

Git Là Gì? Tổng hợp các lệnh Git cơ bản và nâng cao tất cả lập trình viên cần biết

📅 14/01/2021 | 👁 4418 Lượt xem



Nguyễn Hưng

🏠 Home > Tài Liệu Kỹ Thuật > Git Là Gì? Tổng hợp các lệnh Git cơ bản và nâng cao tất cả lập trình viên cần biết

🔍 Tìm kiếm

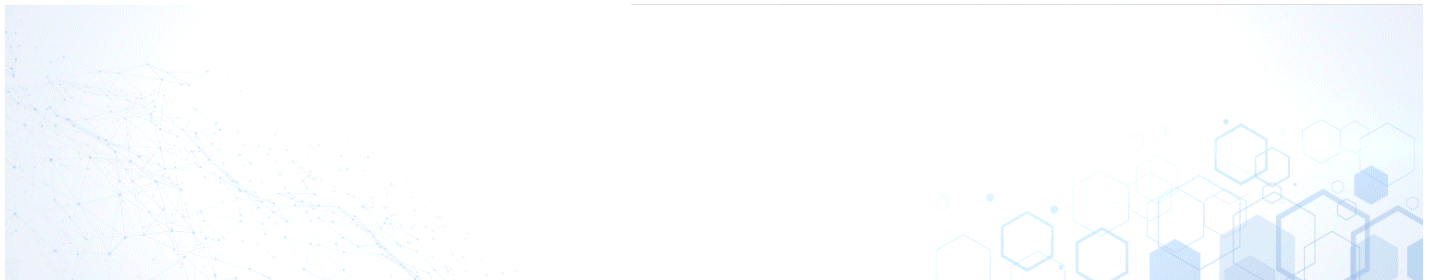
Bạn đang quan tâm về **Git là gì** và một số cách thức hoạt động của Git mà bạn cần nên biết. Vì đây là một công cụ khá hữu ích trong quá trình làm việc của bạn khi phải sử dụng nhiều với giao diện code. Hãy theo dõi bài viết dưới đây của Vietnix nhé!

| Git là gì?



Git là gì?

Git là một hệ thống kiểm soát phiên bản (**Version Control System**) mã nguồn mở miễn phí. Nó theo dõi các project và file khi chúng thay đổi theo thời gian. Cùng với đó là sự trợ giúp của những người đóng góp khác nhau. Sau khi nắm được định nghĩa **Git là gì**, dưới đây sẽ giúp bạn hiểu rõ hơn về Version Control System. Cùng tìm hiểu nhé.



Version Control System là gì?

Version Control System viết tắt là (**VCS**) là hệ thống kiểm soát các phiên bản phân tán mã nguồn mở. VCS sẽ theo dõi và lưu trữ tất cả các file trong toàn bộ dự án và ghi lại lịch sử thay đổi theo thời gian. Mỗi sự thay đổi sẽ được lưu lại thành một version (phiên bản).

Mỗi phiên bản sẽ bao gồm các thành phần: nội dung thay đổi, ngày và giờ sửa đổi, tác giả thay đổi, tên phiên bản,... hỗ trợ cho các lập trình viên tiện theo dõi và xem lại các danh sách thay đổi của file theo dòng thời gian.

>> Xem thêm: [Gitlab là gì?](#)

| Version Control System có tác dụng gì?

Git vốn đã là một (**Version Control System -VCS**) nên Git cũng sẽ ghi nhớ và lưu lại toàn bộ thay đổi trên source code của bạn. Những thành phần Git có thể ghi lại như: thêm code ở đâu, sửa code ở dòng nào, người thay đổi là ai,... tất cả sẽ được Git ghi lại và lưu trữ nó.

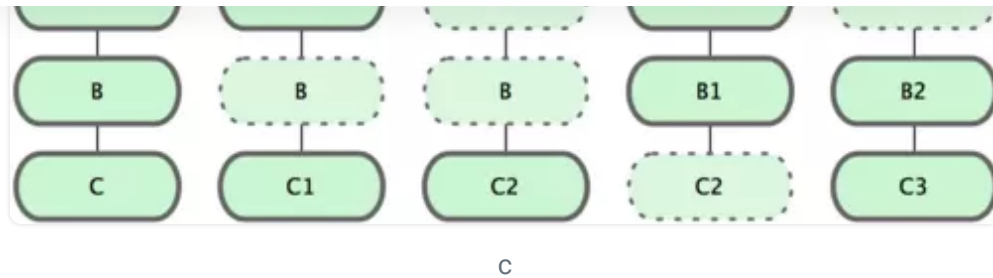
Có 2 lợi ích cơ bản và quan trọng của VCS là:

- Lưu lại toàn bộ lịch sử thay đổi của phiên bản. Giúp lập trình viên theo dõi và khôi phục dễ dàng sau này.
- Việc chia sẻ code có thể public cho bất kỳ ai, và cũng có thể để chế độ private cho những người có thẩm quyền được truy cập và tải code về. Giúp việc chia sẻ đơn và dễ dàng với Git.

| Cách thức hoạt động của Git là gì?

Hầu hết các hệ thống đều lưu trữ thông tin dưới dạng danh sách các thay đổi của file. Ví dụ như các hệ thống: CVS, Subversion, Perforce,...

Git coi các thông tin lưu trữ thành một tập hợp snapshot (ảnh chụp toàn bộ nội dung của file tại thời điểm đó). Mỗi khi commit thì Git sẽ chụp và tạo ra một snapshot cùng một tham chiếu tới snapshot đó.



