**Deploy an webapp to Render.**

# Some common terminologies.

* **Deployment**: refers to the process of making an application available and ready for use by end users. It involves moving the application's code and resources from a development environment to a production environment where it can be accessed by users. This process typically involves installing and configuring the application on servers or cloud-based platforms, and ensuring that it is running smoothly and securely. The actual process of deploying may be tricky and time-consuming, especially for beginners (and even for pros). Hence, we have platforms like Render that take away the need to do all the infrastructural setup and just move from a local machine to getting a URL to showcase your project.
* **Backend App**: is the part of a software system that runs on the server-side and is responsible for processing data, performing calculations, and responding to requests from the front-end or client-side of the application. It handles the logic and functionality of the application that is not directly visible to the end-user, such as managing databases, processing payments, and authenticating users.
* **Node.js**: is an open-source, cross-platform JavaScript runtime environment built on Google's V8 JavaScript engine. It allows developers to run JavaScript code on the server-side, rather than just in a web browser, which makes it possible to build scalable and high-performance applications using JavaScript. Node.js provides a set of built-in modules for handling common web development tasks such as creating HTTP servers, working with file systems, and handling requests and responses. It also has a vast ecosystem of third-party modules and packages that developers can use to enhance the functionality of their applications. Node.js is widely used for building web servers, APIs, real-time applications, and command-line tools.
* **Express.js**: is a popular open-source framework for building web applications in Node.js. It provides a set of tools and features for creating web servers, handling HTTP requests and responses, and routing requests to specific endpoints or URLs. Express.js is built on top of Node.js and uses the same non-blocking I/O model, making it well-suited for building scalable and high-performance web applications. Express.js is widely used for building web applications, RESTful APIs, and single-page applications (SPAs).
* **PostgreSQL**: (also known as Postgres) is a powerful open-source relational database management system (RDBMS). It is known for its robustness, reliability, and support for advanced features such as transactions, stored procedures, and user-defined functions. PostgreSQL is designed to store and manage large amounts of data, and it can handle complex workloads with high concurrency and performance. It supports a wide range of programming languages and has a flexible data model that can handle structured, semi-structured, and unstructured data. PostgreSQL is widely used in web applications, data analysis, and other areas where a reliable and scalable database is required. It is free and open-source software that can be installed on a variety of operating systems, including Windows, Linux, and macOS.
* **Render**: is a cloud platform that provides a complete solution for deploying and scaling web applications, APIs, and background workers. It offers a range of services including web hosting, managed databases, automatic SSL, and continuous deployment. Render supports a wide range of programming languages and frameworks, including Node.js, Python, Ruby on Rails, and Docker containers. Render's platform is built on top of modern cloud infrastructure, which ensures high availability, scalability, and security for its customers. It also provides a simple and intuitive user interface, making it easy for developers to deploy and manage their applications. With Render, developers can focus on building their applications and leave the infrastructure management to the experts.

# Step to set the project up.

## Registration.

* Visit the <https://render.com> website and ready to go.

