[개발환경] Neon Postgres과 Drizzle ORM을 활용해 테이블 만들기

구현 어려움 정도	⭐⭐⭐
구현 완료	☑
기간	@December 7, 2024 → December 8, 2024
정리 완료	☑

Neon Postgres과 Drizzle ORM을 사용하여 데이터베이스를 간단하게 세팅할 수 있습니다.

Neon Postgres

PostgreSQL을 기반으로 한 서버리스 데이터베이스 서비스(무료로도 사용 가능)

클라우드 타입에 배포한 서비스를 손쉽게 연동할 수 있음

Drizzle ORM

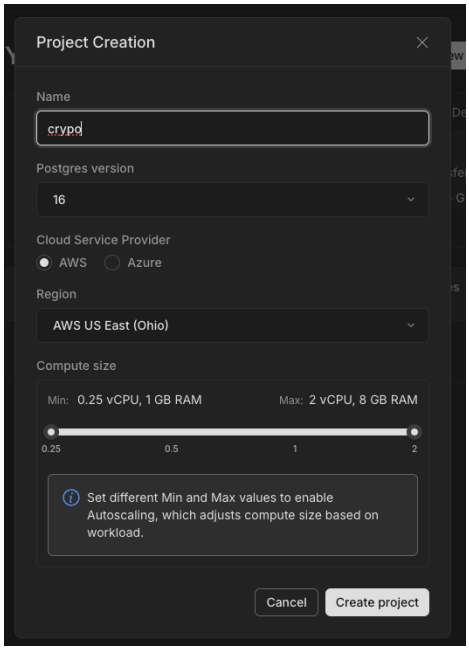
Drizzle ORM은 TypeScript/JavaScript 환경에서 사용할 수 있는 경량화된 ORM(Object Relational Mapping) 라이브러리

- ORM?

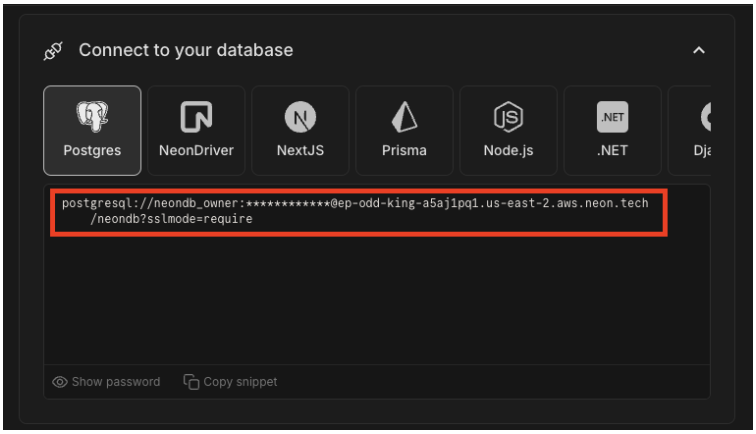
  ORM은 데이터베이스와 객체 지향 언어 간의 매핑을 자동으로 처리해 주는 도구로, SQL 쿼리를 작성하는 대신 JavaScript 코드로 데이터베이스 작업을 할 수 있게 해줍니다.

Neon Postgres과 Drizzle ORM을 활용해 DB 세팅하기

1. https://neon.tech/에 접속합니다.
2. 프로젝트 이름음 정해 Project Creation을 해줍니다.

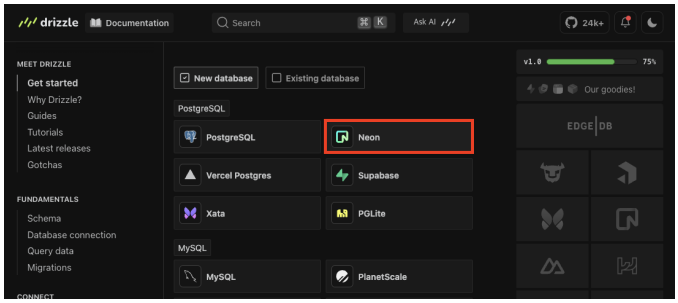


3. 프로젝트가 만들어지면 Connect to your database가 뜨는데 외부에 노출되지 않도록 키를 .env 환경변수 파일에 설정해줍니다.



