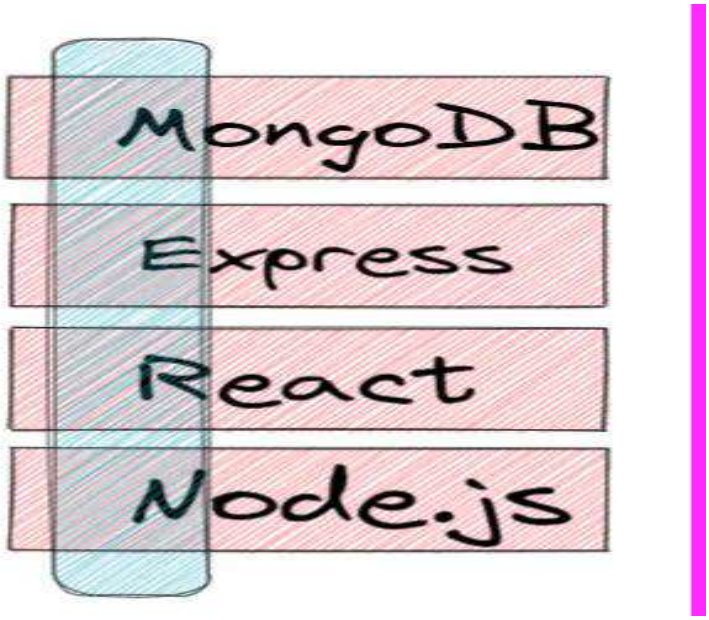
Create Blog website using MERN stack

Introduction

The MERN stack is a popular set of technologies for creating a modern Single Page Application (SPA). MERN stands for MongoDB, Express, React, and Node.js:

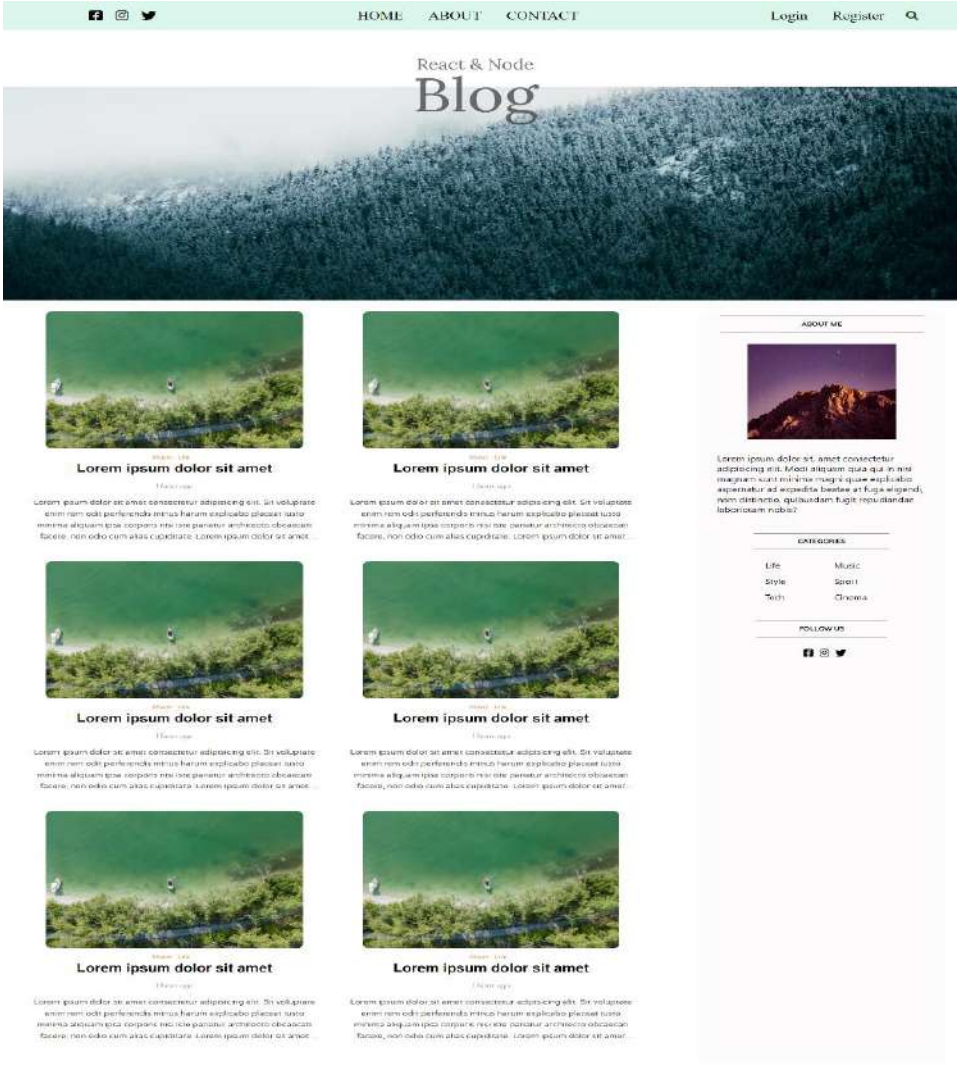
- **Node.js** is a popular server-side framework that allows us to run JavaScript code on a web server.
- **Express** is a Node.js web application framework that makes Node application development simpler and faster.
- **MongoDB** is a NoSQL database that stores data persistently in the form of collections and documents.
- **React** is a JavaScript frontend library for creating user interfaces.



What you will learn

After completing this project, you will be able to create a full-stack blog application that performs CRUD operations by utilizing the MERN stack. This blog tutorial should help you understand the fundamental operations of the MERN stack.

Final Design:



Pre-requisites:

- Basic understanding about HTML , CSS, JSX , React , Node and MongoDB.
- VS code or any other code editor system.
- NodeJS installed on your system.

How to download and install VS code
https://code.visualstudio.com/download

How to Download and Install NodeJS
https://nodejs.org/en/download/

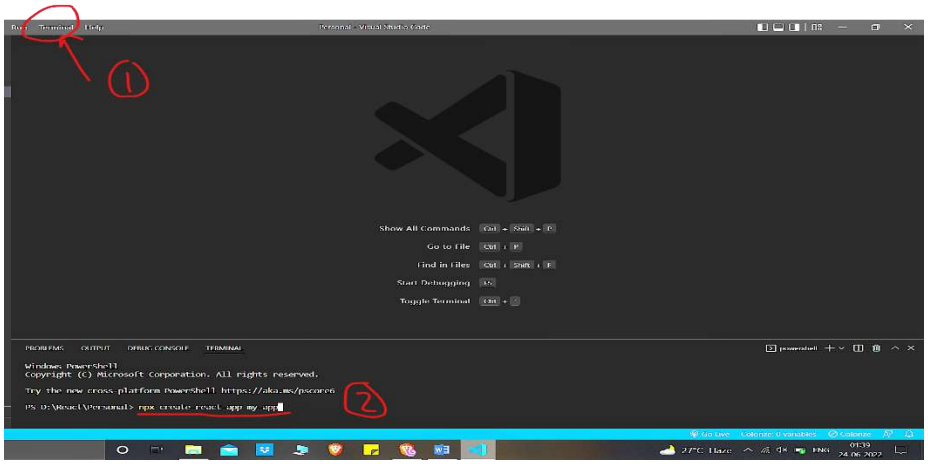
Implementation Step

Firstly we will create front-end , and back-end and database afterwards.

