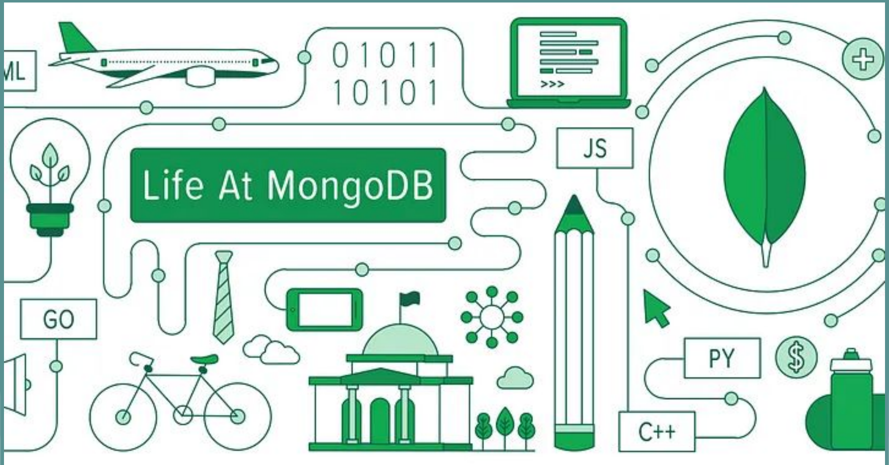
Deploy Node Express MongoDB on Render

Project Description

"Deploy Node Express MongoDB on Render" is a project that involves deploying a Node.js application with an Express.js backend and MongoDB database on the Render platform. The project showcases how to setup and deploy a full-stack web application, enabling users to interact with data stored in a MongoDB database through RESTful API endpoints. By following this project, you'll learn how to configure and deploy a Node.js app along with a MongoDB database on Render, demonstrating a practical approach to building and hosting a modern web application stack in a seamless and scalable manner.

Create a cluster in MongoDB



What is MongoDB Atlas?

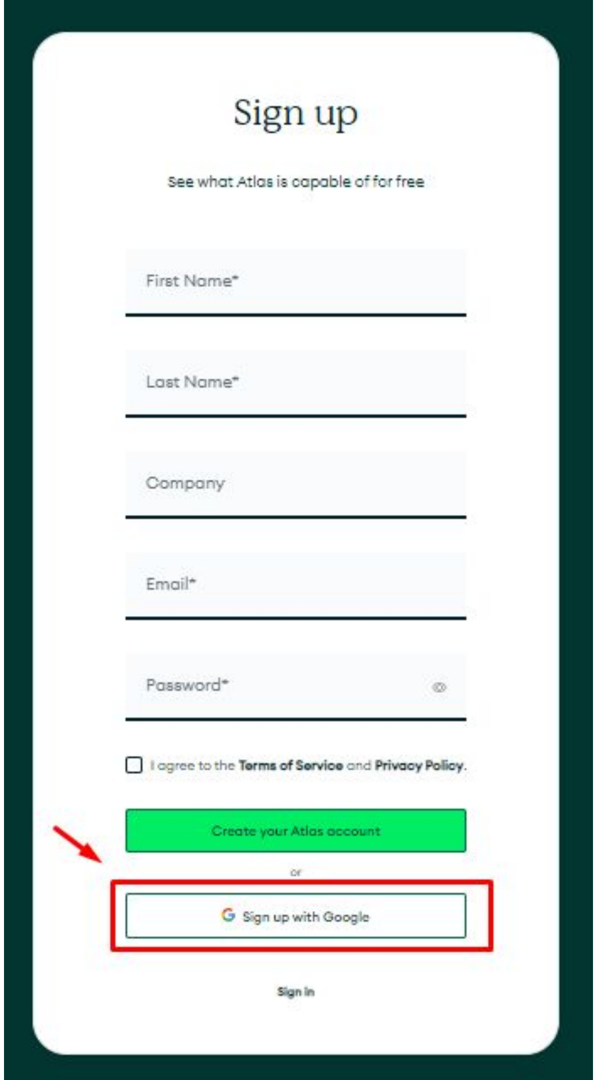
MongoDB Atlas is MongoDB’s fully-managed cloud database service that comes with a free tier. The service is built to handle enterprise workloads, with support for global clusters.

You can store your data with Amazon Web Services (AWS), Google Cloud Platform, or Microsoft Azure. However, you don’t need to set up an account with any of these platforms. MongoDB Atlas takes care of all this behind the scenes.

Set up a free MongoDB Atlas cluster

1. Go to the [MongoDB Atlas landing page]

2. Fill in the required information
(email address, first name, last name, and password)
or Sign Up with Google



Sign up

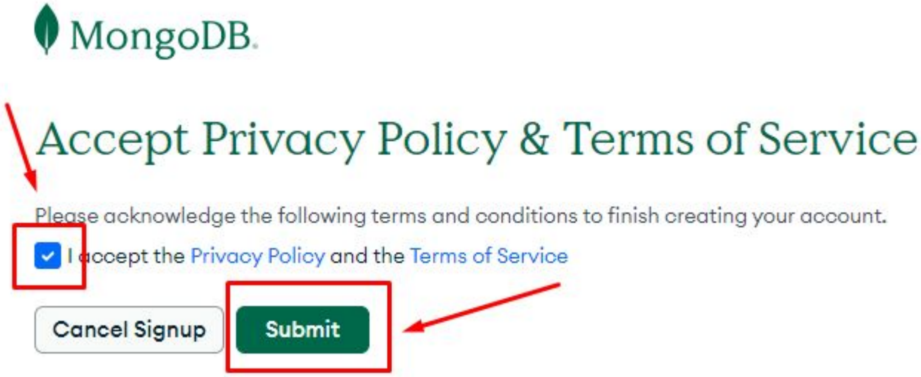
See what Atlas is capable of for free

☐ I agree to the **Terms of Service** and **Privacy Policy**.

Set up a free MongoDB Atlas cluster

3. Click the terms of service and privacy policy links, which should open on a new tab.
If you want to continue with the registration, select the I agree to the terms of service and privacy policy check box.

4. Click the Submit at the bottom of the form.



MongoDB.

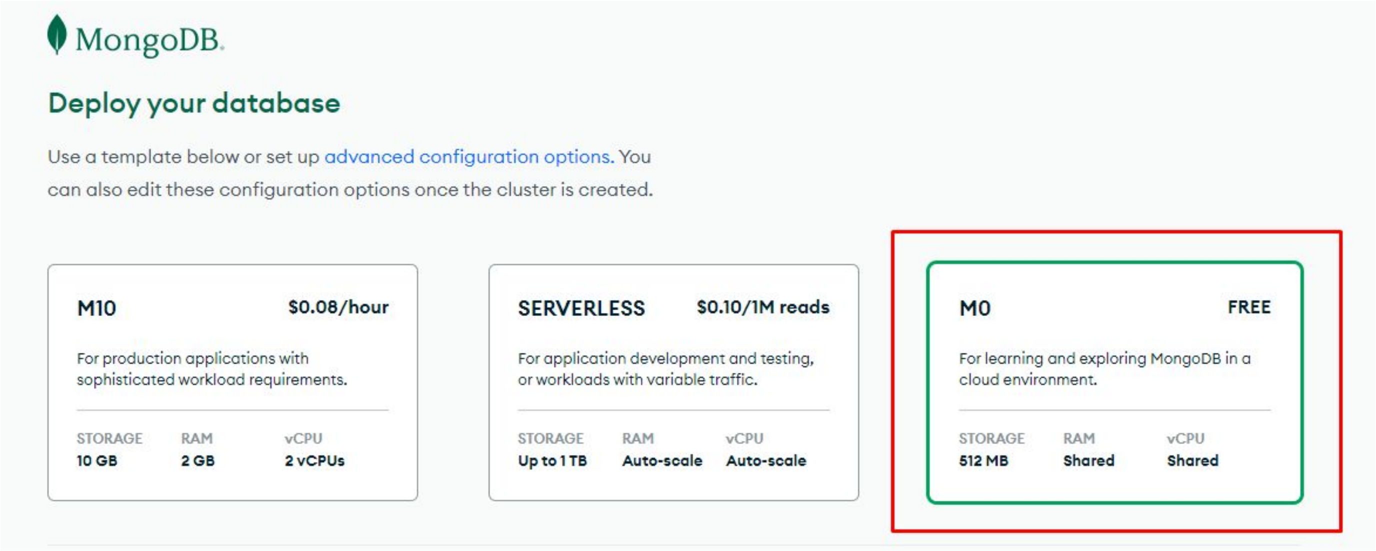
Accept Privacy Policy & Terms of Service

Please acknowledge the following terms and conditions to finish creating your account.

☑ I accept the Privacy Policy and the Terms of Service

Set up a free MongoDB Atlas cluster

5. The website will ask to choose a cluster. Choose **Free Clusters**



MongoDB.

Deploy your database

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

M10 $0.08/hour

For production applications with sophisticated workload requirements.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

SERVERLESS $0.10/1M reads

For application development and testing, or workloads with variable traffic.

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

M0 FREE

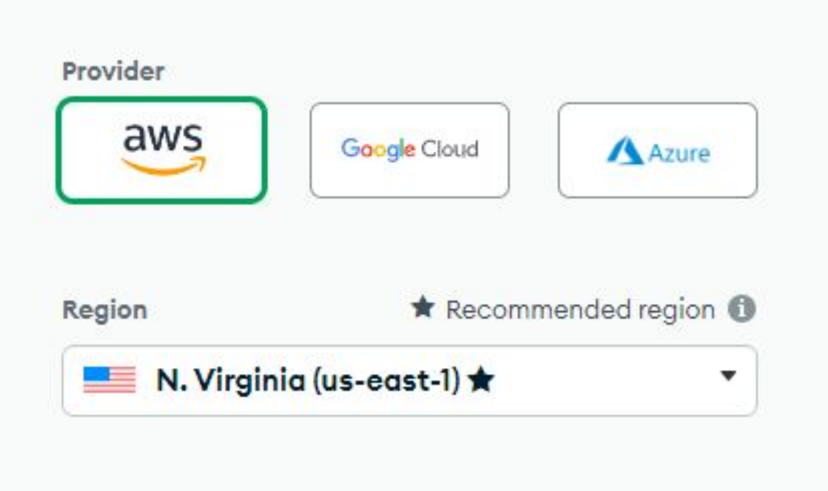
For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

Set up a free MongoDB Atlas cluster

6. In the **Cloud Provider & Region** section, the **AWS** option should be selected as the default provider, but you can select any provider. All three platforms support the free tier.

7. Beneath the list of providers, select a region.

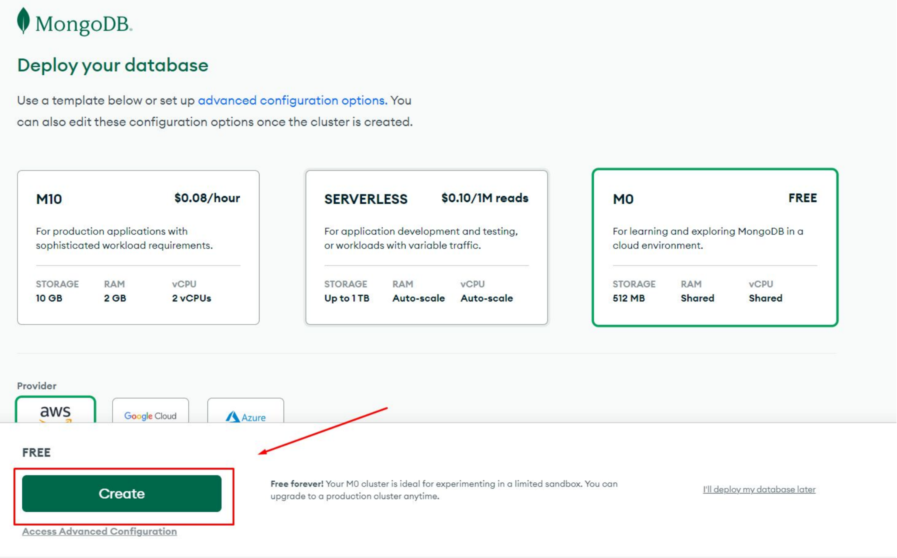


Region

N. Virginia (us-east-1) ★

Set up a free MongoDB Atlas cluster

8. Click the **Create Cluster** button at the bottom of the web page.



Deploy your database

Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.

