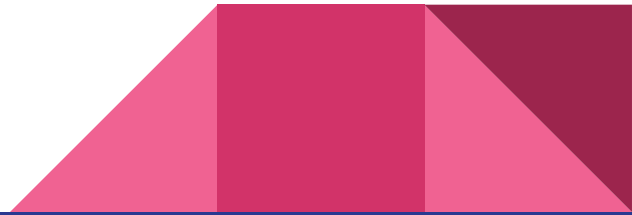
# Git workflow

Hoàng-Nguyên Vũ

# Table of Contents





# 1. Git basic concepts

- **Git provider**: Github, Gitlab, Bitbucket...

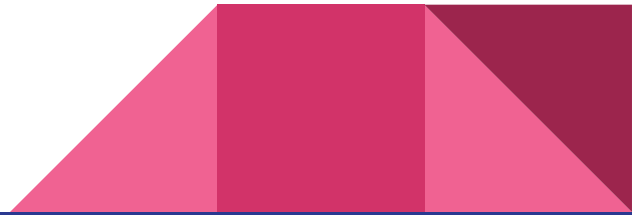


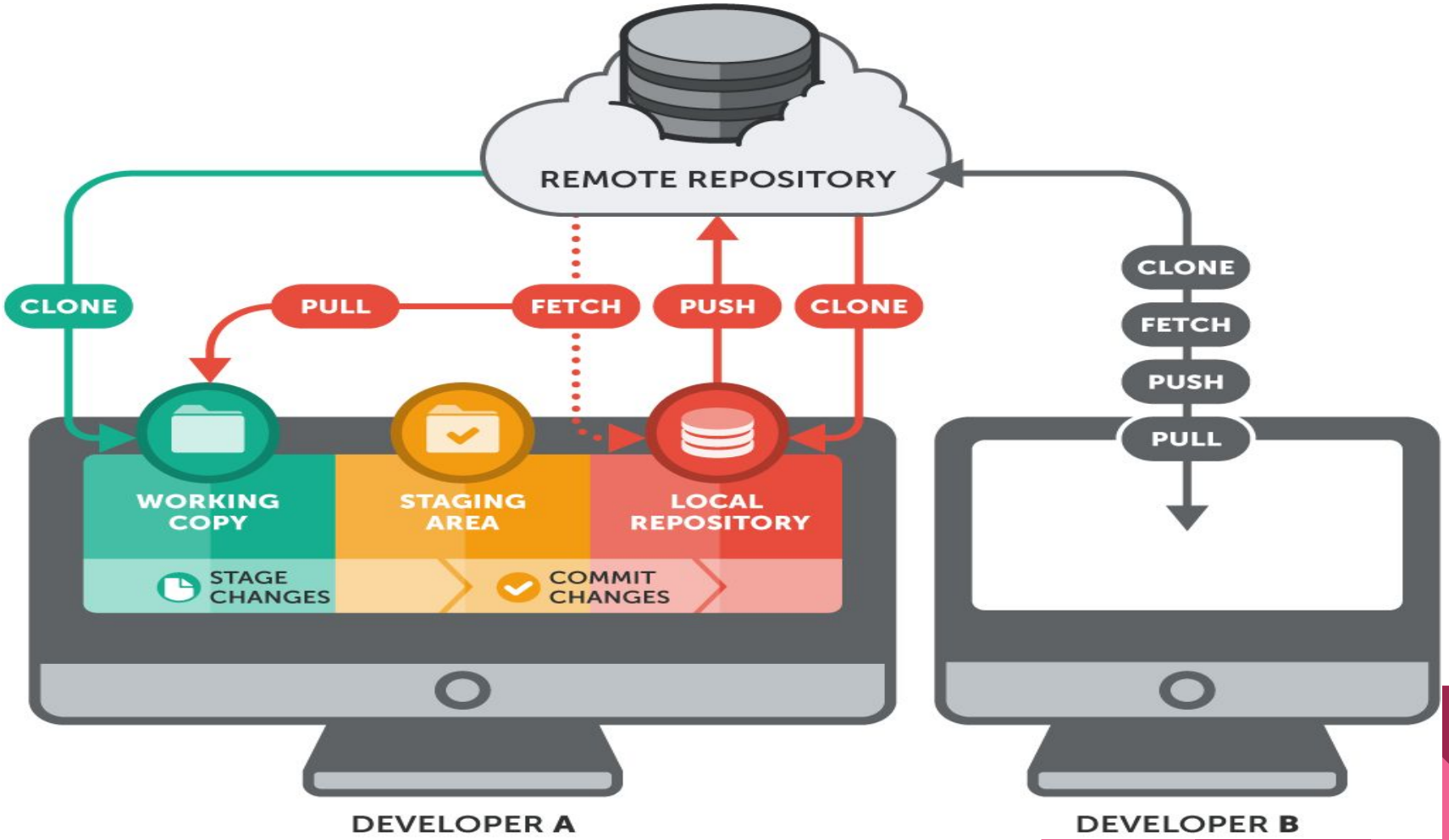

# 1. Git basic concepts

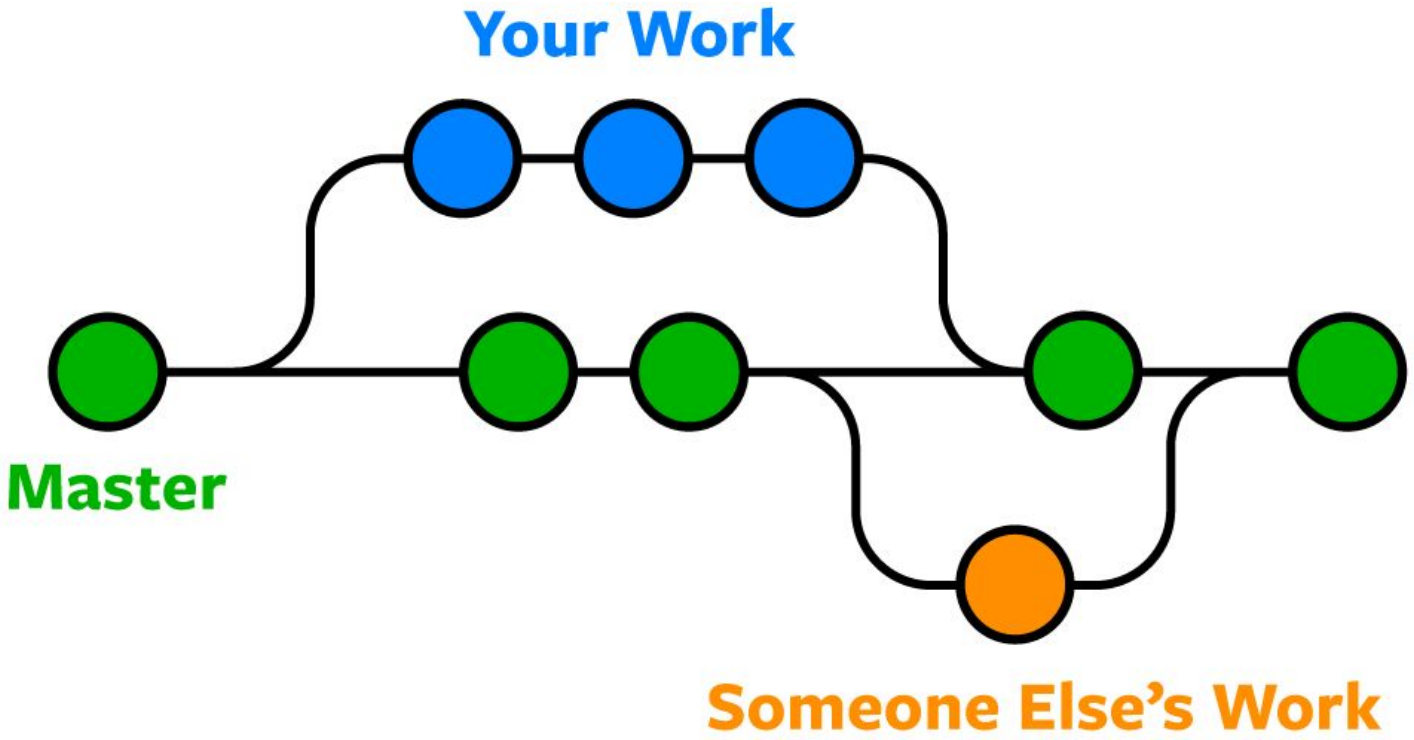
