**◯　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://hogehoge

◯　**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