M10 $0.08/hour

For production applications with sophisticated workload requirements.

STORAGE	RAM	vCPU
10 GB	2 GB	2 vCPUs

SERVERLESS $0.10/1M reads

For application development and testing, or workloads with variable traffic.

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

M0 FREE

For learning and exploring MongoDB in a cloud environment.

STORAGE	RAM	vCPU
512 MB	Shared	Shared

Provider

FREE

Access Advanced Configuration

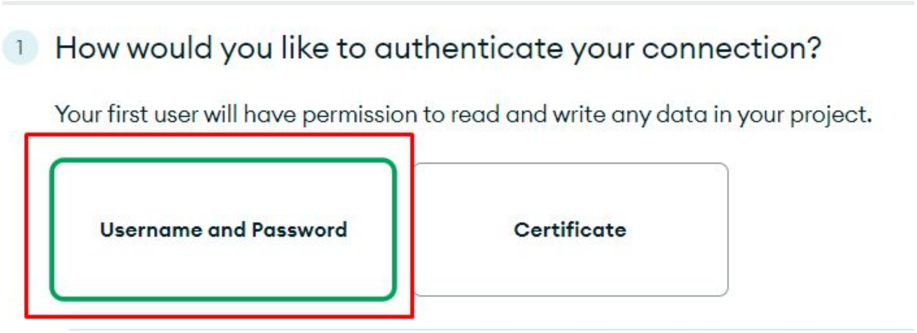
Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

I'll deploy my database later

Configure IP address and connection string

1. How would you like to authenticate your connection?

   Choose “Username and password”



1 How would you like to authenticate your connection?

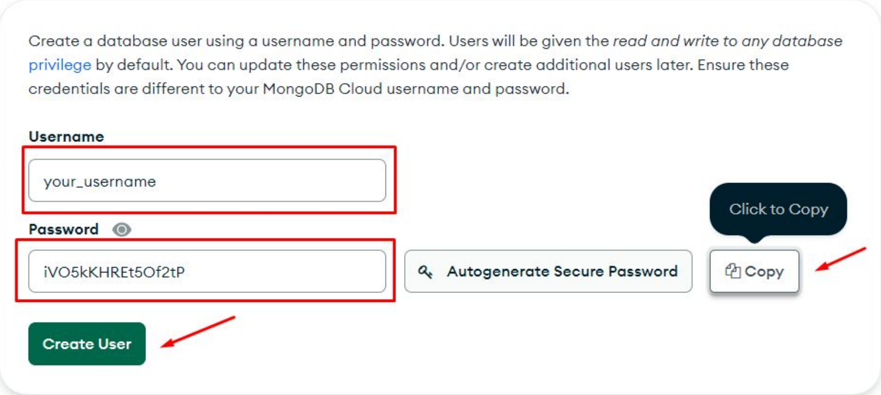
Your first user will have permission to read and write any data in your project.

Configure IP address and connection string

Type admin in the Username text box (or whatever name you want to use), and then type a password in the Password text box.

To make it easier to connect to MongoDB Atlas from Studio 3T, your password should include only alphanumeric characters, that is, letters and numbers only with no special characters.

If you use special characters, you will need to encode them when creating a connection string for accessing the MongoDB service.



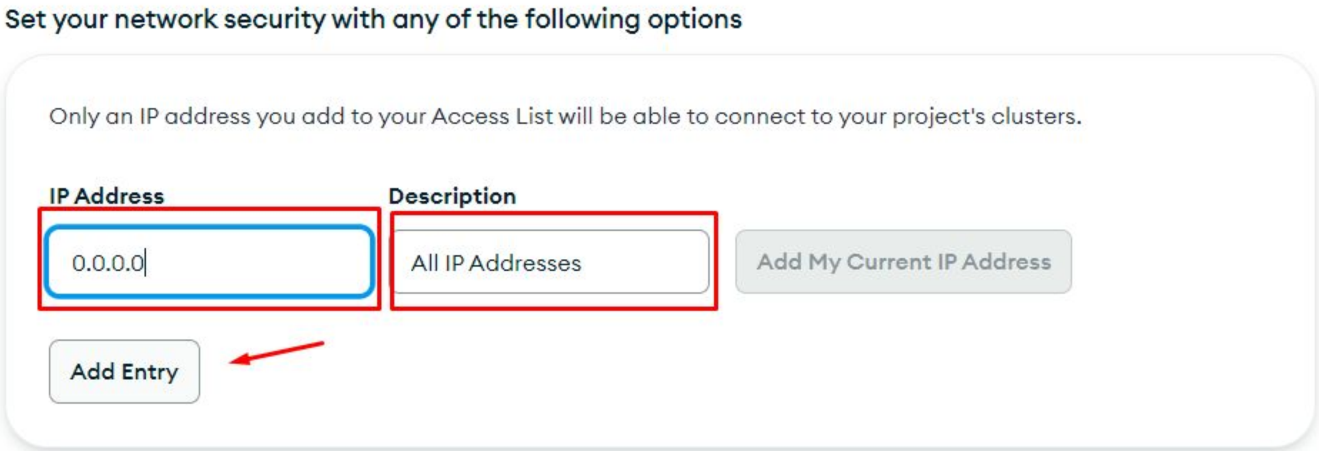
Create a database user using a username and password. Users will be given the *read and write to any database* privilege by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

Username: your_username

Password: iVO5kKHREt5Of2tP

Configure IP address and connection string

For security reasons, MongoDB Atlas blocks all outside connections by default. In order to connect from Render, you must Allow Access from Anywhere



Set your network security with any of the following options

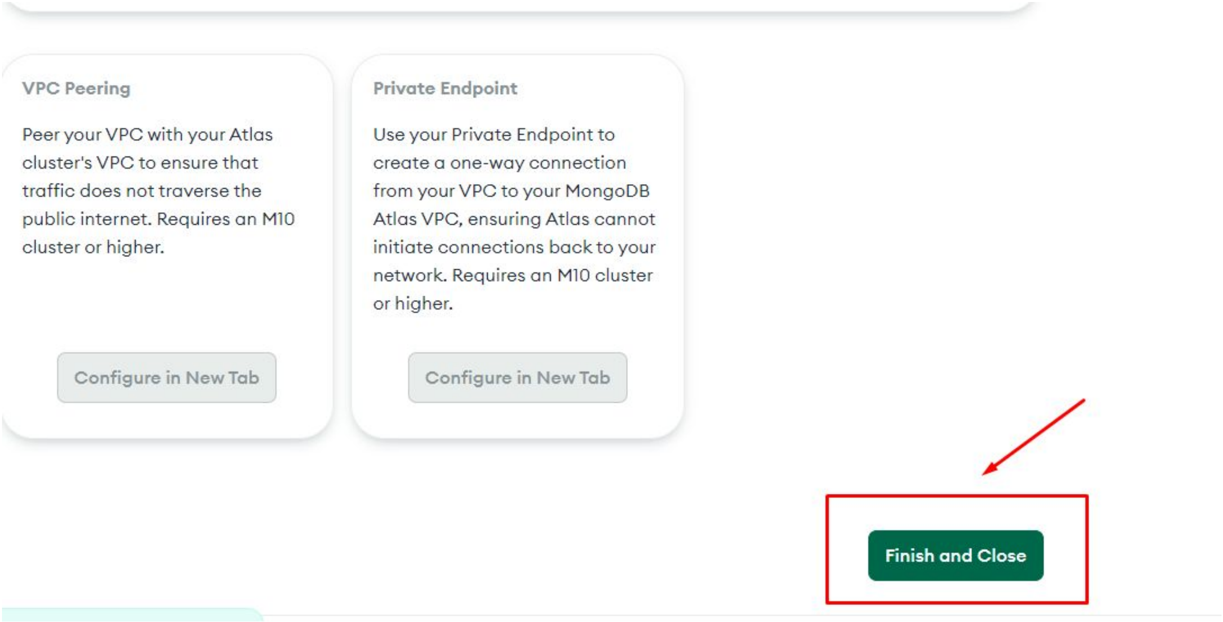
Only an IP address you add to your Access List will be able to connect to your project's clusters.

IP Address: 0.0.0.0

Description: All IP Addresses

Configure IP address and connection string

Click the **Finish and Close** button



VPC Peering

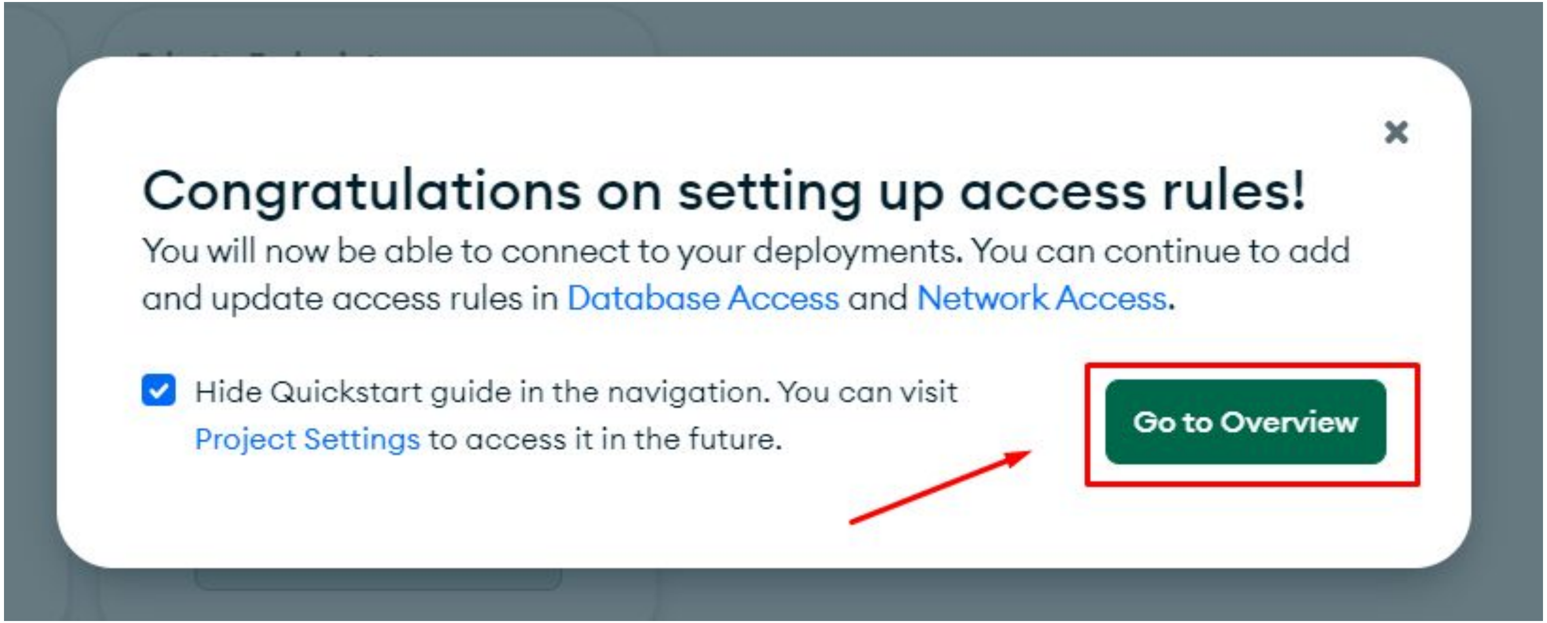
Peer your VPC with your Atlas cluster's VPC to ensure that traffic does not traverse the public internet. Requires an M10 cluster or higher.

