Deploy a Node App and Postgres Database to Render

What is Render?

Render is a unified cloud platform that allows you to build and run all of your apps and websites while providing free TLS certificates, a global CDN, DDoS protection, private networks, and Git auto deploys. In addition, Render allows you to host static sites, backend APIs, databases, cron jobs, and other types of applications in a single location.

Prerequisites

To follow this tutorial, ensure you have the following installed on your computer.

- Node.js version 14 or later
- PostgreSQL database version 14 or later

Also, the code for this tutorial is available on GitHub. Feel free to clone and follow along.

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.

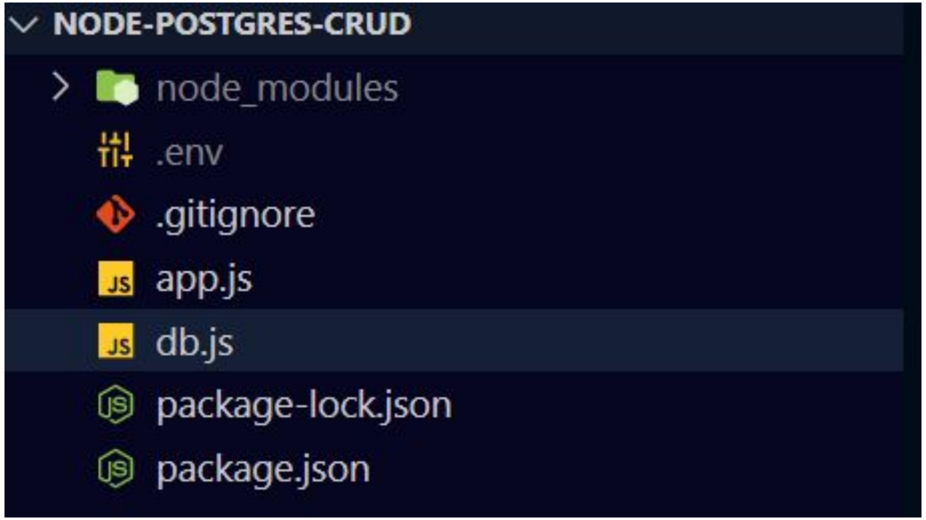
```
npm init -y
```

Install dependencies

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

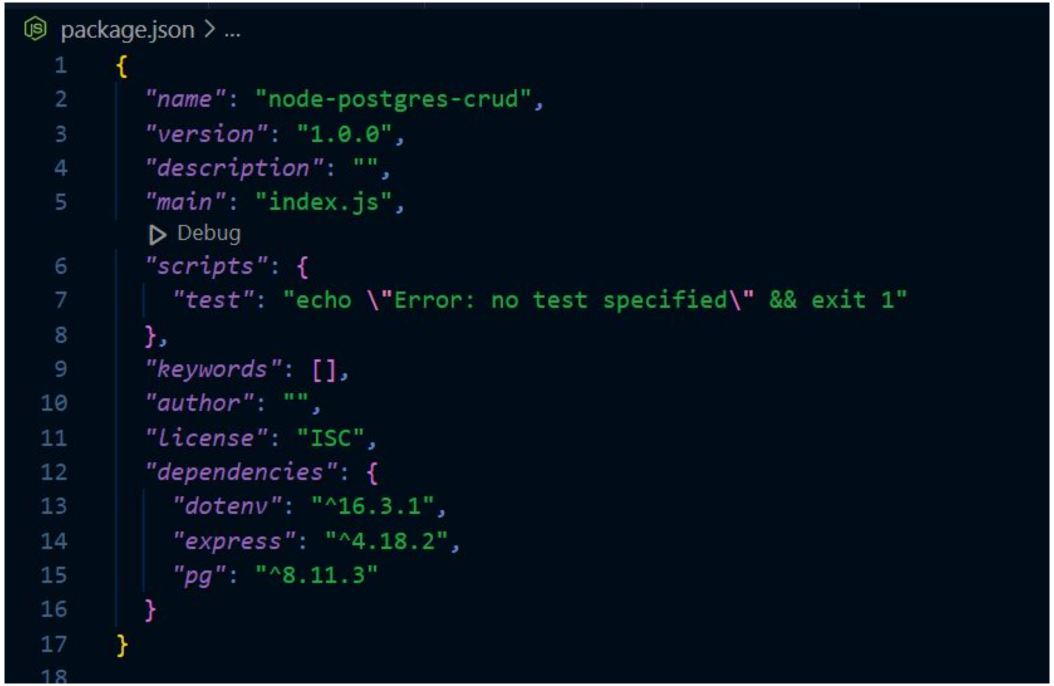
```
npm install dotenv pg express
```

Project structure



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

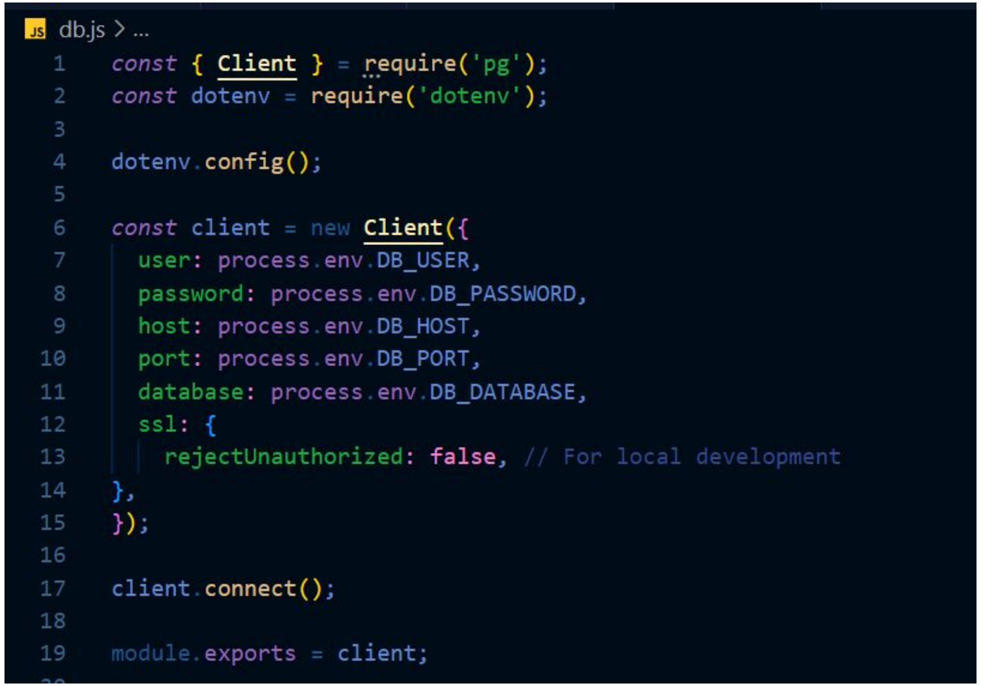
package.json



package.json > ...

```
{
  "name": "node-postgres-crud",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  ▷ Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "pg": "^8.11.3"
  }
}
```

