CICD

CI/CD là một phương pháp trong lập trình nhằm tự động hóa các quy trình trong phát triển phần mềm và nó thường được hiểu là sự kết hợp của hai thực tiễn sau:

**Integration liên tục (Continuous Integration - CI):** CI là quá trình tự động hóa việc hợp nhất các bản sửa đổi về mã nguồn từ nhiều nhà phát triển vào một kho chung. Thay vì các nhà phát triển làm việc độc lập rồi hợp nhất tất cả các thay đổi sau nhiều tuần hoặc nhiều tháng, CI đảm bảo rằng mã nguồn được cập nhật thường xuyên, có thể là vài lần trong một ngày. Việc này giúp phát hiện và sửa lỗi sớm, giảm thiểu vấn đề khi hợp nhất mã nguồn, và cải thiện chất lượng phần mềm.

**Phân phối liên tục (Continuous Delivery hoặc Continuous Deployment - CD):**

**Continuous Delivery:** Là bước tiếp theo của CI, trong đó mọi thay đổi về mã nguồn sau khi đã được kiểm tra và xác nhận thông qua CI, sẽ được chuẩn bị sẵn sàng để phát hành vào môi trường sản xuất. Mục tiêu là giảm thiểu thời gian từ khi một tính năng, sửa đổi hoặc sửa lỗi được hợp nhất vào kho mã nguồn đến khi nó có sẵn để sử dụng thực tế.

**Continuous Deployment:** Là một bước tiến xa hơn Continuous Delivery. Mọi thay đổi về mã nguồn không chỉ sẵn sàng để phát hành, mà còn được tự động phân phối vào môi trường sản xuất mà không cần sự chấp thuận thủ công. Điều này đảm bảo quá trình phát hành phần mềm diễn ra nhanh chóng và liên tục.

CI/CD là một phần thiết yếu của phát triển phần mềm hiện đại, giúp các tổ chức có thể phản hồi nhanh chóng với các thay đổi trên thị trường và yêu cầu của người dùng, đồng thời duy trì sự ổn định và an toàn của ứng dụng.

Các nguyên tắc của phân phối liên tục (Continuous Delivery) được thiết kế để giúp các nhóm phát triển phần mềm phát hành các sản phẩm phần mềm một cách nhanh chóng, hiệu quả và an toàn. Dưới đây là phân tích chi tiết về từng nguyên tắc:

1. \*\*Repeatable Reliable Process (Quy trình Lặp đi lặp lại và Đáng tin cậy):\*\*

- Nguyên tắc này nhấn mạnh việc tạo ra một quy trình có thể tái sử dụng và đáng tin cậy để phát triển, kiểm thử, và phát hành phần mềm. Mọi lần phát hành đều phải thông qua một loạt các bước được xác định rõ ràng và có thể dự đoán được, giúp đảm bảo tính nhất quán và giảm thiểu rủi ro.

2. \*\*Automate Everything (Tự động hóa Mọi thứ):\*\*

- Tự động hóa mọi khía cạnh của quá trình phát triển, từ tích hợp, kiểm thử, đến phát hành. Điều này giúp giảm thiểu các lỗi do con người gây ra và tăng tốc độ phát hành sản phẩm.

3. \*\*Version Control Everything (Kiểm soát phiên bản Mọi thứ):\*\*

- Mọi thành phần của dự án, từ mã nguồn, cấu hình hệ thống, đến tài liệu, đều cần được quản lý bằng hệ thống kiểm soát phiên bản. Điều này không chỉ giúp theo dõi thay đổi mà còn cho phép quay lại phiên bản trước đó nếu có sự cố.

4. \*\*Bring the Pain Forward (Đưa khó khăn về phía trước):\*\*

- Đối mặt và giải quyết các thách thức càng sớm càng tốt trong quá trình phát triển. Việc xác định và khắc phục sớm các vấn đề giúp tránh được chi phí lớn hơn và công sức nhiều hơn ở các giai đoạn sau.

5. \*\*Build-in Quality (Xây dựng Chất lượng từ bên trong):\*\*

- Chất lượng phải được tích hợp ngay từ giai đoạn đầu của quá trình phát triển, không chỉ là một bước cuối cùng. Điều này bao gồm việc áp dụng các phương pháp kiểm thử tốt, đánh giá code liên tục, và đảm bảo sự tuân thủ các tiêu chuẩn chất lượng.

6. \*\*"Done" Means Released (Hoàn thành nghĩa là đã Phát hành):\*\*

- Một tính năng hoặc sửa đổi được coi là "hoàn thành" chỉ khi nó đã được phát hành thành công vào môi trường sản xuất. Điều này đảm bảo rằng mọi sản phẩm làm ra đều sẵn sàng cho người dùng cuối ngay lập tức.

7. \*\*Everyone is Responsible (Mọi người đều có Trách nhiệm):\*\*

- Trách nhiệm đối với chất lượng sản phẩm và quá trình phát triển không chỉ nằm ở một cá nhân hay nhóm nào, mà là của toàn bộ tổ chức. Điều này thúc đẩy sự hợp tác, chia sẻ kiến thức và hỗ trợ lẫn nhau.

8. \*\*Continuous Improvement (Cải tiến Li

ên tục):\*\*

- Luôn tìm cách cải thiện quy trình phát triển phần mềm. Điều này bao gồm việc học hỏi từ những lỗi lầm, phản hồi từ người dùng, và đổi mới công nghệ để tăng hiệu quả và giảm thiểu sự lặp lại không cần thiết.

Các nguyên tắc này cung cấp một khung sườn để giúp các nhóm phát triển có thể phản hồi nhanh chóng đối với các yêu cầu thay đổi và đảm bảo sản phẩm luôn đạt được chất lượng cao nhất.