- **Git provider**: Github, Gitlab, Bitbucket...

- **Repository**: Kho lưu trữ nơi chứa các files của dự án. Các file đó có thể là code, hình ảnh, âm thanh hoặc mọi thứ liên quan đến dự án.

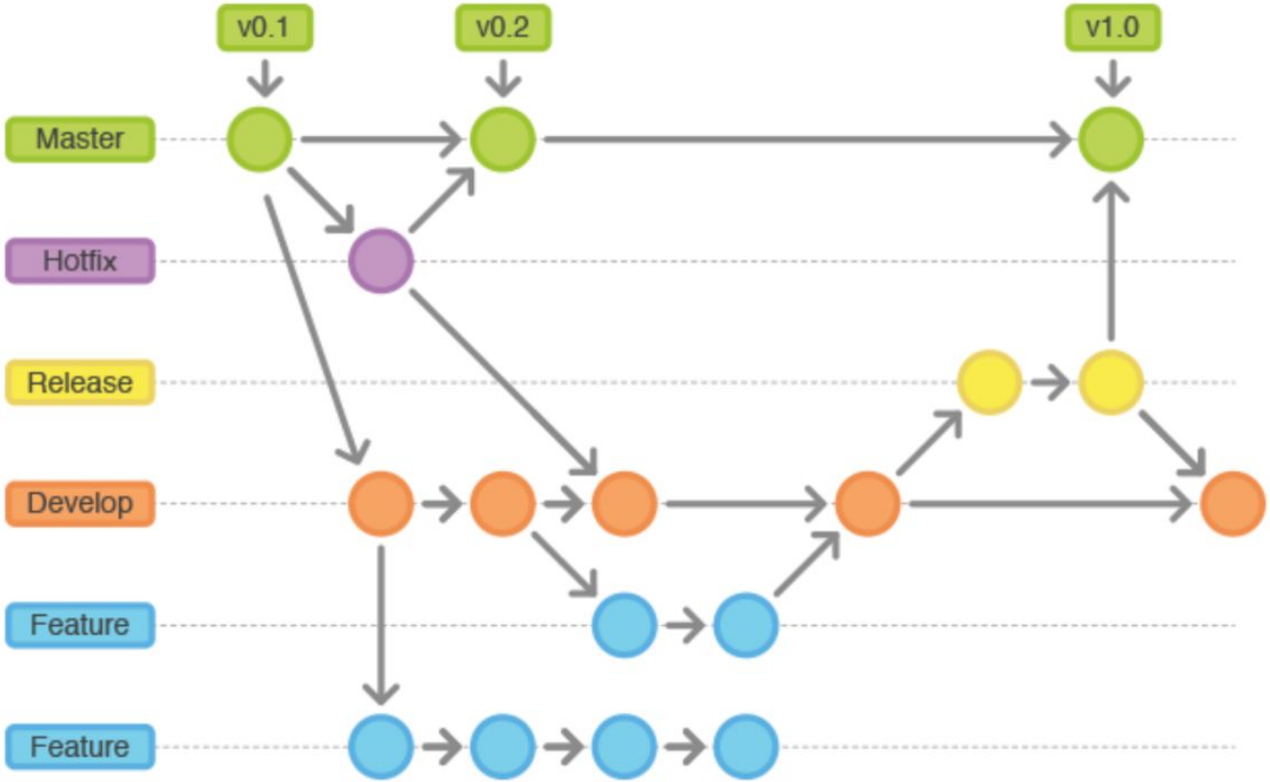
- **Branch**: Dùng để phân nhánh, mà sự thay đổi trên nhánh không ảnh hưởng đến nhánh chính của dự án.




