

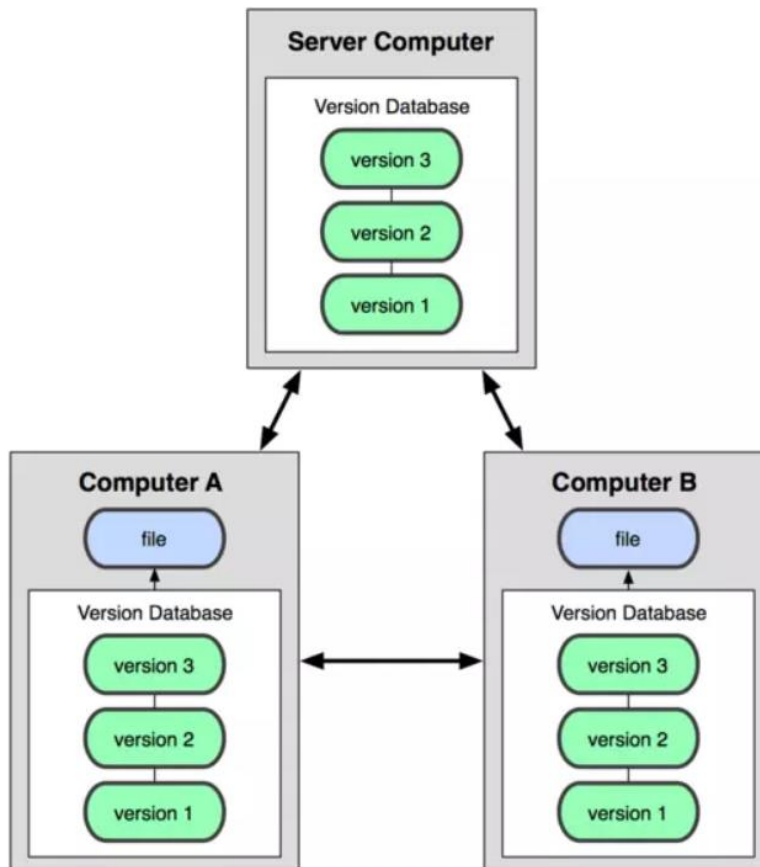
# CƠ BẢN VỀ GIT – GITHUB

## I. GIT

### 1. Khái niệm

Git là tên gọi là một hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay (Distributed Version Control System – DVCS), với các ưu điểm: tốc độ, đơn giản, phân tán, phù hợp với dự án lớn nhỏ.

DVCS nghĩa là hệ thống giúp mỗi máy tính có thể lưu trữ nhiều phiên bản khác nhau của một mã nguồn được nhân bản (clone) từ một kho chứa mã nguồn (repository), mỗi thay đổi vào mã nguồn trên máy tính sẽ có thể ủy thác (commit) rồi đưa lên máy chủ nơi đặt kho chứa chính. Và một máy tính khác (nếu họ có quyền truy cập) cũng có thể clone lại mã nguồn từ kho chứa hoặc clone lại một tập hợp các thay đổi mới nhất trên máy tính kia. Trong Git, thư mục làm việc trên máy tính gọi là Working Tree.



Hiệu đơn giản thì Git sẽ giúp người dùng lưu lại các phiên bản của những lần thay đổi vào mã nguồn và sẽ dễ dàng cho việc khôi phục lại mà không cần phải thủ công copy rồi paste vào đâu đó, phiên bản đó đã được sao lưu. Khi chúng ta phát hiện ra lỗi ở đâu đó và muốn backup lại phiên làm việc trước khi bị lỗi xảy ra thì sẽ thật đơn giản khi chúng ta sử dụng Git. Một điểm đặc biệt nữa là một thành viên trong cùng

một team khi làm việc với nhau hoàn toàn có thể theo dõi online được các thay đổi của các thành viên khác ở từng phiên bản làm việc mà không nhất thiết phải ngồi ngay cạnh nhau, họ cũng có thể đối chiếu được những thay đổi đó để rồi gộp phiên bản của thành viên khác vào phiên bản của họ. Cuối cùng là tất cả có thể đưa các thay đổi vào mã nguồn của mình lên một kho chứa mã nguồn.

