TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KỲ MÔN KIẾN TẬP CÔNG NGHIỆP**

**TÌM HIỂU GITHUB/ AZURE/ JENKINS**

*Người hướng dẫn*: **TS TRẦN THANH PHƯỚC**

*Người thực hiện*: **NGUYỄN THIỆN HƯNG – 51603131**

**TRẦN BẢO DUY – 51603080**

Lớp , **16050301**

Khoá , **20**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KỲ MÔN KIẾN TẬP CÔNG NGHIỆP**

**TÌM HIỂU GITHUB/ AZURE/ JENKINS**

*Người hướng dẫn*: **TS TRẦN THANH PHƯỚC**

*Người thực hiện*: **NGUYỄN THIỆN HƯNG – 51603131**

**TRẦN BẢO DUY – 51603080**

Lớp , **16050301**

Khoá , **20**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

LỜI CẢM ƠN

Đây là phần tác giả **tự viết** ngắn gọn, thể hiện sự biết ơn của mình đối với những người đã giúp mình hoàn thành Luận văn/Luận án. Tuyệt đối không sao chép theo mẫu những “lời cảm ơn” đã có.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi / chúng tôi và được sự hướng dẫn của TS Nguyễn Văn A;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 14 tháng 6 năm 2019*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Thiện Hưng*

*Trần Bảo Duy*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Trình bày tóm tắt vấn đề nghiên cứu, các hướng tiếp cận, cách giải quyết vấn đề và một số kết quả đạt được, những phát hiện cơ bản trong vòng 1 -2 trang.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc11161340)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc11161341)

[TÓM TẮT iv](#_Toc11161342)

[MỤC LỤC 1](#_Toc11161343)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc11161344)

[CHƯƠNG 1 – GITHUB 5](#_Toc11161345)

[1.1 GitHub là gì? 5](#_Toc11161346)

[1.1.1 Repository là gì? 6](#_Toc11161347)

[1.1.2 Branch là gì? 7](#_Toc11161348)

[1.1.3 Clone là gì? 8](#_Toc11161349)

[1.1.4 Commit là gì? 8](#_Toc11161350)

[1.1.5 Fork là gì? 8](#_Toc11161351)

[1.1.6 Merge là gì? 9](#_Toc11161352)

[1.1.7 Push là gì? 9](#_Toc11161353)

[1.1.8 Pull là gì? 9](#_Toc11161354)

[1.1.9 Pull Request là gì? 9](#_Toc11161355)

[1.1.10 Issue là gì? 10](#_Toc11161356)

[1.1.11 Webhook là gì? 10](#_Toc11161357)

[1.1.1.1 Tiểu mục cấp 3 11](#_Toc11161358)

[1.1.1.2 Tiểu mục cấp 3 tiếp theo. 12](#_Toc11161359)

[1.1.2 Tiểu mục cấp 2 tiếp theo 12](#_Toc11161360)

[1.2 Nội dung của chương này 12](#_Toc11161361)

[2.1 TÍCH HỢP LIÊN TỤC 13](#_Toc11161362)

[2.1.1 Tích hợp liên tục là gì ? 13](#_Toc11161363)

[2.1.2 Đặc điểm của việc tích hợp liên tục 15](#_Toc11161364)

[2.1.3 Lợi ích và khó khăn của tích hợp liên tục, 16](#_Toc11161365)

[2.2 JENKINS 16](#_Toc11161366)

[2.2.1 Jenkins là gì ? 16](#_Toc11161367)

[2.2.2 CI với Jenkins 19](#_Toc11161368)

[CHƯƠNG 3 – CƠ SỞ LÝ THUYẾT / NGHIÊN CỨU THỰC NGHIỆM 20](#_Toc11161369)

[3.1 Chèn bảng: 20](#_Toc11161370)

[3.2 Viết tắt 20](#_Toc11161371)

[3.3 Trích dẫn 21](#_Toc11161372)

[3.3.1 Tài liệu tham khảo và cách trích dẫn 21](#_Toc11161373)

[3.3.2 Qui định của Khoa Công nghệ thông tin 21](#_Toc11161374)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

**CÁC KÝ HIỆU**

*f Tần số của dòng điện và điện áp (Hz)*

*p Mật độ điện tích khối (C/m3)*

**CÁC CHỮ VIẾT TẮT**

CSTD Công suất tác dụng

MF Máy phát điện

BER Tỷ lệ bít lỗi

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 2.1: Kiến trúc FTP 1](#_Toc387689394)

**DANH MỤC BẢNG**

[Bảng 3.1 Ví dụ cho chèn bảng 1](#_Toc387689363)

CHƯƠNG 1 – GITHUB

* 1. GitHub là gì?