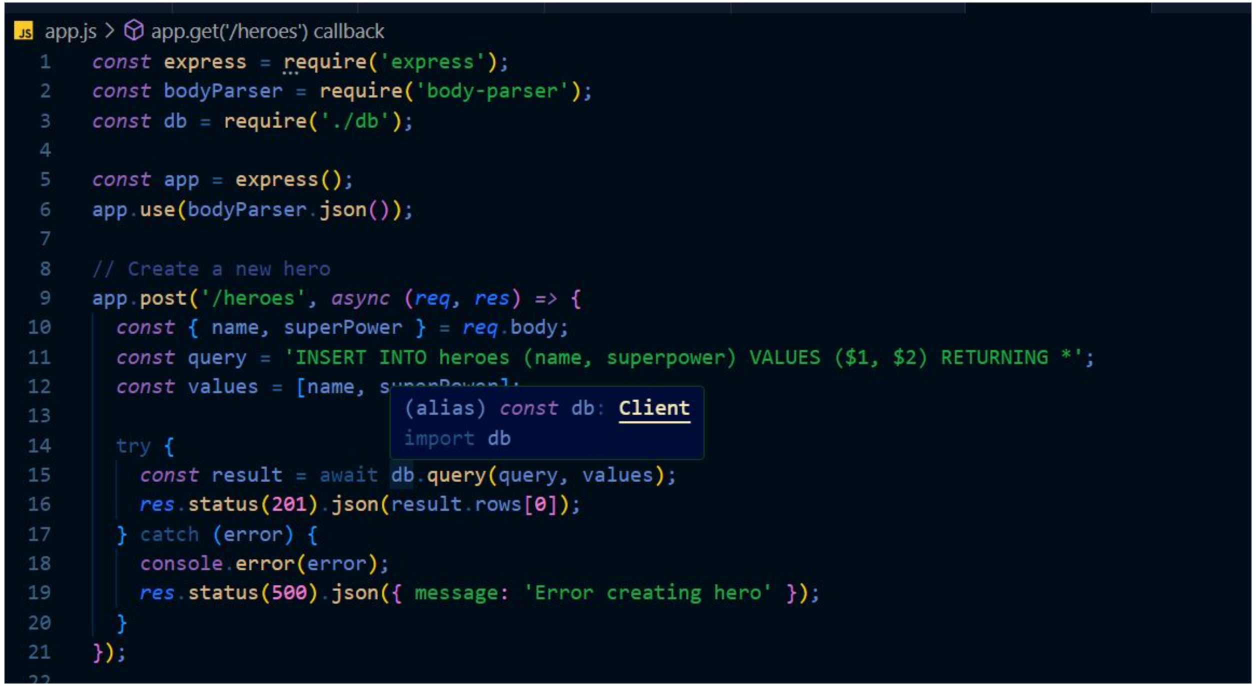
db.js



db.js > ...

```javascript
const { Client } = require('pg');
const dotenv = require('dotenv');

dotenv.config();

const client = new Client({
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  host: process.env.DB_HOST,
  port: process.env.DB_PORT,
  database: process.env.DB_DATABASE,
  ssl: {
    rejectUnauthorized: false, // For local development
},
});

client.connect();

module.exports = client;
```

app.js

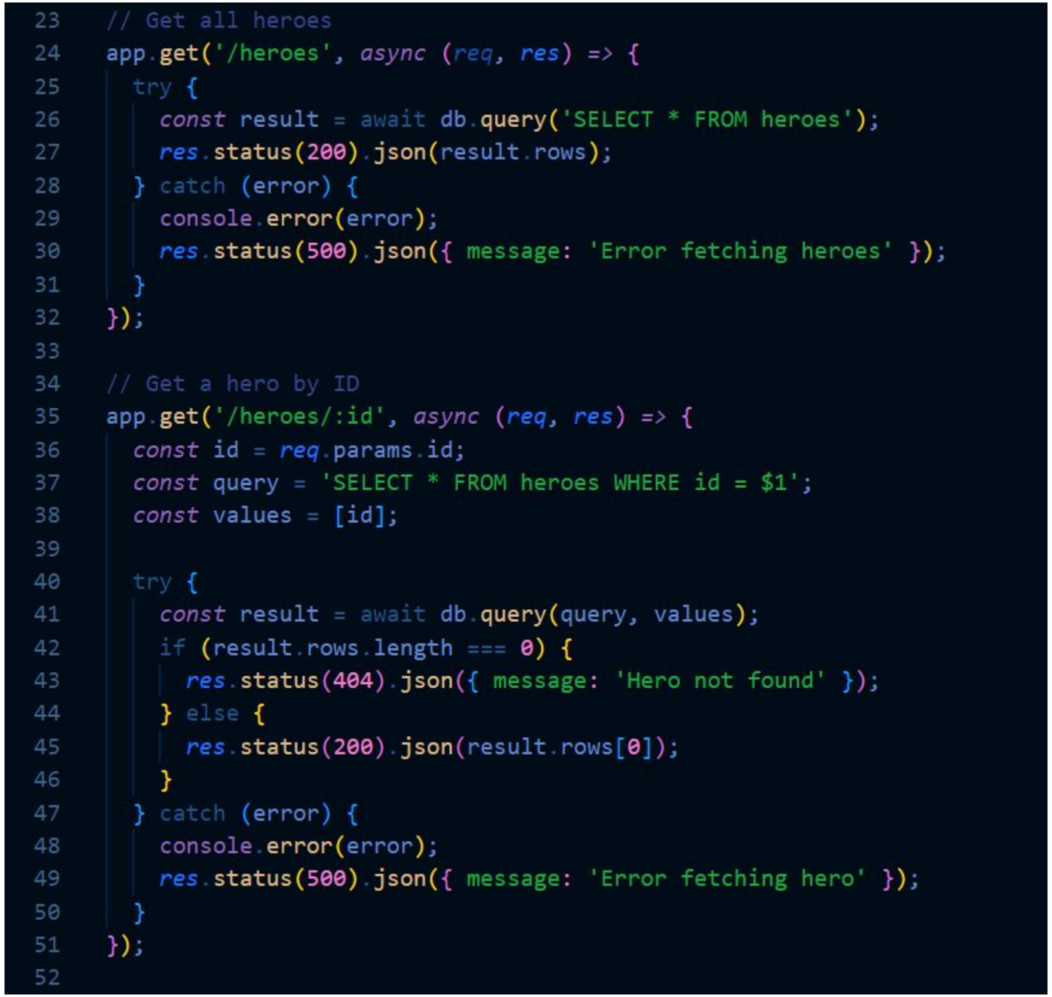


app.js > app.get('/heroes') callback

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

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

// Create a new hero
app.post('/heroes', async (req, res) => {
  const { name, superPower } = req.body;
  const query = 'INSERT INTO heroes (name, superpower) VALUES ($1, $2) RETURNING *';
  const values = [name, superPower];