- `NEON_DATABASE_URL=postgresql://neondb_owner:**********@ep-odd-king-a5aj1pq1.us-east-2.aws.neon.tech/neondb?sslmode=require` env file에 설정해줍니다.
- `**********` 에 본인의 key(password) 값을 입력해주세요.
- gitignore 파일에 env 파일이 포함되어 있는지 꼭 확인해주세요.

4. deizzle orm docimentation 사이트인 https://orm.drizzle.team/docs/get-started에 접속해 neon postgres에 들어갑니다.



Drizzle ORM - PostgreSQL
Drizzle ORM is a lightweight and performant TypeScript ORM with developer experience in mind.
https://orm.drizzle.team/docs/get-started/neon-new

5. 설치 명령어를 통해 라이브러리를 설치해줍니다.

```
// drizzle orm과 neon 데이터베이스를 연결
pnpm add drizzle-orm @neondatabase/serverless

// drizzle-kit 설치
pnpm add -D drizzle-kit tsx //
```

- `drizzle-kit`: drizzle orm의 커맨드라인 인터페이스로, drizzle kit을 통해 SQL 마이그레이션 파일을 생성해서 데이터베이스 스키마의 변경사항을 관리하거나 기존 데이터베이스의 스키마를 가져오는 등의 작업을 할 수 있음

6. db/index.ts 파일 만들어서 drizzle orm을 데이터베이스에 연결해줍니다.

```
// db/index.ts
import { neon } from '@neondatabase/serverless';
import { drizzle } from 'drizzle-orm/neon-serverless';

const sql = neon(process.env.DATABASE_URL!); // neon 데이터베이스 연결
```

```
const db = drizzle({ client: sql }); // 위에서 만들어진 neon 클라이언트 객체를 drizzle 함수에 넣어서 drizzle orm을 만
const result = await db.select().from(...) // 생성된 인스턴스의 메소드를 이용해 아래처럼 데이터베이스 query를 만들고 조작
```

- `const sql = neon(process.env.DATABASE_URL)` : neon 데이터베이스 연결
- `const db = drizzle({ client: sql })` : 위에서 만들어진 neon 클라이언트 객체를 drizzle 함수에 넣어서 drizzle orm을 만들어주는 코드
- `const result = await db.select().from(...)` : 생성된 인스턴스의 메소드를 이용해 아래처럼 데이터베이스 query를 만들고 조작할 수 있음

7. docs의 step4에 table 생성 예시를 통해 테이블을 생성

```
// src/db/schema.ts
import { integer, pgTable, varchar } from "drizzle-orm/pg-core";
export const usersTable = pgTable("users", {
  id: integer().primaryKey().generatedAlwaysAsIdentity(),
  name: varchar({ length: 255 }).notNull(),
  age: integer().notNull(),
  email: varchar({ length: 255 }).notNull().unique(),
});
```

8. drizzle.config.ts 를 예시처럼 만들어줍니다.

```
import 'dotenv/config';
import { defineConfig } from 'drizzle-kit';
export default defineConfig({
  out: './drizzle',
  schema: './src/db/schema.ts',
  dialect: 'postgresql',
  dbCredentials: {
    url: process.env.NEON_DATABASE_URL!,
  },
});
```

- `dialect` : 사용할 데이터베이스 종류를 넣어줌
- `schema` : 테이블을 선언한 스키마 파일의 경로를 넣어줌
- `out` : drizzle kit 명령어를 통해 현재의 스키마를 기반으로 마이그레이션 파일을 생성하게 되는데 그 파일들을 위치시킬 폴더
- `dbCredentials` : 환경변수