REMOTE REPOSITORY
CLONE
PULL
FETCH
PUSH
CLONE
WORKING COPY
STAGING AREA
LOCAL REPOSITORY
STAGE CHANGES
COMMIT CHANGES
DEVELOPER A
CLONE
FETCH
PUSH
PULL
DEVELOPER B

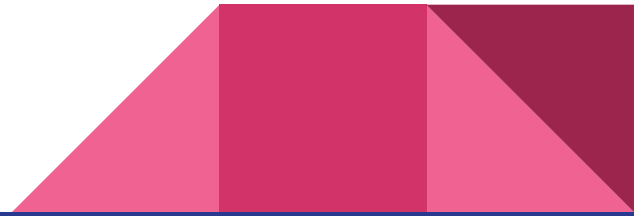

Your Work
Master
Someone Else's Work

# 2. Git workflow



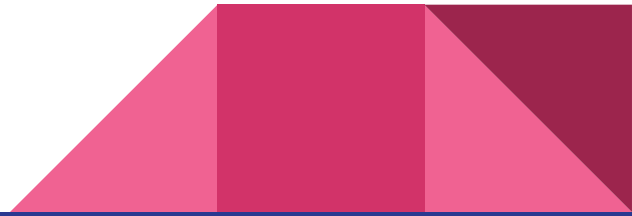
# Lợi ích của Gitflow

- Dù có bao nhiêu developer cùng thời điểm phát triển thì vẫn có thể quản lý branch dễ dàng.
- Bằng cách đặt trước ra các branch và một số rule nhất định khi merge code

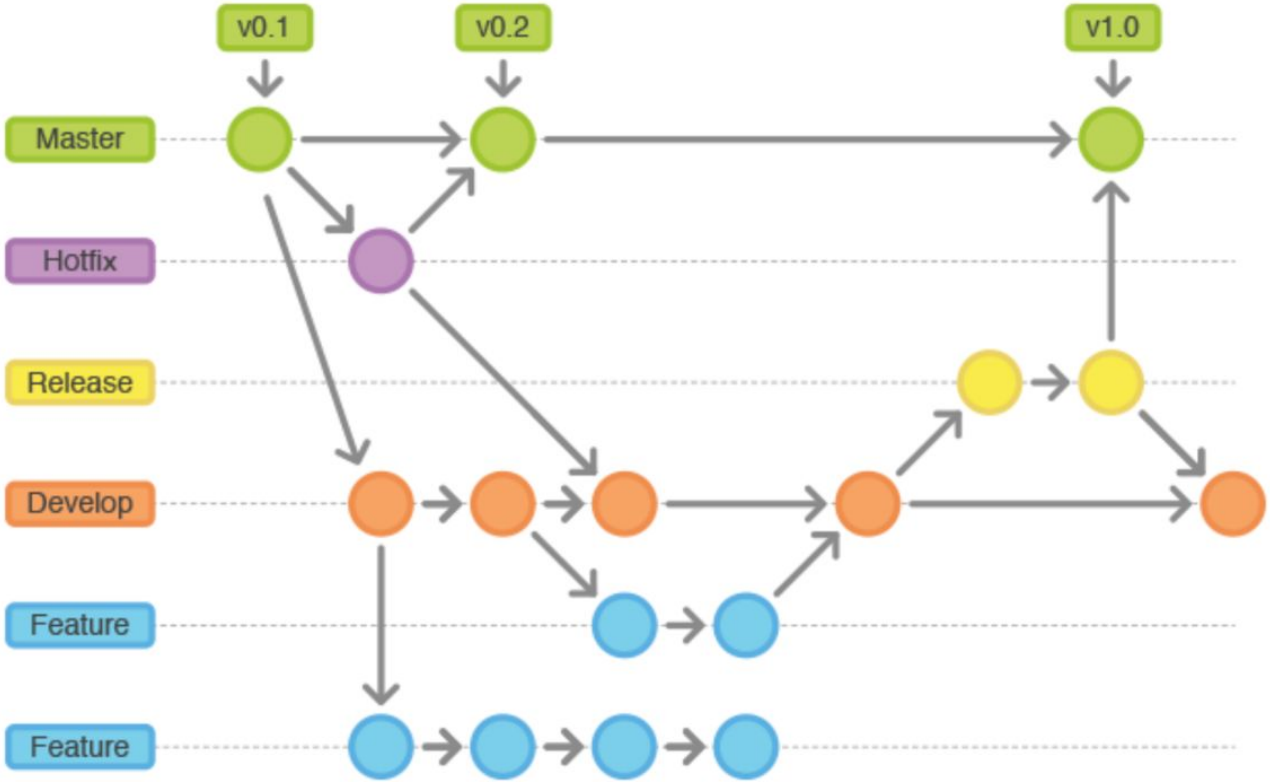


# 2.1 Master branch

- Branch **“master”** là branch chính của repository, luôn chứa source code đang sử dụng trên production.
- Không commit các thay đổi trực tiếp trên branch này.
- Mọi thay đổi source đều diễn ra ở branch khác và được merge ngược lại vào “master”.
- Trên remote repository, chỉ những người nhất định mới được phép merge/commit vào “master”.