Để bắt đầu Git, chuyển đến terminal và chạy lệnh sau trong thư mục dự án của bạn để khởi tạo một thư mục dự án.

```
git init
```

Chạy lệnh sau để thêm file cho Git theo dõi. Thao tác này sẽ thêm các file này vào staging area.

```
git add <filename_one>
```

Chạy lệnh sau để commit các thay đổi của bạn đối với các file này.

```
git commit <filename_one>
```

Và chúng ta có thể push các thay đổi của mình sau khi hoàn tất.

```
git push
```

Cùng với việc thực hiện thêm bất kỳ thay đổi nào trong branch master sẽ yêu cầu các thay đổi này phải được commit lại.

Tính năng vượt trội của Git so với SVN

Git là hệ thống quản lý mã nguồn theo hướng phân tán nên Git mang lại những lợi thế lớn trong việc hỗ trợ teamwork, phân chia task và tổng hợp code dễ dàng hơn.

Sắp xếp công việc hiệu quả

Sử dụng nhiều task linh hoạt

Như đã đề cập ở trên, thì khi sử dụng Git nó mang lại cho việc phân chia và sắp xếp các task dễ dàng. Vì vậy, bạn cũng có thể dễ dàng thực hiện nhiều task cùng một lúc đơn giản hơn.

Thử nghiệm ý tưởng mới

Bạn có thể thử nghiệm nhiều tính năng mới và được tách biệt ra khỏi dự án chính. Điều này, giúp bạn có thể thử nghiệm nhiều ý tưởng sáng tạo và cũng là để kiểm tra cho dự án một cách chính chu hơn. Nâng cao được chất lượng code cho dự án chính.

Git được đánh giá rất cao trong ngành, bạn nên tìm hiểu về Git là gì? Cũng như cách sử dụng nó để đem lại hiệu quả cao trong công việc của mình.

Git có thể làm offline

Git ngày càng ưu việt hơn so với hệ thống quản lý phiên bản tập trung (SVN). Git có thể cho phép bạn làm việc offline trong một khoảng thời gian. Bạn chỉ cần kết nối Internet khi làm việc nhóm, hoặc lưu lịch sử commit code lên remote repos. Còn SVN khi sử dụng bạn cần kết nối đến máy chủ SVN.

Lưu trữ thông tin

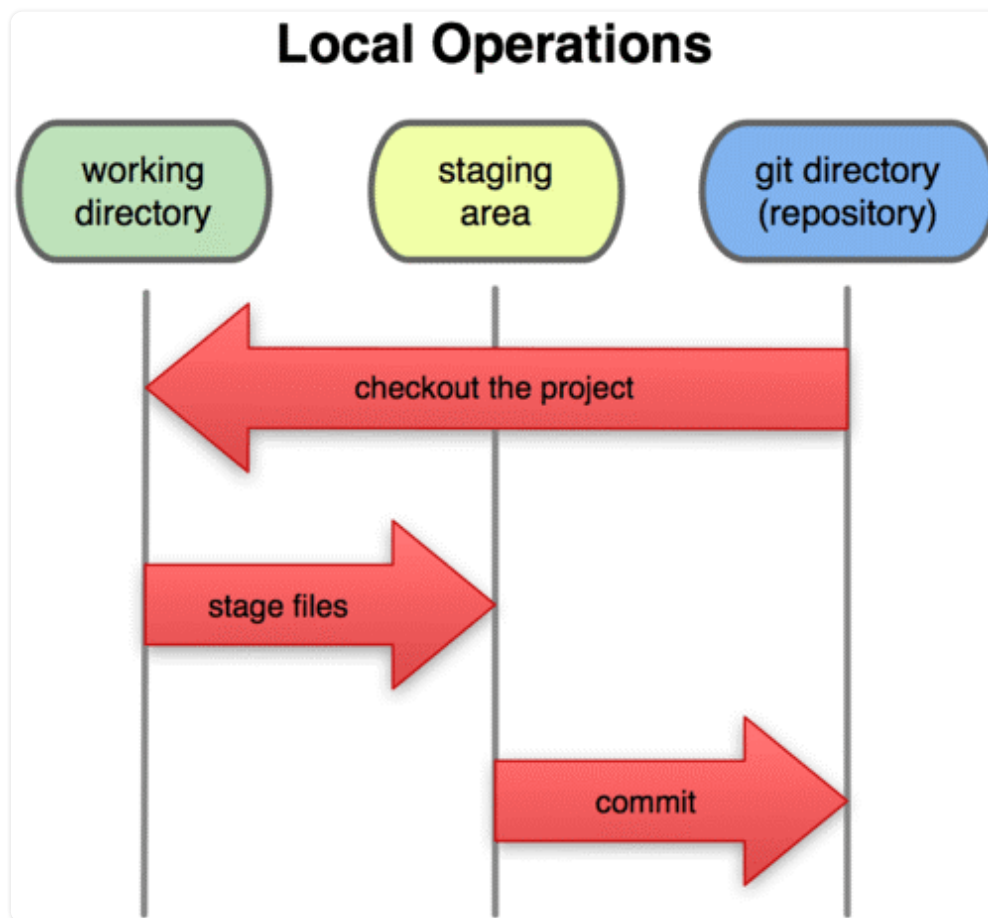
Git lưu trữ thông tin mà bạn có thể sử dụng nhiều điều thú vị để viết lại lịch sử commit. Khi tách nhánh, Git sử dụng 41 bytes cho một nhánh mới, giúp tiết kiệm không gian lưu trữ mà vẫn đảm bảo tốt nhu cầu làm việc. Thường SVN sẽ copy toàn bộ source code thành một bản mới khi tách nhóm.

