Giới thiệu một mô hình sử dụng Git branches hiệu quả

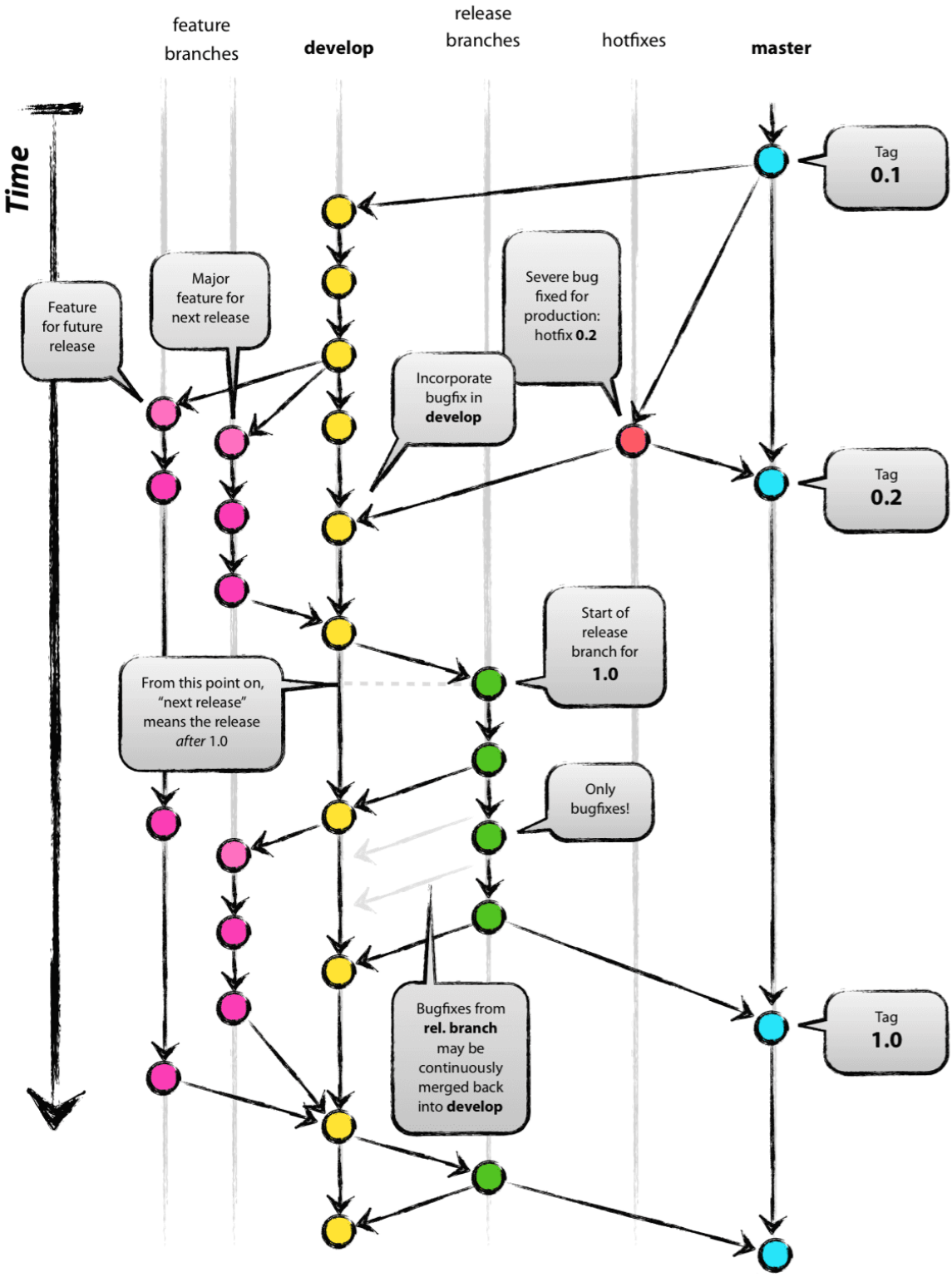
21/11/2015 / Bởi **Techmaster Team** / trong lập trình

Mới đầu mình tính viết về git-flow - một tool đơn giản dùng để quản lý Git branches, xong thấy bài viết này về một ví dụ sử dụng Git branches hay quá, nên quyết định dịch luôn.