Create react app

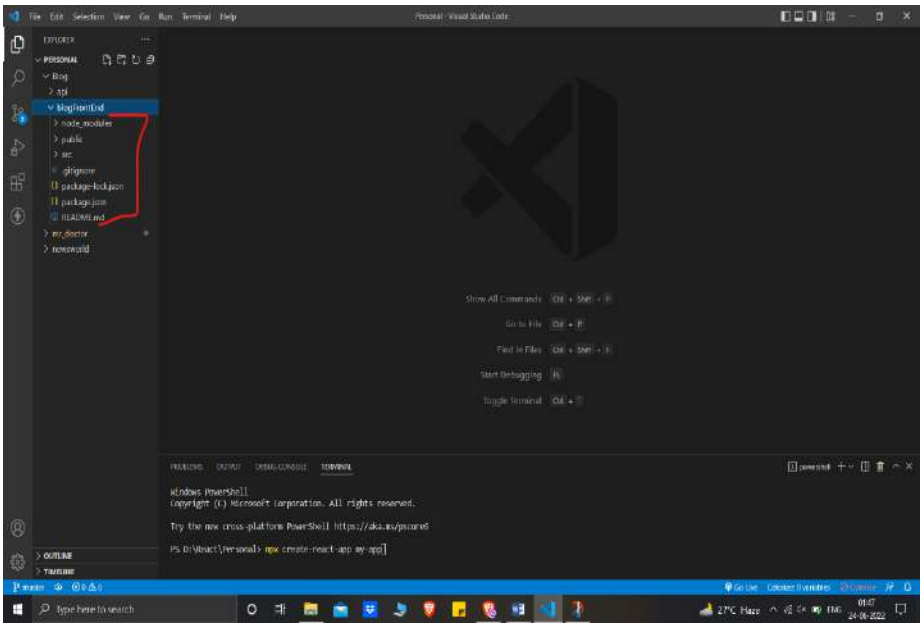
Under desired folder , open VS code and in terminal , type :

npx create-react-app my-app



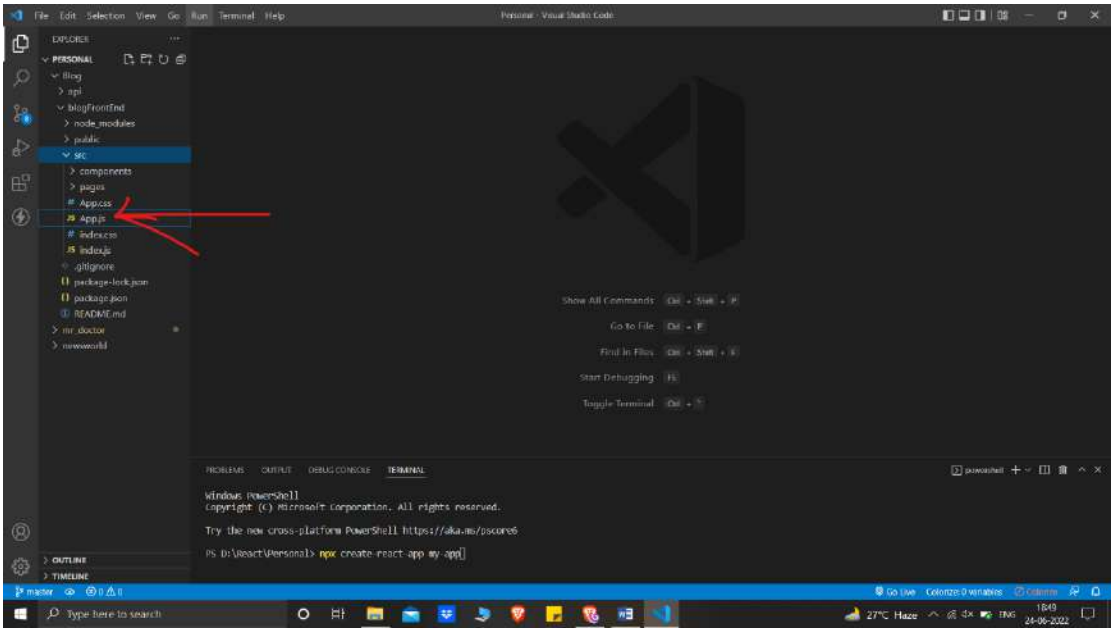
Instead of my-app you can type name of the app you want to create.

After successfull creation of react app , you should be able to view a folder structure like this

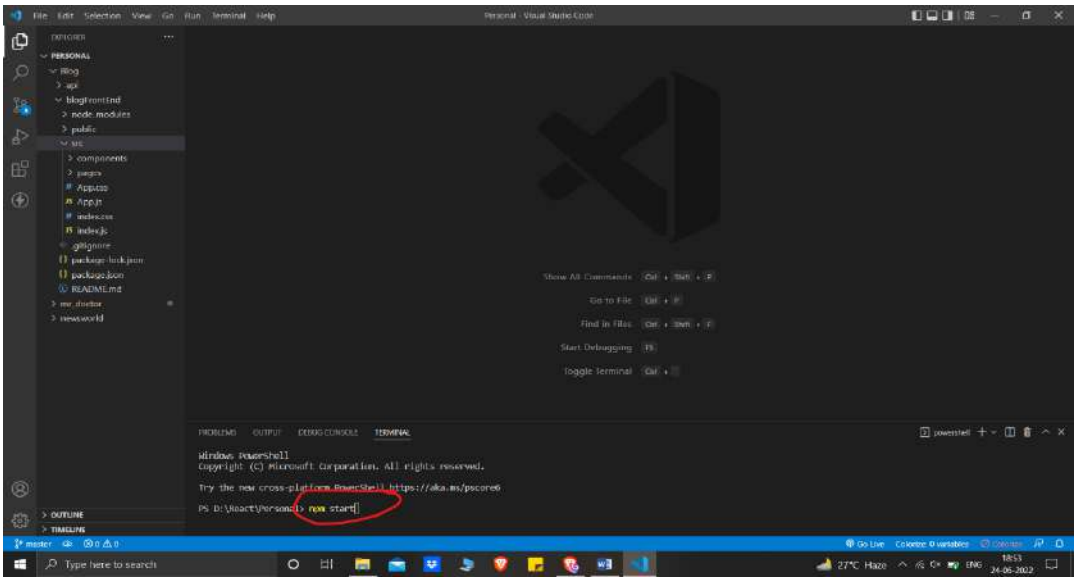


Now inside src folder of react , your will see app.js file

It is the entry point of our react application.