Git miễn phí

Ngoài những tính năng ở trên, thì Git sử dụng phổ biến cũng vì là Git miễn phí. Tất cả mọi người đều có thể sử dụng những chức năng cơ bản của Git mà không cần bất kì cơ sở hạ tầng server nào.

Quy trình xử lý công việc (workflow) trên Git là gì?

Khi sử dụng Git bạn nên lưu ý là dùng Git giúp bạn quản lý mã nguồn mà không thể chỉnh code trong Git. Bạn có thể thực hiện các công việc của mình trên môi trường làm việc với các chức năng tiện ích trên IDE của ngôn ngữ lập trình khác. Quy trình xử lý công việc trên Git sẽ như sau:



Quy trình xử lý các công việc của Git

Lợi ích của Git là gì?

Git là gì? Chắc bạn cũng có thể nắm qua. Thì thường các lập trình viên sẽ làm các dự án song song, nên sử dụng Git là cần thiết để đảm bảo việc quản lý và không bị xung đột code giữa các thành viên khác.

- Dễ phân nhánh các branch, có thể giúp quy trình làm việc được hiệu quả hơn.
- Bạn có thể clone mã nguồn từ kho lưu trữ hoặc có thể clone một file thay đổi nào đó, hay các nhánh nào đó.
- Deployment sản phẩm của bạn một cách dễ dàng.

Các thuật ngữ quan trọng trong Git

1. Kho lưu trữ (Repository)

Kho lưu trữ (thường được gọi là repo) là một tập hợp các mã nguồn. Repo chứa các commit của dự án hoặc một tập hợp các tham chiếu đến các commit (ví dụ như heads).

2. Commit

Một commit ghi lại một thay đổi hoặc một loạt các thay đổi mà bạn đã thực hiện đối với một file trong repo. Một commit có hash SHA1 duy nhất được sử dụng để theo dõi các file đã thay đổi trong quá khứ. Git History là danh sách một loạt các commit. Sử dụng lệnh commit kết hợp với lệnh git add để cho git biết những thay đổi của bạn và lưu vào kho lưu trữ repository.

3. Branch

Một branch về cơ bản là một tập hợp các mã thay đổi duy nhất với một tên duy nhất. Mỗi repo có thể có một hoặc nhiều branch. Branch chính – branch mà tất cả các thay đổi cuối cùng được merge vào – được gọi là branch master. Đây là phiên bản làm việc chính thức cho dự án của bạn và là phiên bản mà bạn sẽ thấy khi truy cập kho dự án tại github.com/yourname/projectname.

4. Checkout

Bạn có thể sử dụng lệnh `git checkout` để chuyển các branch. Bằng cách nhập git checkout sau tên branch mà bạn muốn chuyển đến hoặc nhập git master để trở về branch chính (master branch).

thay đổi các thay đổi mà không ảnh hưởng đến kết quả của dự án.

6. Fetch

Sử dụng lệnh `git fetch` để tìm nạp các bản sao và tải xuống các tệp branch vào máy tính của bạn. Có thể sử dụng nó lưu các thay đổi mới nhất vào repository và có thể tìm nạp branch cùng một lúc.

7. Head

Head đại diện cho commit mới nhất của repository mà bạn đang làm việc và commit ở đầu của một branch được gọi là head.