Link nguồn bài viết của bạn kanamikiii

Mở đầu

Ngay bây giờ, tôi sẽ giới thiệu với các bạn mô hình sử dụng Git mà tôi đã và đang sử dụng trong các dự án khoảng một năm trở lại đây. Đó là một mô hình thực sự thành công, nhưng mãi đến giờ tôi mới có cơ hội để có thể viết về nó và chia sẻ với các bạn. Tôi sẽ không đi vào chi tiết dự án, mà chỉ xoay quanh chiến lược quản lý các phân nhánh mà thôi.



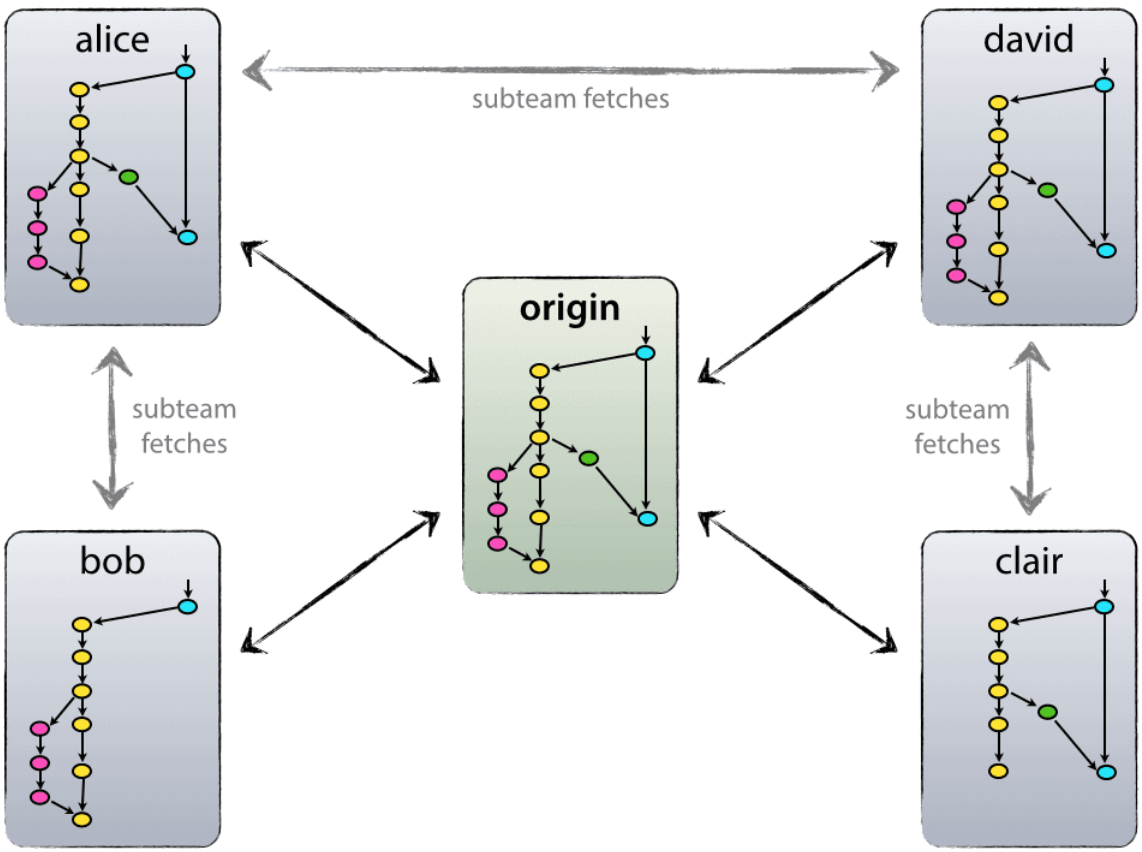
Trong bài viết dưới đây, tôi sử dụng Git để quản lý version cho toàn bộ source code. Nếu các bạn có hứng thú với việc phân tích dữ liệu thời gian thực dựa vào hoạt động trên Git, có thể tham khảo phần mềm mà GitPrime mà công ty tôi phát triển.

Tại sao lại chọn Git?

