**TÀI LIỆU DEV-OPS CHO WEBSITE DỊCH VỤ OCR**

**1. Giới thiệu**

* **Mục tiêu:** Tạo tài liệu hướng dẫn DevOps cho website dịch vụ OCR, bao gồm phát triển, triển khai, và bảo trì.
* **Phạm vi:** Áp dụng cho toàn bộ quy trình từ phát triển, tích hợp, kiểm thử, triển khai đến giám sát và bảo mật hệ thống.

**2. Kiến trúc hệ thống**

* **Kiến trúc tổng quan:**
  + Website bao gồm frontend (ReactJS) và backend (Expressjs).
  + API kết nối với cơ sở dữ liệu (MongoDB) để quản lý thông tin về người dùng, tài liệu, kết quả, phản hồi người dùng, thanh toán

**Công nghệ sử dụng:**

* **Frontend:**
  + ReactJS: Để xây dựng giao diện người dùng.
  + Redux: Quản lý trạng thái của ứng dụng.
  + React Router: Điều hướng giữa các trang.
  + Axios: Thực hiện các yêu cầu HTTP từ frontend đến backend.
* **Backend:**
  + ExpressJS: Xây dựng API backend.
  + Node.js: Chạy ứng dụng backend.
  + JWT (JSON Web Token): Xác thực và quản lý phiên người dùng.
  + Mongoose: Thư viện kết nối và tương tác với MongoDB.
* **Cơ sở dữ liệu:**
  + MongoDB: Cơ sở dữ liệu NoSQL để lưu trữ dữ liệu người dùng, tài liệu, và các kết quả OCR.
  + MongoDB Atlas: Sử dụng MongoDB trên nền tảng đám mây để quản lý cơ sở dữ liệu.
* **DevOps:**
  + Docker: Đóng gói ứng dụng trong các container.
  + Kubernetes: Quản lý và triển khai các container trong môi trường sản xuất.
  + Jenkins/GitLab CI: Tích hợp và triển khai liên tục (CI/CD).
  + Nginx: Reverse proxy và load balancer.
  + Prometheus/Grafana: Giám sát hệ thống và hiển thị các số liệu.
* **Security:**
  + SSL/TLS: Bảo mật giao tiếp giữa frontend và backend.
  + OWASP: Thực hiện các quy tắc bảo mật dựa trên các khuyến nghị của OWASP.
* **Logging:**
  + Winston: Thư viện logging cho Node.js.
  + ELK Stack (Elasticsearch, Logstash, Kibana): Hệ thống quản lý và phân tích log.
* **Test:**
  + Postman: Kiểm thử API thủ công.

**3. Quản lý mã nguồn (Source Control Management)**

* **Công cụ sử dụng**: Git
* **Kho lưu trữ mã nguồn**: Sử dụng GitHub hoặc GitLab để lưu trữ mã nguồn của dự án.
* **Chiến lược nhánh**:
  + main branch: Chứa mã nguồn ổn định, đã qua kiểm thử và sẵn sàng triển khai.
  + develop branch: Nơi hợp nhất các tính năng mới, sẽ được kiểm thử trước khi đưa vào main.
  + feature branches: Tạo riêng cho mỗi tính năng hoặc sửa lỗi. Sau khi hoàn thành, sẽ được merge vào develop.
  + **Pull Requests (PR)**: Mỗi thay đổi được gửi lên từ các branch tính năng phải thông qua PR, có ít nhất một người khác trong nhóm xem xét và phê duyệt trước khi merge.

**4. Tích hợp liên tục (Continuous Integration - CI)**

* **Công cụ sử dụng**: Jenkins hoặc GitLab CI/CD
* **Pipeline CI**:
  + **Xây dựng (Build)**: Mỗi khi có commit mới lên nhánh develop, hệ thống sẽ tự động kiểm tra xem mã có thể build được không.
  + **Kiểm thử (Test)**: Chạy các bộ kiểm thử tự động (unit test, integration test) để đảm bảo mã không bị lỗi.
  + **Phản hồi (Feedback)**: Nếu build hoặc test không thành công, gửi thông báo cho nhóm phát triển để xử lý ngay.