## 2. Cách lưu dữ liệu của Git

Git lưu dữ liệu dưới dạng một loạt "ảnh chụp" (snapshot) của một tập hợp các file, có nghĩa là mỗi khi bạn commit (lưu lại) thì Git tiến hành chụp lại hệ thống các file thời điểm đó và lưu giữ một tham chiếu đến ảnh chụp đó, nhớ rằng các file không có thay đổi thì Git sẽ không lưu lại file đó lần nữa mà chỉ có một liên kết đến file đã lưu ở lần trước.

## 3. Hầu hết mọi thao tác với Git diễn ra ở Local

Vì mỗi máy trạm có một Database Git nên hầu hết thao tác với Git như thêm mới, xem lại lịch sử ... đều không cần đến Server (trừ cần commit lên Server, lấy về một file do người khác đã cập nhật).

## 4. Dữ liệu lưu trữ trong Git đảm bảo tính toàn vẹn

Mọi thứ trước khi được lưu trữ vào Git đều được kiểm tra bởi mã băm (hash, checksum), có nghĩa là không thể thay đổi nội dung của file mà Git không biết về sự thay đổi đó. Chức năng này giúp cho bạn không thể mất thông tin khi trao đổi dữ liệu hay file lỗi mà không thể nhận ra được. Git sử dụng mã hash SHA-1, mỗi chuỗi hash SHA-1 sinh ra căn cứ theo nội dung của file dài 40 ký tự (tạo ra từ các ký tự trong khoảng thập lục phân : 0-9, a-f) có dạng:

```
62FC2DBFB0CB299DD8548286FE1BB1D2B2041379
```

Mã hash kiểu này bạn sẽ gặp thường xuyên khi làm việc với Git vì mọi thứ lưu trong dữ liệu đều căn cứ theo mã hash này.

## 5. Với Git cơ bản là thêm vào nó các dữ liệu

Các hành động trong Git hầu hết là các thao tác là thêm dữ liệu vào hệ thống Git. Rất khó để yêu cầu hệ thống thực hiện việc gì đó mà không phục hồi lại được, cũng như khó mà xóa hoàn toàn dữ liệu khỏi hệ thống

## 6. Ba trạng thái với Git

Với Git, các file của bạn có thể nằm ở một trong ba trạng thái: **committed**, **modified**, **staged**

**committed** có nghĩa là dữ liệu đã lưu trữ an toàn trong Database (local)

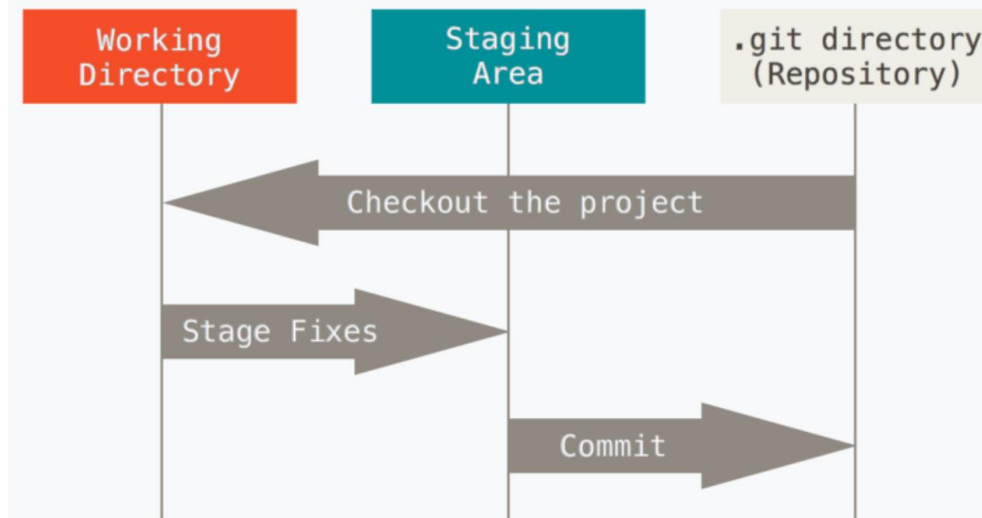
**modified** có nghĩa là dữ liệu (file) có thay đổi nhưng chưa lưu trong Database (local)

