21521997 – Lê Văn Duy

<u>Đề tài</u>: Triển khai CI/CD pipeline toàn diện với Jenkins, Docker cho dự án Golang + React

1. Giới thiệu

Trong thời đại công nghệ hiện nay, việc tự động hóa quy trình phát triển phần mềm thông qua các phương pháp CI/CD (Continuous Integration/Continuous Deployment) là vô cùng quan trọng. Đề tài này tập trung vào việc triển khai CI/CD pipeline toàn diện cho một ứng dụng web bao gồm Backend (Golang) và Frontend (React), sử dụng Jenkins làm công cụ tự động hóa và Docker để đảm bảo môi trường triển khai nhất quán.

2. Mục tiêu

- Xây dựng một pipeline CI/CD hoàn chỉnh từ giai đoạn viết mã, kiểm thử (testing), đến triển khai (deployment).
- Sử dụng Jenkins để tự động hóa các bước trong pipeline.
- Tận dụng Docker để container hóa các dịch vụ Golang và React nhằm tạo môi trường triển khai nhất quán.
- Đưa ra các kịch bản CI/CD khác nhau để học tập và nghiên cứu, đồng thời hỗ trợ cho đồ án môn học.

3. Nội dung

- Đề tài sẽ triển khai một hệ thống CI/CD pipeline cho ứng dụng Golang + React với các thành phần chính là:
  - Jenkins: Điều khiển toàn bộ pipeline tự động từ giai đoạn kiểm thử đến triển khai.
  - Docker: Đảm bảo rằng các dịch vụ Frontend và Backend luôn hoạt động nhất quán trong môi trường container hóa.
- Mục tiêu của đề tài là học tập, nghiên cứu, và thực hành quy trình triển khai phần mềm CI/CD. Điều này giúp tiết kiệm thời gian, giảm thiểu lỗi thủ công và cải thiện chất lượng mã nguồn.
- Ứng dụng sẽ được container hóa, từ đó Jenkins sẽ tự động build, test, và deploy ứng dụng trong các môi trường Docker. Quy trình sẽ gồm:
  - Continuous Integration: Jenkins sẽ tự động kích hoạt mỗi khi có thay đổi trong codebase, chạy unit test, và kiểm tra chất lượng mã.
  - Continuous Deployment: Nếu các bài kiểm tra thành công, Jenkins sẽ tiếp tục build Docker image và triển khai nó.

4. Xây dựng ứng dụng demo

- Chi tiết dự án demo:

- Frontend: Sử dụng ReactJS (có thể kết hợp với TailwindCSS) để tạo giao diện người dùng cho ứng dụng TodoList.

- Backend: API được viết bằng Golang, cung cấp các endpoint cho việc tạo, đọc, cập nhật, và xóa (CRUD) các mục Todo.
- Database: Sử dụng MySQL để lưu trữ các mục TodoItem

- Các tính năng cơ bản:

- CRUD (Create, Read, Update, Delete): Tính năng cho Todolist.
- UI Feedback: Sử dụng Toast hoặc Snackbar để hiển thị thông báo cho người dùng về các hành động thành công/thất bại.
- Logging và Error Handling: Golang sẽ ghi log chi tiết các hành động và lỗi xảy ra.

- Sử dụng mô hình Git Flow để quản lý source code:

- Main: Chứa code đã qua kiểm tra và sẵn sàng triển khai.
- Develop: Làm việc chính, chứa code tính năng mới, được kiểm thử liên tục.
- Feature branches: Tạo cho từng tính năng hoặc fix bug riêng biệt.
- Release branches: Chuẩn bị cho việc triển khai.
- Hotfix branches: Fix bug nhanh chóng trong môi trường production.

- Các kịch bản CI/CD Chi Tiết:

a) **Kịch bản 1: Kiểm thử và tích hợp code (Continuous Integration)**

Trigger: Khi có Pull Request từ bất kỳ nhánh feature nào vào nhánh develop.