Đã có quá nhiều cuộc chiến nổ ra khi bàn về ưu điểm nhược điểm của Git so với các công cụ quản lý source code tập trung khác (như CVS, SVN). Là một developer, tôi cảm thấy yêu thích Git hơn. Git thực sự đã thay đổi cách suy nghĩ về merging và branching. Những ai đã từng làm việc với CVS/Subversion đều hiểu merging/branching không hề đơn giản, trong tài liệu về CVS/Subversion branching và merging chỉ được nhắc đến ở chapter cuối, dành cho các users có kinh nghiệm. Còn với Git, branching/merging là một phần trong workflow hàng ngày của bạn, được mô tả ở những phần đầu tiên trong Git book.

Phân tán, nhưng tập trung

Mô hình Git mà tôi sử dụng hoạt động xoay quanh một repo trung tâm. Nên nhớ đó "chỉ được xem như" là repo trung tâm, chứ về mặt kỹ thuật thì GIT không hề có cái gì gọi là repo trung tâm cả. Ví dụ như repo origin ở hình bên dưới.



Mỗi developer sẽ pull và push với origin. Bên cạnh đó, mỗi người có thể pull những thay đổi từ những người khác để tạo thành các sub teams. Điều này sẽ thực sự có ích khi phải làm việc nhóm 2-3 người trở lên để hoàn thành một

feature lớn, mà không phải đẩy source code vẫn đang dở dang lên origin. Ở hình vẽ bên trên, các subteams là Alice-Bob, Alice-David, Clair-David.

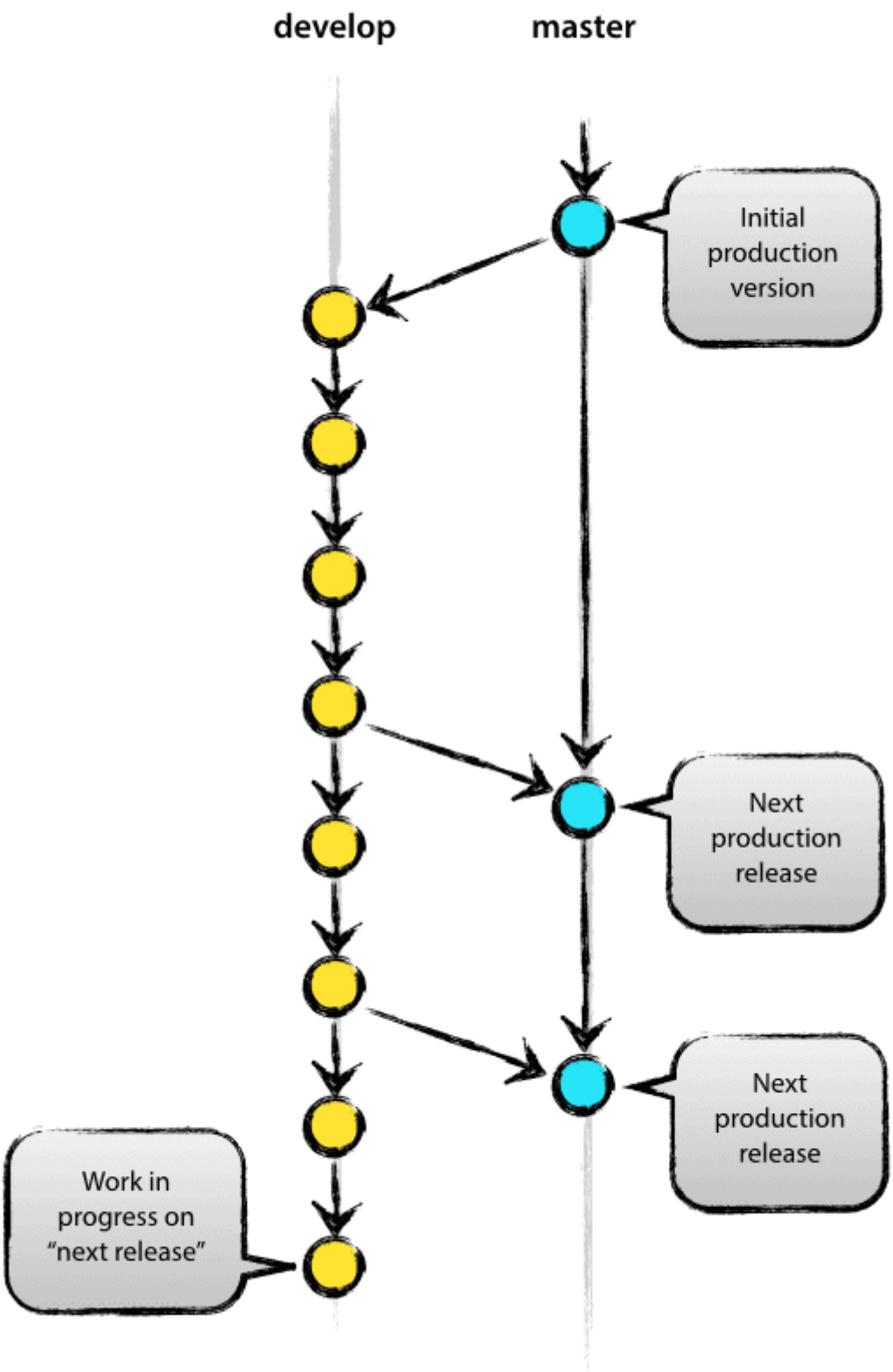
Về mặt kỹ thuật, đơn giản là Alice định nghĩa một Git remote là bob, trỏ đến repo của Bob, và ngược lại.