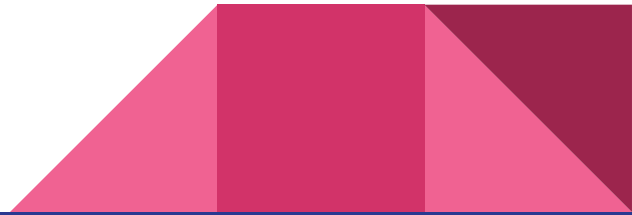


# 2.1 Master branch

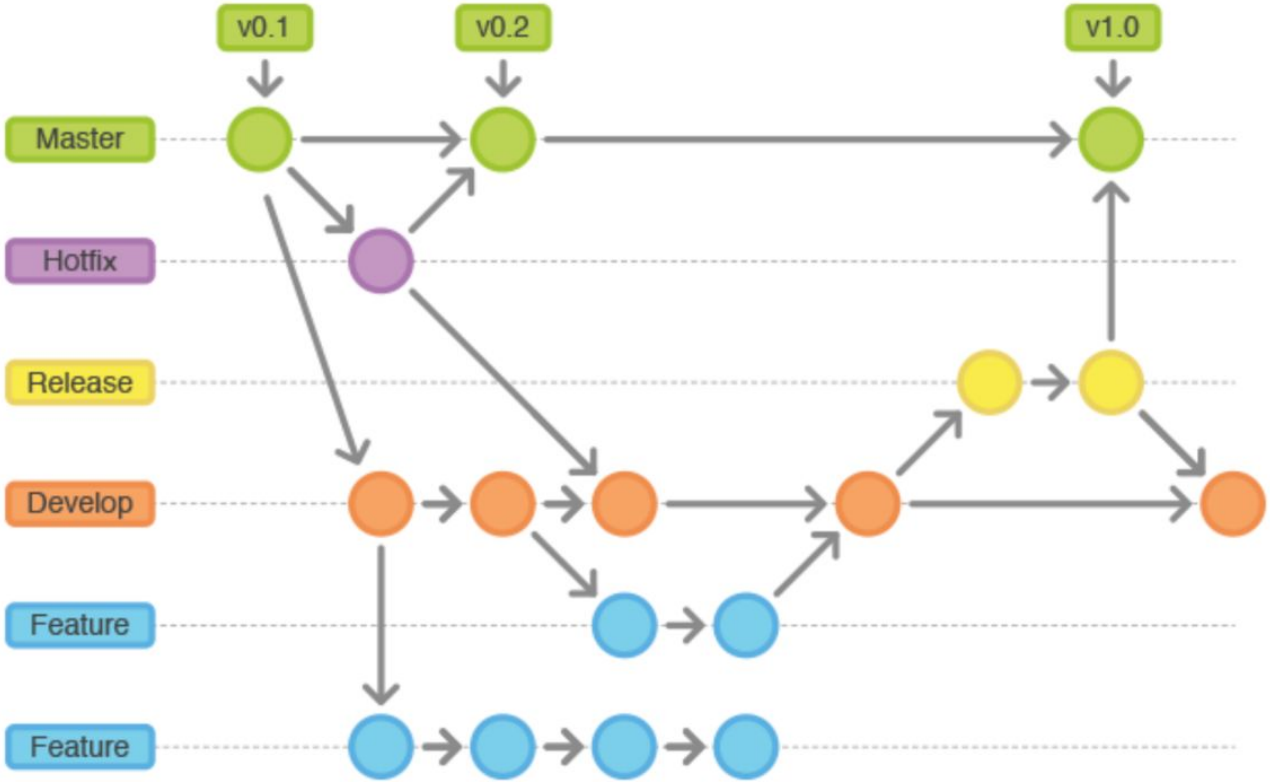


# 2.2 Develop branch

- Branch **“develop”** là branch chính của repository, luôn chứa source code mới nhất đang được phát triển cho lần release tiếp theo.
- Sau các lần thay đổi, “develop” sẽ được merge ngược trở lại vào “master” để release lên production.
- Trên remote repository, chỉ những người nhất định mới được phép merge/commit vào “develop”.

