**GIT**

1. **Sự khác nhau giữa Git, Github, Gitlab**

**Nguyễn Hoài Nam Update ab demoedfsfsdfsdf**

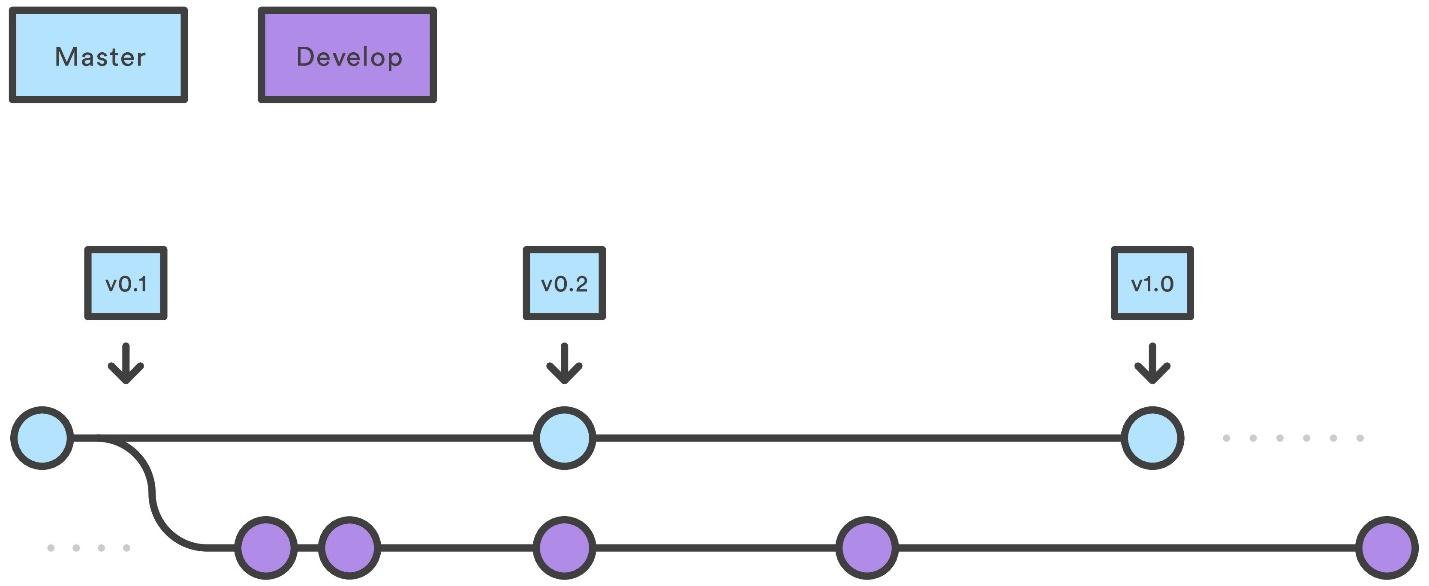
* **Git:** Phần mềm xử lý phiên bản mã nguồn, cho phép thực hiện và theo dõi các thay đổi tệp cục bộ và chia sẻ các thay đổi với kho lưu trữ từ xa.
* **Github:** Một dịch vụ đám mây để lưu trữ từ xa các kho git. Ngoài việc lưu trữ code, trang web còn giúp quản lý các dự án phát triển phần mềm với các tính năng như theo dõi issue, cộng tác với những người dùng GitHub khác và lưu trữ các trang web. GitHub cung cấp các dịch vụ miễn phí cho các dự án nguồn mở (có thể truy cập công khai) và các mức phí khác nhau cho các dự án tư nhân. Đối với các dự án chung, bất kỳ ai cũng có thể thấy code bạn đẩy lên GitHub và đưa ra đề xuất, hoặc thậm chí code, cải tiến dự án.
* **GitLab:** là một phần mềm có nhiệm vụ quản lý kho code Git. Gitlab sở hữu các tính năng đơn giản, góp phần lớn trong việc giúp các doanh nghiệp, cá nhân, tổ chức lưu trữ code một cách nhanh chóng, người dùng hoàn toàn có thể truy cập mọi lúc mọi nơi có kết nối Internet. Với Gitlab bạn có thể ẩn kho code của mình, không công khai chúng cho bất kỳ ai.

