Lab Series: Xây Dựng Web App Căn Bản
Lab 5: Docker & Deployment

1 Mục Tiêu

- Hiểu và vận dụng các khái niệm cơ bản của Docker (Image, Container, Dockerfile).
- Thực hành triển khai (deploy) ứng dụng Web Service lên nền tảng Render.com.
- Cấu hình Cơ sở dữ liệu PostgreSQL trên nền tảng Neon.tech.

2 Tổng Quan về Docker

2.1 Khái Niệm

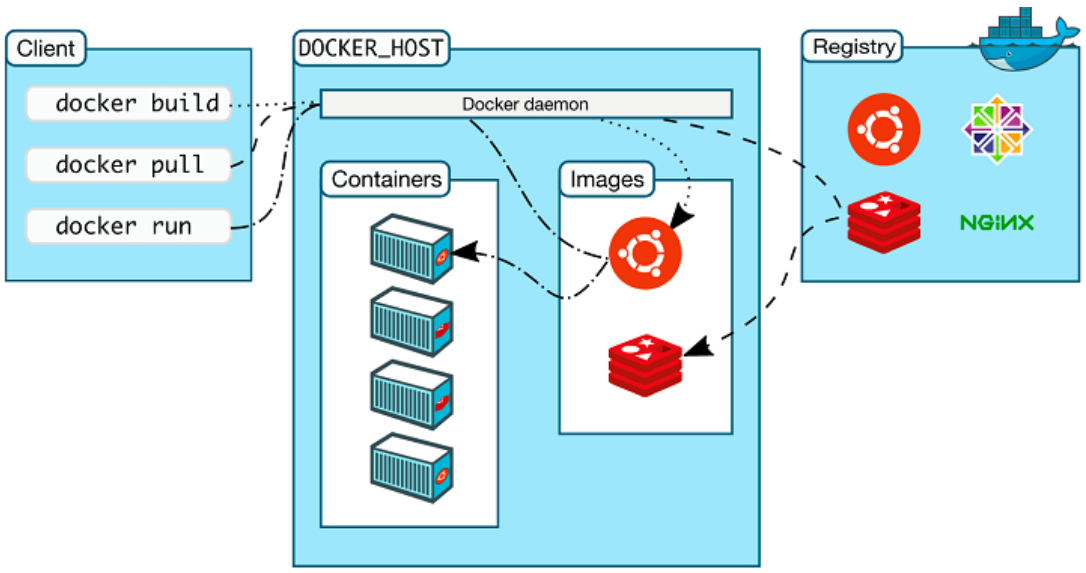
Docker là một nền tảng mở cho phép phát triển, vận chuyển và chạy các ứng dụng. Docker cho phép tách biệt ứng dụng khỏi cơ sở hạ tầng, giúp quy trình chuyển giao phần mềm trở nên nhanh chóng hơn.

Lợi ích chính của Docker:

- Đóng gói ứng dụng cùng với toàn bộ thư viện, runtime và cấu hình cần thiết vào một đơn vị chuẩn hóa.
- Đảm bảo tính nhất quán của ứng dụng trên nhiều môi trường khác nhau (máy cá nhân, máy chủ kiểm thử, hệ thống Cloud).

Docker cung cấp:

- Công cụ để xây dựng Image.
- Công cụ để tạo và quản lý Container.
- Cơ chế chia sẻ Image thông qua Registry (như Docker Hub).