8. Index

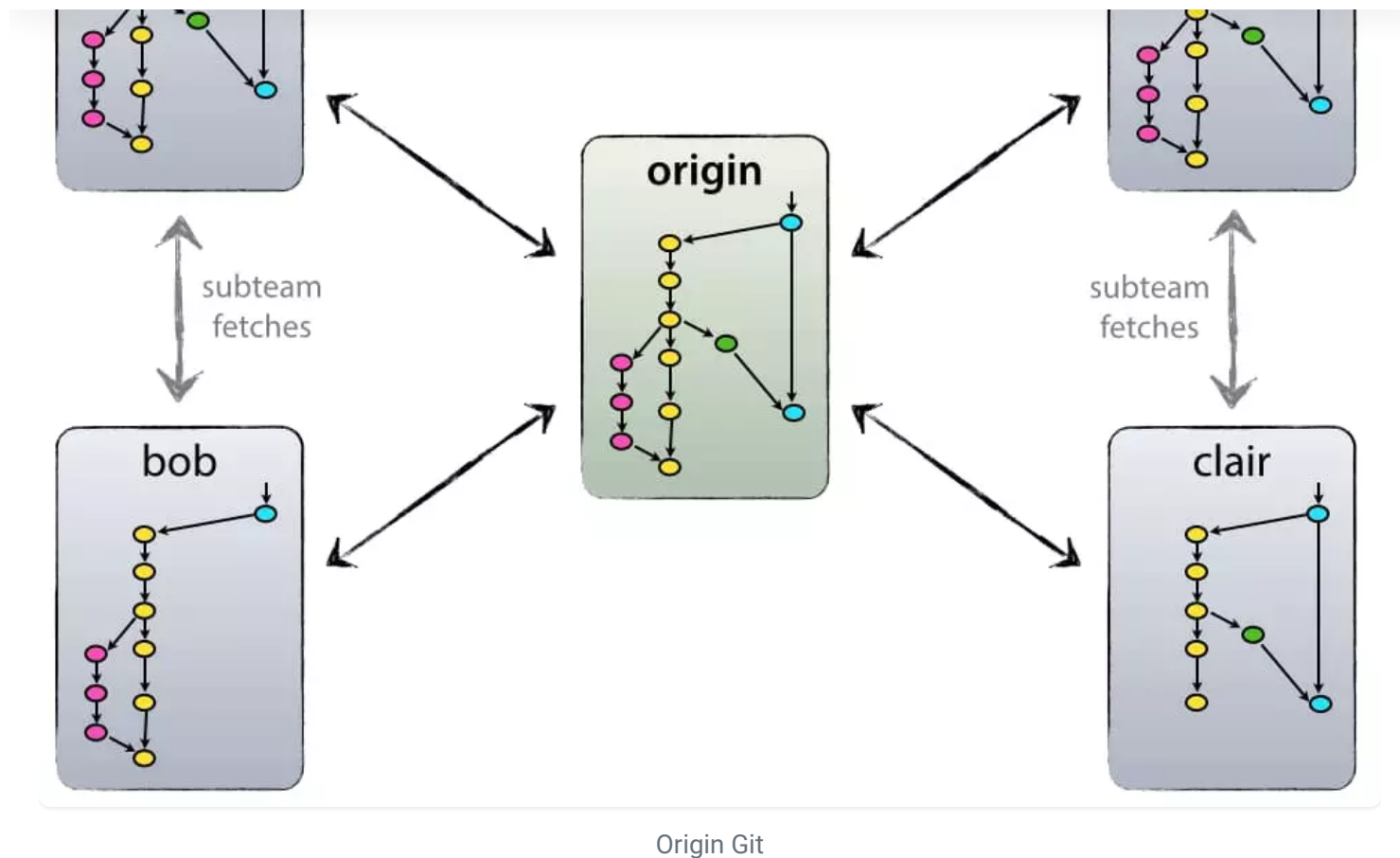
Khi sử dụng mà bạn thêm, xóa hoặc thay đổi file thì nó vẫn nằm trong mục index cho đến khi bạn sẵn sàng commit các thay đổi. Bạn dùng lệnh `git status` để xem nội dung index của bạn.

9. Merge

Lệnh git kết hợp với các yêu cầu kéo (pull request) để thêm các thay đổi từ nhánh này sang nhánh khác.

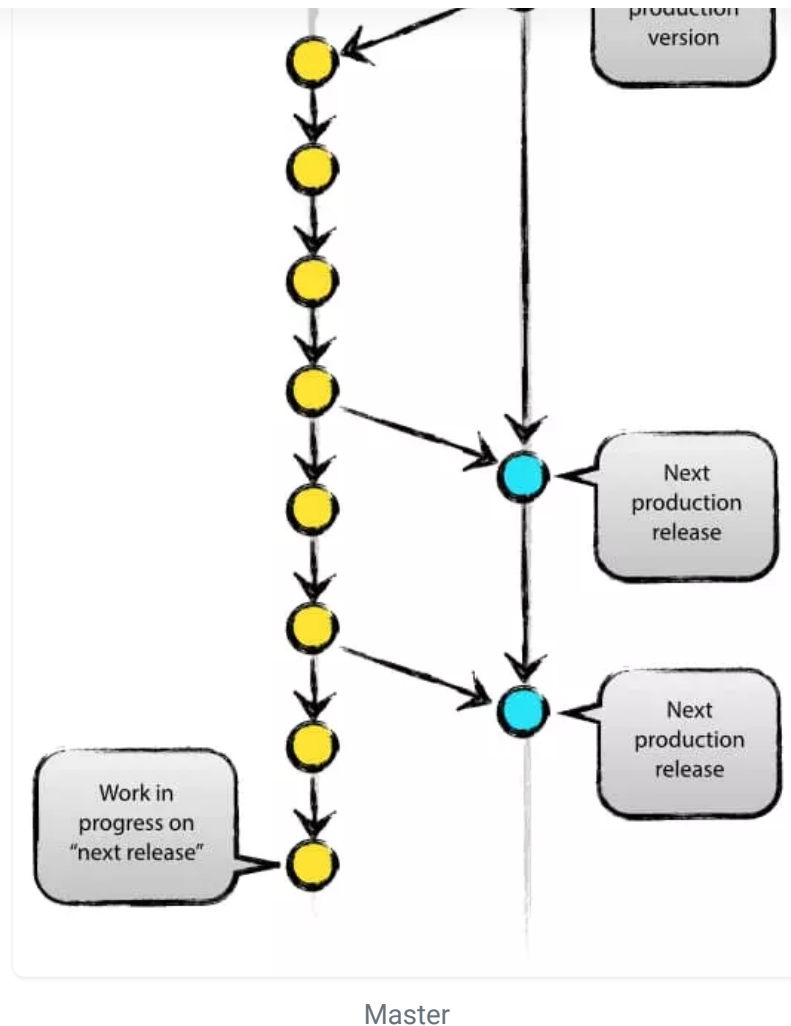
10. Origin

Là phiên bản mặc định của repository và origin đóng vai trò đặc biệt để liên lạc với nhánh chính. Lệnh `git push origin master` để đẩy các thay đổi cục bộ đến nhánh chính.



11. Master

Master là nhánh chính của tất cả các repository, nó bao gồm cả những thay đổi gần đây nhất.



12. Pull

Pull request thể hiện cho banjc ác đề xuất thay đổi trong nhánh chính. Khi bạn làm việc với một nhóm, bạn có thể tạo các pull request để yêu cầu người bảo trì kho lưu trữ xem xét các thay đổi và hợp nhất chúng.