1. **Git Work-flow:**

* Gitflow là một dạng quy trình làm việc với Git đặt ra những quy ước trong việc tổ chức các branch trên Git

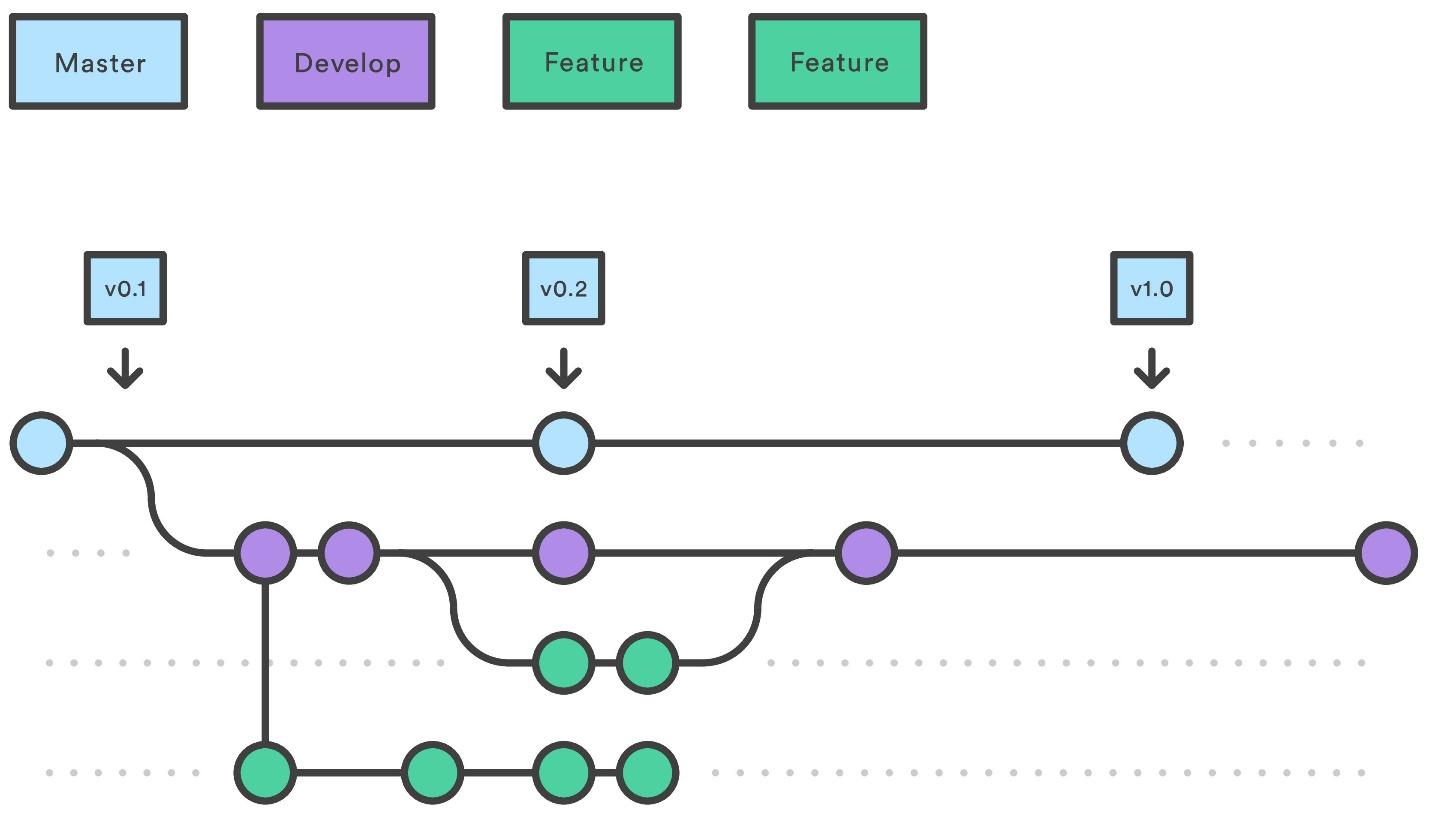


* Gitflow được thiết kế cho các dự án phần mềm theo kiểu version release. Đây là quy trình làm việc, nó sẽ không thêm các tính năng mới cho Git, nó chỉ định vai trò của mỗi **branch**, tại sao, khi nào các branch sẽ tương tác qua lại với nhau.
* **Mục đích của từng branch:**
* **Develop**