2.2 Các Thành Phần Cốt Lõi

A. Docker Image

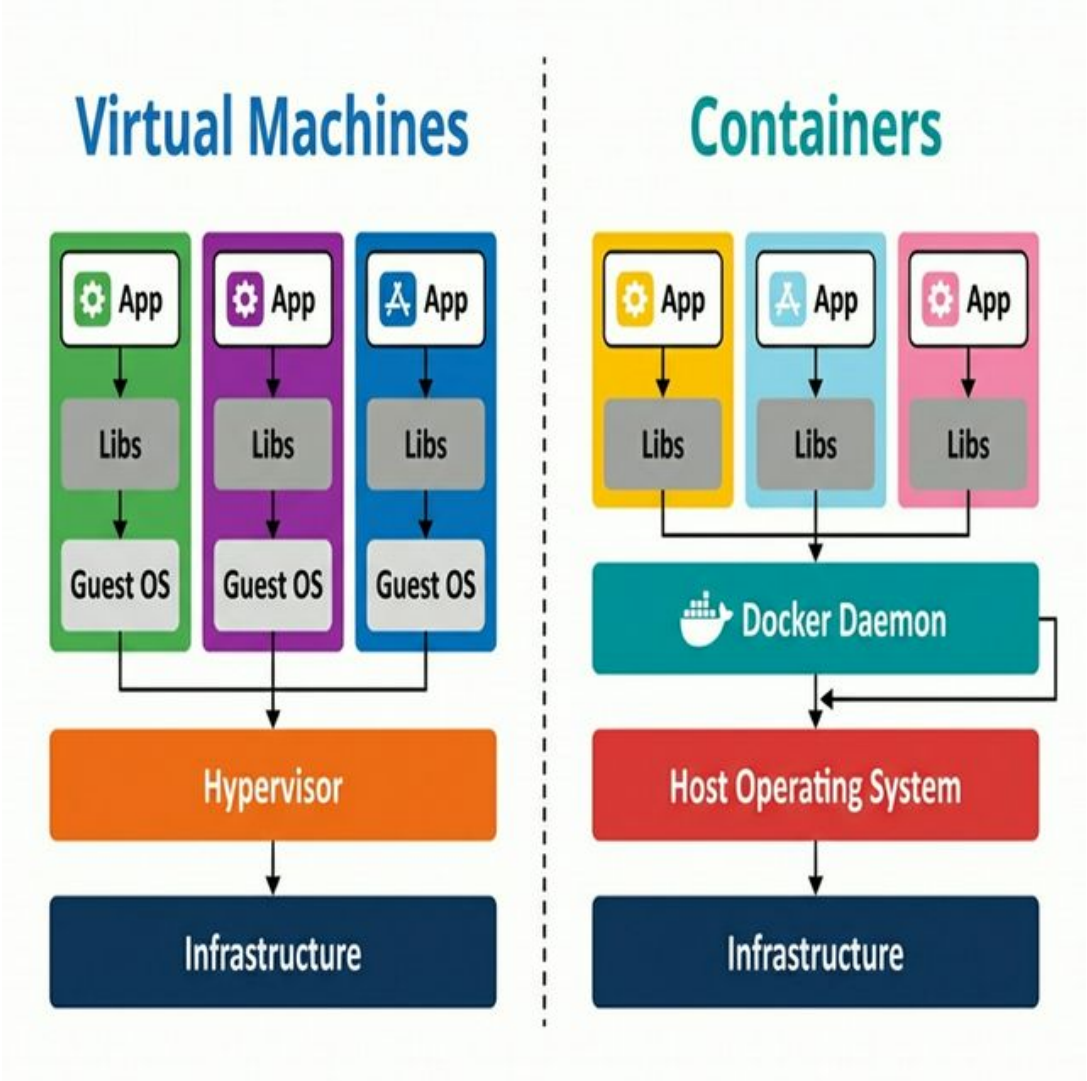
Là một gói bất biến (immutable package) bao gồm:

- Mã nguồn ứng dụng.
- Thư viện, Runtime, Cấu hình hệ thống.
- Đặc điểm: Chỉ đọc (Read-only), được xây dựng từ tập tin chỉ dẫn `Dockerfile`.
- *Ví dụ: Tương tự như đĩa cài đặt hệ điều hành (Image) có thể dùng để cài đặt trên nhiều máy tính khác nhau.*

B. Docker Container

Là một phiên bản thực thi (runtime instance) của Docker Image.

- Cung cấp môi trường cô lập để chạy ứng dụng.
- Chia sẻ kernel của máy chủ (Host OS) nhưng hoạt động trong không gian riêng biệt (tiến trình, mạng, hệ thống tập tin).
- Vòng đời: Create (Tạo) → Start (Chạy) → Stop (Dừng) → Remove (Xóa).