Những branches chính

Repo trung tâm sẽ chứa hai branches chính hoạt động mãi mãi:

- master
- develop

Nhánh master tại origin là nhánh quen thuộc với tất cả các Git users. Song song là nhánh develop.



origin/master được coi là nhánh chính với HEAD phản ánh trạng thái production-ready.

origin/develop được coi là nhánh chính với HEAD phản ánh trạng thái thay đổi mới nhất trong quá trình phát triển, chuẩn bị cho release tiếp theo.

Khi source code bên develop đạt đến một mức độ ổn định nào đó và sẵn sàng để release thì sẽ được merge sang bên master và đánh dấu với release

number.

Như vậy, theo định nghĩa về nhánh master, chúng ta mặc định hiểu rằng khi có thay đổi được merge vào master thì tức là sẽ có một phiên bản production mới được release. Nhờ đó chúng ta có thể sử dụng script để tự động build lên production server mỗi khi có commit ở master.

Những branches phụ

Bên cạnh hai branches chính master và develop, mô hình mà tôi đang sử dụng còn có thêm rất nhiều những branches phụ để giúp các team members có thể phát triển song song, dễ dàng tracking theo features, chuẩn bị cho release hoặc fix nhanh các vấn đề production. Khác với hai branches chính kia, các branches phụ này chỉ tồn tại trong một khoảng thời gian ngắn, rồi sẽ bị xoá đi.

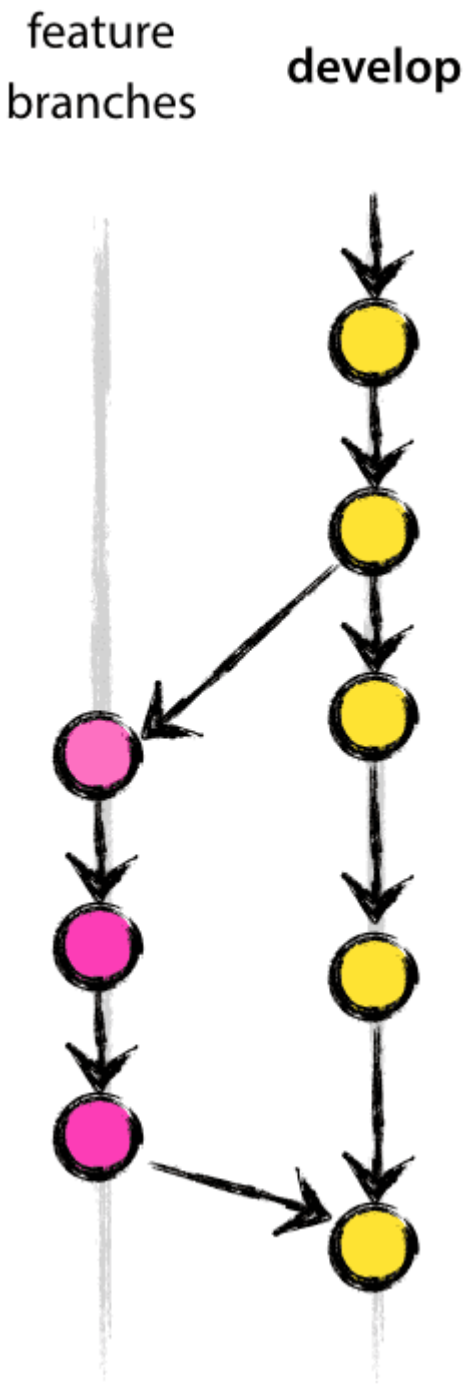
- Feature branches
- Release branches
- Hotfix branches