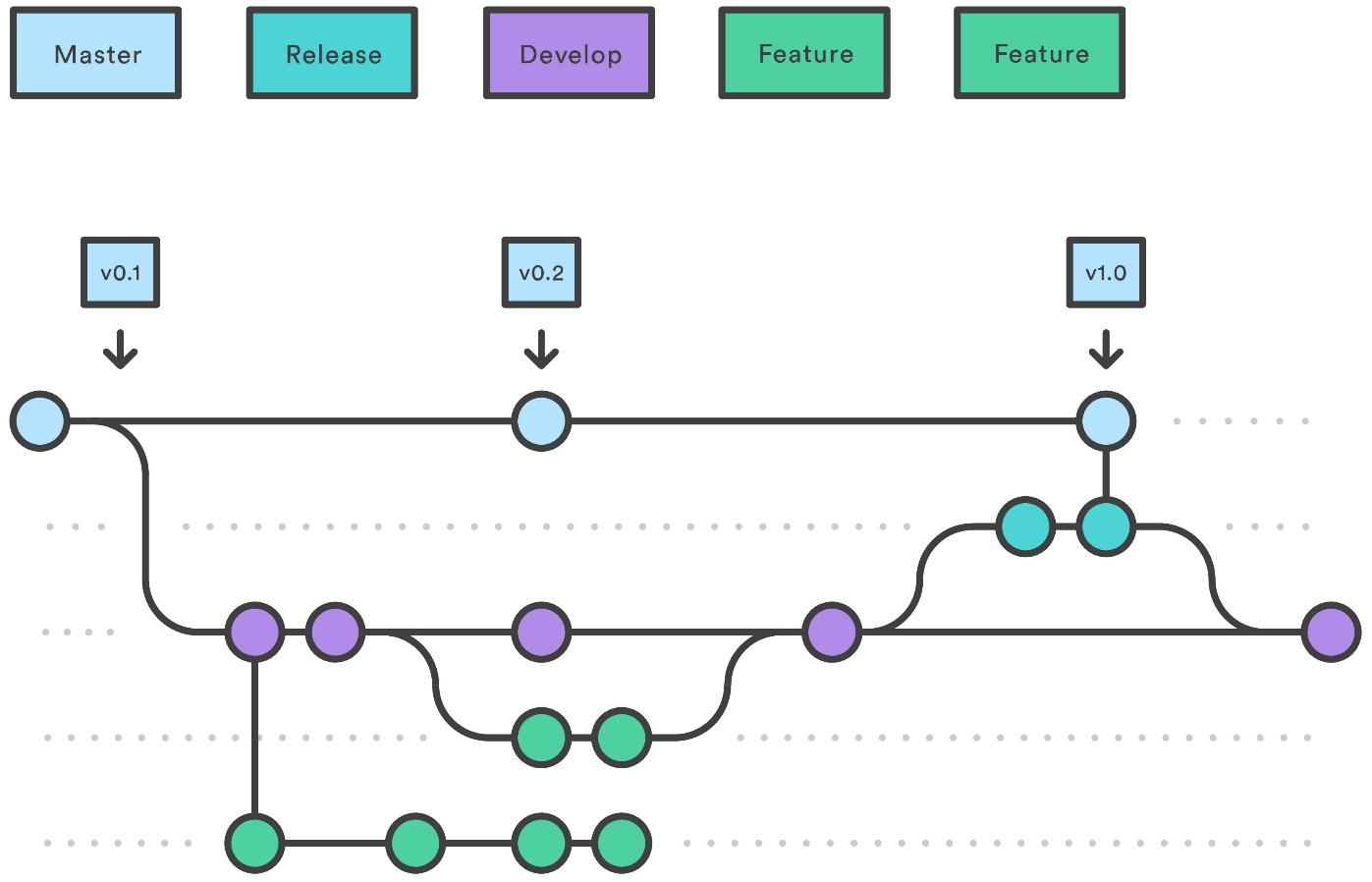
Nếu **master** để lưu trữ lịch sử những bản release chính thức, **develop** mục đích như một branch cho việc tích hợp các **feature** branch. Trên **master** branch chúng ta sẽ thêm tag theo từng version release.

* **Feature Branch**



Mỗi feature mới sẽ có branch riêng, tên branch sẽ được đặt theo feature đang phát triển, **Feature** branch xem **develop** branch như là branch cha của nó. Khi đã hoàn tất feature này, chúng ta *merge* nó lên trên **develop**, không bao giờ tương tác lên **master**, đồng thời xóa branch này.

