**Những lệnh Git**

\* các khái niệm:

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 repositoty.

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

Có thể sử dụng lệnh:

 git checkout

để chuyển các branch. Bằng cách nhập git checkout sao tên branch mà bạn muốn chuyển đến hoặc nhập git master để trở về branch chính (master branch).

5. Fork

Fork là một bản sao của kho lưu trữ (repository). Có thể tận dụng các lợi ích của fork để chạy thử nghiệm các thay đổi mà không ảnh hưởng đến kết quả của dự án.

6. Fetch

Lệnh:

git fetch

để tìm nạp các bản sao và tải xuống các tệp branch vào máy tính. 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 thêm, xóa hoặc thay đổi file thì nó vẫn nằm trong mục index cho đến khi sẵn sàng commit các thay đổi.

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.

Origin Git

**11. Master**

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

Master

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.

14. Remote

Remote (kho lưu trữ từ xa) là một bản sao của một chi nhánh. Remote có thể giao tiếp ngược lại với 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

commit area

Working directory là nơi các file mới được tạo, file cũ bị xóa hoặc nơi thực hiện các thay đổi đối với các file đã có. Sau khi thay đổi được thực hiện, chúng sẽ được thêm vào staging area. Do đó, staging 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.

Local repo

**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 để điều chỉnh git config, có thể tùy chỉnh kết quả output màu và thay đổi hành vi git status.

Sử dụng: Khi bạn muốn kiểm tra trên và email trong cấu hình dùng lệnh:

$ git config <setting> <command>

Thực tế:

# 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

Công dụng: Dùng để ghi lại những thay đổi được thực hiện đối với file vào local repository. Để dễ dàng theo dõi commit và mỗi commit sẽ có một mã ID theo dõi duy nhất.

Sử dụng:

# 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 stagging area nhưng không được commit thì nó sẽ hiển thị với git status. Hoặc nếu không có thay đổi nào no 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'.

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

new file: homepage/index.html

# 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

$ git branch -d new\_feature

Deleted branch new\_feature (was 0254c3d).

7. Git checkout

Công dụng: Sử dụng git checkoutđể 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 thau 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

Fast-forward

homepage/index.html | 297 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++

1 file changed, 297 insertions(+)

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:/acccount\_name/repository\_name.git

# List named remote repositories

$ git remote -v

origin git@account\_name.git.beanstalkapp.com:/acccount\_name/repository\_name.git (fetch)

origin git@account\_name.git.beanstalkapp.com:/acccount\_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>

Trong thực tế:

$ git clone git@account\_name.git.beanstalkapp.com:/acccount\_name/repository\_name.git

Cloning into 'repository\_name'...

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:/acccount\_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

Công dụng: Dùng để gửi commit đến kho lưu trữ từ xa. git push sử dụng 2 tham số: kho lưu trữ từ xa và nhánh mà push dành cho.

Sử dụng:

$ 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:/acccount\_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:/acccount\_name/repository\_name.git

0d56917..948ac97 master -> master

ad189cb..0254c3d SecretTesting -> SecretTesting

Các lệnh Git nâng cao

1. Git stash

Công dụng: Dùng để lưu các thay đổi được thực hiện nó chưa ở trạng thái commit đến repository.

Sử dụng:

# Store current work with untracked files

$ 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

$ git log

# Show git log with date pameters

$ git log --<after/before/since/until>=<date>

# 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 pameters

$ 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

$ git log --author="Brian Kerr"

commit 4c0f37c711623d20fc60b9cbcf393d515945952f

Author: Brian Kerr <my@emailaddress.com>

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

Updating the wording of the homepage footer

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.

Sử dụng:

# To remove a file from the working index (cached):

$ git rm --cached <file name>

# 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 cheet sheet

Khi làm việc bạn cũng không thể nào nhớ toàn bộ các lệnh, thì bạn nên sử dụng công cụ hỗ trợ là Git Cheet Sheet để tìm kiếm được lệnh git cần thiết khi làm việc:

[h](https://rogerdudler.github.io/git-guide/)[ttps://rogerdudler.github.io/git-guide/](https://rogerdudler.github.io/git-guide/)

[htt](https://rogerdudler.github.io/git-guide/)[ps://git-scm.com/docs/gittutorial](https://git-scm.com/docs/gittutorial)

[https://gitsheet.wtf/](https://git-scm.com/docs/gittutorial)

[http://ndps](https://git-scm.com/docs/gittutorial)[oftware.com/git-cheatsheet.html](http://ndpsoftware.com/git-cheatsheet.html)

[https://gitexplorer.com/](http://ndpsoftware.com/git-cheatsheet.html)

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 dễ 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.