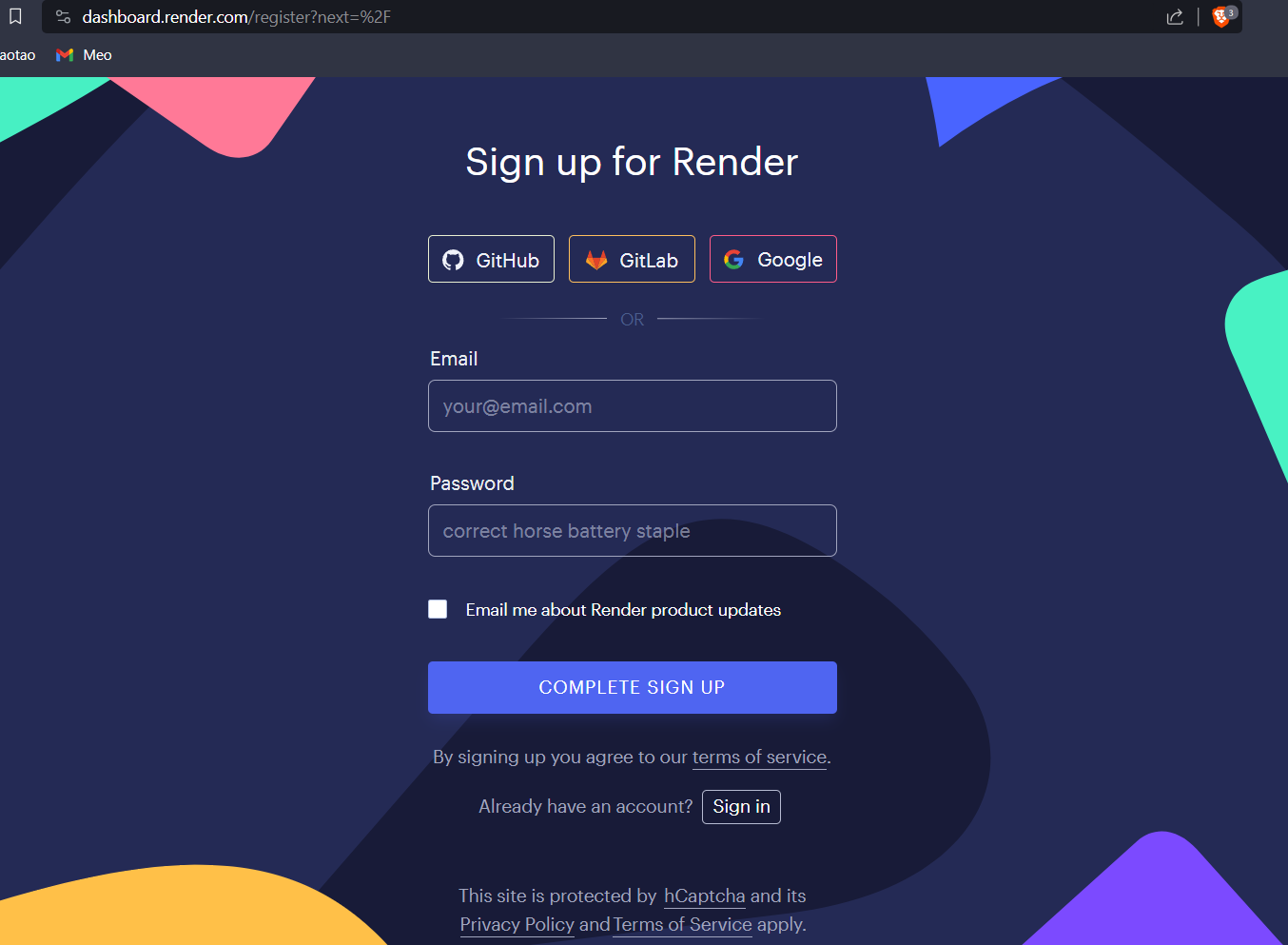


Fig 1: Fill out some credentials and you are ready to go.

* After the registration, now you can log in to the website.

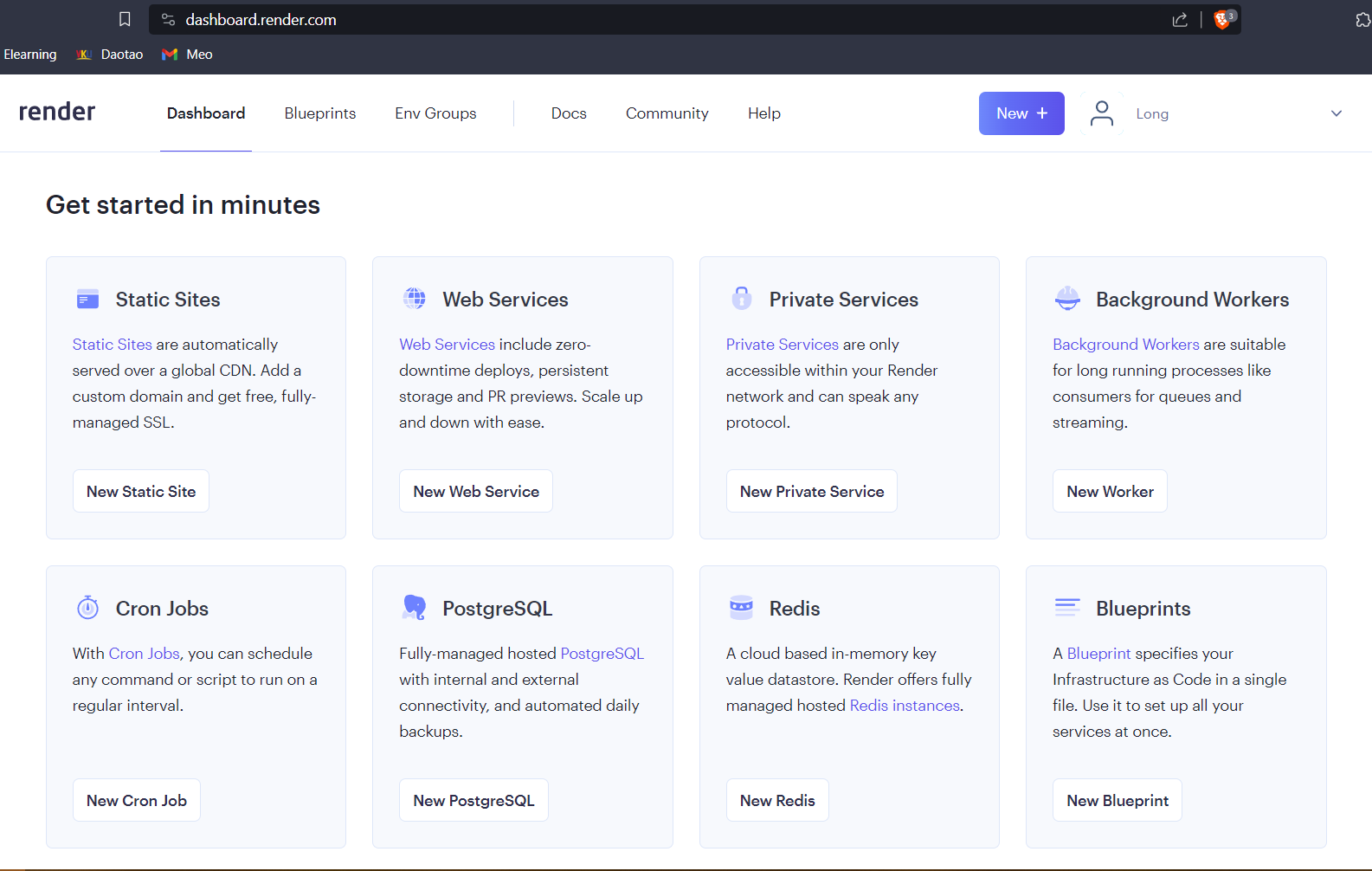


Fig 2: You will have some options for individual purposes.

## Building a NodeJS app.

* Here we did a student management project.