* **Release Branch**

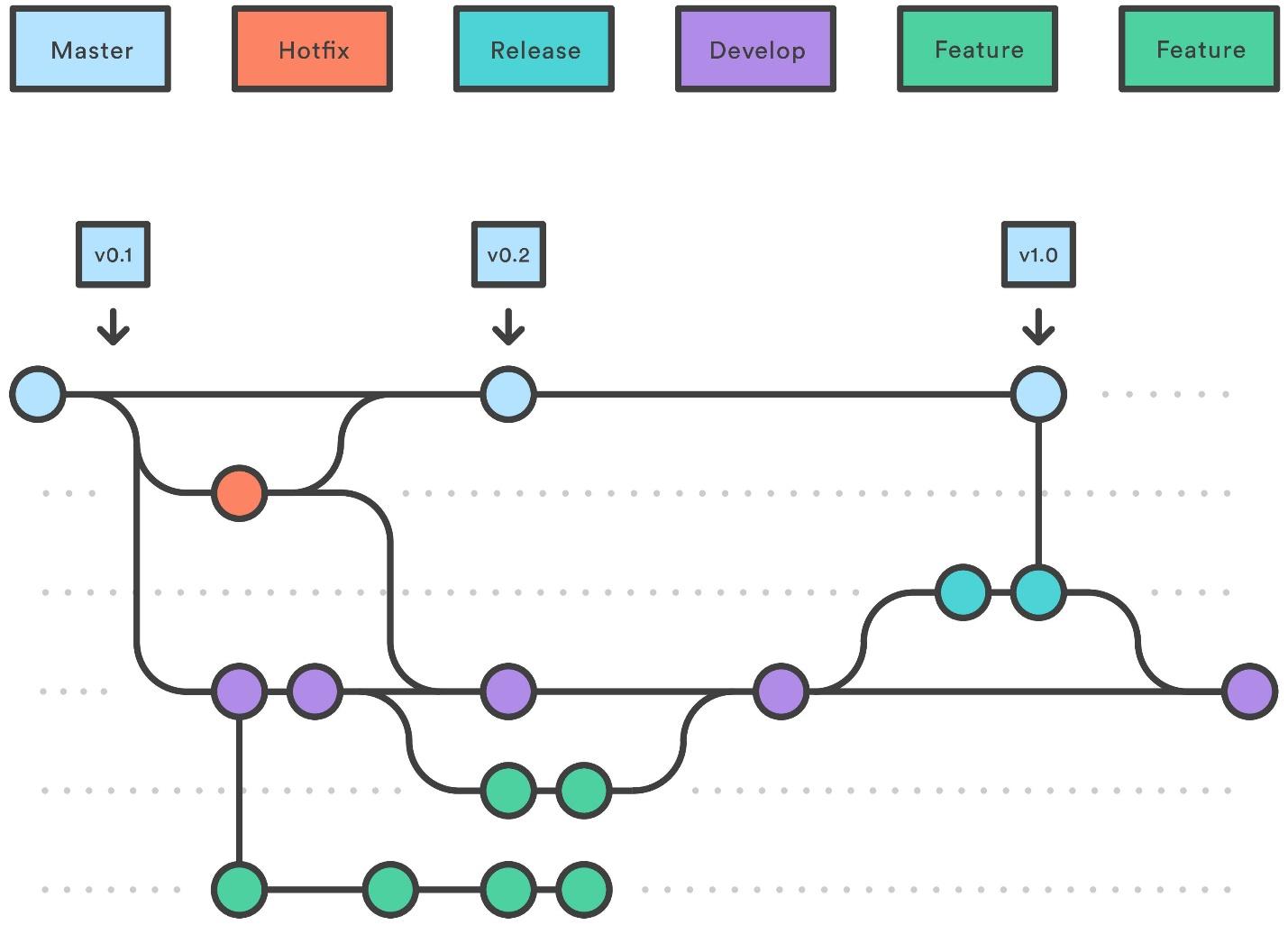


Khi tất cả feature đã hoàn thành để có thể release. Ta tạo thêm branch **release**, tên branch là tên của của release version. Tạo branch này cũng đồng nghĩa với việc bắt đầu một vòng phát triển mới, không thêm các tính năng mới nữa, mà tập trung vào bug fix, tạo document. Khi đã hoàn tất tiếp tục *merge* lên **master**

Bằng cách này, ta có thể tách một team để đẩy việc đưa ra release và một team khác tiếp tục phát triển tính năng cho release tiếp theo.