***Trước khi tìm hiểu về GitHub thì ta cần biết về Version Control là gì?***

Version Control giúp các developer kiểm soát và quản lý được những thay đổi đến với phần code của dự án phần mềm. Vì sự phát triển của phần mềm, version control trở nên cần thiết hơn.

Với **branching**, một developer nhân đôi phần mã nguồn (được gọi là **repository)**. Developer có thể thực hiện thay đổi một cách an toàn cho phần code mà không ảnh hưởng đến phần còn lại của dự án.

Sau đó, khi developer lấy phần code hoạt động chính xác, người đó có thể **merge**  code đó trở lại mã nguồn chính.

***Vậy GitHub là gì?***

GitHub là một công ty vì lợi nhuận cung cấp dịch vụ lưu trữ kho Git dựa trên Cloud. Về cơ bản, nó giúp các cá nhân và nhóm sử dụng Git để kiểm soát phiên bản và cộng tác dễ dàng hơn rất nhiều.

Giao diện GitHub, đủ thân thiện với người dùng để ngay cả các lập trình viên mới làm quen cũng có thể tận dụng được Git. Không có GitHub, sử dụng Git thường đòi hỏi một chút hiểu biết về kỹ thuật và sử dụng dòng lệnh (command line).

GitHub rất thân thiện với người dùng, mặc dù vậy, một số người thậm chí còn sử dụng GitHub để quản lý các dự án khác – như viết sách.

Ngoài ra, bất kỳ ai cũng có thể đăng ký và lưu trữ kho lưu trữ mã công khai miễn phí, điều này khiến Github đặc biệt phổ biến với các dự án nguồn mở.

Là một công ty, GitHub kiếm tiền bằng cách bán các kho **private repository** được lưu trữ, cũng như các kế hoạch tập trung vào kinh doanh khác giúp các tổ chức dễ dàng quản lý các thành viên nhóm và bảo mật.

Ở đây chúng em sẽ nói những thứ cơ bản nhất và mang tính thiết yếu về WorkFlow GitHub trong: Repositories, Branches, commits và Pull Request. Ví dụ về cách tạo ra một Repository có tên Hello World, và workflow của việc Pull Request, một cách quen thuộc để create và review code. Github được xem là một ứng dụng giúp xây dựng các remote repository, các chức năng của nó đều phát triển dựa vào thư viện Git.

Ví dụ khi chúng ta làm một dự án và cần lưu trữ trên remote repo, lúc này bắt buộc phải mua server và xây dựng remote trên đó, điều này khá tốn chi phí và có khi lại không bảo mật. Thay vào đó chúng ta nên sử dụng Github để tạo remote repo, sau đó các thành viên sẽ liên kết tới remote này để làm việc.

* + 1. Repository là gì?

Repository hay còn gọi là Repo, dịch ra tiếng Việt có nghĩa là kho, đây chính là nơi chứa tất cả mã nguồn cho một dự án được

Phân loại Repo:

***Local repository:*** Là repo được cài đặt trên máy tính của lập trình viên, repo này sẽ đồng bộ hóa với remote repo bằng các lệnh của git

***Remote repository:*** Là repo được cài đặt trên server chuyên dụng, điển hình hiện nay là GitHub.



Nguồn, <https://stackoverflow.com/questions/13072111/gits-local-repository-and-remote-repository-confusing-concepts>

Work follow.

Khi thực hiện các thay đổi dự án một cách cục bộ (locally), bạn có thể cập nhật chúng với kho lưu trữ từ xa. Trong Git, một điều khiển từ xa là máy chủ nơi code lưu trữ. Trong trường hợp, máy chủ đó là một kho lưu trữ trên GitHub hoặc GitHub Enterprise.

* + 1. Branch là gì?