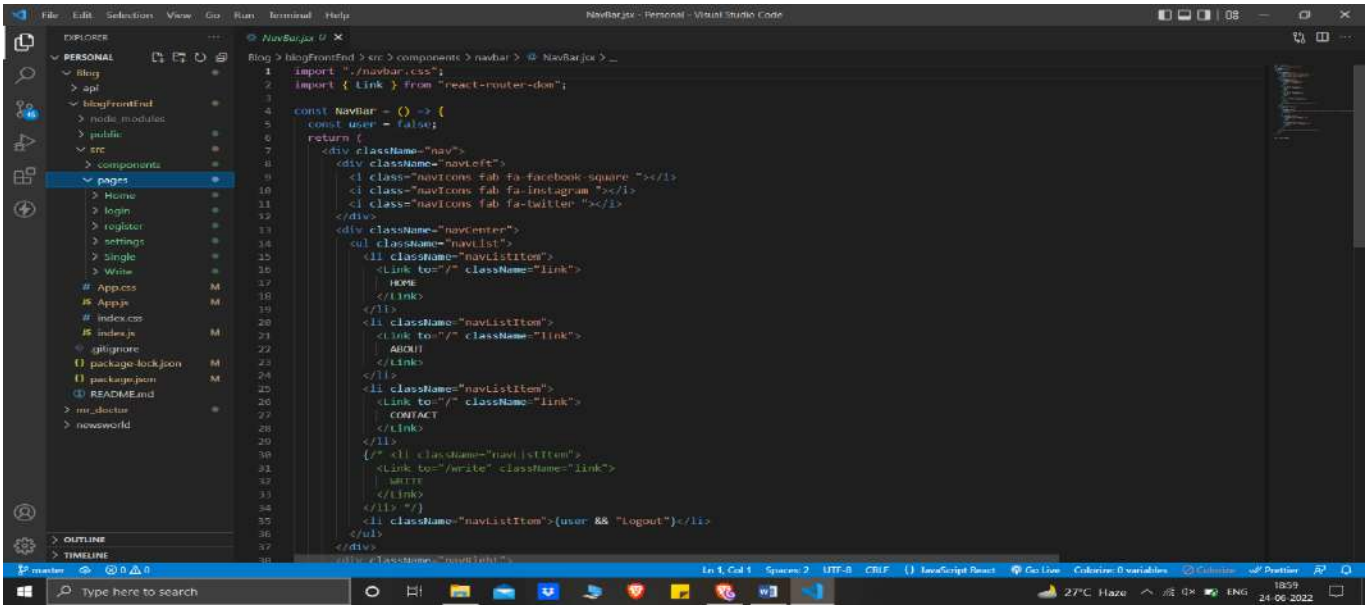


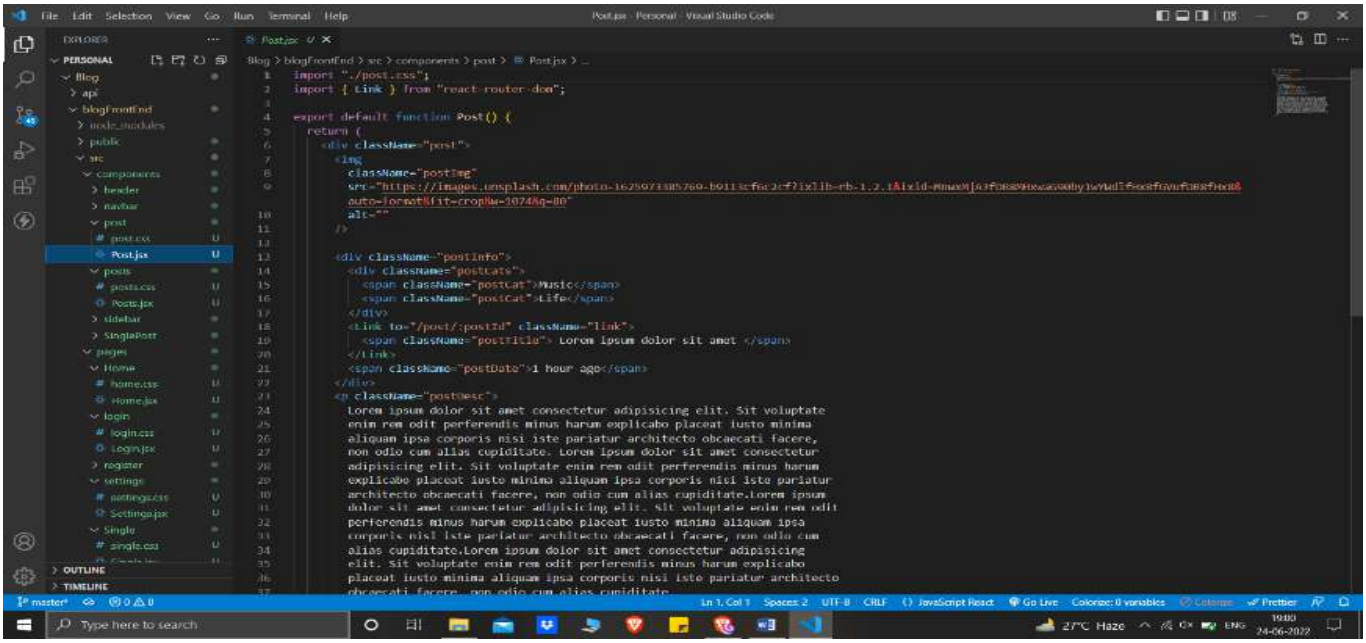
Now to run your react app , simply type npm start or npm run start in the terminal and hit enter :



Now wait for sometine , you will be redirected in your browser , where you can see a dummy react applcation.

Now just start with creating basic wesite structure using knowledge of HTML , CSS and JSX ,

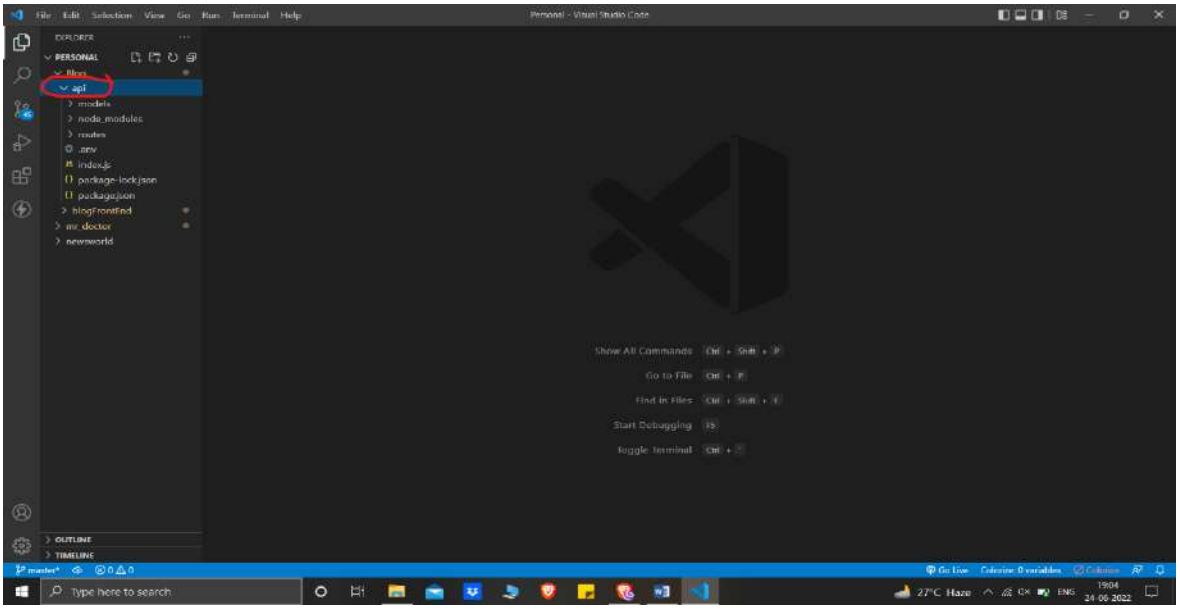




If you need reference regarding front-end , you can visit to
below git repository to see code of front-end

https://github.com/DEVcoder0602/Blog_mern/tree/front-end

Now For backend and database , create a separate folder like api in our case :



Here you will have to install some npm packages like : node , express , mongoose , etc. for that open terminal and run below command :

npm install express mongoose dotenv multer nodemon

Here we are isntalling 4 packages :

1. ***Express*** - Express is a node js web application framework that provides broad features for building web and mobile applications.

   Official website : https://expressjs.com

2. ***Mongoose*** – It is used to connect our backend with database. Mongoose acts as a front end to MongoDB

   Official website : https://mongoosejs.com

3. ***Dotenv*** - It is used to hide uneccesary info inside our main file. For eg : we can use .env file to store secret keys , ids or any other sesitive data and directly call that data from .env file , without actually revealing the data.