  try {
    const result = await db.query(query, values);
    res.status(201).json(result.rows[0]);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error creating hero' });
  }
});
```

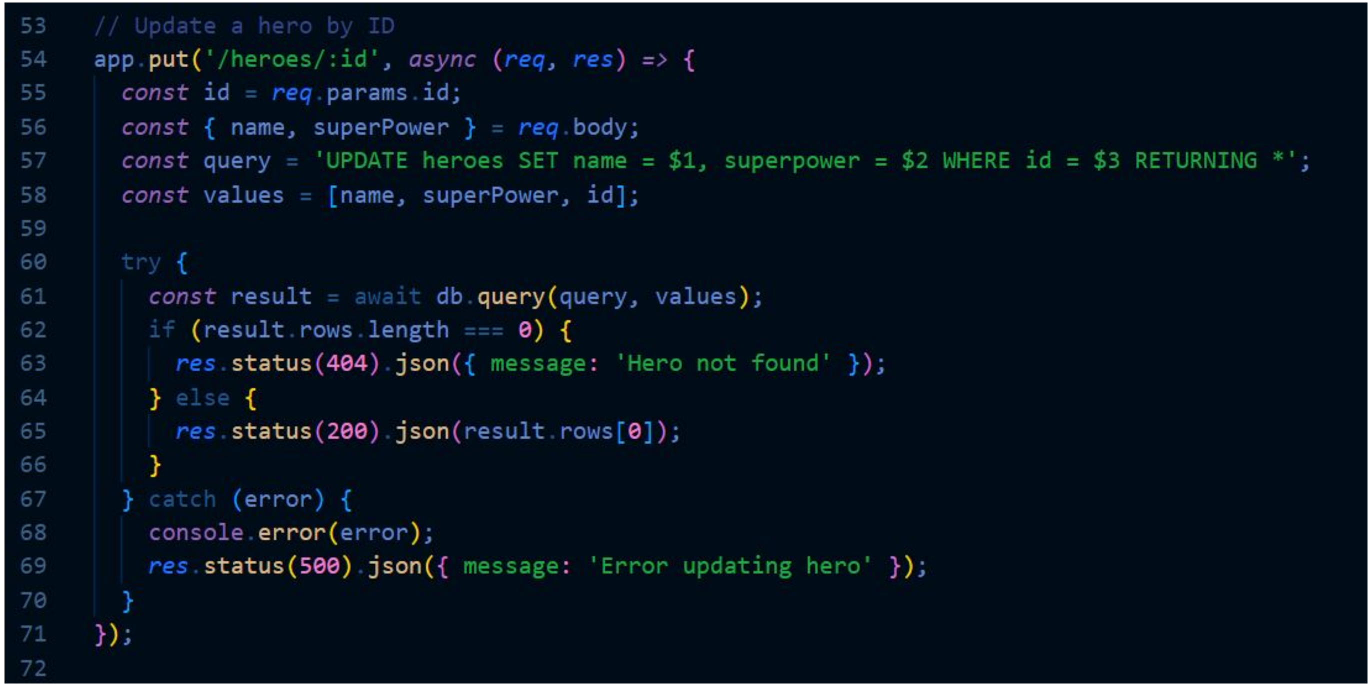
```
(alias) const db: Client
import db
```

app.js



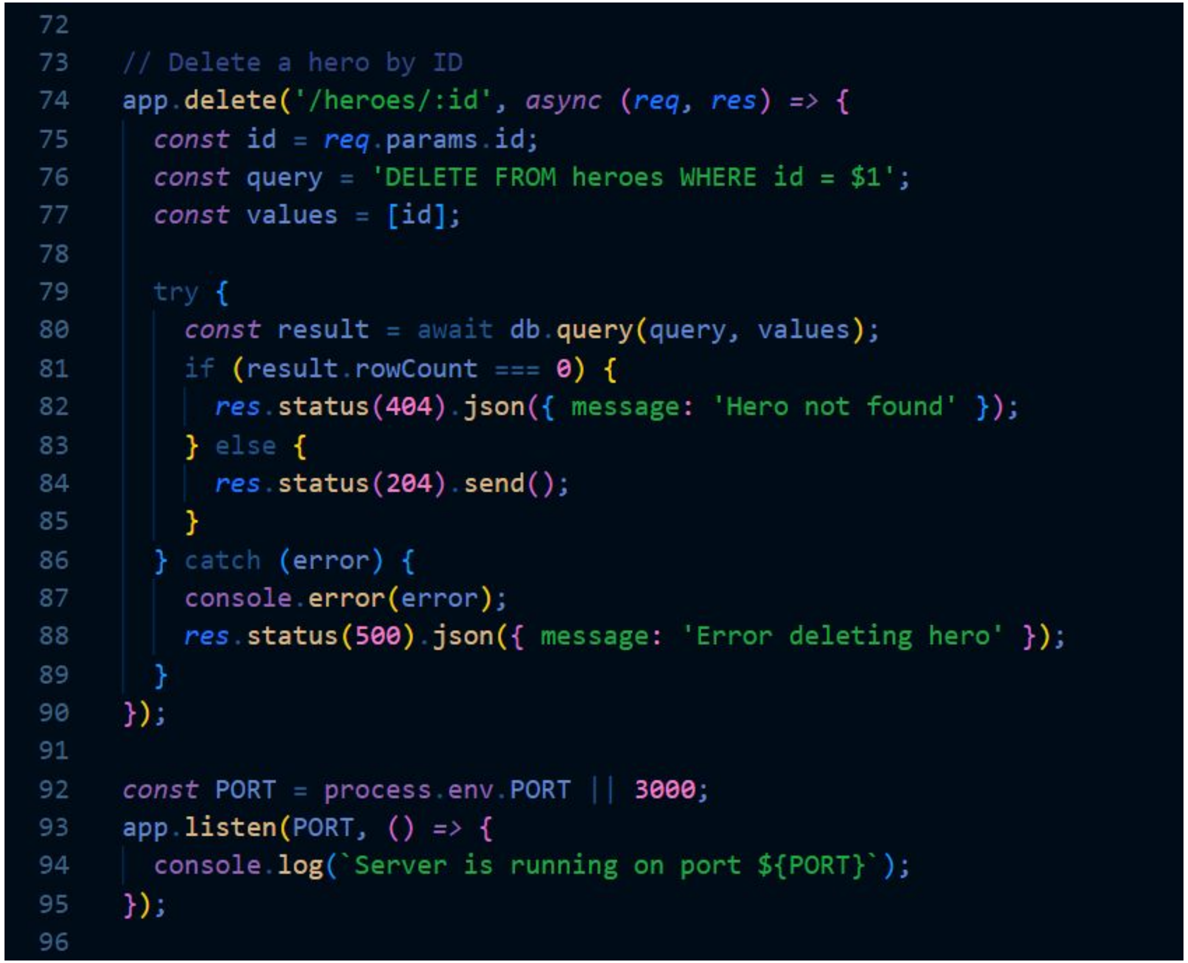
```javascript
// Get all heroes
app.get('/heroes', async (req, res) => {
  try {
    const result = await db.query('SELECT * FROM heroes');
    res.status(200).json(result.rows);
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error fetching heroes' });
  }
});

// Get a hero by ID
app.get('/heroes/:id', async (req, res) => {
  const id = req.params.id;
  const query = 'SELECT * FROM heroes WHERE id = $1';
  const values = [id];

  try {
    const result = await db.query(query, values);
    if (result.rows.length === 0) {
      res.status(404).json({ message: 'Hero not found' });
    } else {
      res.status(200).json(result.rows[0]);
    }
  } 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;
  const query = 'UPDATE heroes SET name = $1, superpower = $2 WHERE id = $3 RETURNING *';
  const values = [name, superPower, id];

  try {
    const result = await db.query(query, values);
    if (result.rows.length === 0) {
      res.status(404).json({ message: 'Hero not found' });
    } else {
      res.status(200).json(result.rows[0]);
    }
  } catch (error) {
    console.error(error);
    res.status(500).json({ message: 'Error updating hero' });
  }
});

```

app.js



```javascript

// Delete a hero by ID
app.delete('/heroes/:id', async (req, res) => {
  const id = req.params.id;
  const query = 'DELETE FROM heroes WHERE id = $1';
  const values = [id];

  try {
    const result = await db.query(query, values);
    if (result.rowCount === 0) {
      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.

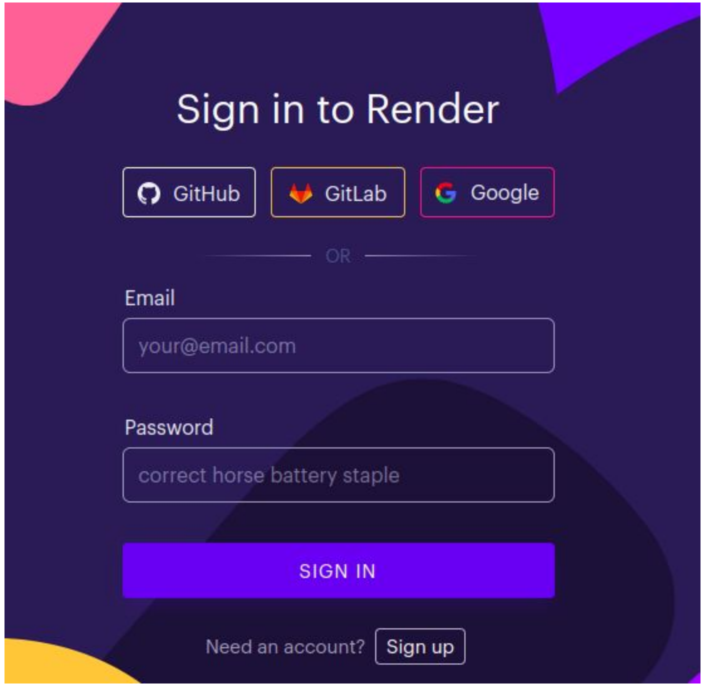
```
DB_USER=your_db_user
DB_HOST=your_db_host
DB_NAME=your_db_name
DB_PASSWORD=your_db_passsword
DB_PORT=26257
```

Deploy on Render

At this point, our application is set. Let’s proceed to deploy it on Render. Sign up for free on Render with your GitHub, GitLab, or Gmail account to get started.