13. Push

Lệnh `git push` dùng để cập nhật các nhánh từ xa với những thay đổi mới nhất mà bạn mới commit.

nhánh gốc (origin branch) của chúng và các remote khác trong kho lưu trữ.

15. Rebase

git rebase cho phép bạn phân tách, di chuyển và thoát commit. Và cũng có thể sử dụng nó để kết hợp hai nhánh lại với nhau.

16. Stash

Nếu bạn muốn loại bỏ các thay đổi khỏi index của bạn và xáo stashes chúng đi sau thì bạn có thể sử dụng lệnh sau:

```
git stash
```

Tiện lợi cho bạn khi bạn muốn tạm dừng công việc hiện tại và làm công việc khác trong một khoảng thời gian. Bạn có thể đặt stash nhiều hơn bộ thay đổi ở cùng một thời điểm.

17. Tags

Đối với tags, thì nó sẽ cung cấp cho bạn một cách để theo dõi commit quan trọng.

18. Upstream

Upstream đề cập đến nơi bạn push các thay đổi của mình và thường là các nhánh chính (master branch).

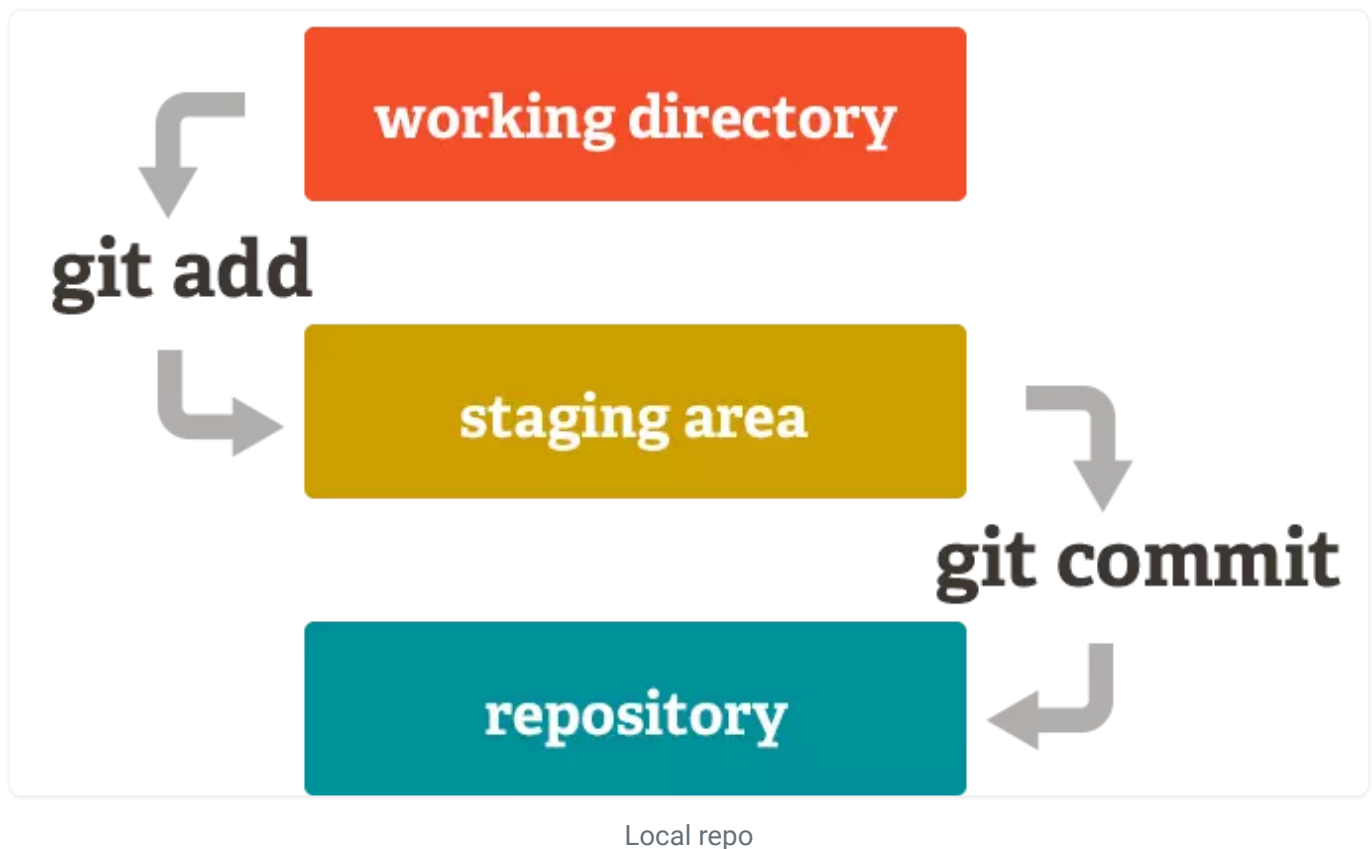
19. Working directory, staging area và local repo

Với mỗi local repo có ba virtual zone khác nhau. Đó là:

- Working Directory
- Staging area

area đôi khi còn được gọi là index.

Sau khi các thay đổi hoàn tất, staging area sẽ chứa một hoặc nhiều file cần được commit. Việc tạo một commit sẽ khiến Git lấy mã mới từ staging area và đưa commit vào repo chính. Sau đó commit này sau đó được chuyển đến commit area.



