**CHƯƠNG 2 – TÍCH HỢP LIÊN TỤC VÀ JENKINS**

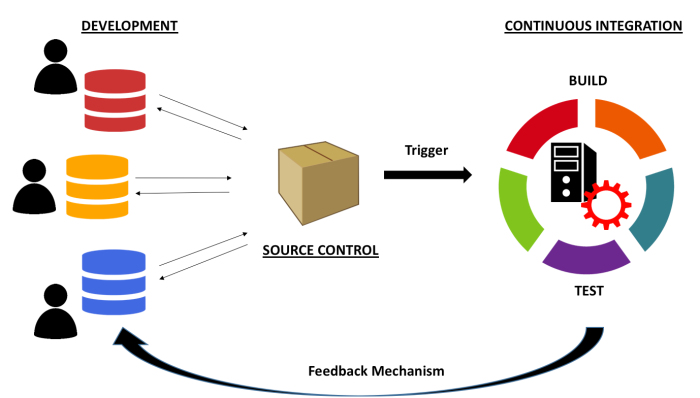
**2.1 TÍCH HỢP LIÊN TỤC**

**2.1.1 Tích hợp liên lục là gì ?**

Trong môi trường tích hợp liên tục , một vấn đề rất phổ biến và thường xuyên gặp phải đối với nhiều nhóm phát triển là một quy trình làm việc phân mảnh. Không hiệu quả trong việc phát triển phần mềm có thể gây ảnh hưởng đến kết quả đầu ra.

Ở môi trường phát triển thông thường , các cá nhân trong nhóm có xu hướng làm việc độc lập . Các kỹ sư phần mềm thường xuyên tạo ra các đoạn mã lớn mà không màng đến việc kiểm soát phiên bản. Sau khi một nhà phát triển , kỹ sư hoàn thành công việc thì họ thêm phần việc của họ vào mã cơ sở. Sau đó, một nhóm khác tự chạy các thử nghiệm để xác minh bản dựng của mã cơ sở . Việc này xảy ra rất nhiều trong môi trường phát triển phần mềm hiện nay và nó trở thành một trong những vấn đề rất nhức nhói và khó chịu trong giai đoạn phát triển phần mềm .

Tích hợp liên tục (Continuous integration - CI) là một thực tiễn phát triển trong đó các nhà phát triển cần phải gửi ( commit ) các thay đổi mã nguồn đến kho lưu trữ chung nhiều lần trong ngày hoặc thường xuyên hơn. Mỗi lần gửi được thực hiện trong kho lưu trữ sau đó được xây dựng ( build ) . Điều này cho phép các nhà phát triển phát hiện sớm các vấn đề. Ngoài ra, tùy thuộc vào công cụ tích hợp liên tục,còn có một số chức năng khác như triển khai ứng dụng xây dựng trên máy chủ thử nghiệm, cung cấp cho các bên liên quan về kết quả xây dựng và thử nghiệm .

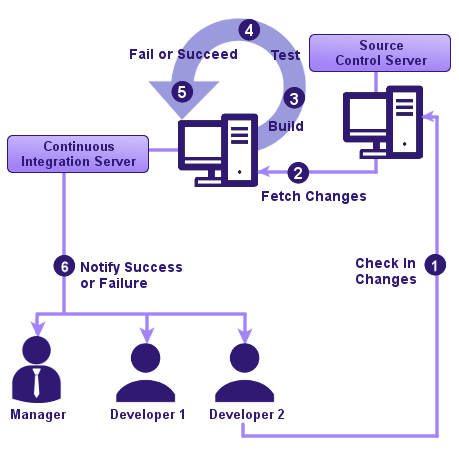


Hình 2.1 Mô hình tích hợp liên tục

Nguồn : dotnetvibes.com

CI nhắm tới mục đích cắt giảm sự thiếu hiệu quả của chu trình phát triển bằng cách cho phép các nhà phát triển biên dịch code của nhóm từ kho lưu trữ kiểm soát phiên bản dùng chung . CI cũng cho phép bạn tự động hóa kiểm tra để bạn có thể thiết lập hệ thống để tự động chạy các thử nghiệm đơn vị hoặc các thử nghiệm tích hợp .

CI tự động theo dõi các lần gửi mà mỗi cá nhân thực hiện. Điều này tinh giản hóa việc xây dựng và xác minh các đoạn code để việc thử nghiệm không thất thoát quá nhiều . CI được chạy trên một máy chủ dùng chung làm tăng khả năng hiển thị giữa các nhà phát triển , vì vậy tất cả các nhà phát triển trong một dự án đều biết về những thay đổi trong mã cơ sở ngày này qua ngày khác. Ngoài ra, họ có thể định cấu hình máy chủ để thông báo cho các nhà phát triển khi họ gửi một mã bị lỗi hoặc hỏng để họ có thể sửa kịp thời các lỗi họ đã gửi lên hệ thống .



