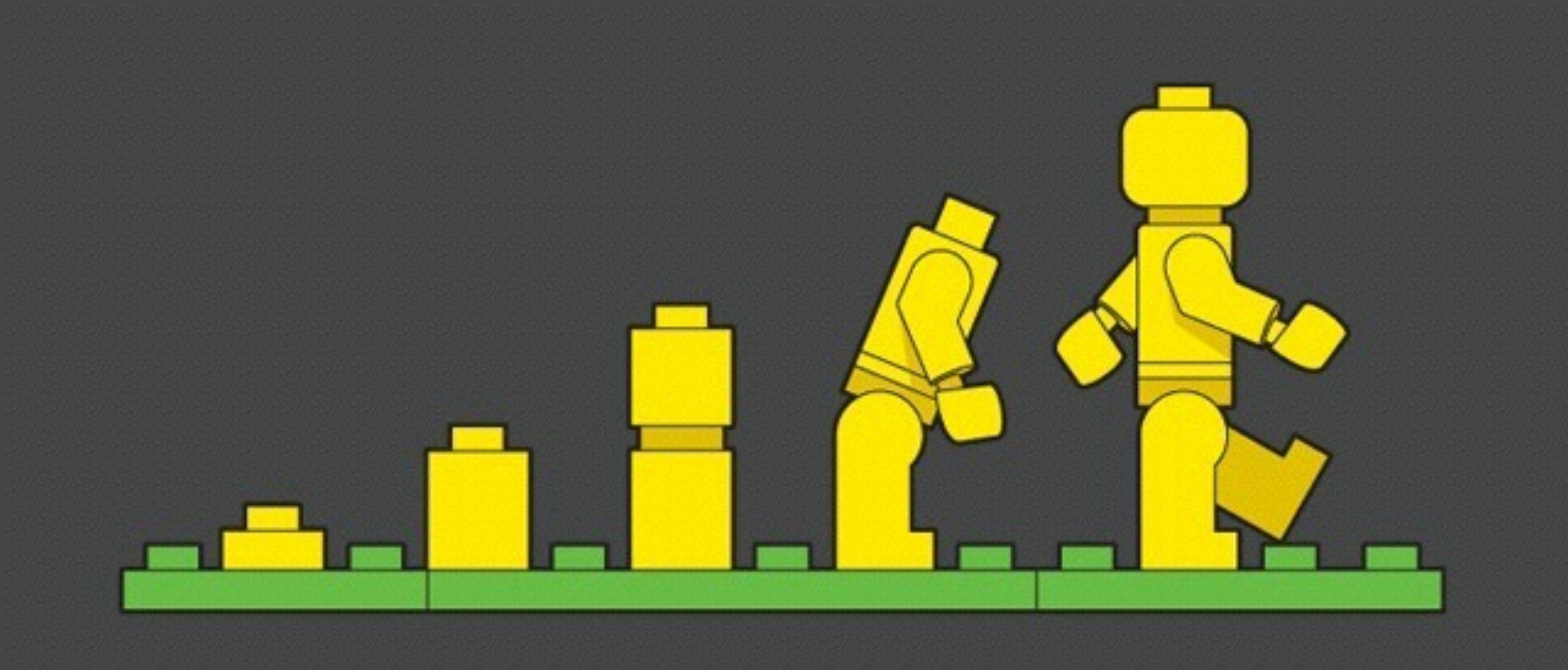
SAKOL ASSAWASAGOOL

@koobitor

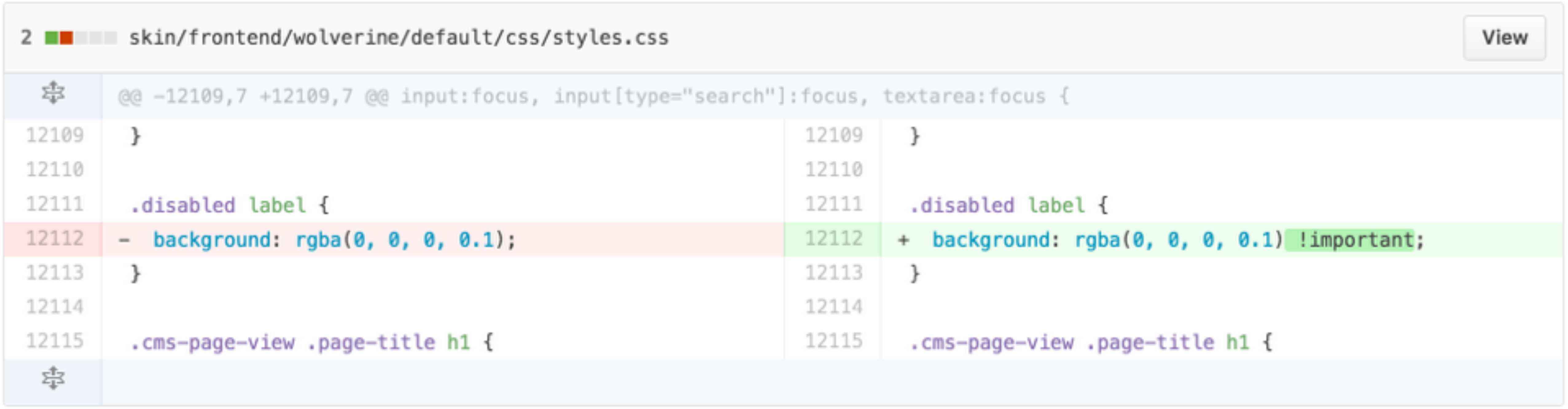
Version control



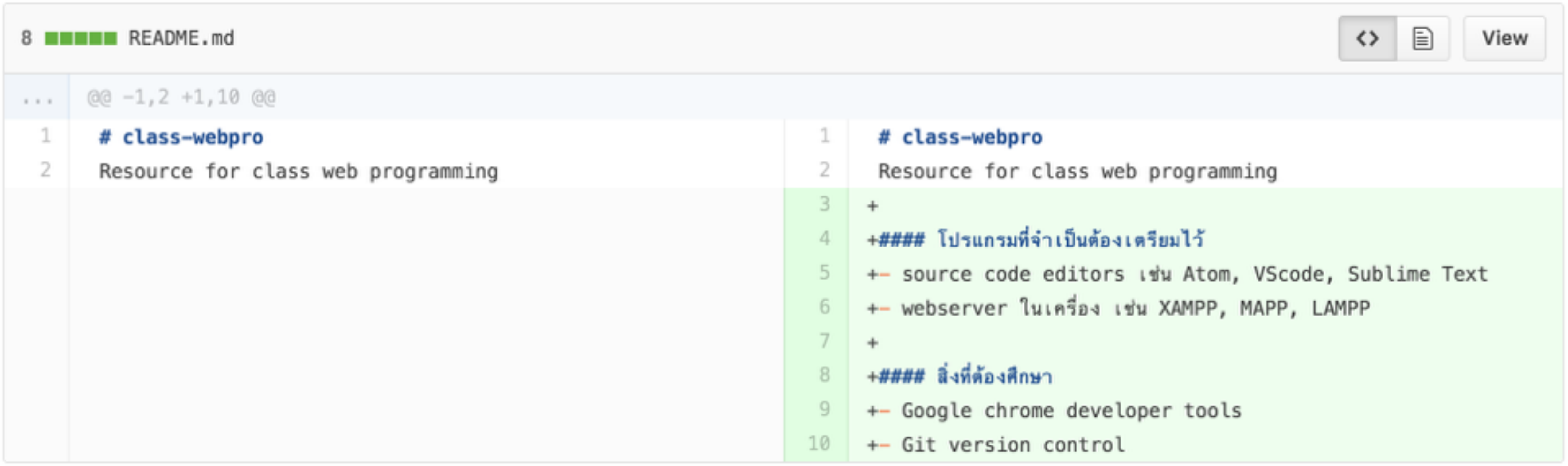
Git คืออะไร ?

https://www.youtube.com/watch?v=OqmSzXDrJBk

git-diff

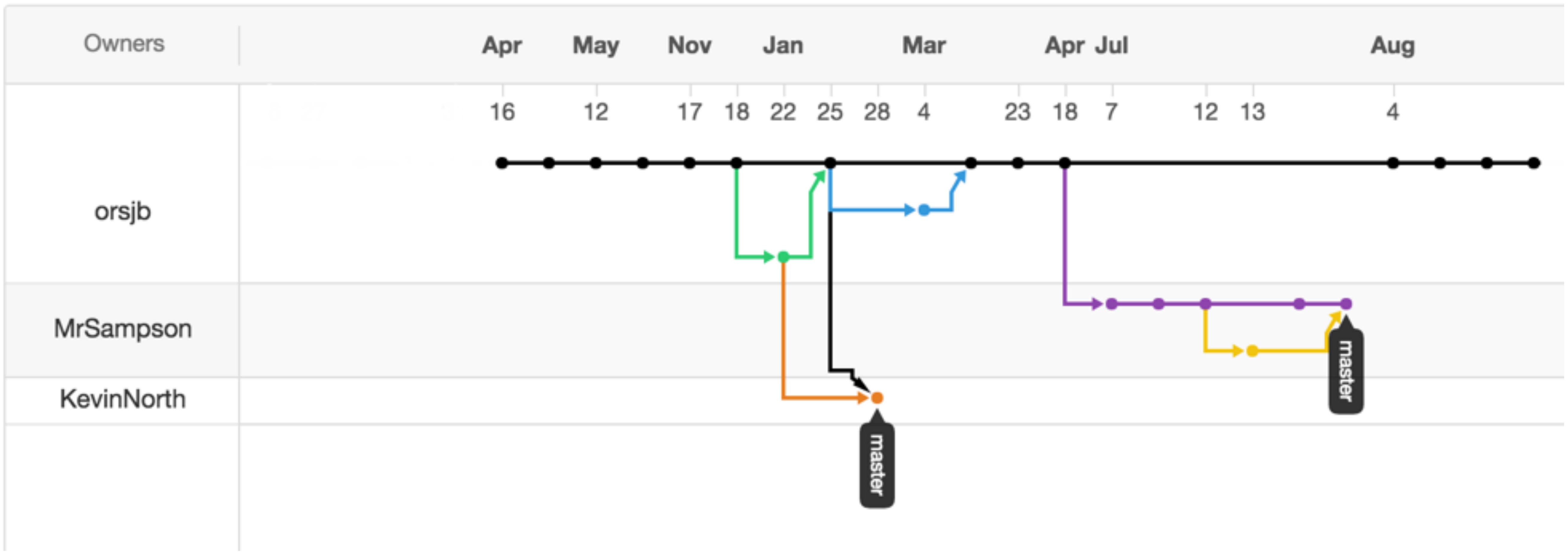


```
2 skin/frontend/wolverine/default/css/styles.css                                   View

@@ -12109,7 +12109,7 @@ input:focus, input[type="search"]:focus, textarea:focus {

 }                                               }

 .disabled label {                               .disabled label {
-  background: rgba(0, 0, 0, 0.1);              +  background: rgba(0, 0, 0, 0.1) !important;
 }                                               }

 .cms-page-view .page-title h1 {                 .cms-page-view .page-title h1 {
```