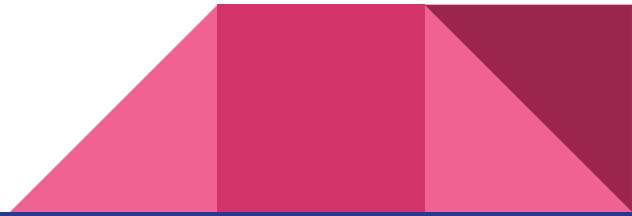


# 2.2 Develop branch

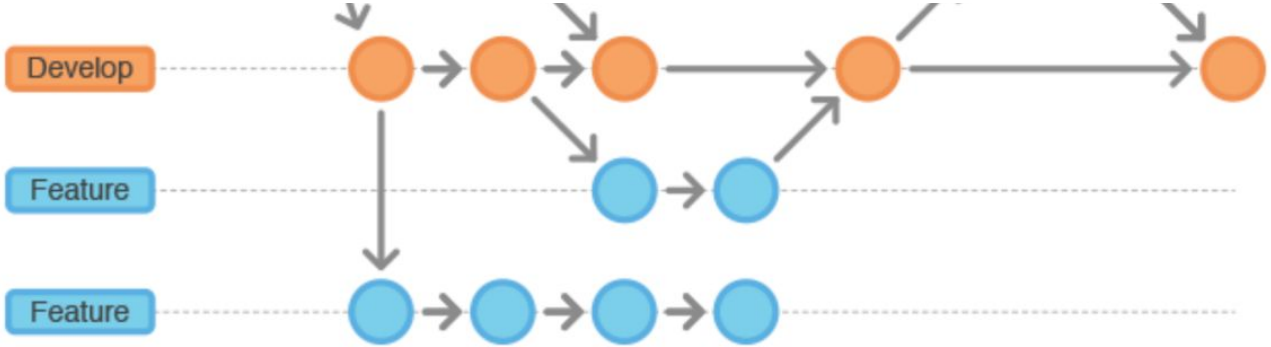


# 2.3 Other branches

- Feature branches
- Hotfix branches
- Release branches

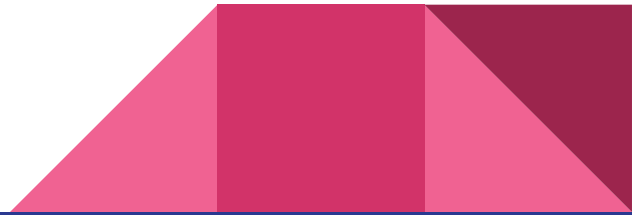


# 2.3.1 Feature branches

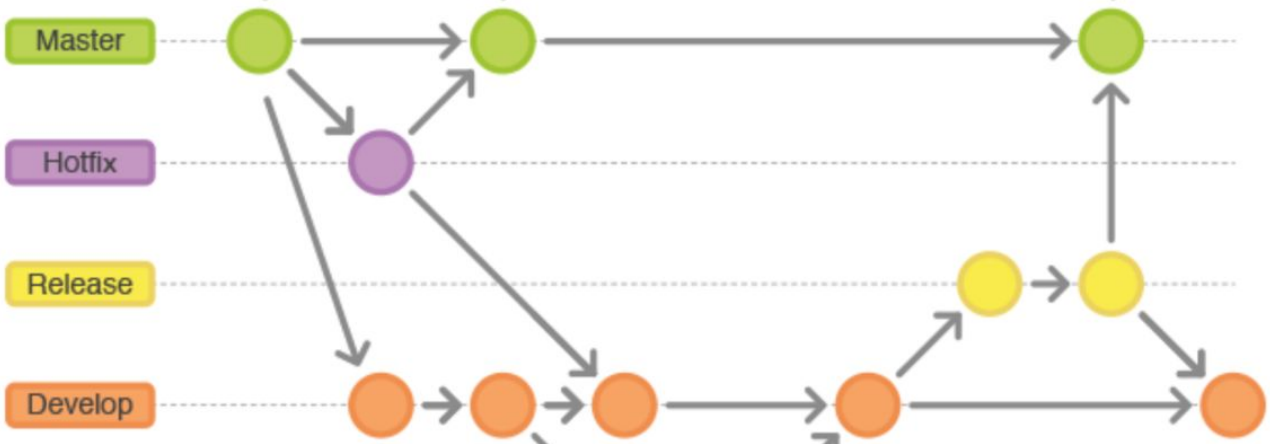


# 2.3.1 Feature branches

- Sử dụng để merge code phát triển cho một chức năng riêng biệt
- Được checkout từ nhánh **“develop”**
- Khi phát triển thì cũng cần phải checkout từ nhánh “feature” thay vì “develop”
- Khi chức năng có thể release thì sẽ merge “feature” vào lại “develop”
- Chỉ người có quyền thì cũng mới có thể merge code vào feature branch, xóa branch.
- Có thể đặt tên theo quy ước **“feature-*”**

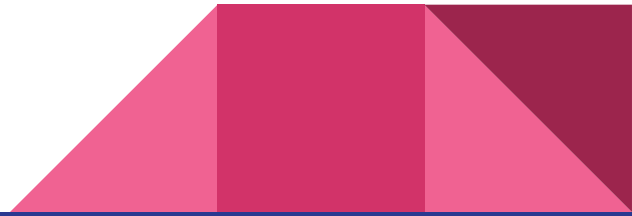


# 2.3.2 Hotfix branches

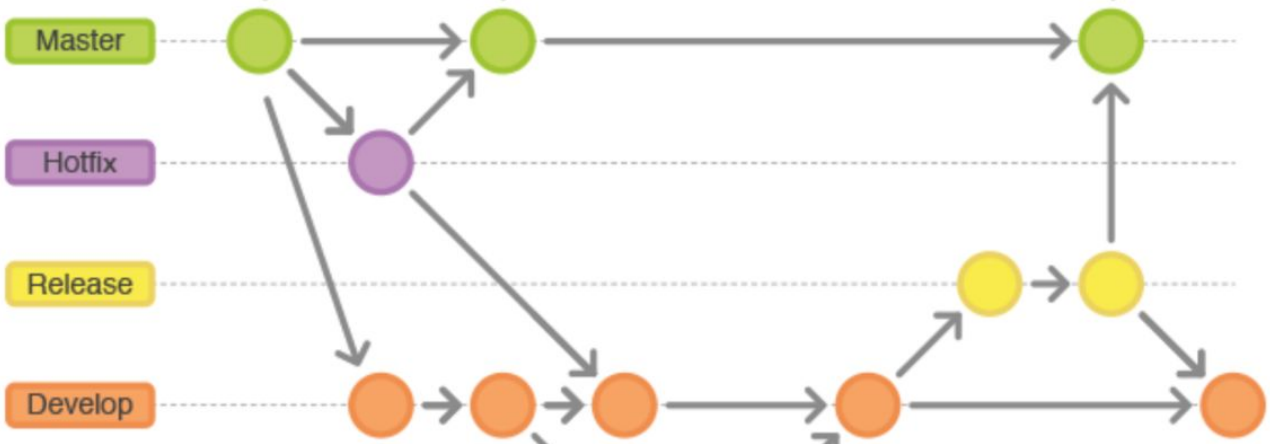


# 2.3.2 Hotfix branches

- Khi cần fix gấp một lỗi nào đó trên production
- Được checkout từ nhánh **“master”**
- Branch này cần được merge và cả “master” lẫn “develop”
- Đặt tên theo quy ước **“hotfix-*”**



