

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**Seminar các vấn đề hiện đại của CNPM**

**Đề tài: Phát triển hệ thống quản lý học sinh với kiến trúc đa môi trường, tích hợp Jenkins CI/CD**

Giảng viên hướng dẫn: Đinh Nguyễn Anh Dũng

Lớp: SE400.P11.PMCL

Sinh viên thực hiện:

- Vũ Đức Minh - 21522348
- Nguyễn Thiện - 21521461

**TP. Hồ Chí Minh, Ngày 7 Tháng 10 Năm 2024**

# 1. Tổng quan

## 1.1. Giới thiệu

Trong bối cảnh phát triển phần mềm hiện nay, CI/CD đóng vai trò quan trọng trong việc tự động hóa các quy trình phát triển và triển khai. Đề tài này sẽ tập trung vào việc xây dựng một CI/CD pipeline hoàn chỉnh sử dụng Jenkins cho một ứng dụng demo đơn giản, gồm backend Spring Boot và frontend React.

## 1.2. Mục tiêu và lợi ích

Xây dựng pipeline CI/CD đầy đủ cho một ứng dụng Spring Boot (backend) và React (frontend).

Sử dụng Jenkins để tự động hóa quy trình build, kiểm thử và triển khai.

Triển khai ứng dụng lên các môi trường khác nhau (staging, production).

Đảm bảo CI/CD tuân thủ các tiêu chuẩn enterprise.

## 1.3. Nội dung

Ứng dụng demo có thể là một **ứng dụng quản lý học sinh** (Student Management), với các tính năng cơ bản:

- **Frontend:** React để quản lý giao diện người dùng, quản lý học sinh (tạo, cập nhật, xem danh sách học sinh).
- **Backend:** Spring Boot cung cấp các API RESTful cho quản lý học sinh.
- **Database:** Sử dụng PostgreSQL để lưu trữ thông tin học sinh.

**Các tính năng cơ bản:**

- CRUD (Create, Read, Update, Delete) cho học sinh.
- UI Feedback: Thông báo cho người dùng về hành động thành công hoặc thất bại.
- Xử lý lỗi và ghi log backend.

## 1.4. CI/CD Pipeline chi tiết cho từng môi trường

Môi trường **Development (Dev)**

**Pipeline chi tiết:**

**Kiểm tra chất lượng mã nguồn:**

- Chạy các công cụ kiểm tra mã như **ESLint** (cho React) và **Checkstyle** (cho Java/Spring Boot).
- Đảm bảo mã không có lỗi cú pháp và tuân thủ coding standard.

#### **Build ứng dụng:**

- React: Sử dụng **npm** hoặc **yarn** để build ứng dụng React.
- Spring Boot: Sử dụng **Maven** hoặc **Gradle** để build backend Spring Boot.

#### **Kiểm thử đơn vị (Unit Tests):**

- Chạy các bài kiểm thử đơn vị (các function/method riêng lẻ) cho cả React và Spring Boot.

#### **Triển khai lên môi trường Development:**

- Sau khi các bước trên thành công, ứng dụng được tự động triển khai lên server Development bằng **Docker Compose**.

#### **Thông báo:**

- Gửi thông báo qua email hoặc Slack cho nhóm phát triển để biết rằng quá trình đã thành công hay thất bại.

### Môi trường Staging

#### **Pipeline chi tiết:**

- **Trigger:** Mỗi khi có merge vào branch staging.

#### **Build ứng dụng với cấu hình Production:**

- Sử dụng các biến môi trường và cấu hình Production (ví dụ, sử dụng API keys cho production).

#### **Kiểm thử smoke (Smoke Tests):**

- Chạy kiểm thử nhanh để đảm bảo các chức năng chính hoạt động bình thường (chẳng hạn, trang chủ của ứng dụng phải load được).

#### **Triển khai lên Production:**

- Triển khai ứng dụng lên môi trường Production bằng **Helm**.
- Có thể sử dụng chiến lược **Canary Deployment** hoặc **Blue-Green Deployment** để giảm thiểu rủi ro.

### **Giám sát và cảnh báo:**

- Kết nối với các hệ thống giám sát như **Prometheus** và **Grafana** để theo dõi hiệu suất và tình trạng của hệ thống.

### **Thông báo:**

- Gửi thông báo đến các bên liên quan về việc triển khai thành công hoặc thất bại.

### Môi trường Production

#### **Mục tiêu:**

- Đảm bảo các thay đổi đã được kiểm tra kỹ lưỡng, có thể triển khai an toàn mà không làm gián đoạn hệ thống.

#### **Pipeline chi tiết:**

- **Trigger:** Mỗi khi có merge vào branch master hoặc main.