**staged** có nghĩa đánh dấu các file sửa đổi modified chuẩn bị được commit tiếp theo

**staged** có nghĩa đánh dấu các file sửa đổi modified chuẩn bị được commit tiếp theo

Các khái niệm trên dẫn tới Git tổ chức một dự án ra ba khu vực:

- 1 Thư mục làm việc (Working tree) - Chứa bản sao phiên bản cụ thể của dự án
- 2 Khu vực sắp xếp (staging) - là một file, nằm trong thư mục Git chứa thông tin sẽ commit
- 3 Thư mục Git (Git directory) - nơi Git lưu Database các file Metadata



Từ hình trên, khi làm việc với Git bạn theo tiến trình như sau:

- Bạn code, sửa đổi các file trong 1 Thư mục làm việc (Working tree)
- Bạn tổ chức những sự thay đổi nào muốn commit tiếp theo, cơ bản là việc đánh dấu sự thay đổi vào khu vực 2 Khu vực sắp xếp (staging)
- Bạn thực hiện commit để các file đánh dấu trong staged lưu vào 3 Git (database) như nhưng snapshot.

## II. GITHUB

### 1. Khái niệm

[GitHub](#) là một dịch vụ nổi tiếng cung cấp kho lưu trữ mã nguồn [Git](#) cho các dự án phần mềm. **Github có đầy đủ những tính năng của Git**, ngoài ra nó còn bổ sung những tính năng về social để các developer tương tác với nhau.

[GitHub](#) có 2 phiên bản: miễn phí và trả phí. Với phiên bản có phí thường được các doanh nghiệp sử dụng để tăng khả năng quản lý team cũng như phân quyền bảo mật dự án. Còn lại thì phần lớn chúng ta đều sử dụng Github với tài khoản miễn phí để lưu trữ source code.

Github **cung cấp các tính năng social networking** như feeds, followers, và network graph để các developer học hỏi kinh nghiệm của nhau thông qua lịch sử commit.

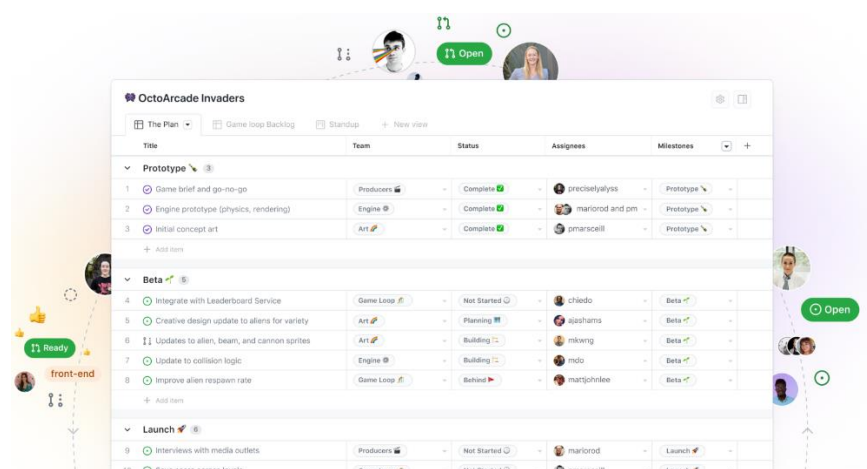
Nếu một comment để mô tả và giải thích một đoạn code. Thì với [Github](#), commit message chính là phần mô tả hành động mà bạn thực hiện trên source code.

Theo số liệu thống kê vào tháng 10 năm 2020, đây là máy chủ lưu trữ mã nguồn nổi bật nhất. Với hơn 60 triệu kho lưu trữ mới được tạo vào năm 2020 và có tổng cộng hơn 56 triệu nhà phát triển, nhà lập trình đăng ký trên nền tảng.