**5. Triển khai liên tục (Continuous Deployment/Delivery - CD)**

* **Môi trường triển khai**:
  + **Development Environment**: Triển khai tự động trên môi trường phát triển sau khi pass CI.
  + **Staging Environment**: Môi trường kiểm thử trước khi triển khai thực tế, mô phỏng gần như chính xác môi trường production.
  + **Production Environment**: Môi trường triển khai thực tế, phục vụ người dùng cuối.
* **Quy trình triển khai**:
  + **Automatic Deployment**: Sử dụng Jenkins hoặc GitLab CI/CD để tự động triển khai lên môi trường phát triển và staging.
  + **Manual Approval for Production**: Triển khai lên môi trường production có thể yêu cầu sự phê duyệt thủ công.

**6. Quản lý cơ sở hạ tầng (Infrastructure Management)**

* **Công cụ sử dụng**: Docker, Kubernetes
* **Môi trường container**:
  + **Docker**: Tất cả các dịch vụ (frontend, backend, database) được đóng gói thành các container Docker.
  + **Docker Compose**: Dùng để cấu hình và khởi động các container trong môi trường phát triển.
* **Quản lý container**:
  + **Kubernetes**: Sử dụng Kubernetes để quản lý và điều phối các container trong môi trường production.
  + **Pod Management**: Mỗi dịch vụ (frontend, backend) chạy trong các Pod Kubernetes riêng biệt, có thể mở rộng linh hoạt.
  + **Service Discovery and Load Balancing**: Kubernetes sẽ quản lý việc khám phá dịch vụ và cân bằng tải giữa các Pod.

**7. Giám sát và Logging (Monitoring and Logging)**

* **Công cụ giám sát**: Prometheus, Grafana
  + **Prometheus**: Thu thập và lưu trữ dữ liệu giám sát từ các container và dịch vụ.
  + **Grafana**: Hiển thị các dashboard để theo dõi hiệu suất hệ thống và dịch vụ.
* **Công cụ logging**: ELK Stack (Elasticsearch, Logstash, Kibana)
  + **Elasticsearch**: Lưu trữ và quản lý log.
  + **Logstash**: Thu thập, phân tích, và gửi log đến Elasticsearch.
  + **Kibana**: Tạo các dashboard để phân tích và giám sát log.

**8. Bảo mật (Security)**

* **Quản lý bảo mật mã nguồn**:
  + **Sử dụng Secret Management**: Quản lý các thông tin nhạy cảm (API keys, database credentials) bằng cách sử dụng công cụ như Vault hoặc Kubernetes Secrets.
  + **Phân quyền truy cập (Access Control)**: Thiết lập các quyền truy cập phù hợp cho các thành viên trong nhóm, đảm bảo chỉ những người được phép mới có thể truy cập và chỉnh sửa mã nguồn.
* **Bảo mật hệ thống**:
  + **SSL/TLS**: Sử dụng chứng chỉ SSL/TLS để mã hóa giao tiếp giữa người dùng và server.
  + **Firewall và Network Policies**: Cấu hình firewall và các chính sách mạng để bảo vệ hệ thống khỏi các cuộc tấn công từ bên ngoài.
* **Kiểm thử bảo mật (Security Testing)**:
  + **Static Code Analysis**: Kiểm tra mã nguồn để phát hiện các lỗ hổng bảo mật tiềm tàng.
  + **Penetration Testing**: Thực hiện các bài kiểm tra xâm nhập để đảm bảo hệ thống an toàn trước các cuộc tấn công.

**9. Quy trình Backup và Khôi phục (Backup and Recovery)**

* **Backup dữ liệu**:
  + **MongoDB Backup**: Thiết lập quy trình sao lưu định kỳ cho cơ sở dữ liệu MongoDB, lưu trữ backup trên các hệ thống lưu trữ an toàn.
* **Khôi phục dữ liệu**:
  + **Disaster Recovery Plan**: Xây dựng kế hoạch khôi phục dữ liệu và hệ thống trong trường hợp xảy ra sự cố nghiêm trọng.

**10. Tài liệu hướng dẫn (Documentation)**

* **Tài liệu triển khai**: Cung cấp hướng dẫn chi tiết về cách triển khai hệ thống từ môi trường phát triển đến production.
* **Tài liệu sử dụng**: Hướng dẫn người dùng cuối về cách sử dụng các tính năng của dịch vụ OCR.
* **Tài liệu bảo trì**: Cung cấp hướng dẫn cho đội ngũ DevOps về cách bảo trì và cập nhật hệ thống sau khi triển khai.