

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ TẠI THÀNH PHỐ HỒ CHÍ MINH**



**LẬP TÀI TIỆU TRIỂN KHAI PHẦN MỀM**

**Họ và tên: Nguyễn Quốc Tuấn**

**MSSV: N19DCCN179**

**GVHD: Lê Hà Thanh**

**Hồ Chí Minh – Tháng 6, năm 2024**

1. Giới thiệu
   1. Mục đích:

Tài liệu này nhằm cung cấp hướng dẫn chi tiết và rõ ràng về cách triển khai website nghe nhạc. Website này sử dụng ReactJS để phát triển phần giao diện người dùng (frontend) và NodeJS cho phần máy chủ và xử lý logic nghiệp vụ (backend). Mục tiêu của tài liệu này là giúp các kỹ sư phát triển và quản trị viên hệ thống hiểu rõ quy trình triển khai, cấu hình và vận hành hệ thống một cách hiệu quả và chính xác.

* 1. Phạm vi:

Tài liệu này sẽ đề cập đến các bước cần thiết để triển khai toàn bộ hệ thống, bao gồm:

1. Cài đặt môi trường phát triển và triển khai.
2. Cấu hình hệ thống để đảm bảo hoạt động ổn định và an toàn.
3. Chạy và kiểm thử hệ thống để xác minh tính đúng đắn và hiệu quả.
4. Quản lý và bảo trì hệ thống sau khi triển khai.

Các phần chính được đề cập trong tài liệu bao gồm:

* Yêu cầu hệ thống: Phần cứng và phần mềm cần thiết.
* Cấu trúc thư mục của cả frontend và backend.
* Hướng dẫn cài đặt môi trường phát triển.
* Hướng dẫn cấu hình các tệp môi trường.
* Các bước xây dựng và khởi động ứng dụng.
* Quy trình kiểm thử để đảm bảo chất lượng phần mềm.
* Các bước triển khai chi tiết cho cả frontend và backend.
* Hướng dẫn bảo trì và cập nhật hệ thống.

Với việc làm theo các hướng dẫn trong tài liệu này, các bên liên quan sẽ có thể triển khai và vận hành hệ thống nghe nhạc một cách suôn sẻ, đảm bảo cung cấp trải nghiệm tốt nhất cho người dùng cuối.

1. Tìm hiểu Forward Proxy và Reverse Proxy

### Forward Proxy và Reverse Proxy

**1. Forward Proxy:**

**Khái niệm:** Forward Proxy là một máy chủ trung gian đứng giữa máy khách (client) và máy chủ đích (target server). Khi máy khách gửi yêu cầu đến máy chủ đích, yêu cầu này được chuyển qua forward proxy. Proxy này sẽ nhận yêu cầu từ máy khách, sau đó gửi yêu cầu đó tới máy chủ đích và chuyển tiếp phản hồi từ máy chủ đích trở lại máy khách.

**Công dụng:**

* **Ẩn danh và bảo mật:** Giúp ẩn địa chỉ IP thực của máy khách, bảo vệ danh tính và tránh theo dõi.
* **Kiểm soát truy cập:** Giúp quản lý và kiểm soát truy cập vào internet, hạn chế truy cập đến các trang web không mong muốn hoặc nguy hiểm.
* **Caching:** Lưu trữ các nội dung thường truy cập để giảm tải và tăng tốc độ truy cập.
* **Bypass restrictions:** Giúp vượt qua các hạn chế địa lý hoặc chặn IP của các trang web.

**2. Reverse Proxy:**

**Khái niệm:** Reverse Proxy là một máy chủ trung gian đứng giữa máy chủ (server) và các máy khách (clients). Khi máy khách gửi yêu cầu tới máy chủ, yêu cầu này được chuyển qua reverse proxy. Proxy này sẽ nhận yêu cầu từ máy khách và sau đó chuyển tiếp yêu cầu đó tới máy chủ đích. Phản hồi từ máy chủ đích sẽ được reverse proxy gửi lại cho máy khách.

**Công dụng:**

* **Load Balancing:** Phân phối lưu lượng truy cập đến nhiều máy chủ khác nhau để giảm tải và tăng hiệu suất hệ thống.
* **Caching:** Lưu trữ tạm thời các nội dung để giảm tải cho máy chủ chính và tăng tốc độ phản hồi.
* **Bảo mật:** Ẩn cấu trúc mạng nội bộ và máy chủ đích, bảo vệ máy chủ khỏi các cuộc tấn công trực tiếp.
* **SSL Termination:** Giải mã các kết nối SSL để giảm tải công việc mã hóa/giải mã cho máy chủ đích.
* **DDoS Protection:** Bảo vệ máy chủ khỏi các cuộc tấn công từ chối dịch vụ phân tán (DDoS).