Các lệnh Git cơ bản

1. Git config

- **Công dụng:** Hai cài đặt quan trọng của git là user name và email

Có nhiều cách để điều chỉnh git config, có thể tùy chỉnh kết quả output màu và thay đổi hành vi git status. Bạn có thể tìm hiểu git config trong [tài liệu Git](#) chính thức.

```
# Running git config globally
$ git config --global user.email "my@emailaddress.com"
$ git config --global user.name "Example"
# Running git config on the current repository settings
$ git config user.email "my@emailaddress.com"
$ git config user.name "Example"
```

2. Git init

- **Công dụng:** Dùng để tạo một kho lưu trữ (repository) và 1 dự án (project) mới.
- **Sử dụng:** Dùng lệnh `git init` trong thư mục gốc của dự án.

3. Git add

- **Công dụng:** Thêm các file vào stage/index. Một số cách khác có thể sử dụng `git add` bằng cách thêm toàn bộ thư mục, các file cụ thể.
- **Sử dụng:**

```
$ git add <file or directory name>
```

- **Trong thực tế:**

```
# To add all files not staged:
$ git add .
# To stage a specific file:
$ git add index.html
# To stage an entire directory:
$ git add css
```

4. Git commit

```
# Adding a commit with message
$ git commit -m "Commit message in quotes"
```

- **Thực tế:**

```
$ git commit -m "My first commit message"
[SecretTesting 0254c3d] My first commit message
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 homepage/index.html
```

5. Git status

- **Công dụng:** Sử dụng lệnh này để trả về trạng thái tại kho lưu trữ (repository). `git status` sẽ trả về nhánh làm việc hiện tại của bạn. Nếu một file nằm trong staging area nhưng không được commit thì nó sẽ hiển thị với git status. Hoặc nếu không có thay đổi nào nó sẽ trả về *nothing to commit, working directory clean*

- **Sử dụng:**

```
$ git status
```

- **Thực tế:**

```
# Message when files have not been staged (git add)
$ git status
On branch SecretTesting
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    homepage/index.html

# Message when files have been not been committed (git commit)
$ git status
On branch SecretTesting
Your branch is up-to-date with 'origin/SecretTesting'.
```

```
# Message when all files have been staged and committed
$ git status
On branch SecretTesting
nothing to commit, working directory clean
```

6. Git branch

- **Công dụng:** Để xác định nhánh nào trong local repository, thêm hoặc xóa một nhánh mới.
- **Sử dụng:**

```
# Create a new branch
$ git branch <branch_name>

# List all remote or local branches
$ git branch -a

# Delete a branch
$ git branch -d <branch_name>
```

- **Trong thực tế thì:**

```
# Create a new branch
$ git branch new_feature

# List branches
$ git branch -a
* SecretTesting
  new_feature
  remotes/origin/stable
  remotes/origin/staging
  remotes/origin/master -> origin/SecretTesting

# Delete a branch
```

- **Công dụng:** Sử dụng `git checkout` để chuyển đổi các chi nhánh.
- **Sử dụng:**

```
# Checkout an existing branch
$ git checkout <branch_name>

# Checkout and create a new branch with that name
$ git checkout -b <new_branch>
```

- **Thực tế:**

```
# Switching to branch 'new_feature'
$ git checkout new_feature
Switched to branch 'new_feature'

# Creating and switching to branch 'staging'
$ git checkout -b staging
Switched to a new branch 'staging'
```

8. Git merge

- **Công dụng:** Hợp nhất các nhánh với nhau, sử dụng `git merge` để kết hợp các thay đổi từ nhánh này sang nhánh khác.
- **Sử dụng:**

```
# Merge changes into current branch
$ git merge <branch_name>
```

- **Trong thực tế:**

```
# Merge changes into current branch
$ git merge new_feature
Updating 0254c3d..4c0f37c
```



```
create mode 100644 homepage/index.html
```

9. Git remote

- **Công dụng:** Để kết nối repository với kho lưu trữ từ xa.
- **Sử dụng:**

```
# Add remote repository
$ git remote <command> <remote_name> <remote_URL>

# List named remote repositories
$ git remote -v
```

- **Trong thực tế:**

```
# Adding a remote repository with the name of beanstalk
$ git remote add origin
git@account_name.git.beanstalkapp.com:/account_name/repository_name.git

# List named remote repositories
$ git remote -v
origin git@account_name.git.beanstalkapp.com:/account_name/repository_name.git
(fetch)
origin git@account_name.git.beanstalkapp.com:/account_name/repository_name.git
(push)
```

10. Git clone

- **Công dụng:** Để tạo một bản sao làm việc cục bộ với kho lưu trữ từ xa. Sử dụng `git clone` để sao chép và tải kho lưu trữ về máy tính. Sao chép giống với Git init khi làm việc với kho lưu trữ từ xa.
- **Sử dụng:**

```
$ git clone <remote_URL>
```

```
remote: Counting objects: 5, done.  
remote: Compressing objects: 100% (3/3), done.  
remote: Total 5 (delta 0), reused 0 (delta 0)  
Receiving objects: 100% (5/5), 3.08 KiB | 0 bytes/s, done.  
Checking connectivity... done.
```

11. Git pull

- **Công dụng:** Chạy `git pull` để tải phiên bản mới nhất của repository. Thao tác với lệnh này kéo các thay đổi từ kho lưu trữ từ xa sang máy tính cục bộ.
- **Sử dụng:**

```
$ git pull <branch_name> <remote_URL/remote_name>
```