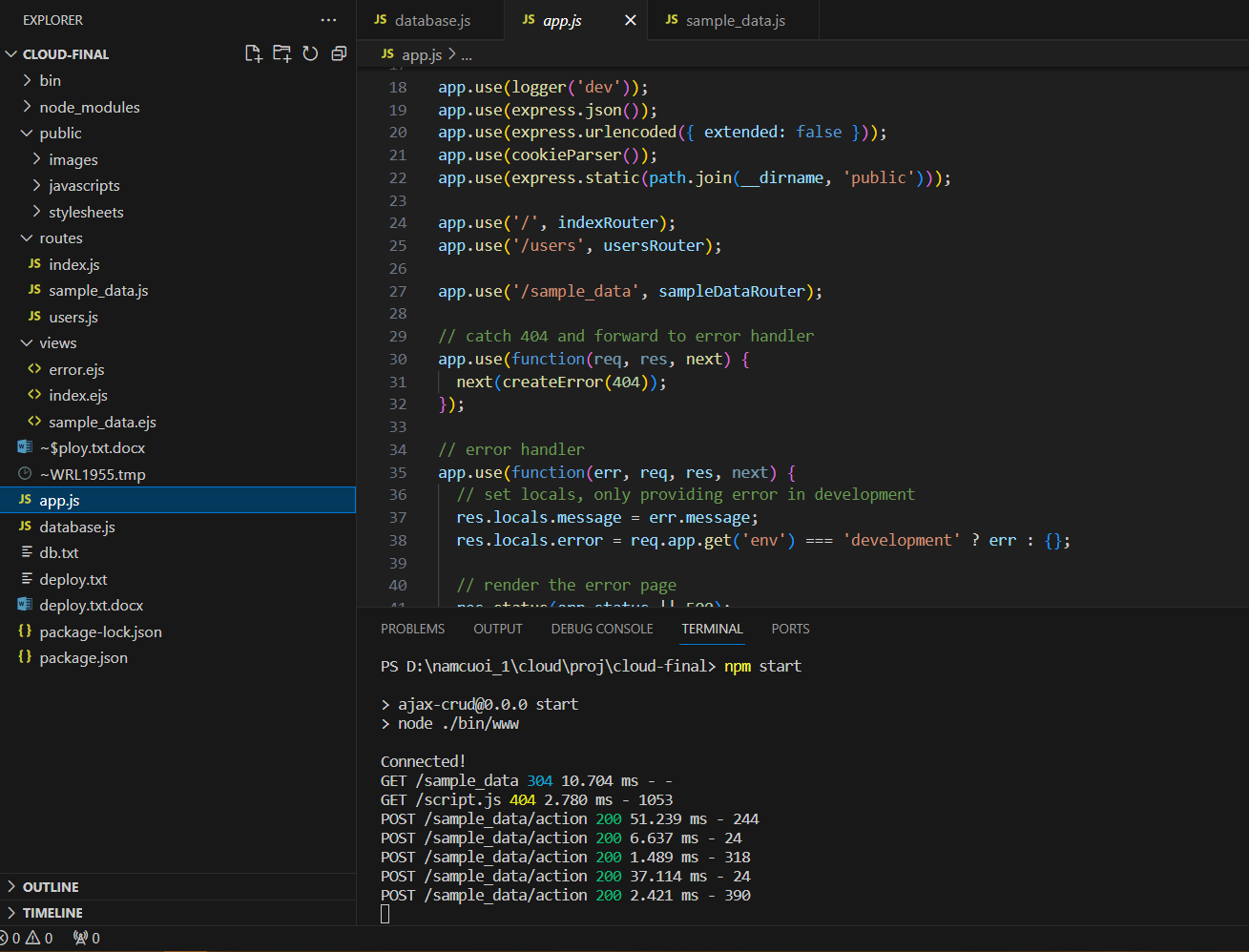


Fig 3: A NodeJS project.

