```
## diff


# head / working dir
$ git diff
$ git difftool

# commit / working dir
$ git difftool master

# commit1 / commit2
$ git difftool master dev

# commit1 / commit2 for file.txt
$ git difftool master dev dir/file.txt
$ git difftool master dev -- "dir/file.txt"

# commit1:file1 / commit2:file2
$ git difftool dev:README.md master:rm.Extensions/Heap.cs

# diff of last 2 commits combined w/ head
$ git difftool head~2 head
# diff of last 2 commits combined w/ working dir
$ git difftool head~2
# diff of last 2 commits combined w/ staged working dir
$ git difftool head~2 --staged

# char level diff
$ git diff --color-words=.
```

```
## ancestry (parent) refs


# ^     parent by breadth
# ~     parent by depth

# ^     1st parent (same as ^1)
# ~     1st parent (same as ~1)
# ^2    2nd parent
# ~2    grandparent (1st parent's 1st parent)

# ^!    no parents (roughly, ^ for parent, ! for no)

# How to see a diff of a commit?
$ git diff 4a5b6c7~ 4a5b6c7
$ git diff 4a5b6c7^!
```

```
## commit ranges


# What's new on origin/master?
(master)
$ git log ..origin/master

# double dot, triple dot

#               dev..master = { d }
#
#   a---b---c---d master
#            \            dev...master = { d, e }
#             --e dev
#
#               master..dev = { e }
#
#               dev master = { a, b, c, d, e }

# .. in e, but not in d
$ git log d..e

# ... in d or in e but not both
$ git log d...e

# ... in d or e or both
$ git log d e
```

```
## commit --amend


$ git commit

#              topic
#   ---o<--o<--A
#

$ git commit --amend

#   ---o<--o<--A
#           \
#            A' topic
```

```
## stash


# -u includes untracked files
$ git stash save -u "comment"

# pop will remove if no conflicts, apply does not
$ git stash apply|pop

# v2.11+, no need to type stash@{1}
$ git stash apply|pop 1

#      // this is the stashed commit
#
#      *-.   WIP on dev
#      |\ \
#      | | |
#      | | * untracked files on dev
#      | |
#      | |
#      | * index on dev
#      |/
#      * (dev)        // point from where `git stash save -u` is ran
#

# use --index to restore index
$ git stash apply 3 --index
```

commit message convention 54/*

```
    subject is 54 chars only

    body is freeform. this can go on and on and on and on and on and on.
```

```
$ git log --oneline -1
8624263 (HEAD -> dev) subject is 54 chars only
```

Please do not do this:

```
    my subject is more than 54 chars so i decided to break
    it down in this horrible, terrible, criminal way

    body is freeform. this can go on and on and on and on and on and on.
```

```
$ git log --oneline -1
8624263 (HEAD -> dev) my subject is more than 54
chars so i decided to break it down in this horrible,
terrible, criminal way
```

There are exceptions:
- Auto-generated messages
- Quoted text

```
    Merge remote-tracking branch 'origin/really-long-branch-name' into next

    body is freeform. this can go on and on and on and on and on and on.
```

Commits are forever, branches are not.
- Try not to make typos in commits
- Try not to make mistakes by putting incorrect ticket

branch

```
# What branches is my branch merged into?
$ git fetch
$ git branch -r --contains origin/branch

# tracking branch info
$ git branch -v  # verbose
$ git branch -vv # very verbose

# force move a branch
$ git branch -f <branch> <commit|branch>
$ git branch -f dev origin/dev
```

```
## checkout/clean


# undo all unstaged changes
$ git checkout .

# undo a file's unstaged changes
$ git checkout file.txt

# delete untracked files, dirs, build files
$ git clean -f [-d] [-x]

# delete interactive
$ git clean -i

# alias: git undo
$ git reset --hard && git clean -f

# to delete dirs also
$ git undo -d
```