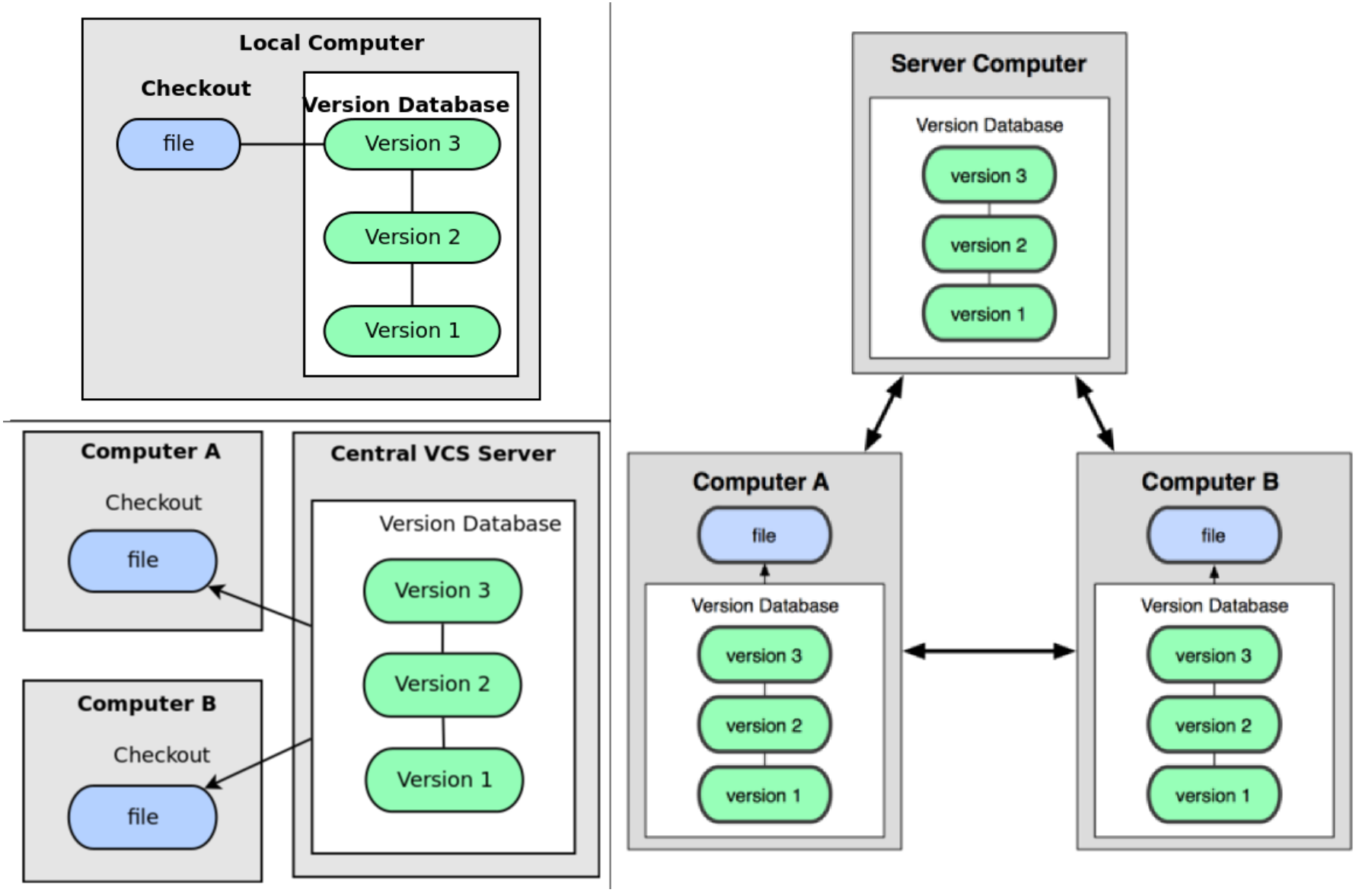
A Practical Introduction to Git

Maktab 64 - Reza Yazdani

Version Control: Local, Centralized, Distributed



From *Pro Git*, S.Chacon 2009, CC 3.0 license.

Survey: `git`

- Q1: Have you heard about **`git`**?
- Q2: Do you use **`git`**?
- Q3: Why the "**`git`**" name? (from **`git`** FAQ)
  1. Random three-letter combination that is pronounceable.
  2. Acronym (global information tracker).
  3. Irony.