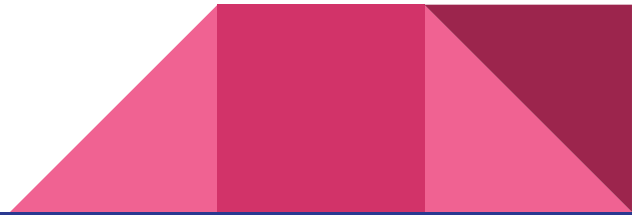
# 2.3.3 Release branches



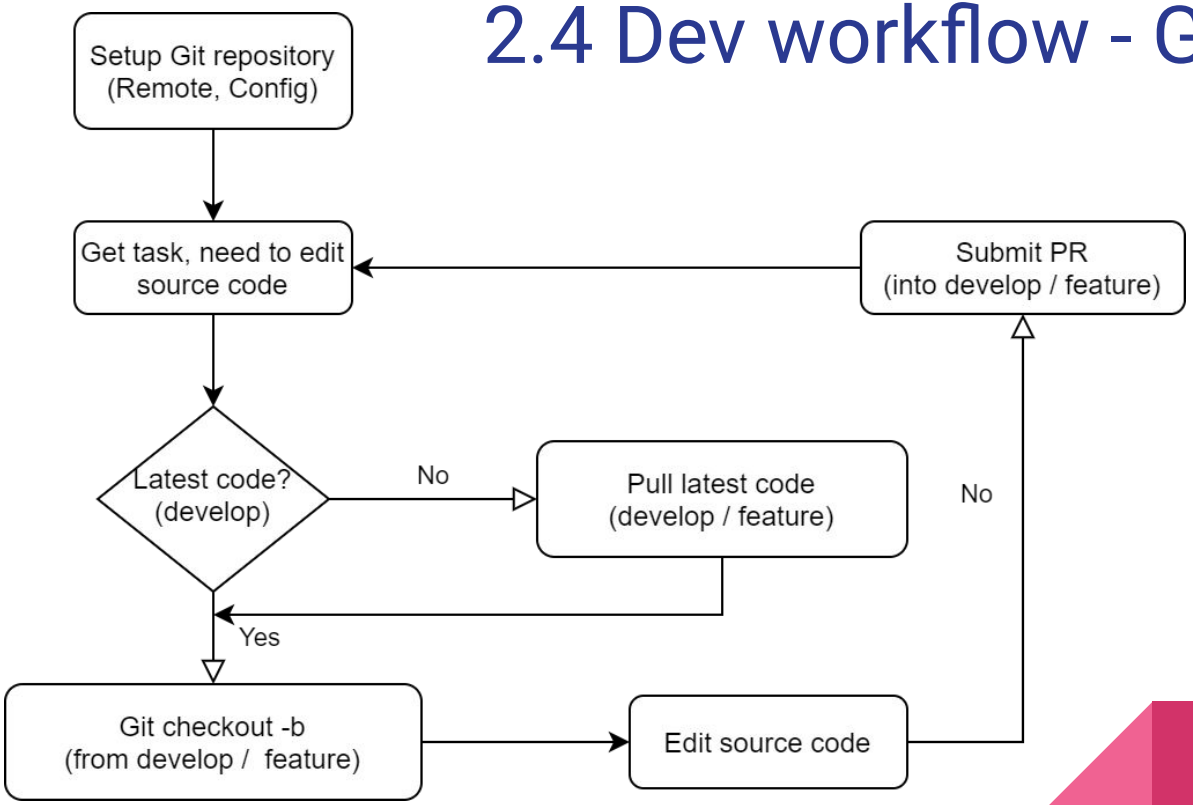
# 2.3.3 Release branches

- Sử dụng để chuẩn bị cho bản release mới lên prodution
- Được tách ra từ nhánh “**develop**”. Sau khi tách thì có thể sẽ có thể sẽ có thêm các commit fix lỗi.
- Cần được merge ngược lại vào cả “develop” và “master”.
- Tên branch release nên đặt theo quy ước **“release-*”**