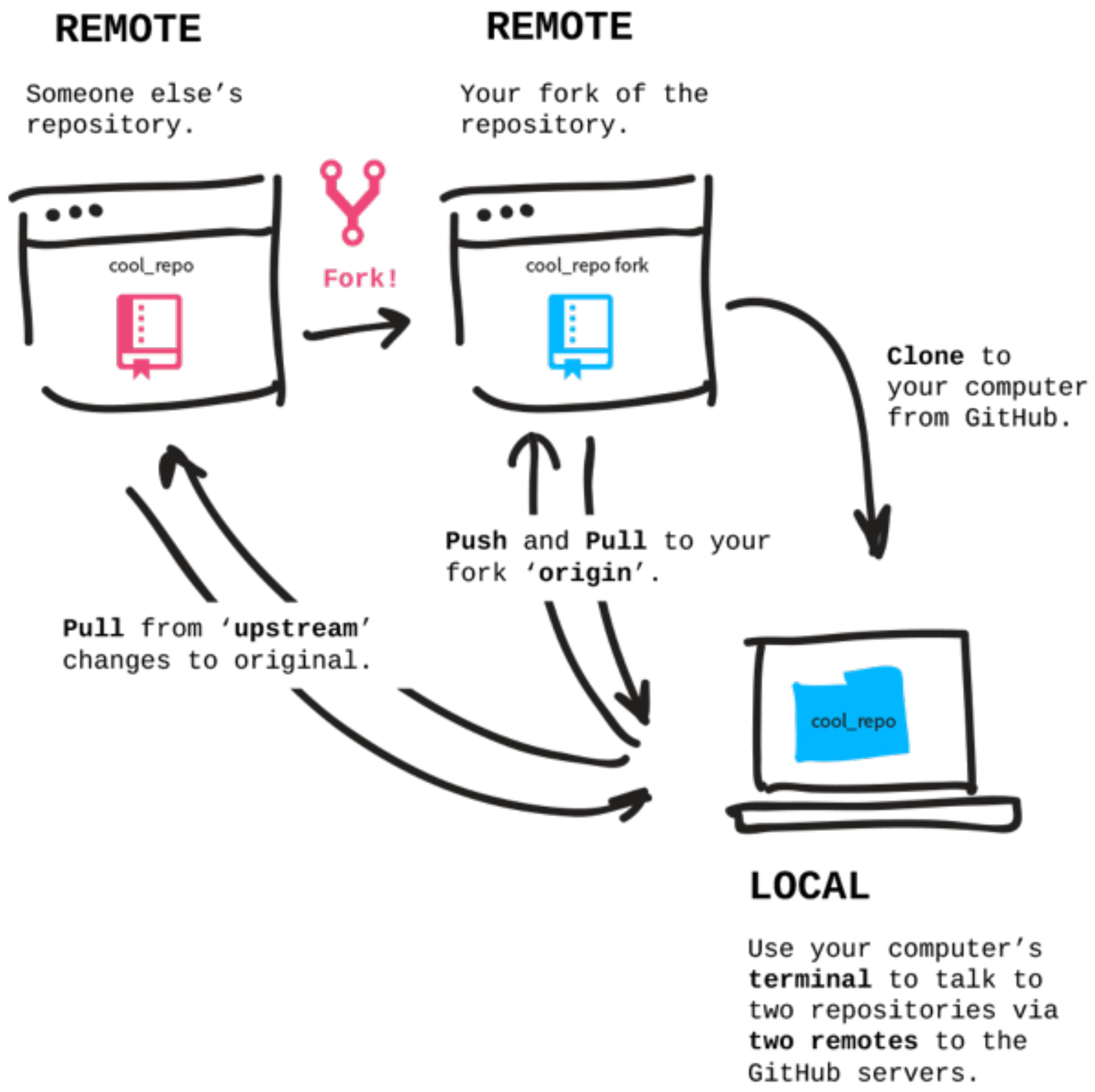


```
8 README.md                                                                        View

@@ -1,2 +1,10 @@
# class-webpro                                   # class-webpro
Resource for class web programming               Resource for class web programming
                                                 +
                                                 +#### โปรแกรมที่จำเป็นต้องเตรียมไว้
                                                 +- source code editors เช่น Atom, VScode, Sublime Text
                                                 +- webserver ในเครื่อง เช่น XAMPP, MAPP, LAMPP
                                                 +
                                                 +#### สิ่งที่ต้องศึกษา
                                                 +- Google chrome developer tools
                                                 +- Git version control
```

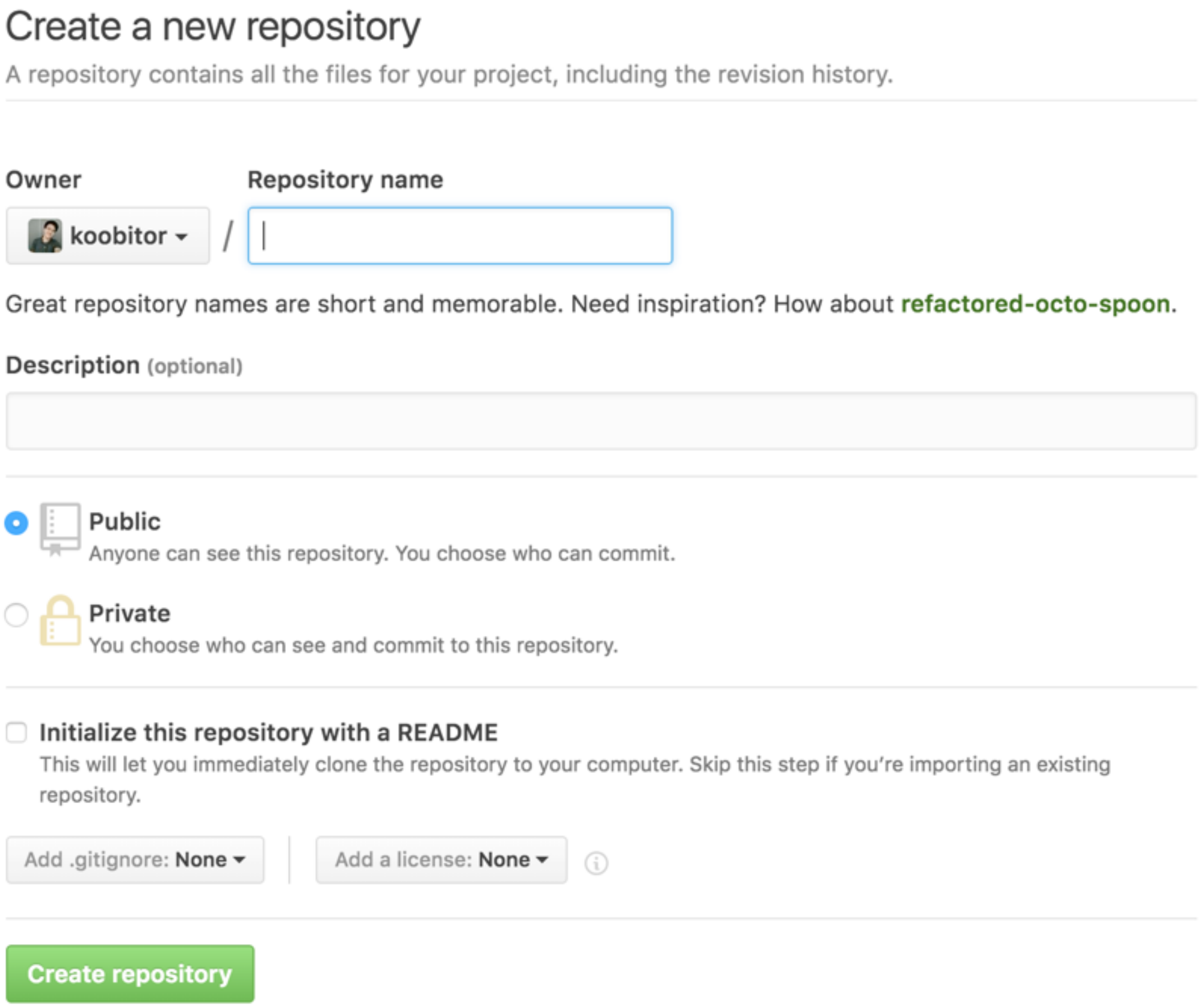
Git Network Graph



Git Fork


REMOTE
Someone else's repository.
cool_repo
Fork!
REMOTE
Your fork of the repository.
cool_repo fork
Clone to your computer from GitHub.
Push and Pull to your fork 'origin'.
Pull from 'upstream' changes to original.
cool_repo
LOCAL
Use your computer's terminal to talk to two repositories via two remotes to the GitHub servers.

github
SOCIAL CODING


Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: koobitor / Repository name:

Great repository names are short and memorable. Need inspiration? How about **refactored-octo-spoon**.

