**Setting up MySQL with Heroku**

**Bradley Dinger & Benjamin Rittenhouse**

Code: <https://github.com/Braddinger13/CIS4282_BasicNodeSetup>

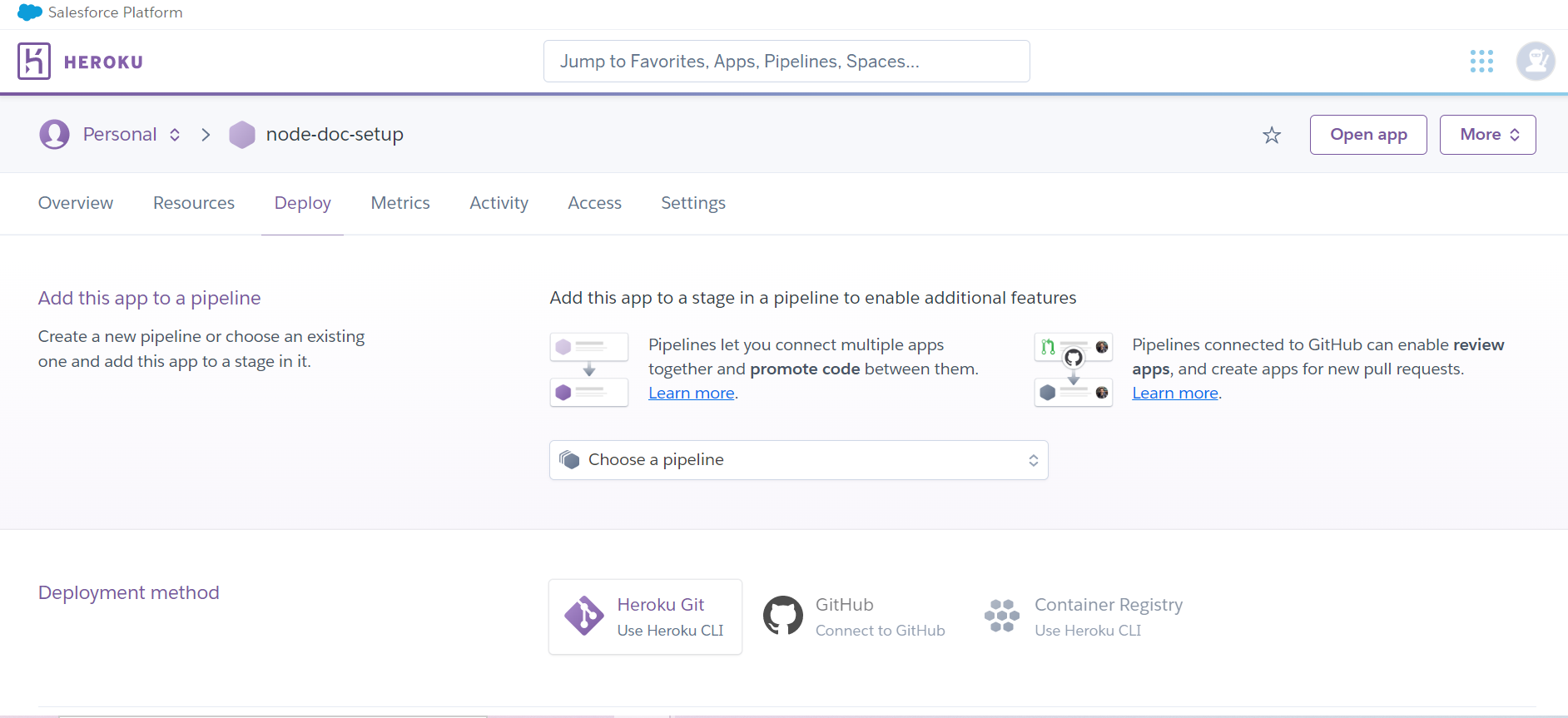
This document will show you how to create a Heroku database and connect it with MySQL Workbench.

