# Hướng dẫn git dễ hiểu- câu lệnh git cơ bản

Bởi

[**Sơn Dương**](https://vntalking.com/author/smobile)

 -

[1](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "comments)

Mình đã có bài viết về tìm hiểu [Github là gì](https://vntalking.com/github-la-gi-cach-su-dung-github.html), cách sử dụng github. Đây là dịch vụ quản lý source code git cực kỳ nổi tiếng. Tuy nhiên, mình vẫn cứ cảm thấy lăn tăn, thiếu gì đó.

Bởi vì, Github nói cho cùng chỉ là một công cụ được xây dựng xoay quanh git. Nếu bạn chưa biết về git thì sẽ rất khó khăn để hiểu và sử dụng Github hiệu quả.

Ngoài ra, khi bạn đi làm, rất nhiều công ty sử dụng git thay vì SVN để quản lý source code. Do vậy, nếu nắm vững được nguyên lý của git, bạn sẽ dễ dàng phối hợp công việc tốt hơn với đồng nghiệp.

Đó là lý do ra đời bài viết này.

Qua đây, mình sẽ chia sẻ những kiến thức nền tảng để hiểu bản chất cách git quản lý source code như thế nào.

Chúng ta bắt đầu nhé!

**Nội dung chính của bài viết**

* [Git là gì? Tổng quan về Git](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "git-la-gi-tong-quan-ve-git)
* [#Những khái niệm cơ bản trong Git](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "nhung-khai-niem-co-ban-trong-git)
  + [1. Repository là gì?](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "1-repository-la-gi)
  + [2. Branch là gì?](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "2-branch-la-gi)
  + [3. Thế nào là git stash?](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "3-the-nao-la-git-stash)
  + [4. Git flow là gì?](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "4-git-flow-la-gi)
    - [Tại sao lại cần đến Git flow?](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "tai-sao-lai-can-den-git-flow)
  + [5. Những khái niệm Git quan trọng khác](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "5-nhung-khai-niem-git-quan-trong-khac)
    - [#Pull](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "pull)
    - [#Commit và Push](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "commit-va-push)
    - [# Merge](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "-merge)
    - [#Giải quyết khi Conflict](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "giai-quyet-khi-conflict)
* [#Cài đặt Git và các công cụ hỗ trợ Git](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "cai-dat-git-va-cac-cong-cu-ho-tro-git)
  + [1. Cài đặt Git trên Window, Linux, MacOs](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "1-cai-dat-git-tren-window-linux-macos)
    - [#Trên Window](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "tren-window)
    - [#Trên Linux](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "tren-linux)
    - [#Trên MacOs](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "tren-macos)
  + [2. Công cụ hỗ trợ quản lý Git](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "2-cong-cu-ho-tro-quan-ly-git)
* [#Thực hành quản lý source code git](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "thuc-hanh-quan-ly-source-code-git)
  + [#Lời kết](https://vntalking.com/cach-quan-ly-source-code-git.html" \l "loi-ket)

## **Git là gì? Tổng quan về Git**

Git là hệ thống quản lý source code phân tán. Ban đầu git được phát triển để quản lý mã nguồn của Linux.

Với Git, bạn có thể quản lý một file dưới dạng lịch sử cập nhật. Vì thế, bạn có thể xem được toàn bộ lịch sử chỉnh sửa của file cũng như khôi phục file về trạng thái ban đầu dễ như bỡn.

Do Git sử dụng mô hình phân tán, mỗi nơi lưu trữ source code sẽ được gọi là repository. Các repository có thể lưu trữ trên máy tính mỗi thành viên, hoặc trên server.

Khi mã nguồn có sự thay đổi, bạn có thể push nó lên server hoặc chia sẻ luôn cho các thành viên khác mà không cần thông qua server.