Private Endpoint

Use your Private Endpoint to create a one-way connection from your VPC to your MongoDB Atlas VPC, ensuring Atlas cannot initiate connections back to your network. Requires an M10 cluster or higher.

Configure IP address and connection string

Click the **Go to Overview** button



Congratulations on setting up access rules!

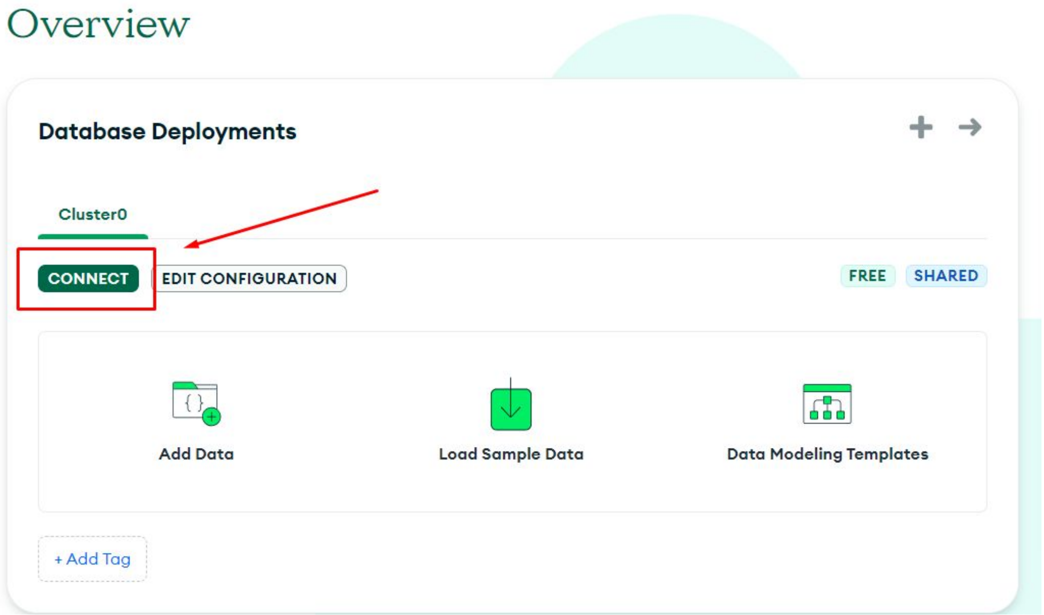
You will now be able to connect to your deployments. You can continue to add and update access rules in Database Access and Network Access.

☑ Hide Quickstart guide in the navigation. You can visit Project Settings to access it in the future.

Connect to Cluster

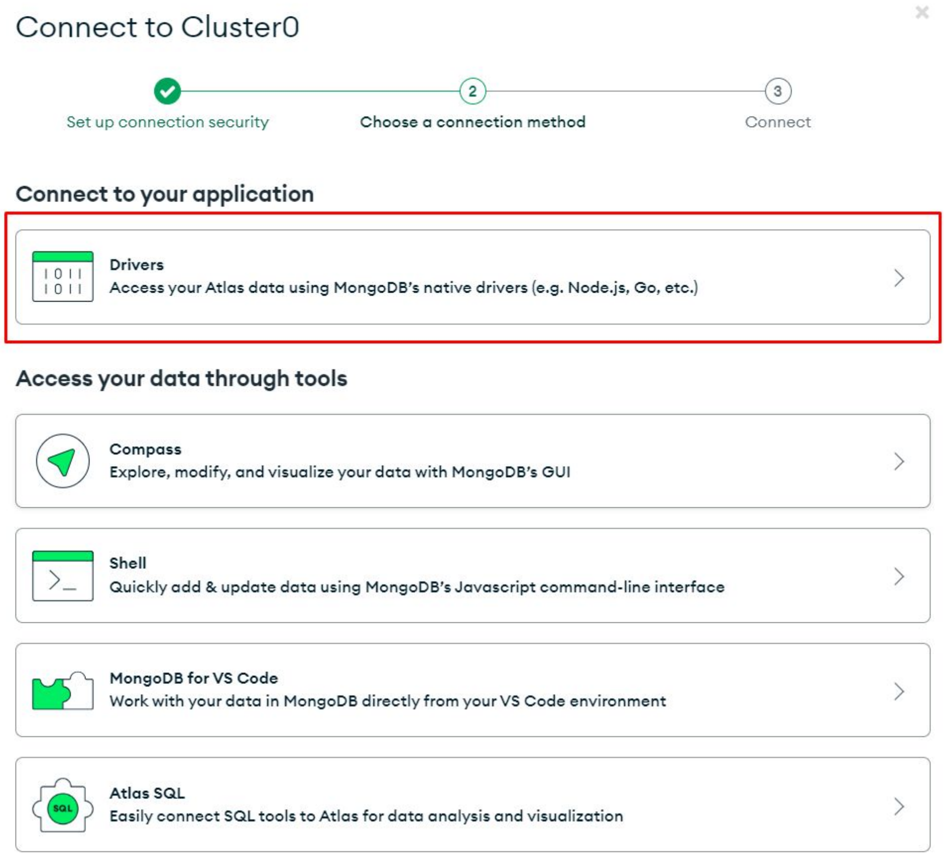
The next task is to generate the connection string or Uniform Resource Identifier (URI).

To start this process, click **Connect** button



Connect to Cluster

Click **Connect Your Application**



Connect to Cluster0

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

Compass
Explore, modify, and visualize your data with MongoDB's GUI

Shell
Quickly add & update data using MongoDB's Javascript command-line interface

MongoDB for VS Code
Work with your data in MongoDB directly from your VS Code environment

Atlas SQL
Easily connect SQL tools to Atlas for data analysis and visualization

Connect to Cluster

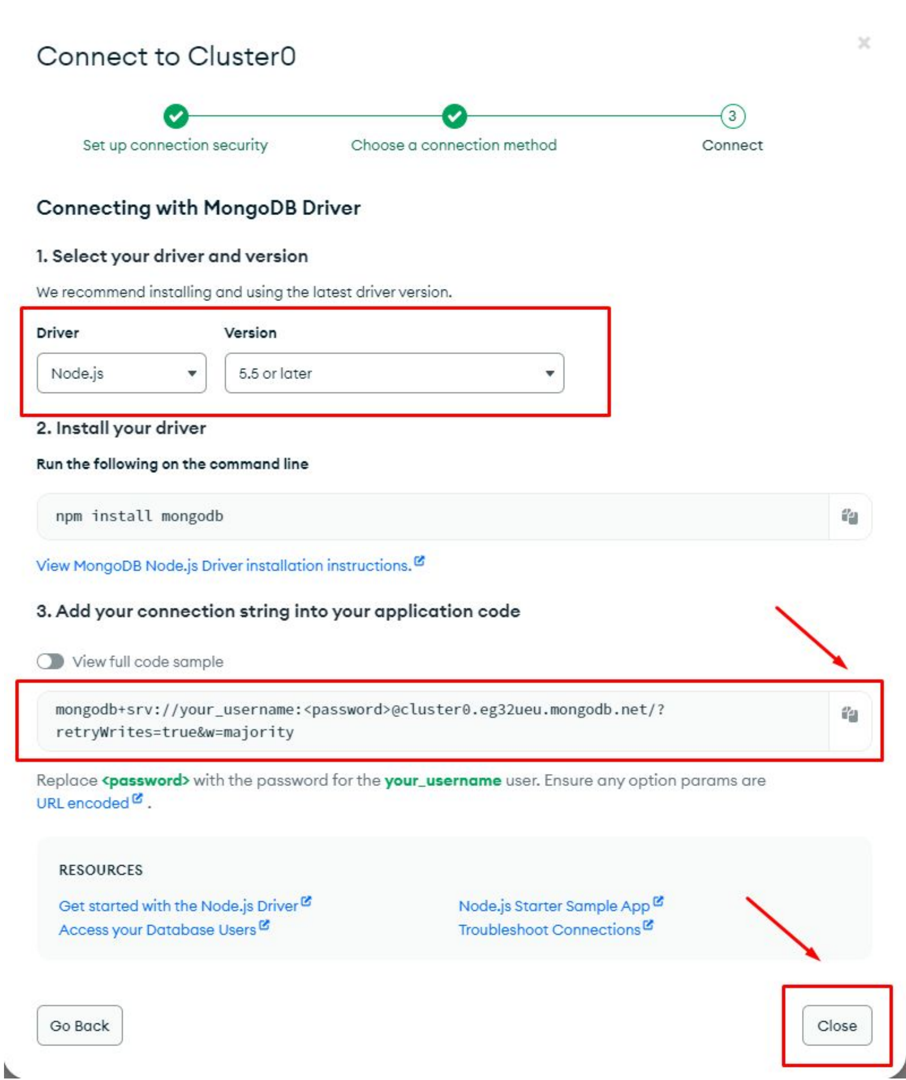
In the Step 1 section, select **Node.js** from the **DRIVER** drop-down list, and select **5.5 or later** from the **VERSION** drop-down list.

In the Step 3 section, click **Copy** to copy the connection string to your clipboard, and then paste the connection string to a safe location.

When you use the connection string or URI, you must replace the placeholder with the password you created for the administrator account. Don’t forget to remove the <> as well.

9. Click **Close** to close the Connect to Cluster0 dialog box, and then sign out of the MongoDB Atlas service.

That’s it! We’re done with creating a cluster and connecting with IP.



Connect to Cluster0

Connecting with MongoDB Driver

1. Select your driver and version

We recommend installing and using the latest driver version.

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

View full code sample

```
mongodb+srv://your_username:<password>@cluster0.eg32ueu.mongodb.net/?
retryWrites=true&w=majority
```

Replace **<password>** with the password for the **your_username** user. Ensure any option params are URL encoded .