Một branch là phiên bản song song của **repository**. Nó chứa trong **repository**, nhưng không ảnh hưởng đến nhánh chính hoặc nhánh chính cho phép làm việc tự do mà không làm gián đoạn phiên bản “trực tiếp”. Khi ta tạo ra những thay đổi, ta có thể **merge** branch vào lại ***branch master*** để publish những thay đổi của bạn.

* + 1. Clone là gì?

Một **clone** là một bản copy của repo đang sống trong máy tính của bạn thay vì trên server website ở nơi nào đó, hoặc hành động tạo ra bản copy đó. Với bản **clone** chúng ta có thể edit các file trong các editor và sử dụng Git để theo dõi những thay đổi của bản mà không cần phải trực tuyến. Tuy nhiên, nó được kết nối với phiên bản từ xa để những thay đổi có thể được đồng bộ hóa giữa hai phiên bản. Ta có thể **push** các thay đổi cục bộ đểgiữ cho chúng được đồng bộ hóa khi ta dùng phiên bản remote/ online.

* + 1. Commit là gì?

Một commit hay “revision”, là một thay đổi riêng lẻ đối với một tệp (hoặc bộ tệp). Giống như khi bạn lưu tệp, ngoại trừ với Git, mỗi khi bạn lưu tệp đó sẽ tạo một ID duy nhất (a.k.a. “SHA” hoặc là “băm”) cho phép bạn ghi lại những thay đổi được thực hiện khi nào và bởi ai.Commit thường chứa một commit message là một mô tả ngắn gọn về những thay đổi đã được thực hiện.

Commit này đang được chứa tại repository, các commit nối tiếp với nhau theo thứ tự thời gian. Bằng việc lần theo commit này từ trạng thái mới nhất thì có thể biết được lịch sử thay đổi trong quá khứ hoặc nội dung thay đổi đó.

* + 1. Fork là gì?

Một Fork là một bản copy của một repository(Kho chứa source code của bạn trên GitHub). Việc fork một repository cho phép bạn dễ dàng chỉnh sửa, thay đổi source code mà không cho phép bạn dễ dàng chỉnh sửa, thay đổi source code mà không ảnh hưởng tới source gốc.

Một ví dụ về việc sử dụng fork, là khi bạn muốn fix bug source code trên repository của một ai đó, khi đó bạn cần thực hiện theo quy trình sau:

1. Fork repository đó về tài khoản Github của mình.
2. Thực hiện fix bug.
3. Gửi một Pull Request tới repository gốc.

Khi chủ sở hữu của repository nơi ta fork, sẽ review chỉnh sửa thay đổi của bạn, và tiến hành merge nội dung chỉnh sửa vào source gốc.

* + 1. Merge là gì?

Merger source từ một nhánh khác vào nhánh hiện tại. Bạn sẽ hiểu sâu hơn về nó trong các ví dụ ở các bài sau.

Chú ý:

- Kiểm tra branch hiện đang làm việc trước khi merge

- Phải đẩy tất cả những thay đổi dưới máy local lên Git trước khi merge

- Trước khi merge phải lấy hết những thay đổi mới nhất của các branch khác, hay ít nhất branch cần merge về máy.

- Merge thành công thì nên đẩy source lên lại server

- Nên merge bằng GUI tool.

* + 1. Push là gì?

Push được sử dụng để đưa nội dung kho lưu trữ cục bộ lên server. Push là cách bạn chuyển giao các commit từ kho lưu trữ cục bộ của bạn lên server.Nếu bạn có những thay đổi gì, thì bạn muốn đẩy những thay đổi đó để người khác có thể truy cập chúng.

* + 1. Pull là gì?

Pull đề cập đến khi ta đang fetching các thay đổi và merging chúng. Ví dụ: nếu ai đó đã chỉnh sửa tệp từ xa mà cả hai bạn đang làm việc, bạn sẽ muốn **pull** các thay đổi đó sao cho bản sao cục bộ của mình để cập nhật.

* + 1. Pull Request là gì?

**Pull Request** để bắt đầu thảo luận về những commit của bạn. Khi commit được mở, bạn có thể thảo luận và xem xét các thay đổi tiềm năng với các collaborator va theo dõi các commit trước khi các thay đổi được merge vào master branch.Vì chúng tích hợp chặt chẽ với repo Git bên dưới, bất cứ ai cũng có thể thấy chính xác những thay đổi sẽ được merge nếu họ chấp nhận lời request của bạn.