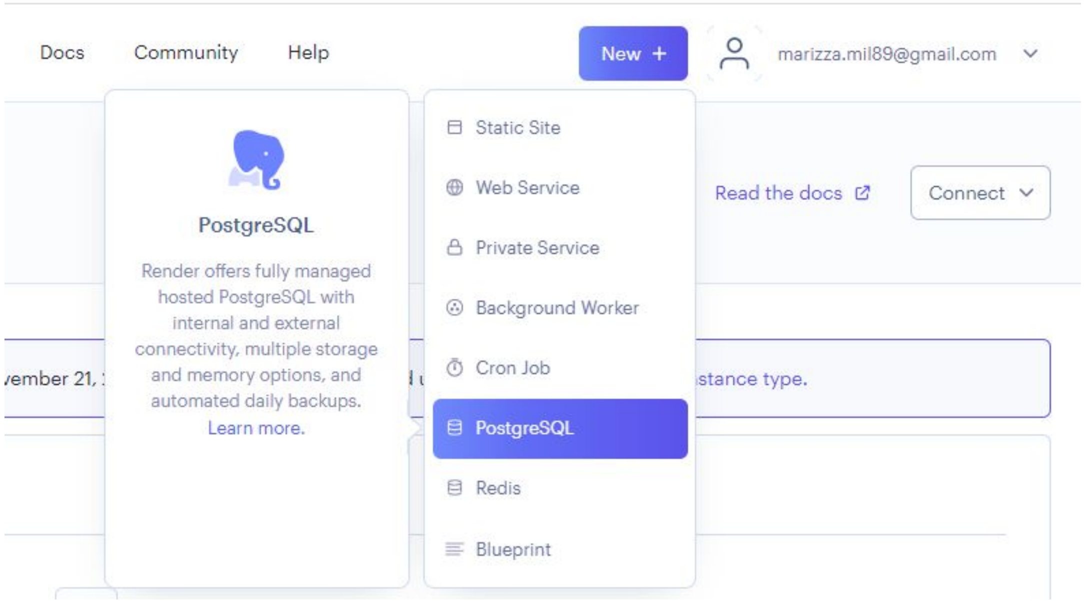
Once you’ve signed up and confirmed your email, you’ll be redirected to your Render dashboard.

Then click on the **New +** button to select the service you want to host.



Host the database

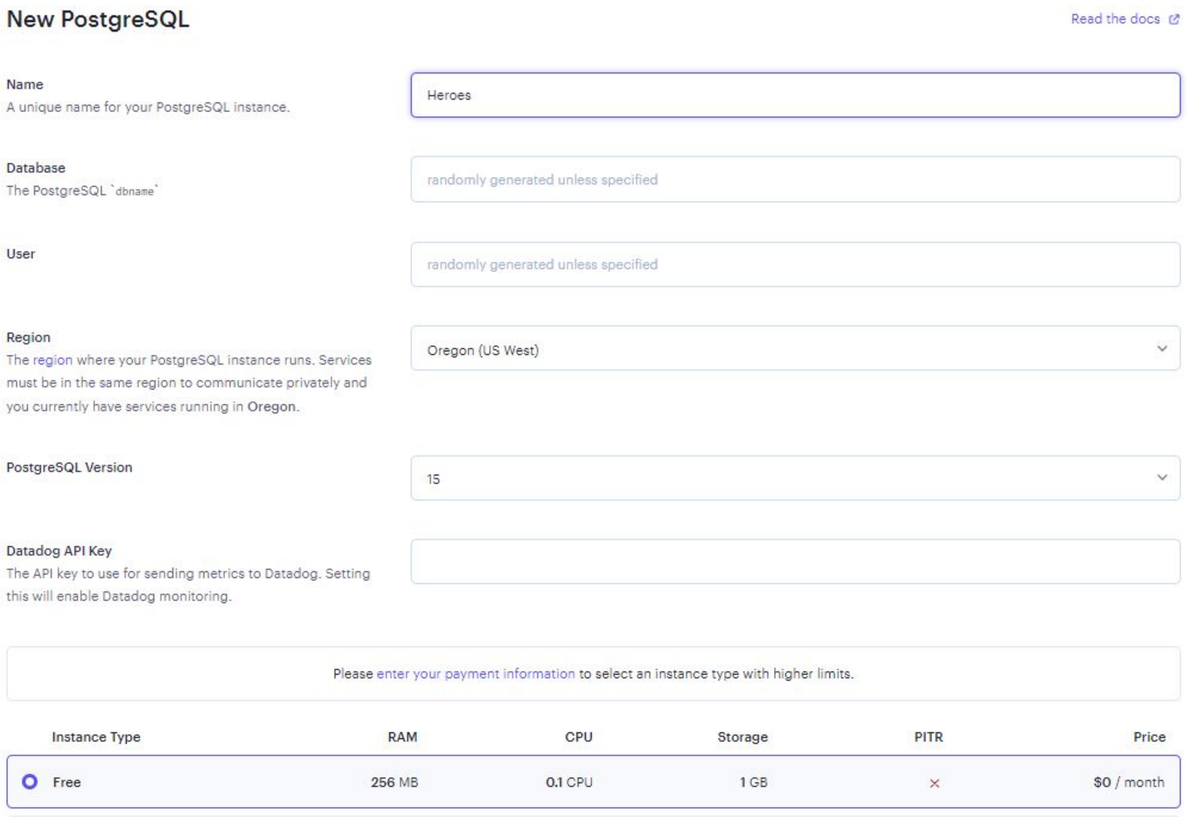
We'll start by hosting our Postgres database, so select PostgreSQL from the dropdown list.



Host the database

Next, enter the details for the database. Enter the name and leave the type as free tier. Choose at least version 14.

Then press the **Create Database** button and save the credentials in a safe place.



New PostgreSQL

Read the docs

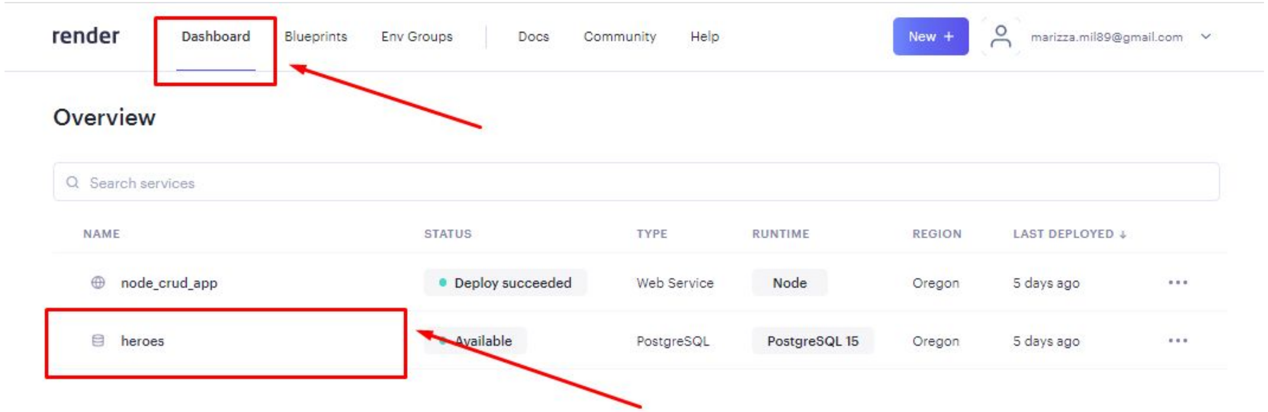
Name A unique name for your PostgreSQL instance.	Heroes
Database The PostgreSQL `dbname`	randomly generated unless specified
User	randomly generated unless specified
Region The region where your PostgreSQL instance runs. Services must be in the same region to communicate privately and you currently have services running in **Oregon**.	Oregon (US West)
PostgreSQL Version	15
Datadog API Key The API key to use for sending metrics to Datadog. Setting this will enable Datadog monitoring.	