C. Docker Compose

Công cụ hỗ trợ định nghĩa và vận hành các ứng dụng đa container (Multi-container applications).

- Sử dụng tập tin cấu hình `docker-compose.yml`.
- Cho phép khởi tạo toàn bộ hệ thống (Ứng dụng + Cơ sở dữ liệu) chỉ với một câu lệnh.

2.3 Hướng Dẫn Cài Đặt và Vận Hành

Phần này sẽ hướng dẫn cài đặt và chạy thử một container đơn giản.

2.3.1 Cài đặt Docker

- Tải về tại: `https://docs.docker.com/engine/install/`

2.3.2 Tạo ứng dụng HTML đơn giản

Tạo file `index.html`:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Docker Web App</title>
</head>
<body>
    <h1>Hello, Docker!</h1>
</body>
</html>
```

2.3.3 Tạo Dockerfile

Tạo file tên là `Dockerfile` (không có đuôi file):

```
FROM nginx:alpine
COPY index.html /usr/share/nginx/html
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

2.3.4 Build Image

```
docker build -t docker-web-app:latest .
```

2.3.5 Run Container

```
docker run -p 8080:80 docker-web-app:latest
```

Truy cập `http://localhost:8080` để xem kết quả.

3 Cấu Hình Cơ Sở Dữ Liệu trên Neon.tech

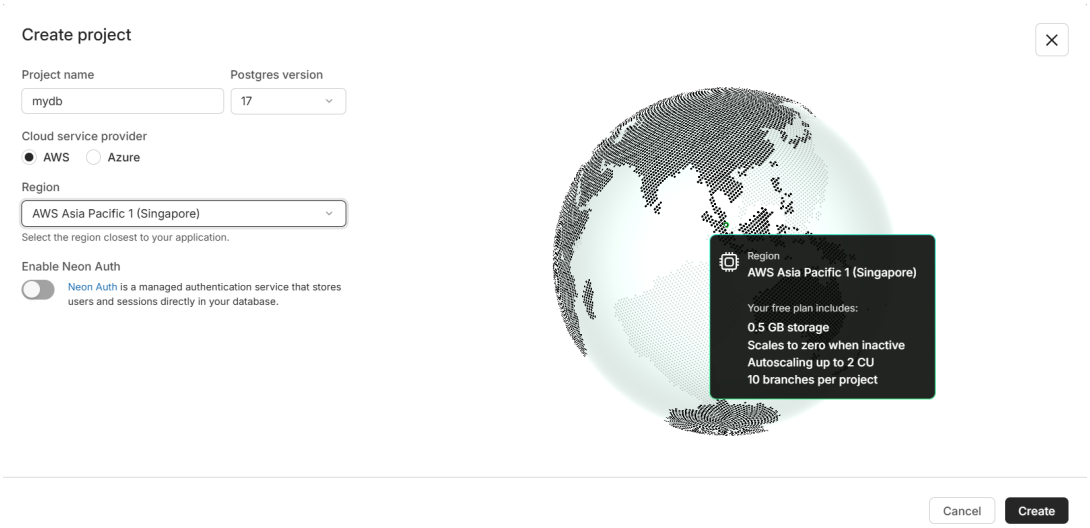
Neon.tech là một nền tảng Database Serverless hiện đại dành cho PostgreSQL. Neon tách biệt lớp lưu trữ (storage) và tính toán (compute), cho phép tự động mở rộng (autoscaling) và tạo nhánh database (branching) nhanh chóng.

Đặc biệt, Neon cung cấp gói **Free Tier** hào phóng, rất phù hợp để sinh viên sử dụng cho mục đích học tập và kiểm thử mà không cần lo lắng về chi phí.