Description (optional)

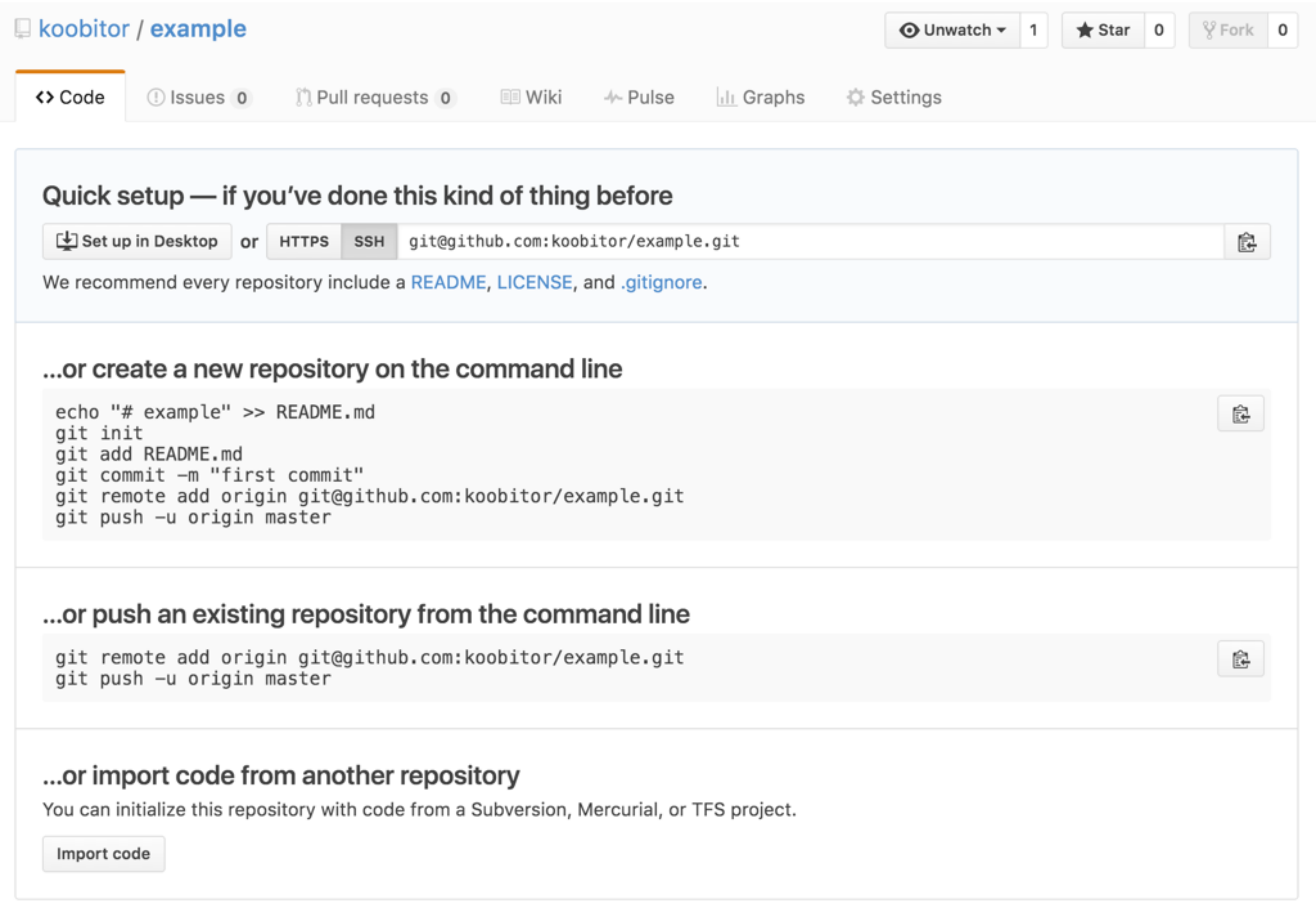
● **Public**
Anyone can see this repository. You choose who can commit.

○ **Private**
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **None**



Quick setup — if you've done this kind of thing before

```
git@github.com:koobitor/example.git
```

We recommend every repository include a README, LICENSE, and .gitignore.

...or create a new repository on the command line

```
echo "# example" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:koobitor/example.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:koobitor/example.git
git push -u origin master
```

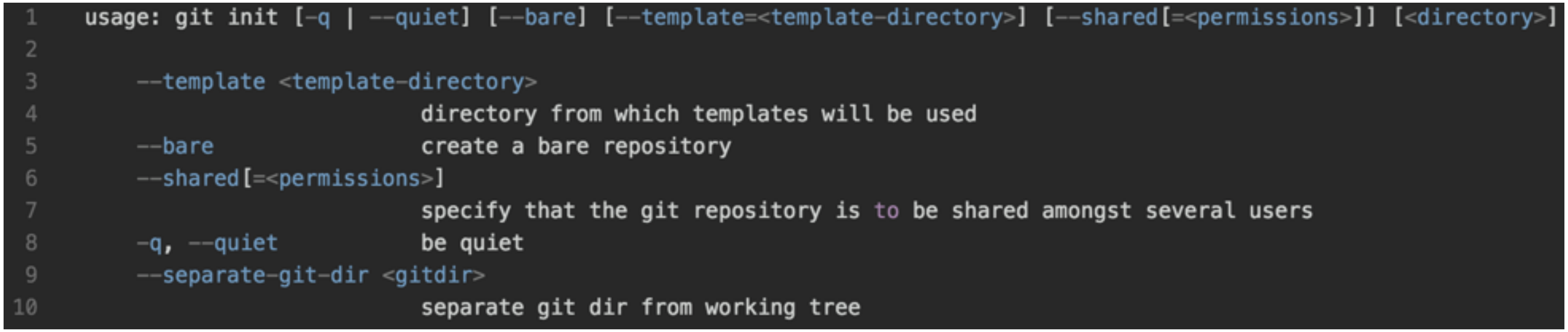
...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

$ git init

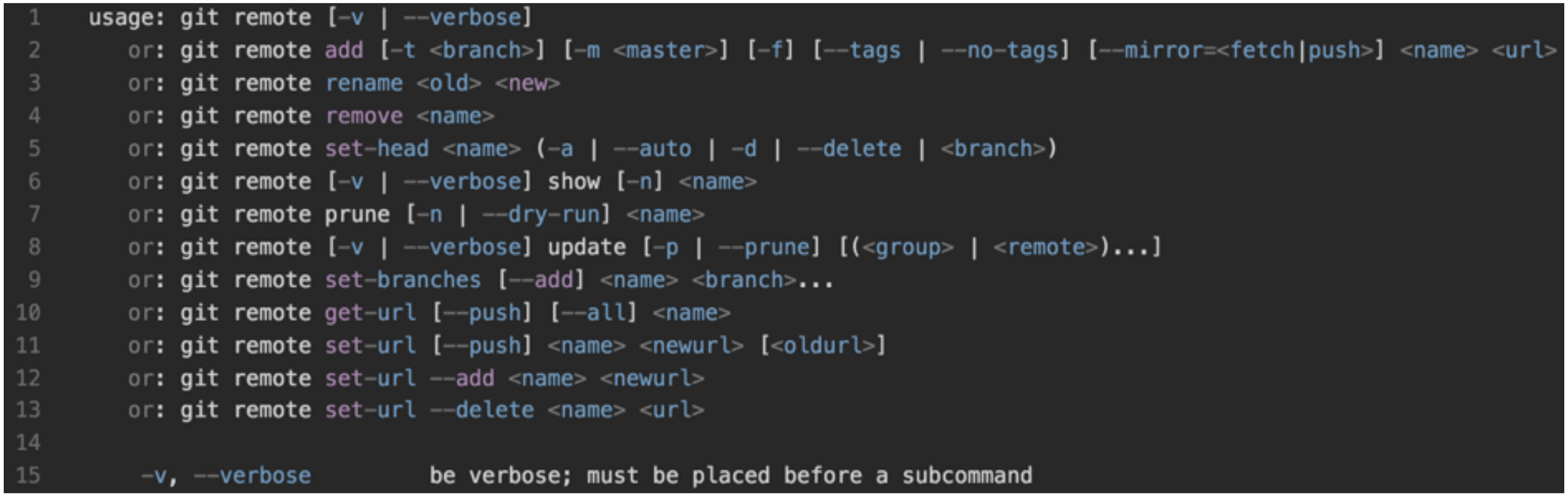
ทำการสร้าง local repo

(ระบบจะสร้างโฟลเดอร์ .git ไว้ใน directory



```
usage: git init [-q | --quiet] [--bare] [--template=<template-directory>] [--shared[=<permissions>]] [<directory>]

    --template <template-directory>
                          directory from which templates will be used
    --bare                create a bare repository
    --shared[=<permissions>]
                          specify that the git repository is to be shared amongst several users
    -q, --quiet           be quiet
    --separate-git-dir <gitdir>
                          separate git dir from working tree
```

$ git remote



```
usage: git remote [-v | --verbose]
   or: git remote add [-t <branch>] [-m <master>] [-f] [--tags | --no-tags] [--mirror=<fetch|push>] <name> <url>
   or: git remote rename <old> <new>
   or: git remote remove <name>
   or: git remote set-head <name> (-a | --auto | -d | --delete | <branch>)
   or: git remote [-v | --verbose] show [-n] <name>
   or: git remote prune [-n | --dry-run] <name>
   or: git remote [-v | --verbose] update [-p | --prune] [(<group> | <remote>)...]
   or: git remote set-branches [--add] <name> <branch>...
   or: git remote get-url [--push] [--all] <name>
   or: git remote set-url [--push] <name> <newurl> [<oldurl>]
   or: git remote set-url --add <name> <newurl>
   or: git remote set-url --delete <name> <url>

    -v, --verbose         be verbose; must be placed before a subcommand
```

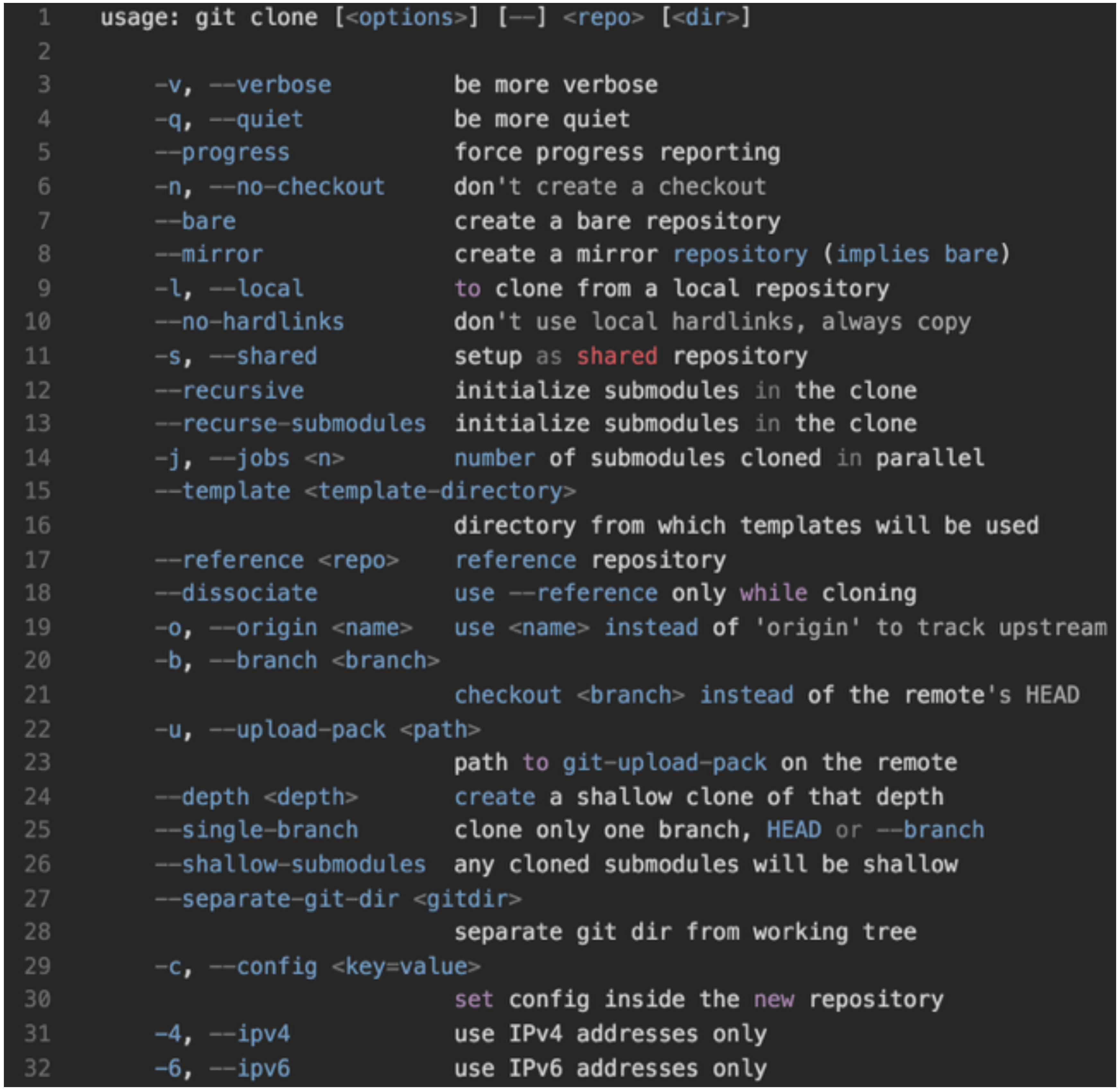
$ git remote add arg1 arg2

เพิ่ม remote มี 2 arguments

arg1 = remote name (default : origin)

