

○ 前回コミット時の状態に戻す

```
$ git reset --hard HEAD
```

commit からは削除するけどファイルそのものは残す

```
$ git reset --soft ハッシュ値
```

前回の commit 時に戻る

```
$ git reset --hard ハッシュ値
```

○ 前回コミットに上書き(コメントだけ書き換え)

```
$ git commit --amend
```

○ 過去のコミットの内容を修正

```
$ git rebase -i <commit>
```

```
$ git commit --amend
```

```
$ git rebase --continue
```

○ rebase を途中で中止したい

```
$ git rebase --abort
```

○ 戻りすぎた時(rebase を無かったことにする)

```
$ git reset --hard HEAD^^ # HEAD^と指定するつもりが間違えた!
```

```
$ git reflog
```

```
f5cb888 HEAD@{0}: head^^: updating HEAD
```

```
b0b8073 HEAD@{1}: merge @{-1}: Merge made by the 'recursive'  
strategy.
```

```
fe3972d HEAD@{2}: checkout: moving from fix/some-bug to master
```

```
$ git reset --hard HEAD@{1} # reset --hard HEAD^^する前に戻れる
```

○ 直前の commit メッセージの書き換え

```
$ git commit -amend -m “新しいメッセージ”
```

○ **commit ファイルを削除(一部)**

```
$ git rm [file name]
```

commit からは削除するけどファイルそのものは残す

```
$ git rm --cached [file name]
```

○ **最初の commit まで rebase**

```
$ git rebase -i --root
```

○ **現在のワークツリーを一時的に保存する**

```
$ git stash (save)
```

```
$ git stash save "message"    <- message 付き
```

○ **stash に保存されている状態の一覧を見る**

```
$ git stash list
```

○ **stash に保存されている状態に戻し、stash から削除する**

```
$ git stash pop
```

```
$ git stash pop stash@{1}
```

○ **stash から削除する**

```
$ git statsh drop
```

○ **差分のファイル名だけを表示**

```
$ git diff --name-only HEAD^
```

○ 差分を表示

```
$ git diff [file|commit name] [file2|commit2 name]
```

○ 特定のコメントを含むコミットを探す

```
$ git log -grep "<pattern>"
```

○ conflict しているファイルの一覧を表示

```
$ git ls-files -u [<path>]
```

```
$ git status で確認も
```

○ remote リポジトリの初期設定

```
$ git remote add origin git@[repository]:[user name]/[repo name].git
```

```
⇒ git hub だと...
```

```
$ git remote add origin git@github.com:foo/bar.git
```

○ remote リポジトリの URL 確認

```
$ git remote -v
```

```
=> .git/config に情報は保存されている
```

○ remote リポジトリの URL 変更

```
$ git remote set-url [old url] [new url]
```

```
ex) $ git remote set-url origin git@github.com:foo/bar.git
```

```
ex) $ git remote set-url git@github.com:foo/bar.git https://hoge hoge
```

○ remote のブランチを削除

```
$ git push origin :[branch name]
```

○ push

```
$ git push [local branch]:[remote branch]
```

```
# 普段はコロンが省略されて使われている
```

```
# local の master branch を remote の test branch へ push
```

```
# $ git push origin master:test
```

○ 簡単 push

```
$ git push -u origin master
```

○ Pull Request 手順

fork した自分のリポジトリを clone する

```
$ git clone git@github.com:[MyName]/[repository].git
```

fork 元リポジトリ(original)を upstream に登録

```
$ git remote add upstream git@github.com:[Name]/[repository].git
```

ブランチで作業して push

```
$ git push origin [local branch]:[branch]
```

push 後 github からプルリクする

プルリク前に rebase しておく

```
$ git checkout -b pullRequest
```

作業用 branch 上の fork 元からの差分を 1 commit にまとめる

```
$ git rebase -i master
```

○ Pull Request merge 手順

変更を取得

```
$ git fetch
```

remote のブランチを確認して

```
$ git checkout -b [pull request branch name] origin/[same name]
```

merge して push する

○ Git の色を変える

```
$ git config --global color.ui true
```

○ Git 便利 alias

```
$ git config --global alias.co checkout  
$ git config --global alias.br branch  
$ git config --global alias.ci commit  
$ git config --global alias.st status
```