Beamer Cloud Web interface. The top bar includes icons for file operations and a project dropdown labeled "BE1.1\_CW". The sidebar on the left contains sections for "Files" and "Tools".

**Files**

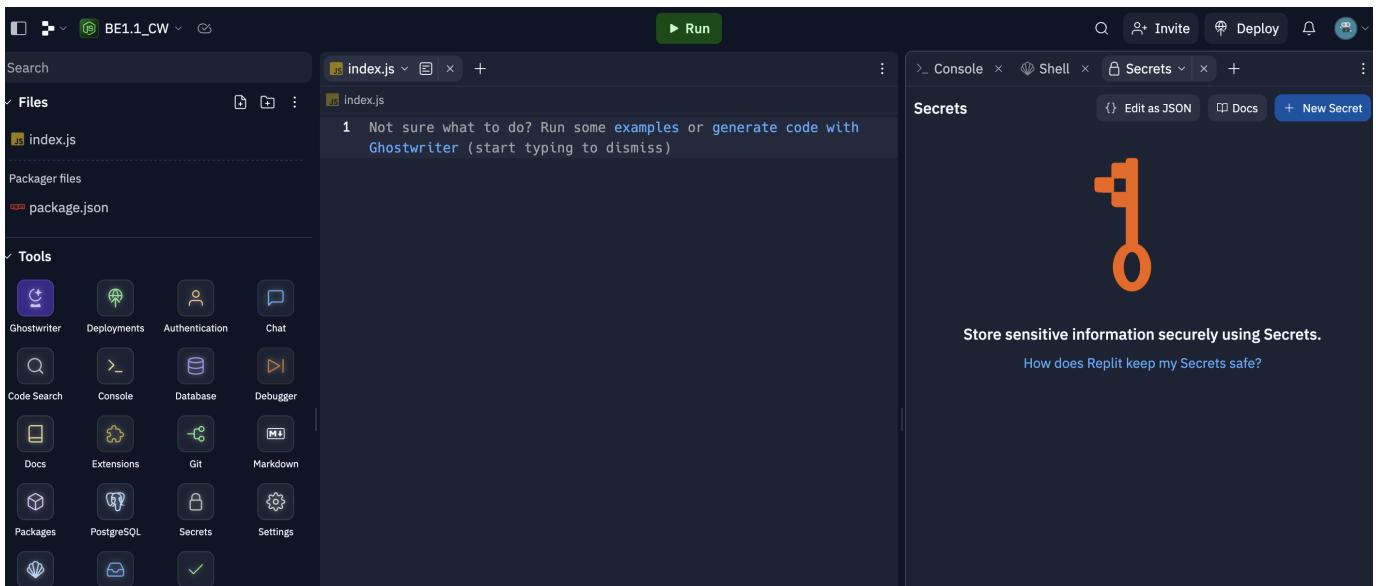
- index.js
- Packager files
- package.json