Một số lợi ích khi sử dụng git:

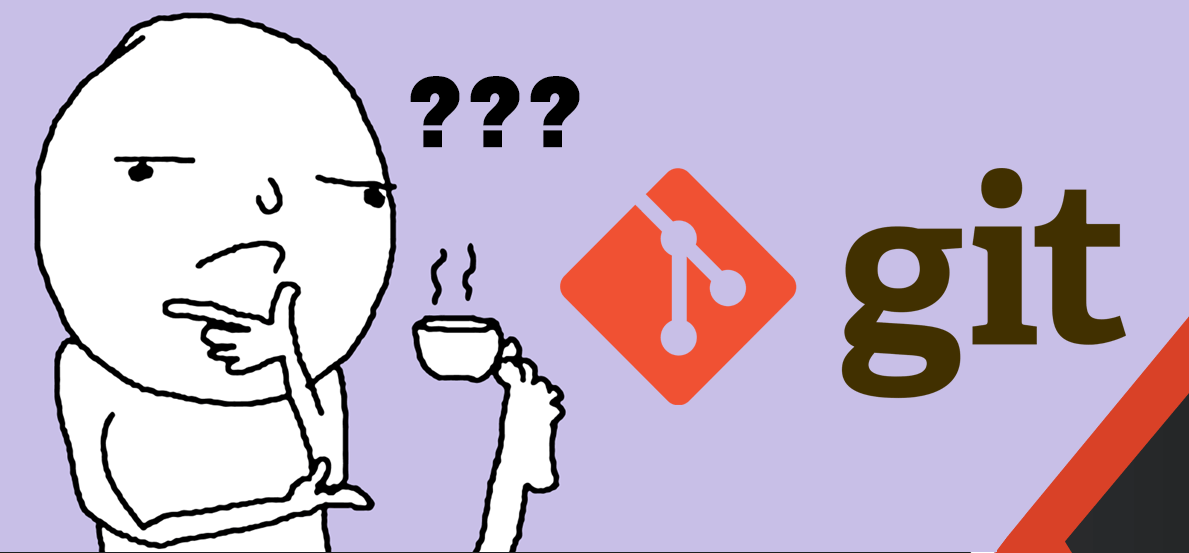
* **An toàn hơn**: với Git, bạn cứ tự tin mà sửa mã nguồn cho thỏa thích. Bạn có thể revert lại phiên bản cũ hơn bất cứ lúc nào. Điều này rất có ích khi một ai đó fix bug hoặc thêm tính năng vô tình gây ảnh hưởng tới tính năng khác. Tốt nhất là revert lại bản trước 🙆 .
* **Làm việc nhóm hiệu quả hơn**: Với các dự án nhiều thành viên, git giúp các thành viên có thể làm việc đồng thời trên các repository khác nhau. Cuối cùng thì merge tất cả lên một nhánh chính (master branch). Đảm bảo không ai phải chờ đợi ai cả.
* **Không phụ thuộc vào server**: Chính vì Git là hệ thống phân tán nên các thành viên vẫn có thể làm việc, commit, update ngay cả khi server offline. Khi nào server online thì lại push lên server bình thường.

💦 **Đọc thêm:**[**Quy tắc học lập trình**](https://vntalking.com/quy-tac-hoc-lap-trinh.html)

Để hiểu cách sử dụng Git, chúng ta sẽ tìm hiểu những khái niệm quan trọng trong Git.

## **#Những khái niệm cơ bản trong Git**

Dưới đây là những khái niệm mà theo mình, bạn cần phải biết để có thể hiểu và sử dụng được git trơn tru.

**1. Repository là gì?**

Repository là nơi chứa toàn bộ source code và thông tin lịch sử thay đổi từng file của toàn bộ project.

Các thông tin lịch sử thay đổi sẽ được lưu trong thư mục ẩn .git. Thư mục này nằm cùng với thư mục chứa source code (Working Directory).