   Official website : https://www.npmjs.com/package/dotenv

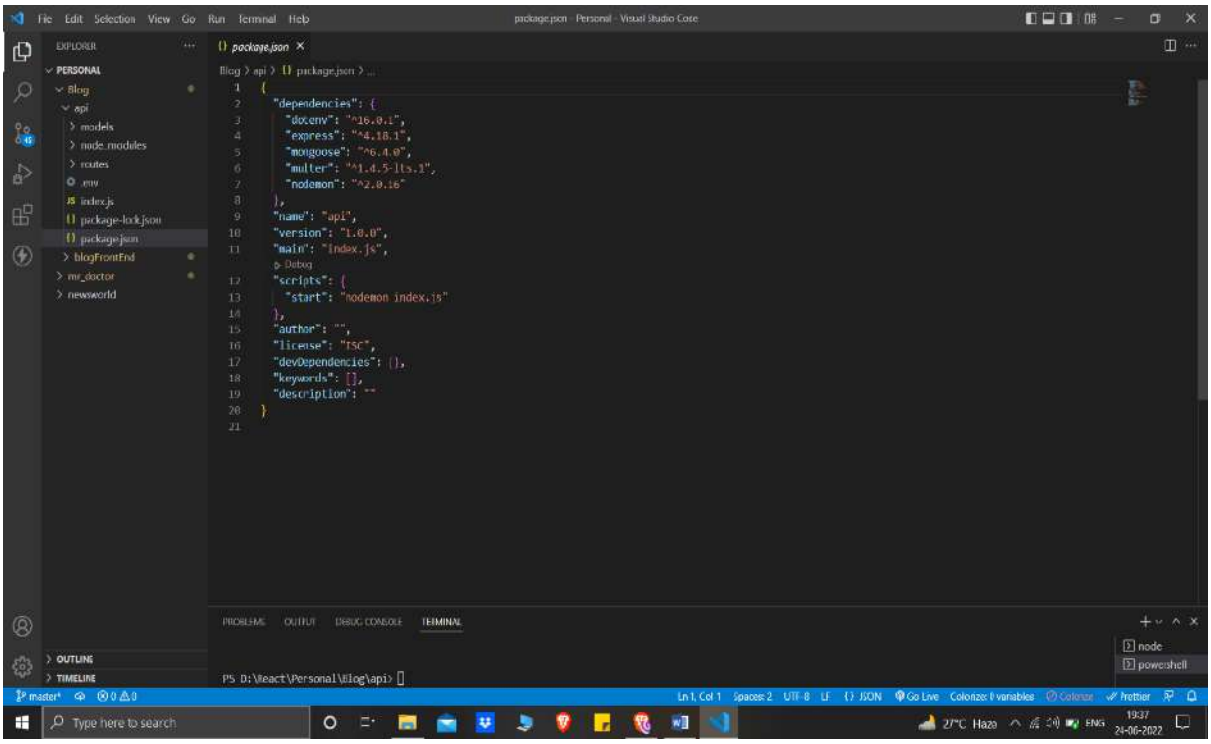
4. ***Multer*** – Used for uploading files.

Official website : https://www.npmjs.com/package/multer

5. ***Nodemon*** – It is a tool that helps develop Node.js based applications by automatically restarting the node application when file changes in the directory are detected

   Official website : https://nodemon.io

Now your package.json file shoul look like this :



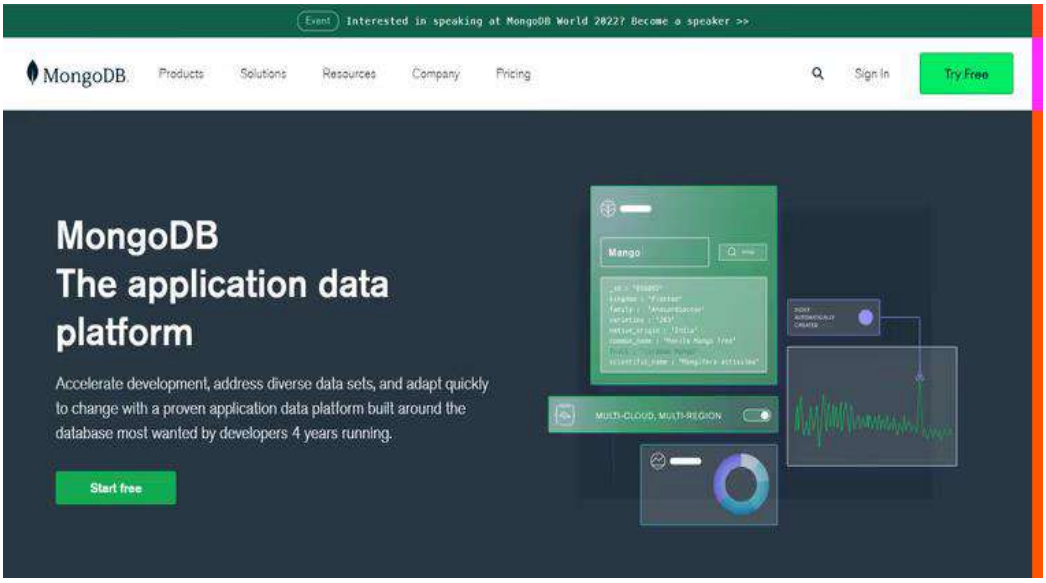
Now create a new file index.js inside api folder only .

Configuring index.js

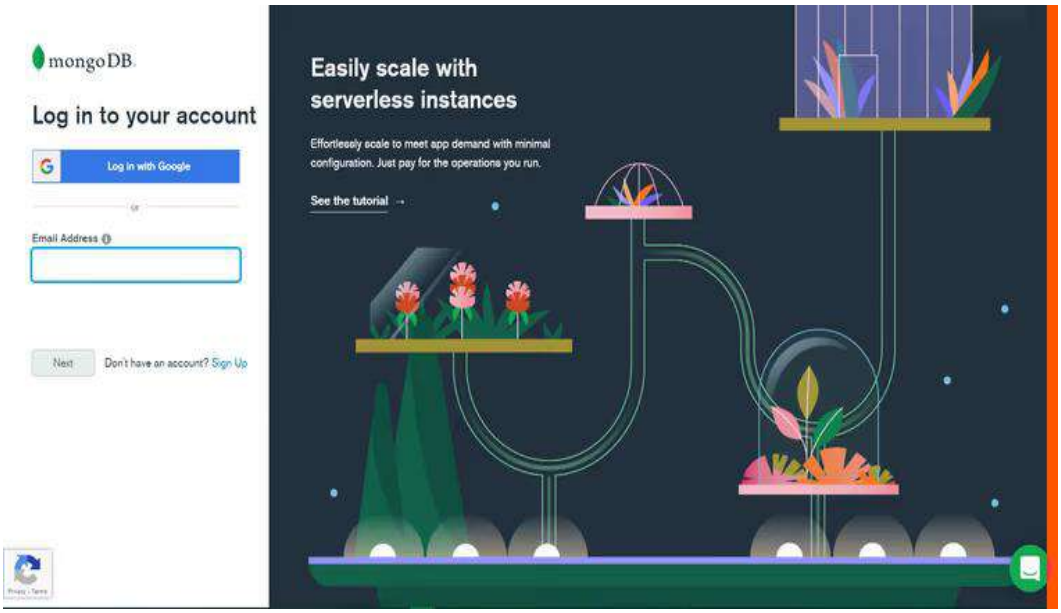
```
• const express = require("express");
• const app = express();
• const dotenv = require("dotenv");
• const mongoose = require("mongoose");
```

Now it's time to link our server application to the real database, so we'll utilize the MongoDB database, especially the MongoDB cloud Atlas version, which means we'll be hosting our database onto their cloud.