Hình 2.2 Các bước của một mô hình tích hợp liên tục

Nguồn : www.thinksys.com

**2.1.2 Đặc điểm của việc tích hợp liên tục**

* Quản lý phiên bản phần mềm , giúp các nhà phát triển kiểm soát các phiên bản đã code đã được gửi lên server .
* Tự động hóa trong việc xây dựng và thử nghiệm code .
* Đội ngũ các nhà phát triển sẽ thường xuyên gửi các đoạn mã của mình đến nơi lưu trữ chính của đội ngũ , thường là một kho lưu trữ được chia sẻ ( shared repository ) .
* Khi code có sự thay đổi thì sẽ được xây dựng lại thông qua máy chủ .
* Khi xây dựng hoặc thử nghiệm có xuất hiện ra lỗi thì server sẽ tự động báo lỗi cho đội ngũ lập trình cũng như nhà quản lý dự án .
* Phát hành phiên bản hoàn chỉnh cho khách hàng khi không còn lỗi nữa .
* Tự động phân phối phiên bản mới lên môi trường phát triển hoặc là tới khách hàng .
* Với cơ chế phản hồi lỗi của tích hợp liên tục , Đội ngũ phát triển cũng như nhà quản lý dự án sẽ nhìn thấy sớm được những khó khăn , lỗi xảy ra trong gian đoạn phát triển phần mềm để có thể nhanh chóng khắc phục để không bị ảnh hưởng trong tương lai .

**2.1.3 Lợi ích và khó khăn của tích hợp liên tục :**

* **Lợi ích :**  
  + Giảm thiểu rủi ro do lỗi được phát hiện sớm trong quá trình phát triển .
  + Giảm thiểu sự lặp lại của các quá trình , tiến trình .
  + Tạo phần mềm có giá trị sử dụng sớm nhất có thể và sẵn sang triển khai mọi lúc mọi nơi .
  + Cung cấp cái nhìn xuyên suốt tổng quan và cụ thể cho từng giai đoạn phát triển .
  + Nâng cao kỹ năng của đội ngũ nhân viên phát triển phần mềm .
  + Cải thiện chất lượng phần mềm .
* **Khó Khăn** 
  + Cần thời gian thiết lập hệ thống ban đầu .
  + Yêu cầu các thành viên , các cá nhân trong đội ngũ dự án phải có các kỹ năng và am hiểu các mô hình như Agile , hệ thống tích hợp CI , các công cụ hỗ trợ cho CI ,..
  + Các chi phí cần thiết để phát triển và duy trì server cho CI .

**2.2 JENKINS**

**2.2.1 Jenkins là gì ?**

Khi nhắc đến CI , những nhà phát triển sẽ nghĩ ngay đến jenkins . Mặc dù có rất nhiều công cụ để hỗ trợ CI như GitLab CI , Travis CI , Bamboo ,… nhưng jenkins vẫn luôn là công cụ được nhắc đến nhiều nhất và được ưu tiên trong việc phát triển , áp dụng CI .

Jenkins là một máy chủ tích hợp liên tục mã nguồn mở có khả năng phối hợp một chuỗi các hành động giúp đạt được quy trình tích hợp liên tục theo một cách cách tự động . Jenkins giúp tự động hóa phần máy móc của quy trình phát triển phần mềm, với sự tích hợp liên tục và tạo điều kiện cho các khía cạnh kỹ thuật của việc phân phối liên tục ( Continuous delivery ) .

Jenkins được sử dụng để xây dựng và thử nghiệm các dự án phần mềm của đội ngũ các nhà phát triển một cách liên tục, giúp các họ dễ dàng tích hợp các thay đổi cho dự án hơn và giúp người dùng dễ dàng có được bản dựng mới. Nó cũng cho phép bạn liên tục cung cấp phần mềm của mình bằng cách tích hợp với một số lượng lớn các công nghệ thử nghiệm và triển khai . Với Jenkins, các tổ chức có thể đẩy nhanh quá trình phát triển phần mềm thông qua tự động hóa. Jenkins tích hợp các quy trình vòng đời phát triển của tất cả các loại, bao gồm xây dựng, tạ o dựng tài liệu, thử nghiệm, đóng gói, giai đoạn, triển khai, phân tích tĩnh và nhiều hơn nữa.