Phía trên là các loại branches khác nhau tôi hay sử dụng. Mỗi loại branches lại có một nhiệm vụ riêng, và cách xử lý riêng. Tôi sẽ đi sâu vào phân tích ở đoạn sau.

Về mặt kỹ thuật, chả có branch nào là "đặc biệt" so với các branches khác cả. Tất cả chỉ là Git branches thông thường, chúng chỉ được phân loại bằng cách ta sử dụng ra sao thôi.

Feature branches

- Tách từ: develop
- Merge vào: develop
- Naming convention: tự do, ngoại trừ master, develop, release-*, hotfix-*



Feature branches (hay còn gọi là topic branches) được sử dụng để phát triển các feature mới phục vụ cho release sau này. Khi bắt đầu phát triển một chức năng, có thể chưa rõ được thời điểm chức năng đó được tích hợp vào hệ thống và release. Feature branch sẽ tồn tại trong quá trình chức năng được phát triển, cuối cùng sẽ được merge lại vào develop (khi quyết định lần release tới bao gồm chức năng đó) hoặc bị bỏ đi (khi thấy chức năng không còn cần thiết).

Về cơ bản thì feature branches chỉ tồn tại ở repos của developers, chứ ko phải ở origin.

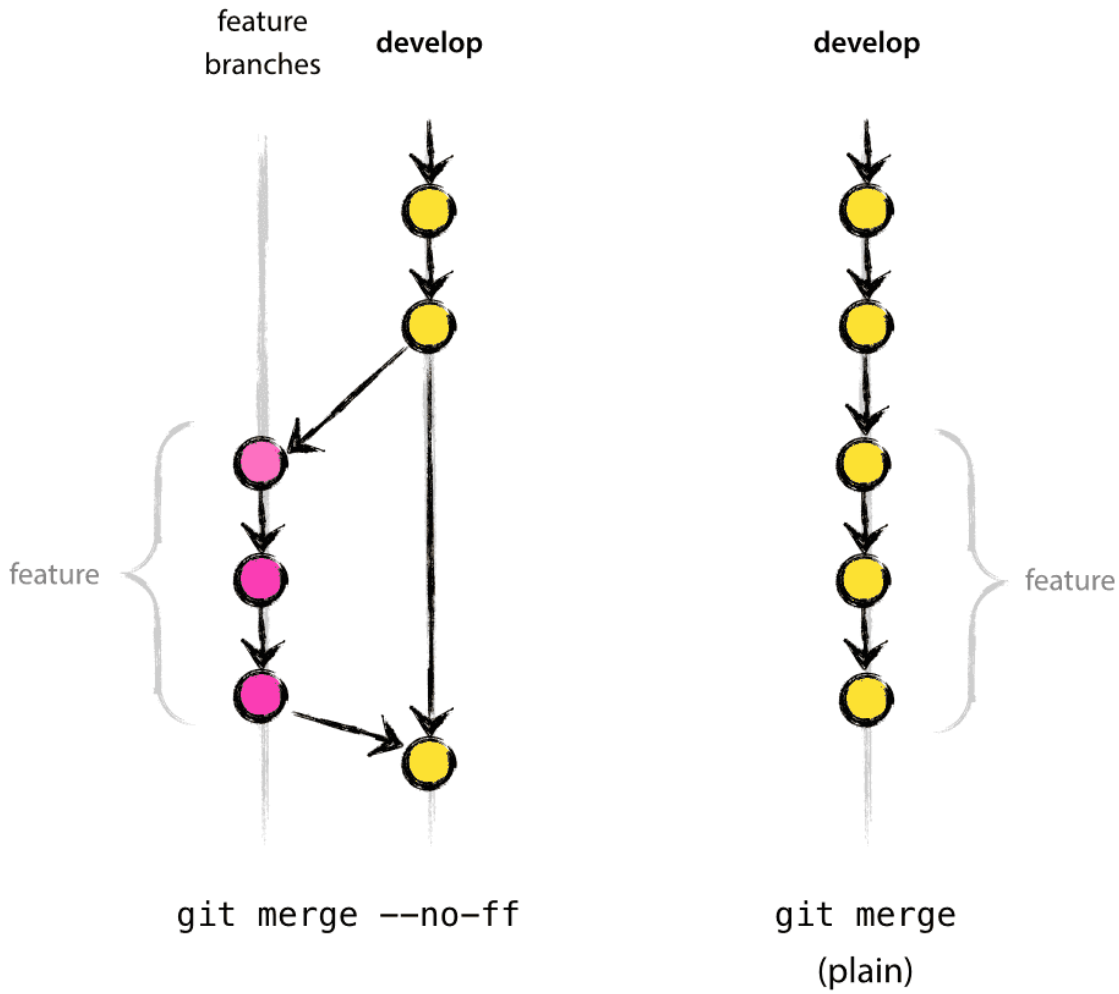
Tạo feature branch

```
$ git checkout -b myfeature develop
```