**Tools**

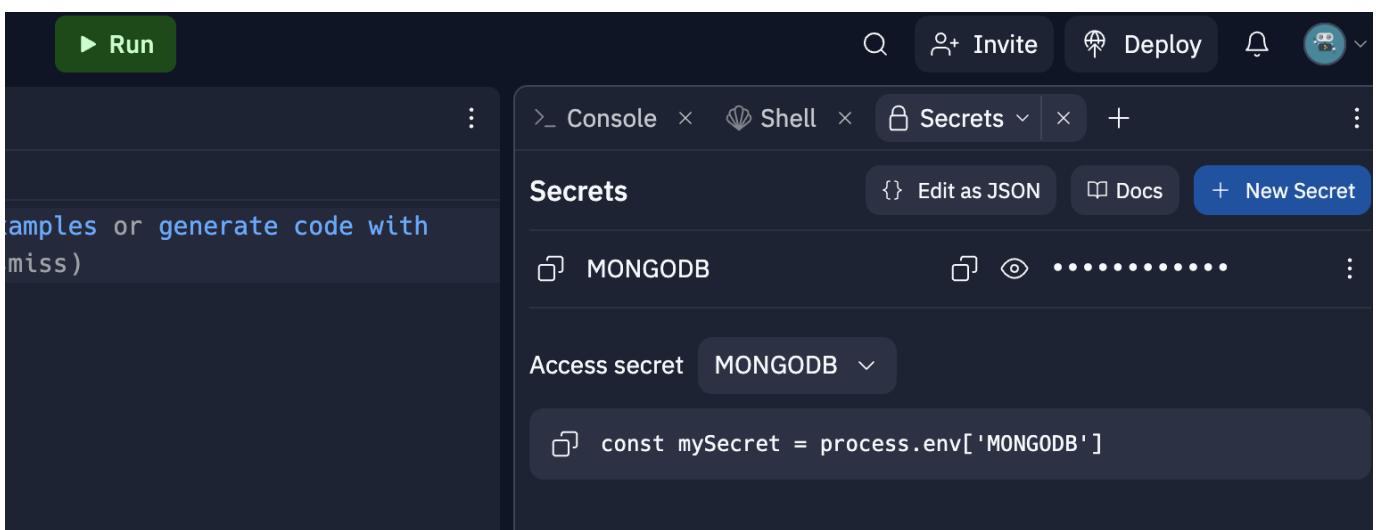
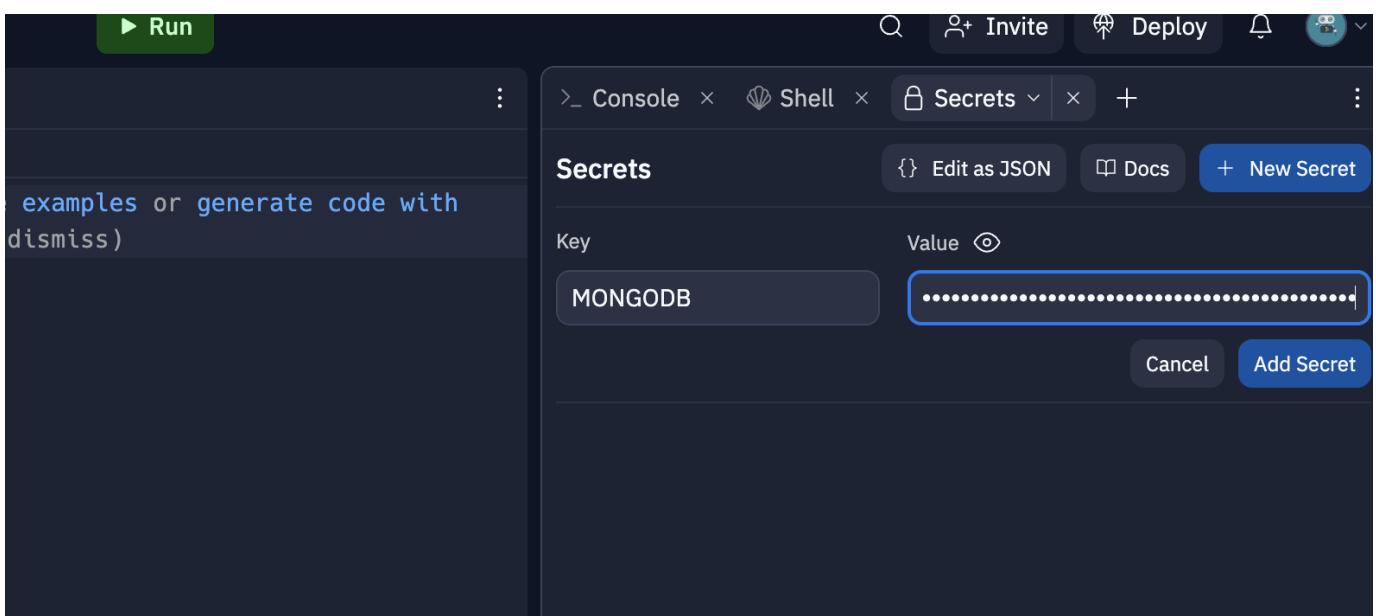
Ghostwriter	Deployments	Authentication	Chat
Code Search	Console	Database	Debugger
Docs	Extensions	Git	Markdown
Packages	PostgreSQL	Secrets	Settings
Shell	Threads	Unit Tests	

The main area on the right displays a code editor for "index.js" with the following content:

```
1 Not sure what to do?  
Ghostwriter (start t)
```



2. Create new secrets for your MongoDB connection string and any other sensitive information you need to keep secure. Make sure to replace the <password> with your actual password in the connection string.



## ex09: connect to mongodb in replit

### challenge

1. Let's write some Node.js code! Here's a snippet to help you connect to MongoDB in your Replit project. There are 2 ways given in the code. Use any one:

```
const mongoose = require('mongoose')
const { MongoClient } = require('mongodb')

// Access your MongoDB connection string from secrets
const mongoURI = process.env.MONGODB_URI

// Use mongoose to connect
mongoose
  .connect(mongoURI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => {
    console.log('Connected to MongoDB')
  })
  .catch((error) => {
    console.error('Error connecting to MongoDB:', error)
  })

// **Or Use MongoClient to connect (alternative)**
const client = new MongoClient(mongoURI)
client
  .connect()
  .then(() => {
    console.log('Connected to MongoDB using MongoClient')
  })
  .catch((error) => {
    console.error('Error connecting to MongoDB using MongoClient:', error)
  })
```

COPY

2. Replace `process.env.MONGODB_URI` with the actual secret name you've used for your MongoDB connection string. See line number 5 in image below.