arg2 = remote URL

$ git clone

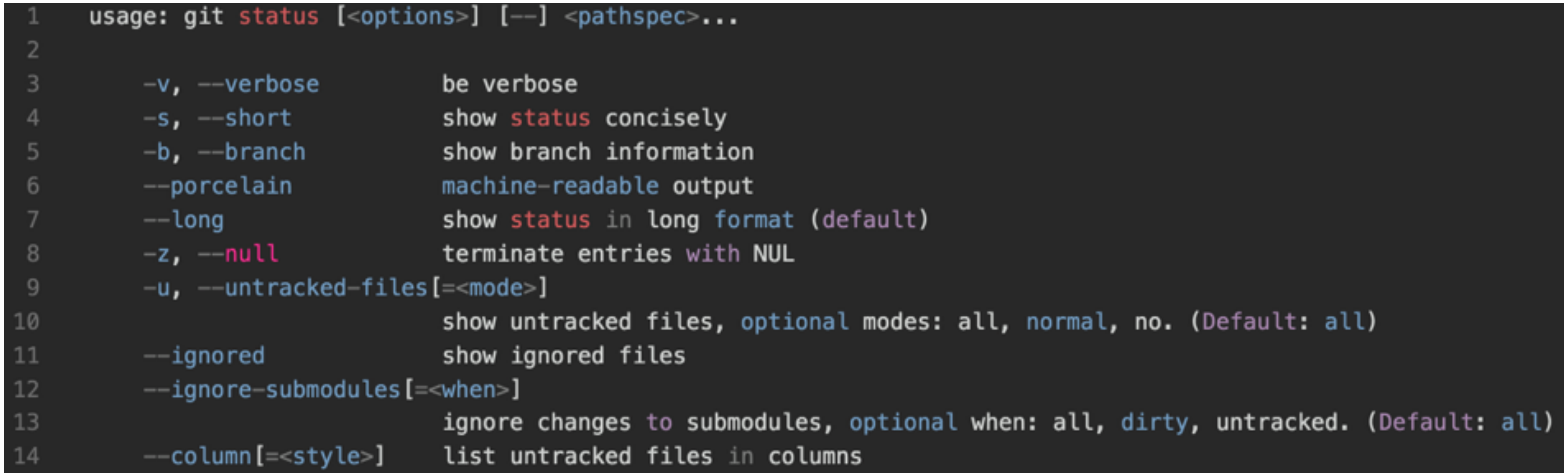


```
usage: git clone [<options>] [--] <repo> [<dir>]

    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --progress            force progress reporting
    -n, --no-checkout     don't create a checkout
    --bare                create a bare repository
    --mirror              create a mirror repository (implies bare)
    -l, --local           to clone from a local repository
    --no-hardlinks        don't use local hardlinks, always copy
    -s, --shared          setup as shared repository
    --recursive           initialize submodules in the clone
    --recurse-submodules  initialize submodules in the clone
    -j, --jobs <n>        number of submodules cloned in parallel
    --template <template-directory>
                          directory from which templates will be used
    --reference <repo>    reference repository
    --dissociate          use --reference only while cloning
    -o, --origin <name>   use <name> instead of 'origin' to track upstream
    -b, --branch <branch>
                          checkout <branch> instead of the remote's HEAD
    -u, --upload-pack <path>
                          path to git-upload-pack on the remote
    --depth <depth>       create a shallow clone of that depth
    --single-branch       clone only one branch, HEAD or --branch
    --shallow-submodules  any cloned submodules will be shallow
    --separate-git-dir <gitdir>
                          separate git dir from working tree
    -c, --config <key=value>
                          set config inside the new repository
    -4, --ipv4            use IPv4 addresses only
    -6, --ipv6            use IPv6 addresses only
```

ทำการ clone repo จาก Server (Github/Bitbucket) มาที่ local

$ git status

ทำการเช็คสถานะใน directory



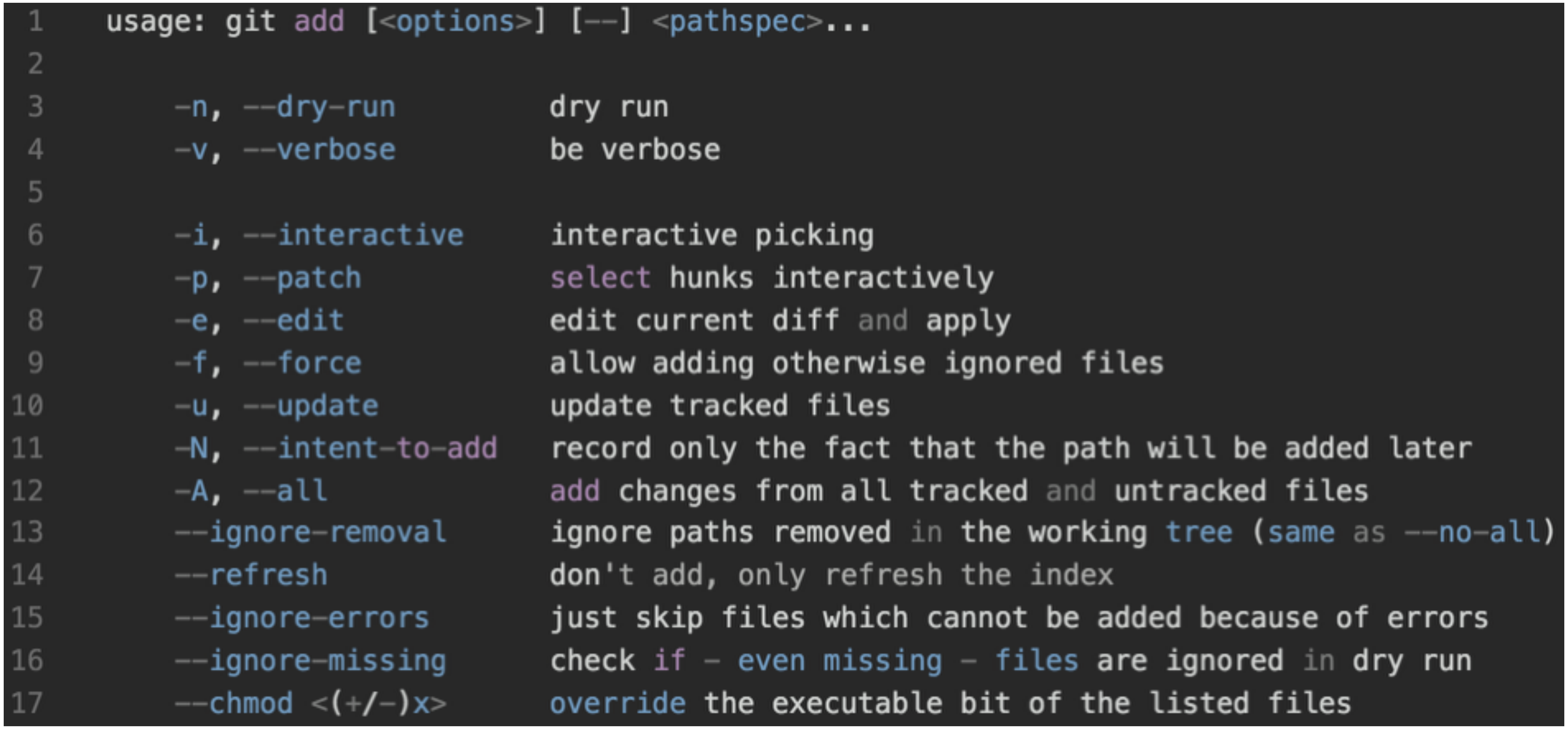
```
usage: git status [<options>] [--] <pathspec>...

    -v, --verbose         be verbose
    -s, --short           show status concisely
    -b, --branch          show branch information
    --porcelain           machine-readable output
    --long                show status in long format (default)
    -z, --null            terminate entries with NUL
    -u, --untracked-files[=<mode>]
                          show untracked files, optional modes: all, normal, no. (Default: all)
    --ignored             show ignored files
    --ignore-submodules[=<when>]
                          ignore changes to submodules, optional when: all, dirty, untracked. (Default: all)
    --column[=<style>]    list untracked files in columns
```

$ git add <filename>

เพิ่มไฟล์ <filename> ไปที่ staging (พร้อมสำหรับ commit)

$ git add .