`git`? Why “`git`”?

Linus Torvalds: *“I name all my projects after myself. First Linux, now* ***`git`****.”*

http://www.merriam-webster.com/dictionary/git

[1]git *noun* \'git\

Definition of GIT

British **:** a foolish or worthless person

Examples of GIT

- That *git* of a brother of yours has ruined everything!
- <oh, don't be such a silly *git*, of course your mates want you around>

Origin of GIT

variant of *get,* term of abuse, from [2]*get*

First Known Use: 1929

Related to GIT

Synonyms: berk [*British*], booby, charlie (*also* charley) [*British*], cuckoo, ding-a-ling, dingbat, ding-dong, dipstick, doofus [*slang*], featherhead, fool [*British*], goose, half-wit, jackass, lunatic, mooncalf, nincompoop, ninny, ninnyhammer, nit [*chiefly British*], nitwit, nut, nutcase, simp,

git

```
usage: git [OPTIONS] COMMAND [ARGS]

The most commonly used git commands are:
   add          Add file contents to the index
   commit       Record changes to the repository
   diff         Show changes between commits, commit
   ...
```

git help <command>

git status

Introduce yourself to `git`:

git config --global user.name "maktab Teams"

git config --global user.email "maktab@Teams.tm"

`git`. Single developer + local repository.

Scenario 1: single developer + local repository.

Single+Local `git`. Motivations.

- **Q:** do you use VC for local repo?
- Why VC for single developer + local repository?
  - First step towards a shared project.
  - Backup.
  - To keep the memory of your work.

Single+Local `git`. Init.

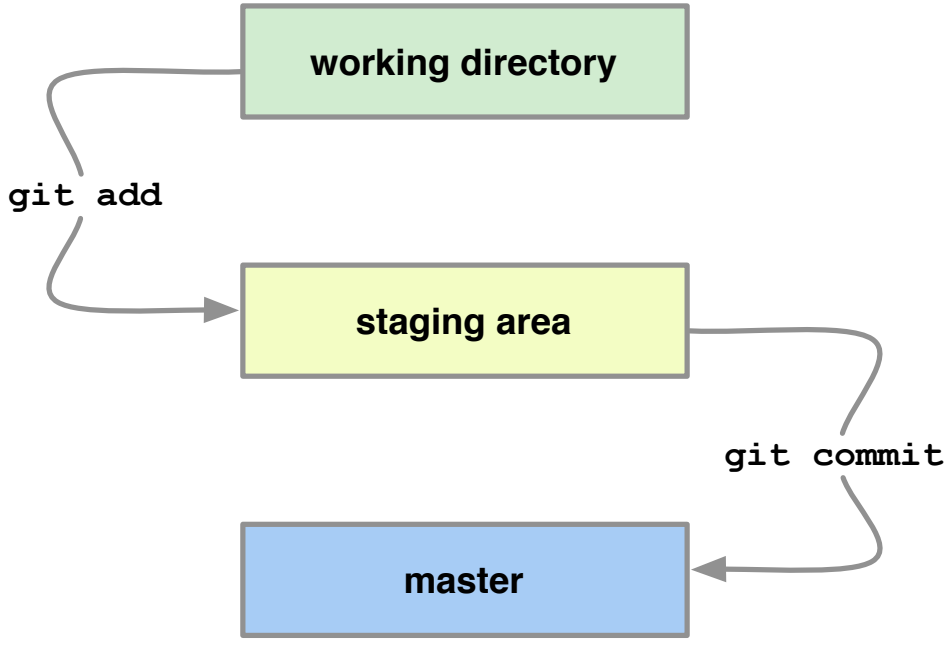
```
git init
```

- Creates an empty **`git`** repository.
- Creates the git directory: `.git/`

Note: it is **safe**. It does not change your pre-existing files.