1. **Heroku App Setup**

Once you have created a Heroku account, shown in [Getting Started with Heroku using Node.js and React.pdf](https://drive.google.com/file/d/1I9mdq_YdqJTtUKDTChB7VHil9ZCy-FMO/view)document, we are ready to create a new app on Heroku, so if you haven’t already, navigate back to the homepage.

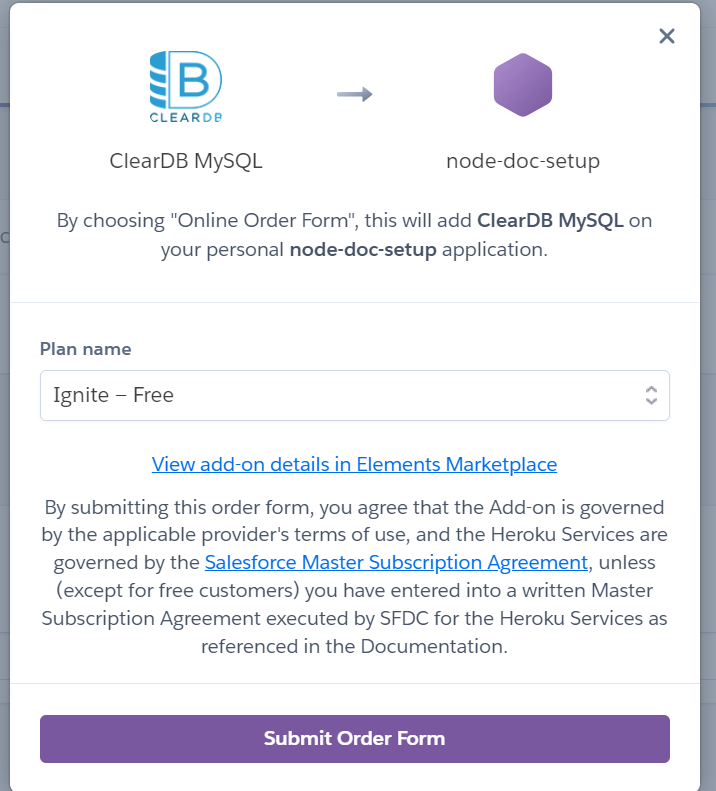
Towards the upper right you will see the  icon. Click that and select **“create new app”.** Submit your app name and select your region.

You should see this screen after creating your app:



From here, click the **Resources** tab. Under Resources, search for “**ClearDB MySQL**”

Upon clicking this you should be prompted with an order screen:

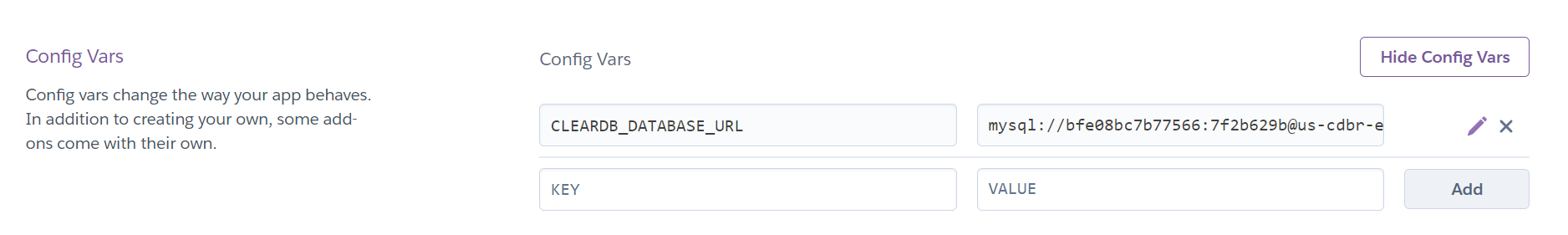


Keep the free “Ignite” version selected and click submit. If you have a credit/debit card on file the form will go through.