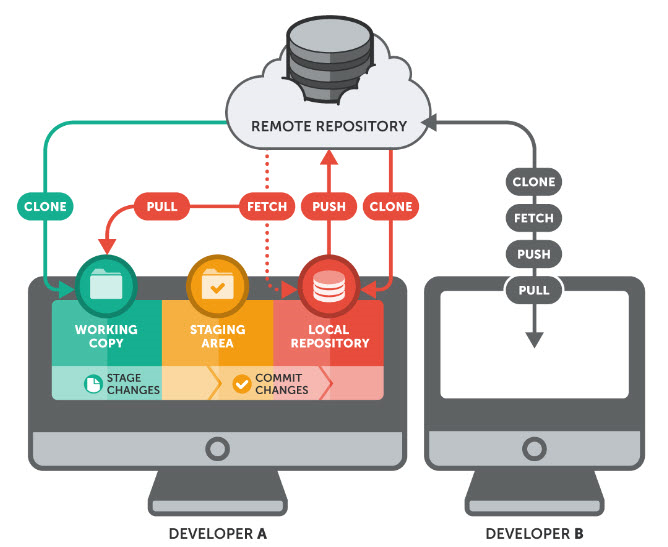
Sau khi khởi tạo PR, bạn sẽ thấy trang đánh giá hiển thị tổng quan high-level về các thay đổi giữa các branch của bạn

Bạn có thể mở 1 cái pull request tại bất kỳ thời điểm nào trong quá trình phát triển: khi bạn có ít hoặc không có mã nhưng muốn chia sẻ một số screenshot hoặc ý tưởng chung. Khi bạn gặp khó và cần giúp đỡ hoặc lời khuyên, hoặc bạn đã sẵn sàng để ai đó xem xét công việc của bạn. Bằng cách sử dụng hệ thống @mention của GitHub trong tin nhắn Pull Request, bạn có yêu cầu phản hồi từ những người hoặc nhóm cụ thể.

* + 1. Issue là gì?

Issue là nơi để đề xuất các cải tiến, nhiệm vụ hoặc câu hỏi liên quan đến kho lưu trữ. Các issue được tạo ra bởi bất kỳ ai (cho public repository), và được kiểm duyệt bởi repository collaborator (người cộng tác trong kho lưu trữ). Mỗi issue chứa diễn đàn thảo luận riêng, có thể được gắn nhãn và gán cho người dùng.

* + 1. Webhook là gì?



Các thao tác xử lý giữa remote repository và local repository

<https://csc.edu.vn/lap-trinh-va-csdl/tin-tuc/kien-thuc-lap-trinh/Git-la-gi--Nhung-khai-niem-co-ban-khi-lam-viec-tren-Git-4133>

1.1.1.1 Tiểu mục cấp 3

- Đây là cấp tiểu mục nhỏ nhất, không thể tiếp tục phân chia.

- Các ý trong tiểu mục được trình bày gạch đầu dòng “-“.

- Các ý nhỏ hơn sử dụng bullet như sau:

* Ý nhỏ 1.
* Ý nhỏ 2.

- Cần lưu ý rằng đây là cấp sâu nhất, không được phép chia thành 1.1.1.1.1.

1.1.1.2 Tiểu mục cấp 3 tiếp theo.

Nội dung của tiểu mục thứ ba, khi soạn thảo hãy dùng Styles có sẵn, để khi tạo mục lục sẽ tự động và đồng nhất mỗi khi chúng ta thay đổi format.

1.1.2 Tiểu mục cấp 2 tiếp theo

Không phải lúc nào cũng chia thành tiểu mục cấp 3, nếu như ý trình bày được gói gọn.