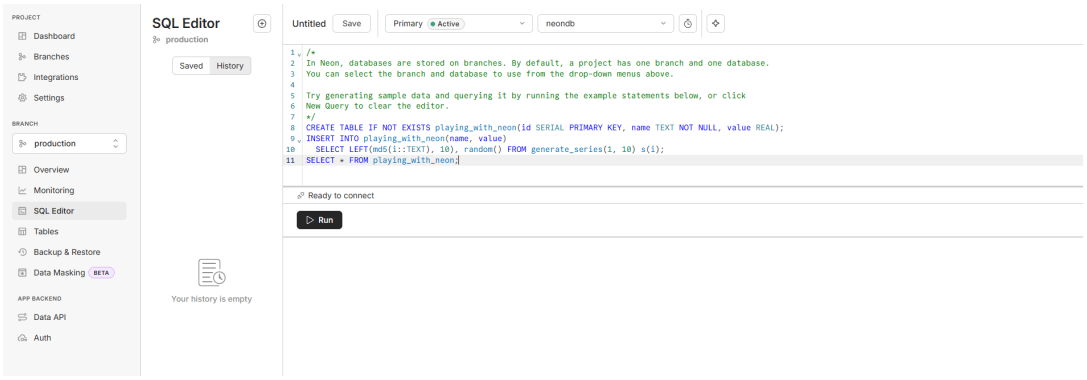
Lưu ý: Sinh viên có thể tùy chọn sử dụng các nhà cung cấp PostgreSQL khác (như Render Postgres, ElephantSQL, hoặc tự host) miễn là đảm bảo ứng dụng kết nối và hoạt động bình thường.

3.1 Khởi Tạo Project

1. Truy cập `https://neon.com/` và đăng ký/đăng nhập.
2. Vào **Projects** → chọn **New Project**.
3. Neon sẽ tự động tạo sẵn một Database PostgreSQL.