Để tạo mới một repository, chúng ta dùng lệnh:

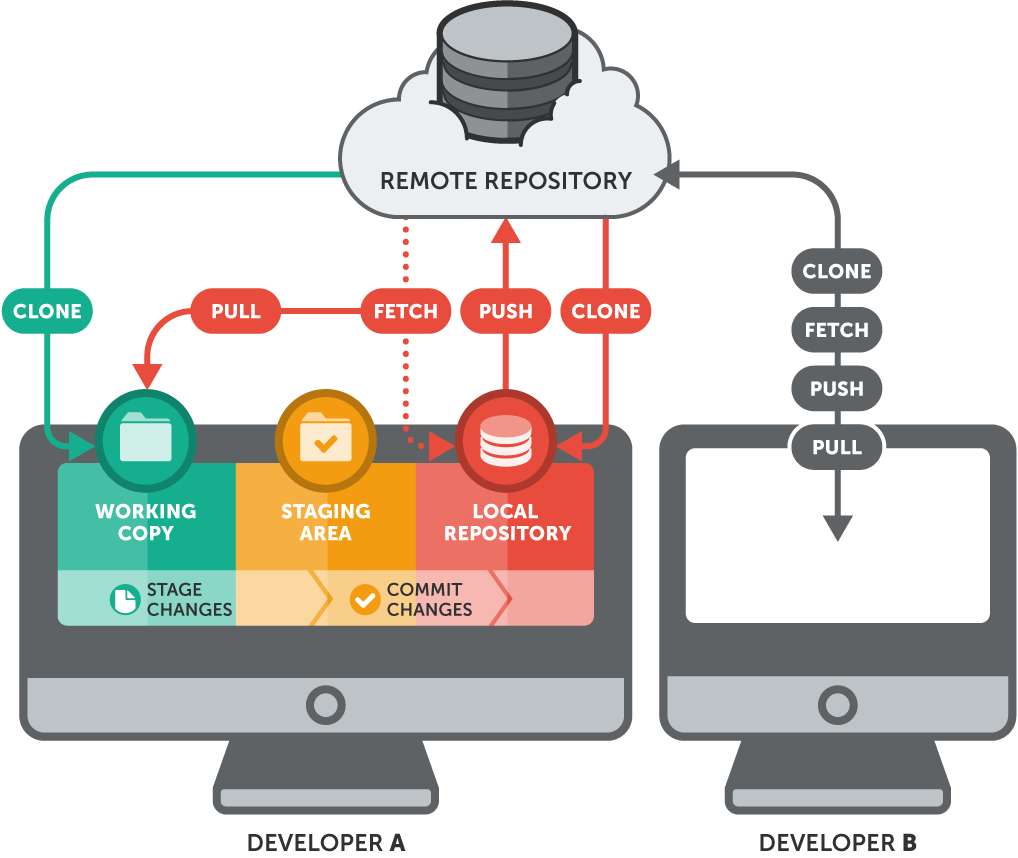
* $ git init

Có 2 loại Repository:

* **Remote repository**: là repository để chia sẻ cho nhiều người. Thường nó sẽ nằm trên server chuyên dụng. Có một số dịch vụ nổi tiếng phân phối remote repo như: Github, Gitlab hay Bitbucket.v.v…
* **Local repository**: là repository nằm trên chính máy tính của bạn.

Với những công việc bình thường, bạn sẽ thao tác trực tiếp trên local repository. Sau đó, khi muốn public nội dung công việc, hay đơn giản là bạn phải giao kết quả công việc cho sếp thì lúc này bạn cần phải push lên remote repository.

Ngược lại, để lấy phần sửa đổi mã nguồn của các thành viên khác trong team, bạn cần phải pull từ remote repo.

**2. Branch là gì?**

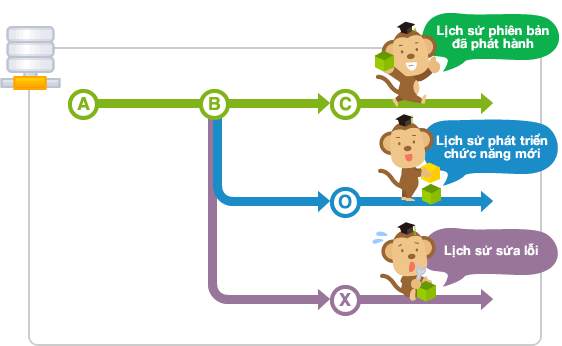
Branch có thể hiểu là các nhánh của một Repository. Một repository có thể có nhiều nhánh, các nhánh này có thể merge vào nhau bất kỳ lúc nào.