RESOURCES

Get started with the Node.js Driver
Node.js Starter Sample App
Access your Database Users
Troubleshoot Connections

Set up
Express app

Set up Express app

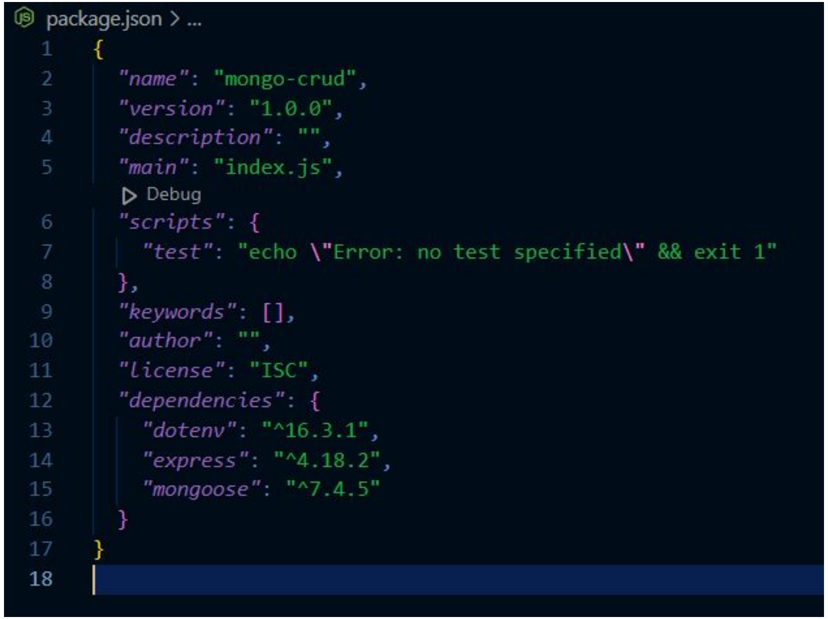
If you’re setting up your project from scratch, create a new folder for the project and enter it. Then create a `package.json` file by running the following command in your terminal.

```
npm init -y
```

Install dependencies

Now, let’s install the third-party dependencies required to run this application. First, we’ll install `dotenv`, `mongoose`, `express` and their types by running this command:

```
npm install dotenv mongoose express
```

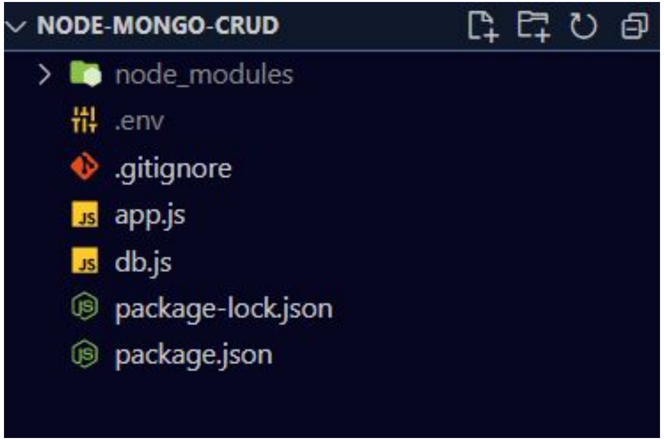


```
{
  "name": "mongo-crud",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "mongoose": "^7.4.5"
  }
}
```

Full Project Code

Project Structure

- app.js: This file is the main entry point of the Node.js application. It contains the configuration and implementation of the Express.js server, where you define your API routes, handle requests and responses, and interact with the MongoDB database.
- db.js: The db.js file is responsible for establishing a connection to the MongoDB database. It utilizes the Mongoose library to create and manage database connections, schemas, and models. This file abstracts away the database operations, making it easier to work with MongoDB in the application.
- .env: The .env file contains environment variables that store sensitive configuration information, such as database credentials, API keys, and other settings. These variables are loaded into the application using a library like `dotenv`, ensuring that sensitive information is kept secure and separate from the codebase.
- .gitignore: The .gitignore file specifies which files and directories should be ignored by version control systems like Git. This helps prevent sensitive or unnecessary files from being included in the repository. Commonly ignored files include node_modules and .env.

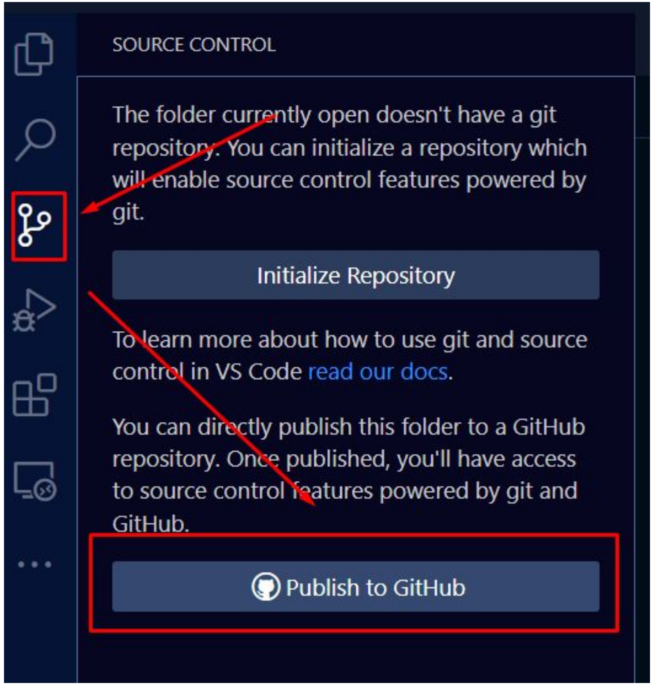


```
NODE-MONGO-CRUD
> node_modules
  .env
  .gitignore
  app.js
  db.js
  package-lock.json
  package.json
```

GitHub Repository

Create a Private GitHub Repository for Source Control and Publish Branch

1



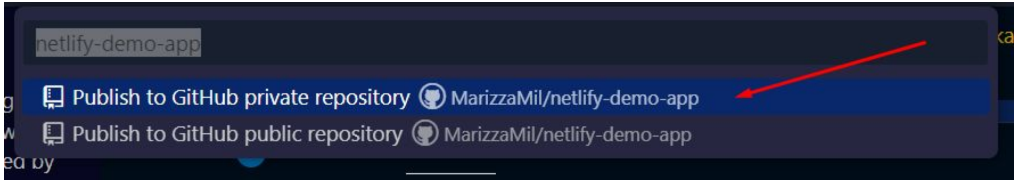
SOURCE CONTROL

The folder currently open doesn't have a git repository. You can initialize a repository which will enable source control features powered by git.

To learn more about how to use git and source control in VS Code read our docs.

You can directly publish this folder to a GitHub repository. Once published, you'll have access to source control features powered by git and GitHub.

2

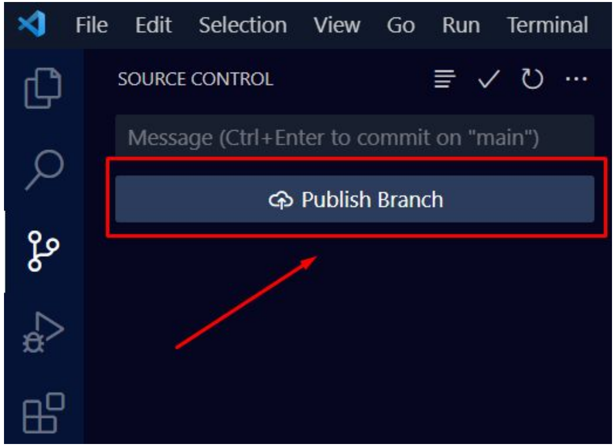


netlify-demo-app

Publish to GitHub private repository MarizzaMil/netlify-demo-app

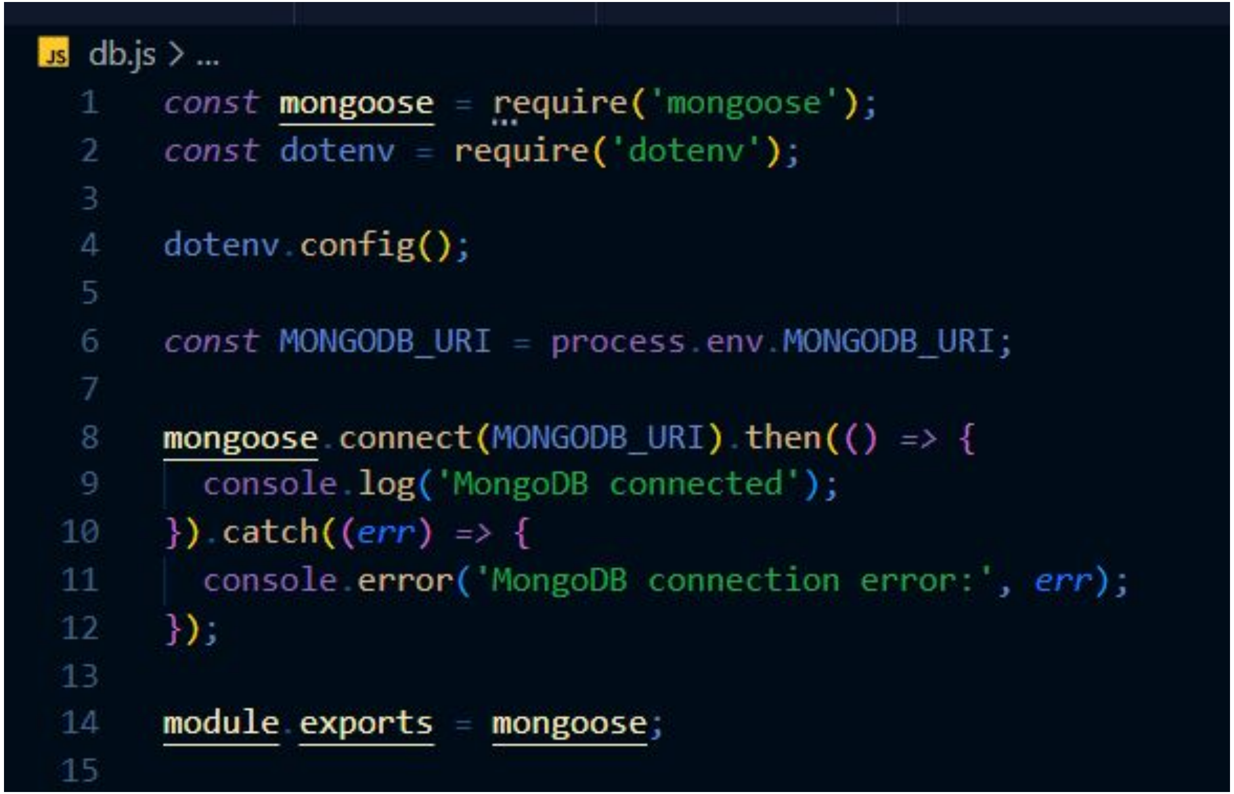
Publish to GitHub public repository MarizzaMil/netlify-demo-app

3



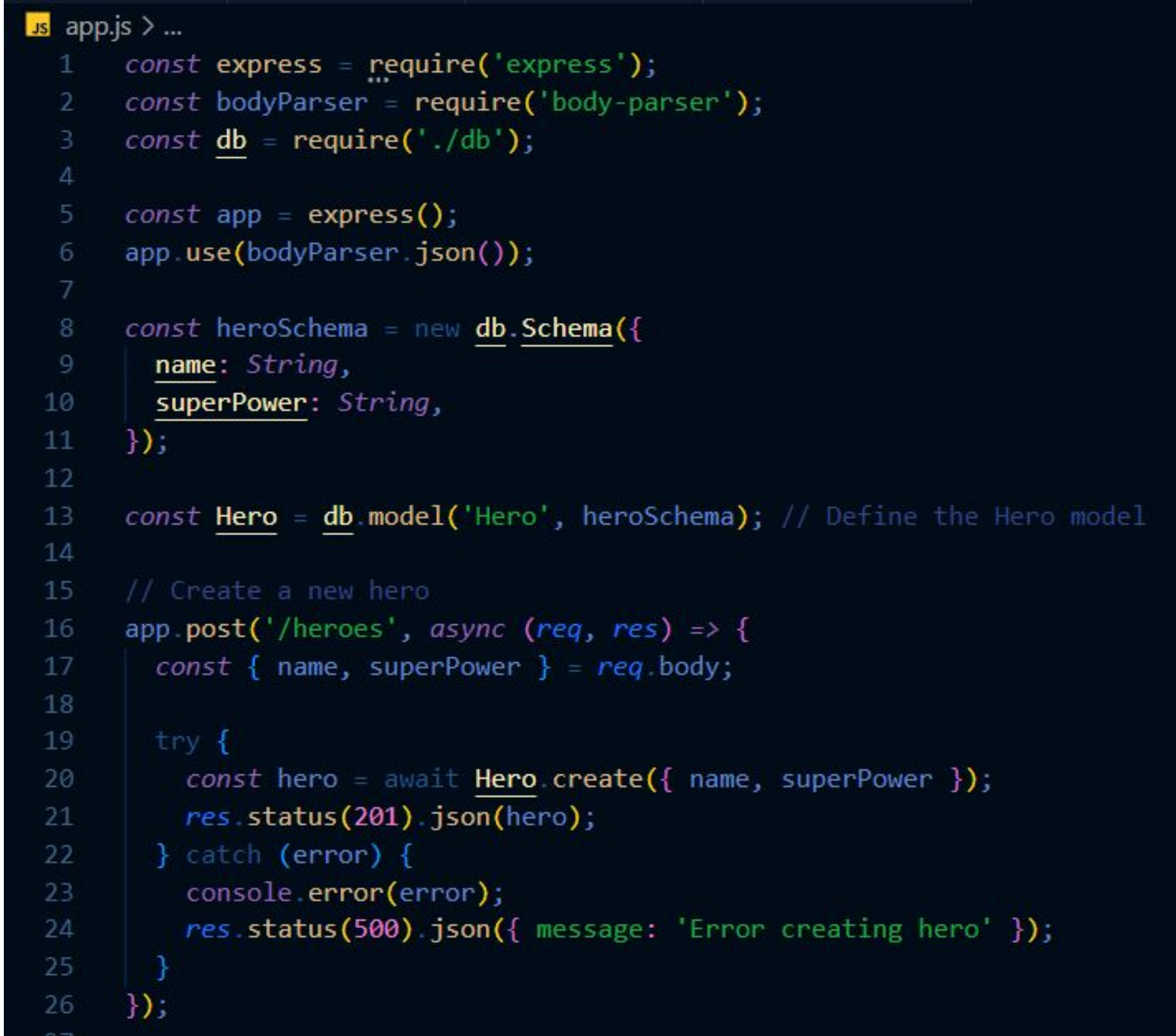
Message (Ctrl+Enter to commit on "main")