\**Heroku is stopping free database service, in this case the Ignite - Free plan will not be available*

Now we have an accessible database through Heroku!

1. **Connect Heroku DB to MySQL Workbench**

In your Heroku app’s homepage, navigate to the **settings** tab. Once you scroll down you should see a **Reveal Config Vars** button. Upon click you will see your database information: 

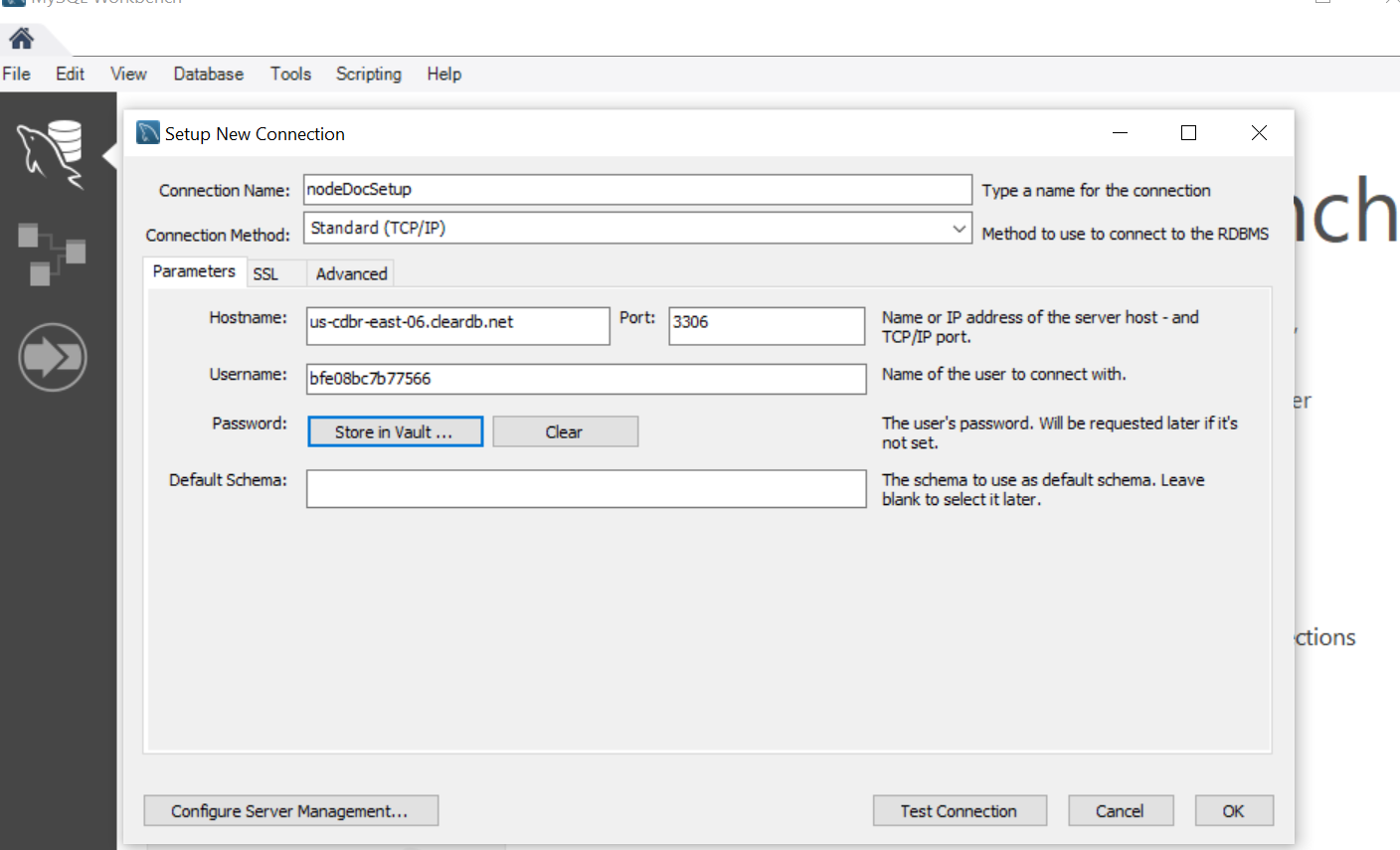
You can copy the “mysql://…” string by clicking the pencil icon on the right. **Do not change any part of the configuration string!** This string contains your database **password, username, and hostname like so:** mysql://**username**:**password**@**hostname**/heroku\_4875da0c9b91089?reconnect=true