Merge vào develop

```
$ git checkout develop
$ git merge --no--ff myfeature
$ git branch -d myfeature
$ git push origin develop
```

--no-ff giúp thao tác merge luôn tạo ra một commit mới, ngay cả khi có thể merge theo fast-forward. Flag này giúp chúng ta không bị mất thông tin liên quan đến lịch sử các commits của feature branch.



Ở trường hợp bên phải, không thể nhận biết được những commits nào phát triển cùng chức năng nếu không ngồi đọc log message của từng commit. Khi dấy, nếu muốn revert lại cả feature (phải revert nhiều commits liên quan) thì thực sự là đau đầu. Và đó là lý do mà --no-ff được sử dụng.

Đương nhiên, nó sẽ tạo ra thêm vài commit, nhưng chả có vấn đề gì cả.

Release branches

- Tách từ: develop
- Merge vào: develop và master
- Naming convention: release-*

Release branches được sử dụng để chuẩn bị cho release bản production mới. Tất cả các công việc cuối cùng trước khi release sẽ được thực hiện ở đây, ngoài ra còn để fix nốt các bugs lẻ tẻ, chuẩn bị meta-data (version number, build dates, etc..). Nhờ việc tách nhánh ra khỏi develop, chúng ta có thể tiếp tục phát triển các features cho đợt release khác một cách bình thường.

Thời điểm được lựa chọn để tách nhánh từ develop là khi develop phản ánh được trạng thái mong muốn cho việc release mới. Ít nhất lúc đó tất cả các features dành cho đợt release phải được merge vào develop rồi. Những features nhắm đến các lần release sau thì chưa được merge vào, phải đợi sau khi tách nhánh.

Chúng ta sẽ tiến hành đánh version theo rule của dự án ngay sau khi tạo release branch.

Tạo release branch

```
$ git checkout -b release-1.2 develop
$ ./bump-version.sh 1.2
$ git commit -a -m "Bumped version number to 1.2"
```

Ở ví dụ trên, bump-version.sh tượng trưng cho một script thay đổi một vài files trong source code để phản ánh version mới. Sau khi tạo branch mới và chuyển sang branch đó, chúng ta sẽ thực hiện nâng version, rồi commit thao tác đó.

Branch mới này sẽ tồn tại cho đến khi việc release được thực hiện gọn ghẽ. Trong khoảng thời gian đó, có thể thực hiện fix bugs ở branch này, tuy nhiên nghiêm cấm việc bổ sung feature mới lên đó. Tốt nhất nếu có features mới thì hãy merge vào develop, và đợi đợt release sau.