## Create a Postgres Database on Render.

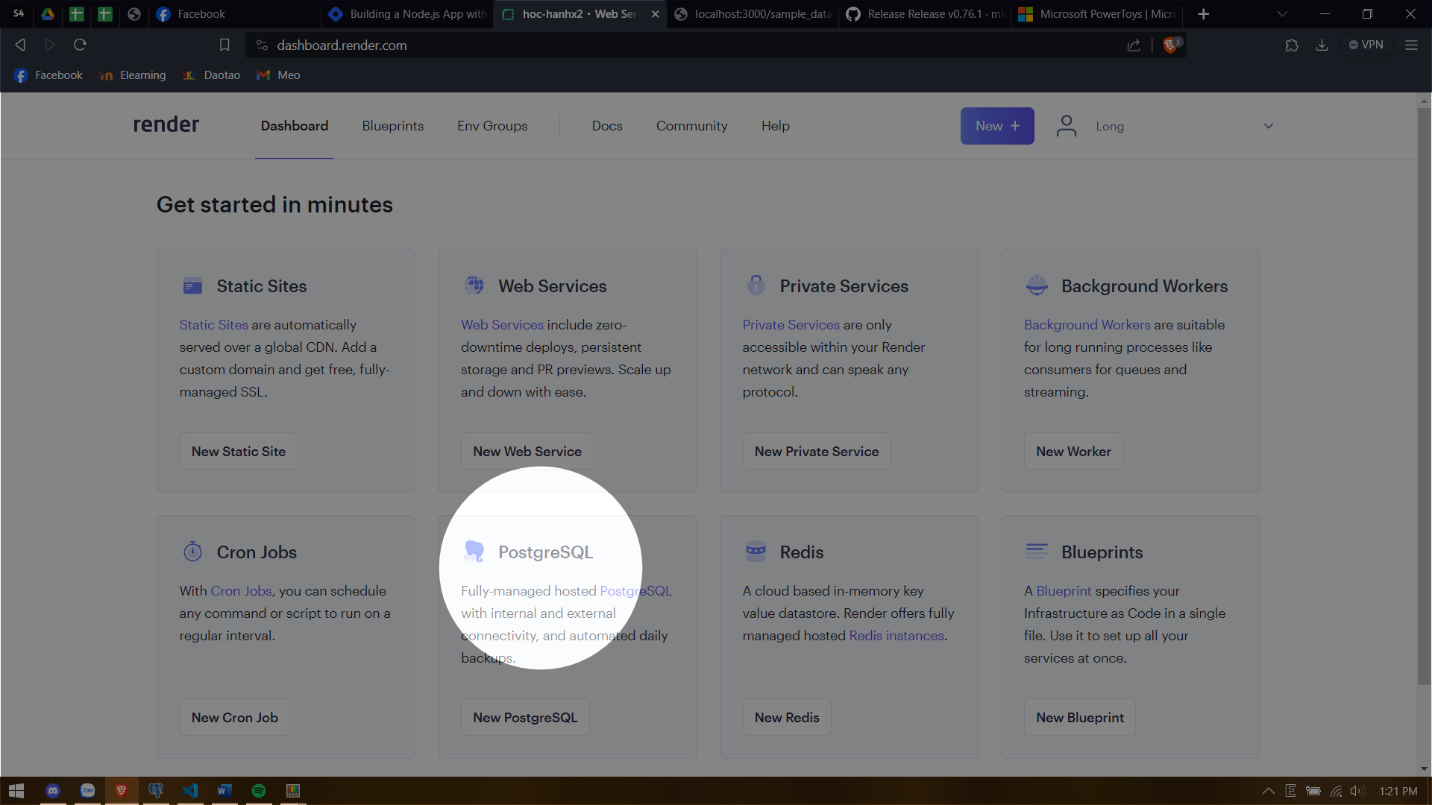


Fig 4: Select the PostgreSQL.

* Then, you can fill out some information for you project.

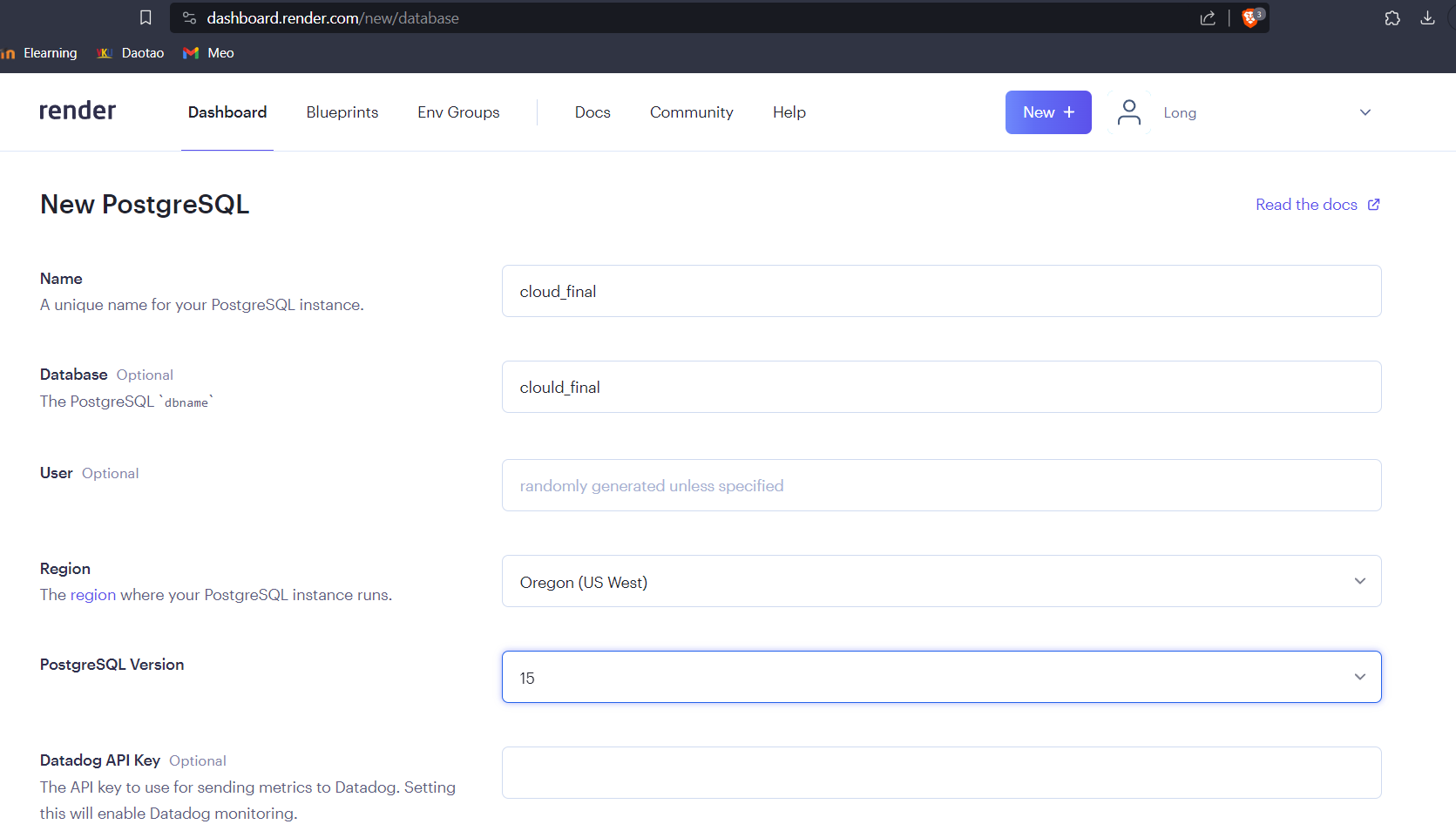


Fig 5: Some information need to be done.