Branch là một điểm mạnh của hệ thống git. Vì vậy nếu bạn sử dụng git mà chỉ có duy nhất một branch thì quả thật quá đáng tiếc.

Khi bạn tạo repository, bạn sẽ có một branch mặc định là master. Master branch sẽ chứa toàn bộ dữ liệu (mã nguồn, tài liệu…) của repository.

Sử dụng branch có rất nhiều lợi ích, mình chỉ liệt kê 2 lợi ích mà mình hay gặp nhất:

* Có thể phát triển, thử nghiệm các tính năng mới mà không ảnh hưởng tới tiến độ của dự án chính.
* Có thể tạo nhiều branch khác nhau, phát triển đồng thời, sau đó thì có thể merge vào master branch bất kì lúc nào. Ví dụ: dự án có một master branch, test branch, hay unstable branch…



Tất nhiên, mỗi một dự án lại có đặc thù riêng nên người ta có thể tạo branch cho phù hợp. Kết hợp giữa việc tạo branch và quy tắc merge các nhánh sẽ tạo ra git flow. Mình sẽ giải thích gitflow ở bên dưới nhé!

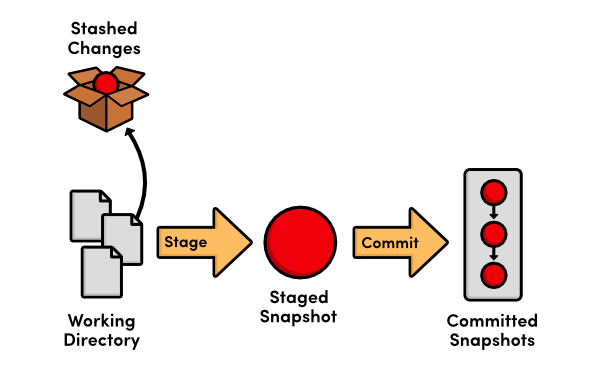
💦 **Mở rộng kiến thức:**[**Series học lập trình Nodejs cơ bản dành cho bạn**](https://vntalking.com/hoc-lap-trinh-node-js-cho-nguoi-moi.html)

### **3. Thế nào là git stash?**

Git stash được sử dụng để lưu lại những thay đổi khi bạn không muốn commit, tránh những commit rác khi thay đổi chưa hoàn thiện.

Git stash đặc biệt hữu ích trong trường hợp bạn muốn chuyển sang một branch khác mà lại đang làm dở ở branch hiện tại.

Bạn có thể git stash bao nhiêu lần cũng được. Mỗi lần tạo stash, git coi toàn bộ thay đối như một phần tử trong stack.



Cách sử dụng git stash, chúng ta sẽ tìm hiểu và thực hành ở phía dưới bài viết.

### **4. Git flow là gì?**

Git Flow thực chất là quy tắc tạo và merge branch khi các thành viên trong team hoàn thành công việc của mình. Git flow có thể coi là một quy trình quản lý branch giúp cải thiện quá trình làm việc với git.

Git Flow làm việc dựa trên câu lệnh merge, chứ không phải thao tác cherry-pick thường thấy.