- Linting và kiểm tra mã nguồn: Jenkins chạy các bước kiểm tra code theo các tiêu chuẩn (ví dụ: ESLint cho React, gofmt cho Golang).
- Unit Test: Chạy unit test cho cả frontend và backend.
    - Frontend: Jest hoặc React Testing Library.
    - Backend: Golang test.
- Build Docker image:
    - Build một Docker image cho cả backend và frontend, bao gồm kiểm thử tính tương thích giữa các dịch vụ.
    - Docker Compose: Jenkins sẽ sử dụng Docker Compose để tạo một môi trường thử nghiệm, chạy các container frontend, backend, và database.
- Integration Test: Chạy các bài kiểm tra tích hợp giữa frontend và backend.

b) **Kịch bản 2: Triển khai lên môi trường staging (Continuous Deployment)**

Trigger: Khi code được merge vào nhánh develop.

- Build Docker image: Jenkins sẽ build lại Docker image cho frontend và backend.
- Push Docker image lên Docker Hub: Các image sẽ được đẩy lên một registry (Docker Hub hoặc private registry).

- Triển khai tự động lên staging: Sử dụng Docker Compose để deploy toàn bộ hệ thống lên môi trường staging trên máy chủ staging.
- Smoke Test: Chạy một loạt smoke test để kiểm tra nhanh các chức năng chính của ứng dụng có hoạt động không.
- Notification: Gửi thông báo qua Slack hoặc email về kết quả triển khai (thành công hoặc thất bại).

c) Kịch bản 3: Triển khai lên production

Trigger: Khi code được merge từ nhánh release vào nhánh main.

- Backup database: Trước khi triển khai, Jenkins sẽ thực hiện sao lưu database.
- Build và push Docker image: Build lại Docker image từ nhánh main và đẩy lên Docker Hub.
- Triển khai lên production: Triển khai các container trên môi trường production.
- Rollback (trong trường hợp lỗi): Nếu phát hiện lỗi sau khi deploy, Jenkins có thể rollback và khởi động lại phiên bản cũ của Docker image từ registry.

d) Kịch bản 4: Xử lý khi ứng dụng gặp sự cố (Failure Handling)

Scenario: Ứng dụng bị lỗi hoặc gặp sự cố không mong muốn sau khi triển khai.

- Health Check: Jenkins thiết lập các health check endpoint cho cả frontend và backend. Nếu một trong các dịch vụ không hoạt động, Jenkins sẽ kích hoạt rollback tự động.
- Logs và Alerting: Sử dụng Jenkins kết hợp với Docker logs để ghi lại thông tin chi tiết về lỗi xảy ra.
- Rollback: Jenkins sẽ tự động kích hoạt lại Docker container của phiên bản trước đó trong trường hợp phát hiện lỗi nghiêm trọng.
- Notification: Jenkins gửi thông báo qua email hoặc Slack với chi tiết lỗi và hành động rollback.

- Một số tình huống sẽ lên kế hoạch để mô phỏng sau:

- Khi một Pull Request không vượt qua kiểm thử: Jenkins sẽ gửi thông báo và yêu cầu sửa code.
- Khi ứng dụng gặp sự cố trên production: Jenkins sẽ tự động rollback và khởi động lại phiên bản ổn định.
- Khi cần triển khai phiên bản mới mà không làm gián đoạn dịch vụ: Docker Swarm hoặc Docker Compose có thể được sử dụng để đảm bảo zero-downtime deployment (triển khai không gián đoạn).

5. Tài liệu tham khảo

https://viblo.asia/p/ci-cd-va-devops-07LKXYXDZV4

https://docs.docker.com/

https://www.jenkins.io/

What is CI/CD? - GeeksforGeeks

The Go Programming Language

How to Set Up Jenkins with Docker for CI/CD Pipelines: A Step-by-Step Guide - DEV Community

Building an End-to-End CI/CD Pipeline with Jenkins, Ansible, Docker, Kubernetes, and Terraform | by Manikandan prakash | Medium

Documentation (zabbix.com)