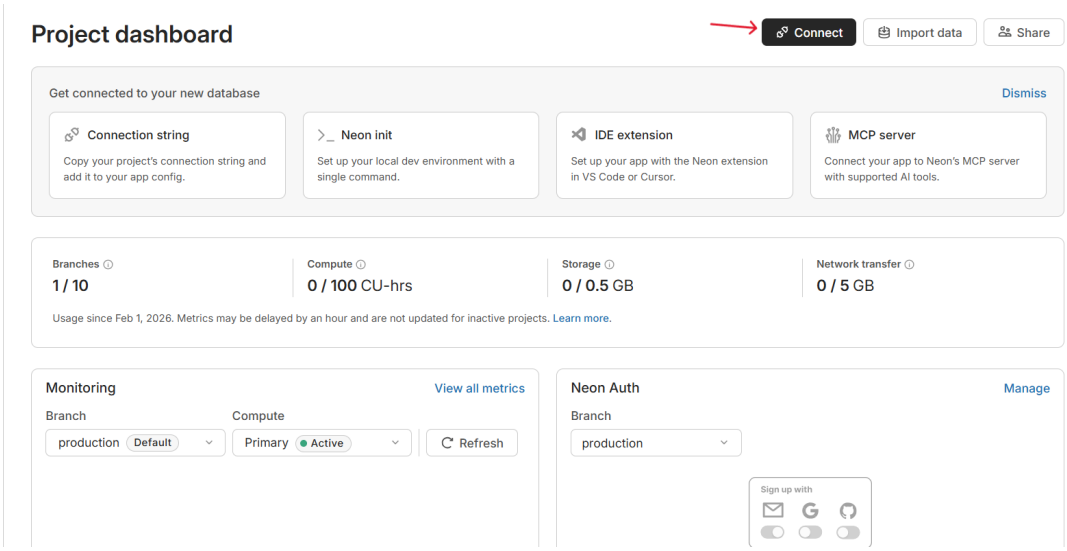


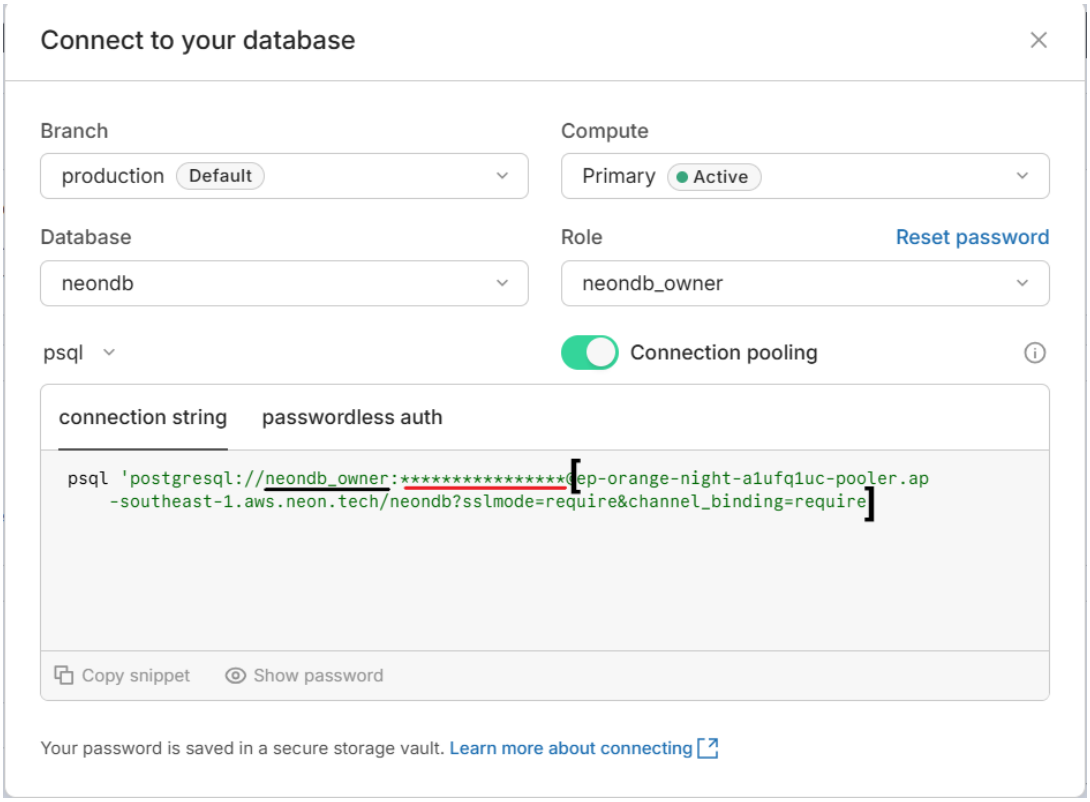
SQL Editor: Bạn có thể dùng công cụ SQL Editor ngay trên web để tạo bảng, truy vấn dữ liệu test.



3.2 Lấy Chuỗi Kết Nối (Connection String)

1. Tại trang Dashboard, chọn Database vừa tạo và nhấn nút **Connect**.
2. Sao chép chuỗi kết nối (Connection String). Định dạng chuẩn như sau: `postgresql://username:password@host/neondb?sslmode=require`
3. *Lưu ý quan trọng: Đối với ứng dụng Spring Boot, chuỗi kết nối cần có tiền tố* `jdbc:`*. Ví dụ:* `jdbc:postgresql://...`





Lưu ý: Chuỗi kết nối này sẽ được dùng để cấu hình biến môi trường `DATABASE_URL` cho Backend.