#### **Tại sao lại cần đến Git flow?**

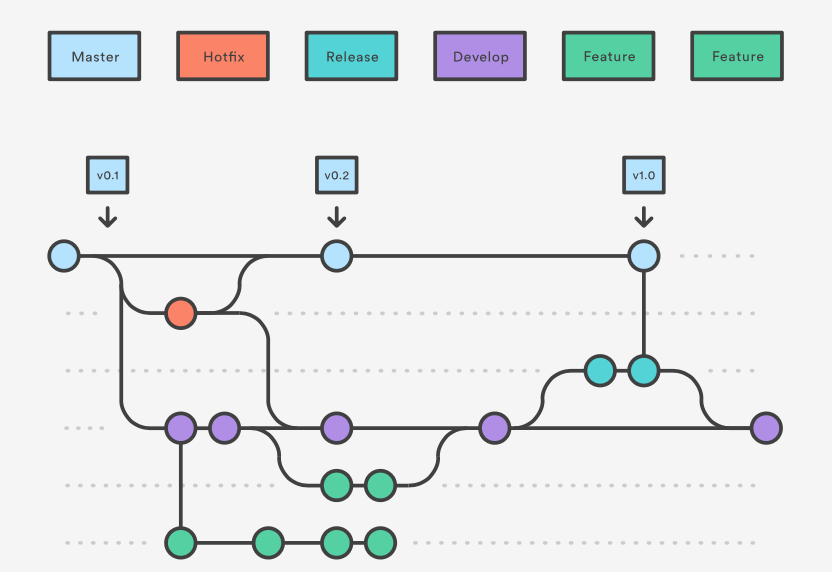
Để hiểu tại sao người ta phải nghĩ ra quy trình git flow này, mình sẽ lấy một ví dụ cách quản lý source code git điển hình:

* Project leader sẽ tạo một repo có 1 master branch duy nhất.
* Các developer sẽ checkout master branch này về để làm việc. Tất cả các implement tính năng, fix bug đều được push lên nhánh này.
* Định kỳ project leader sẽ review và cherry-pick các commit vào master branch để build các bản release.

Từ tình huống trên, bạn có nhận ra nhược điểm lớn nhất của nó là gì không?

Nhược điểm lớn nhất chính là Project Leader sẽ gặp rất nhiều conflict khi cherry-pick vào master branch. Đặc biệt là với hot fix với nhiều vấn đề có khả năng ghi đè lên nhau.

Để khắc phục nhược điểm này, người ta nghĩ ra quy trình git flow. Về cơ bản, git flow sẽ tách master branch thành nhiều branchs: release, hotfix, develop, feature 1, feature 2,…



Phần này mình sẽ không đi chi tiết cách vận hành, quy tắc commit, merge nhánh trong git flow.

Mục tiêu của phần này là để bạn biết đến sự tồn tại của git flow. Mình sẽ viết cụ thể hơn về Git flow nếu mọi người có hứng thú.

### **5. Những khái niệm Git quan trọng khác**

Ngoài những khái niệm trụ cột ở trên, Git còn những khái niệm nhỏ khác nhưng các bạn sẽ phải thao tác nó thường xuyên.

#### #Pull

Lệnh này cực đơn giản, nó được cùng để lấy tất cả sự thay đổi từ remote server về local repo của bạn.

git pull

#### #Commit và Push

Mục đích của commit là lưu lại những thay đổi của một file hoặc cả folder vào một repository. Mỗi một commit là một phiên bản lịch sử của file, các commit nối tiếp nhau theo thứ tự thời gian.

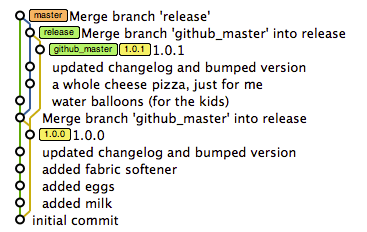
git commit –m "nội dung miên tả cho commit"

**Lưu ý:** Với hầu hết các bạn làm với git thì commit mới chỉ lưu lại trên local của bạn thôi (local repository) . Nếu muốn đẩy một commit lên server (Remote Repository), bạn cần phải sử dụng lệnh push.

git push origin <tên\_branch>

#### # Merge

Lệnh merge được dùng để gộp 1 branch bất kỳ vào một branch do bạn chỉ định.



git merge <tên\_branch>

Trước khi merge, để hạn chế những conflict đáng tiếc và mất thời gian, bạn cần lưu ý:

* Kiểm tra kỹ lưỡng branch hiện tại và branch đích muốn merge vào.
* Phải push tất cả commit trên local repo lên remote repository trước khi merge.
* Trước khi merge, cần phải pull tất cả những thay đổi của cả 2 nhánh muốn merge.
* Sau khi merge xong, cần push lại source lên server.