## **2. Những tính năng nổi bật của GitHub**

### **2.1. Quản lý dự án ở mọi quy mô dễ dàng**

GitHub cung cấp một nơi các nhà quản lý, nhà phát triển dự án, lập trình viên có thể cùng nhau phối hợp để theo dõi tiến độ, cập nhật hiệu quả của dự án. Hệ thống hoàn toàn minh bạch và kiểm tra được liên tục.



### **2.2. Chia sẻ khi cần thiết, tái sử dụng code**

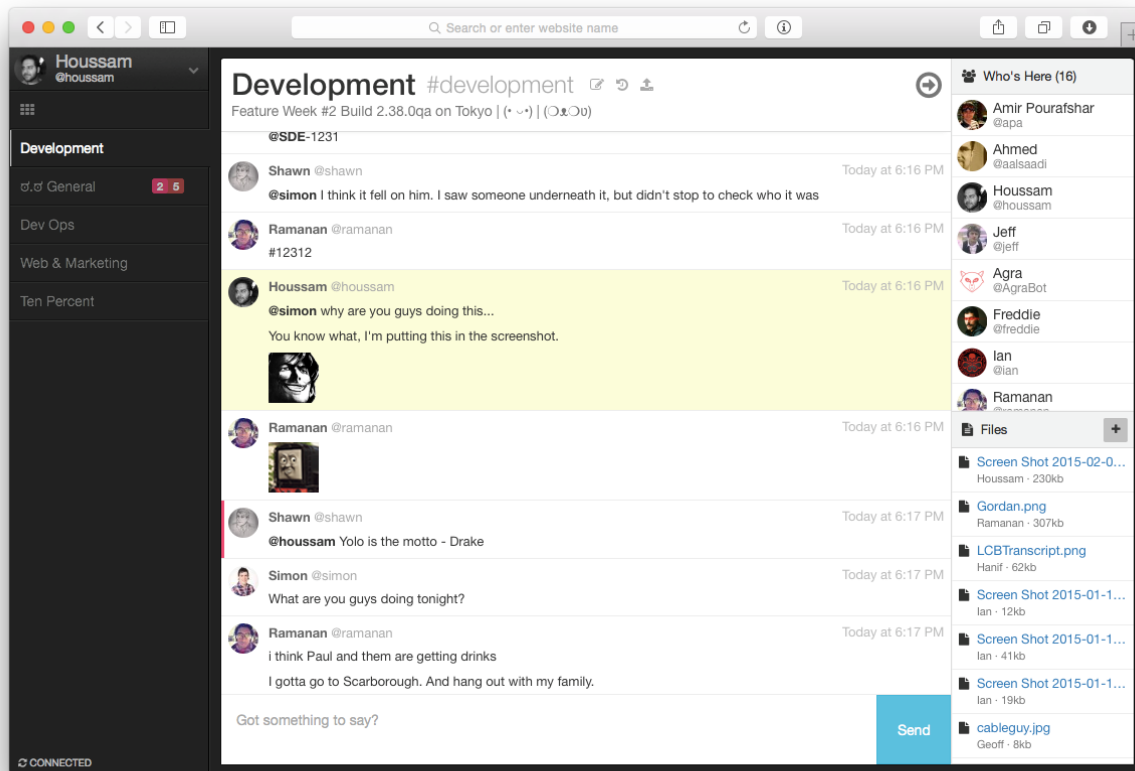
Các dự án có thể được đặt ở chế độ riêng tư, chia sẻ trong một nhóm hoặc công khai với cộng đồng (tạo nên các mã nguồn mở). Những mã nguồn của dự án khi được chia sẻ công khai, bạn hoàn toàn tự do tải xuống và sử dụng lại, nếu có phần mã tái sử dụng trong dự án.

### **2.3. Khả năng quản lý, làm việc nhóm hiệu quả**

GitHub cung cấp cho tất cả thành viên trong nhóm làm việc một cách trơn tru và hiệu quả khi cung cấp giao diện trực quan, theo dõi được tiến độ, các công cụ đánh giá công việc của từng thành viên.