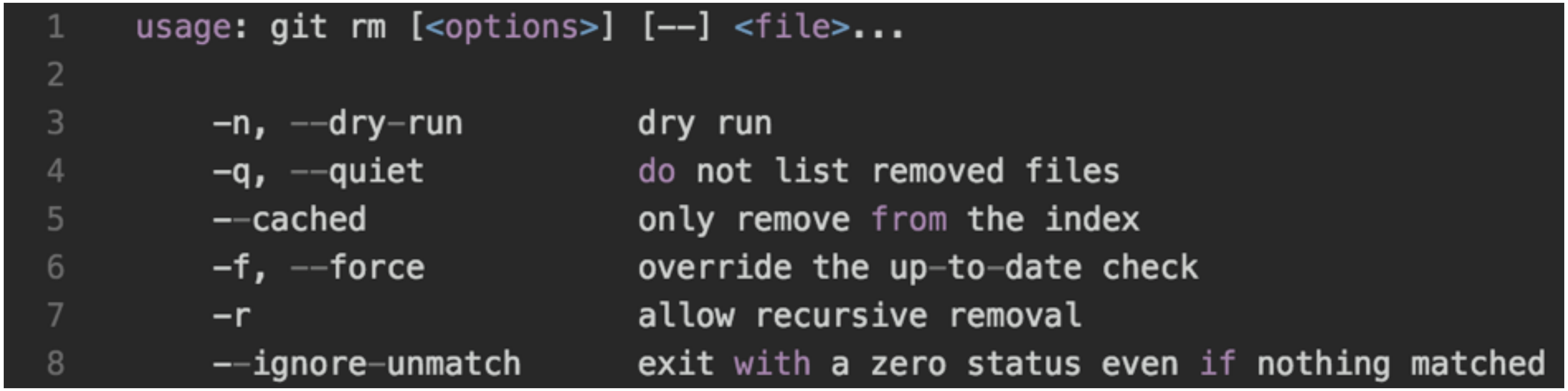
เพิ่มทุกไฟล์ที่มีการแก้ไข/เปลี่ยนแปลง



```
usage: git add [<options>] [--] <pathspec>...

    -n, --dry-run         dry run
    -v, --verbose         be verbose

    -i, --interactive     interactive picking
    -p, --patch           select hunks interactively
    -e, --edit            edit current diff and apply
    -f, --force           allow adding otherwise ignored files
    -u, --update          update tracked files
    -N, --intent-to-add   record only the fact that the path will be added later
    -A, --all             add changes from all tracked and untracked files
    --ignore-removal      ignore paths removed in the working tree (same as --no-all)
    --refresh             don't add, only refresh the index
    --ignore-errors       just skip files which cannot be added because of errors
    --ignore-missing      check if - even missing - files are ignored in dry run
    --chmod <(+/-)x>      override the executable bit of the listed files
```

$ git rm

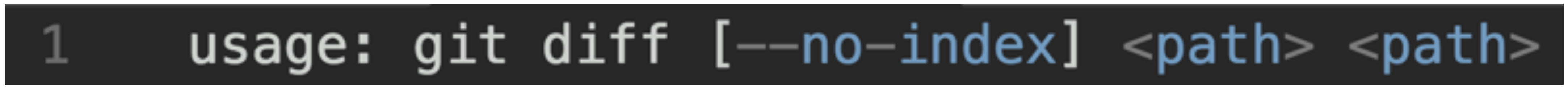
ทำการลบไฟล์ และให้ git ทำการ untracked ไฟล์ด้วย



```
usage: git rm [<options>] [--] <file>...

    -n, --dry-run         dry run
    -q, --quiet           do not list removed files
    --cached              only remove from the index
    -f, --force           override the up-to-date check
    -r                    allow recursive removal
    --ignore-unmatch      exit with a zero status even if nothing matched
```

$ git diff

แสดงการเปลี่ยนแปลงของไฟล์



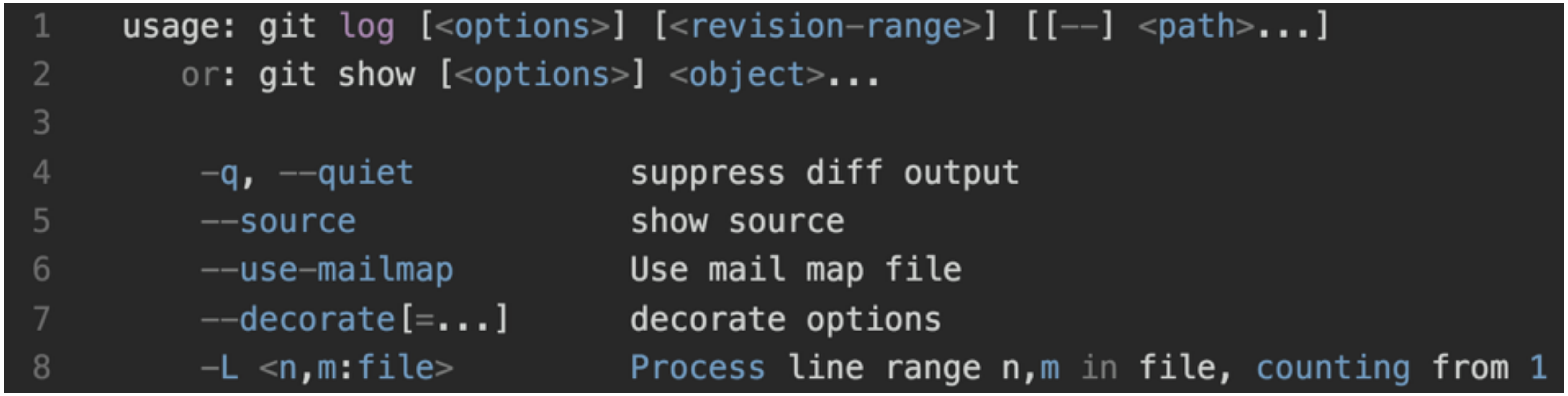
```
usage: git diff [--no-index] <path> <path>
```

$ git diff branch1 branch2

ทำการเปรียบเทียบระหว่าง branch1 กับ branch2

$ git log

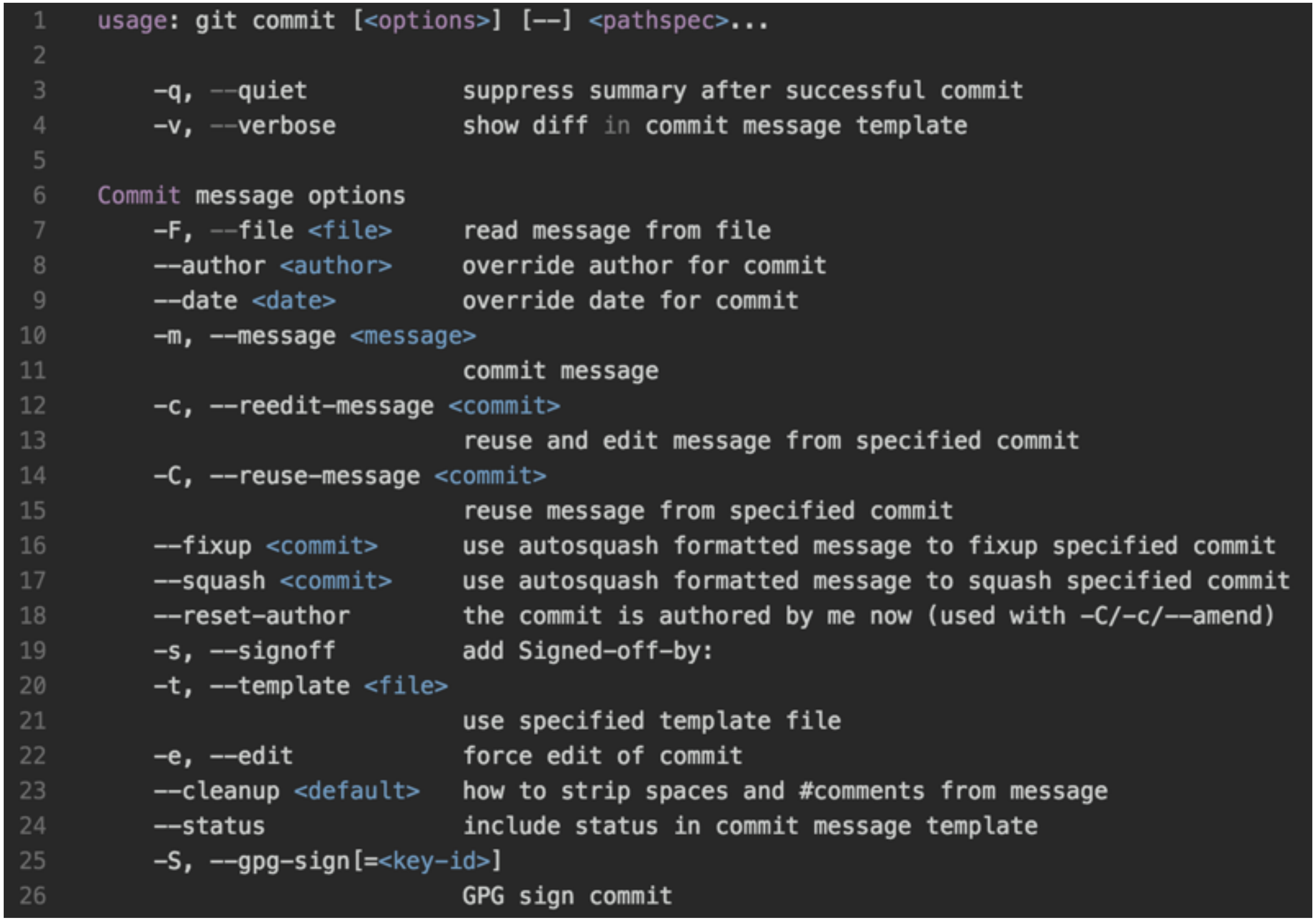
โชว์ log history ของ git



```
usage: git log [<options>] [<revision-range>] [[--] <path>...]
   or: git show [<options>] <object>...

    -q, --quiet           suppress diff output
    --source              show source
    --use-mailmap         Use mail map file
    --decorate[=...]      decorate options
    -L <n,m:file>         Process line range n,m in file, counting from 1
```

$ git commit -m "Message"

