**Đồ án 2: Nghiên cứu và Triển khai CI/CD trên Nhiều Công Cụ**

**1. Đặt vấn đề**

Trong các doanh nghiệp, các **developer** chịu trách nhiệm viết code cho dự án, còn các **operator** hoặc **system engineer** đảm nhận việc lấy code và triển khai. Khi chỉ có một hoặc hai source code, việc triển khai chưa quá phức tạp. Tuy nhiên, thực tế một doanh nghiệp có thể có hàng chục, thậm chí hàng trăm source code. Chỉ riêng với 20 dự án trên Git, đã đặt ra câu hỏi: cần bao nhiêu **operator** và **system engineer** để triển khai kịp tiến độ?

Quá trình phát triển dự án thường diễn ra trên ba môi trường chính:

* **Development:** môi trường để dev viết và kiểm tra code hàng ngày.
* **Staging:** môi trường kiểm thử, đảm bảo mọi thứ chạy ổn trước khi lên production.
* **Production:** môi trường phục vụ người dùng cuối.

Trong môi trường development, dev commit code hàng giờ, thậm chí hàng phút, và họ cần kiểm tra code trên server trước khi chuyển sang các môi trường quan trọng hơn. Nếu phải chờ system triển khai thủ công, điều này sẽ rất tốn thời gian và ảnh hưởng đến hiệu suất dự án.

Không chỉ dev gặp khó khăn, đội ngũ system cũng gặp nhiều trở ngại khi triển khai thủ công: copy source code giữa các server có thể dẫn đến sai sót; họ phải truy cập từng server, clone code, build, run, rồi kiểm tra logs… Ngoài ra, triển khai dự án còn bao gồm nhiều công việc khác như cài đặt hệ thống, giám sát, bảo trì và nâng cấp, làm quá trình triển khai trở nên phức tạp và tốn thời gian.

**2. Giới thiệu CI/CD**

Để giải quyết các khó khăn trên, **CI/CD** ra đời. CI/CD gồm hai phần chính:

**CI (Continuous Integration – Tích hợp liên tục)**

Bước CI bao gồm: clone code, build dự án, và tích hợp các bước kiểm thử như:

* Kiểm tra hiệu năng (Performance Test)
* Kiểm tra chất lượng code (Clean Code Test)
* Kiểm tra bảo mật (Security Test)
* Và nhiều loại kiểm thử khác

Mục tiêu là đảm bảo chất lượng source code trước khi triển khai.

**CD (Continuous Deployment / Continuous Delivery – Triển khai liên tục)**

* **Continuous Deployment:** triển khai hoàn toàn tự động. Dev chỉ cần commit code, chức năng mới sẽ được triển khai ngay mà không cần thao tác thủ công.
* **Continuous Delivery:** triển khai vẫn yêu cầu bước xác nhận thủ công. Code sau khi build và kiểm thử chỉ được triển khai khi có sự đồng ý từ team.

Mỗi chiến lược CD có ưu nhược điểm riêng:

* Triển khai tự động tối ưu hiệu suất và rút ngắn thời gian ra môi trường thực tế.
* Triển khai thủ công giúp tăng khả năng kiểm soát, giảm rủi ro lỗi khi đưa code vào môi trường quan trọng.

Tùy theo nhu cầu doanh nghiệp, có thể kết hợp cả hai để xây dựng quy trình **CI/CD tối ưu**, vừa nhanh, vừa an toàn.

**3. Các công cụ cần tìm hiểu và công việc thực hiện**

Hiện nay có nhiều công cụ hỗ trợ xây dựng CI/CD, được áp dụng trong doanh nghiệp từ quy mô nhỏ đến lớn, bao gồm:

* **GitHub Actions**
* **GitLab CI/CD**
* **AWS CI/CD**
* **Jenkins**

Để mô phỏng một quy trình CI/CD hoàn chỉnh, cần kết hợp thêm các công cụ hỗ trợ hạ tầng:

* **VMware Workstation:** tạo máy ảo phục vụ môi trường triển khai.
* **Docker:** triển khai và quản lý container.
* **Docker Registry** và các công cụ khác để lưu trữ image.
* **Gitlab Runner** công cụ để chạy CI/CD trên Gitlab CI/CD

**Công việc thực hiện:**

1. Tìm hiểu cách thiết lập môi trường và cấu hình từng loại công cụ trên máy ảo.
2. Xây dựng quy trình CI/CD phù hợp với từng công cụ, bao gồm:
   * Thiết lập pipeline CI (clone, build, test).
   * Thiết lập pipeline CD (deploy tự động hoặc deploy có xác nhận).
   * Kiểm thử quy trình CI/CD hoàn chỉnh trên các môi trường Development, Staging và Production.
3. So sánh ưu nhược điểm, đánh giá hiệu quả từng công cụ.

**4. Kết quả đạt được**

Sau khi hoàn thành đồ án, các kết quả em sẽ đạt được:

* Hiểu rõ cơ chế và cách triển khai **CI/CD** trên nhiều công cụ phổ biến.
* Xây dựng các pipeline CI/CD hoàn chỉnh trên các công cụ khác nhau.
* Mô phỏng quy trình triển khai dự án thực tế trên các môi trường Development, Staging và Production.
* So sánh và đánh giá hiệu quả của các công cụ CI/CD để lựa chọn giải pháp phù hợp cho doanh nghiệp.

**5. Timeline học tập**

|  |  |
| --- | --- |
| **Tuần đồ án** | **Các công việc cần thực hiện** |
| 1-7 | - Tìm hiểu và cài đặt các công cụ cần thiết.  - Xây dựng quy trình CI/CD trên GitLab CI/CD và GitHub Actions. |
| 8-12 | - Tìm hiểu công cụ Jenkins.  - Xây dựng quy trình CI/CD trên Jenkins. |
| 13-16 | - Tìm hiểu về môi trường Cloud (AWS).  - Xây dựng quy trình CI/CD và triển khai ứng dụng trên AWS. |

**6. Đề xuất xây dựng báo cáo đồ án**

Chương 1: Giới thiệu chung

* 1. Đặt vấn đề
  2. Giới thiệu về hệ thống CI/CD

1.2.1 Khái niệm CI/CD

1.2.2 CI/CD pipeline

1.2.3 Các nguyên tắc trong CI/CD

1.2.3 Lợi ích của CI/CD

Chương 2: Các công cụ được sử dụng trong hệ thống

2.1 GitHub Actions

2.2 Gitlab CI/CD

2.3 Jenkins

2.4 AWS CI/CD

2.5 Các công cụ khác

Chương 3: Xây dựng hệ thống CICD trên nhiều công cụ

3.1 Cấu hình hệ thống CI/CD

3.2 Kiến trúc hệ thống CI/CD

3.3 Cài đặt hệ thống CI/CD

3.3.1 Cài đặt hệ thống CI/CD trên GitHub Actions

3.3.1 Cài đặt hệ thống CI/CD trên GitLab CI/CD

3.3.1 Cài đặt hệ thống CI/CD trên Jenkins

3.3.1 Cài đặt hệ thống CI/CD trên AWS CI/CD

3.4 Kịch bản vận hành hệ thống CI/CD