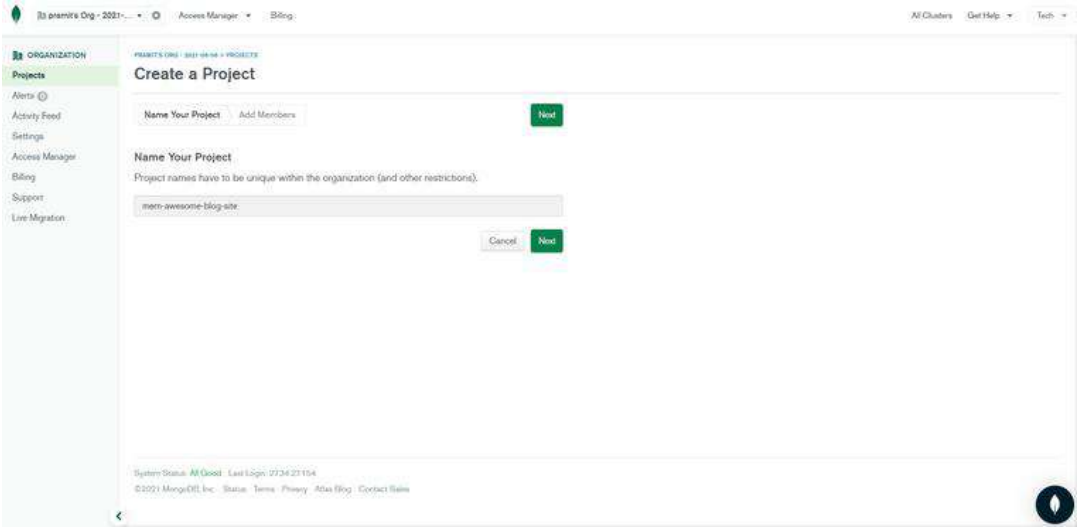
Official MongoDB website



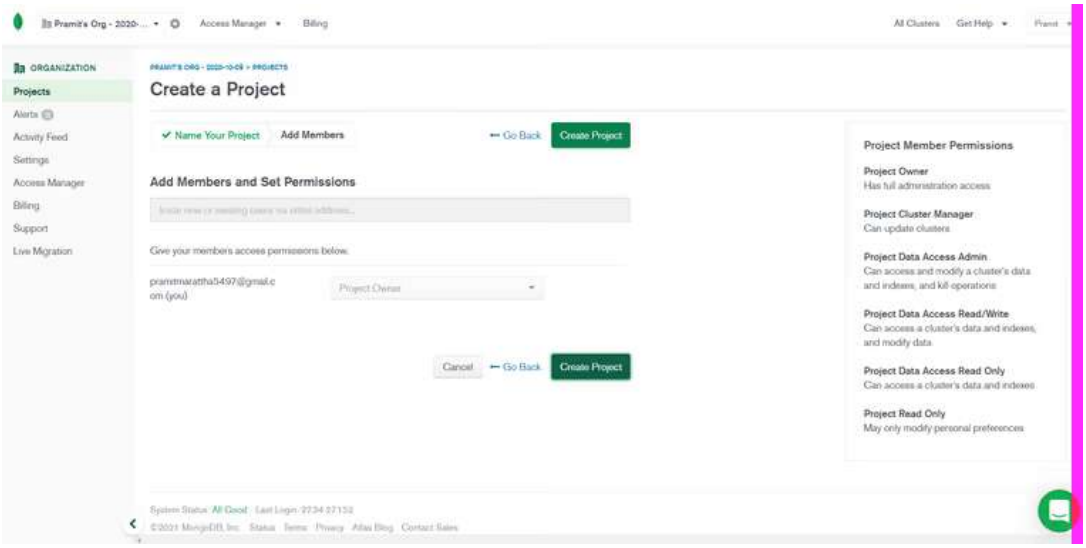
Sign in to MongoDB



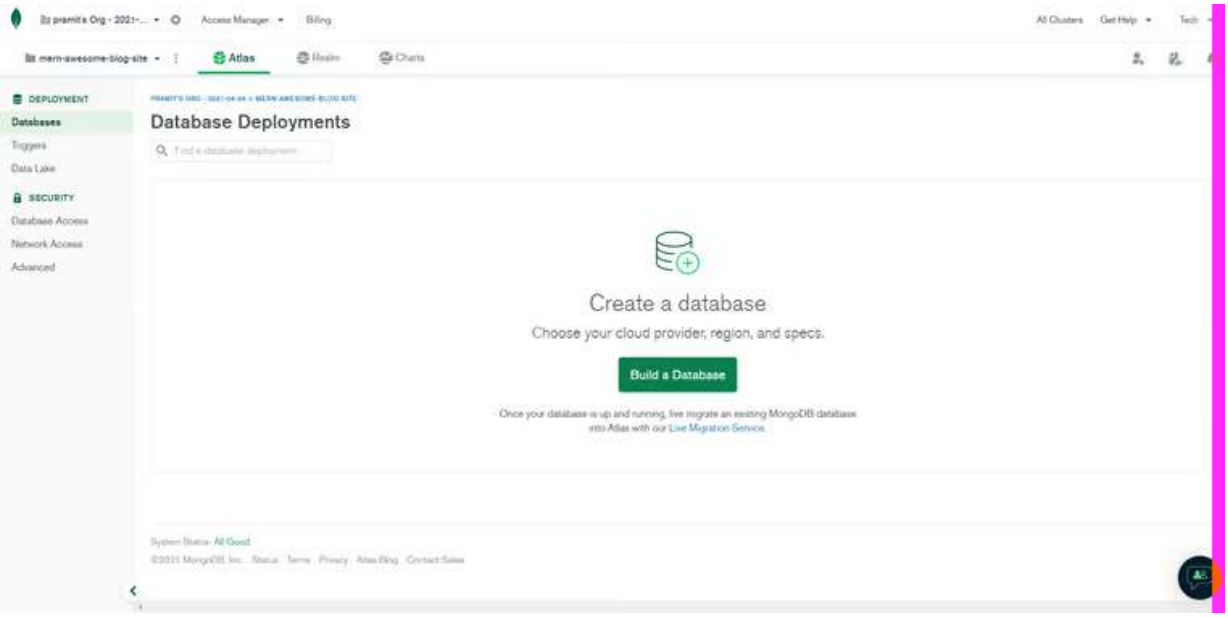
Create a Project



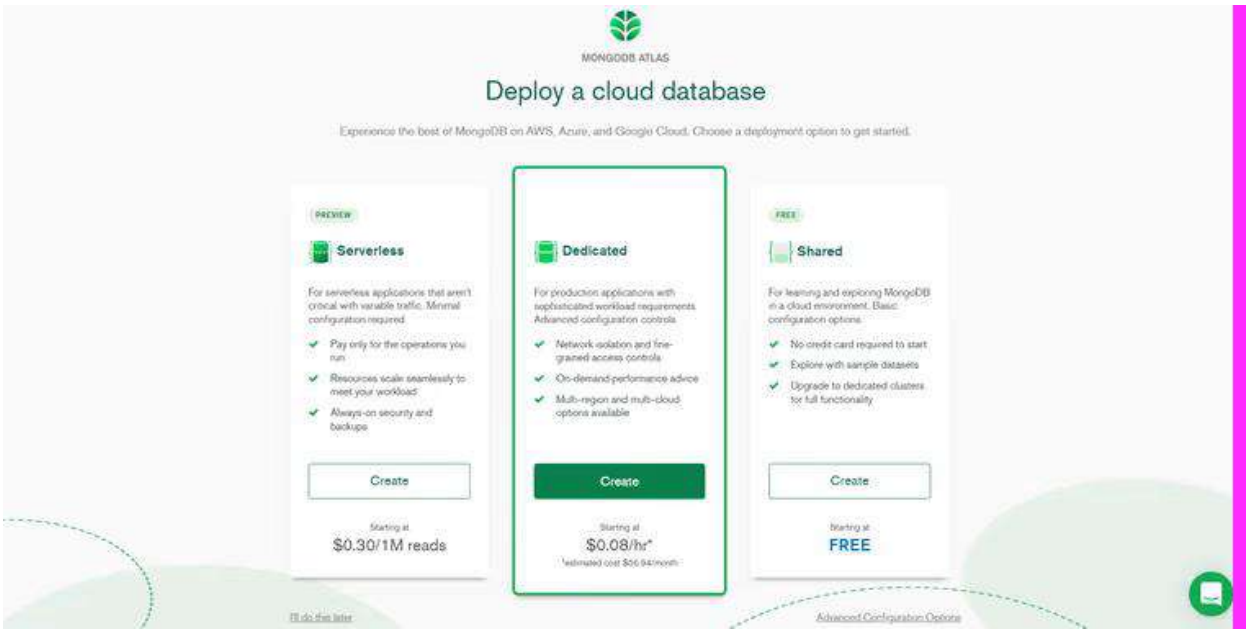
Adding members



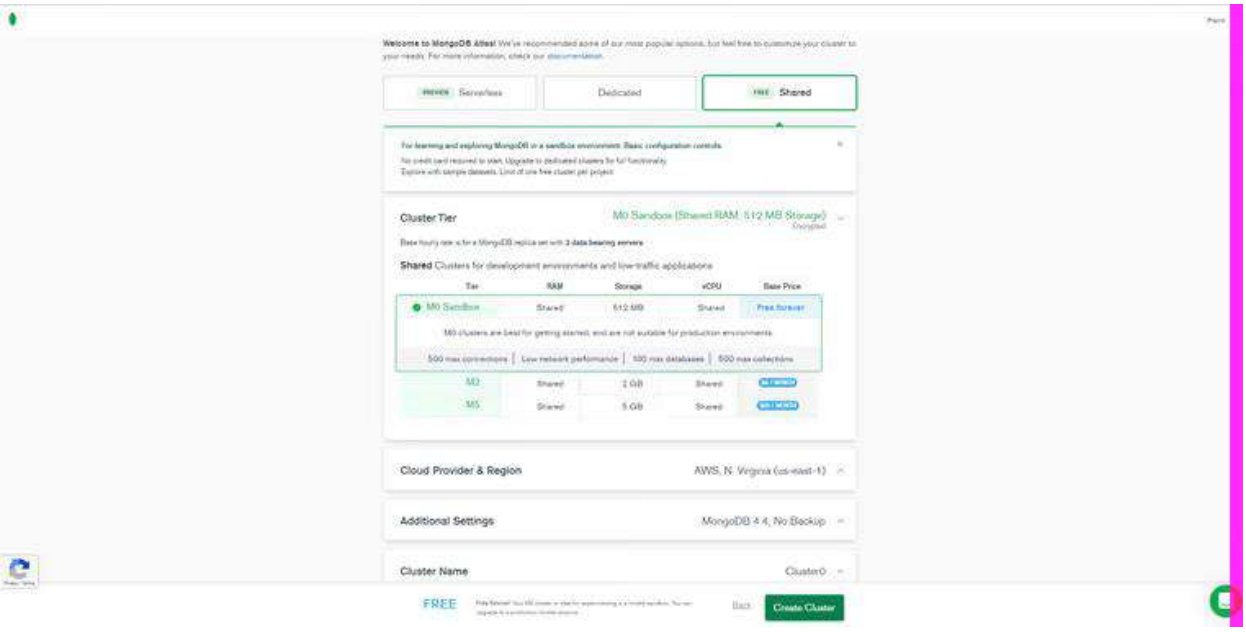
