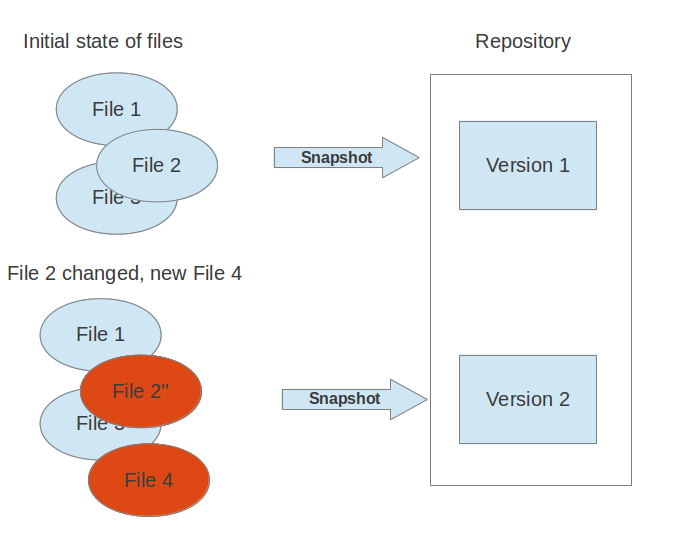
Day 1: Learn about Git

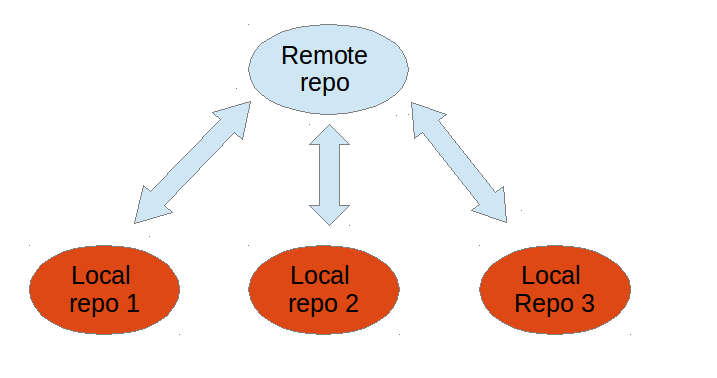
### git là hệ thống quản lý mã nguồn

### 1.1. What is a version control system(VCS)?

VCS cho ta theo dõi sự thay đổi của tập tin. Nó hỗ trợ việc tạo ra các phiên bản khác nhau trên 1 tập tin. Nó sẽ chụp nhanh 1 tập tin tại 1 thời điểm và có thể cho phép bạn chuyển đổi qua các thơi gian khác nhau. Những file này thường được lưu trú tại 1 nơi nó được gọi là repository



Đây là hình miêu tả:



+ thiết lập username và pass bằng lệnh:

$ **git config --global user.name *"<Tên người dùng>"***

$ **git config --global user.email *"<Địa chỉ mail>"***

## 2.Tạo repository

Có 2 cách tạo local repository mà mình có sẵn.

Cách thứ nhất là tạo repository hoàn toàn mới, cách thứ hai là sao chép remote repository rồi tạo.

+ tạo repository mới:

- tạo một thư mục và cd vào thư mục đó chạy lệnh git init

+commit file len.

===> những bước ở trên là chúng ta tạo reponsitory trên thư mục máy tính cá nhân

**Bước 2:**

Chia sẻ respository

- tạo resposity trên github

-push resposity từ local tới remote server

Để thêm remote repository, sử dụng lệnh remote. <name> là tên đăng ký, <url> là chỉ định URL của remote repository.

$ git remote add <name> <url>

Để push repository, sẽ sử dụng lệnh "push". <repository> là chỉ địa chỉ nơi sẽ push, <refspec> là chỉ branch sẽ push. Về branch thì sẽ trình bày chi tiết trong phần Phát triển.

$ git push <repository> <refspec>...

**Branch trong git:**

## **Branch là gì?**

Branch là cái dùng để phân nhánh và ghi lại luồng của lịch sử. Branch đã phân nhánh sẽ không ảnh hưởng đến branch khác nên có thể tiến hành nhiều thay đổi đồng thời trong cùng 1 repository.