Kết thúc release branch

Khi source code trên release branch sẵn sàng để release, đầu tiên, phải merge vào master, sau đó phải đc merge lại vào develop để những lần release sau cũng chứa những thay đổi ở lần này.

```
$ git checkout master
$ git merge --no-ff release-1.2
$ git tag -a 1.2
```

Vậy là source code đã được release lên master, và đã được tag để tiện sau này tham chiếu.

```
$ git checkout develop
$ git merge --no-ff release-1.2
```

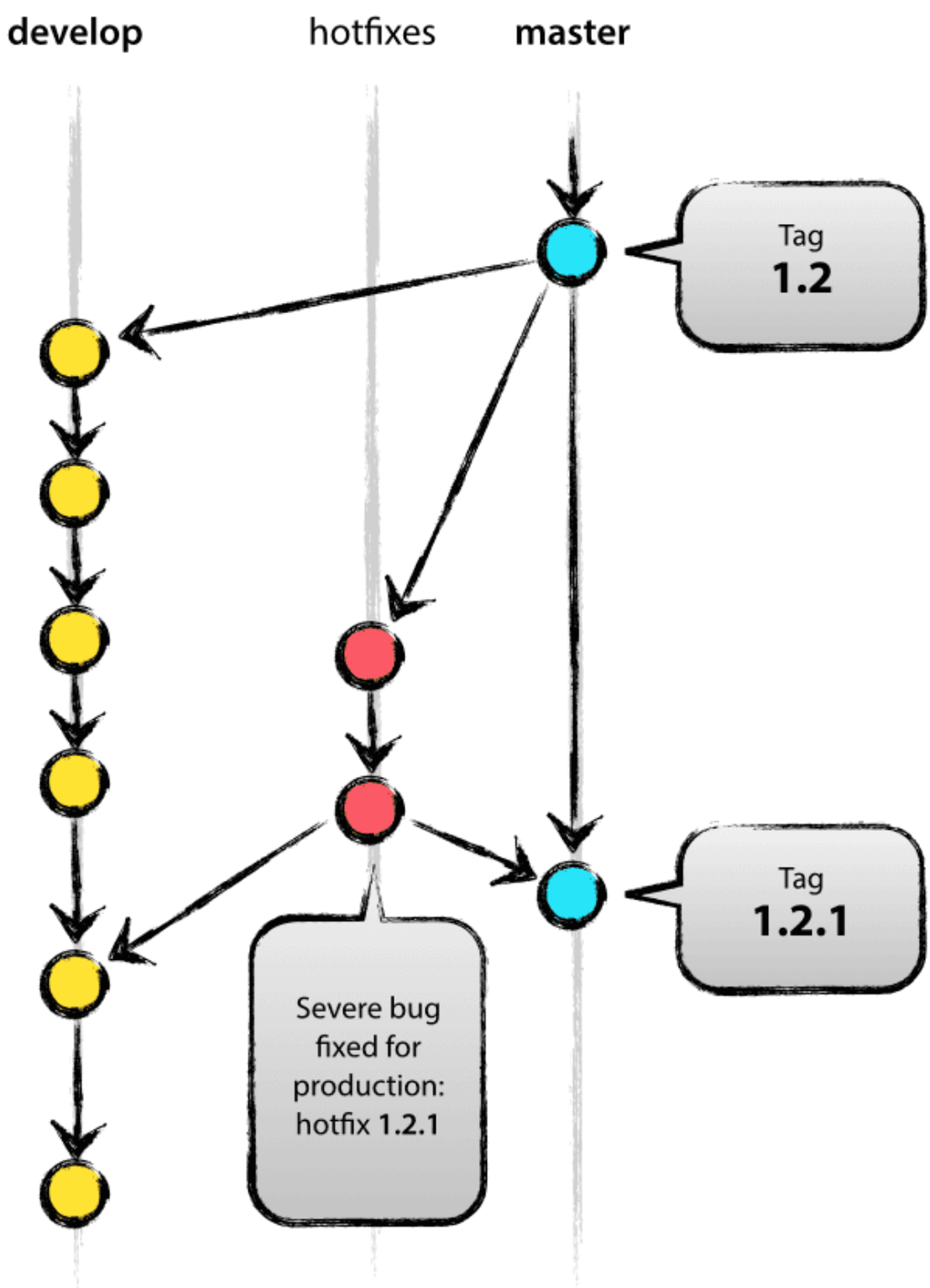
Ở bước này, rất có thể sẽ có confict, nên hãy fix nó rồi commit nhé.