Please enter your payment information to select an instance type with higher limits.

Instance Type	RAM	CPU	Storage	PITR	Price
○ Free	256 MB	0.1 CPU	1 GB	×	$0 / month

Host the database

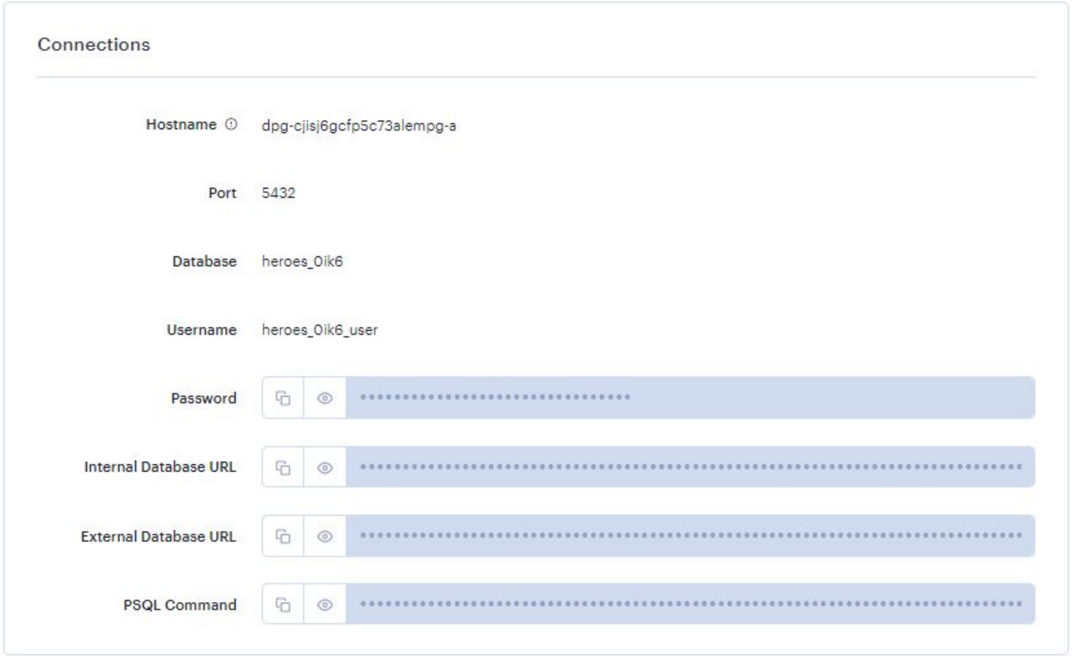
- Go to Dashboard
- Click on the created database



NAME	STATUS	TYPE	RUNTIME	REGION	LAST DEPLOYED ↓
node_crud_app	Deploy succeeded	Web Service	Node	Oregon	5 days ago
heroes	Available	PostgreSQL	PostgreSQL 15	Oregon	5 days ago

Host the database

Update the code in the `.env` file to use the credentials from Render:

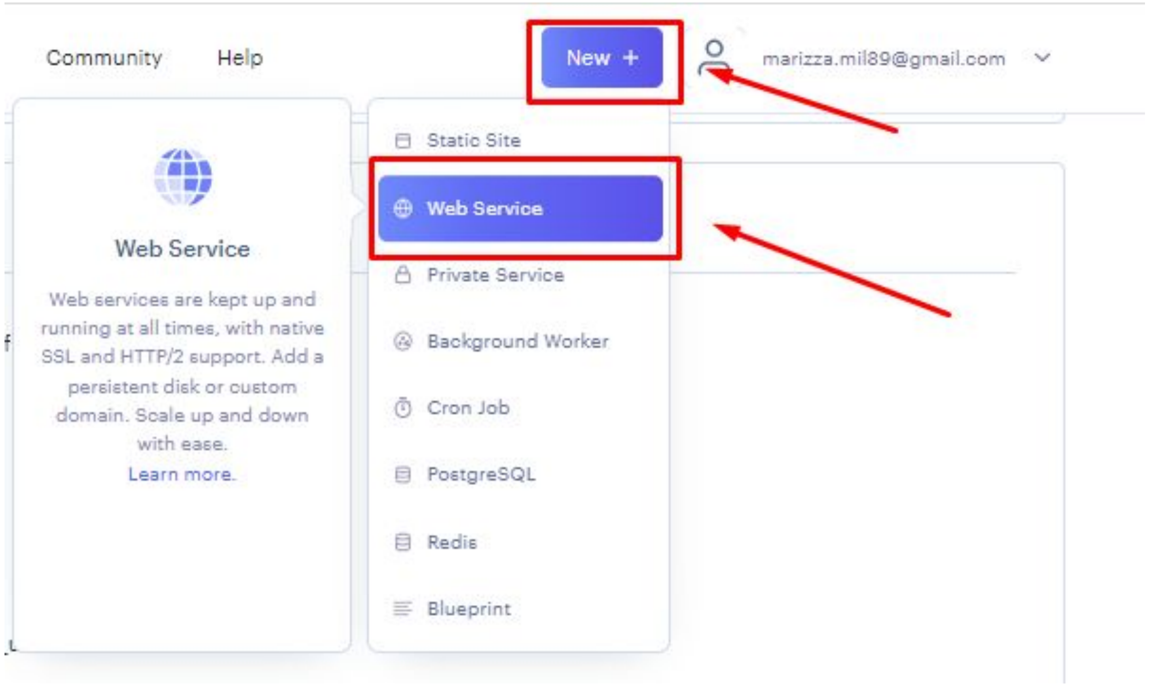


Connections

Hostname	dpg-cjisj6gcfp5c73alempg-a
Port	5432
Database	heroes_0ik6
Username	heroes_0ik6_user
Password	••••••••••••••••••••••••••••••••
Internal Database URL	••
External Database URL	••
PSQL Command	••

Host the App

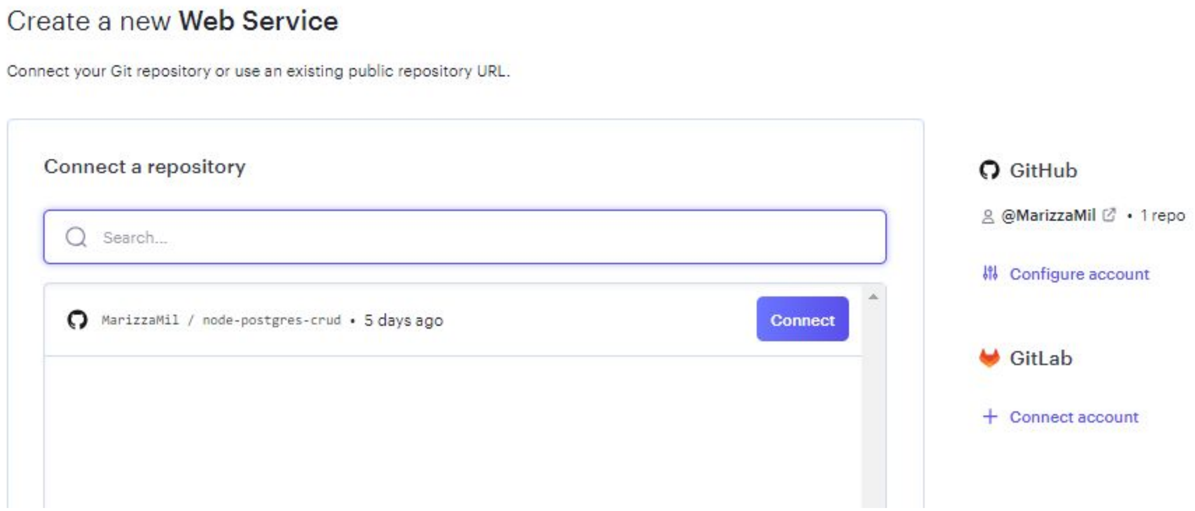
Push the code to GitHub. Now click on the **Web Service** tab from your Render dashboard:



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.
Learn more.