You will only use what is underlined.

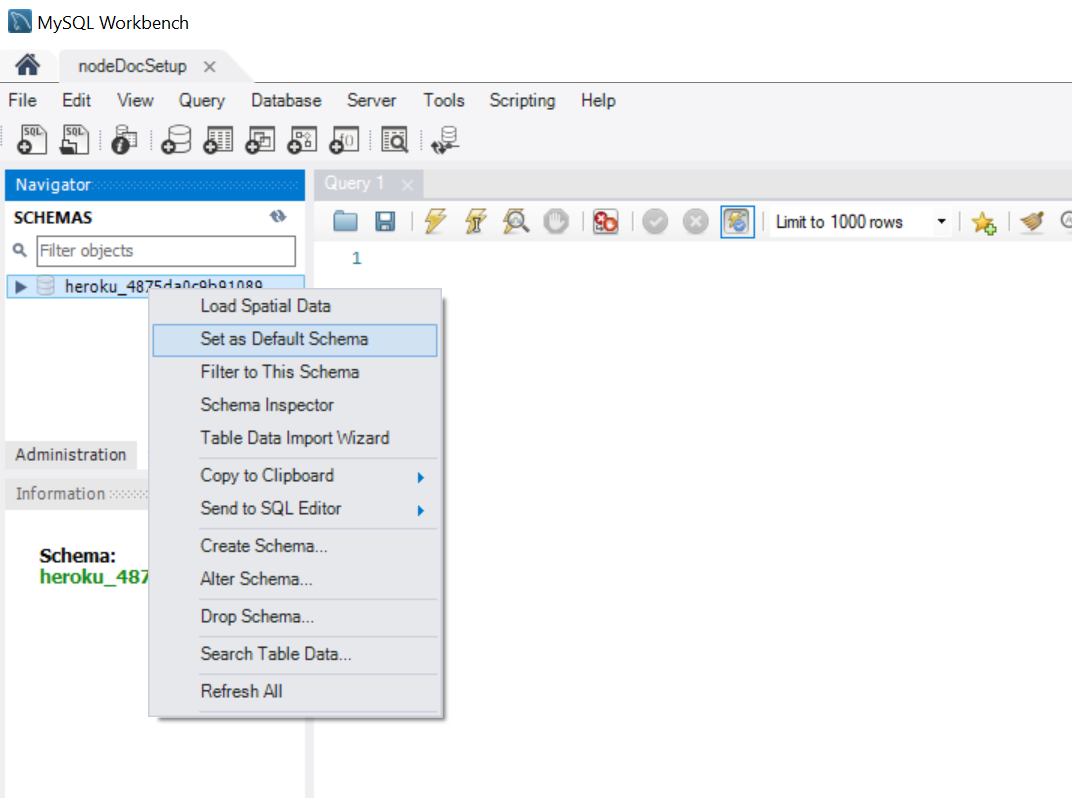
Now all we have to do is create the MySQL Connection in MySQL Workbench.

If you do not already have MySQL Workbench installed, Professor Kyvernitis has a tutorial on getting started: <http://cis-linux2.temple.edu/~sallyk/#/MySQL>

Simply input the configurations into a new connection, keeping the **Standard (TCP/IP)** connection method. If there is the option, store the password in the vault.