4 Triển Khai Web Service lên Render.com

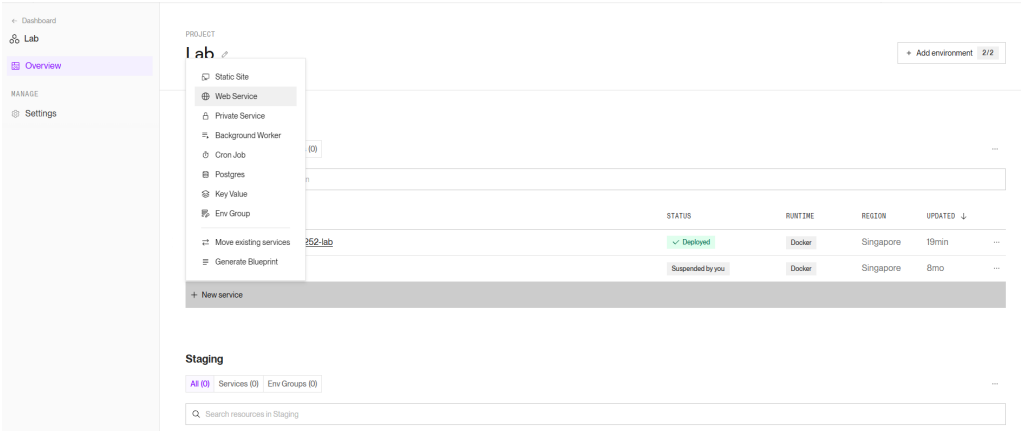
Render là một nền tảng Cloud hiện đại (Unified Cloud), giúp các nhà phát triển xây dựng, triển khai và vận hành ứng dụng một cách dễ dàng và nhanh chóng. Render hỗ trợ mạnh mẽ cho Docker, giúp việc deploy ứng dụng trở nên đơn giản chỉ với vài bước cấu hình.

Tương tự như Neon, Render cung cấp gói **Free Tier** cho Web Service, rất thuận tiện cho việc học tập và triển khai các dự án cá nhân.

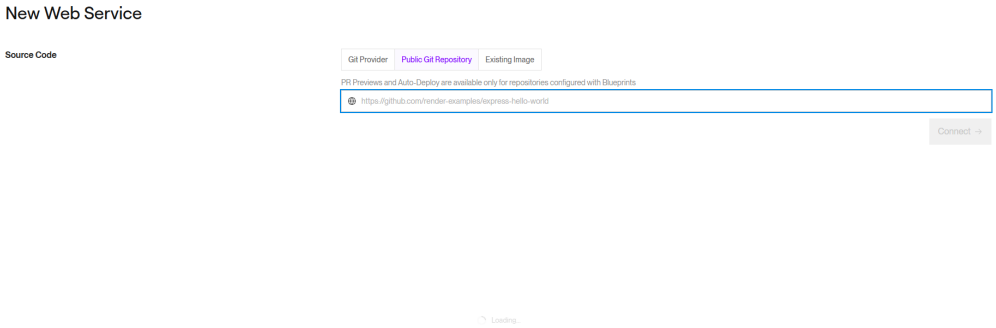
Lưu ý: Sinh viên có thể tùy chọn sử dụng các nền tảng PaaS khác (như Heroku, Railway, Fly.io,...) để triển khai ứng dụng, miễn là kết quả cuối cùng là ứng dụng được public trên Internet.

4.1 Kết Nối Render với GitHub

1. Truy cập `https://render.com/` và tạo tài khoản.
2. Tại Dashboard, chọn **New** + → **Web Service**.



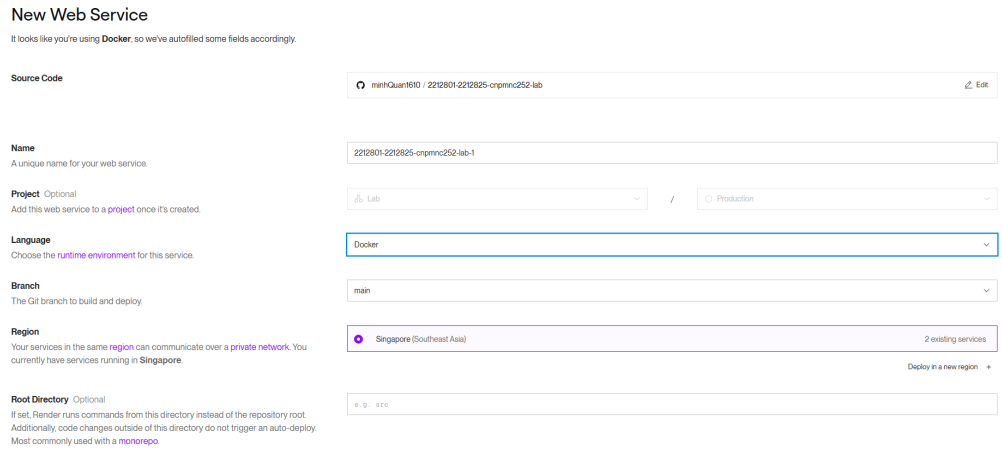
3. Chọn **Build and deploy from a Git repository**.