* After creating process finish, you can move on to the next step.

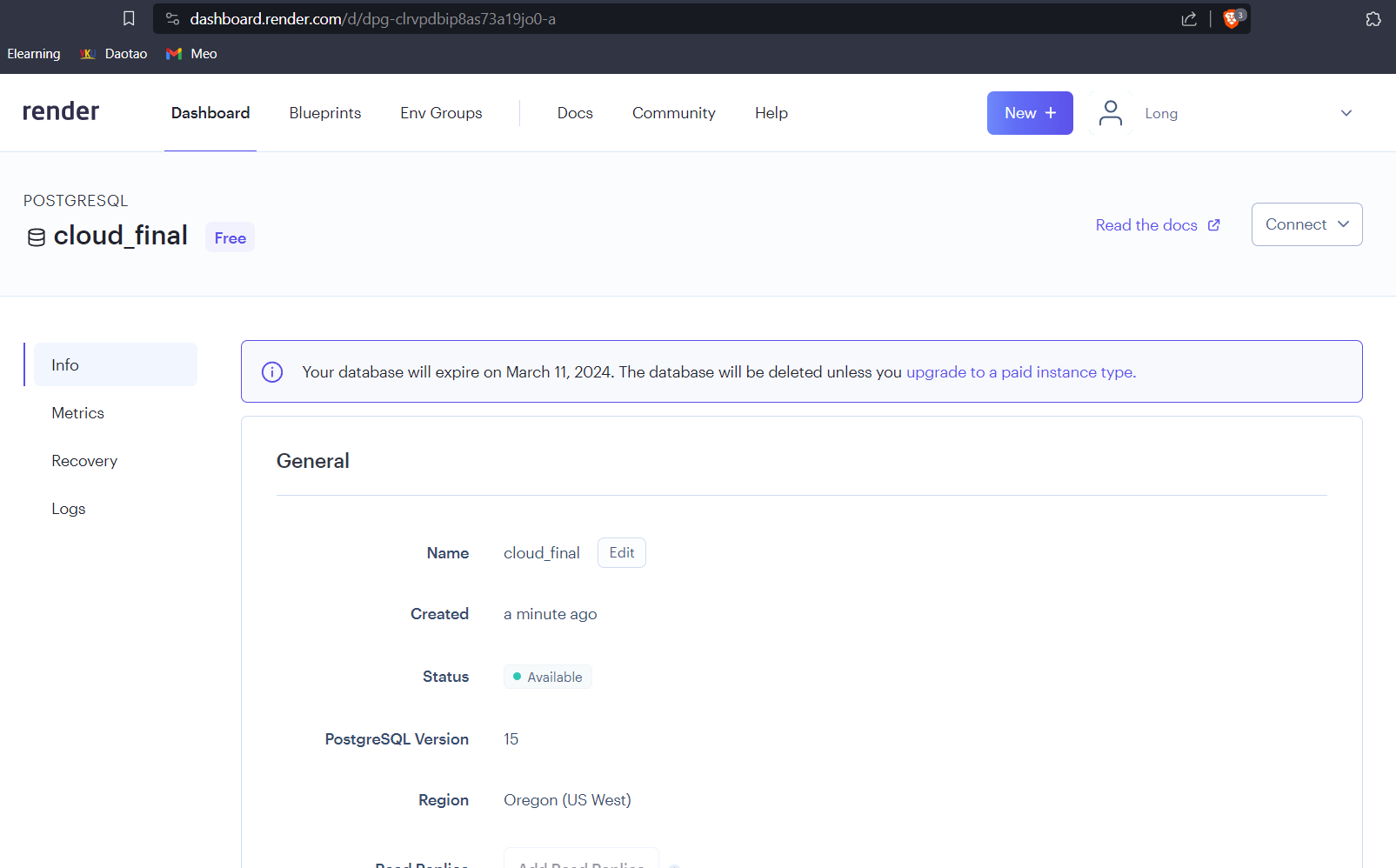


Fig 6: Ready to connect.

* You can get the information for the connection (right-top corner).