#### **Build ứng dụng với cấu hình Production:**

- Sử dụng các biến môi trường và cấu hình Production (ví dụ, sử dụng API keys cho production).

#### **Kiểm thử smoke (Smoke Tests):**

- Chạy kiểm thử nhanh để đảm bảo các chức năng chính hoạt động bình thường (chẳng hạn, trang chủ của ứng dụng phải load được).

#### **Triển khai lên Production:**

- Triển khai ứng dụng lên môi trường Production bằng Kubernetes hoặc Helm.
- Có thể sử dụng chiến lược Canary Deployment hoặc Blue-Green Deployment để giảm thiểu rủi ro.

#### **Giám sát và cảnh báo:**

- Kết nối với các hệ thống giám sát như Prometheus và Grafana để theo dõi hiệu suất và tình trạng của hệ thống.

#### **Thông báo:**

- Gửi thông báo đến các bên liên quan về việc triển khai thành công hoặc thất bại.

## 1.5. Mô hình quản lý code (Git Flow)

**Main:** Chứa code sẵn sàng triển khai.

**Develop:** Nhánh phát triển chính, chứa code đang phát triển và thử nghiệm.

**Feature branches:** Cho các tính năng mới hoặc sửa lỗi.

**Release branches:** Chuẩn bị code để triển khai.

**Hotfix branches:** Sửa lỗi khẩn cấp trong production.

## 2. Kế hoạch

Giai đoạn	Thời gian	Công việc	Kết quả
<b>Tìm hiểu lý thuyết</b>	1 tuần	- Tìm hiểu về CI/CD pipeline, Jenkins, Docker,	Nắm rõ quy trình CI/CD và các công cụ cần thiết.
<b>Thiết kế hệ thống</b>	2 tuần	- Thiết kế cơ sở dữ liệu - Thiết kế kiến trúc hệ thống - Wireframes và mockups UI/UX	- Sơ đồ cơ sở dữ liệu - Thiết kế kiến trúc hệ thống chi tiết - Wireframes và mockups giao diện người dùng
<b>Thiết lập môi trường và CI/CD</b>	2 tuần	- Cấu hình Jenkins cho CI/CD pipeline - Thiết lập các môi trường (Dev, Feature, Test, Staging, Production)	- Jenkins CI/CD pipeline hoạt động ổn định - Môi trường phát triển và thử nghiệm sẵn sàng
<b>Phát triển Backend (API)</b>	1 tuần	- Xây dựng API sản phẩm - Xử lý người dùng và đơn hàng	- API hoàn chỉnh cho quản lý học sinh - Test coverage cho backend $\geq 80\%$
<b>Phát triển Frontend</b>	1 tuần	- Xây dựng giao diện - Tích hợp API backend	- Giao diện người dùng hoạt động mượt mà - Kết nối thành công với backend qua API

<b>Kiểm thử và tối ưu hóa</b>	1 – 2 tuần	<ul style="list-style-type: none"> <li>- Unit test, integration test và end-to-end test</li> <li>- Load testing và tối ưu hiệu suất</li> <li>- Tối ưu bảo mật</li> </ul>	<ul style="list-style-type: none"> <li>- Ứng dụng không có lỗi lớn</li> <li>- Hiệu suất đạt yêu cầu với số lượng người dùng dự kiến</li> <li>- Tăng cường bảo mật</li> </ul>
<b>Triển khai Pre-production</b>	2 - 3 tuần	<ul style="list-style-type: none"> <li>- Triển khai lên môi trường staging</li> <li>- Thực hiện thử nghiệm người dùng(UAT)</li> <li>- Sửa lỗi phát sinh</li> </ul>	<ul style="list-style-type: none"> <li>- Môi trường staging ổn định và sẵn sàng cho triển khai sản xuất</li> <li>- Phản hồi từ UAT được xử lý</li> </ul>
<b>Triển khai Production</b>	1 - 2 tuần	<ul style="list-style-type: none"> <li>- Triển khai lên môi trường production bằng chiến lược Blue-Green hoặc Canary</li> </ul>	<ul style="list-style-type: none"> <li>- Ứng dụng thương mại điện tử chạy ổn định trên production</li> </ul>
<b>Làm hồ sơ nghiệm thu</b>	1 tuần	<ul style="list-style-type: none"> <li>- Chuẩn bị các tài liệu liên quan.</li> <li>- Làm hồ sơ nghiệm thu.</li> </ul>	Bản báo cáo nghiệm thu.

### 3. Tài liệu tham khảo

[1] <https://docs.docker.com/>

[2] <https://www.jenkins.io/>

[3] <https://viblo.asia/p/ci-cd-va-devops-07LKXYXDZV4>