  VD: release-20200916, release-1.1.0

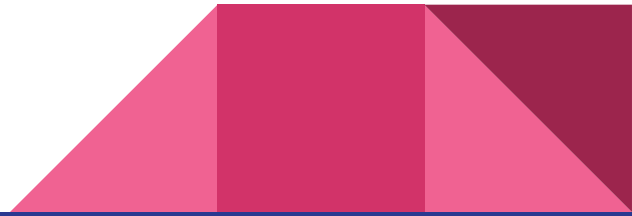


# 2.4 Dev workflow - Guideline



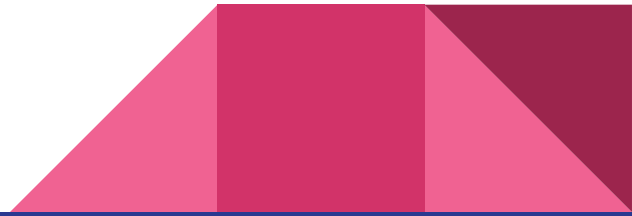
# 3. Gitflow - Các vấn đề thường gặp

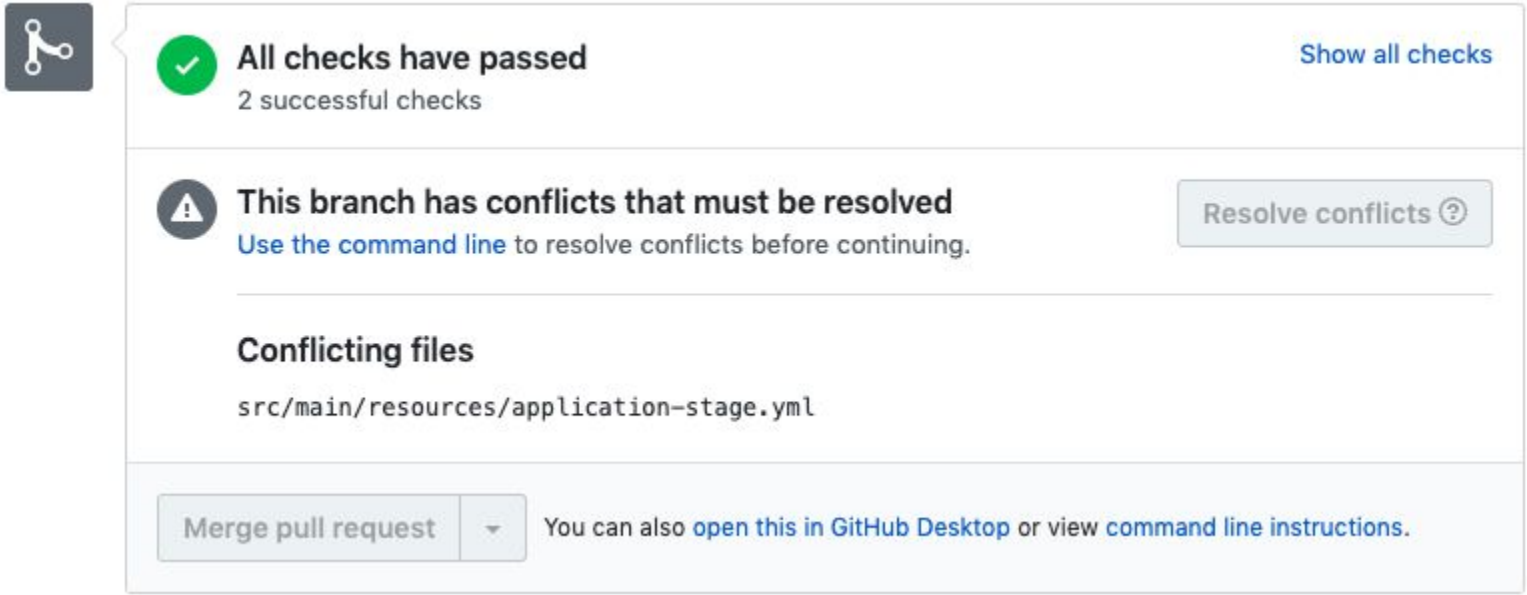
- Conflict khi tạo submit Pull Request
- Khi bắt đầu sửa code, ***checkout -b*** nhầm branch
- Sử dụng **git push -f** lên remote repo
- Sử dụng **git rebase** trên **public branch**
- Revert một commit không mong muốn
- Commit message không đem lại nhiều ý nghĩa



# 3.1 Conflict

- Conflict là trạng thái code bị xung đột khi sửa
- Do sửa 2 commit cùng sửa một dòng code, mỗi commit lại sửa một cách khác nhau




All checks have passed
2 successful checks
Show all checks
This branch has conflicts that must be resolved
Use the command line to resolve conflicts before continuing.
Resolve conflicts
Conflicting files
src/main/resources/application-stage.yml
Merge pull request
You can also open this in GitHub Desktop or view command line instructions.

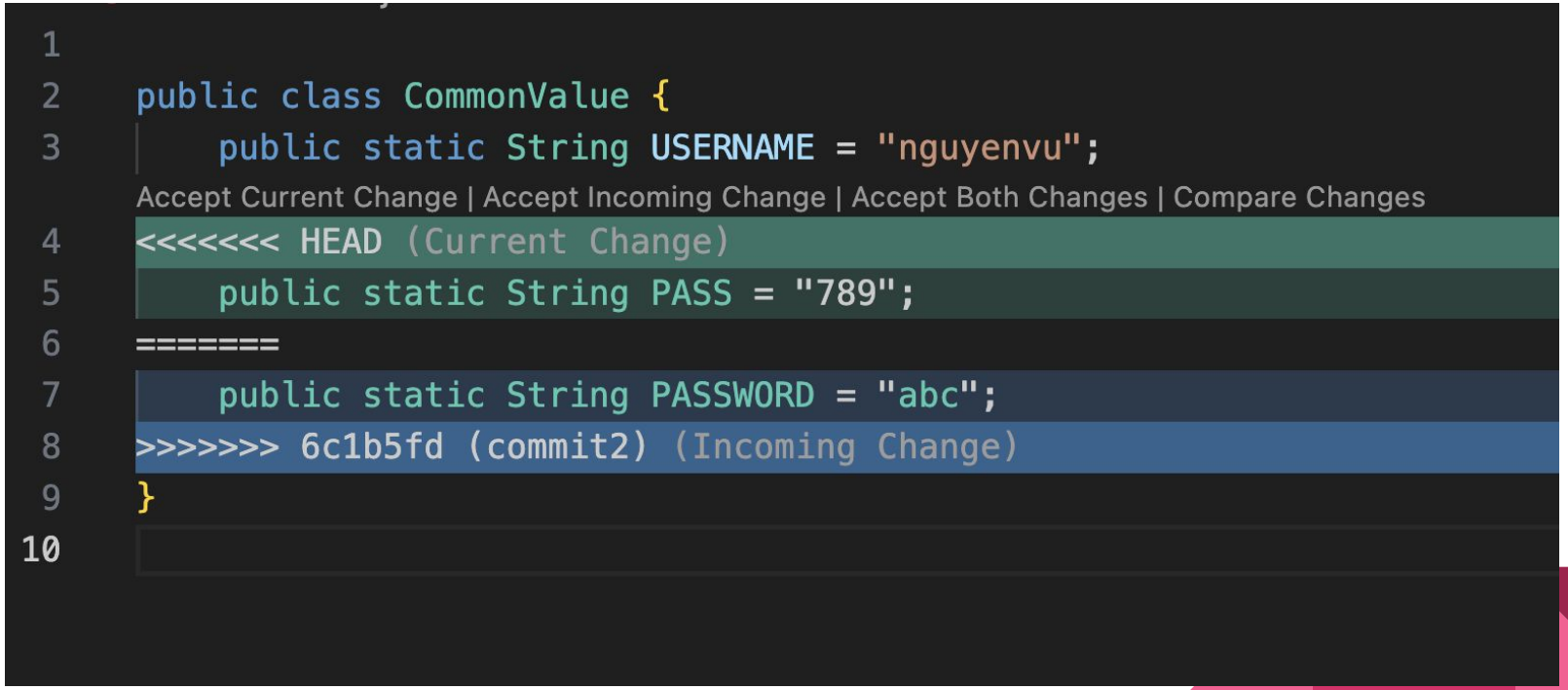
# Xử lý conflict

- Khi bị báo conflict, chúng ta chỉ cần **rebase** working branch về target branch
- VD:  tạo request merge từ nhánh mình đang làm -> develop => *có conflict*

