**Git config**

git config --global user.name "John Doe"

git config --global user.email "john@example.com"

--global được sử dụng để áp dụng cho tất cả các projects. Nếu bạn ko sử dụng --global thì settings sẽ chỉ dùng cho riêng project đó.

**Giúp Git bỏ qua file modes**

cd project/

git config core.filemode false

Câu lệnh trên hữu dụng khi chúng ta không cần quan tâm đến quyền truy cập files (ví dụ như khi sử dụng Windows).

**Liệt kê những settings đang sử dụng**

git config --list

**Khởi tạo Git repo cho code có sẵn**

cd existing-project/

git init

**Clone một remote repo**

git clone https://github.com/user/repository.git

Câu lệnh trên sẽ tạo một thư mục mới có tên giống trên của repo.

**Clone một remote repo tại thư mục hiện tại**

git clone https://github.com/user/repository.git .

**Xem thông tin trợ giúp cho một câu lệnh git**

git help clone

**Update và merge branch hiện tại với một remote repo**

cd repo/

git pull origin master

với origin là remote repo, master là remote branch.

Nếu bạn không muốn merge những thay đổi của bạn, hãy sử dụng git fetch

**Liệt kê các remote urls**

git remote -v

**Thay đổi origin url**

git remote set-url origin https://github.com/repo.git

**Thêm remote repo**

git remote add remote-name https://github.com/user/repo.git

**Xem thay đổi (chưa đc add) của những file hiện tại**

git diff

**Xem thay đổi (đã được add, chưa commit)**

git diff --cached

**Xem thay đổi giữa local mà master**

git diff origin/master

**Xem thay đổi giữa hai commits**

git diff COMMIT1\_ID COMMIT2\_ID

**Xem những files thay đổi giữa hai commits**

git diff --name-only COMMIT1\_ID COMMIT2\_ID

**Xem những files thay đổi tại một commit bất kỳ**

git diff-tree -no-commit-id --name-only -r COMMIT\_ID

hoặc

git show --pretty="format:" --name-only COMMIT\_ID

**Xem thay đổi trước khi push**

git diff --cached origin/master

**Xem thông tin cụ thể của một commit**

git show COMMIT\_ID

**Kiểm tra status của working tree**

git status

**Tạo vài thay đổi, rồi commit**

git add changed\_file.txt

git add folder-with-changed-files/

git commit -m "Commiting changes"

**Đổi tên/Di chuyển/Xoá files**

git rm removeme.txt tmp/crap.txt

git mv file oldname.txt file\_newname.txt

git commit -m "deleteing 2 files, renaming 1"

**Đổi message của commit cuối**

git commit --amend -m "New commit mesage"

**Push local commits sang nhánh remote**

git push origin master

**Xem commit history**

git log

**Xem commit history cho hai commits gần nhất**

git log -2

**Xem commit history cho hai commits gần nhất, bao gồm cả thay đổi**

git log -p -2

**Xem commit history dưới dạng một dòng**

git log --pretty=oneline

**Revert một commit rồi push**

git revert COMMIT\_ID

git push origin master

**Revert đến thời điểm trước một commit**

git reset COMMIT\_ID

git reset --soft HEAD@{1}

git commit -m "Revert to COMMIT\_ID"

git reset --hard

**Undo commit gần nhất, vẫn giữ thay đổi ở local**

git reset --soft HEAD~1

**Undo commit gần nhất, không giữ thay đổi ở local**

git reset --hard HEAD~1

**Undo commit gần nhất, vẫn giữ thay đổi ở index**

git reset --mixed HEAD~1

hoặc

git reset HEAD~1

**Undo commits chưa push**

git reset origin/master

**Reset về trạng thái của remote**

git fetch origin

git reset --hard origin/master

**Xem các nhánh local**

git branch

**Xem tất cả các nhánh**

git branch -a

**Tạo một patch**

git diff > patch-issue-1.patch

**Thêm một file rồi tạo patch**

git add newfile

git diff --staged > patch-issue-2.patch

**Thêm một file, thay đổi rồi tạo patch**

git add newfile

git diff HEAD > patch-issue-2.patch

**Tạo patch từ một commit**

git format-patch COMMIT\_ID

**Tạo patch từ hai commit cuối**

git format-patch HEAD~2

**Tạo patch từ tất cả những commits chưa push**

git format-patch origin/master

**Tạo patch chứa dữ liệu nhị phân**

git format-patch --binary --full-index origin/master

**Apply một patch**

git apply -v patch-name.patch

**Apply một patch được tạo bằng format-patch**

git am patch1.patch

**Tạo một tag**

git tag 7.x-1.3

**Push một tag**

git push origin 7.x-1.3

**Tạo một nhánh**

git checkout master

git branch new-branch-name

Lưu ý với hai câu lệnh trên thì chúng ta chưa chuyển sang nhánh mới, mà vẫn ở nhánh master. Phải sử dụng thêm git checkout new-branch-name để chuyển nhánh.

Ngoài ra có thể tạo nhánh mới và chuyển sang luôn bằng 1 câu git checkout -b new-branch-name

**Chuyển nhánh**

git checkout new-branch-name

**Xem commit history so với branch hiện tại**

git cherry -v master

master ở đây là branch mà bạn muốn so sánh

**Merge commit từ branch khác**

git checkout master

git merge branch-name

Ở đây chúng ta merge các commits của branch-name vào master.

**Merge branch mà không commit**

git merge branch-name --no-commit --no-ff

**Xem thay đổi giữa state hiện tại và một branch**

git diff branch-name

**Xem thay đổi trong một file, giữa state hiện tại và một branch**

git diff branch-name path/to/file

**Xoá branch**

git branch -d branch-name

**Push lên một branch**

git push origin branch-name

**Lấy tất cả các branches**

git fetch orgin

**Lấy thư mục root**

git rev-parse --show-toplevel

**Xoá các file bị xoá ở local trên repo**

git rm $(git ls-files --deleted)

**Xoá toàn bộ các files chưa đc track**

git clean -f

xoá cả folder:

git clean -f -d

xem các file trước khi xoá:

git clean -n -f -d

**Unstage các files**

git reset HEAD file.txt

**Xem tag gần nhất**

git describe --tags `git rev-list --tag --max-count=1`

**Liệt kê các nhánh theo trình tự sử dụng gần nhất**

git for-each-ref --sort=-committerdate refs/heads/ | head

**Tar cả project, ngoại trừ thư mục .git**

cd ..

tar cJf project.tar.xz project/ --exclude-vcs

**Tar tất cả các files bị thay đổi ở local**

git diff --name-only | xargs tar -cf project.tar -T -

**Tìm conflict**

grep -H -r "<<<" \*

grep -H -r ">>>" \*

grep -H -r '^=======$' \*

**Apply một patch không sử dụng git**

patch < file.patch