git branch --set-upstream-to=origin/master\_xc master\_xc

git reset --hard origin/master

### Rename git branch

<https://multiplestates.wordpress.com/2015/02/05/rename-a-local-and-remote-branch-in-git/>

### **Delete git branch**

<https://www.educative.io/edpresso/how-to-delete-remote-branches-in-git>

### **Show log file name**

<https://git-scm.com/docs/git-log>

<https://stackoverflow.com/questions/1230084/how-to-have-git-log-show-filenames-like-svn-log-v>

Git log -p --author=”847311033” [file]

git log -p -2

Git log --stat[OR --numstat] --oneline --pretty=[oneline/short/full/fuller]

|  |  |
| --- | --- |
| -p | 按补丁格式显示每个提交引入的差异。 |
| --stat | 显示每次提交的文件修改统计信息。 |
| --shortstat | 只显示 --stat 中最后的行数修改添加移除统计。 |
| --name-only | 仅在提交信息后显示已修改的文件清单。 |
| --name-status | 显示新增、修改、删除的文件清单。 |
| --abbrev-commit | 仅显示 SHA-1 校验和所有 40 个字符中的前几个字符。 |
| --relative-date | 使用较短的相对时间而不是完整格式显示日期（比如，“2 weeks ago”）。 |
| --graph | 在日志旁以 ASCII 图形显示分支与合并历史。 |
| --pretty | 使用其他格式显示历史提交信息。可用的选项包括 oneline，short，full，fuller 和 format（用来定义自己的格式）。 |

| Table 3. 限制 git log 输出的选项 | |
| --- | --- |
| 选项 | 说明 |
| -(n) | 仅显示最近的 n 条提交。 |
| --since, --after | 仅显示指定时间之后的提交。 |
| --until, --before | 仅显示指定时间之前的提交。 |
| --author | 仅显示作者匹配指定字符串的提交。 |
| --committer | 仅显示提交者匹配指定字符串的提交。 |
| --grep | 仅显示提交说明中包含指定字符串的提交。 |
| -S | 仅显示添加或删除内容匹配指定字符串的提交。 |

### **scenario**

The upstream branch of your current branch does not match the name of your current branch.

1) To push to the upstream branch on the remote, use

Git push origin HEAD:kingdomdong

2) To push to the branch of the same name on the remote, use

Git push origin HEAD

3) Change upstream branch to the remote king, use

git branch --set-upstream-to=origin/king

### **Remove files from index (staging area == cached)**

Git rm --cached [file]

Git’s internal state management systems (three “trees” - node and pointer-based structures)

### The commit tree(HEAD)

Is sync with local filesystem and is representative of the immediate changes made to content in files or directories.

Git status - show changes to the Working Directories

### The staging index tree

Tracking working directories

### The working directory tree

Git add ...

Update what will be committed

Git checkout -- ...

To discard changes in working directory

**git grep -R ‘keyword’ ./directory**

**Git diff filename**

**Git diff -p [filename]**

|  |  |
| --- | --- |
| git diff **head** | 工作区 vs **版本库** |
| git diff | 工作区 vs 暂存区 |
| git diff --cached | 暂存区 vs 版本库 |

暂存区（stage）-> 工作区

git reset --soft

版本库 -> 暂存区

git reset --mixed

版本库 -> 暂存区 -> 工作区

git reset --hard

放弃工作区修改

git checkout --

git reset HEAD <file>..." to unstage

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

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

创建分支

git checkout -b <BRANCH\_NAME>

创建远程分支

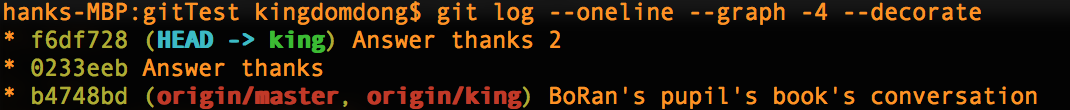
git push <REMOTE\_NAME> <BRANCH\_NAME>

E.g. git push origin king

git push <REMOTE\_NAME> <LOCAL\_BRANCH\_NAME>:<REMOTE\_BRANCH\_NAME>

E.g. git push origin king:develop

### For example:



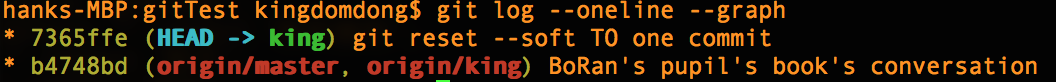
1. give up the latest two commits:

**git reset --soft head~2**



1. Merge the latest two commits to one

**git commit -m “git reset --soft TO one commit”**



### Changes from master in dev\_branch

**Scenario:** If others haven’t cloned this repository

$ Git checkout dev\_branch

$ Git reset --hard master

Scenario: if you have pushed to a remote already, you have to do:

$ git push --force

It's not the same as git add . as this would add untracked files that aren't being ignored, git add -u only stages changes (including deletions) to already tracked files.

It's not the same as git add . as this would add untracked files that aren't being ignored, git add -u only stages changes (including deletions) to already tracked files.