### So sánh giữa Forward Proxy và Reverse Proxy:

* **Vị trí triển khai:**
  + Forward Proxy: Được triển khai phía trước máy khách.
  + Reverse Proxy: Được triển khai phía trước máy chủ.
* **Đối tượng phục vụ:**
  + Forward Proxy: Chủ yếu phục vụ các máy khách trong việc truy cập ra ngoài internet.
  + Reverse Proxy: Chủ yếu phục vụ các máy chủ trong việc quản lý và điều phối lưu lượng từ các máy khách.
* **Chức năng chính:**
  + Forward Proxy: Bảo mật, kiểm soát truy cập, caching cho máy khách.
  + Reverse Proxy: Load balancing, caching, bảo mật, SSL termination, DDoS protection cho máy chủ.

Như vậy, dù cả hai loại proxy đều là máy chủ trung gian, nhưng chúng phục vụ các mục đích khác nhau tùy theo vị trí và vai trò trong mạng.

1. Yêu cầu hệ thống

2.1 Phần cứng

 **Máy chủ cần thiết:**

* Một máy chủ để chạy backend và một máy chủ để phục vụ frontend hoặc sử dụng dịch vụ đám mây như AWS, Azure, hoặc Google Cloud.

 **Bộ vi xử lý (CPU):**

* Tối thiểu: 4 nhân (4 vCPU)
* Khuyến nghị: 8 nhân (8 vCPU) hoặc cao hơn, tùy thuộc vào lượng người dùng dự kiến và khối lượng công việc.

 **Bộ nhớ (RAM):**

* Tối thiểu: 8GB
* Khuyến nghị: 16GB hoặc cao hơn để đảm bảo hiệu suất tốt cho cả backend và frontend.

 **Ổ cứng (Storage):**

* Tối thiểu: 100GB SSD
* Khuyến nghị: 250GB SSD hoặc cao hơn, với khả năng mở rộng để lưu trữ nhạc và dữ liệu người dùng.

2.2 Phần mềm

* **Hệ điều hành:**
  + Tối thiểu: Ubuntu 18.04 LTS
  + Khuyến nghị: Ubuntu 20.04 LTS hoặc phiên bản mới hơn.
* **Phiên bản NodeJS:**
  + Tối thiểu: Node.js 18.x
  + Khuyến nghị: Node.js 22.x hoặc phiên bản mới hơn để tận dụng các tính năng và cải tiến hiệu suất.
* **Phiên bản NPM/Yarn:**
  + NPM: >= 10.x
* **Phiên bản ReactJS:**
  + Tối thiểu: React 17.x
  + Khuyến nghị: React 18.x hoặc phiên bản mới hơn để đảm bảo khả năng tương thích và hiệu suất tối ưu.
* **Các công cụ khác:**
  + **Git:** Để quản lý mã nguồn
  + **CI/CD tools:** Như Jenkins, GitHub Actions, hoặc GitLab CI/CD để tự động hóa quy trình triển khai

Ví dụ về cấu hình máy chủ:

**Máy chủ Backend:**

* **CPU:** 8 vCPU
* **RAM:** 16GB
* **Storage:** 250GB SSD
* **Hệ điều hành:** Ubuntu 20.04 LTS

**Máy chủ Frontend:**

* **CPU:** 4 vCPU
* **RAM:** 8GB
* **Storage:** 100GB SSD
* **Hệ điều hành:** Ubuntu 20.04 LTS

Nếu sử dụng dịch vụ đám mây, bạn có thể lựa chọn các cấu hình tương tự từ các nhà cung cấp như AWS (EC2 instances), Azure (Virtual Machines), hoặc Google Cloud (Compute Engine).

1. Cấu trúc thư mục

**Backend (NodeJS):**

* src/: Chứa mã nguồn chính của ứng dụng
* config/: Các file cấu hình
* controllers/: Các controller xử lý logic nghiệp vụ
* models/: Các model của cơ sở dữ liệu
* routes/: Các định tuyến (API routes)
* middlewares/: Các middleware
* utils/: Các tiện ích
* tests/: Các bài kiểm thử

**Frontend (ReactJS):**

* src/: Chứa mã nguồn chính của ứng dụng
* components/: Các thành phần React
* pages/: Các trang
* services/: Các dịch vụ API
* utils/: Các tiện ích
* tests/: Các bài kiểm thử

1. Thiết lập Server và môi trường:

Tham số thiết lập Server:

Phiên bản Ubuntu: 20.6.4

Server frontend:

Địa chỉ IP: 192.168.1.120

Tên server: frontend server

Network Adapter: Bridged

Server gitlab:

Địa chỉ IP: 192.168.1.100

Tên server: gitlab server

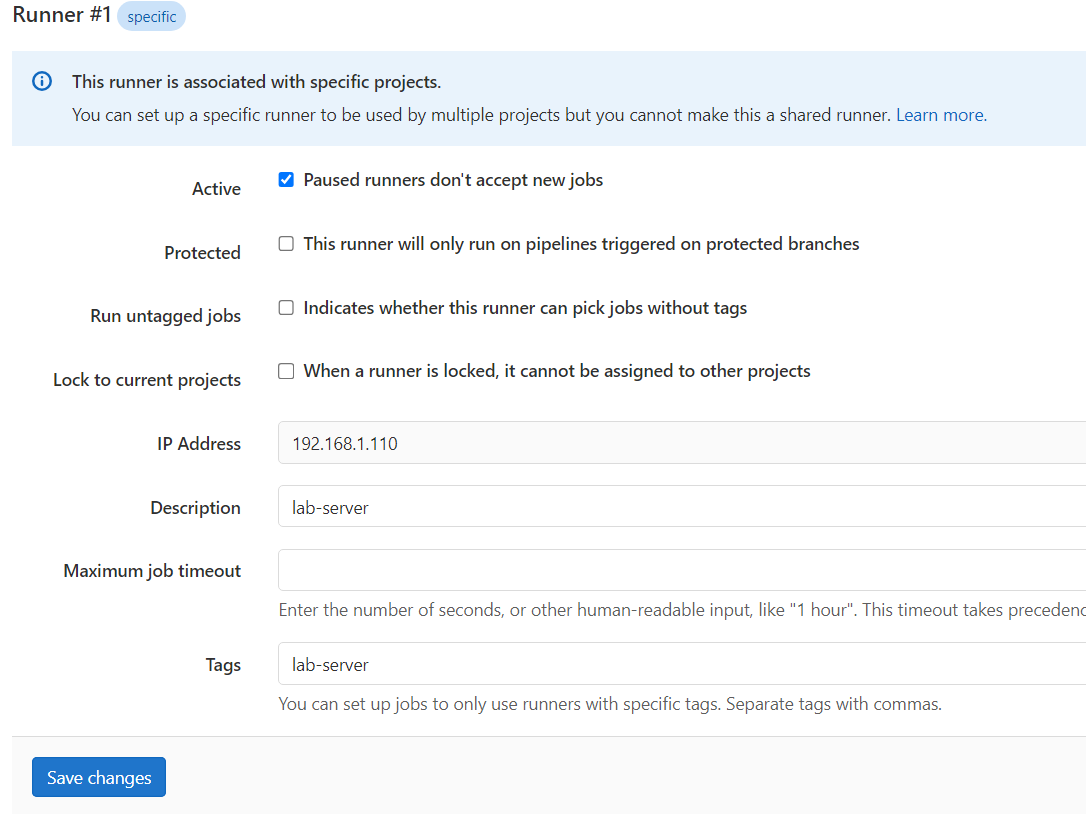
Network Adapter: Bridged

Server backend:

Địa chỉ IP: 192.168.1.110

Tên server: lab server

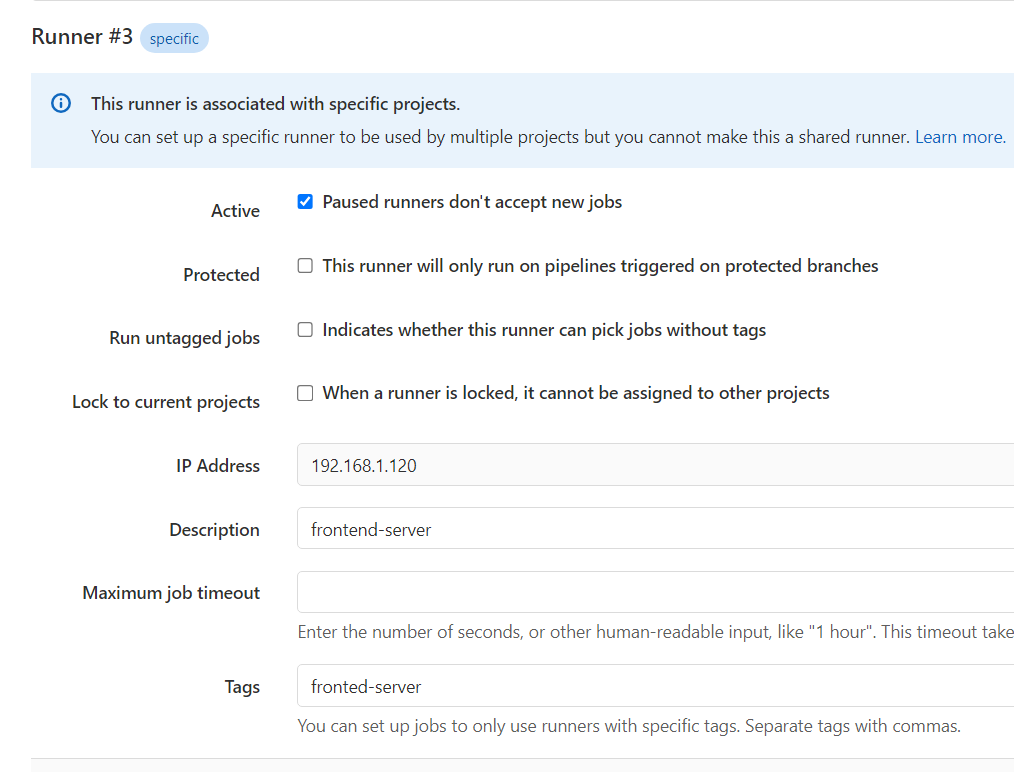
Network Adapter: Bridged

+script và thông số thiết lập CI/CD  
  
Thiết lập Runner cho backend:   


Cấu hình file .gitlab-ci.yml cho backend (Nodejs):



Thiết lập Runner cho frontend:



Cấu hình file .gitlab-ci.yml cho frontend:  
  
+ liên kết tới repo chứa code sites, tài liệu, scripts:  
https://github.com/QuocTuanIT87/Chuy-n-CNPM\_Final-Report.git

1. Cài đặt môi trường phát triển

**Cài đặt NodeJS và NPM:**

# Cài đặt NodeJS và NPM: sudo apt-get install -y nodejs npm

Clone repository: git clone <https://github.com/QuocTuanIT87/MusicWebPlayer.git>

Cài đặt các dependencies:

Backend: cd backend npm install

Frontend: cd frontend npm install

1. Cấu hình môi trường

Tạo file .env cho backend:

PORT=5000

DB\_HOST=<database-host>

DB\_USER=<database-user>

DB\_PASS=<database-password>

JWT\_SECRET=<jwt-secret>

Tạo file .env cho frontend:

REACT\_APP\_API\_URL=<backend-url>

1. Triển khai

6.1 Build ứng dụng:

Chạy Backend:

npm start

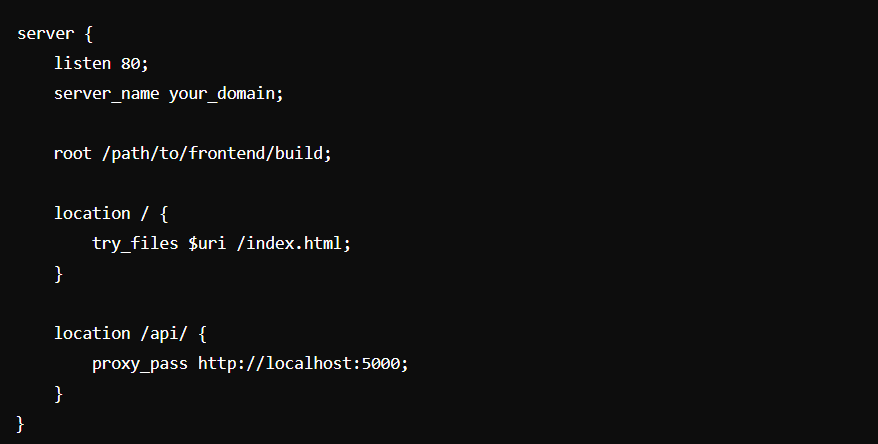
Chạy Frontend:

npm run build

Npm start

6.2 Triển khai lên một web server (Nginx, Apache, hoặc một dịch vụ như Vercel, Netlify).

Cấu hình Nginx để phục vụ ứng dụng React:



1. Giám sát và bảo trì

8.1. Cập nhật phần mềm

Cập nhật định kỳ:

Định kỳ cập nhật các gói phần mềm và thư viện để đảm bảo rằng bạn luôn sử dụng các phiên bản mới nhất với các bản vá bảo mật và cải tiến hiệu suất.

Quản lý phiên bản:

Sử dụng công cụ quản lý phiên bản như Git để theo dõi và quản lý các thay đổi mã nguồn.

Áp dụng quy trình CI/CD để tự động hóa việc kiểm tra và triển khai các thay đổi.

8.2. Sao lưu dữ liệu

Sao lưu định kỳ:

Thiết lập lịch trình sao lưu định kỳ cho cơ sở dữ liệu và các dữ liệu quan trọng khác.

Sử dụng các công cụ sao lưu như AWS Backup, Google Cloud Storage, hoặc các giải pháp sao lưu tự động khác.

Kiểm tra sao lưu:

Thường xuyên kiểm tra các bản sao lưu để đảm bảo rằng chúng hoạt động đúng và có thể khôi phục được khi cần.

1. Phụ lục