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.

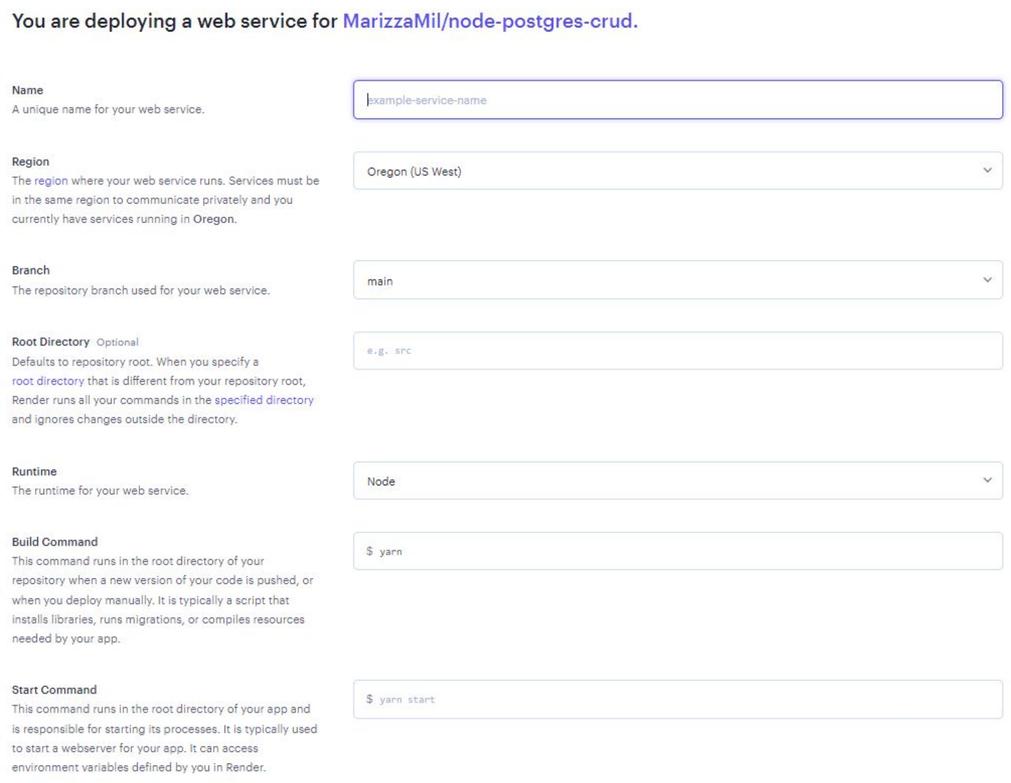
Connect a repository

MarizzaMil / node-postgres-crud • 5 days ago

Host the App

Next, enter the following details:

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



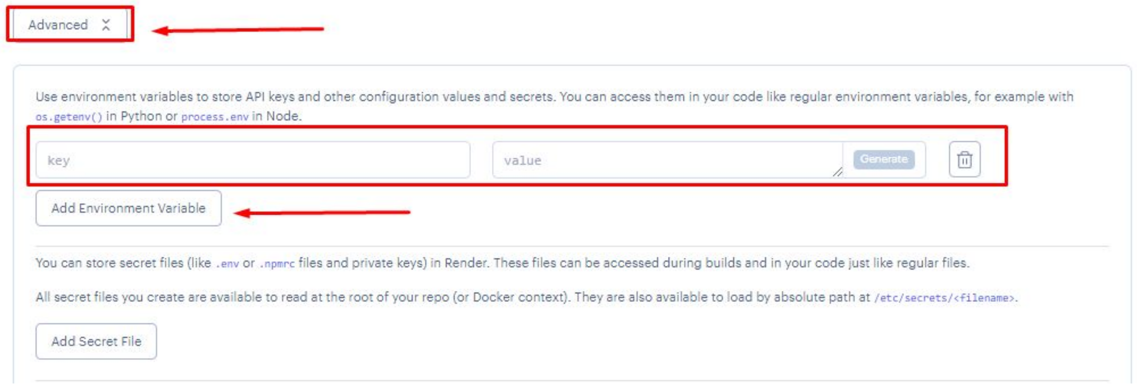
You are deploying a web service for MarizzaMil/node-postgres-crud.

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:

```
HOSTNAME= <HOSTNAME>
USERNAME= <USERNAME>
PASSWORD= <PASSWORD>
DATABASE_NAME= <DATABASE_NAME>
PORT = 5432
```



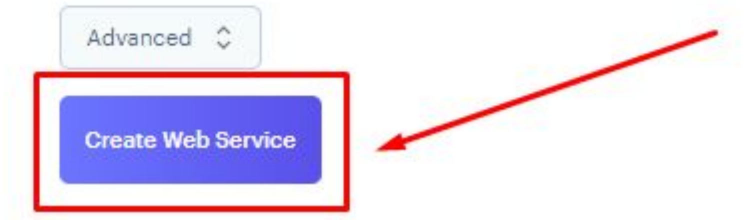
Use environment variables to store API keys and other configuration values and secrets. You can access them in your code like regular environment variables, for example with `os.getenv()` in Python or `process.env` in Node.

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>`.