1. Working branch đang là ***feature/OTM-456***, update code mới nhất của develop
2. Rebase develop: ***git rebase develop***
3. Quan sát vị trí bị conflict và tiến hành sửa. Ở vị trí conflict Git sẽ chia làm 2 phần trên / dưới.
    a. Trên là code hiện tại ở **target branch**
    b. Dưới là code mình muốn sửa đổi thành

# Xử lý conflict (tiếp)

# Xử lý conflict (tiếp)



# Xử lý conflict (tiếp)

4. **git rebase --continue** (để hoàn tất nếu sửa hết code bị conflict)

Hoặc **git rebase --abort** (để tạm thời hủy quá trình rebase, nếu không sửa được hết conflict)

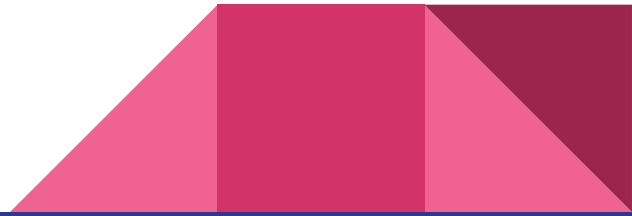
5. Git push => Done

# 3.4 Revert commit

- Mình có một commit A không mong muốn trên **“master”**
- Mình muốn hủy commit A, dùng **git revert <commit_id>**

**VD: git revert A**

git revert A -m 1 *(Với TH A là merge commit)*



# 4. Commit Conventional

```
<type>[optional scope]: <description>

[optional body]

[optional footer]
```

- **< >**: Bắt buộc phải có trong commit message
- **[ ]**: Tùy chọn, không bắt buộc phải có

# Rút gọn

```
<type>[optional scope]: <description>

feat(organization): add feature A
```

- type: viết thường - fix, feat, docs...
- scope: là một danh từ, để phân loại nhóm các thay đổi của các commit
- description: một câu mô tả ngắn gọn về các sự thay đổi trong commit

# Lợi ích - Commit Conventional

- Bộ quy tắc chung đơn giản, dễ áp dụng
- Dễ truyền đạt nội dung đã sửa đổi
- Tương thích với Semantic Versioning
- Tự động tạo CHANGELOGS
- Tự động xác định version mới dựa trên commits
- Mọi người dễ contribute hơn: tạo commit, tìm đọc lại lịch sử commit

# 5. Các lỗi thường gặp

1. Đặt tên nhánh không có ý nghĩa.
2. Quá nhiều commit rác trên 1 nhánh feature và message commit không có ý nghĩa.
3. Copy source code/cả project nơi khác vào git nhưng không hiểu ý nghĩa.

# Đặt tên nhánh không có ý nghĩa.

Các nhánh feature sau khi tạo từ develop, phải đặt theo quy định như sau: feature/tên chức năng hoặc feature/mã bài tập ….

Ví dụ:

✅ **Đúng:** feature/homework-module1-week1, feature/prepare-dataset

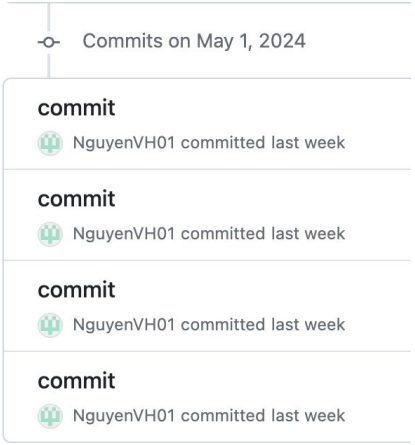
🚫 **Sai:** feature

(*) Trường hợp bạn làm sai, nếu nhắc 3 lần không sửa xem như không đạt yêu cầu và điểm tiêu chí git workflow là 0 điểm cho tuần đó.

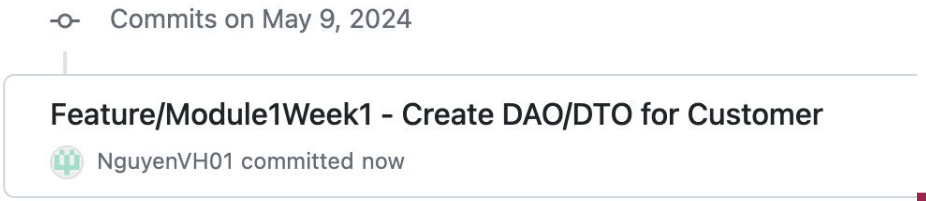
# Quá nhiều commit rác trên 1 nhánh feature.

Trong quá trình làm có thể bạn sẽ commit rất nhiều trên máy, nhưng khi push toàn bộ commit, yêu cầu bạn gom tất cả commit thành 1 trước khi push lên git cho nhánh feature đó. Đồng thời, message commit phải có ý nghĩa rõ ràng, mô tả ngắn gọn commit đó làm gì, thay đổi những gì.

Ví dụ: 🚫 **Sai:**




✅ **Đúng:**



# Copy source code/cả project nơi khác vào git nhưng không hiểu ý nghĩa.

- Trong quá trình làm project AIO hoặc project mở rộng, các bạn được quyền copy hoặc tham khảo bất kỳ nguồn nào trên mạng: Youtube, GitHub người khác, Coursera, … Nhưng nếu đã lên git của các bạn thì bạn phải thực sự hiểu đoạn code đó, hoặc cả 1 project copy đó chạy như thế nào, supporter/leader có quyền từ chối chấm điểm nếu có dấu hiệu nghi ngờ copy khi hỏi về đoạn code đó hoặc file đó hoặc giải thích ý nghĩa trong chính project các bạn nhưng bạn không trả lời được.

# Thank you