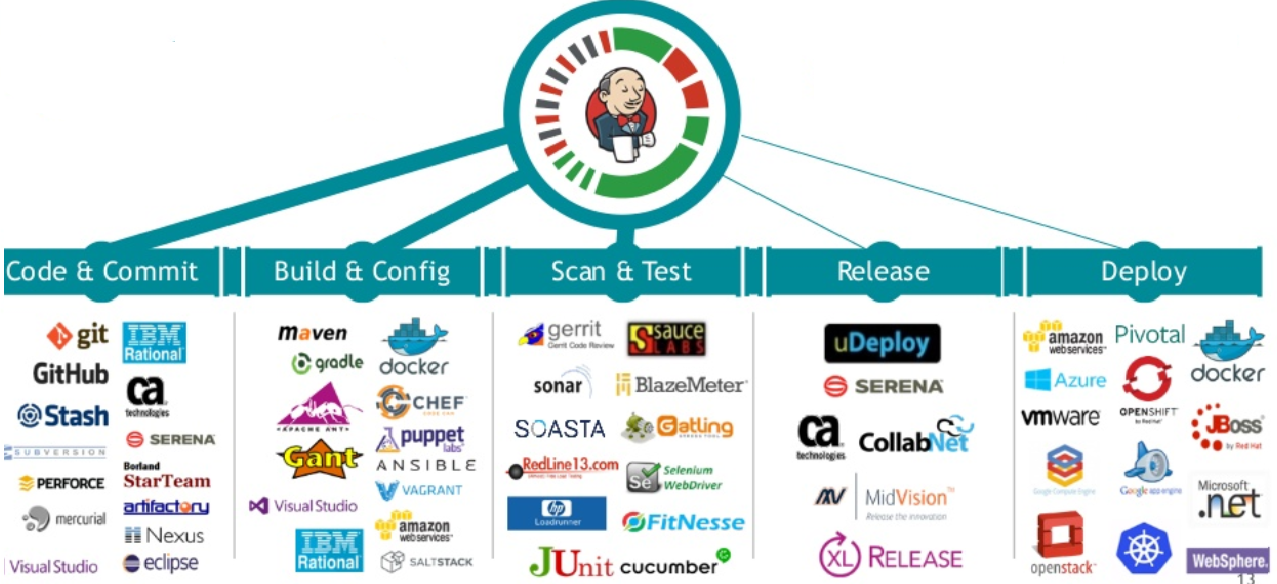
Nó cũng được viết bằng ngôn ngữ Java . Điều này khiến cho nó có thể được tích hợp va áp dụng dễ dàng trên nhiều nền tảng khác nhau như Windows , Linux , macOS . Việc này cũng ảnh hưởng đến số lượng người sử dụng Jenkins , với hơn 300 ngàn lượt cài đặt và đang ngày càng tăng .

Hình 2.3 Build , test và deploy với Jenkins

Nguồn : tech.vccloud.vn

Một trong những điểm nổi bật của Jenkins là sự hỗ trợ plugins . Ở cài đặt mặc định , Jenkins không hỗ trợ nhiều tính năng cho quá trình CI . Nhưng với sự hỗ trợ của plugins , việc này khiến cho Jenkins thành một công cụ mạnh mẽ để có thể tích hợp với bất kì công cụ nào cũng như bất kì nền tảng nào .

Jenkins khác biệt so với các máy chủ CI khác vì sự đa dạng plugins của nó . Các plugins được viết bởi cộng đồng các nhà phát triển , các kỹ sư phần mềm với ý định hướng tới sự đa dạng hóa cho Jenkins . Hiện tại với hơn 1000+ plugins và đang tăng , Jenkins trở thành một công cụ đáng chú ý cho cộng đồng các nhà phát triển phần mềm để áp dụng thực tiễn CI của họ .



Hình 2.4 Một số các công cụ và nền tảng có thể tích hợp với Jenkins

Nguồn : awesome-tech.readthedocs.io

**2.2.2 CI với Jenkins**



Hình 2.5 Lược đồ mô tả CI khi tích hợp với Jenkins

Nguồn www.edureka.co

Dựa vào biểu đồ trên , nó diễn tả những hành động sau :

* Đầu tiên, một nhà phát triển gửi code vào kho lưu trữ mã nguồn. Trong khi đó, máy chủ Jenkins kiểm tra kho lưu trữ theo định kỳ để thay đổi ( hoặc có thể kiểm tra theo event được trigger ) .
* Ngay sau khi một lần gửi xảy ra ở kho, máy chủ Jenkins phát hiện các thay đổi đã xảy ra trong kho lưu trữ mã nguồn. Jenkins sẽ kéo ( pull ) những thay đổi đó và sẽ bắt đầu chuẩn bị một bản dựng mới.
* Nếu việc xây dựng thất bại, thì đội ngũ phát triển sẽ được thông báo.
* Nếu quá trình xây dựng thành công, thì Jenkins sẽ triển khai máy chủ thử nghiệm .
* Sau khi thử nghiệm, Jenkins tạo phản hồi và sau đó thông báo cho các nhà phát triển về kết quả xây dựng và thử nghiệm.
* Nếu việc thử nghiệm thành công , nó sẽ tiếp tục đưa đến máy chủ làm việc .
* Nó sẽ tiếp tục kiểm tra kho lưu trữ mã nguồn để biết các thay đổi được thực hiện trong mã nguồn và toàn bộ quá trình tiếp tục lặp lại.