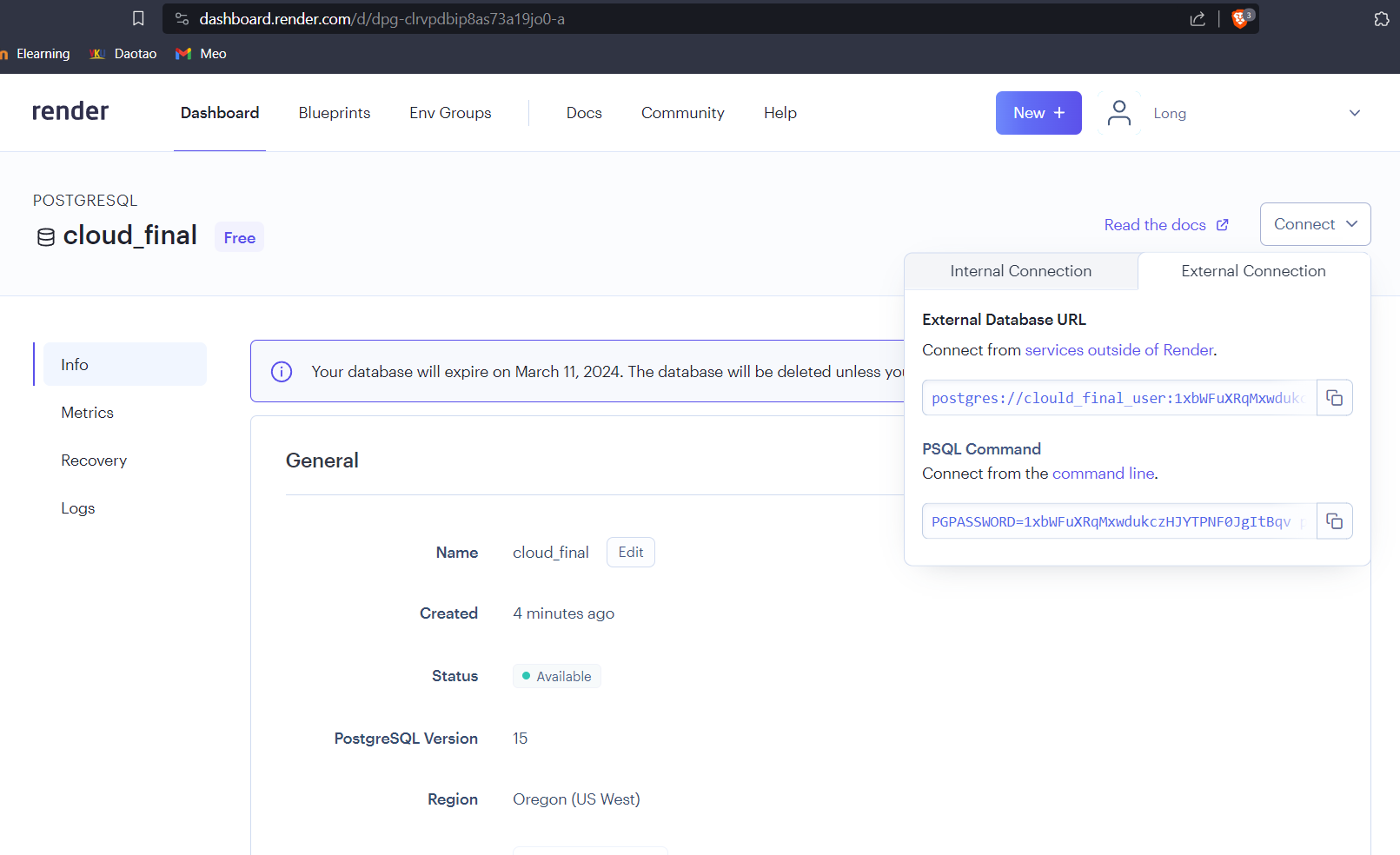


Fig 7: Credentials for the connection.

* You’d get a text similar to this:

**‘PGPASSWORD=<password> psql -h <hostname> -U <username> <databasename>’**

* Then, you are able to test the connection from your local machine. Just follow this step.
  + - Open psql shell.

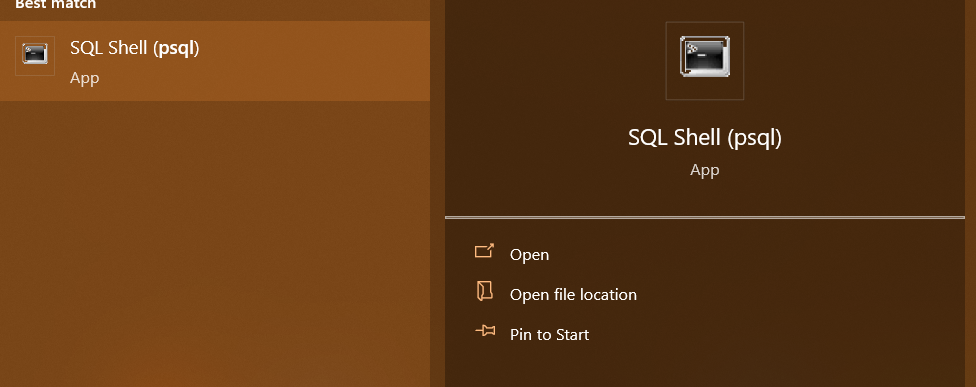
****

Fig 8: psql shell

* + - Fullfill the credentials.

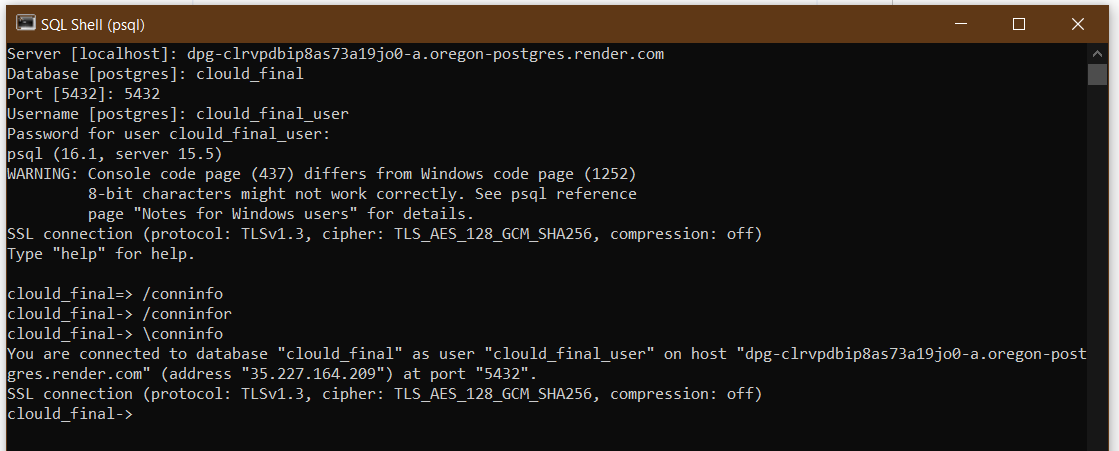
****

Fig 9: Fill out the information.