The screenshot shows the Replit IDE interface. In the top left, it says 'E1.1\_CW' with a dropdown arrow. On the right, there's a 'Stop' button. The main area has a dark theme with two tabs: 'index.js' (the active one) and another 'index.js'. The code in 'index.js' is:

```
1 const mongoose = require('mongoose');
2
3 // Access your MongoDB connection string from secrets
4 const mongoURI = process.env.MONGODB;
5
6 mongoose.connect(mongoURI, {
7   useNewUrlParser: true,
8   useUnifiedTopology: true
9 }).then(() => {
10   console.log('Connected to MongoDB');
11 }).catch(error => {
12   console.error('Error connecting to MongoDB:', error);
13 });
14
```

introducing mongoose: basic operations with the "Car" model

ex01: add mongoose to your project

challenge

1. Open your Replit project and make sure you're connected to your MongoDB Atlas cluster.
2. Install Mongoose by running `npm install mongoose` in the Replit terminal.
3. In your `index.js` code, require Mongoose at the top:

solution

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

[COPY](#)

ex02: introduce the "Car" model

challenge

1. Let's start by creating a basic Mongoose model. In the same file (`index.js`), add the following code below your Mongoose connection:

## solution

```
const carSchema = new mongoose.Schema({  
  make: String,  
  model: String,  
  year: Number,  
)  
  
const Car = mongoose.model('Car', carSchema)
```

COPY

## \*\*ex03: define functions for mongoose operations\*\*

### challenge

Define a named function to add data to the "Car" model:

## solution

```
async function addCarData() {  
  const newCar = new Car({  
    make: 'Toyota',  
    model: 'Corolla',  
    year: 2022,  
)  
  
  try {  
    const savedCar = await newCar.save()  
    console.log('Car data saved successfully:', savedCar)  
  } catch (error) {  
    console.error('Error saving car data:', error)  
  }  
}  
  
addCarData()
```

COPY

## ex04: add one more data entry

### challenge

Define a function to add another data entry:

## solution

```
async function addAnotherCarData() {  
  const anotherCar = new Car({  
    make: 'Honda',  
    model: 'Civic',  
    year: 2023,  
)  
}
```

```
try {
  const savedAnotherCar = await anotherCar.save()
  console.log('Another car data saved successfully:', savedAnotherCar)
} catch (error) {
  console.error('Error saving another car data:', error)
}
}

addAnotherCarData()
```

COPY

## ex05: perform a findAll operation

### challenge

Define a function to retrieve all car data from the database:

### solution

```
async function findAllCars() {
  try {
    const allCars = await Car.find()
    console.log('All cars in the database:', allCars)
  } catch (error) {
    console.error('Error fetching all cars:', error)
  }
}

findAllCars()
```

COPY

## get ready for production

## ex01: create a "models" directory

### challenge

In your Replit project, create a new directory named "models" to house your Mongoose models.

## ex02: move the "Car" model

## challenge

Move your existing "Car" model code from the main file to a new file named `car.js` inside the "models" directory. The code in `car.js` should look like this:

## solution

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

const carSchema = new mongoose.Schema({
  make: String,
  model: String,
  year: Number,
})

const Car = mongoose.model('Car', carSchema)

module.exports = Car
```

[COPY](#)

## ex03: create a "db.js" file for database connection

## challenge

In the root of your project, create a new file named `db.js` to handle the database connection. The code in `db.js` should look like this:

## solution

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

// Access your MongoDB connection string from secrets
const mongoURI = process.env.MONGODB_URI

mongoose
  .connect(mongoURI, {
    useNewUrlParser: true,
    useUnifiedTopology: true,
  })
  .then(() => {
    console.log('Connected to MongoDB')
  })
  .catch((error) => {
    console.error('Error connecting to MongoDB:', error)
  })
```

[COPY](#)

## ex04: update your main file

## challenge

1. In your main file (the one where you were working earlier), require the db.js file at the top:
2. Next, require the "Car" model at the top as well:

## solution

```
require('./db')
const Car = require('./models/car')
```

COPY

## ex05: test the database connection #

## challenge

1. Open your Replit project's console and run your main file.
2. If the connection is successful, you'll see the message "Connected to MongoDB."

```
index.js
1  require('./db');
2  const Car = require('./models/car');
3
4  async function addCarData() {
5    const newCar = new Car({
6      make: 'Toyota',
7      model: 'Corolla',
8      year: 2022,
9    });
10
11  try {
12    const savedCar = await newCar.save();
13    console.log('Car data saved successfully:', savedCar);
14  } catch (error) {
15    console.error('Error saving car data:', error);
16  }
17}
18
19 addCarData();
20
21 async function findAllCars() {
22  try {
23    const allCars = await Car.find();
24    console.log('All cars in the database:', allCars);
25  } catch (error) {
26    console.error('Error fetching all cars:', error);
27  }
28}
29
30 findAllCars();
```

Console output:

```
Hint: hit control+c anytime to enter REPL.
Connected to MongoDB
Car data saved successfully: {
  make: 'Toyota',
  model: 'Corolla',
  year: 2022,
  _id: new ObjectId("64e6498cae74492135d3ade9"),
  __v: 0
}
All cars in the database: [
  {
    _id: new ObjectId("64e64931f13cf5159a10ee51"),
    make: 'Toyota',
    model: 'Corolla',
    year: 2022,
    __v: 0
  },
  {
    _id: new ObjectId("64e6498cae74492135d3ade9"),
    make: 'Toyota',
    model: 'Corolla',
    year: 2022,
    __v: 0
  }
]
```