db.js



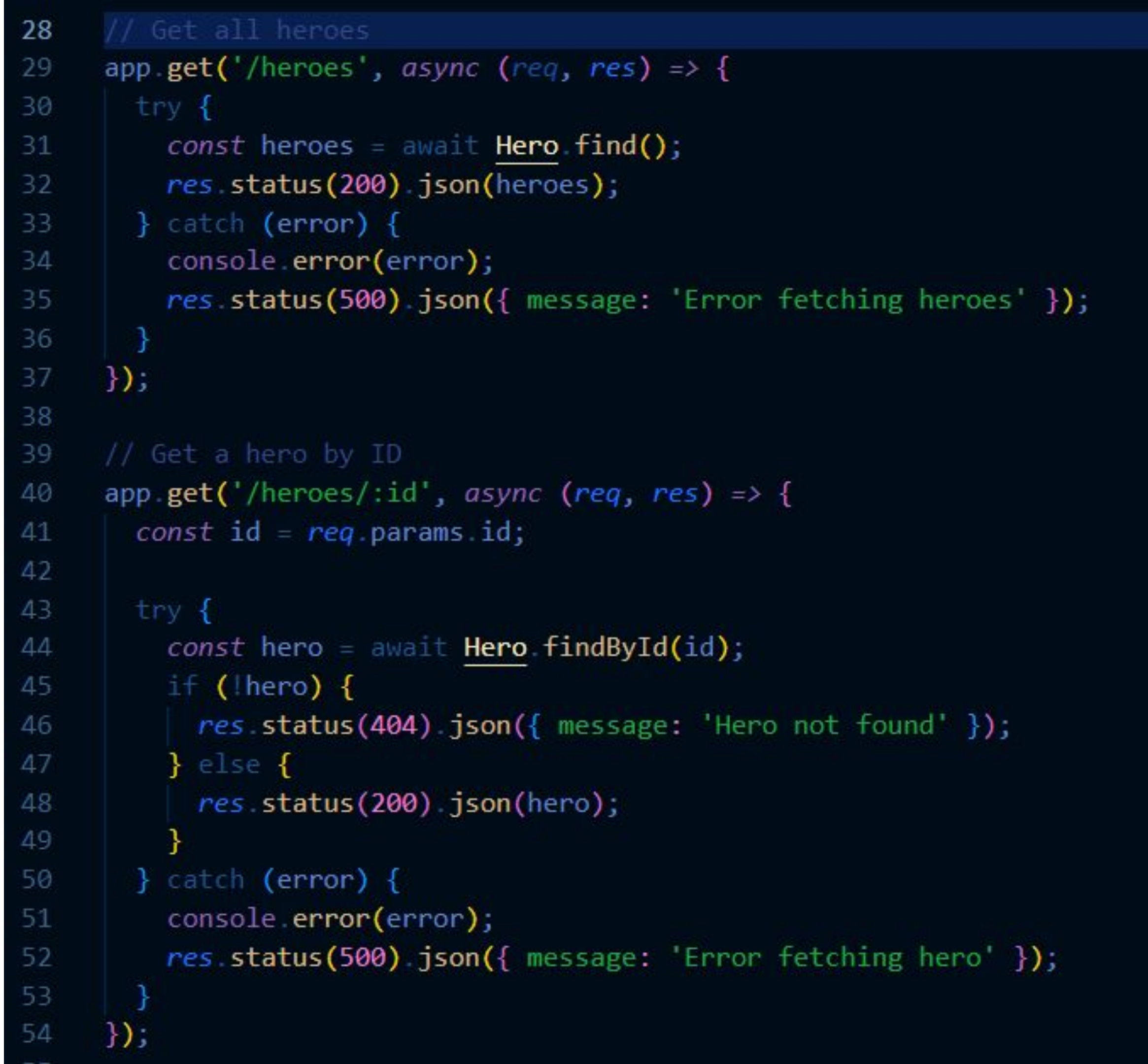
```javascript
const mongoose = require('mongoose');
const dotenv = require('dotenv');

dotenv.config();

const MONGODB_URI = process.env.MONGODB_URI;

mongoose.connect(MONGODB_URI).then(() => {
  console.log('MongoDB connected');
}).catch((err) => {
  console.error('MongoDB connection error:', err);
});

module.exports = mongoose;

```

app.js



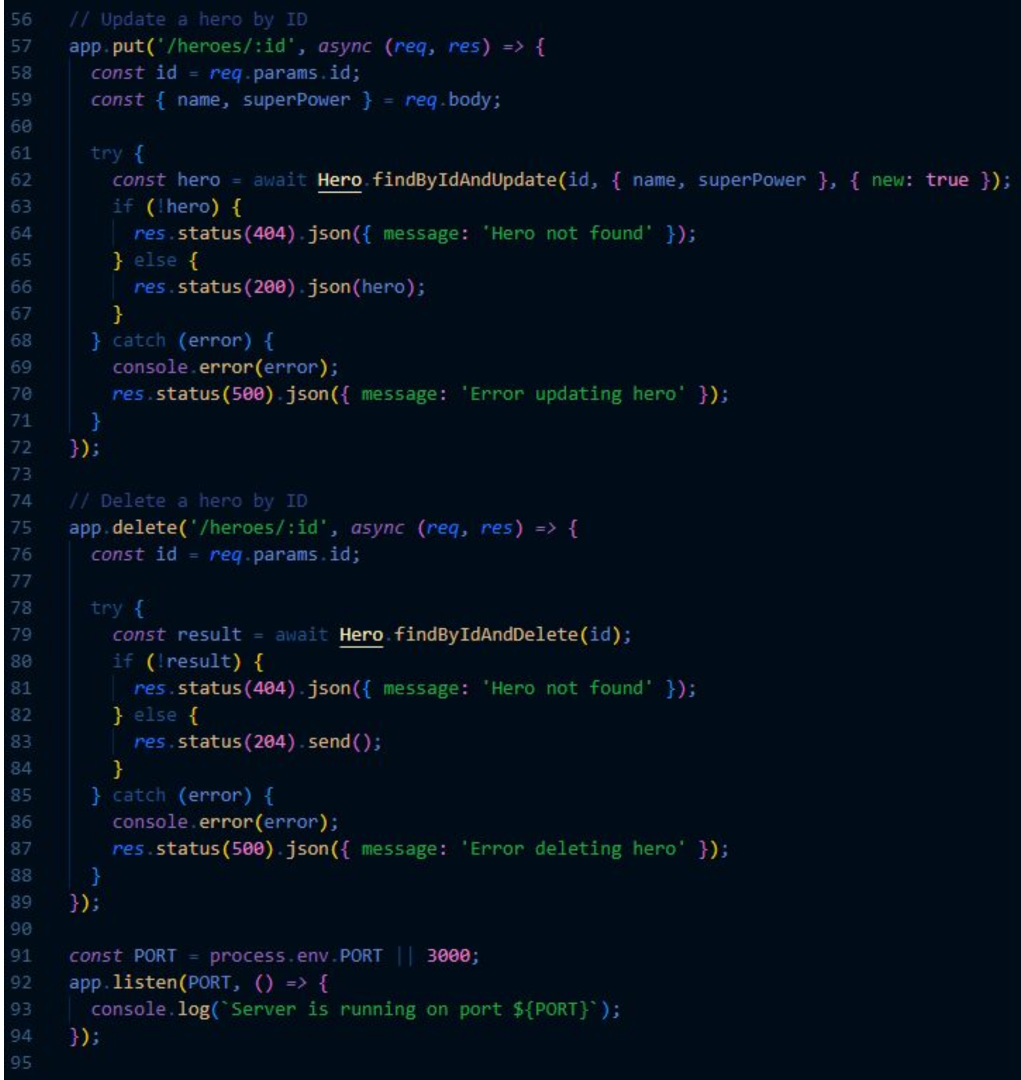
```javascript
const express = require('express');
const bodyParser = require('body-parser');
const db = require('./db');

const app = express();
app.use(bodyParser.json());

const heroSchema = new db.Schema({
  name: String,
  superPower: String,
});

const Hero = db.model('Hero', heroSchema); // Define the Hero model

// Create a new hero
app.post('/heroes', async (req, res) => {
  const { name, superPower } = req.body;

  try {
    const hero = await Hero.create({ name, superPower });
    res.status(201).json(hero);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error creating hero' });
  }
});
```

app.js



```
// Get all heroes
app.get('/heroes', async (req, res) => {
  try {
    const heroes = await Hero.find();
    res.status(200).json(heroes);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error fetching heroes' });
  }
});

// Get a hero by ID
app.get('/heroes/:id', async (req, res) => {
  const id = req.params.id;

  try {
    const hero = await Hero.findById(id);
    if (!hero) {
      res.status(404).json({ message: 'Hero not found' });
    } else {
      res.status(200).json(hero);
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error fetching hero' });
  }
});
```