9. package.json에 명령어를 추가해줍니다.

```
"scripts": {
    ...
  "generate": "drizzle-kit generate",
  "migrate": "drizzle-kit migrate"
}
```

- `generate 명령어` : 현재 스키마를 기반으로 sql 마이그레이션 파일을 생성하는 명령어
- `migrate 명령어` : 생성된 sql 마이그레이션 파일을 실제 데이터베이스에 적용하는 명령어

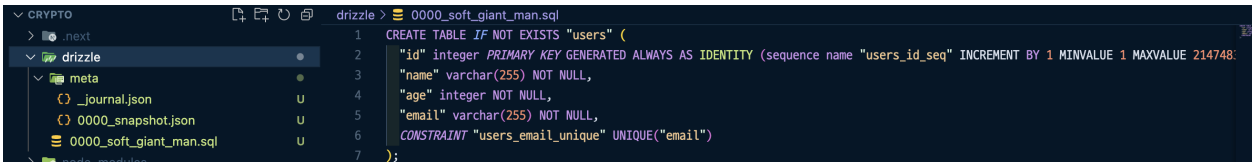
```
pnpm generate

> my-app@0.1.0 generate /Users/eunjee/Documents/file/crypto
> drizzle-kit generate

No config path provided, using default 'drizzle.config.ts'
Reading config file '/Users/eunjee/Documents/file/crypto/drizzle.config.ts'
1 tables
users 4 columns 0 indexes 0 fks

[✓] Your SQL migration file ➜ drizzle/0000_soft_giant_man.sql 🚀
```

- `generate` 명령어를 실행하면 아래처럼 drizzle 폴더가 생기고 사용한 SQL 문을 생성된 것을 확인할 수 있음

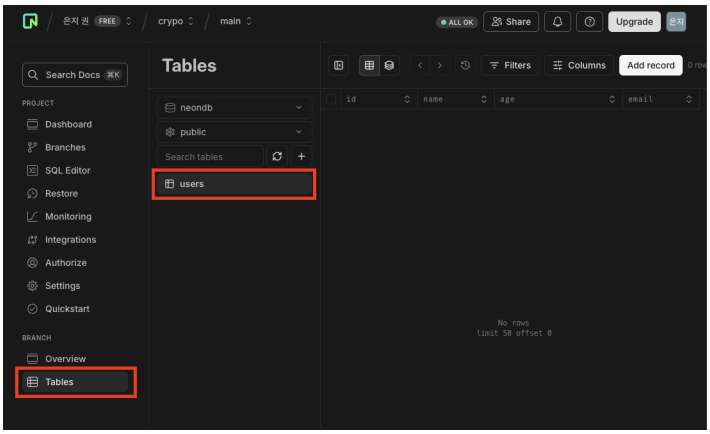



```
pnpm migrate

> my-app@0.1.0 migrate /Users/eunjee/Documents/file/crypto
> drizzle-kit migrate

No config path provided, using default 'drizzle.config.ts'
Reading config file '/Users/eunjee/Documents/file/crypto/drizzle.config.ts
'
Using '@neondatabase/serverless' driver for database querying
Warning '@neondatabase/serverless' can only connect to remote Neon/Verce
l Postgres/Supabase instances through a websocket
[✓] migrations applied successfully!
```

- `migrate` 명령어를 실행하고 neon에서 확인해보면 아래 사진처럼 user 테이블이 정상적으로 잘 생성된 것을 확인할 수 있음



사용할 테이블 정의하기

- drizzle의 pgTable함수를 통해 정의
- pgTable 함수의 첫 번째 인수로 테이블 이름을 지정해주면 됨 → "users"로 지정
- 두 번째 인수로 컬럼을 정의
- 컬럼의 타입이 오고 해당 컬럼에 제약사항이 있다면 함수 체이닝을 통해 필요한 제약사항을 추가하면 됨


```
import { pgTable, timestamp, uuid, text } from "drizzle-orm/pg-core";

export const user = pgTable('users', {
  id: uuid('id').defaultRandom().notNull().primaryKey(),
  name: text('name').notNull(),
  email: text('email').notNull().unique(),
  password: text('password').notNull(),
  createdAt: timestamp("created_at").defaultNow().notNull(),
  updatedAt: timestamp("updated_at").defaultNow().notNull(),
});
```