Single+Local `git`. The tracking process.

```
git add <filename>
```




```
git commit -m "Let us begin."
```

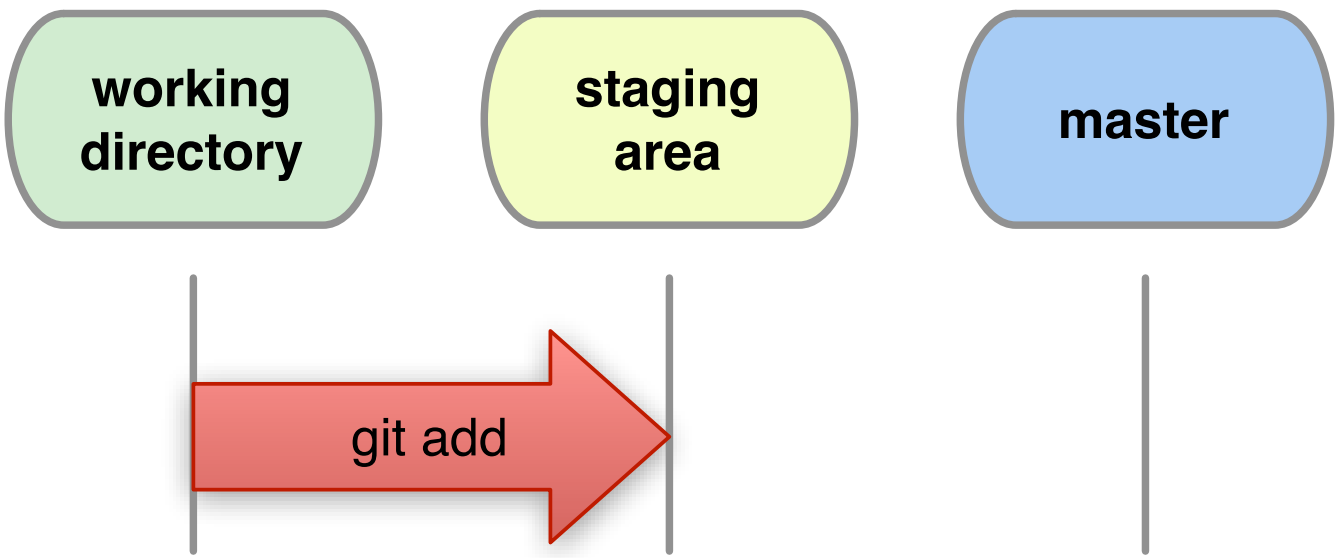
Wikipedia

"A *staging area* is a location where organisms, people, vehicles, equipment or material are assembled before use".

Single+Local `git`. Add.

```
git add file1 [file2 ...]
```

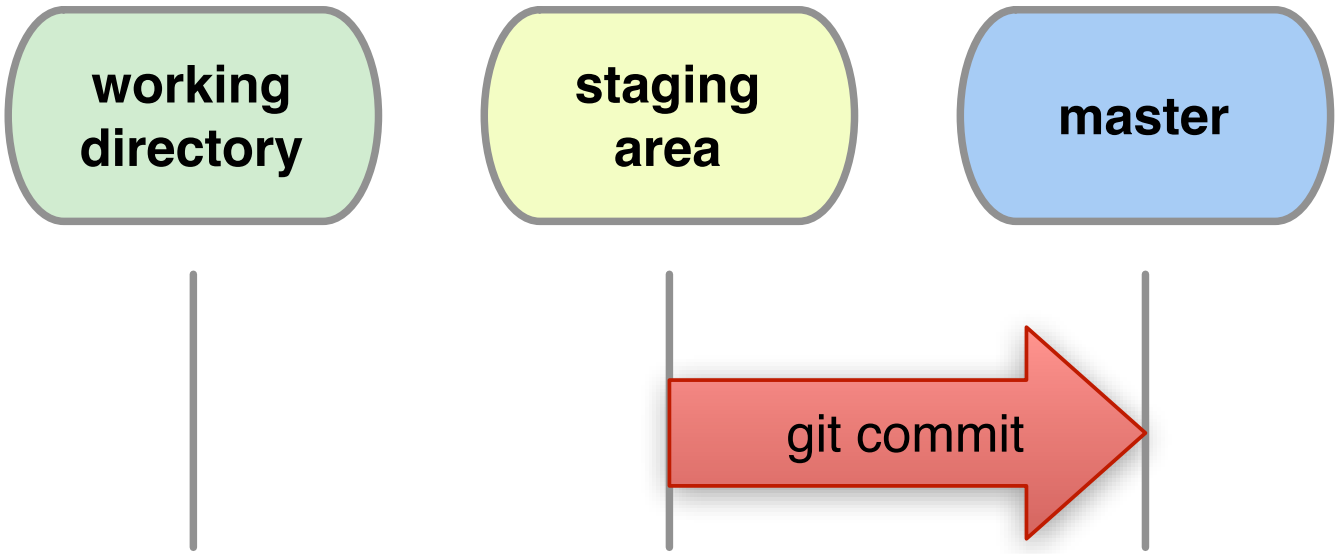
- Adds new files for next commit.
- Adds content from working dir to the staging area (index) for next commit.
- DOES NOT add info on file permissions other than *exec/noexec* (755 / 644).
- DOES not add directories *per se*.