Building a database



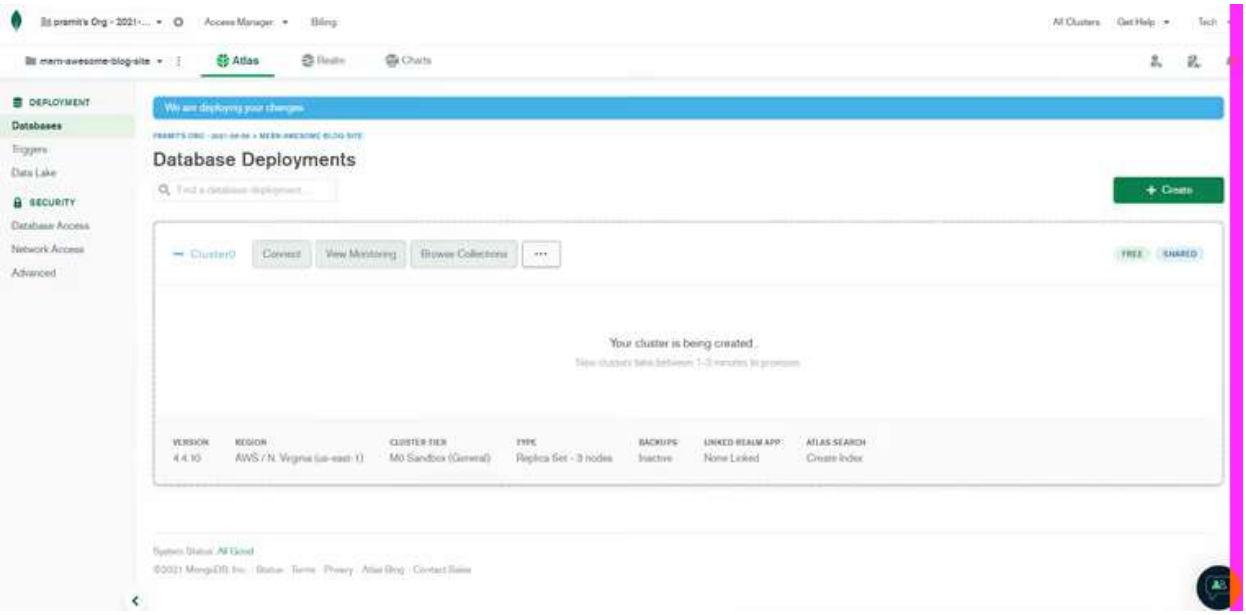
Creating a cluster



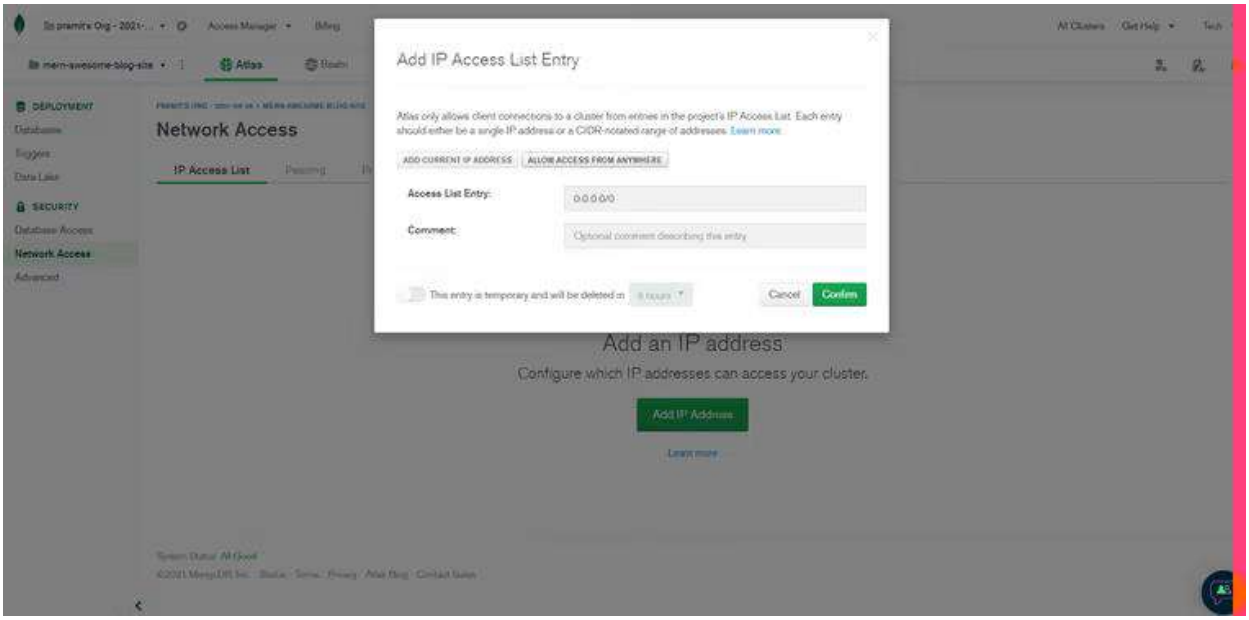
Selecting a cloud service provider



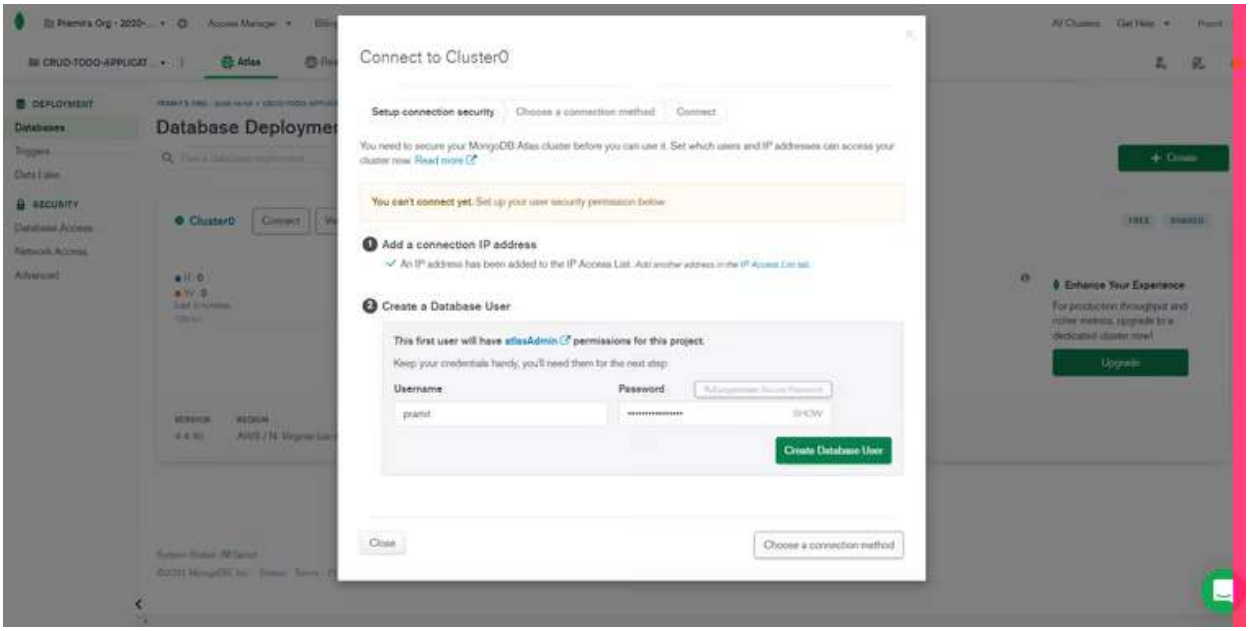
Make a cluster and wait for the cluster to be built before proceeding (usually takes around 5 -10 minutes)