app.js

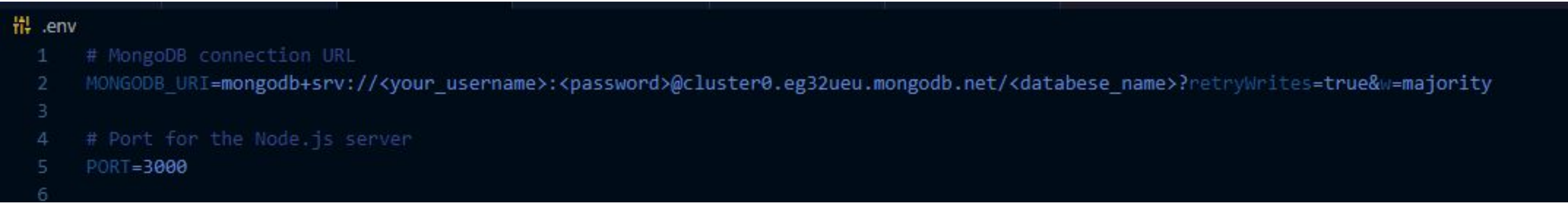


```javascript
// Update a hero by ID
app.put('/heroes/:id', async (req, res) => {
  const id = req.params.id;
  const { name, superPower } = req.body;

  try {
    const hero = await Hero.findByIdAndUpdate(id, { name, superPower }, { new: true });
    if (!hero) {
      res.status(404).json({ message: 'Hero not found' });
    } else {
      res.status(200).json(hero);
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error updating hero' });
  }
});

// Delete a hero by ID
app.delete('/heroes/:id', async (req, res) => {
  const id = req.params.id;

  try {
    const result = await Hero.findByIdAndDelete(id);
    if (!result) {
      res.status(404).json({ message: 'Hero not found' });
    } else {
      res.status(204).send();
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error deleting hero' });
  }
});

const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

.env

`./.env`

It is important to note that the '.env' file contains sensitive information, including the database connection URL and database name.



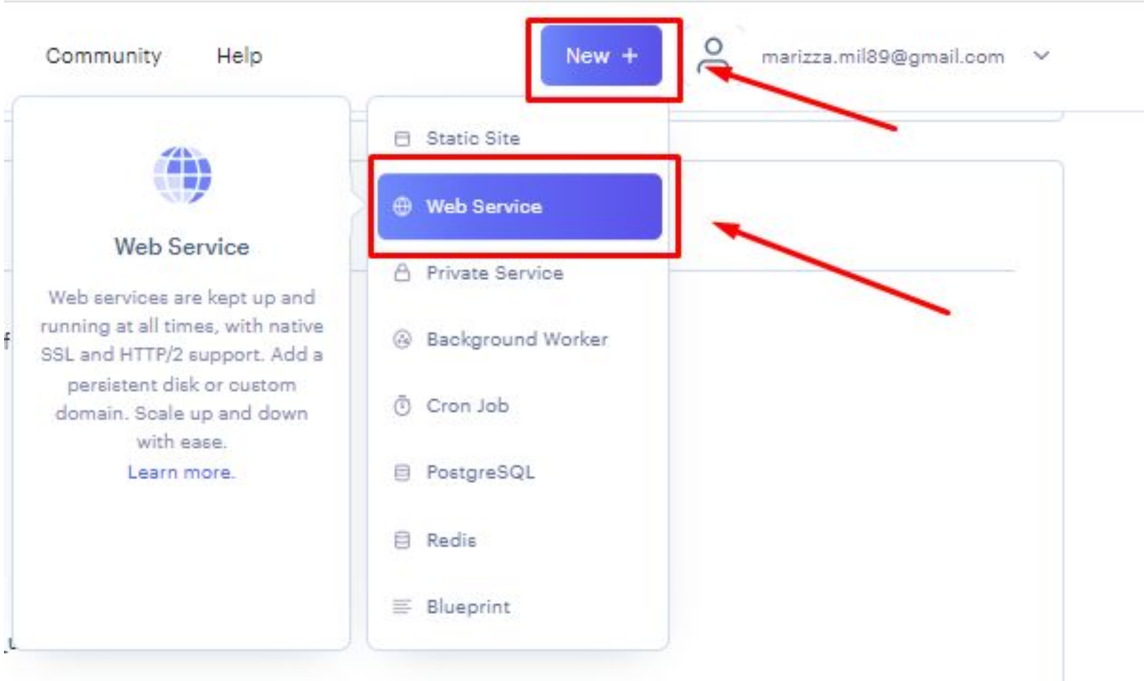
.env

```
# MongoDB connection URL
MONGODB_URI=mongodb+srv://<your_username>:<password>@cluster0.eg32ueu.mongodb.net/<databese_name>?retryWrites=true&w=majority

# Port for the Node.js server
PORT=3000

```

Deploy Project on the Render

Host the App

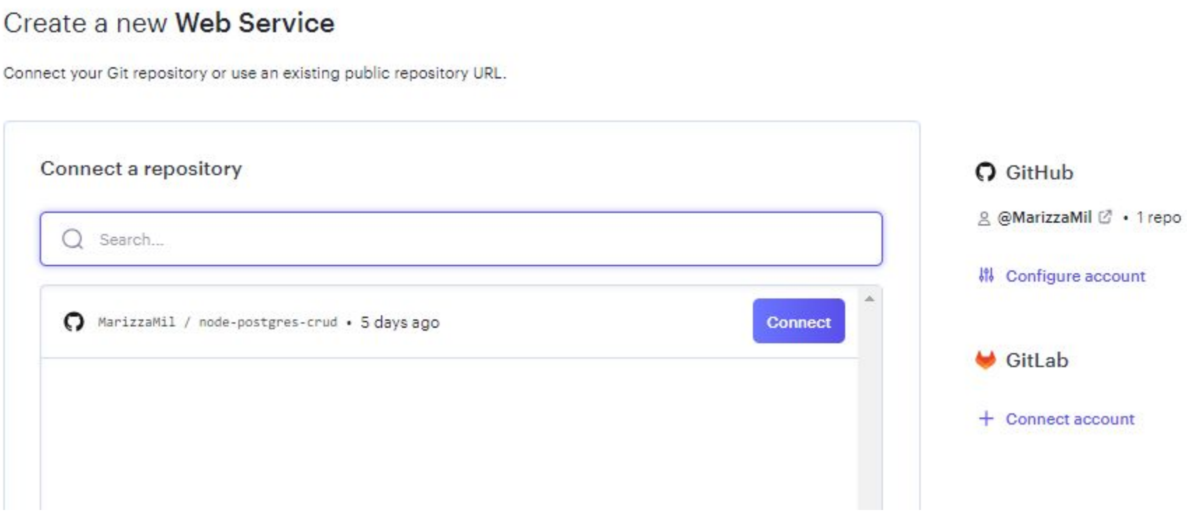
Push the code to GitHub. Now click on the **Web Service** tab from your Render dashboard:



Web services are kept up and running at all times, with native SSL and HTTP/2 support. Add a persistent disk or custom domain. Scale up and down with ease.

Host the App

Next, select the project's GitHub repository



Create a new **Web Service**

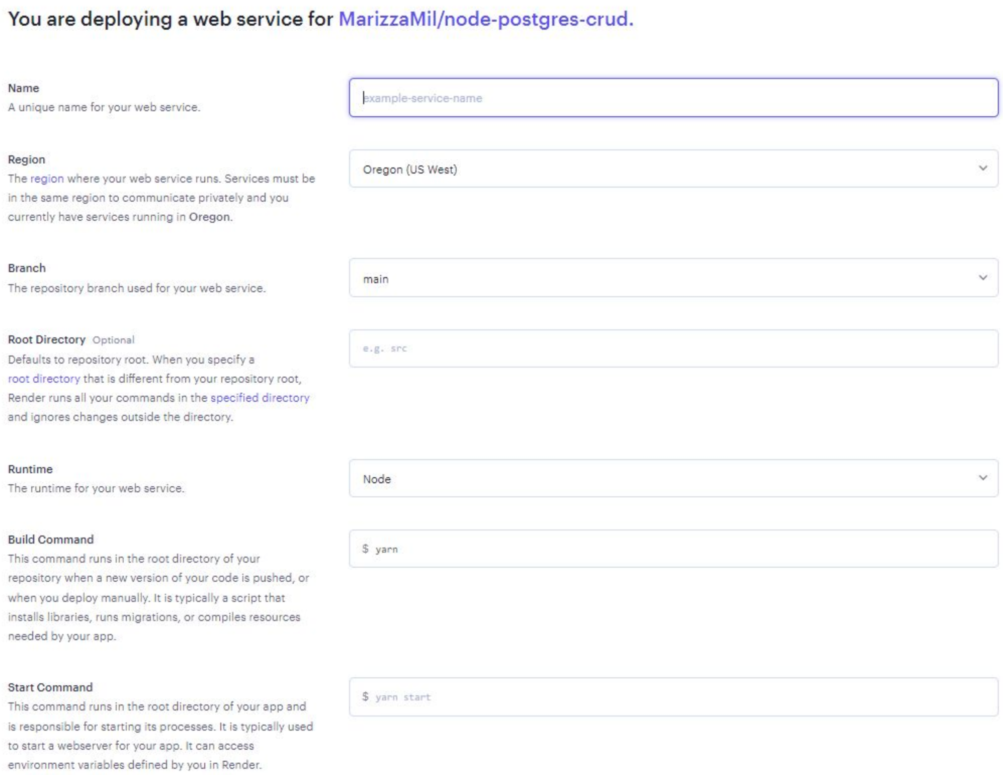
Connect your Git repository or use an existing public repository URL.

Connect a repository

Host the App

Next, enter the following details:

- Name: `my-app-demo`
- Build Command: `npm install`
- Start Command: `node app.js`
- Plan Type: `Free`



You are deploying a web service for MarizzaMil/node-postgres-crud.

Name
A unique name for your web service.
example-service-name

Region
The region where your web service runs. Services must be in the same region to communicate privately and you currently have services running in Oregon.
Oregon (US West)

Branch
The repository branch used for your web service.
main

Root Directory Optional
Defaults to repository root. When you specify a root directory that is different from your repository root, Render runs all your commands in the specified directory and ignores changes outside the directory.
e.g. src

Runtime
The runtime for your web service.
Node

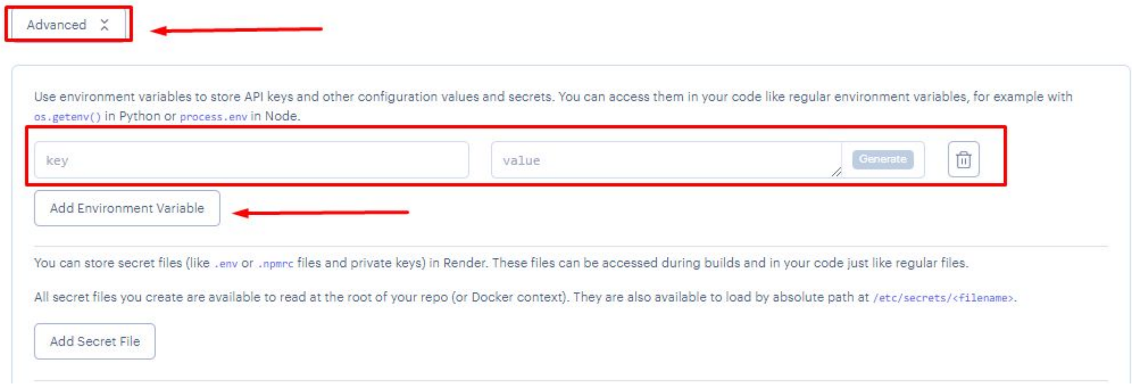
Build Command
This command runs in the root directory of your repository when a new version of your code is pushed, or when you deploy manually. It is typically a script that installs libraries, runs migrations, or compiles resources needed by your app.
$ yarn

Start Command
This command runs in the root directory of your app and is responsible for starting its processes. It is typically used to start a webserver for your app. It can access environment variables defined by you in Render.
$ yarn start

Host the App

Then scroll down and click on the Advanced button, click on the Add Environment Variable and add the following database credentials from your Render database for:

```
MONGODB_URI = <CONNECTION STRING>
```



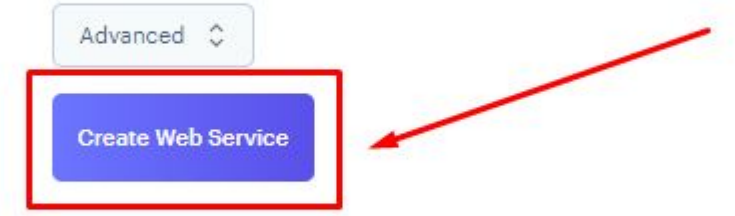
Use environment variables to store API keys and other configuration values and secrets. You can access them in your code like regular environment variables, for example with `os.getenv()` in Python or `process.env` in Node.

You can store secret files (like `.env` or `.npmrc` files and private keys) in Render. These files can be accessed during builds and in your code just like regular files.

All secret files you create are available to read at the root of your repo (or Docker context). They are also available to load by absolute path at `/etc/secrets/<filename>`.

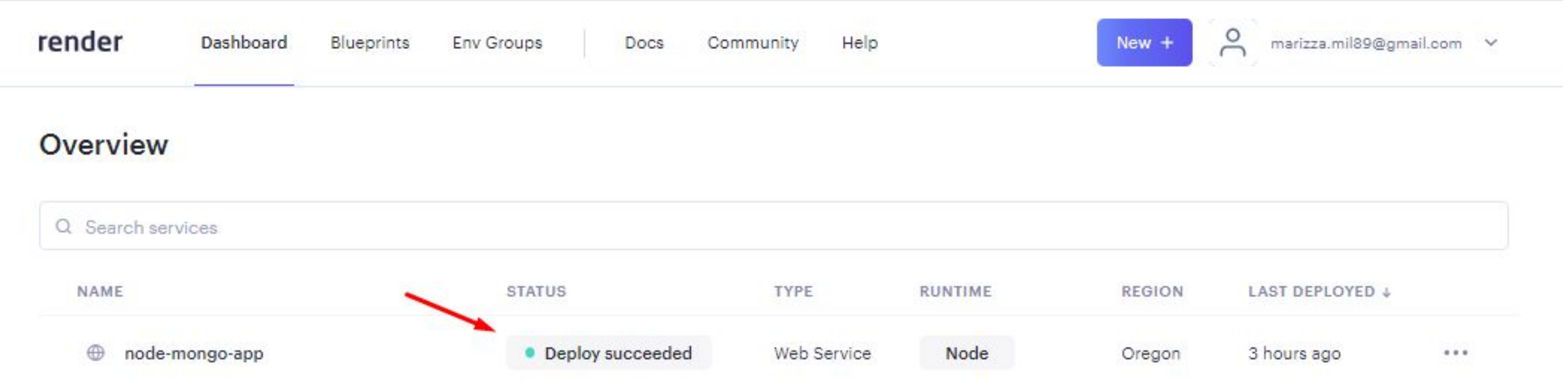
Host the App

Finally, click the Create Service button and wait for the application deployment to complete.



Host the App

Once the deployment is finished, the application status will show `Deploy succeeded`.

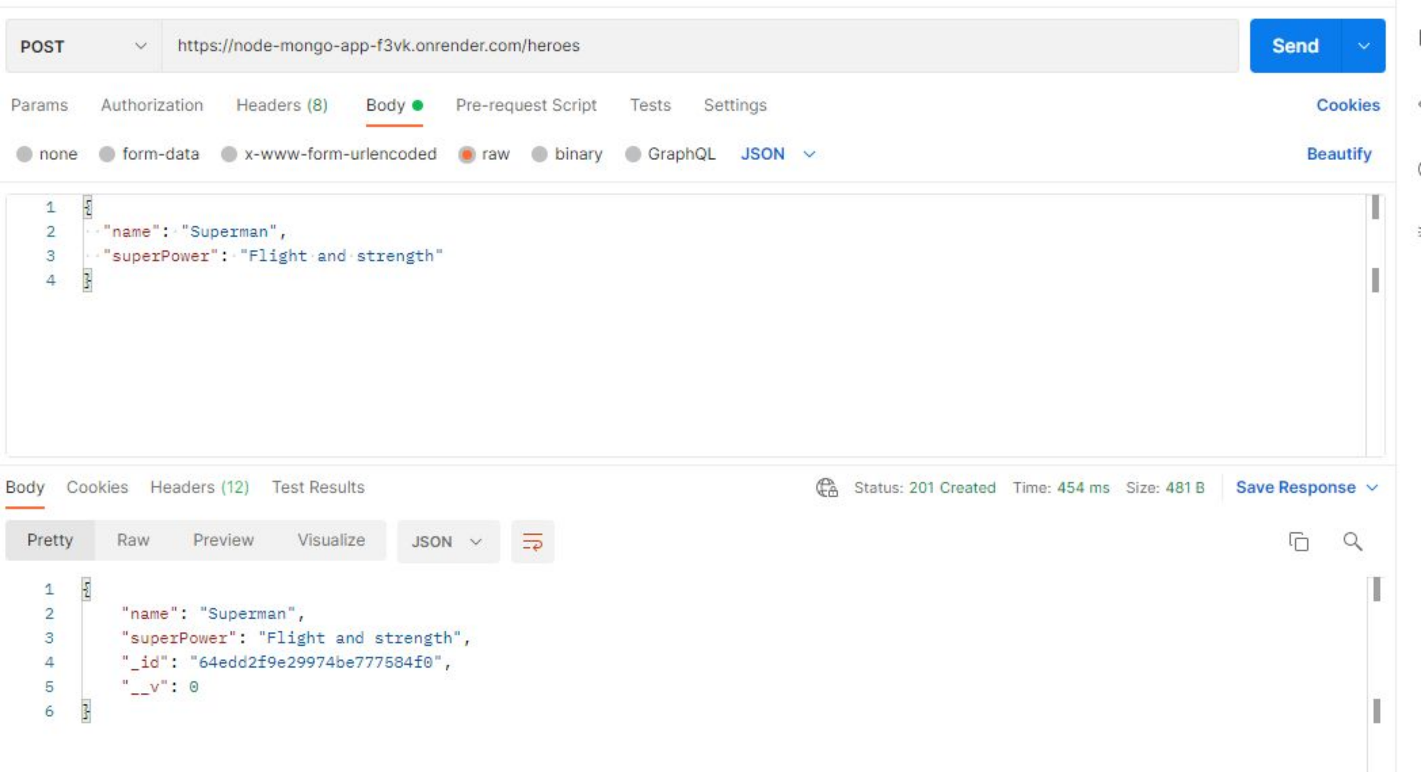


Overview

NAME	STATUS	TYPE	RUNTIME	REGION	LAST DEPLOYED ↓
node-mongo-app	● Deploy succeeded	Web Service	Node	Oregon	3 hours ago

Test the API using Postman

POST_REQUEST



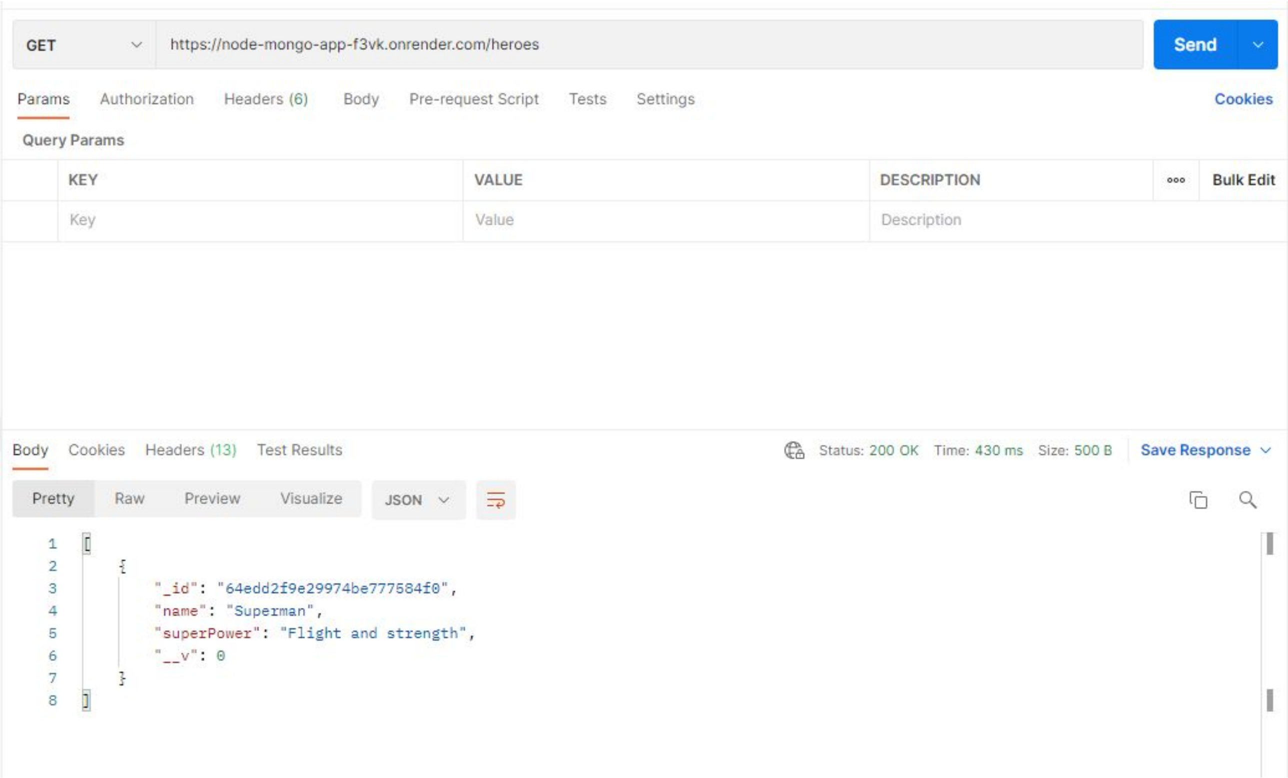
POST https://node-mongo-app-f3vk.onrender.com/heroes

```
{
  "name": "Superman",
  "superPower": "Flight and strength"
}
```

Status: 201 Created Time: 454 ms Size: 481 B