- `id: uuid("id")`: uuid 형식의 문자열을 ID로 만들어줌
  - `id: serial("id")`: serial type: 자동 증가하는 4바이트 정수 타입
  - 값을 넣지 않아도 랜덤하게 값이 생성될 수 있도록 `defaultRandom()` 추가
  - `notNull()` 제약사항을 추가해 null을 허용하지 않음
  - `primaryKey()` 지정을 통해 기본 키로 지정
- `unique()`: email에 중복된 값이 들어가지 않도록 제약사항 추가
- `defaultNow()` 만들어 질 때의 현재 시간을 디폴트 값으로 넣음

스키마 작성

스키마는 데이터베이스의 구조를 정의하는 정보입니다.

간단히 말하면 테이블(table)이 어떤 열(column)으로 구성되어 있고, 각각 열(column)에 어떤 데이터를 저장할지 정의하는 것입니다.

```
// src/db/schema.ts

import { relations } from "drizzle-orm";
import { pgTable, timestamp, uuid, text, varchar } from "drizzle-orm/pg-core";

export const user = pgTable('users', {
  id: uuid('id').defaultRandom().notNull().primaryKey(),
  name: text('name').notNull(),
  email: text('email').notNull().unique(),
  password: text('password').notNull(),
  createdAt: timestamp("created_at").defaultNow().notNull(),
  updatedAt: timestamp("updated_at").defaultNow().notNull(),
});
```

이렇게 작성해주고 db/index.ts에 schema를 추가해줍니다.

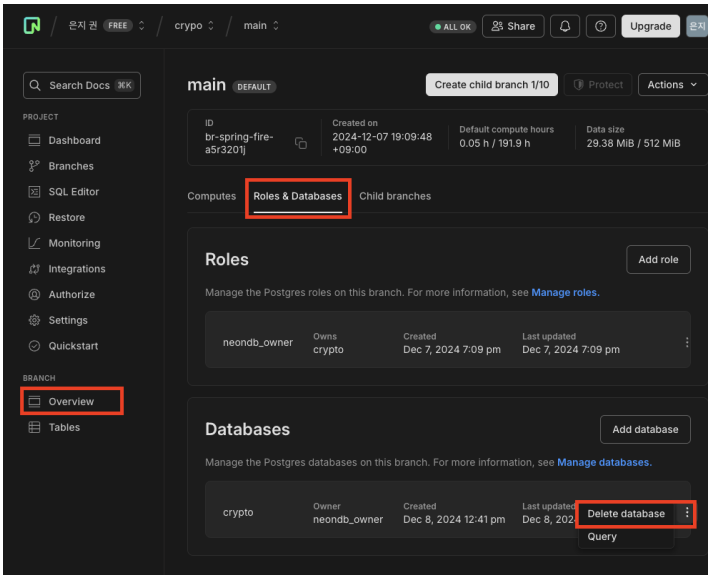
```
import { neon } from "@neondatabase/serverless";
import { drizzle } from "drizzle-orm/neon-http";
import * as schema from "./schema";

const sql = neon(process.env.NEON_DATABASE_URL!);
export const db = drizzle(sql, {schema});
```

```
export default db;
```

그리고 `pnpm generate` 명령어와 `pnpm migrate` 명령어를 실행해주면 테이블에 users 값이 잘 들어간 것을 확인할 수 있습니다.

- 참고: Overview의 Roles & Databases에서 불필요한 테이블을 삭제할 수 있습니다.



- Add record를 통해 테이블을 생성해 볼 수 있음. 값 입력이 완료되면 save 버튼 눌러주기