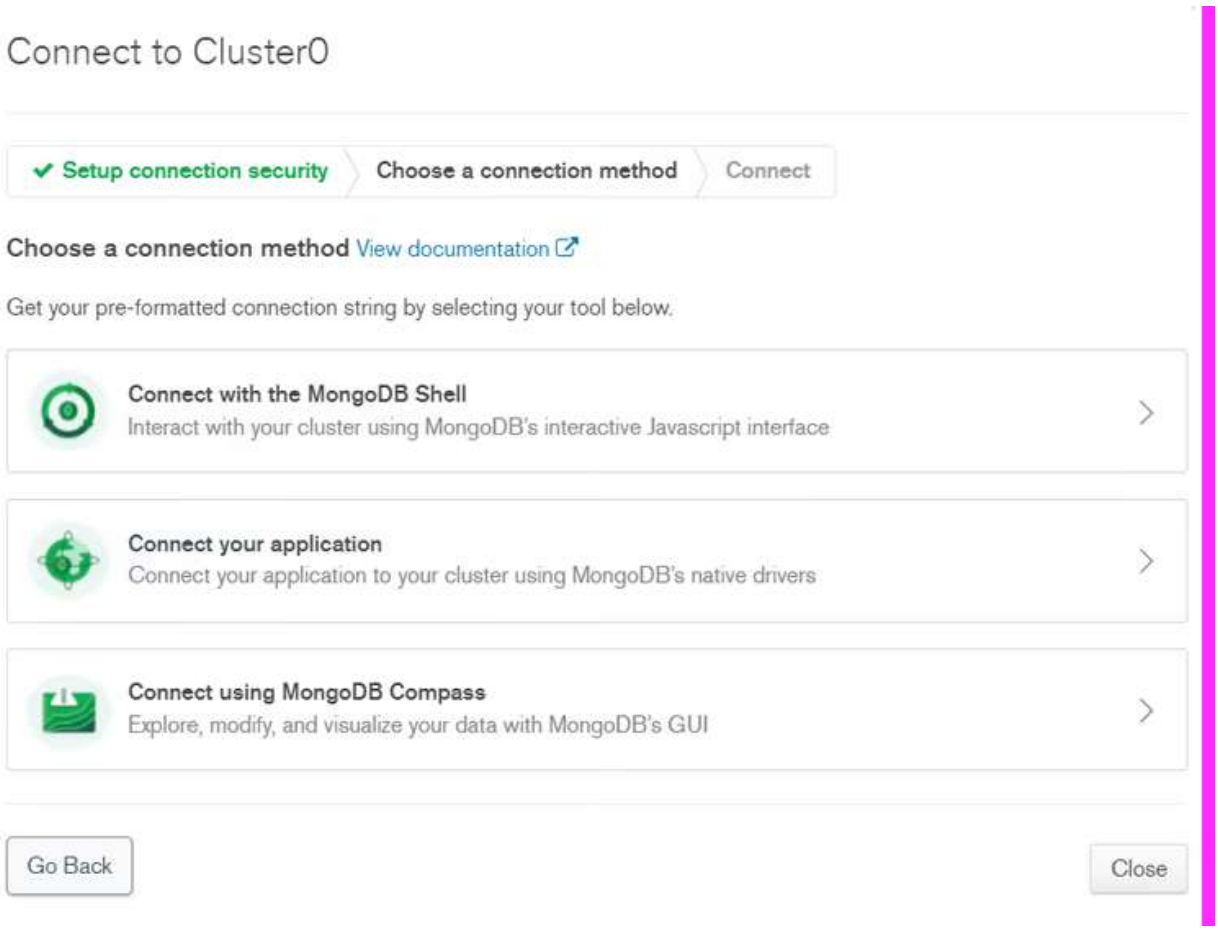
Navigate to the network access tab and select "Add IP address."



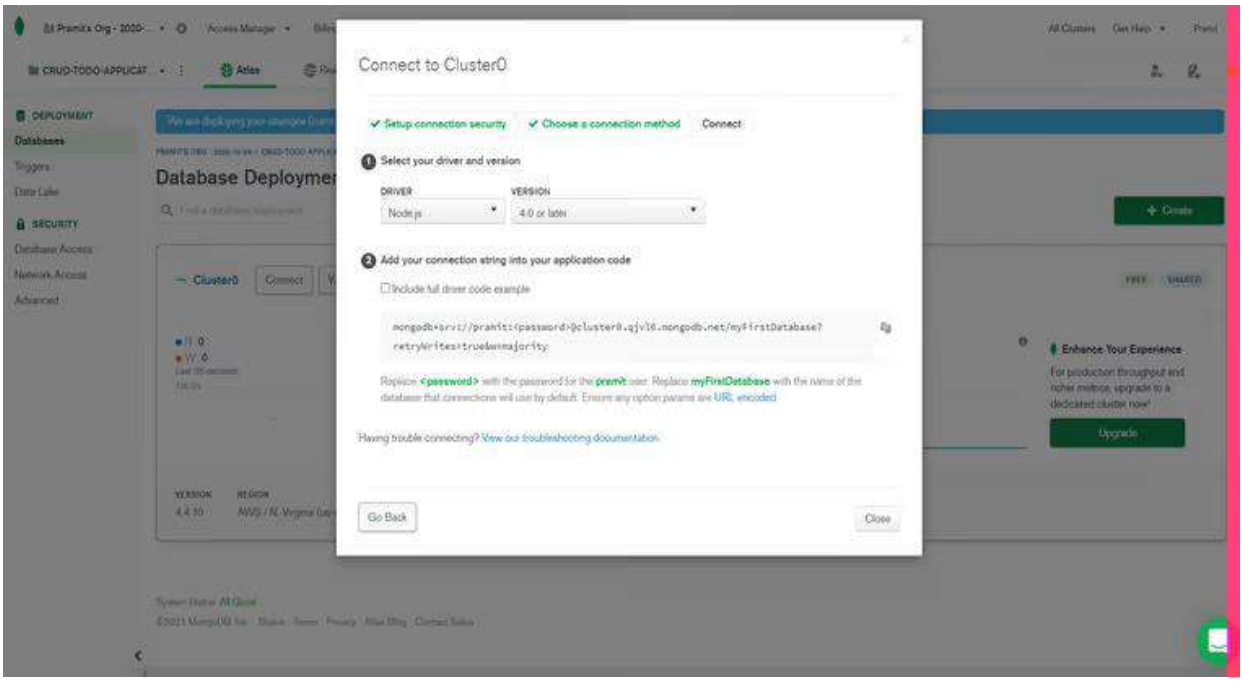
In the database, create a user. You'll need the username and password for the MongoDB URI and finally, create a database user.