#### #Giải quyết khi Conflict

Đúng như tên gọi, conflict là trường hợp khi 2 developer cùng sửa ở một dùng của một file, và git không biết dòng sửa nào là đúng.

Để giải quyết conflict, bắt buộc phải có sự can thiệt của con người, đó chính là bạn.

Git còn rất nhiều lệnh hay ho khác nữa, nhưng bước đầu bạn chỉ cần nắm được những khái niệm như ở trên là được. Trong quá trình học và làm việc với git, bạn sẽ khám phá dần dần.

Phần tiếp theo, chúng ta sẽ cùng nhau cài đặt những tool cần thiết để có thể làm việc với Git nhé

## **#Cài đặt Git và các công cụ hỗ trợ Git**

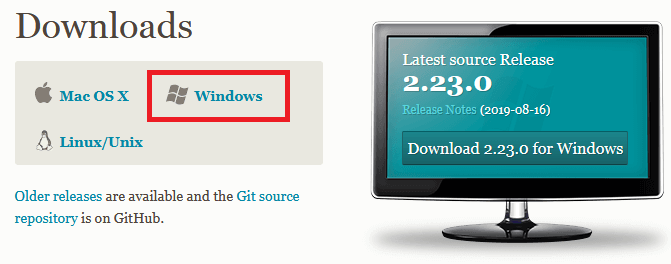
Phần này, mình sẽ hướng dẫn các bạn cài đặt những công cụ cần thiết để sử dụng git. Còn việc dựng một Git server để làm remote repositoty, mình sẽ làm ở phần sau (giờ mọi người chủ yếu dùng Github, Bitbucket rồi, khỏi cần đến server git chuyên dụng).

### **1. Cài đặt Git trên Window, Linux, MacOs**

Git là phần mềm hỗ trợ đa nền tảng, nên bạn có thể sử dụng git cho bất kỳ hệ điều hành nào bạn ưu thích.

#### #Trên Window

Việc cài đặt git trên window rất đơn giản, cũng giống như bao phần mềm khác thôi. Bạn vào trang chủ của git để [tải về](https://git-scm.com/downloads).



Sau khi tải về máy tính thì tiến hành cài đặt, bạn cứ nhắm mắt chọn Yes, với next cho tới lúc xong là được.

#### #Trên Linux

Với Linux còn đơn giản hơn nữa, bạn chỉ cần gõ đúng một dòng lệnh:

**Debian and Ubuntu:**

* sudo apt-get install git

**CentOS:**

* sudo yum install git

**Fedora:**

* sudo yum install git-core

**Arch Linux:**

* sudo pacman -Sy git

**Gentoo:**

* sudo emerge --ask --verbose dev-vcs/git

#### #Trên MacOs

Bạn có thể cài đặt git thông qua [Homebrew](https://brew.sh/).

Trước tiên, bạn cần cài Homebrew đã:

* /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

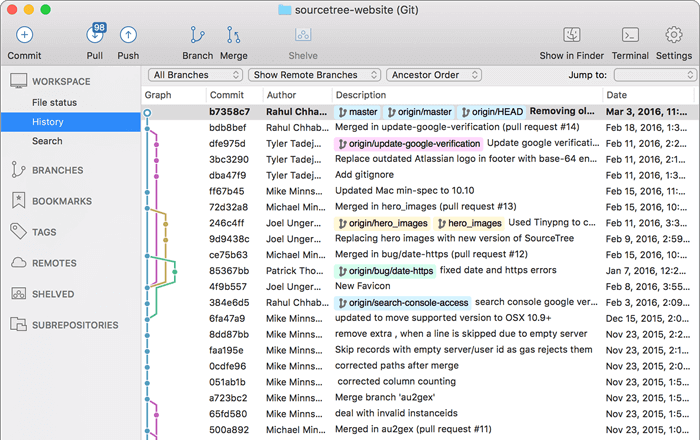
Sau đó thì cài đặt git thôi.