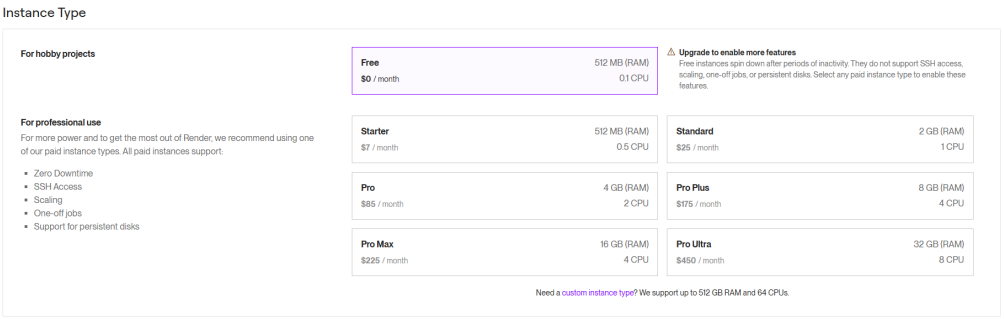
4. Kết nối với tài khoản GitHub và chọn Repository chứa mã nguồn project (Student Management).

4.2 Cấu Hình Service

1. **Name**: Đặt tên cho service (ví dụ: `student-management-api`).
2. **Branch**: Chọn nhánh `main` (hoặc nhánh bạn đang code).
3. **Runtime**: Chọn `Docker`. Render sẽ tự động tìm `Dockerfile` trong repo để build.



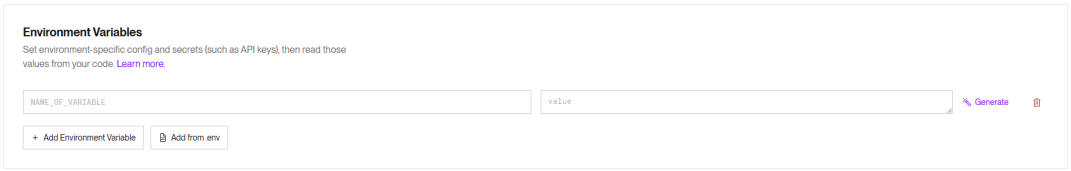
4. **Instance Type**: Chọn **Free**.



4.3 Cấu Hình Biến Môi Trường (Environment Variables)

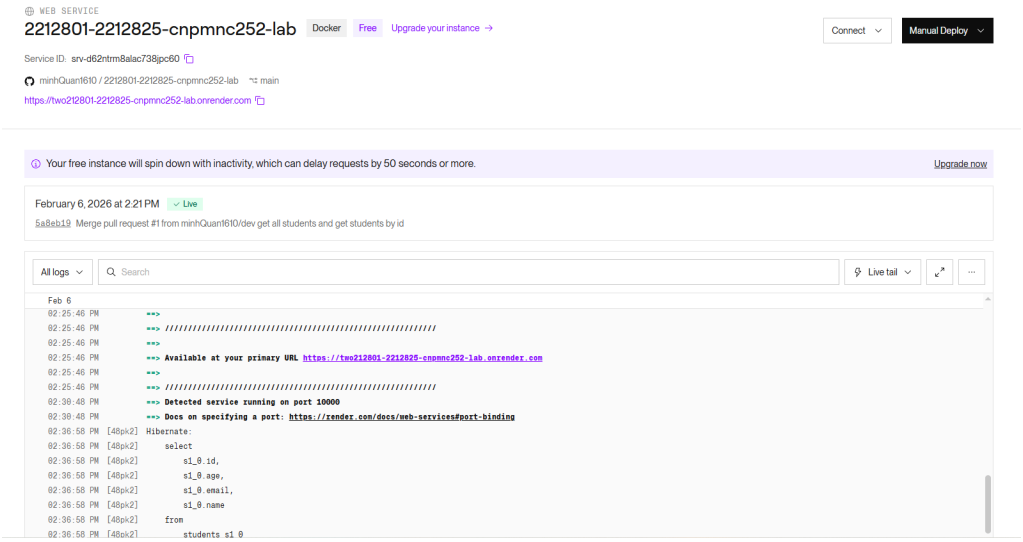
Đây là bước quan trọng để ứng dụng kết nối được với Database trên Neon. Kéo xuống phần **Environment Variables** và thêm các biến sau:

- `DATABASE_URL`: Dán chuỗi kết nối đã copy từ Neon. **Bắt buộc** thêm tiền tố `jdbc:` vào đầu chuỗi (ví dụ: `jdbc:postgresql://...`) để ứng dụng kết nối thành công.
- *Lưu ý: Nếu code của bạn tách riêng username/password, hãy cấu hình thêm các biến tương ứng như `DB_USERNAME`, `DB_PASSWORD`.*



4.4 Deploy & Kiểm Tra

1. Nhấn nút **Create Web Service**.
2. Render sẽ bắt đầu quá trình Build và Deploy. Theo dõi tiến trình trong tab **Logs**.
3. Khi quá trình hoàn tất (Logs hiển thị `Your service is live`), truy cập vào đường dẫn do Render cung cấp để kiểm tra hoạt động của API.



4.5 Thiết lập Tự Động Triển Khai (CI/CD)

Mặc định, Render đã bật chế độ **Auto Deploy**. Mỗi khi bạn push code mới lên branch `main`, Render sẽ tự động build và deploy phiên bản mới nhất.

5 Yêu Cầu Bài Tập

Nội dung thực hiện:

- Rà soát và hoàn thiện các chức năng CRUD của ứng dụng Quản lý Sinh viên (kết quả của Lab 4).
- Thực hiện quy trình triển khai ứng dụng lên **Render** (Web Service) và kết nối với **Neon** (Database).
- Đảm bảo ứng dụng hoạt động ổn định và có thể truy cập được thông qua mạng Internet.

Yêu cầu nộp bài:

- Nộp đường dẫn công khai (**Public URL**) của trang web đã triển khai thành công.

Phụ Lục

A. Dockerfile Tham Khảo (Spring Boot)

File `Dockerfile` đặt tại thư mục gốc của project:

```
# Stage 1: Build
FROM maven:3.9.4-eclipse-temurin-21 AS build
WORKDIR /app
COPY pom.xml .
COPY src ./src
RUN mvn clean package -DskipTests

# Stage 2: Run
FROM eclipse-temurin:21-jre
WORKDIR /app
COPY --from=build /app/target/*.jar app.jar
EXPOSE 8080
ENTRYPOINT ["java", "-jar", "app.jar"]
```

B. Mẫu application.properties cho Lab Deployment

Cấu hình để nhận biến môi trường từ Render:

```
spring.application.name=student-management
server.port=${PORT:8080}

# Database (PostgreSQL)
# Lay gia tri tu bien moi truong DATABASE_URL, neu khong co thi dung localhost (cho dev)
spring.datasource.url=${DATABASE_URL:jdbc:postgresql://localhost:5432/student-management}
spring.datasource.username=${DB_USERNAME:postgres}
spring.datasource.password=${DB_PASSWORD:password}
spring.datasource.driver-class-name=org.postgresql.Driver

# JPA/Hibernate
spring.jpa.hibernate.ddl-auto=update
```

```
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect
```

Cảm ơn các bạn ***Từ Văn Nguyễn Anh Quân*** *và* ***Nguyễn Minh Quân*** *đã hỗ trợ hoàn thiện tài liệu này.*