**GIT**

# I/ Thiết lập Git

## Thiết lập username và email

**$** git config --global user.name "PhamDat "

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

## Xem lại thiết lập

**$** cat ~/.gitconfig

**$** git config --list // Hoặc sử dụng lệnh này

## Hiện thi thông báo khi push lên Github đơn giản hơn

**$** git config --global push.default simple

# II/ Khởi tạo Local repository

* Đi đến thư mục cần tạo repo

**$** git init tên\_folder // Tạo folder và file .git

**$** git init // Chỉ tạo file .git

* Đưa các file mã nguồn về trạng thái tracked và đưa vào Staging Area

**$** git add tên\_file

**$** git add . // Tracked toàn bộ file trong folder

**$** git add \* // Tracked toàn bộ file trong folder

* Kiểm tra các tệp đã được tracked chưa *(Có thể bỏ qua)*

**$** git status

* Sau đó đưa về Staging Area bằng lệnh
* Lệnh commit

**$** git commit -m "Lời nhắn"

**$** git commit -a -m "Skipped Staging Are to commit" // Bỏ qua việc đưa file vào Staging và

Commit luôn

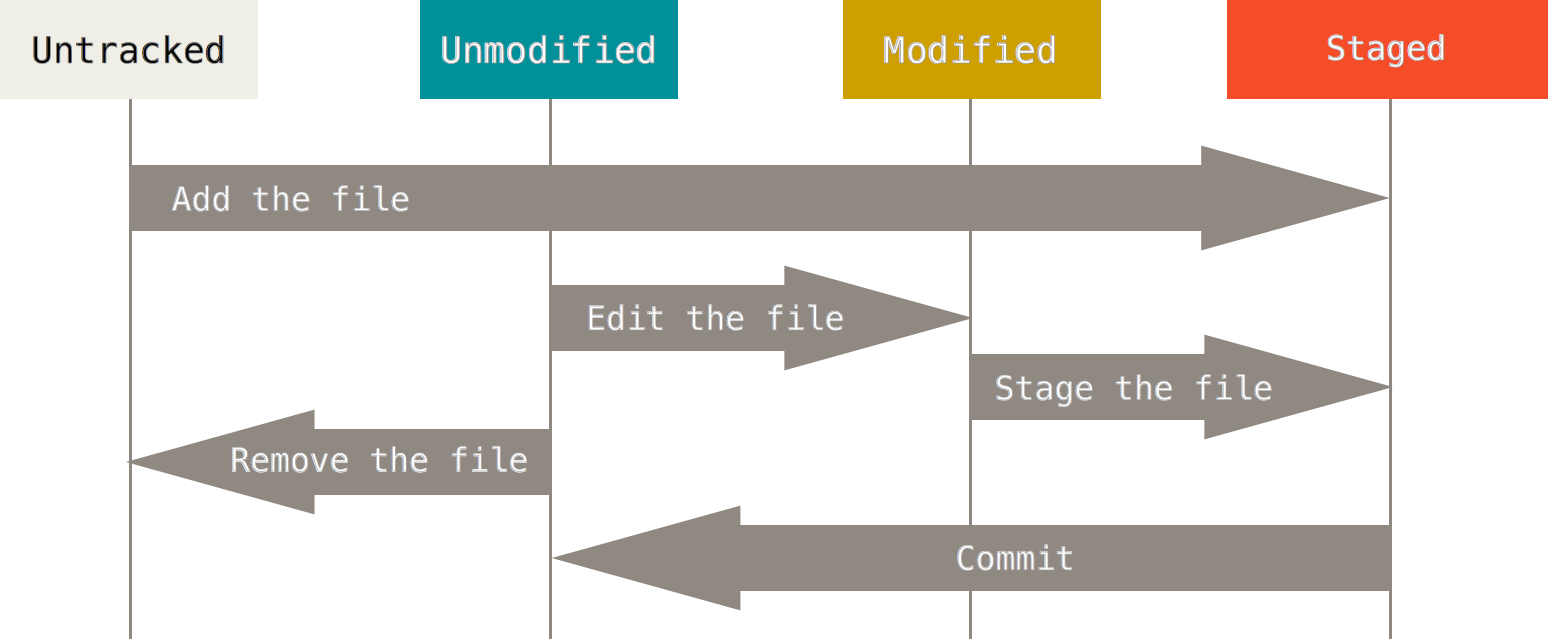
# III/ Git & Github

* Khởi tạo repo trên github
* Git clone repo (Download repo về local)