* brew update
* brew install git

### **2. Công cụ hỗ trợ quản lý Git**

Hiện có rất nhiều công cụ dạng GUI hỗ trợ bạn quản lý và làm việc với Git cực hiệu quả. Nhiều bạn lại thích dùng git dưới dạng command (chắc nhìn cho chuyên nghiệp).

Nhưng thời đại công nghệ 4.0 rồi, khi công cụ sản xuất đã phát triển thì tội gì chúng ta lại vẫn bảo thủ sử dụng cuốc thuổng gậy gộc để “cầy” thay vì dùng máy móc.



Mình đề xuất một số công cụ hỗ trợ quản lý git để bạn cài và sử dụng nhé:

* Window, MacOS: [Sourcetree](https://www.sourcetreeapp.com/)
* Linux: [Gitkraken](https://www.gitkraken.com/)

Các bạn tự cài đặt và trải nghiệm nhé!

Ngoài ra, bạn có thể kết nối tới các git server như Github, [Gitlab](https://vntalking.com/gitlab-ci-dich-vu-git-thay-the-github-hoan-hao.html)… thông qua SSH thì tham khảo cách làm ở đây nhé: [Sử dụng SSH để kết nối server không cần mật khẩu](https://vntalking.com/giao-thuc-ssh-la-gi-huong-dan-su-dung-ssh.html)

## **#Thực hành quản lý source code git**

Sau khi bạn đã phần nào nắm rõ được những khái niệm cơ bản của git. Chúng ta sẽ tìm hiểu một số lệnh, thao tác cần thiết hay sử dụng trong git.

**1. Clone một remote repo về máy tính**

Đây là thao tác rất hay làm khi bắt đầu dự án, bạn cần lấy source code từ remote repository về máy tính, đồng thời tạo luôn một local repo.

Ví dụ, một lệnh clone một repo từ Github:

* git clone git@github.com:<user\_name>/the-repo-you-are-cloning.git

**2. Commit những thay đổi của mã nguồn**

* // Add một file vào stag
* git add <file>
* // Commit
* git commit -m 'the message goes here for the commit'

**3. Checkout một branch**  
Bạn có thể checkout một branch bất kỳ bằng lệnh

* git checkout <branch name>

Hoặc tạo mới một branch:

* git branch <branch name>

**4. Pushing & Pulling**

Để đẩy những thay đổi lên server thì thông qua lệnh push.

* git push -u origin **master** (remote repo name[origin], local branch name[master])

Còn lấy thay đổi từ server về máy thì thông qua lệnh pull

* git pull

Cơ bản là như thế, còn nhiều lệnh lắm. Và mỗi lệnh lại có những tham số khác nhau trong những yêu cầu cụ thể của bạn.

Mình khuyên các bạn tham khảo hai bài viết này, cực chi tiết về các lệnh của Git luôn (chịu khó đọc tiếng Anh chút nhé):

* [Practice git](https://github.com/grayghostvisuals/practice-git)
* [Git training](https://github.com/UnseenWizzard/git_training)

### **#Lời kết**

Qua bài viết này, chúng ta đã cùng nhau hiểu được những khái niệm cơ bản nhất của Git rồi. Nhờ có Git mà việc quản lý source code của team trở nên dễ dàng hơn rất nhiều.

Mình hi vọng, các bạn sẽ tìm được kiến thức cần thiết cho mình qua bài này.

Bạn nghĩ sao về Git? Bạn có gặp vấn đề gì khi sử dụng Git trong dự án không? Hãy chia sẻ dưới phần bình luận cho mọi người nhé!