### **2.4. Trao đổi dễ dàng, kịp thời**

Khi có những vấn đề phát sinh trong quá trình phát triển dự án. Các thành viên trong nhóm sẽ thảo luận và trình bày những ý kiến cụ thể, đúng lúc để dự án có thể tiếp tục được triển khai tiếp, tránh tình trạng không rõ yêu cầu, lỗi phát sinh, làm gián đoạn dự án.



## 2.5. Mức an toàn, bảo mật cao

Github được phát triển và luôn cập nhật lỗ hổng bảo mật, đảm bảo các mã code của người dùng luôn được an toàn.

## 2.6. Lưu trữ các dòng mã code dễ dàng

Tất cả các phần mã code đều được lưu trữ trên Github, bạn hoàn toàn có thể truy cập bất cứ lúc nào, trên đa dạng thiết bị. Hiện có hàng triệu dự án được lưu trữ trên Github và mỗi dự án đều được bảo mật cao.

## 3. Các loại tài khoản trên Github

Hiện Github có hai tùy chọn tài khoản miễn phí và trả phí để phục vụ nhu cầu đa dạng của các cá nhân và tổ chức khác nhau.

- **Tài khoản miễn phí:** Đa số được sử dụng bởi các lập trình viên cá nhân, các dự án nhỏ dùng để lưu trữ mã nguồn.
- **Tài khoản trả phí:** Thường được các doanh nghiệp, nhóm phát triển dự án lớn để tăng người dùng tham gia vào dự án, tăng khả năng quản lý nhóm, phân quyền để tăng tính bảo mật của dự án.

Free	Pro	Team	Enterprise	GitHub One
GitHub basics for every developer	Advanced collaboration for your projects	Essential management and security for small teams	Security, compliance, and flexible deployment for enterprises	All of our best tools, support, and services
<ul style="list-style-type: none"> <li>Unlimited repositories</li> <li>3 collaborators/private repository</li> <li>2,000 Action minutes/month Free for public repositories</li> <li>500MB of GitHub Packages storage</li> <li>Automated security updates</li> </ul>	<ul style="list-style-type: none"> <li>Everything included in Free</li> <li>Unlimited collaborators</li> <li>3,000 Action minutes/month Free for public repositories</li> <li>1GB of GitHub Packages storage</li> <li>Code owners</li> </ul>	<ul style="list-style-type: none"> <li>Everything included in Pro</li> <li>Team access controls</li> <li>10,000 Action minutes/month Free for public repositories</li> <li>2GB of GitHub Packages storage</li> <li>GitHub Security Advisories</li> </ul>	<ul style="list-style-type: none"> <li>Everything included in Team</li> <li>SAML single sign-on</li> <li>50,000 Action minutes/month Free for public repositories</li> <li>50GB of GitHub Packages storage</li> <li>Advanced auditing</li> </ul>	<ul style="list-style-type: none"> <li>Everything included in Enterprise</li> <li>Community-powered security</li> <li>Actionable metrics</li> <li>Continuous learning</li> <li>24/7 support</li> </ul>
\$0 /month	\$7 /month	\$9 per user/month	Contact Sales	Learn more
<a href="#">Join for free</a>	<a href="#">Continue with Pro</a>	<a href="#">Continue with Team</a>	<a href="#">Start a free trial</a>	<a href="#">Contact Sales</a>

