

---

# Git Cheatsheet Documentation

*Release 0.0.1*

**Alpesis**

**Dec 21, 2017**



---

## Contents

---

<b>1</b>	<b>Basic</b>	<b>3</b>
1.1	New Init . . . . .	3
1.2	Staging and Remotes . . . . .	4
1.3	Cloning and Branching . . . . .	4
1.4	History and Configuration . . . . .	5
<b>2</b>	<b>Collaboration</b>	<b>7</b>
2.1	Collaboration . . . . .	7
2.2	Branching . . . . .	7
2.3	Rebase . . . . .	8



## Table of Contents



# CHAPTER 1

---

## Basic

---

### 1.1 New Init

to get help

```
$ git help
$ git help config
```

to create a user for all repos

```
$ git config --global user.name "XX"
$ git config --global user.email xxx@xx.com
$ git config --global color.ui true
```

to creat a new repo

```
$ mkdir store
$ cd store
$ git init
```

to check the status

```
$ git status
```

to add and commit files

```
$ git add index.html
$ git commit -m "new file index.html created"

$ git add readme.md LICENSE
$ git add --all
$ git commit -m "Add LICENSE and finish README."

$ git add *.txt
$ git add "*.css"
```

```
$ git add css/*.css
$ git add css/
```

to read the log file

```
$ git log
```

## 1.2 Staging and Remotes

to compare

```
$ git diff
$ git diff --staged
```

to unstage

```
$ git checkout (filename)
$ git checkout -- cats.html index.html
$ git reset ostrich.html
```

to stage and commit, modify and commit

```
$ git commit -a -m "file updated"
$ git commit --amend -m "xxxx"
```

undo the commit

```
$ git reset --soft HEAD^
$ git reset --hard HEAD^
$ git reset --hard HEAD^^
$ git reset HEAD LISENCE
```

to add a remote

```
$ git remote add origin https://github.com/XXX/xx.git
$ git remote set-url origin https://xxxx
$ git remote -v
$ git push -u origin master
$ git pull
```

## 1.3 Cloning and Branching

to clone a repo

```
$ git clone git@example.com:example/petshop.git
```

to create and edit a branch

```
$ git branch cat
$ git checkout cat
$ git add cat.txt
$ git commit -m "xxxx"
```



```
$ git checkout master
$ ls

$ git merge cat

$ git checkout -b admin
$ git pull
$ git add readme.txt
$ git commit -m "xxxx"
$ git push
```

## 1.4 History and Configuration

### log

```
$ git config --global color.ui true
$ git log
$ git log --pretty=oneline
$ git log --pretty=format:"%h %ad- %s [%an]"
$ git log --oneline -p
$ git log --oneline --stat
$ git log --oneline --graph
$ git log --until=1.minute.ago
$ git log --since=1.day.ago
$ git log --since=1.hour.ago
$ git log --since=1.month.ago --until=2.weeks.ago
$ git log --since=2000-01-01 --until=2012-12-21
$ git show <id>
```

### diff

```
$ git diff
$ git diff HEAD
$ git diff HEAD^
$ git diff HEAD^^
$ git diff HEAD~5
$ git diff HEAD^..HEAD

$ git diff f5a6sdfsfsdfsdf9..4sdsdfsdfsdfsb063f
$ git log --oneline
$ git diff 4fb063f..f5a6ff9

$ git diff master bird
$ git diff --since=1.week.ago --until=1.minute.ago
```

### blame

```
$ git blame index.html --date short
```

### excluding files

```
$ git status
$ git rm README.txt
$ git status
```

```
$ git commit -m "Remove readme"
$ git rm --cached development.log
```

### untracking files

```
$ git add .gitignore
$ git commit -m "Ignore all log files."
```

### config

```
$ git config --global user.name "Gregg Pollack"
$ git config --global user.email "gregg@codeschool.com"
$ git config --global core.editor emacs
$ git config --global core.editor /usr/bin/vim
$ git config --global merge.tool opendiff
$ git config --local user.name "<your_name>"
$ git config --local user.email "<your_email>"

$ git config user.email "spamme@example.com"
$ git config --list
$ git config user.email

$ git config --global alias.mylog \
  "log --pretty=format:'%h %s [%an]'" --graph"

$ git config --global alias.lol \
  "log --graph --decorate --pretty=oneline --abbrev-commit --all"

$ git mylog

$ git config --global alias.st status
$ git config --global alias.co checkout
$ git config --global alias.br branch
$ git config --global alias.ci commit
$ git config --global alias.logtree "log --color --graph --pretty=format:'%Cred%h
→%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset' --abbrev-
→commit"
$ git st
```

### 2.1 Collaboration

to edit

```
$ git pull
$ git push

$ git fetch
$ git merge origin/master

$ git commit -a

# update the fork
$ git remote add upstream <git://url/xx.git>
$ git fetch upstream
$ git pull upstream <master>
```

### 2.2 Branching

to update branch

```
$ git checkout -b shopping_cart
$ git push origin shopping_cart

$ git add cart.rb
$ git commit -a -m "Add basic cart ability."
$ git push

$ git pull
```

to list all remote branches

```
$ git branch
$ git branch -r
$ git checkout shopping_cart
$ git remote show origin
```

to delete branch

```
$ git push origin :shopping_cart
$ git branch -d shopping_cart
$ git remote prune origin
```

to deploy

```
$ git push heroku-staging staging:master
```

to tag

```
$ git tag
$ git checkout v0.0.1
$ git tag -a v0.0.3 -m "version 0.0.3"
$ git push --tags
```

## 2.3 Rebase

```
$ git commit -am "Update the readme."
$ git fetch origin
$ git rebase origin
$ git checkout admin
$ git rebase master
$ git checkout master
$ git merge admin
$ git status
$ git rebase --continue
```