1.2 Nội dung của chương này

Chương này trình bày lý do chọn đề tài, mục đích, đối tượng và phạm vi nghiên cứu, ý nghĩa khoa học và thực tiễn của đề tài; cơ sở khoa học của việc chọn đề tài...;

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

*Tổng quan*: phân tích, đánh giá các công trình nghiên cứu đã có của tác giả, các tác giả khác trong và ngoài nước liên quan mật thiết đến đề tài; nêu những vấn đề còn tồn tại; chỉ ra những vấn đề mà đề tài cần tập trung nghiên cứu, giải quyết;

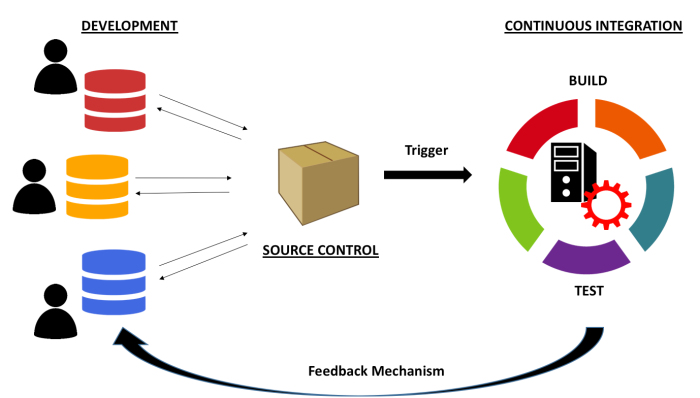
2.1 TÍCH HỢP LIÊN TỤC

2.1.1 Tích hợp liên tục 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 nhóm 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 nhóm liên quan về kết quả xây dựng và thử nghiệm.

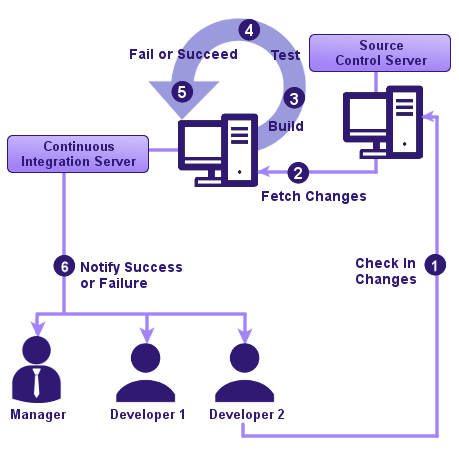


Hình 2.1 Mô hình tích hợp liên tục

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 bất kỳ lỗi nào 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 ( repository )
* Khi code có sự thay đổi thì sẽ được xây dựng lại thông qua build server.
* 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 hang 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.
  + Nâng cáo 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.

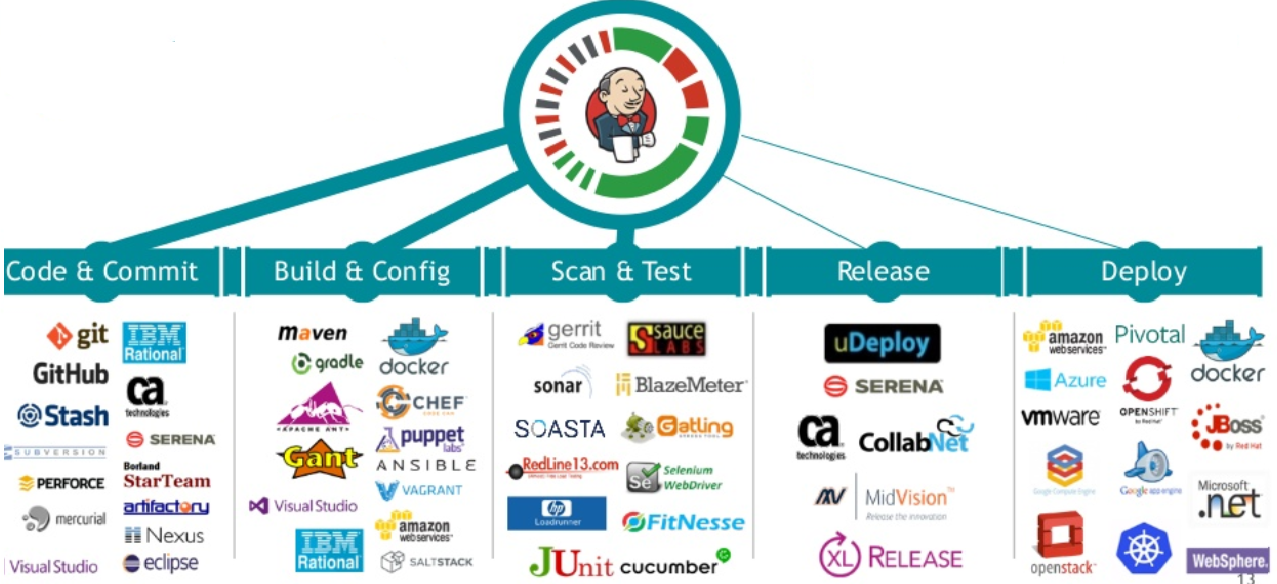


Hình 2.3 Build, test và deploy với Jenkins

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.