## III. CÁC THAO TÁC VỚI GIT - GITHUB

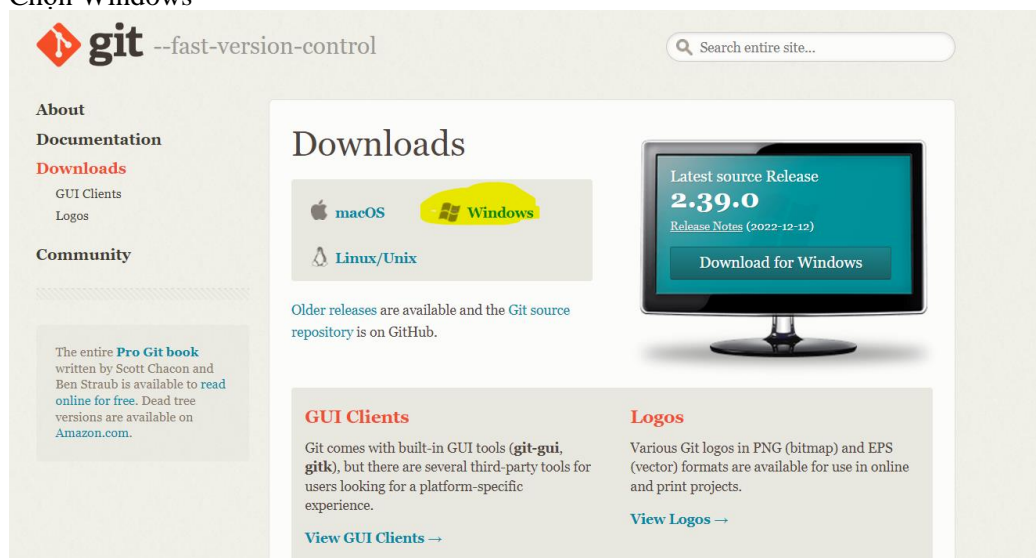
### 1. Cài đặt Git

Git có thể được cài đặt trên hầu hết hệ điều hành như Windows, Mac và Linux. Trên thực tế, Git được cài đặt sẵn trên hầu hết Mac và Linux.

#### Các bước cài đặt Git trên Windows

##### *Bước 1: Tải Git*

- Vào trang Web <https://git-scm.com/downloads>
- Chọn Windows

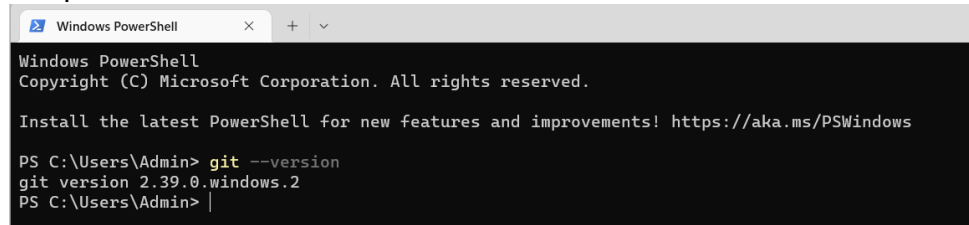


- Và chọn phiên bản Git phù hợp với máy tính

##### *Bước 2: Kiểm tra cài đặt Git*

- Mở Windows Powershell/command prompt rồi gõ `git --version`
- Nếu cài đặt thành công thì màn hình sẽ hiển thị phiên bản của Git

Ví dụ:



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\Admin> git --version
git version 2.39.0.windows.2
PS C:\Users\Admin> |
```

*Bước 3: Thiết lập tên người dùng*

```
git config --global user.name "Tên của Bạn"
```

*Bước 4: Thiết lập email người dùng:*

```
git config --global user.email emailcuaban@domain.com
```

## **2. Tạo một kho chứa git tại máy của bạn (local git repository)**

Để bắt đầu, mở terminal và di chuyển tới nơi mà bạn muốn tạo project của mình với dòng lệnh `cd` (viết tắt của change directory). Ví dụ, bạn có một project là "myproject" tại desktop:

```
$ cd ~/Desktop

$ mkdir myproject

$ cd myproject/
```

(mkdir myproject: tạo folder tên là myproject)

Để khởi tạo kho chứa git (git repo) ở thư mục gốc, sử dụng câu lệnh `git init`:

```
$ git init
```

## **3. Tạo file mới vào git repo, git add**

Bạn có thể tạo thủ công một file mới rồi save, hoặc sử dụng lệnh `touch`. Ví dụ `touch newfile.txt` sẽ tạo và lưu một file rỗng có tên là newfile.txt.

Khi bạn thêm hoặc chỉnh sửa một file trong thư mục chứa git repo, file đấy sẽ tồn tại trong git repo. Nhưng git sẽ không theo dõi file nếu bạn không yêu cầu cụ thể. Git chỉ lưu hay quản lý những thay đổi đối với những file mà nó theo dõi, vì vậy chúng ta cần có dòng lệnh yêu cầu Git làm điều đấy.

