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
- Nguyên 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 và Gitlab 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) sử dụng Gitlab.
Đảm bảo CI/CD tuân thủ các tiêu chuẩn enterprise.

1.3. Nội dung

Ứng dụng demo là một **ứng dụng quản lý học sinh** (Student Management), với các tính năng cơ bản như CRUD (Create, Read, Update, Delete) cho học sinh, UI Feedback, xử lý lỗi và ghi log backend.

1.4. So sánh tổng quan về các framework

Tiêu chí	Spring Boot (Java)	**Node.js (Express)**	**Django (Python)**	Go (Gin/Gorilla)	ASP.NET Core (C#)
Hiệu năng	Cao, ổn định với JVM	Tốt cho I/O, hạn chế về tính toán nặng	Tốt cho I/O, không mạnh về CPU	Hiệu năng cao, ngôn ngữ biên dịch	Hiệu năng cao với .NET tối ưu hóa
Dễ sử dụng	Phức tạp hơn, cần hiểu về Spring	Dễ học, ít cấu hình	Dễ học, Python đơn giản	Hơi phức tạp hơn với Goroutines	Dễ cho người quen .NET
Mở rộng	Rất tốt cho hệ thống lớn, doanh nghiệp	Tốt với microservices nhỏ	Tốt cho ứng dụng vừa và nhỏ	Rất tốt, phù hợp hệ thống lớn	Mở rộng tốt, tích hợp dịch vụ Microsoft

Lý do sử dụng Spring Boot:

1. **Tính mở rộng và ổn định**: Spring Boot dễ mở rộng và phù hợp cho các hệ thống doanh nghiệp.
2. **Tích hợp dễ dàng**: Hỗ trợ nhiều công cụ và dịch vụ phổ biến như Jenkin, Docker và các hệ thống cơ sở dữ liệu.
3. **Cộng đồng lớn và tài liệu phong phú**: Nhiều tài liệu và ví dụ thực tiễn giúp quá trình học tập và phát triển dễ dàng hơn.
4. **Tích hợp bảo mật**: Hỗ trợ mạnh mẽ các giải pháp bảo mật như OAuth2, JWT, giúp xây dựng hệ thống bảo mật hiệu quả.

1.5. 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**.
- 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:

- **T**riển 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.

1.6. 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
Tìm hiểu lý thuyết	1 tuần	- Tìm hiểu về CI/CD pipeline, Jenkins, Docker,	Nắm rõ quy trình CI/CD và các công cụ cần thiết.
Thiết kế hệ thống	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
Thiết lập môi trường và CI/CD	1 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
Phát triển Backend (API)	2 - 3 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 >= 80%
Phát triển Frontend	2 - 3 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

Kiểm thử và tối ưu hóa	2 tuần	- Unit test, integration test và end-to-end test - Load testing và tối ưu hiệu suất - Tối ưu bảo mật	- Ứng dụng không có lỗi lớn - Hiệu suất đạt yêu cầu với số lượng người dùng dự kiến - Tăng cường bảo mật
Triển khai Pre-production	2 tuần	- Triển khai lên môi trường staging - Thực hiện thử nghiệm người dung(UAT) - Sửa lỗi phát sinh	- Môi trường staging ổn định và sẵn sàng cho triển khai sản xuất - Phản hồi từ UAT được xử lý
Triển khai Production	1 - 2 tuần	- Triển khai lên môi trường production bằng chiến lược Blue-Green hoặc Canary	- Ứng dụng thương mại điện tử chạy ổn định trên production
Làm hồ sơ nghiệm thu	1 tuần	- Chuẩn bị các tài liệu liên quan. - Làm hồ sơ nghiệm thu.	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