ทำการ commit staged บันทึกลง Project History

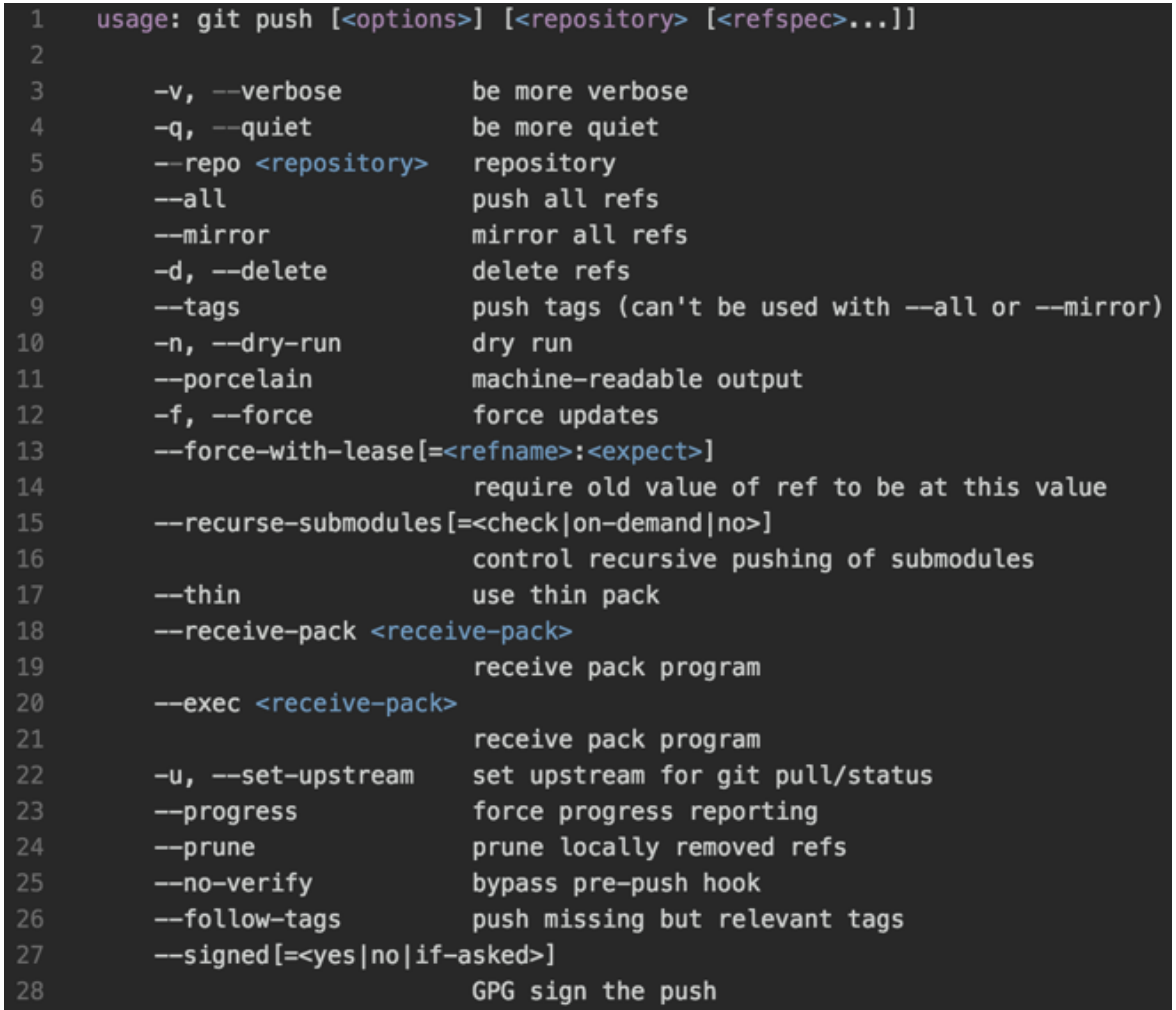


```
usage: git commit [<options>] [--] <pathspec>...

    -q, --quiet           suppress summary after successful commit
    -v, --verbose         show diff in commit message template

Commit message options
    -F, --file <file>     read message from file
    --author <author>     override author for commit
    --date <date>         override date for commit
    -m, --message <message>
                          commit message
    -c, --reedit-message <commit>
                          reuse and edit message from specified commit
    -C, --reuse-message <commit>
                          reuse message from specified commit
    --fixup <commit>      use autosquash formatted message to fixup specified commit
    --squash <commit>     use autosquash formatted message to squash specified commit
    --reset-author        the commit is authored by me now (used with -C/-c/--amend)
    -s, --signoff         add Signed-off-by:
    -t, --template <file>
                          use specified template file
    -e, --edit            force edit of commit
    --cleanup <default>   how to strip spaces and #comments from message
    --status              include status in commit message template
    -S, --gpg-sign[=<key-id>]
                          GPG sign commit
```

$ git push origin master

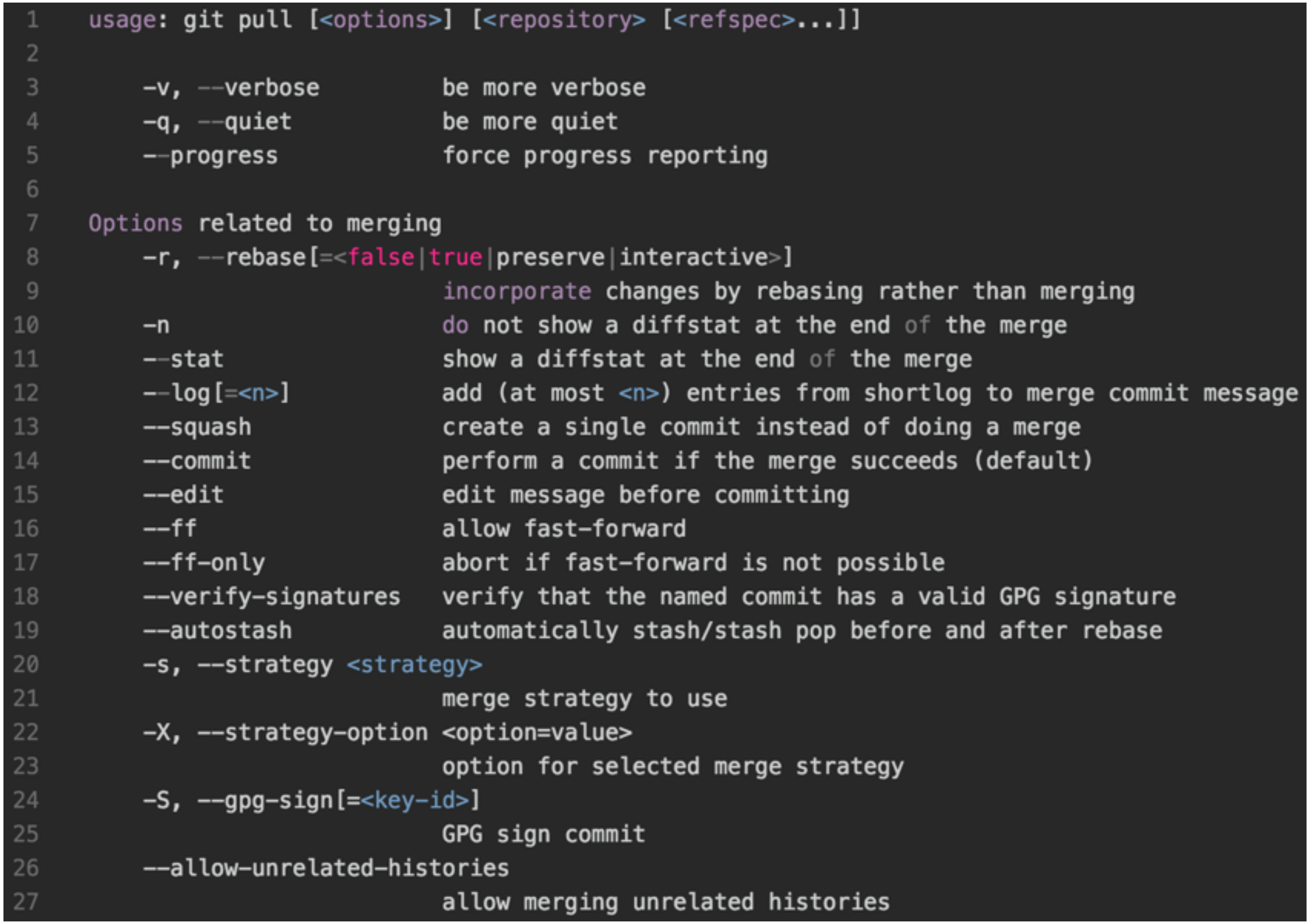
ทำการ push โปรเจ็คไป remote repository

(origin ชื่อ remote name, master คือชื่อ default ของ branch)



```
usage: git push [<options>] [<repository> [<refspec>...]]

    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --repo <repository>   repository
    --all                 push all refs
    --mirror              mirror all refs
    -d, --delete          delete refs
    --tags                push tags (can't be used with --all or --mirror)
    -n, --dry-run         dry run
    --porcelain           machine-readable output
    -f, --force           force updates
    --force-with-lease[=<refname>:<expect>]
                          require old value of ref to be at this value
    --recurse-submodules[=<check|on-demand|no>]
                          control recursive pushing of submodules
    --thin                use thin pack
    --receive-pack <receive-pack>
                          receive pack program
    --exec <receive-pack>
                          receive pack program
    -u, --set-upstream    set upstream for git pull/status
    --progress            force progress reporting
    --prune               prune locally removed refs
    --no-verify           bypass pre-push hook
    --follow-tags         push missing but relevant tags
    --signed[=<yes|no|if-asked>]
                          GPG sign the push
```

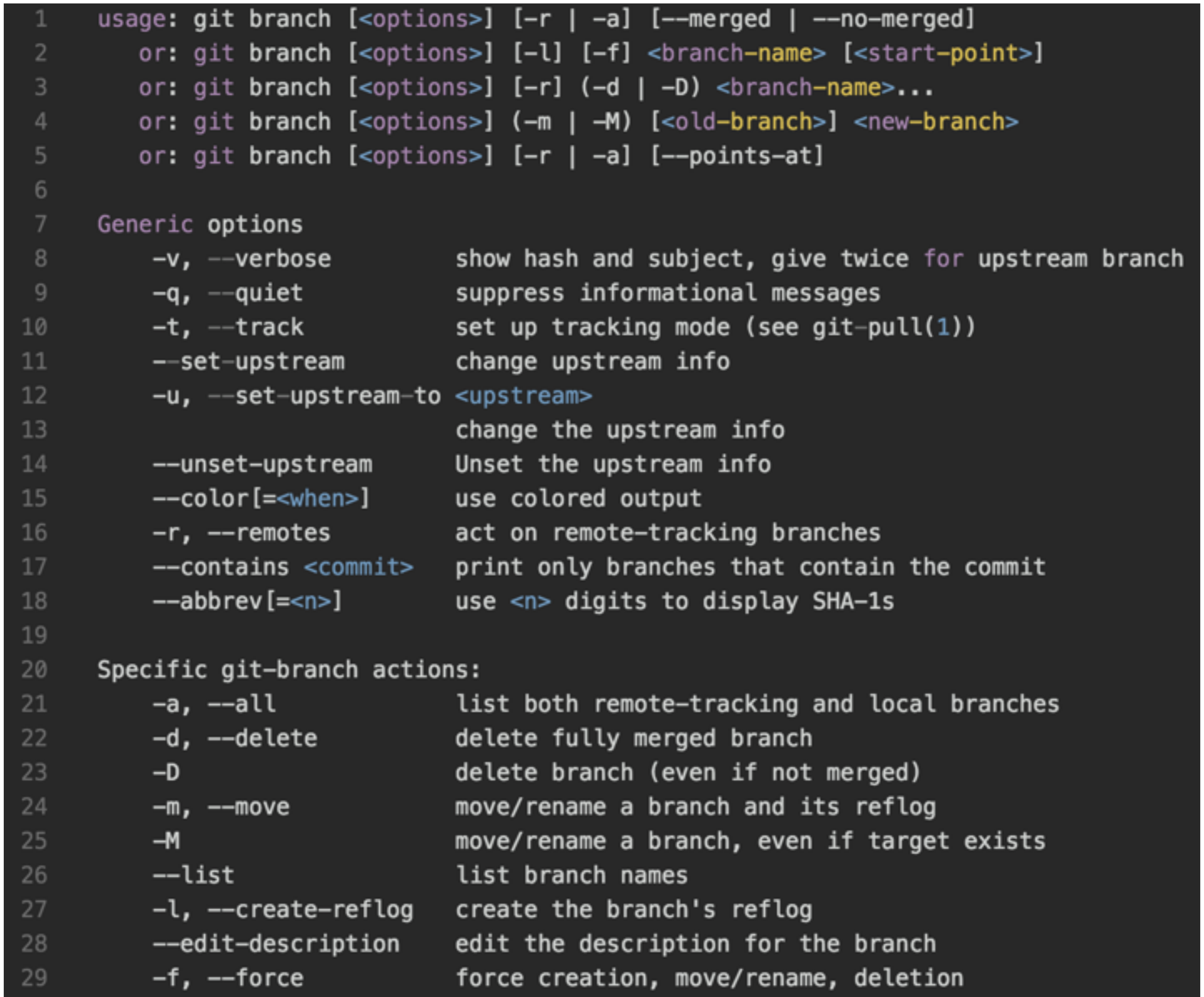
$ git pull เช็คการเปลี่ยนแปลงและรวม (เหมือนกับการทำ git fetch และต่อด้วย git merge)



```
usage: git pull [<options>] [<repository> [<refspec>...]]

    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --progress            force progress reporting

Options related to merging
    -r, --rebase[=<false|true|preserve|interactive>]
                          incorporate changes by rebasing rather than merging
    -n                    do not show a diffstat at the end of the merge
    --stat                show a diffstat at the end of the merge
    --log[=<n>]           add (at most <n>) entries from shortlog to merge commit message
    --squash              create a single commit instead of doing a merge
    --commit              perform a commit if the merge succeeds (default)
    --edit                edit message before committing
    --ff                  allow fast-forward
    --ff-only             abort if fast-forward is not possible
    --verify-signatures   verify that the named commit has a valid GPG signature
    --autostash           automatically stash/stash pop before and after rebase
    -s, --strategy <strategy>
                          merge strategy to use
    -X, --strategy-option <option=value>
                          option for selected merge strategy
    -S, --gpg-sign[=<key-id>]
                          GPG sign commit
    --allow-unrelated-histories
                          allow merging unrelated histories
```