Sau khi được merge vào **master** và **develop**, có thể xóa branch **release**.

* **Hotfix**



Khá giống với **release** và **feature** trừ việc nó sẽ dựa trên **master** chứ không phải **develop**, để sửa các lỗi trên production. Một khi sửa xong, thì merge vào cả **master** và **develop** rồi xóa.

**3.Những lệnh cơ bản sử dụng trong GIT**

**-Thiết lập chứng thực cá nhân**

$ git config --global user.name "User Name"

$ git config --global user.email "username@gmail.com"

Lưu ý: --global được sử dụng để áp dụng cho tất cả các projects. Nếu ko sử dụng --global thì settings sẽ chỉ dùng cho riêng project đó.

### **-Tạo một kho chứa Git**

$ git init

Nếu như muốn theo dõi một dự án cũ trong Git, bạn cần ở trong thư mục của dự án đó. Lệnh này sẽ tạo một thư mục mới có tên .git, thư mục này chứa tất cả các tập tin cần thiết cho kho chứa.

### **-Clone một kho chứa đã tồn tại**

$ git clone https://github.com/user/repository.git

Câu lệnh trên sẽ tạo một thư mục mới có tên giống trên của repo.

### **-Nhánh trong git**

Khi sử dụng Git, bạn có thể tạo ra nhiều nhánh (branch) khác nhau. Câu lệnh Git này dùng để kiểm tra branch hiện tại:

$ git branch

**-Để tạo mới một branch:**

$ git branch <name\_branch>

Trước khi tạo mới một branch ta cần kiểm tra xem branch đó đã tồn tại chưa, có thể dùng lệnh:

$ git checkout <name\_branch>

Trường hợp branch đã tồn tại câu lệnh trên sẽ chuyển đến nhánh được chọn

**-Để chuyển và tạo mới:**

$ git branch -b <name\_branch>

### **-Cập nhật thay đổi**

Sau khi thay đổi source code: thêm mới, sửa, xoá files,… Cần phải cập nhật lên Staging Area. Để cập nhật hết các files:

$ git add .

Sau lệnh add sử dụng câu lệnh **Commit** để đẩy thông tin thay đổi lên Local Respository:

$ git commit -m "Message"

### **-Cập nhật lên server**

Sau câu lệnh Commit, thông tin mới chỉ được cập nhật lên Local Repository. Nếu muốn cập nhật lên server thì phải sử dụng câu lệnh **push**:

$ git push origin <name\_branch>

Ngoài ra, nếu chưa tồn tại **remote** trên server thì bạn cần phải add mới một remote trước rồi mới **push**:

$ git remote add origin <remote\_url>

$ git push origin <name\_branch>

### **-Gộp nhánh**

Sau một thời gian cập nhật các file và push lên git trên branch mới, bây giờ cần ghép (merge) code lại vào nhánh gốc (master). Trước tiên, cần phải checkout ra khỏi branch hiện tại cần gộp để vào branch master, sau đó thì dùng lệnh merge để ghép branch mới vào master:

$ git checkout <branch>

$ git merge <new\_branch>

### **-Xem lại lịch sử commit**

$ git log

Lệnh git log sẽ cho biết về người commit, ngày giờ, message của những lần commit đó.

### **-Xem thay đổi trước khi push**

$ git diff

Lệnh này để biết những gì đã được thay đổi giữa nhánh hiện tại và nhánh trước nó.

### **-Gộp commit**

$ git rebase -i HEAD~

Sau dấu ~ là số commit muốn gộp. Sau khi gõ lệnh này một cửa sổ trình soạn thảo hiện ra. Thay đổi ký tự pick của dòng các dòng sau dòng đầu thành s rồi lưu lại/kết thúc. Khi đó, trình soạn thảo để chỉnh sửa giải thích commit thiết lập cho commit sau khi đã tổng hợp sẽ được hiển thị, nên hãy chỉnh sửa lưu lại/kết thúc.

### **-Pull từ remote repository**

$ git pull origin master

Lệnh trên sẽ gộp những thay đổi mới kéo về từ máy chủ từ xa với nhánh hiện tại trên máy local.