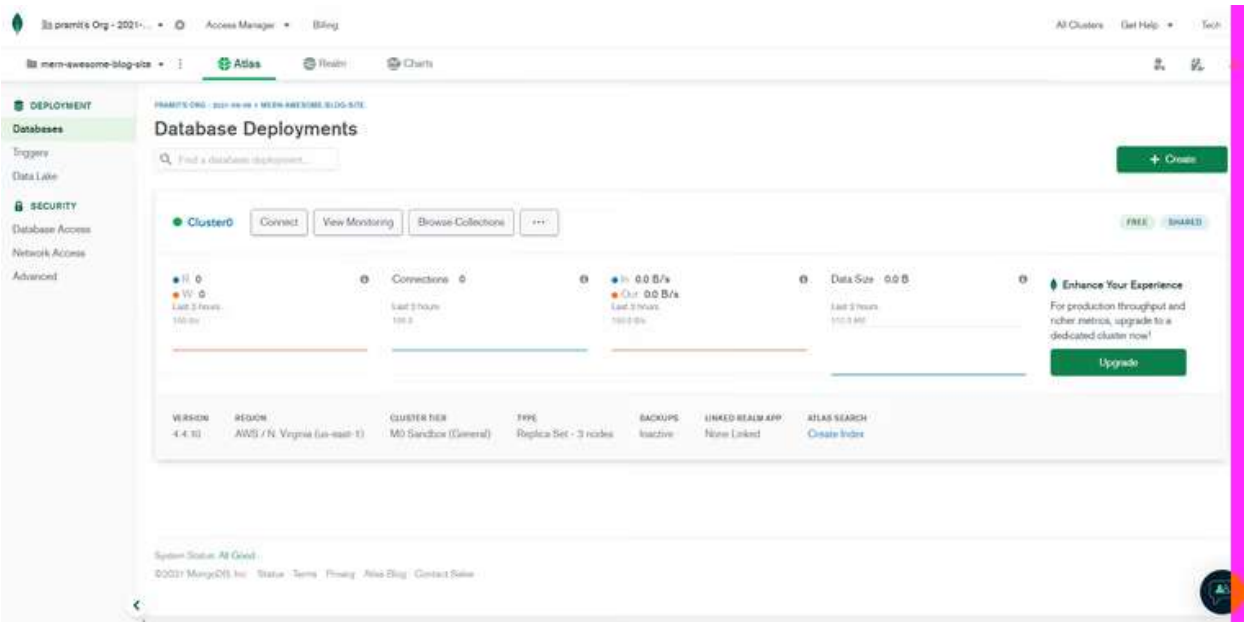


Now, select the `Choose a connection method.`



Connect your application by clicking on it and finally select the correct driver and version.





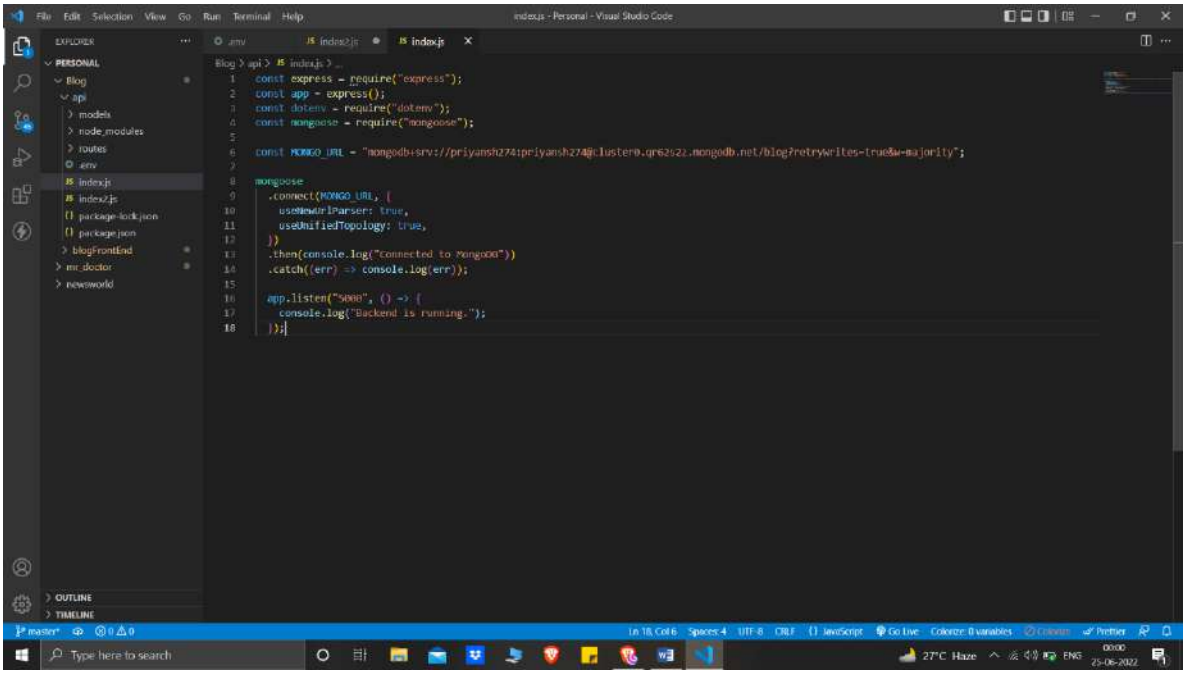
Now your database is also ready to connect .

Now, inside `index.js` create a new variable and name it MONGO_URL. Inside it, create a string and simply paste the copied mongo DB connection URL.

Now, inside it, enter your username and password, making sure to remove all the brackets and enter your own credentials. We'll secure the credential later by creating environmental variables, but for now, let's add it this way.

The first will be the DB CONNECTION, and the second will be an object with two different options. The first is useNewUrlParser, which we will set to true, and the second is useUnifiedTopology, which we will also set to true. These objects are not required, but we will see some errors or warnings on our console. Following that, let's chain a.then() and.catch() because this will return a promise, so inside .then() will console log “Connected to mongo” that will be executed if our application is successfully connected and finally, if the connection to the database is not successful we will simply console log our error message.

At last , call the app and invoke listen, which has two parameters, the first of which is PORT and the second of which is the callback function that will be executed if our application is successfully connected



That's it; we've successfully linked our server to the database.

Now you can start creating APIs and models and schemas
for your blog app , according to your convenience.

If you want to refer the code, you can visit the below link:

https://github.com/safak/youtube/tree/blog-rest-api