$ git branch โชว์ list ของ branch ทั้งหมด

$ git branch <name> สร้าง branch ใหม่



```
usage: git branch [<options>] [-r | -a] [--merged | --no-merged]
   or: git branch [<options>] [-l] [-f] <branch-name> [<start-point>]
   or: git branch [<options>] [-r] (-d | -D) <branch-name>...
   or: git branch [<options>] (-m | -M) [<old-branch>] <new-branch>
   or: git branch [<options>] [-r | -a] [--points-at]

Generic options
    -v, --verbose         show hash and subject, give twice for upstream branch
    -q, --quiet           suppress informational messages
    -t, --track           set up tracking mode (see git-pull(1))
    --set-upstream        change upstream info
    -u, --set-upstream-to <upstream>
                          change the upstream info
    --unset-upstream      Unset the upstream info
    --color[=<when>]      use colored output
    -r, --remotes         act on remote-tracking branches
    --contains <commit>   print only branches that contain the commit
    --abbrev[=<n>]        use <n> digits to display SHA-1s

Specific git-branch actions:
    -a, --all             list both remote-tracking and local branches
    -d, --delete          delete fully merged branch
    -D                    delete branch (even if not merged)
    -m, --move            move/rename a branch and its reflog
    -M                    move/rename a branch, even if target exists
    --list                list branch names
    -l, --create-reflog   create the branch's reflog
    --edit-description    edit the description for the branch
    -f, --force           force creation, move/rename, deletion
```

```
$ git checkout <name>
```

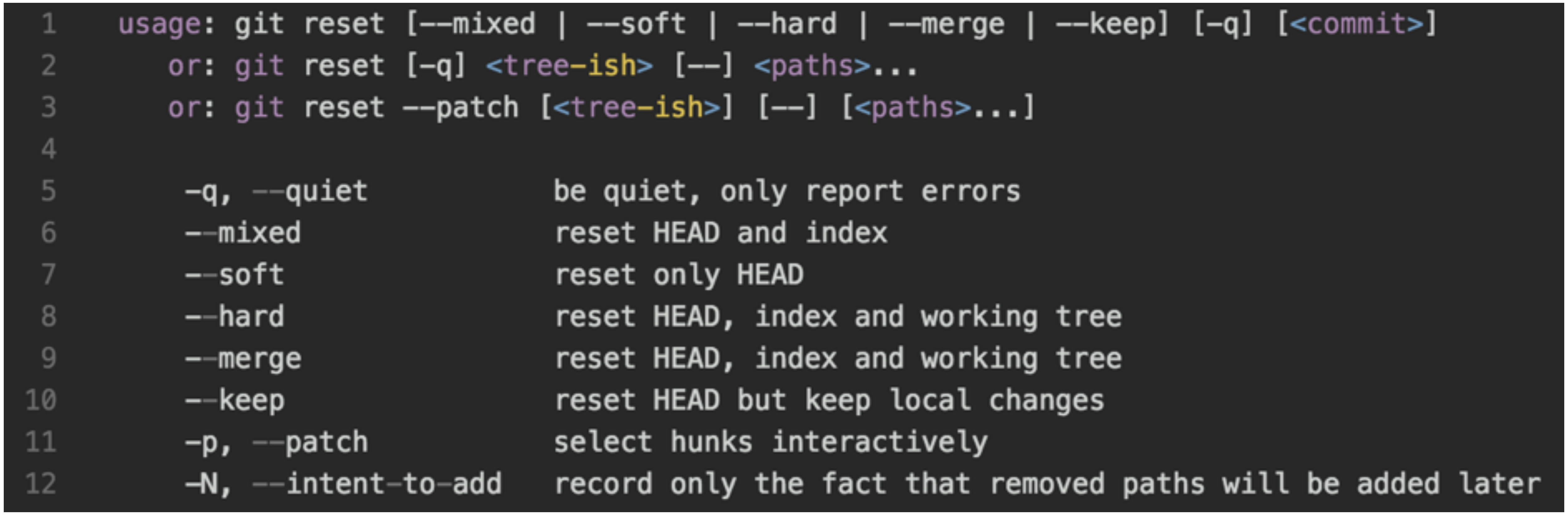
ทำการเปลี่ยน branch (ย้าย HEAD ไป branch ใหม่)
ต้องมี branch อยู่

```
$ git checkout -b <name>
```

ทำการสร้างและเปลี่ยนไป branch ใหม่
(มีค่าเท่ากับ git branch <name> ต่อด้วย git checkout <name>)

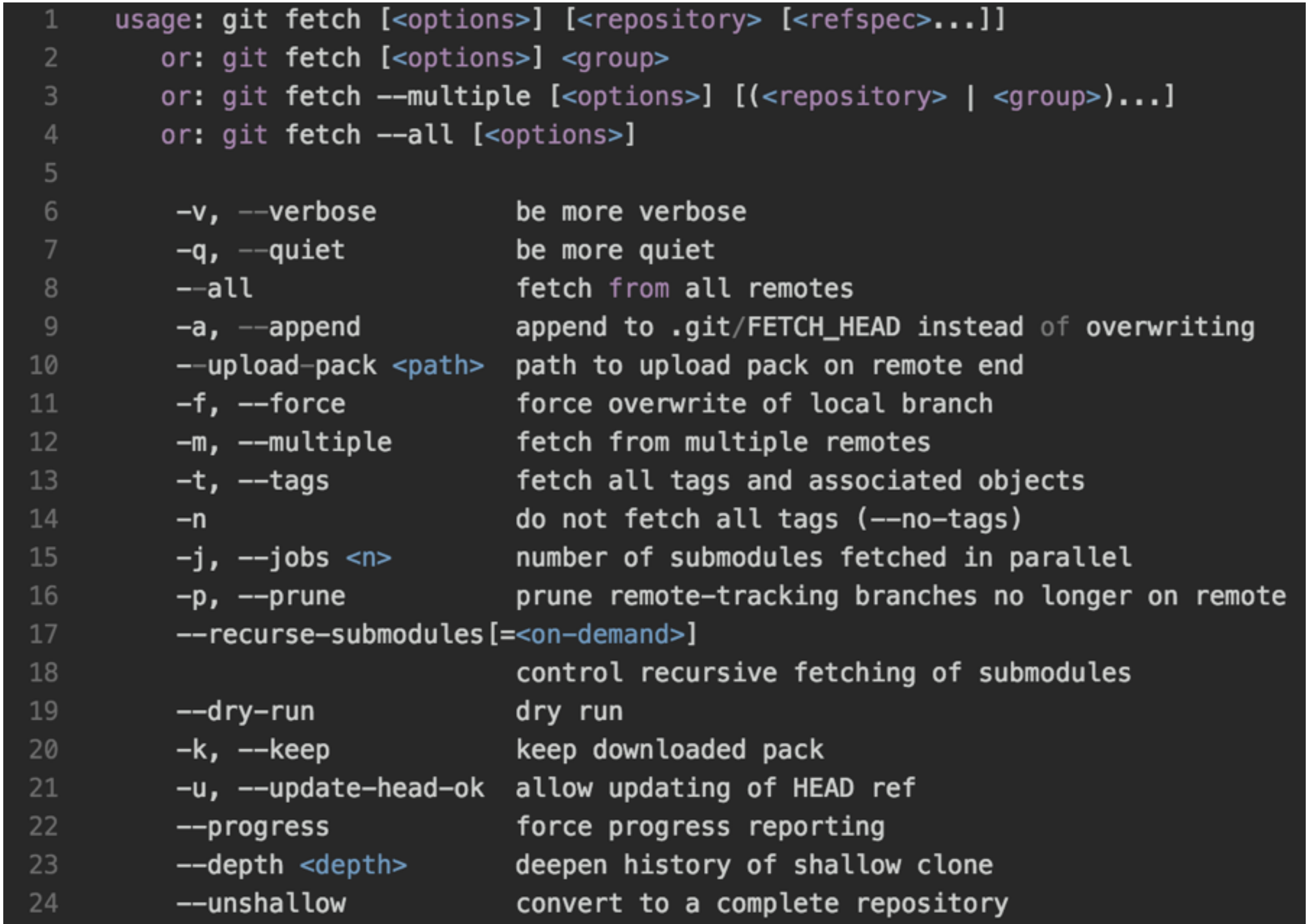
$ git reset HEAD

reset local repo



```
usage: git reset [--mixed | --soft | --hard | --merge | --keep] [-q] [<commit>]
   or: git reset [-q] <tree-ish> [--] <paths>...
   or: git reset --patch [<tree-ish>] [--] [<paths>...]

    -q, --quiet           be quiet, only report errors
    --mixed               reset HEAD and index
    --soft                reset only HEAD
    --hard                reset HEAD, index and working tree
    --merge               reset HEAD, index and working tree
    --keep                reset HEAD but keep local changes
    -p, --patch           select hunks interactively
    -N, --intent-to-add   record only the fact that removed paths will be added later
```