When you test your connection a message will pop up confirming that you are connected.

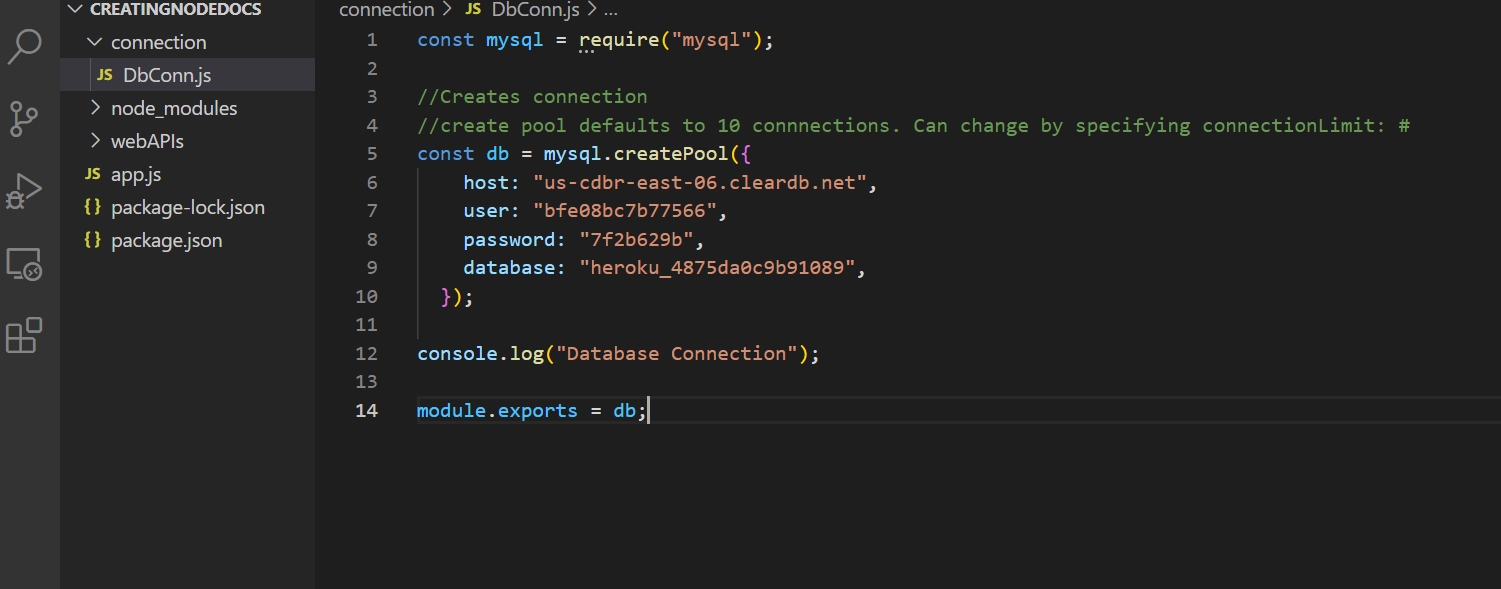
Lastly, I recommend setting the Heroku Schema as the default schema. 

1. **Create DB Table in MySQL Workbench**

For my database, I referred to Professor Kyverniti’s tutorial on creating database tables in MySQL Workbench. I encourage you to reference this document for setting up your web\_user/user\_role tables with a foreign key: <http://cis-linux2.temple.edu/~sallyk/tut/MySQL/howTo_MySQL/MySqlWB_SQL_Tutorial.pdf>

1. **Connect Node.js/Express App to the Database**

Now we have to create the database connection in our app. In your connection folder, create a DbConn.js file and declare a connection pool.