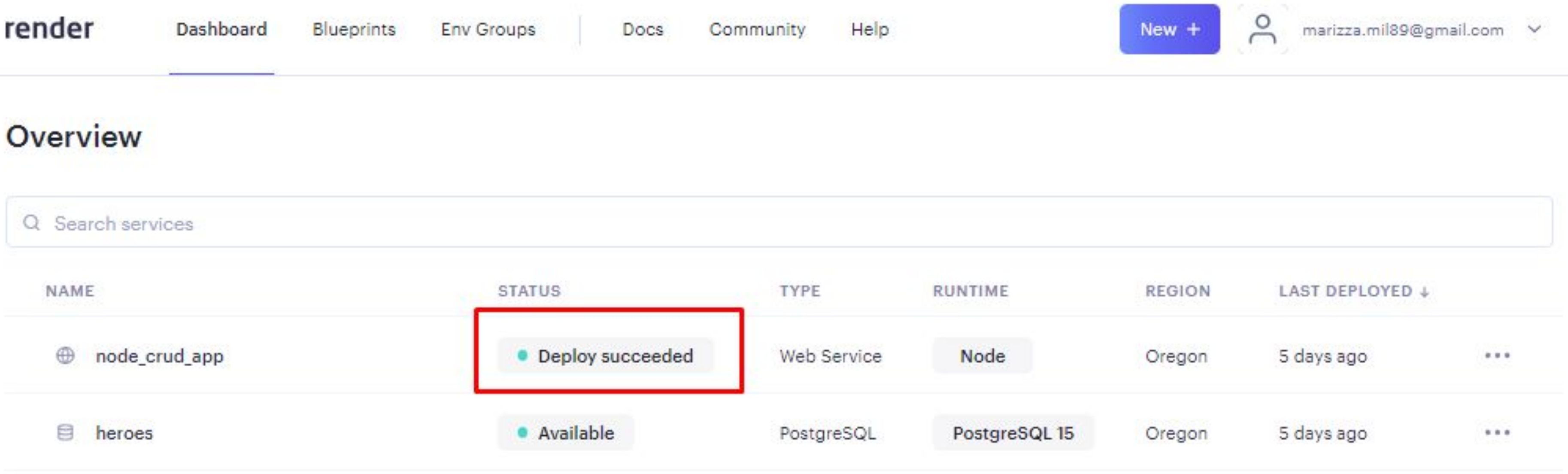
Host the App

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



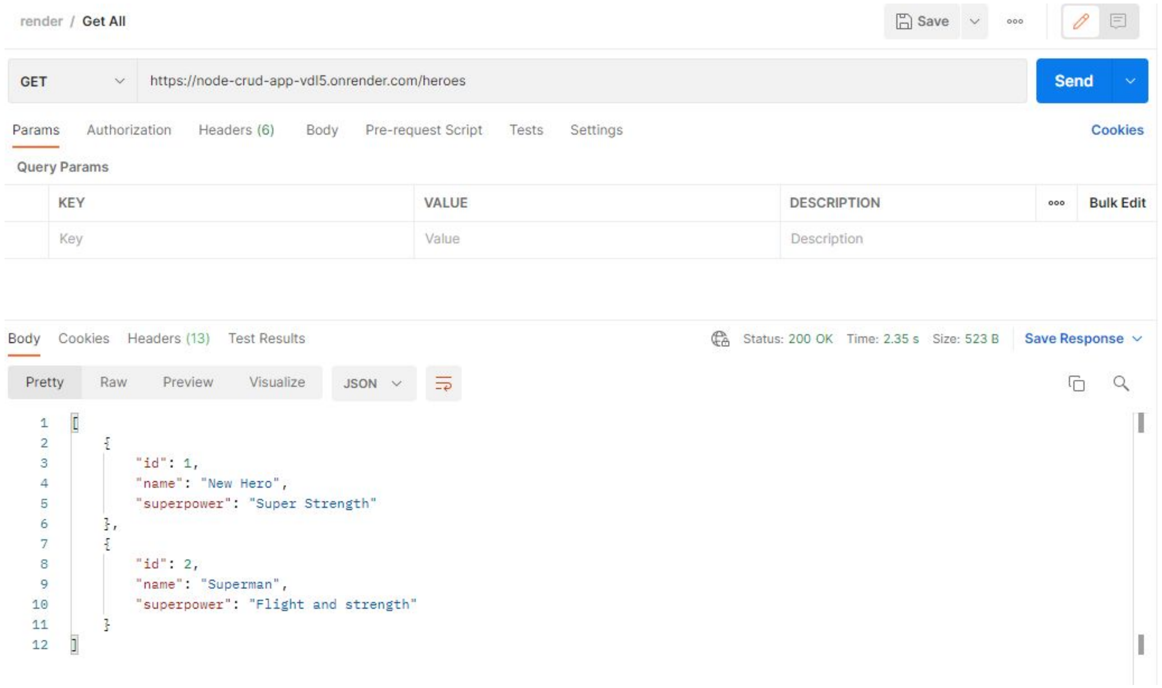
Host the App

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



NAME	STATUS	TYPE	RUNTIME	REGION	LAST DEPLOYED ↓
node_crud_app	Deploy succeeded	Web Service	Node	Oregon	5 days ago
heroes	Available	PostgreSQL	PostgreSQL 15	Oregon	5 days ago

Test the API using Postman



https://node-crud-app-vdl5.onrender.com/heroes

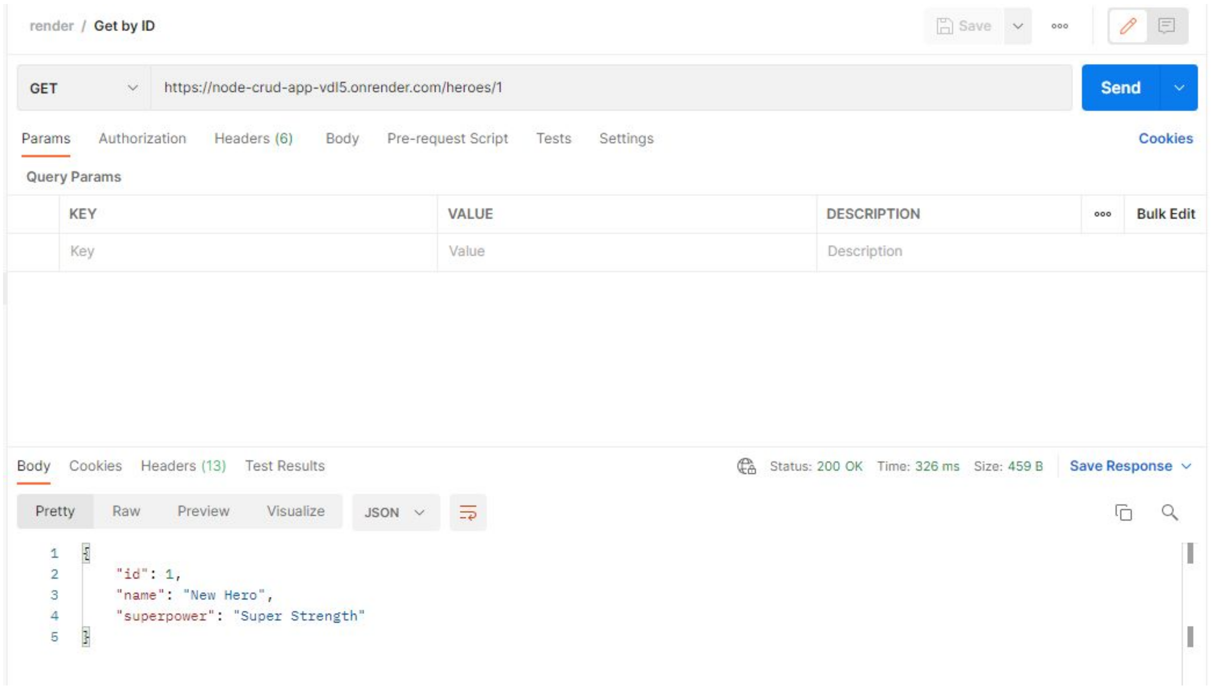
KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 200 OK Time: 2.35 s Size: 523 B

```
[
    {
        "id": 1,
        "name": "New Hero",
        "superpower": "Super Strength"
    },
    {
        "id": 2,
        "name": "Superman",
        "superpower": "Flight and strength"
    }
]
```