Single+Local `git`. Commit.

```
git commit [-m "Commit message."]
```

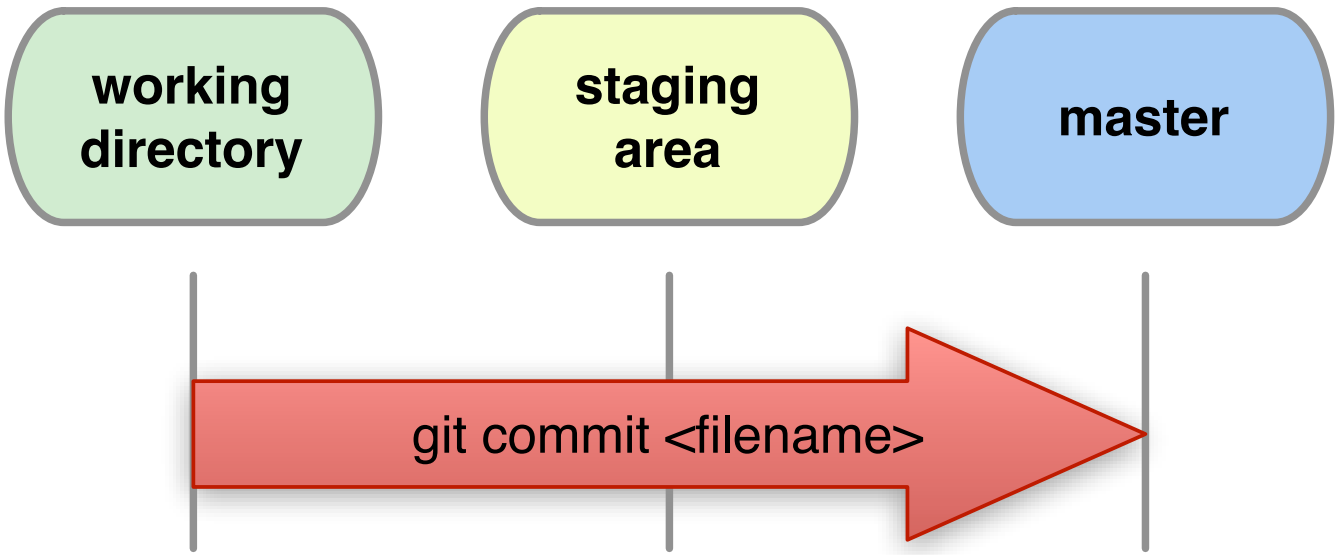
Records changes from the staging area to master.



Single+Local `git`. Commit.

`git commit file1 file2`

Records all changes of **`file1`**, **`file2`** from working dir and staging area to master.



`git commit -a`

Records all changes in working dir and staging area. *Be Careful!*

Single+Local `git`. Commit names. OPTIONAL

- Every *commit* is a **`git`**-object.
- The history of a project is a graph of objects referenced by a 40-digit **`git`**-name: *SHA*1(*object*).
- *SHA*1(*object*) = 160-bit Secure Hash Algorithm.
- Examples:

```
$ git commit README -m "Added README."
[master dbb4929] Added README.
1 files changed, 1 insertions(+), ...
```