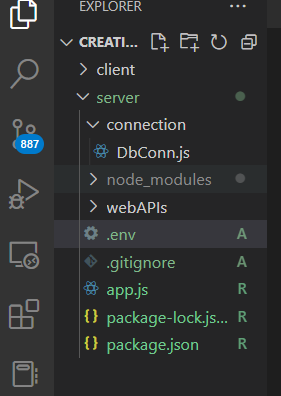
Your DbConn file may look like this:

The host, user, and password are the same ones we took from the Heroku Config Vars. I got the database name straight from MySQL Workbench.

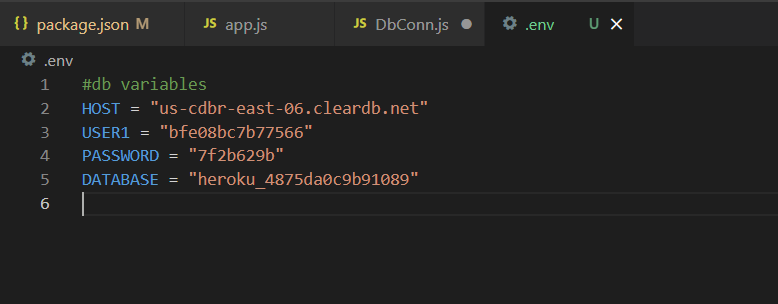
**Environment Variables-**

Since we don’t want our database configuration variables to be seen by everyone, we can utilize environment variables. To do this, we will add a **.env** file, as well as the **dotenv** dependency. Now type in the terminal “**npm install dotenv**”, and it should appear in your package.json file.

Create a .env file in your top-level server folder:

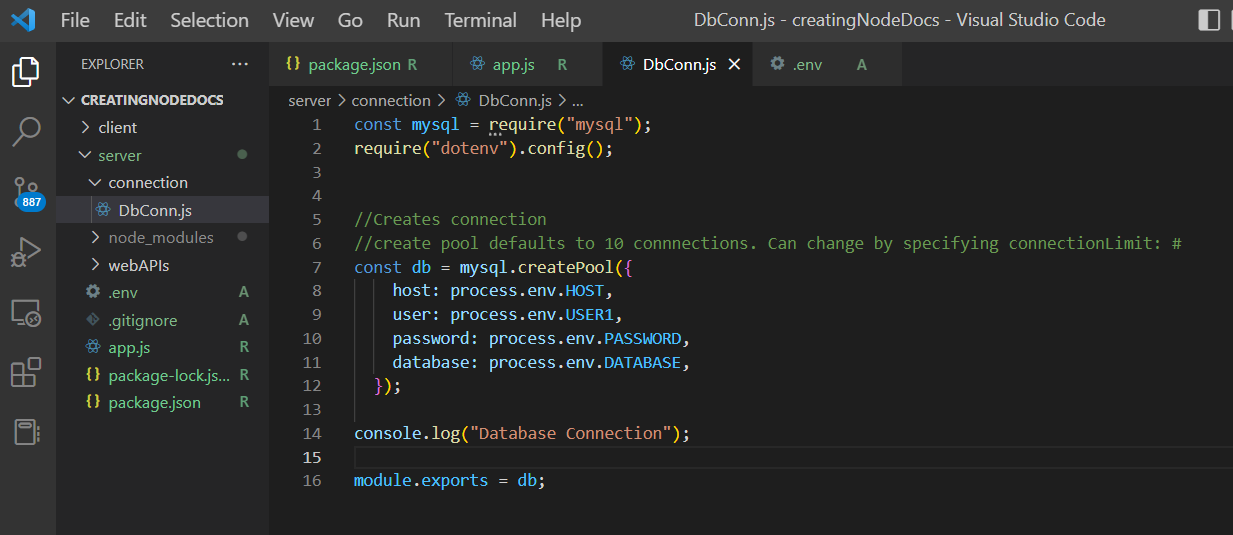


Here is how you can add new environment variables to your file:



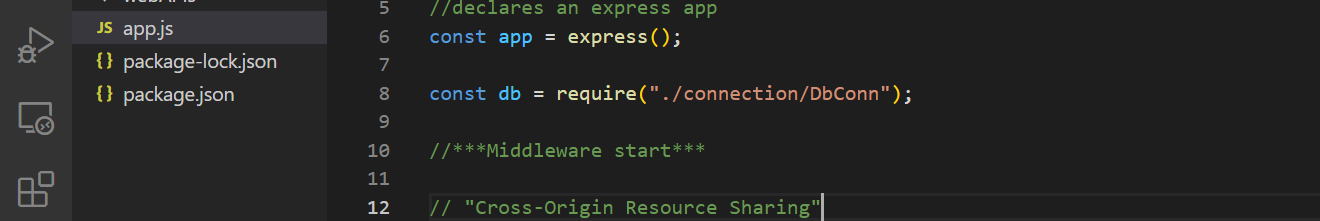
*\* I used USER1 because mac and windows each have a USER/USERNAME environment variable specific to your machine*

Copy this format with your database variables, then replace the code in DbConn.js like this,



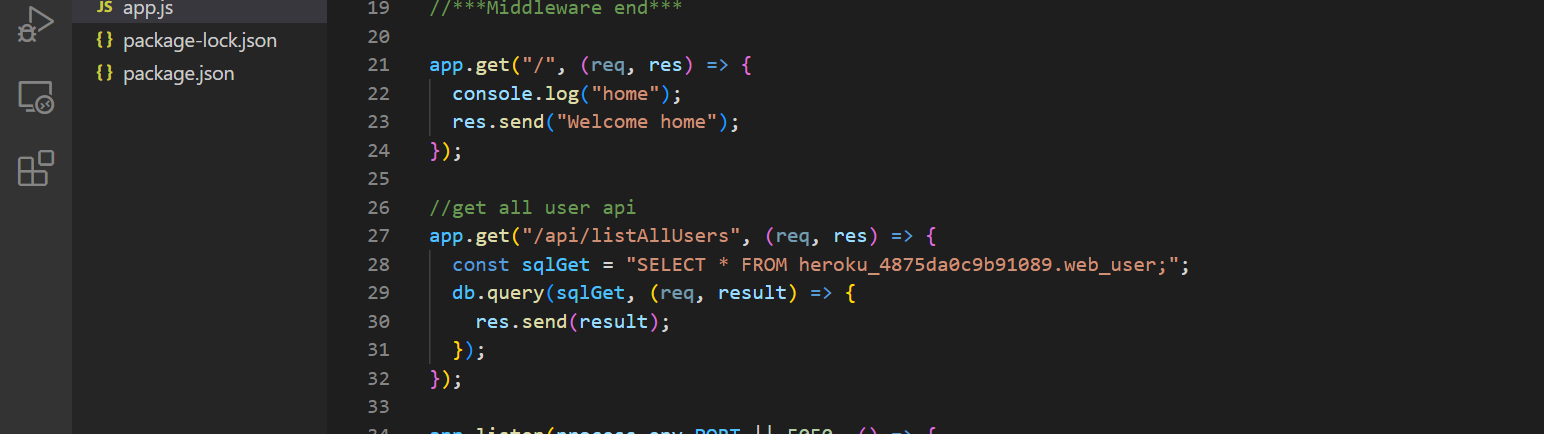
Don’t forget to add the code on **line 2,** this allows you to access your environment variables in this file. I would now recommend testing that you can still access your database with these variables set up.

Once you have confirmed that you can connect, add your **.env** file to a **.gitignore** file so it does not get tracked when pushing to github.