```
$ touch newfile.txt
```

```
$ ls  
newfile.txt
```

Sau khi tạo file, sử dụng **git status** để xem file nào mà git biết nó tồn tại:

```
$ git status  
  
On branch master  
  
Initial commit  
  
Untracked files:  
  
  (use "git add <file>..." to include in what will be committed)  
  
      newfile.txt  
  
nothing added to commit but untracked files present (use "git add" to track)
```

Untracked files: những file chưa được theo dõi bởi git. Ta cần sử dụng `git add` để đưa những file mình muốn git theo dõi vào vùng theo dõi, bằng cách sử dụng câu lệnh `git add <file_name>`

```
$ git add new.txt  
  
$ git status  
  
On branch master  
  
No commits yet  
  
Changes to be committed:  
  
  (use "git rm --cached <file>..." to unstage)  
  
      new file:   new.txt
```

Nếu có nhiều file bạn muốn add, thay vì chỉ định `file_name` bạn có thể sử dụng `git add .` để đưa tất cả các file vào vùng theo dõi.

Sau khi sử dụng `git add`, bạn thấy rằng git đã add các file vào vùng theo dõi, sẵn sàng để commit.

#### **4. Tạo một commit**

Đây là thời điểm để tạo commit đầu tiên của bạn. Sử dụng câu lệnh `git commit -m "Your message about the commit"`



```
$ git commit -m"This is my first commit"
```

```
[master (root-commit) d1f07b8] This is my first commit
```

```
1 file changed, 0 insertions(+), 0 deletions(-)
```

```
create mode 100644 new.txt
```

Message ở cuối commit nên liên quan đến những cái mà bạn đã làm, thay vì chỉ nguệch ngoạc vài cái kiểu "mdsfdsf" hay "dejavu", để sau này khi tìm lại code, bạn sẽ biết được commit này mình đã làm gì.

Để xem các commit của mình, sử dụng **git log** để xem chi tiết, **git log --oneline** để xem các commit mà mỗi commit chỉ hiển thị trên 1 dòng.

```
$ git log
```

```
commit d1f07b8b892e9b6a6a2e151727a8b30e9cd6200d (HEAD -> master)
```

```
Author: thaidoandat <thaidoandat1@gmail.com>
```

```
Date: Wed Aug 11 23:18:32 2021 +0700
```

```
    This is my first commit
```

```
$ git log --oneline
```

```
d1f07b8 (HEAD -> master) This is my first commit
```

## **5. Tạo một branch mới**

Bạn muốn làm một chức năng nhưng lại lo rằng sẽ thay đổi main project trong khi đang phát triển tính năng. Đây là lúc git branch lên ngôi. Nghĩa là bạn sẽ tạo một nhánh mới, sẽ chứa các phần của nhánh hiện tại nơi mà bạn đang đứng để tạo ra nhánh đó, và việc thay đổi trên nhánh này sẽ không ảnh hưởng gì đến luồng chạy chính của project. Và sau khi bạn thấy code trên nhánh này hoạt động tốt rồi, bạn có thể **merge** hoặc **rebase** để gộp nó vào nhánh chính. Để tạo nhánh mới, sử dụng câu lệnh git checkout -b <branch\_name>

```
$ git checkout -b new_branch
```

```
Switched to a new branch 'new_branch'
```

Sử dụng git branch để kiểm tra các nhánh hiện có:

```
$ git branch  
  
master  
  
* new_branch
```

Dấu \* để chỉ rõ hiện tại mình đang ở nhánh nào.

Ở đây, mặc định, branch đầu tiên của mọi git repo đều có tên là master (một số nhóm người có thể sử dụng tên thay thế như main, primary). Nhưng bất kể tên gì, bạn nên để nhánh master làm luồng chạy chính cho chương trình của mình, coi như phiên bản chính thức của project.

## **6. Tạo một kho chứa trên Github**

Nếu bạn chỉ muốn theo dõi code của mình ở local, bạn không cần sử dụng GitHub. Nhưng nếu bạn muốn làm việc với một nhóm, bạn có thể sử dụng GitHub để phối hợp cùng mọi người.

Để tạo một repo trên GitHub, đăng nhập tài khoản mà bạn đã tạo ở mục 1 và đi đến trang chủ. Bạn có thể tìm thấy lựa chọn "New repository" khi click vào dấu "+" ở góc phải trên màn hình. Sau khi chọn, GitHub sẽ yêu cầu bạn nhập tên repo và mô tả (nếu muốn).

Bạn nên tích vào **Add a README file** và **Add .gitignore** (nếu chọn mục này bạn sẽ cần chọn thêm ngôn ngữ mình sử dụng) để sử dụng cho project của mình. Sau đó, **Create repository** để tạo thôi. Vậy là ta đã có remote repository rồi. Để kết nối local repo với remote repo ta sử dụng git remote add <link\_git\_clone>

```
$ git remote add origin https://github.com/thaidoandat/abcdef.git
```

```
$ git push origin master
```

```
Enumerating objects: 3, done.
```

```
Counting objects: 100% (3/3), done.
```

```
Writing objects: 100% (3/3), 217 bytes | 217.00 KiB/s, done.
```

```
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
```

```
remote:
```

```
remote: Create a pull request for 'master' on GitHub by visiting:
```

```
remote:   https://github.com/thaidoandat/abcdef/pull/new/master
```

```
remote:
```

```
To https://github.com/thaidoandat/abcdef.git
```

```
* [new branch]    master -> master
```

Link ở câu lệnh git remote add bạn có thể lấy bằng cách nhấn vào Code -> HTTPs -> copy link

### 7. Push branch lên GitHub

Bây giờ chúng ta sẽ đẩy commit trong branch của bạn vào repo GitHub. Điều này cho phép người khác xem những thay đổi bạn đã thực hiện. Nếu chúng được chủ sở hữu kho lưu trữ chấp thuận, các thay đổi sau đó có thể được hợp nhất vào nhánh chính. Để đẩy các thay đổi lên một nhánh mới trên GitHub, bạn chạy git push origin branch\_name. GitHub sẽ tự động tạo nhánh cho bạn trên remote repo:

```
$ git push origin new_branch  
  
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0  
  
remote:  
  
remote: Create a pull request for 'new_branch' on GitHub by visiting:  
  
remote:   https://github.com/thaidoandat/abcdef/pull/new/new_branch  
  
remote:  
  
To https://github.com/thaidoandat/abcdef.git  
  
* [new branch]    new_branch -> new_branch
```

Nếu bạn refresh GitHub, bạn sẽ thấy dòng thông báo một branch với tên của bạn vừa push lên repo. Bạn có thể click vào "branches" để xem các branch của bạn. Còn bây giờ thì click vào **Compare & pull request** để tạo pull request thôi nào.

### 8. Tạo pull request (PR)

PR là một cách để báo với chủ sở hữu repo rằng bạn muốn thay đổi code của họ, dùng trong làm việc nhóm. Nó cho phép họ review code và chắc chắn rằng nó ổn trước khi nhập nó vào branch chính. Sau khi tạo PR, bạn có thể thấy một button màu xanh lá cây "Merge pull request" ở dưới cùng. Click vào button nghĩa là bạn sẽ merge thay đổi của mình vào nhánh chính, bạn nên làm điều này sau khi được sự chấp thuận của mọi người trong team.

### 9. Lấy code từ repo remote về local remote với git pull

Khi code ở repo remote khác với code ở local của bạn (do teammate push lên), bạn muốn nhập những thay đổi kia vào repo ở local, câu lệnh **git pull** là dành cho bạn.

```
$ git pull origin master
```

Khi pull về thành công, nó sẽ hiển thị tất cả các files đã thay đổi và cách mà nó thay đổi.