* + - After testing the connection, you’d be able to create the table. And you can check it with ‘\dt’ command.

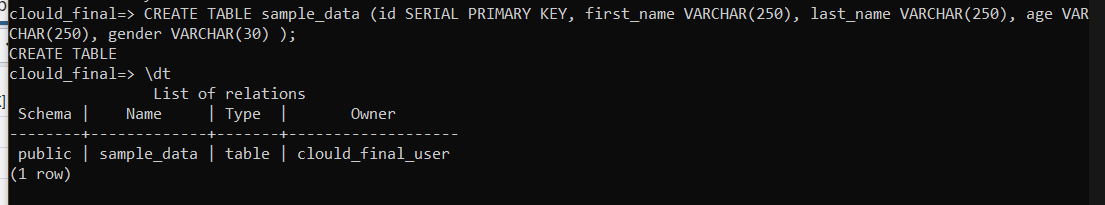
****

Fig 10: Create table and test.

## Connect the NodeJS with the Database.

* Fire up the project and create theDBConfig.js file.

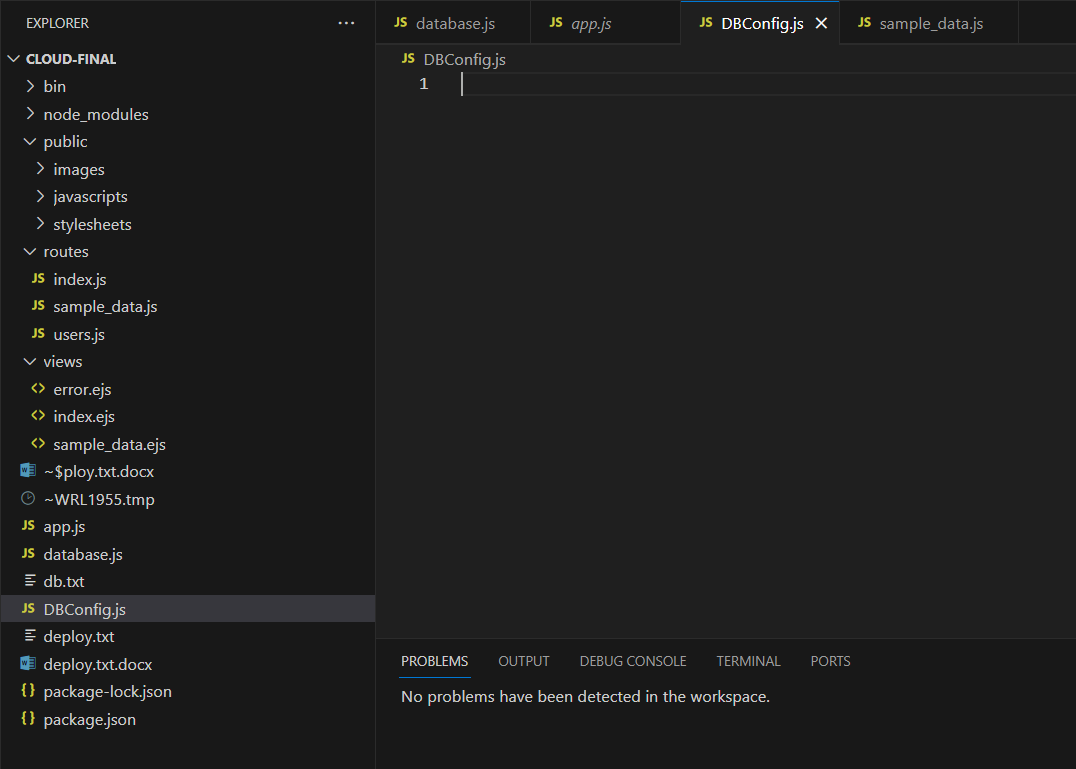


Fig 11: Create DBConfig.js file.

Fig 12: Create DBConfig.js file.

* Check the database dashboard on Render, go to “Connect” and copy the link for “External connection”. It’d be similar to:

**‘postgres://<username>:<password>@<host>/<database>’**

* Since we will deploy this project later on, then we need to hide some ‘sensitive’ information by toggle up the terminal in your project and run this command:

**‘npm I dotenv’**

* Then you can you it like this on your app.js.

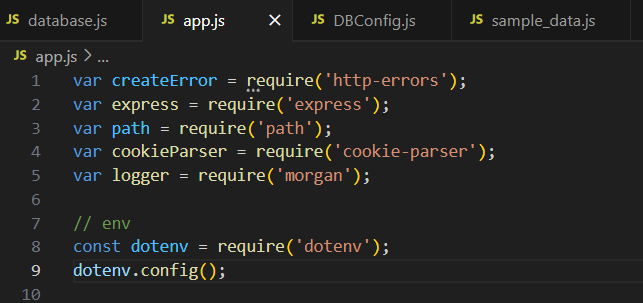
****

Fig 13: Use dotenv.

