

## PROMPT

Jan 19, 2026, 10:52 AM

Write a step-by-step tutorial for Nuxt aimed at beginner developers for connecting their frontend to Neon. Include prerequisites, clear learning objectives, hands-on exercises, and troubleshooting tips.

## Response from Kimi-K2-Instruct

markdown

```
# Beginner's Guide: Connect a Nuxt 3 Frontend to Neon Postgres

> Learning Goal
> By the end of this 30-minute tutorial you will have a working Nuxt 3 app that reads and writes data
hosted on [Neon](https://neon.tech). You will also know how to keep sensitive credentials out of your
Git history.

---

## 1. Prerequisites
- Node 18+ and npm (or pnpm/yarn) installed
- A free Neon account ([https://neon.tech](https://neon.tech))
- Basic familiarity with Vue/Nuxt components (no Postgres experience required)
- Git (optional, but recommended)

---

## 2. Learning Objectives
1. Create a Neon project and obtain a connection string.
2. Add a .env file to Nuxt and load it safely with the runtimeConfig.
3. Install & configure @neondatabase/serverless (Postgres driver).
4. Build a simple API route (/server/api/tasks.get.ts) that queries Neon.
5. Display the data in a Vue component.
6. Deploy the finished project to Vercel/Netlify without leaking credentials.

---

## 3. Hands-On Walk-Through

### 3.1 Create a Neon Project
1. Log in 'Create project' choose the region closest to you.
2. Keep the default database name (neondb).
3. Copy the connection string (it looks like postgres://<user>:<pwd>@<host>/neondb).
   Save it somewhere safe—you'll need it in 3.3.

### 3.2 Scaffold a Nuxt 3 App
```

```
npx nuxi@latest init neon-nuxt-demo
```

```
cd neon-nuxt-demo
```

```
npm install
```

```
Run the starter to verify everything works:
```

```
npm run dev
```

Visit <http://localhost:3000> 'you should see the Nuxt welcome page.

```
### 3.3 Install the Postgres Driver
```

```
npm install @neondatabase/serverless
```

```
### 3.4 Store the Neon URL Safely  
Create a file called `.env` in the root of the project:
```

## `.env`

```
NEONDATABASEURL=postgres://alex:AbCdEf@ep-cold-dark-  
ness-123456.us-east-1.aws.neon.tech/neondb?sslmode=require
```

```
> Important  
> Add `.env` to `.gitignore` so you never commit secrets.
```

```
### 3.5 Expose the URL to Nuxt Runtime  
Edit nuxt.config.ts`:
```

```
// nuxt.config.ts  
  
export default defineNuxtConfig({  
  runtimeConfig: {  
    // Private keys (only available on server-side)  
    neonUrl: process.env.NEONDATABASEURL,  
    // Public keys (exposed to client)  
    public: {}  
  }  
})
```

```
### 3.6 Create a Simple Table in Neon  
Open the SQL Editor in the Neon console and run:
```

```
CREATE TABLE tasks (  
  id SERIAL PRIMARY KEY,  
  title TEXT NOT NULL,  
  done BOOLEAN DEFAULT FALSE
```

```
);  
INSERT INTO tasks (title, done)  
VALUES  
( 'Learn Neon', true ),  
( 'Connect Nuxt', false );
```

```
### 3.7 Build the API Route  
Create the folder structure:  
`server/api/tasks.get.ts`  
  
Paste the following:
```

```
// server/api/tasks.get.ts  
  
import { neon } from '@neondatabase/serverless'  
export default defineEventHandler(async (event) => {  
  // 1. Grab the private runtime URL  
  const url = useRuntimeConfig().neonUrl  
  // 2. Connect to Neon  
  const sql = neon(url)  
  // 3. Run query  
  const rows = await sql`SELECT * FROM tasks ORDER BY id`  
  // 4. Return JSON  
  return rows  
})
```

```
Test it in the browser:  
[http://localhost:3000/api/tasks](http://localhost:3000/api/tasks)  
  
Expected JSON result:
```

```
[  
  { "id": 1, "title": "Learn Neon", "done": true },  
  { "id": 2, "title": "Connect Nuxt", "done": false }  
]
```

```
### 3.8 Display Tasks in a Vue Component  
Replace `app.vue` (or create `pages/index.vue` if you enabled pages) with:
```

```
<template>
```

```
<section class="p-8">
<h1 class="text-2xl font-bold mb-4">My Neon Tasks</h1>
<ul class="list-disc pl-6">
<li v-for="task in tasks" :key="task.id">
<span :class="{ 'line-through': task.done }">
  {{ task.title }}
</span>
</li>
</ul>
</section>
</template>
<script setup>
// Fetch once on server-side (SSR)
const { data: tasks } = await $fetch('/api/tasks')
</script>
```

Refresh the page ' you should see the two tasks rendered.

```
### 3.9 Add a New Task (Optional Exercise)
Create `server/api/tasks.post.ts`:
```

```
// server/api/tasks.post.ts
import { neon } from '@neondatabase/serverless'
export default defineEventHandler(async (event) => {
  const body = await readBody(event) // { title: string }
  const url = useRuntimeConfig().neonUrl
  const sql = neon(url)
  const result = await sql`
INSERT INTO tasks (title, done)
VALUES (${body.title}, false)
RETURNING *
`
  return result[0]
})
```

In your component add a small form that `POST`s to `/api/tasks` and re-fetches the list.

This reinforces **POST** handling and reactivity.

---

#### ## 4. Troubleshooting Tips

Symptom	Likely Cause & Fix
`ECONNREFUSED` or `getaddrinfo ENOTFOUND`	Neon URL is wrong or contains spaces. Re-copy from the console.
`relation "tasks" does not exist`	You are connected to a different database. Check the Neon <b>branch</b> and <b>database</b> name in the URL.
`500 internal server error` in `/api/tasks`	Check the terminal where `npm run dev` is running—Node will print the stack trace.
Environment variable is `undefined`	Make sure the key is under `runtimeConfig` (not `public`) and the <code>.env` file is in the <b>project root</b>, then restart the dev server.</code>
Vercel build fails after git push	Add <code>NEON_DATABASE_URL` to your Vercel dashboard under <b>Settings</b> `Environment Variables`.</code>

---

#### ## 5. Quick Checklist for Production

- [ ] Use **Neon pooled connection string** (`?pgbouncer=true`)` if you expect many concurrent users.
- [ ] Enable **SSL** (`?sslmode=require`)`—Neon enforces it anyway.
- [ ] Restrict your Neon user to the minimum required privileges.
- [ ] Rotate credentials periodically and store them in your platform's **secret manager**.
- [ ] Turn on **Point-in-Time Restore** in Neon for disaster recovery.

---

#### ## 6. Next Steps / Ideas

- Swap the inline SQL for an ORM such as **Drizzle ORM** or **Prisma**.
- Add authentication (NuxtAuth) and store user-specific tasks.
- Wire up **server-sent events** or **WebSockets** to reflect live changes.
- Deploy to **Edge** functions (Vercel, Netlify, Cloudflare)—Neon's serverless driver is optimized for that.

**Happy coding!** =€