CHƯƠNG 3 – CƠ SỞ LÝ THUYẾT / NGHIÊN CỨU THỰC NGHIỆM

*Những nghiên cứu thực nghiệm hoặc lý thuyết*: trình bày các cơ sở lý thuyết, lý luận, giả thuyết khoa học và phương pháp nghiên cứu sẽ được sử dụng trong Luận văn, Luận án;

3.1 Chèn bảng:

|  |  |  |
| --- | --- | --- |
| **STT** | **Tiêu đề A** | **Tiêu đề B** |
| 1 | Nội dung 1 | Nội dung 4 |
| 2 | Nội dung 2 | Nội dung 5 |
| 3 | Nội dung 3 | Nội dung 6 |

Bảng 3.1Ví dụ cho chèn bảng

Khi cần chèn tên bảng thì chọn References \ Caption và chọn “Bảng …”

3.2 Viết tắt

Không lạm dụng việc viết tắt. Chỉ viết tắt những từ, cụm từ hoặc thuật ngữ *được sử dụng nhiều lần trong luận văn*. Không viết tắt những cụm từ dài, những mệnh đề hoặc những cụm từ ít xuất hiện. Nếu cần viết tắt những từ, thuật ngữ, tên các cơ quan, tổ chức... thì được viết tắt sau lần viết thứ nhất có kèm theo chữ viết tắt trong ngoặc đơn. Nếu có quá nhiều chữ viết tắt thì phải có bảng danh mục các chữ viết tắt (xếp theo thứ tự A, B, C) ở phần đầu luận văn.

3.3 Trích dẫn

3.3.1 Tài liệu tham khảo và cách trích dẫn

Mọi ý kiến, khái niệm, phân tích, phát biểu, diễn đạt... có ý nghĩa, mang tính chất gợi ý *không phải của riêng tác giả* và mọi tham khảo khác **phải được trích dẫn và chỉ rõ nguồn trong danh mục Tài liệu tham khảo của luận văn**. Phải nêu rõ cả việc sử dụng những đề xuất hoặc kết quả của đồng tác giả (*đối với công trình đã công bố khác thì phải trích dẫn bình thường như một tài liệu tham khảo*). Nếu sử dụng tài liệu của người khác và của đồng tác giả (bảng biểu, hình vẽ, công thức, đồ thị, phương trình, ý tưởng...) mà không chú dẫn tác giả và nguồn tài liệu thì **luận văn không được duyệt để bảo vệ**.

Không trích dẫn những kiến thức phổ biến, mọi người đều biết tránh làm nặng nề phần tham khảo trích dẫn.

Nếu người dẫn liệu không có điều kiện tiếp cận được một tài liệu gốc mà phải trích dẫn thông qua một tài liệu khác của một tác giả khác, thì phải nêu rõ cách trích dẫn (*lưu ý phải ghi* *đúng nguyên văn từ chính tài liệu tham khảo và hạn chế tối đa hình thức này).* Nếu cần trích dẫn dài hơn thì phải tách phần này thành một đoạn riêng khỏi phần nội dung đang trình bày, in nghiêng, với lề trái lùi vào thêm 2 cm. Khi mở đầu và kết thúc đoạn trích này không phải sử dụng dấu ngoặc kép. Việc trích dẫn là theo thứ tự của tài liệu ở danh mục Tài liệu tham khảo và được đặt trong ngoặc vuông, khi cần có cả số trang, ví dụ [15, tr.314-315]. Đối với phần trích dẫn từ nhiều tài liệu khác nhau, số của từng tài liệu được đặt độc lập trong từng ngoặc vuông, theo thứ tự tăng dần, ví dụ [19], [25], [41], [42].