Bây giờ thì việc release đã hoàn thành, và chúng ta ko cần đến branch này nữa.

```
$ git branch -d release-1.2
```

Hotfix branches

- Tách từ: master
- Merge vào: develop và master
- Naming convention: hotfix-*



Hotfix branches cũng giống release branches ở chỗ được sử dụng để chuẩn bị cho việc release production mới, chỉ khác ở chỗ là ko có plan từ trước. Khi có một bug nghiêm trọng trên bản production cần được giải quyết ngay lập tức, một hotfix branch sẽ được tách ra từ master và được đánh version để nhận biết.

Ưu điểm của việc tách nhánh này ở chỗ các team members khác có thể tiếp tục công việc ở develop trong khi những người khác có thể tập trung vào fix

bug của production.

Tạo hotfix branch

Hotfix branch được tạo ra từ master. Ví dụ hiện tại version 1.2 là phiên bản production đang chạy và xuất hiện lỗi nghiêm trọng. Tuy nhiên source code trên develop vẫn chưa ổn định, vì thế chúng ta phải tách nhánh hotfix và tiến hành sửa lỗi.

```
$ git checkout -b hotfix-1.2.1 master
$ ./bump-version.sh 1.2.1
$ git commit -a -m "Bumped version number to 1.2.1"
```

Sau khi tách nhánh phải tiến hành up version luôn nhé!

Sau khi sửa lỗi, hãy thực hiện commit.

```
$ git commit -m "Fixed severe production problem"
```

Kết thúc hotfix branch

Sau khi kết thúc sửa lỗi, những thay đổi phải được merge lại master, đồng thời cũng phải merge vào develop để ngăn lỗi xảy ra ở những lần release sau. Nghe rất giống với xử lý trên release branch phải không.

```
$ git checkout master
$ git merge --no-ff hotfix-1.2.1
$ git tag -a 1.2.1

$ git checkout develop
$ git merge --no-ff hotfix-1.2.1
```

Tuy nhiên, có một điểm cần lưu ý rằng: khi đang tồn tại một release branch thì cần phải merge hotfix vào release branch đó, thay cho develop. Khi release branch được merge vào develop thì cuối cùng những thay đổi trong hotfix cũng được merge vào develop, nên không có vấn đề gì cả. Trừ khi thực sự công việc ở develop cần phần hotfix ngay lập tức và ko thể đợi release branch được merge, thì cần cẩn thận merge hotfix vào develop.

Cuối cùng, chúng ta cũng ko cần đến branch này nữa:

```
$ git branch -d hotfix-1.2.1
```

Link nguồn bài viết : Giới thiệu một mô hình sử dụng Git branches hiệu quả

DevOps
23/05/2016 Techmaster team

BLOG HOME

Một lập trình viên nên biết 6 công nghệ cần học trong 2013
08/11/2013 Techmaster team

Chủ sở hữu website

Công ty TNHH TechMaster Vietnam Ltd

Số ĐKDN: 0105392153

Ngày cấp: 4-7-2011

Nơi cấp: Sở kế hoạch - đầu tư Hà nội

Người đại diện pháp luật: Lê Minh Thu

Chịu trách nhiệm nội dung: Trịnh Minh Cường

Thông tin chung

Thông tin trung tâm

Giảng viên

Quy định

Hướng dẫn mua khóa học

Hoàn trả - Ưu đãi học phí

Chính sách bảo vệ thông tin khách hàng

Contact

Mr. Cường: 090 220 9011

cuong@techmaster.vn

Ms. Huyền : 038 309 7229

huyen@techmaster.vn

Ms. Mai Anh: 096 247 1397

maianh@techmaster.vn

Địa chỉ

Số 14, ngõ 4, Nguyễn Đình Chiểu, Hai Bà Trưng, Hà Nội

Giờ mở cửa: **từ 9:30 - 18:00**

46 Tố Hữu, Trung Văn, Nam Từ Liêm, Hà Nội

Giờ mở cửa: **từ 9:30 - 18:00**

TTTM V+ Hòa Bình, 505 Minh Khai, quận Hai Bà Trưng, Hà Nội (Y-NEST CO-WORKING SPACE)

Giờ mở cửa: **từ 9:30 - 18:00**