```
{
    "name": "Superman",
    "superPower": "Flight and strength",
    "_id": "64edd2f9e29974be777584f0",
    "__v": 0
}
```

GET REQUEST

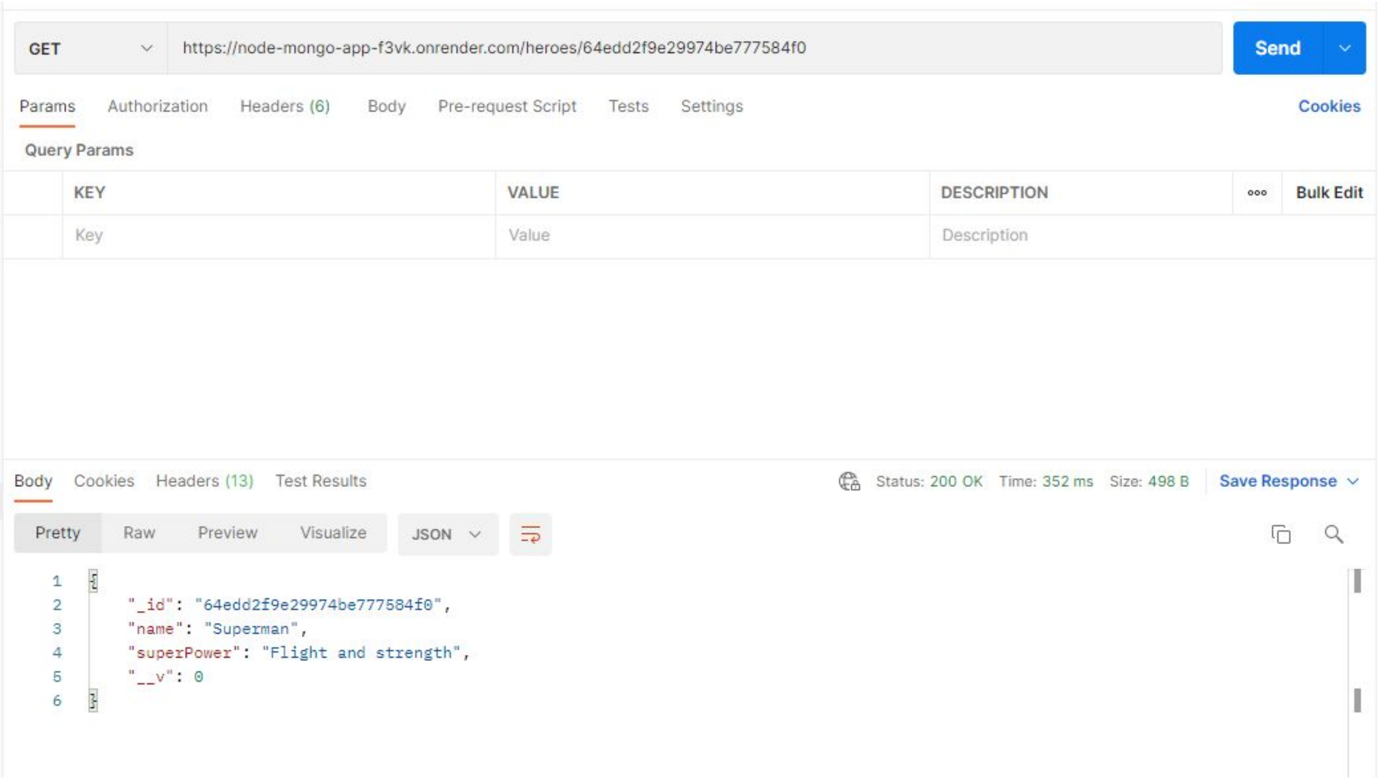


https://node-mongo-app-f3vk.onrender.com/heroes

Status: 200 OK Time: 430 ms Size: 500 B

```
[
    {
        "_id": "64edd2f9e29974be777584f0",
        "name": "Superman",
        "superPower": "Flight and strength",
        "__v": 0
    }
]
```

GET_BY ID REQUEST



GET https://node-mongo-app-f3vk.onrender.com/heroes/64edd2f9e29974be777584f0

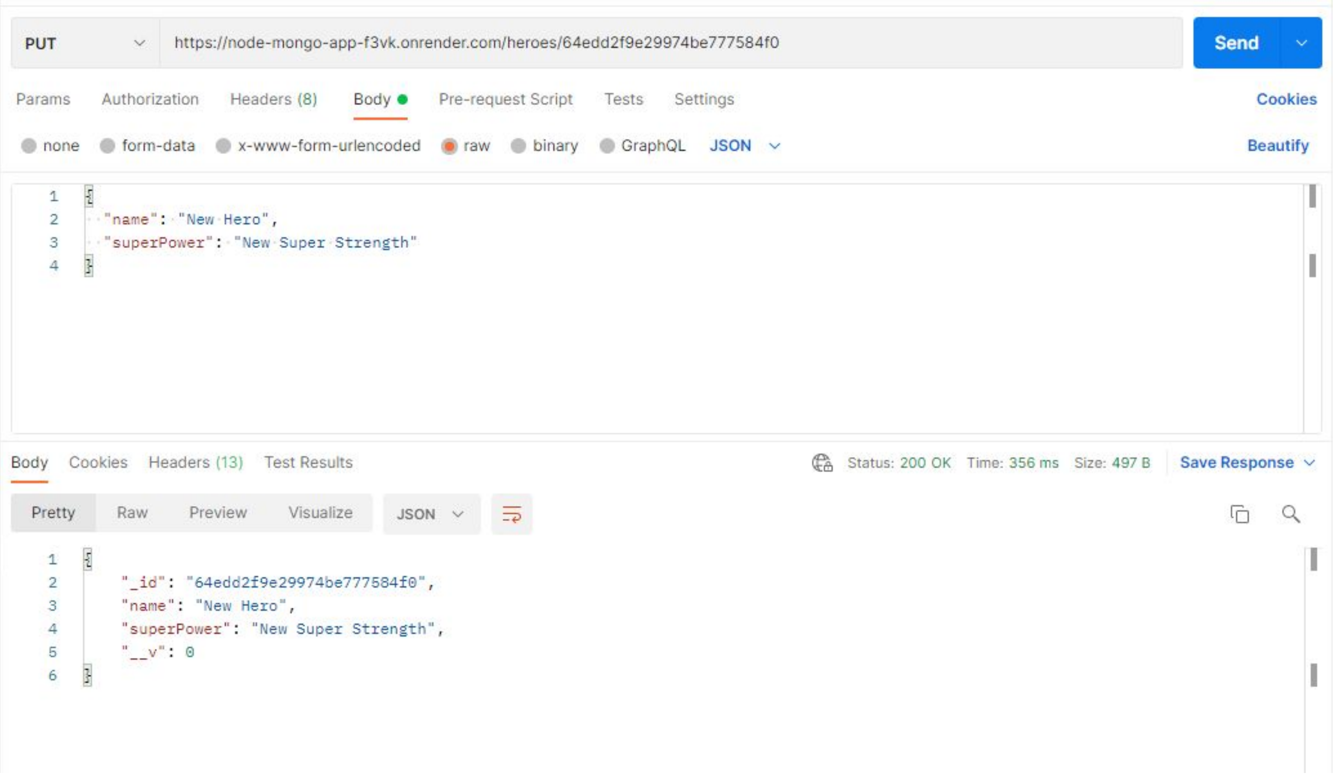
Query Params

	KEY	VALUE	DESCRIPTION	ooo	Bulk Edit
	Key	Value	Description		

Status: 200 OK Time: 352 ms Size: 498 B

```
{
    "_id": "64edd2f9e29974be777584f0",
    "name": "Superman",
    "superPower": "Flight and strength",
    "__v": 0
}
```

UPDATE_REQUEST



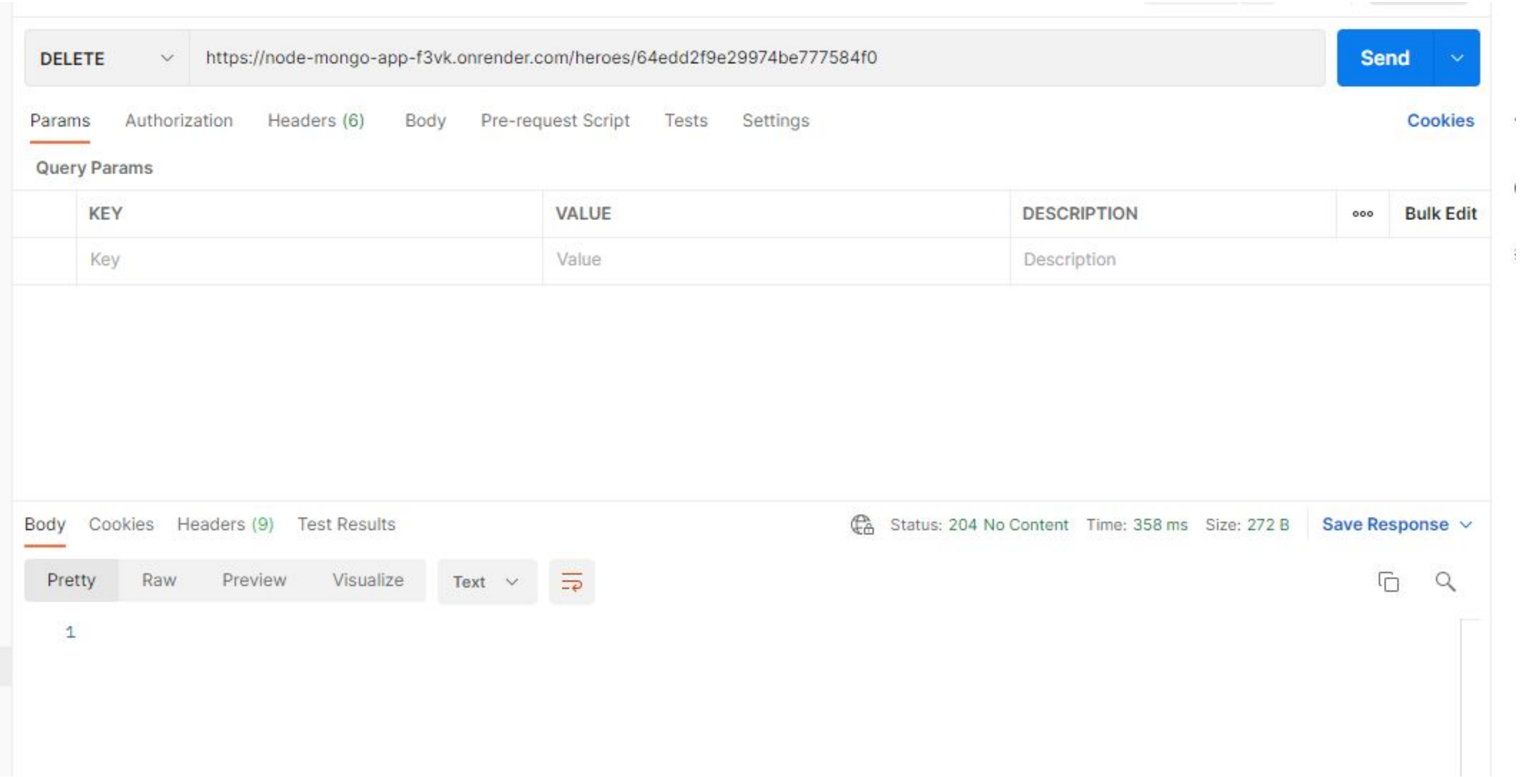
PUT https://node-mongo-app-f3vk.onrender.com/heroes/64edd2f9e29974be777584f0

```
{
  "name": "New Hero",
  "superPower": "New Super Strength"
}
```

Status: 200 OK Time: 356 ms Size: 497 B

```
{
    "_id": "64edd2f9e29974be777584f0",
    "name": "New Hero",
    "superPower": "New Super Strength",
    "__v": 0
}
```

DELETE_REQUEST



DELETE https://node-mongo-app-f3vk.onrender.com/heroes/64edd2f9e29974be777584f0

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 204 No Content Time: 358 ms Size: 272 B

```
1
```

Conclusion

In conclusion, the our project demonstrates the process of building a robust and scalable web application using Node.js, Express.js, and MongoDB, and then deploying it on the Render platform. Throughout the project, we learned how to:

- Develop a RESTful API using Express.js: We built API routes to handle CRUD (Create, Read, Update, Delete) operations for interacting with a MongoDB database. Express.js provided a powerful framework for defining endpoints and managing requests and responses.
- Establish a Connection to MongoDB: We utilized the Mongoose library to establish a connection to a MongoDB database, manage database models and schemas, and perform database operations efficiently.
- Manage Environment Variables: By using a .env file and the dotenv library, we securely stored sensitive configuration details such as database credentials and API keys outside of the codebase.
- Deploy on Render: The project concluded with deploying the application on the Render platform. This process involved configuring the application environment, handling the deployment pipeline, and ensuring the MongoDB Atlas IP Whitelist was set up to allow the application to access the database.
- Handling Deployment Issues: We encountered and resolved common deployment issues, such as connection timeouts and IP Whitelist restrictions, demonstrating the importance of troubleshooting and maintaining a reliable application.

By completing the "Deploy Node Express MongoDB on Render" project, we gained practical experience in building and deploying real-world applications, integrating databases, and ensuring security considerations. This project serves as a foundation for further learning and exploration of web development and deployment technologies.