3.3.2 Qui định của Khoa Công nghệ thông tin

**- Đạo văn** là việc sử dụng từ ngữ hay ý tưởng của người khác như là của mình trong hoạt động học thuật nói riêng và trong hoạt động sáng tạo nói chung. Tại Đại học Hoa Sen, những hành vi sau đây được xem là đạo văn:

* Sao chép nguyên văn **02** (hai) câu liên tiếp mà không dẫn nguồn đúng quy định;
* Sao chép nguyên văn **03** (ba) câu không liên tiếp mà không dẫn nguồn đúng quy định;
* Diễn đạt lại (rephrase) hoặc dịch (translate) toàn bộ một ý nào đó của người khác mà không dẫn nguồn đúng quy định;
* Sử dụng hơn 30% nội dung của một báo cáo cuối kỳ do chính mình viết để nộp cho 2 lớp khác nhau (cùng học kỳ hoặc khác học kỳ) mà không có sự đồng ý của giảng viên;
* Sao chép một phần hoặc toàn bộ bài làm của người khác.

- Khi luận văn, đồ án, bài tập lớn, được chấm điểm, nếu bị phát hiện đạo văn thì ngay lập tức bị điểm 0. Sinh viên sẽ tiếp tục bị xử lý kỷ luật theo các qui định của Nhà trường.

**TÀI LIỆU THAM KHẢO**

**Tiếng Việt**

1. Quách Ngọc Ân (1992), “Nhìn lại hai năm phát triển lúa lai”, *Di tuyền học ứng dụng*, 98(1), tr. 10-16.
2. Bộ nông nghiệp & PTNT (1996), *Báo cáo tổng kết 5 năm (1992-1996) phát triển lúa lai,* Hà Nội.
3. Nguyễn Hữu Đống, Đào Thanh Bằng, Lâm Quang Dụ, Phan Đức Trực (1997), *Đột biến –* *Cơ sở lý luận và ứng dụng,* Nhà xuất bản nông nghiệp, Viện khoa học kỹ thuật nông nghiệp Việt Nam, Hà Nội.
4. Nguyễn Thị Gấm (1996), *Phát hiện và đánh giá một số dòng bất dục đực cảm ứng nhiệt* *độ,* Luận văn thạc sĩ khoa học nông nghiệp, Viện khoa học kỹ thuật nông nghiệp Việt Nam, Hà Nội.

……….

1. Võ Thị Kim Huệ (2000), *Nghiên cứu chẩn đoán và điều trị bệnh…,* Luận án Tiến sĩ y khoa, Trường đại học y Hà Nội, Hà Nội.

**Tiếng Anh**

1. Anderson J.E. (1985), The Relative Inefficiency of Quota, The Cheese Case, *American* *Economic Review*, 75(1), pp. 178-90.
2. Borkakati R. P.,Virmani S. S. (1997), Genetics of thermosensitive genic male sterility in Rice, *Euphytica* 88, pp. 1-7.
3. Boulding K.E. (1955), *Economics Analysis*, Hamish Hamilton, London.
4. Burton G. W. (1988), “Cytoplasmic male-sterility in pearl millet (penni-setum glaucum L.)”, *Agronomic Journal* 50, pp. 230-231.
5. Central Statistical Oraganisation (1995), *Statistical Year Book*, Beijing.
6. FAO (1971), *Agricultural Commodity Projections (1970-1980)*, Vol. II. Rome.
7. Institute of Economics (1988), *Analysis of Expenditure Pattern of Urban Households in* *Vietnam,* Departement pf Economics, Economic Research Report, Hanoi.

**PHỤ LỤC**

Phần này bao gồm những nội dung cần thiết nhằm minh họa hoặc hỗ trợ cho nội dung luận văn như số liệu, biểu mẫu, tranh ảnh.... nếu sử dụng những câu trả lời cho một *bảng câu hỏi thì bảng câu hỏi mẫu này phải được đưa vào phần Phụ lục ở dạng nguyên bản* đã dùng để điều tra, thăm dò ý kiến; **không được tóm tắt hoặc sửa đổi**. Các tính toán mẫu trình bày tóm tắt trong các biểu mẫu cũng cần nêu trong Phụ lục của luận văn. Phụ lục không được dày hơn phần chính của luận văn