Việc áp dụng Jenkins khiến cho hiệu quả công việc cũng như quy trình phát triển phần mềm thay đổi rõ rệt , sau đây là bảng so sánh “trước khi áp dụng Jenkins” và “Sau khi áp dụng Jenkins” :

|  |  |
| --- | --- |
| Trước khi áp dụng Jenkins | Sau khi áp dụng Jenkins |
| Các nhà phát triển khi đã hoàn thành xong phần code của họ , họ sẽ gửi cùng một lúc và việc xây dựng và thử nghiệm sẽ diễn ra sau .  Quy trình xảy ra rất ít và một việc xây dựng có thể làm theo nhiều cách . | Các code được xây dựng và kiểm tra ngay khi nhà phát triển gửi mã lên kho. Jenkin sẽ xây dựng và kiểm tra code nhiều lần trong ngày .  Nếu quá trình xây dựng thành công, thì Jenkins sẽ triển khai code vào máy chủ thử nghiệm và thông báo cho nhóm triển khai.  Nếu quá trình xây dựng thất bại, thì Jenkins sẽ thông báo lỗi cho nhóm nhà phát triển. |
| Vì mã được xây dựng cùng một lúc, một số nhà phát triển sẽ cần đợi cho đến khi các nhà phát triển khác hoàn thành việc code của họ để kiểm tra bản dựng . | Code được xây dựng ngay lập tức sau khi bất kỳ lần gửi nào của nhà phát triển. |
| Không phải là một nhiệm vụ dễ dàng để cô lập, phát hiện và sửa lỗi cho nhiều lần gửi . | Vì code được xây dựng sau mỗi lần gửi của một nhà phát triển, nên rất dễ phát hiện code của ai gây ra lỗi khi xây dựng |
| Quá trình xây dựng code và kiểm tra là hoàn toàn thủ công, vì vậy có rất nhiều cơ hội cho thất bại . | Tự động xây dựng và quá trình kiểm tra tiết kiệm thời gian và giảm khuyết điểm , lỗi lầm . |
| Code được triển khai khi tất cả các lỗi được sửa chữa và kiểm tra. | Mã được triển khai sau mỗi lần xây dựng và thử nghiệm thành công. |
| Chu kỳ phát triển chậm | Chu kỳ phát triển nhanh. Các tính năng mới có sẵn hơn cho người dùng. Tăng lợi nhuận. |

Như đã miêu tả , với Jenkins , quá trình phát triển phần mềm được thay đổi rõ rệt ,

linh động hơn , giảm bớt rủi ro cũng như làm giảm đi khó khăn cho nhà phát triển .

**2.2.3 Lợi ích và khó khăn của Jenkins**

**Lợi ích :**

* Jenkins đang được quản lý bởi cộng đồng rất cởi mở. Hàng tháng, họ tổ chức các cuộc họp công cộng và lấy ý kiến của công chúng để phát triển dự án Jenkins.
* Khi công nghệ phát triển, Jenkins cũng vậy. Cho đến nay Jenkins có khoảng hơn 1000 plugin được xuất bản trong cơ sở dữ liệu plugin của nó. Với các plugin, Jenkins càng trở nên mạnh mẽ và giàu tính năng.
* Jenkins cũng hỗ trợ kiến trúc dựa trên đám mây để bạn có thể triển khai Jenkins trong các nền tảng dựa trên đám mây.
* Lý do tại sao Jenkins trở nên phổ biến là vì nó được tạo ra bởi một nhà phát triển cho các nhà phát triển.

**Khó khăn :**

* Giao diện của nó đã lỗi thời và không thân thiện với người dùng so với các xu hướng UI hiện tại.
* Mặc dù Jenkins được nhiều nhà phát triển yêu thích, nhưng không dễ để duy trì nó vì Jenkins chạy trên máy chủ và yêu cầu một số kỹ năng như quản trị viên máy chủ để theo dõi hoạt động của nó cũng như chi phí để duy trì máy chủ Jenkins .
* Một trong những lý do khiến nhiều người không triển khai Jenkins là do khó cài đặt và định cấu hình Jenkins.
* CI thường xuyên bị phá vỡ do một số thay đổi cài đặt nhỏ. CI sẽ bị tạm dừng và do đó đòi hỏi một số nhà phát triển chú ý.