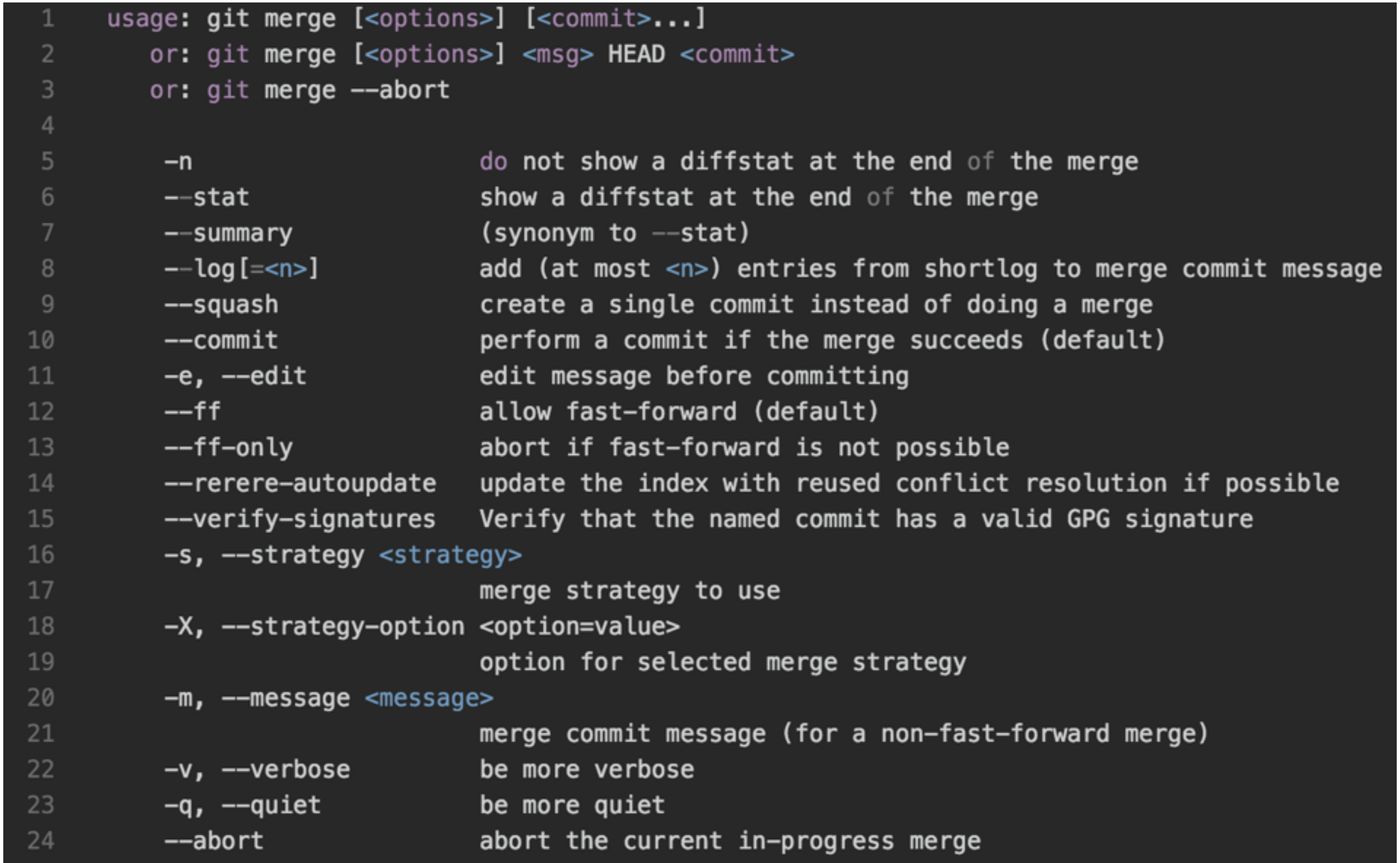
$ git fetch เช็คการเปลี่ยนแปลงจาก remote repo



```
usage: git fetch [<options>] [<repository> [<refspec>...]]
   or: git fetch [<options>] <group>
   or: git fetch --multiple [<options>] [(<repository> | <group>)...]
   or: git fetch --all [<options>]

    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --all                 fetch from all remotes
    -a, --append          append to .git/FETCH_HEAD instead of overwriting
    --upload-pack <path>  path to upload pack on remote end
    -f, --force           force overwrite of local branch
    -m, --multiple        fetch from multiple remotes
    -t, --tags            fetch all tags and associated objects
    -n                    do not fetch all tags (--no-tags)
    -j, --jobs <n>        number of submodules fetched in parallel
    -p, --prune           prune remote-tracking branches no longer on remote
    --recurse-submodules[=<on-demand>]
                          control recursive fetching of submodules
    --dry-run             dry run
    -k, --keep            keep downloaded pack
    -u, --update-head-ok  allow updating of HEAD ref
    --progress            force progress reporting
    --depth <depth>       deepen history of shallow clone
    --unshallow           convert to a complete repository
```

$ git merge

ทำการรวมการเปลี่ยนแปลงจาก remote มาที่ local repo



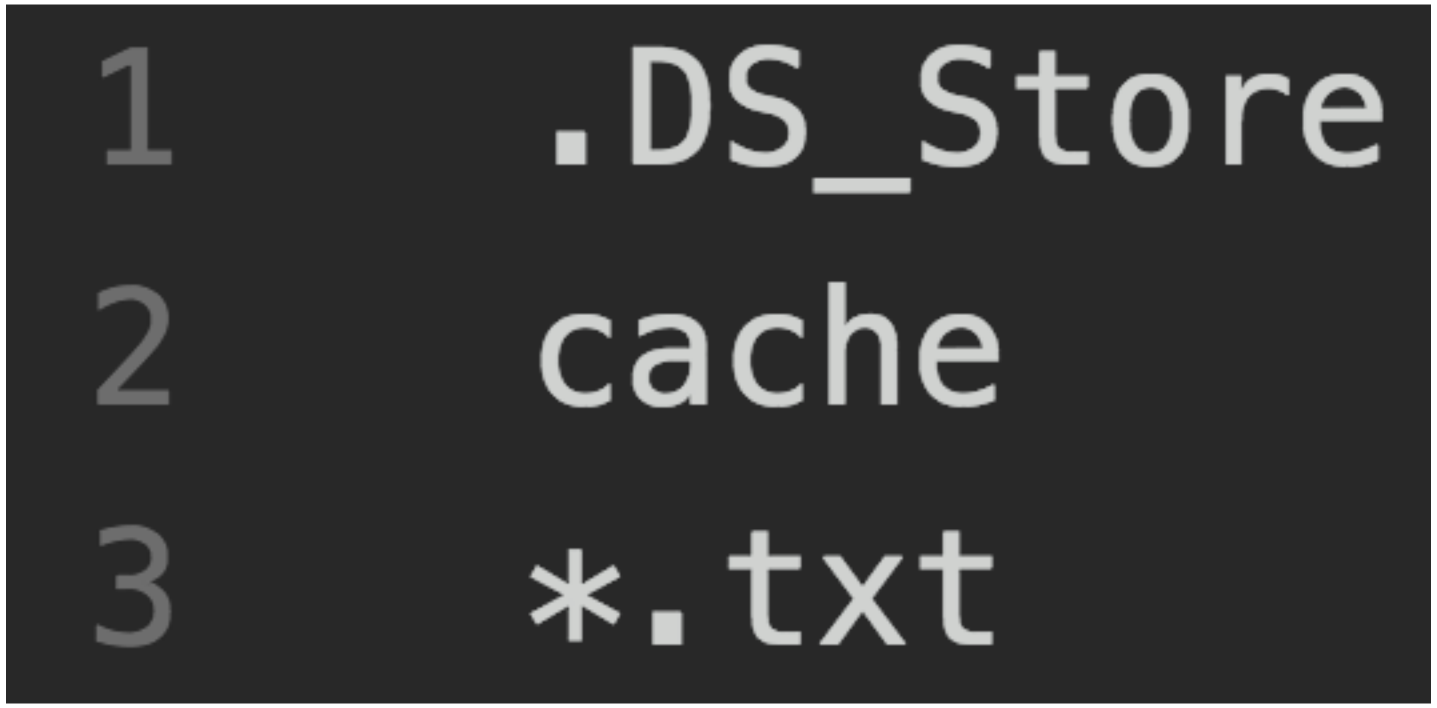
```
usage: git merge [<options>] [<commit>...]
   or: git merge [<options>] <msg> HEAD <commit>
   or: git merge --abort

    -n                    do not show a diffstat at the end of the merge
    --stat                show a diffstat at the end of the merge
    --summary             (synonym to --stat)
    --log[=<n>]           add (at most <n>) entries from shortlog to merge commit message
    --squash              create a single commit instead of doing a merge
    --commit              perform a commit if the merge succeeds (default)
    -e, --edit            edit message before committing
    --ff                  allow fast-forward (default)
    --ff-only             abort if fast-forward is not possible
    --rerere-autoupdate   update the index with reused conflict resolution if possible
    --verify-signatures   Verify that the named commit has a valid GPG signature
    -s, --strategy <strategy>
                          merge strategy to use
    -X, --strategy-option <option=value>
                          option for selected merge strategy
    -m, --message <message>
                          merge commit message (for a non-fast-forward merge)
    -v, --verbose         be more verbose
    -q, --quiet           be more quiet
    --abort               abort the current in-progress merge
```

บทสรุป

git init	-> เริ่มต้น
git remote	-> เชื่อมโยง
git clone	-> ก๊อปต้นฉบับมา
git status	-> สถานะ
git add	-> เพิ่มไฟล์
git rm	-> ลบไฟล์
git diff	-> แตกต่าง
git log	-> ย้อนหลัง
git commit	-> สัญญา
git push	-> โยนขึ้น
git pull	-> ดึงลงมา
git branch	-> สาขา
git checkout	-> ย้ายสาขา
git reset	-> ยกเลิก
git fetch	-> ดูด
git merge	-> ร่วมกัน

git ignore คือ ?

ข้ามไฟล์ที่อยู่ในรายการไป



```
.DS_Store
cache
*.txt
```