**Ở đây sẽ giới thiệu phương pháp vận dụng đã sử dụng với 2 loại branch là branch chủ đề (Topic branch) và branch tích hợp (Intergration branch).**

**Branch có thể tạo bằng lệnh branch.**

$ git branch <branchname>

**để commit tới 1 branch vừa tạo thì đầu tiên ta phải checkout code và sau đó thực hiện commit lên lại thì code sẽ vào branch tương ứng.**

**Gitk: dùng để hiển thị giao diện để check data giữa local và server**

**để chuyển đổi 1 branch được chọn ta dùng : git checkout <name\_branch>**

**để merge 1 nhánh tới thư mục gốc ta dùng**

git merge <name\_branch>

+ Xóa 1 branch:

Để xóa 1 branch ta dùng lệnh : git branch -d <name\_branch>

Ví dụ : git branch -d isses1

Các lệnh cơ bản trong git

+Pull: Lệnh này sẽ tự động lấy toàn bộ dữ liệu từ remote repository và gộp vào cái branch hiện tại bạn đang làm việc.

#### git clone

Lệnh này sẽ sao chép toàn bộ dữ liệu trên repository và sao chép luôn các thiết lập về repository, tức là nó sẽ tự động tạo một master branch trên máy tính của bạn. Lệnh này chỉ nên sử dụng khi bạn cần tạo mới một Git mới trên máy tính với toàn bộ dữ liệu và thiết lập của một remote repository.

+git fetch

Lệnh này sẽ lấy toàn bộ dữ liệu từ remote repository nhưng sẽ cho phép bạn gộp thủ công vào một branch nào đó trên thư mục Git ở máy tính.

Sau khi fetch về thì ta có 2 cách để add code vào nhánh minh muốn:

-Cách 1: merge code. ==> nó là hành động không phá hủy và các branch hiện tại nó không thay đổi

ví dụ: git merge origin/master

- Cách 2: dùng rebase

==> nó sẽ viết lại cái project và tạo ra nhánh mới

===> pull = fetch + merge;

+push: dùng để đẩy dữ liệu từ local tới server

# Basic Git commands

Here is a list of some basic Git commands to get you going with Git.

For more detail, check out the  [Atlassian Git Tutorials](http://atlassian.com/git?utm_source=basic-git-commands&utm_medium=link&utm_campaign=git-microsite)  for a visual introduction to Git commands and workflows, including examples.

|  |  |  |
| --- | --- | --- |
| Git task | Notes | Git commands |
| [Tell Git who you are](https://www.atlassian.com/git/tutorials/setting-up-a-repository/git-config) | Configure the author name and email address to be used with your commits.  Note that Git [strips some characters](http://stackoverflow.com/questions/26159274/is-it-possible-to-have-a-trailing-period-in-user-name-in-git/26219423" \l "26219423) (for example trailing periods) from user.name. | git config --global user.name "Sam Smith"  git config --global user.email sam@example.com |
| [Create a new local repository](http://atlassian.com/git/tutorial/git-basics" \l "!init) |  | git init |
| [Check out a repository](http://atlassian.com/git/tutorial/git-basics" \l "!clone) | Create a working copy of a local repository: | git clone /path/to/repository |
| For a remote server, use: | git clone username@host:/path/to/repository |
| [Add files](http://atlassian.com/git/tutorial/git-basics" \l "!add) | Add one or more files to staging (index): | git add <filename>  git add \* |
| [Commit](http://atlassian.com/git/tutorial/git-basics" \l "!commit) | Commit changes to head (but not yet to the remote repository): | git commit -m "Commit message" |
| Commit any files you've added with git add, and also commit any files you've changed since then: | git commit -a |
| [Push](http://atlassian.com/git/tutorial/remote-repositories" \l "!push) | Send changes to the master branch of your remote repository: | git push origin master |
| [Status](http://atlassian.com/git/tutorial/git-basics" \l "!status) | List the files you've changed and those you still need to add or commit: | git status |
| [Connect to a remote repository](http://atlassian.com/git/tutorial/remote-repositories" \l "!remote) | If you haven't connected your local repository to a remote server, add the server to be able to push to it: | git remote add origin <server> |
| List all currently configured remote repositories: | git remote -v |
| [Branches](http://atlassian.com/git/tutorial/git-branches) | Create a new branch and switch to it: | git checkout -b <branchname> |
| Switch from one branch to another: | git checkout <branchname> |
| List all the branches in your repo, and also tell you what branch you're currently in: | git branch |
| Delete the feature branch: | git branch -d <branchname> |
| Push the branch to your remote repository, so others can use it: | git push origin <branchname> |
| Push all branches to your remote repository: | git push --all origin |
| Delete a branch on your remote repository: | git push origin :<branchname> |
| [Update from the remote repository](http://atlassian.com/git/tutorial/remote-repositories) | Fetch and merge changes on the remote server to your working directory: | git pull |
| To merge a different branch into your active branch: | git merge <branchname> |
| View all the merge conflicts:  View the conflicts against the base file:  Preview changes, before merging: | git diff  git diff --base <filename>  git diff <sourcebranch> <targetbranch> |
| After you have manually resolved any conflicts, you mark the changed file: | git add <filename> |
| Tags | You can use tagging to mark a significant changeset, such as a release: | git tag 1.0.0 <commitID> |
| CommitId is the leading characters of the changeset ID, up to 10, but must be unique. Get the ID using: | git log |
| Push all tags to remote repository: | git push --tags origin |
| [Undo local changes](http://atlassian.com/git/tutorial/undoing-changes) | If you mess up, you can replace the changes in your working tree with the last content in head:  Changes already added to the index, as well as new files, will be kept. | git checkout -- <filename> |
| Instead, to drop all your local changes and commits, fetch the latest history from the server and point your local master branch at it, do this: | git fetch origin  git reset --hard origin/master |
| Search | Search the working directory for foo(): | git grep "foo()" |