- **Trong thực tế:**

```
# Pull from named remote  
$ git pull origin staging  
From account_name.git.beanstalkapp.com:/account_name/repository_name  
* branch          staging    -> FETCH_HEAD  
* [new branch]     staging    -> origin/staging  
Already up-to-date.  
  
# Pull from URL (not frequently used)  
$ git pull  
git@account_name.git.beanstalkapp.com:/account_name/repository_name.git staging  
From account_name.git.beanstalkapp.com:/account_name/repository_name  
* branch          staging    -> FETCH_HEAD  
* [new branch]     staging    -> origin/staging  
Already up-to-date.
```

12. Git push

```
$ git push <remote_URL/remote_name> <branch>
```

```
# Push all local branches to remote repository
```

```
$ git push -all
```

- **Trong thực tế:**

```
# Push a specific branch to a remote with named remote
```

```
$ git push origin staging
```

```
Counting objects: 5, done.
```

```
Delta compression using up to 4 threads.
```

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

```
Writing objects: 100% (5/5), 734 bytes | 0 bytes/s, done.
```

```
Total 5 (delta 2), reused 0 (delta 0)
```

```
To git@account_name.git.beanstalkapp.com:/account_name/repository_name.git  
ad189cb..0254c3d SecretTesting -> SecretTesting
```

```
# Push all local branches to remote repository
```

```
$ git push --all
```

```
Counting objects: 4, done.
```

```
Delta compression using up to 4 threads.
```

```
Compressing objects: 100% (4/4), done.
```

```
Writing objects: 100% (4/4), 373 bytes | 0 bytes/s, done.
```

```
Total 4 (delta 2), reused 0 (delta 0)
```

```
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
```

```
To git@account_name.git.beanstalkapp.com:/account_name/repository_name.git  
0d56917..948ac97 master -> master  
ad189cb..0254c3d SecretTesting -> SecretTesting
```

Các lệnh Git nâng cao

1. Git stash

```
$ git stash -u
```

```
# Bring stashed work back to the working directory
```

```
$ git stash pop
```

- **Trong thực tế:**

```
# Store current work
```

```
$ git stash -u
```

```
Saved working directory and index state WIP on SecretTesting: 4c0f37c Adding new  
file to branch
```

```
HEAD is now at 4c0f37c Adding new file to branch
```

```
# Bring stashed work back to the working directory
```

```
$ git stash pop
```

```
On branch SecretTesting
```

```
Your branch and 'origin/SecretTesting' have diverged,  
and have 1 and 1 different commit each, respectively.
```

```
(use "git pull" to merge the remote branch into yours)
```

```
Changes not staged for commit:
```

```
(use "git add <file>..." to update what will be committed)
```

```
(use "git checkout -- <file>..." to discard changes in working directory)
```

```
modified:   index.html
```

```
no changes added to commit (use "git add" and/or "git commit -a")
```

```
Dropped refs/stash@{0} (3561897724c1f448ae001edf3ef57415778755ec)
```

2. Git log

- **Công dụng:** Để hiển thị lịch sử commit theo thời gian cho một repository.

- **Sử dụng:**

```
# Show entire git log
```

```
# Show git log based on commit author
```

```
$ git log --<author>="Author Name"
```

- **Trong thực tế:**

```
# Show entire git log
```

```
$ git log
```

```
commit 4c0f37c711623d20fc60b9cbcf393d515945952f
```

```
Author: Brian Kerr <my@emailaddress.com>
```

```
Date: Tue Oct 25 17:46:11 2016 -0500
```

```
Updating the wording of the homepage footer
```

```
commit 0254c3da3add4ebe9d7e1f2e76f015a209e1ef67
```

```
Author: Ashley Harpp <my@emailaddress.com>
```

```
Date: Wed Oct 19 16:27:27 2016 -0500
```

```
My first commit message
```

```
# Show git log with date parameters
```

```
$ git log --before="Oct 20"
```

```
commit 0254c3da3add4ebe9d7e1f2e76f015a209e1ef67
```

```
Author: Ashley Harpp <my@emailaddress.com>
```

```
Date: Wed Oct 19 16:27:27 2016 -0500
```

```
My first commit message
```

```
# Show git log based on commit author
```

3. Git rm

- **Công dụng:** Xóa file hoặc folder khỏi index (staging area). Với `git rm` có hai tùy chọn cần lưu ý: buộc và lưu vào bộ nhớ cache. Lệnh được lưu trong bộ nhớ cache sẽ xóa file và folder khỏi index.

```
# To delete a file (force):  
$ git rm -f <file name>  
  
# To remove an entire directory from the working index (cached):  
$ git rm -r --cached <directory name>  
  
# To delete an entire directory (force):  
$ git rm -r -f <file name>
```

- Trong thực tế:

```
# To remove a file from the working index:  
$ git rm --cached css/style.css  
rm 'css/style.css'  
  
# To delete a file (force):  
$ git rm -f css/style.css  
rm 'css/style.css'  
  
# To remove an entire directory from the working index (cached):  
$ git rm -r --cached css/  
rm 'css/style.css'  
rm 'css/style.min.css'  
  
# To delete an entire directory (force):  
$ git rm -r -f css/  
rm 'css/style.css'  
rm 'css/style.min.css'
```

| Lời khuyên khi sử dụng Git trong công việc

1. Git cheat sheet

- <https://git-scm.com/docs/gittutorial>
- <https://gitsheet.wtf/>
- <http://ndpsoftware.com/git-cheatsheet.html>
- <https://gitexplorer.com/>

2. Nên commit thường xuyên

Nên tách nhỏ commit và thực hiện commit thường xuyên nhất. Nó có ích cho các thành viên trong nhóm để dàng tích hợp công việc với nhau mà không gặp phải xung đột hợp nhất.