Next, import this file into app.js by adding this call on line 8:

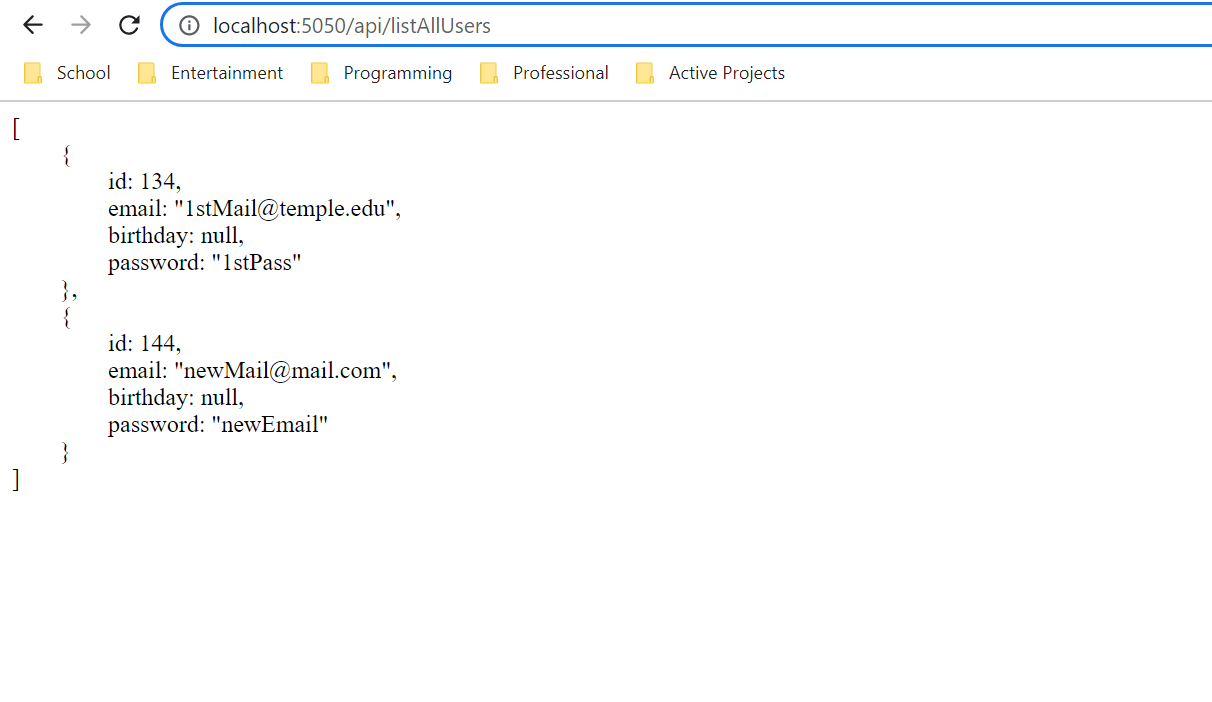
Now, if everything is filled out correctly, you should be able to access your DB through your express app.

To test this, we can use a basic listAllUsers api call in our app.js file.



This call will return a JSON response with all of the users listed in your database.

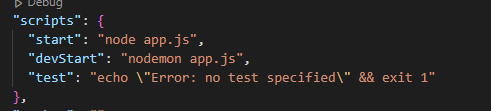
Make sure you save all files, then go back to your browser and type in the url “**localhost:5000/api/listAllUsers**”**,** you should see a response similar to this:



1. **Deployment**

The deployment for our backend is fairly similar to our frontend. However, there is an extra step.

In your package.json you should add this:



The **start** script lets Heroku, or the hoster, know which command to execute

upon starting. The **devStart** script can be used by the developer, or you can just use nodemon.

You may also use these scripts through the command line by typing:

“**npm run [script name]**”

Now you can refer to [Getting Started with Heroku using Node.js and React.pdf](https://drive.google.com/file/d/1I9mdq_YdqJTtUKDTChB7VHil9ZCy-FMO/view) for the rest of the deployment process, as it is the same.

Upon clicking **Open app,** you will see your page with your app’s prefix in the URL. You can now run your api calls by changing the URL.