**+Switch checkout:**

dùng để chuyển đồi qua lại giữa các branch với nhau và có thể check code từ trước đó.

**+ Stash:**

khi chúng ta code 1 chức năng hay function nào đó chưa xong mà chúng ta phải làm 1 việc khác thì chúng ta có thể dùng: git stash

Để xem danh sách các stash đã lưu, gõ lệnh:

git stash list

Để phục hồi stash mới nhất vừa lưu nhưng không xoá nó:

git stash apply

Để phục hồi stash mới nhất vừa lưu đồng thời xoá nó ra khỏi danh sách stash:

git stash pop

Để phục hồi một stash bất kì:

git stash apply stash@{1} (với stash@{1} là tên của stash đó)

Để huỷ bỏ một stash, dùng lệnh:

git stash drop stash@{1} (với `stash@{1} là tên stash cần xoá)

Xem chi tiết một stash:

git stash show -p stash@{0}

+**config ssh key:**

[**https://help.github.com/articles/generating-ssh-keys/**](https://help.github.com/articles/generating-ssh-keys/)

**+ cai dat nhieu multiple key**

**dau tien chung ta can tao nhieu ssh key khac nhau**

Host github.com

HostName github.com

User git

IdentityFile ~/.ssh/id\_rsa

Host github.com-New

HostName github.com

User git

IdentityFile ~/.ssh/id\_rsa\_new

Ý nghĩa của file này là sao? Nếu liên kết đến Github của repos có host là github.com thì git sẽ dùng tập tin id\_rsa còn nếu liên kết này có host là github.com-New thì tạp tin id\_rsa\_new sẽ được sử dụng. Việc đặt tên github.com-New là hoàn toàn ngẫu nhiên, các bạn có thể đặt github.com-ABC hoặc example.com. Miễn sao phần HostName ở dưới phải đúng là Github.

Lưu lại tập tin cấu hình này, chúng ta sẽ tiếp tục kiểm tra và cấu hình remote URL cho các git repos đang có trên máy tính. Thông thường, Github sẽ cung cấp cho bạn đường link SSH đến repos có dạng: git@github.com:<tên tài khoản>/<tên repos>.git. Địa chỉ sau dấu @ chính là tương ứng với phần Host mà bạn đã cấu hình trong file config ở trên. Giả sử tôi clone project này về máy bằng lệnh sau:

PowerShell

$ git clone git@github.com:xuanchien/test.git

|  |  |
| --- | --- |
| 1 | $ git clone git@github.com:xuanchien/test.git |

thì sẽ tương ứng với sử dụng tài khoản xuanchien trên Github. Còn nếu dùng lệnh

PowerShell

$ git clone git@github.com-New:xuanchien/boko.git

|  |  |
| --- | --- |
| 1 | $ git clone git@github.com-New:xuanchien/boko.git |

thì sẽ tương ứng với tài khoản mới trên Github. Lưu ý sự khác biệt trong đường dẫn của 2 đường link, và nó phải tương ứng với Host đã được cấu hình trong file config