GET_REQUEST

Test the API using Postman



render / Get by ID

GET https://node-crud-app-vdl5.onrender.com/heroes/1

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

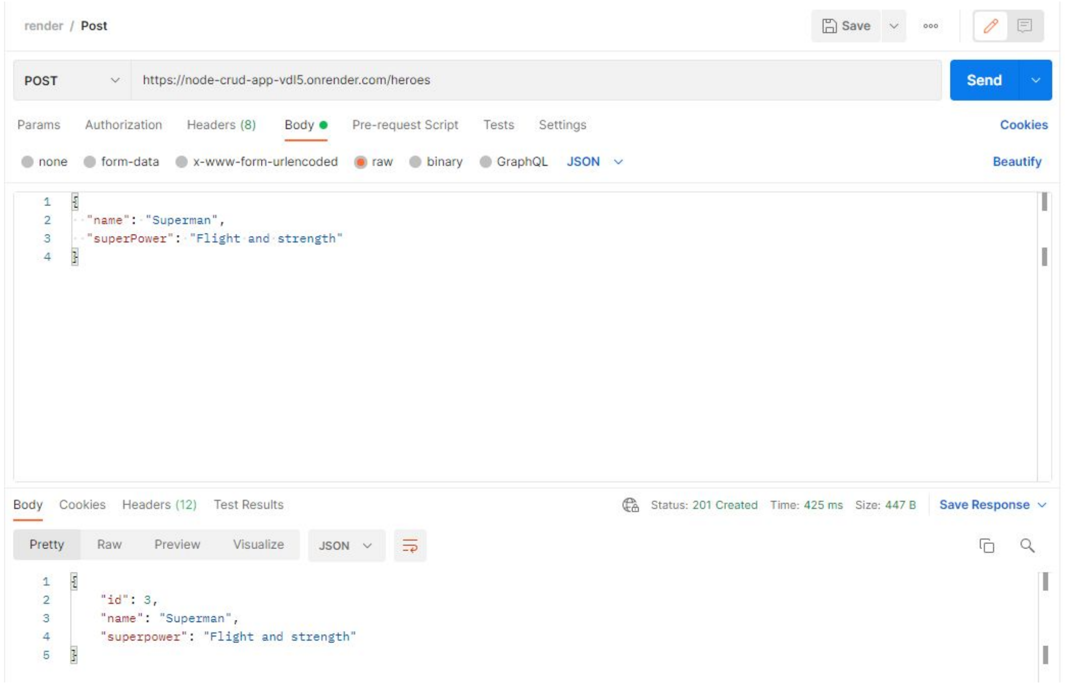
Body Cookies Headers (13) Test Results Status: 200 OK Time: 326 ms Size: 459 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "id": 1,
    "name": "New Hero",
    "superpower": "Super Strength"
}
```

GET BY ID REQUEST

Test the API using Postman



render / Post

POST https://node-crud-app-vdl5.onrender.com/heroes

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

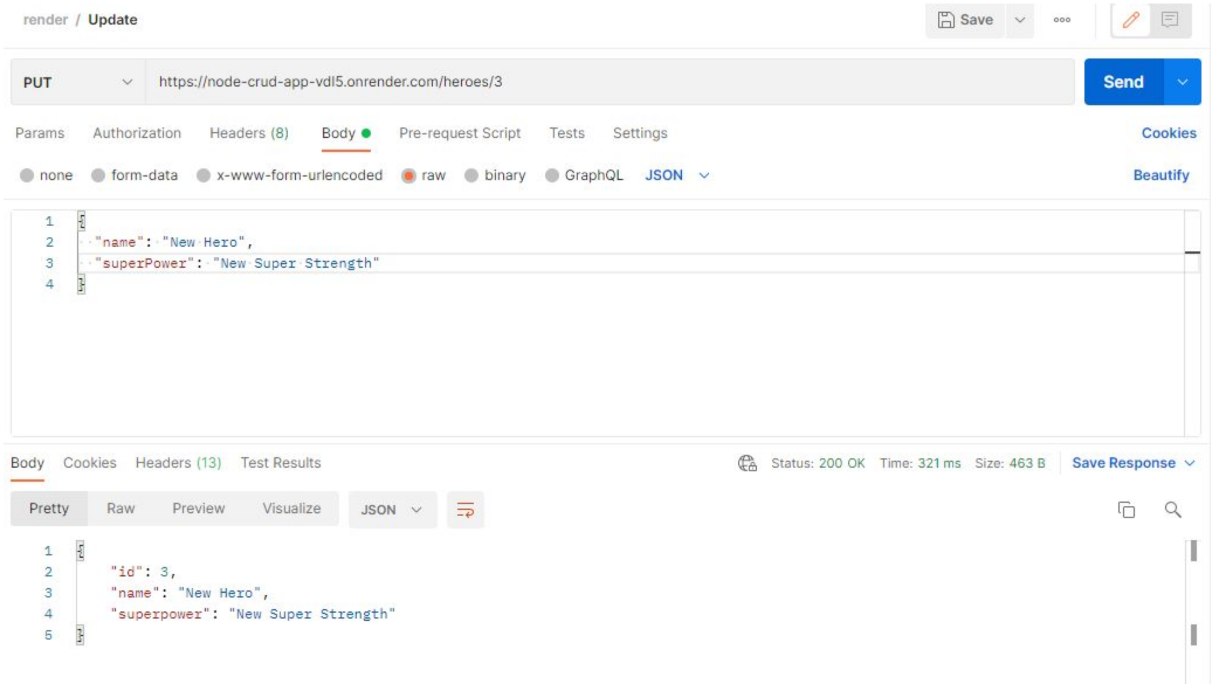
Status: 201 Created Time: 425 ms Size: 447 B

```
{
    "id": 3,
    "name": "Superman",
    "superpower": "Flight and strength"
}
```

POST_REQUEST

Test the API using Postman

UPDATE_REQUEST



render / Update

Save

PUT https://node-crud-app-vdl5.onrender.com/heroes/3

Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON Beautify

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

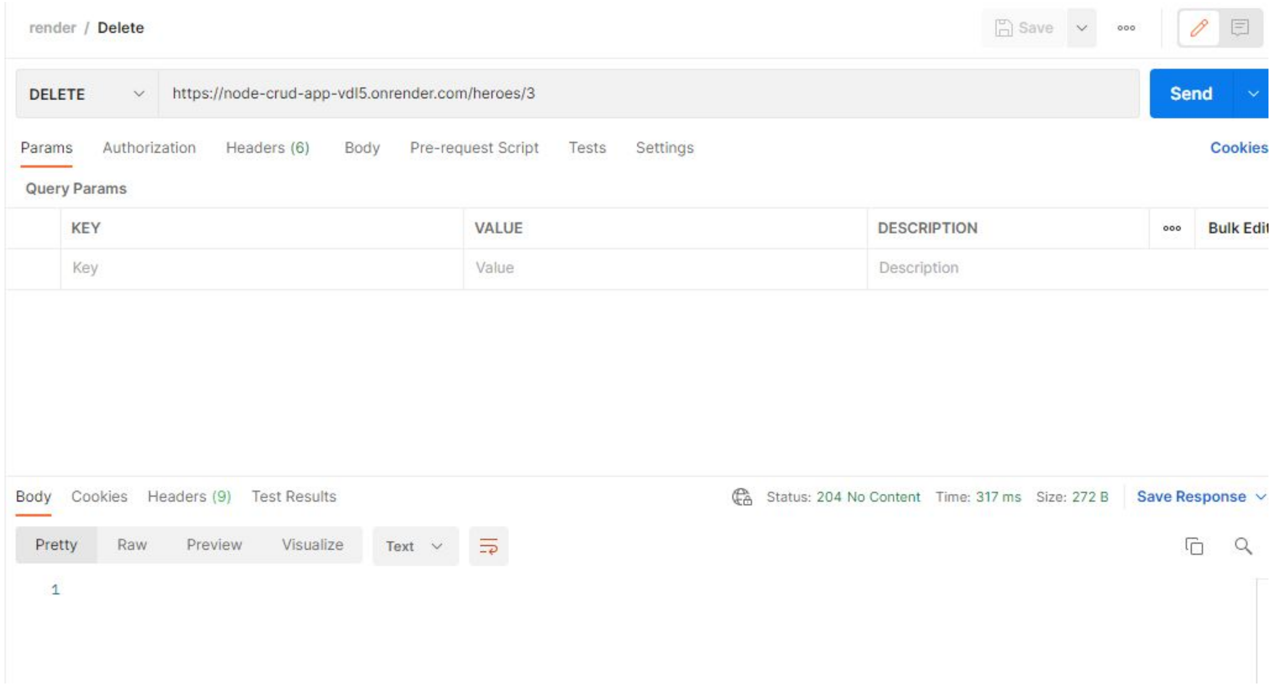
Body Cookies Headers (13) Test Results Status: 200 OK Time: 321 ms Size: 463 B Save Response

Pretty Raw Preview Visualize JSON

```
{
    "id": 3,
    "name": "New Hero",
    "superpower": "New Super Strength"
}
```

Test the API using Postman

DELETE REQUEST


render / Delete

DELETE https://node-crud-app-vdl5.onrender.com/heroes/3

Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (9) Test Results

Status: 204 No Content Time: 317 ms Size: 272 B

Save Response

Pretty Raw Preview Visualize Text

```
1
```

Conclusion

This tutorial taught us how to deploy a Node.js and PostgreSQL App on Render. First, we started by introducing Render. Then, as a demonstration, we built a Node.js RESTful web application to manage a task and deployed it on Render.

Render is an exciting tool, and I recommend checking out the documentation to learn more. I hope you enjoyed this article and happy coding!