**$** git clone link\_git

# IV/ Các trạng thái trong Git





# V/ Các lệnh trong Git

## Chuyển đổi trạng thái trong Git

**$** rm tên\_file //Chuyển file từ **tracked về untracked**

**$** rm –f tên\_file // **Xóa file trong ổ và đưa về untracked**

**$** git reset HEAD tên\_file // Đưa tệp **ra khỏi Staging Area**

**$** git checkout --tên\_file // Xóa bỏ những thay đổi của file

## Log trong Git

**$** git log

**$** git log –p // Xem chi tiết lịch sử commit

* Xem 1 lần commit gần nhất **thì thêm tham số -1 vào.**
* Một số tùy chọn xem log sau để tối ưu hơn quy trình đọc log.
* **--since, --after:** Xem các lần commit kể từ ngày nhất định.
* **--until:** Xem các lần commit trước từ ngày nhất định.
* **--author:** Xem các lần commit của một người nào đó.
* **--grep:** Lọc các chuỗi trong log và in ra.
* **Thiết lập log đẹp mắt hơn:**

**$** git config --global alias.lg "log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-commit"

* **git lg** để xem log
* **Lọc log với --pretty**

Tham số **--pretty** sử dụng khi muốn **lọc 1 thuộc tính** nào đó trong lịch sử commit. ***VD:*** Chỉ xem *lời nhắn commit* hoặc *chỉ xem email* của người commit.

**$** git log –pretty=”%tag” // Các tag nối tiếp nhau bởi kí tự **-**

* Các **%tag**phải dược đặt trong cặp dấu ngoặc kép và bạn có thể sử dụng nhiều **%tag** khác nhau.
* **%H –**  Commit hash
* **%h –** Abbreviated commit hash
* **%T –** Tree hash
* **%t –**Abbreviated tree hash
* **%P –** Parent hashes
* **%p –** Abbreviated parent hashes
* **%an**– Author name
* **%ae**– Author e-mail
* **%ad**– Author date (format respects the –date=option)
* **%ar –** Author date, relative
* **%cn**– Committer name
* **%ce**– Committer email
* **%cd**– Committer date
* **%cr –** Committer date, relative
* **%s** – Subject

## Commit trong Git

**$** git commit -m “Fisrt commit” // Commit

**$** git commit --amend -m "Hehe" //**Undo Commit:** Xóa bỏ commit gần nhất và tạo commit mới

**$** git show mã\_commit // **Xem 1 commit**

**$** git diff mã\_commit // **Xem thay đổi file được modified trong commit**

## Undo commit trong Git

**$** git reset mã\_checksum //Di chuyển HEAD về phía commit chỉ định, giữ nguyên trạng thái thay đổi của các file và bỏ các file đó ra khỏi staged

**$** git reset --soft mã\_checksum //Tương tự git reset, các file vẫn ở trạng thái staged

**$** git reset --mixed mã\_checksum //Tương tự git reset, các file đưa về working dictory

**$** git reset --hard mã\_checksum //Tương tự git reset, **loại bỏ sự thay đổi của file**

**$** git revert mã\_checksum //Xóa bỏ xử thay đổi của commit và đảo commit đó lên đầu

## Tag trong Git

* + **Lightweight Tag**: Các tag chỉ đơn thuần là đánh dấu snapshot của commit.
  + **Annotated Tag**: Sử dụng tag này để đặt tiêu đề cho tag, và xem thông tin về người tag, ngày tag,….
* Xem danh sách các tag bằng lệnh: **git tag**
* Cách tạo **Lightweight tag**

**$** git tag tên\_tag

**$** git tag //Xem tất cả các tag

**$** git show tên\_tag //Xem commit được gắn tag <tag sẽ tự động gán vào commit gần nhất>

* Cách tạo **Annotated Tag**

**$** git tag -a v1.0 -m "Ra mat phien ban 1.0" // -a tên tag và –m tên lời nhắn

* Gắn tag cho commit bất kì

**$** git tag -a v0.0 mã\_checksum -m "Tag for inintial commit"

* Push Tag

Mặc định sử dụng lệnh **git push** các tag đã sẽ không được *push* lên *repository* mà cần sử dụng lệnh:

**$** git push –tags

* Nhập tag vào branch

**$** git checkout -b tên\_branch tên\_tag

## Remote trong Git

**$** git remote –v //**Kiểm tra các remote**

**$** git remote rename tên\_cũ tên\_mới // **Đổi tên remote**

**$** git remote add tên\_remote URL // **Thêm 1 remote mới**

**$** **git remote show** tên\_remote//Xem toàn bộ branch trong remote

## Push trong Git

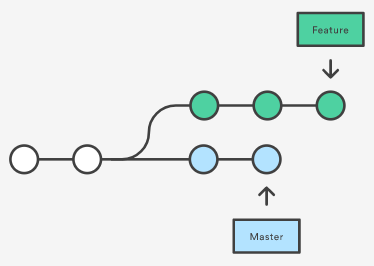
**$** git push tên\_remote tên\_branch //Push branch lên repo

**$** git push tên\_remote tên\_branch:tên\_remote\_branch //Đổi tên của remote branch trên server

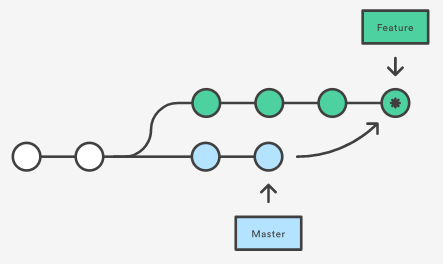
## Rebase và Merge trong Git

**$** git merge tên\_branch // Gộp 2 branch

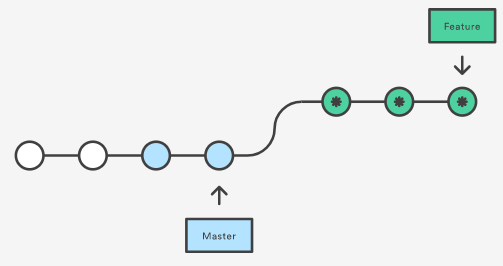
**$** git rebase tên\_branch //Gộp 2 branch



Repo



Sử dụng merge



Sử dụng rebase

## Đồng bộ trong Git

**$** git clone // Sao chép dữ liệu, thiết lập từ repo về máy

**$** git pull // Lấy dữ liệu từ remote repo gộp vào branh hiện tại

**$** git pull tên\_remote tên\_remote\_branch // Hoặc

**$** git fetch // Lấy dữ liệu từ remote repo sẽ lưu về máy nhưng không merge

**$** git fetch tên\_remote tên\_remote\_branch // Hoặc

**git pull = git fetch + git merge**

## Branch trong Git

**$** git branch // **Xem branch** trong working tree

**$** git branch tên\_branch //**Tạo 1 branch**

**$** git checkout -b tên\_branch //**Tạo và checkout đến branch**

**$** git checkout tên\_branch //**Checkout 1 branch**

**$** git branch --d tên\_branch //**Xóa 1 local branch đã được merge**

**$** git branch --D tên\_branch //**Xóa 1 local branch**

**$** git push –delete tên\_remote tên\_branch //**Xóa 1 remote branch**

**(Xóa nhiều branch bằng cách liệt kê tến các branch bởi khoảng trắng)**

**Gộp dữ liệu từ một branch**

**$** **git checkout master // Chuyển về nhánh master**

**$** **git merge develop // Gộp branch**

## Tìm hiểu về Git

* **HEAD – con trỏ vị trí:** Từ khóa **HEAD** tượng trưng cho biết vị trí hiện tại trong working tree.
* **HEAD** chính là commit hiện tại ta đang đứng.
* **HEAD~1** là commit ngay trước HEAD trên cùng nhánh
* **HEAD~n** là commit trước HEAD n commit trên cùng nhánh
* Có thể dùng **@** thay cho **HEAD**: git log @

**.gitignore**

* Bỏ qua file không muốn commit
* Liệt kê tên các file/folder không muốn đưa lên commit