PROMPT Jan 19, 2026, 09:10 PM

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

=B Beginner’s Guide: Connect a Nuxt 3 Frontend to Neon Postgres

step-by-step

Prerequisites

- Node.js e 18 installed (node -v)
- A free Neon account ’ https://neon.tech
- Git & GitHub (optional, for deployment)
- Basic JavaScript / Vue knowledge (no Postgres experience required)

Learning Objectives

By the end of this tutorial you will be able to:

1. Create a Neon Postgres database and obtain a connection string.
2. Add a server-only API route in Nuxt that queries the database with Drizzle ORM.
3. Display the data safely in a Nuxt page (SSR & client-side hydration).
4. Handle common connection errors and deploy to production.

Part 1 – Neon Setup (5 min)

1. Sign-up ’ Neon dashboard ’ “Create a project”.
2. Choose the region closest to you ’ “Create project”.
3. After the project spins up, click the “Connect” button ’ “.env” tab.

Copy the Pooled connection string (starts with postgres://...).

Part 2 – Nuxt Project Bootstrap (5 min)

bash

```
# 1. Scaffold
npx nuxi@latest init neon-nuxt-app
cd neon-nuxt-app

# 2. Install dependencies we’ll use
npm install drizzle-orm postgres drizzle-orm/neon-serverless
npm install -D drizzle-kit dotenv
```

Add a .env file in the root (git-ignored by default):

```bash
# .env
DATABASE_URL="postgres://username:pass-
word@ep-cool-name-pool.us-east-1.aws.neon.tech/neondb?sslmode=require"
```

Part 3 – Database Schema & Migrations

Create folder structure:

```
neon-nuxt-app/
 server/
  db/
  schema.ts
  drizzle.config.ts
```

server/db/schema.ts

```ts
import { pgTable, serial, text, timestamp } from 'drizzle-orm/pg-core'

export const messages = pgTable('messages', {
 id: serial('id').primaryKey(),
 name: text('name').notNull(),
 body: text('body').notNull(),
 createdAt: timestamp('created_at').defaultNow()
})
```

drizzle.config.ts (root level)

```ts
import type { Config } from 'drizzle-kit'
export default {
 schema: './server/db/schema.ts',
 out: './server/db/migrations',
 driver: 'pg',
 dbCredentials: { connectionString: process.env.DATABASE_URL! }
} satisfies Config
```

Run first migration (creates table):

```bash
npx drizzle-kit generate:pg
npx drizzle-kit push:pg   # applies directly to Neon
```

Part 4 – Server-Only API Route

server/api/messages.get.ts

```ts
import { neon } from '@neondatabase/serverless'
import { drizzle } from 'drizzle-orm/neon-http'
import { messages } from '../db/schema'
import { desc } from 'drizzle-orm'

export default defineEventHandler(async (event) => {
 const sql = neon(process.env.DATABASE_URL!)
 const db = drizzle(sql)

 const rows = await db.select().from(messages).orderBy(desc(messages.createdAt))
 return rows
})
```

> *Security note: Because the file is inside server/, Nuxt never bundles it to the client.*

Part 5 – Frontend Page (SSR)

pages/index.vue

```vue
<script setup>
// This runs on server + client (SSR + hydration)
const { data: msgs, pending, error, refresh } = await useFetch('/api/messages')
</script>

<template>
<h1>Guestbook</h1>
<p v-if="pending">Loading…</p>
<p v-else-if="error"> {{ error.data?.message || 'Network error' }}</p>
<ul v-else>
<li v-for="m in msgs" :key="m.id">
<strong>{{ m.name }}</strong>: {{ m.body }}
</li>
</ul>
</template>
```

Run the dev server:

```bash
npm run dev
```

Visit http://localhost:3000 ’ you should see an empty list (no rows yet).

Part 6 – Hands-On Exercise: Add a Message

1. Create server/api/messages.post.ts

```ts
import { neon } from '@neondatabase/serverless'
import { drizzle } from 'drizzle-orm/neon-http'
import { messages } from '../db/schema'

export default defineEventHandler(async (event) => {
 const { name, body } = await readBody(event)

 if (!name || !body) throw createError({ status: 400, statusMessage: 'Missing fields' })

 const sql = neon(process.env.DATABASE_URL!)
 const db = drizzle(sql)

 const [inserted] = await db.insert(messages).values({ name, body }).returning()
 return inserted
})
```

2. Add a small form in pages/index.vue:

```vue
<form @submit.prevent="send">
<input v-model="form.name" placeholder="Your name" required />
<input v-model="form.body" placeholder="Message" required />
<button type="submit">Send</button>
</form>
```

```js
// inside <script setup>
const form = reactive({ name: '', body: '' })
async function send() {
 await $fetch('/api/messages', { method: 'POST', body: form })
 form.body = ''
 refresh() // reload list
}
```

Try it ’ new messages appear instantly (SSR + client hydration).

Part 7 – Troubleshooting Tips

Symptom	Likely Cause	Fix
ECONNREFUSED or getaddrinfo ENOTFOUND	Wrong DATABASE_URL	Re-copy the pooled string from Neon dashboard
relation "messages" does not exist	Migration not run	npx drizzle-kit push:pg
500 Internal error in API	Neon free tier compute suspend	First request wakes DB; retry or ping DB on app boot
CORS errors in browser	Calling Neon from client (don't!)	Keep DB calls inside server/ directory
Vercel/Netlify build fails	.env not set in dashboard	Add DATABASE_URL environment variable in hosting UI

Part 8 – Deploy (Vercel Example)

1. Push repo to GitHub.
2. Vercel ' "New Project" ' import repo.
3. Add Environment Variable DATABASE_URL (same value as local).
4. Deploy ' visit live URL ' guestbook works!

Outcome Checklist

```markdown
[ ] Neon project created & pooled string copied
[ ] Nuxt 3 project runs with Drizzle ORM
[ ] /api/messages GET returns rows
[ ] /api/messages POST inserts rows
[ ] Frontend displays & submits messages
[ ] .env ignored by git (security)
[ ] Deployed to production without errors
```

Happy! <‰

You have a full-stack Nuxt app reading & writing to Neon Postgres—ready to scale.