* After that, create a .env file and save the information within it.

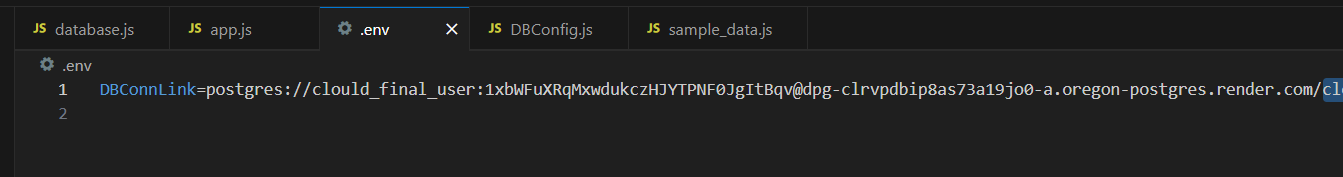
****

Fig 14: Save connection information.

* Build the connection function in DBConfig.js file.

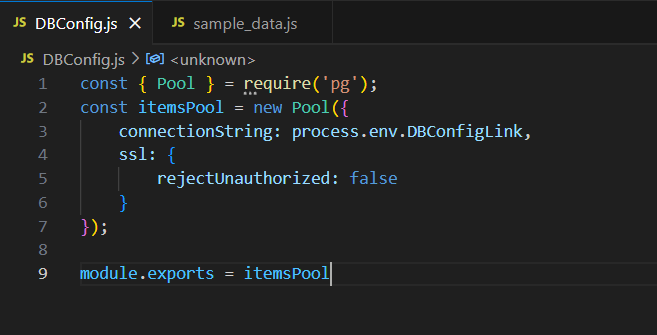


Fig 15: Build connection function.

* Use the function to read and send items into the database import this function into your routes.

## Deploy the entire project to Render.

* After connect the NodeJS app with the Database, we are ready for the deployment.
* Remember to push your project to a remote platform such as GitHub where Render can find and access to the resources and deploy it publicly.
  + - Visit the render website again and create a web service.

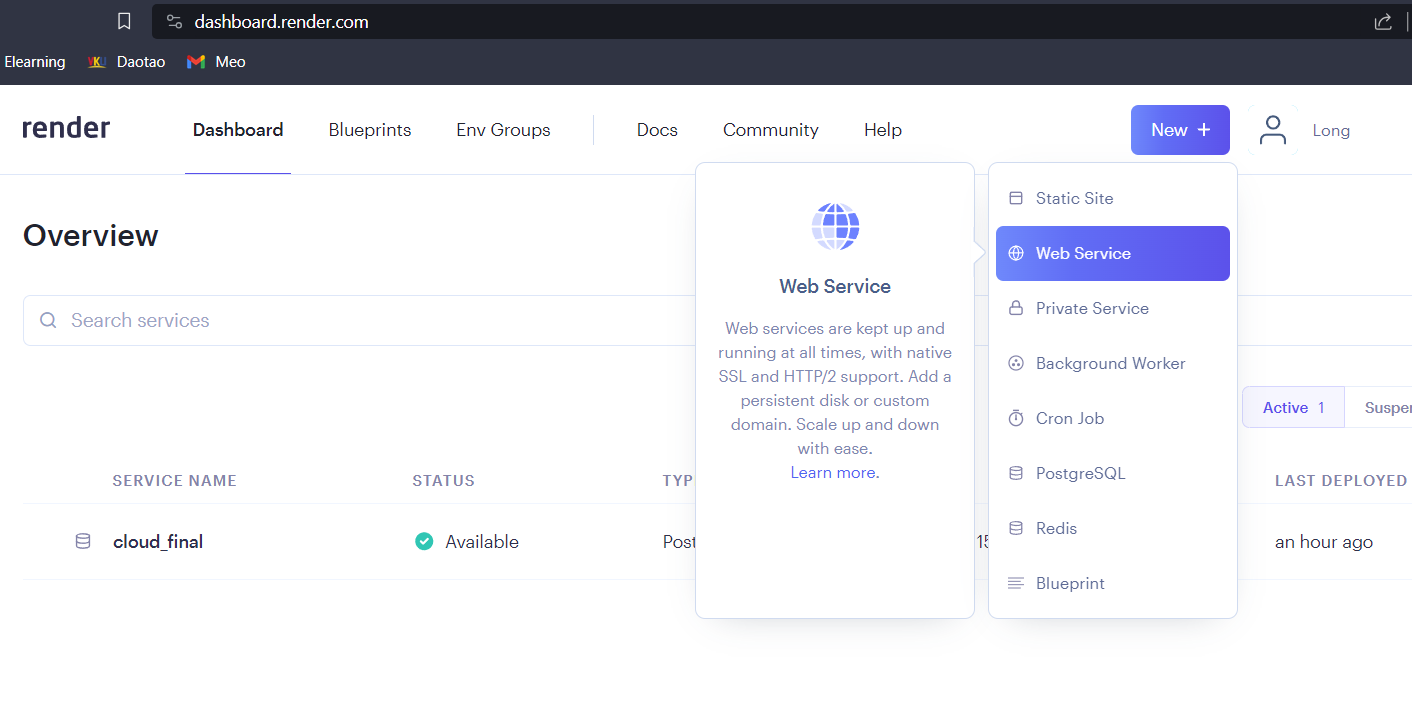


Fig 16: Create a web service.