3. Test rồi mới commit

Lưu ý không được commit nếu chưa hoàn tất quá trình. Ngoài ra trước khi chia sẻ những thay đổi của bạn với người khác phải được test kĩ càng.

4. Viết ghi chú khi commit

Viết ghi chú khi commit để các thành viên trong nhóm nhận biết được loại thay đổi bạn đã thực hiện.

Một số lưu ý khi làm việc với Git

- Hãy lưu ý không nên commit nếu chưa hoàn tất process. Bạn nên test các thay đổi của bạn trước khi public.
- Nên commit thường xuyên bằng cách tách nhỏ từng commit của bạn và commit thường xuyên. Điều này sẽ hỗ trợ cho các thành viên trong nhóm để dàng tích hợp các công việc của họ hơn.
- Viết các ghi chú commit để cho các thành viên khác trong nhóm biết các thay đổi của bạn đã thực hiện. Nếu thực hiện càng cụ thể và chi tiết thì càng tốt.
- Bạn cần tận dụng các lợi ích của branch để giúp bạn dễ dàng theo dõi các dòng phát triển khác nhau.

- [GitHub là gì.](#)
- [GitLab là gì.](#)

Lời kết

Trên đây là bài viết khá đầy đủ và chi tiết để giúp bạn hiểu hơn về **Git là gì**, cùng một số lệnh mà bạn có thể tham khảo để sử dụng trong quá trình làm việc của mình đạt được hiệu suất cao hơn. Hy vọng bài viết trên của Vietnix sẽ hỗ trợ kiến thức và thông tin hữu ích cho bạn. Chúc các bạn thành công.

Chia sẻ bài viết



Đánh giá ★★★★★ 5/5 - (4 bình chọn)

Nguyễn Hưng



Mình là Bo - admin của Quản Trị Linux. Mình đã có 10 năm làm việc trong mảng System, Network, Security và đã trải nghiệm qua các chứng chỉ như CCNP, CISSP, CISA, đặc biệt là chống tấn công DDoS. Gần đây mình trải nghiệm thêm Digital Marketing và đã hoàn thành chứng chỉ CDMP của PearsonVUE. Mình rất thích được chia sẻ và hỗ trợ cho mọi người, nhất là các bạn sinh viên. Hãy kết nối với mình nhé!

Đăng ký nhận tin

[Đăng ký](#)

Bài viết liên quan

[TÀI LIỆU KỸ THUẬT](#)

Live stream là gì? Cách live stream hiệu quả trên một số mạng xã hội phổ biến

[TÀI LIỆU KỸ THUẬT](#)

Multicast là gì? Cách thức Multicast hoạt động

[TÀI LIỆU KỸ THUẬT](#)

Card mạng là gì? Vai trò và nguyên tắc sử dụng network card

[TÀI LIỆU KỸ THUẬT](#)

Repeater là gì? Vì sao nên sử dụng wifi Repeater?

[TÀI LIỆU KỸ THUẬT](#)[TÀI LIỆU KỸ THUẬT](#)

Bình luận

☒ Theo dõi ▼



Hãy trở thành người đầu tiên bình luận!

B *I* U “ ” </> {} [+]



0 COMMENTS



1800 1093

Hotline liên hệ



Gọi lại cho tôi

Nhập số điện thoại, chúng tôi sẽ gọi lại



Gửi ticket

Gửi yêu cầu đến trung tâm hỗ trợ



Live chat

Nhắn tin trực tiếp với chúng tôi

[Liên Hệ](#)

[Khuyến Mãi](#)

[Thông Báo](#)

[Sự Kiện](#)

[Blog](#)

[Thông Tin Thanh Toán](#)

[Chương Trình Affiliate](#)

DỊCH VỤ

[Hosting Giá Rẻ](#)

[Hosting Cao Cấp](#)

[Business Hosting](#)

[SEO Hosting](#)

[Email Hosting](#)

[Firewall Anti DDoS](#)

[Đăng Ký Tên Miền](#)

[SSL Certificate](#)

DỊCH VỤ MÁY CHỦ

[VPS Giá Rẻ](#)

[VPS Phổ Thông](#)

Thuê Máy Chủ

Thuê Chỗ Đặt Máy Chủ

THÔNG TIN DỊCH VỤ

Hướng Dẫn Thanh Toán

Hướng Dẫn Dịch Vụ

Cam Kết Dịch Vụ

Báo Cáo Lạm Dụng

Sang Nhượng Tên Miền

Quy Định Sử Dụng Tên Miền Quốc Tế

CHÍNH SÁCH & ĐIỀU KHOẢN

Chính Sách Bảo Mật

Chính Sách Khiếu Nại

Chính Sách Hoàn Tiền

Quy Định Chống Thư Rác

Điều Khoản Sử Dụng Dịch Vụ

CÔNG TY CỔ PHẦN GIẢI PHÁP VÀ CÔNG NGHỆ VIETNIX

Điện thoại: 1800 1093 – 07 088 44444 - 02836 22 33 11


✉ Email liên hệ: sales@vietnix.com.vn

KẾT NỐI VỚI VIETNIX

 Facebook

 LinkedIn

 Twitter

 Youtube

 Telegram

 Zalo