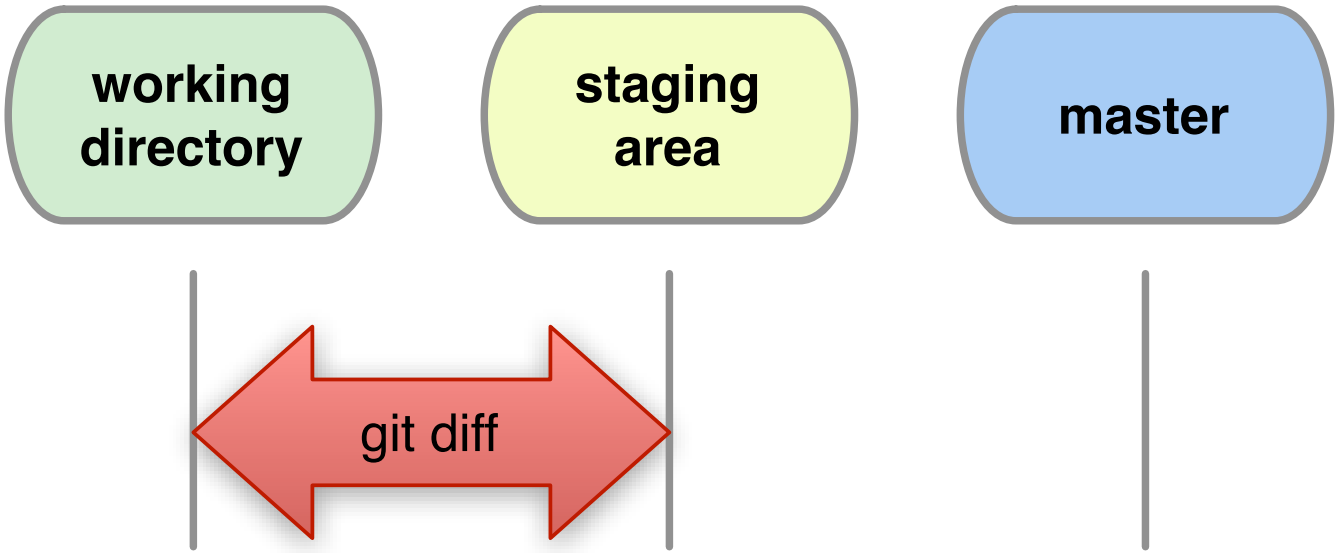
or

```
$ git log
commit dbb49293790b84f0bdcd74fd9fa5cab0...
Author: Emanuele Olivetti <olivetti@fbk.eu>
Date:   Wed Sep 15 00:08:46 2010 +0200
...
```

Single+Local `git`. Diff.

`git diff`

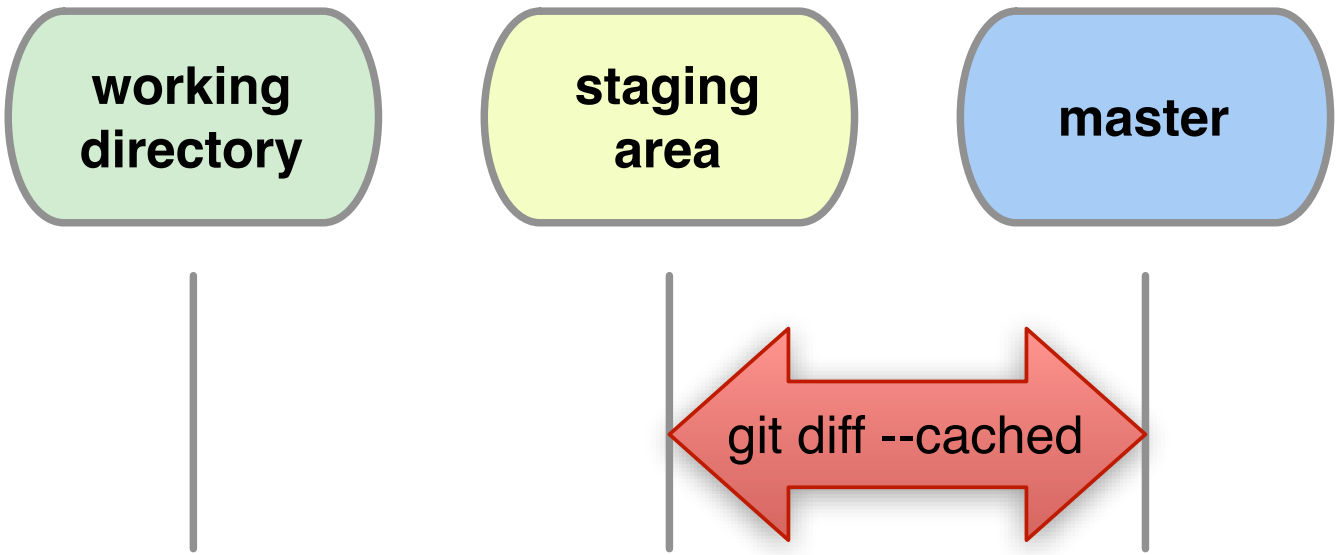
Shows what changes between *working directory* and *staging area* (*index*).



Single+Local `git`. Diff. OPTIONAL

Q: "`git add`" then "`git diff`". What output?

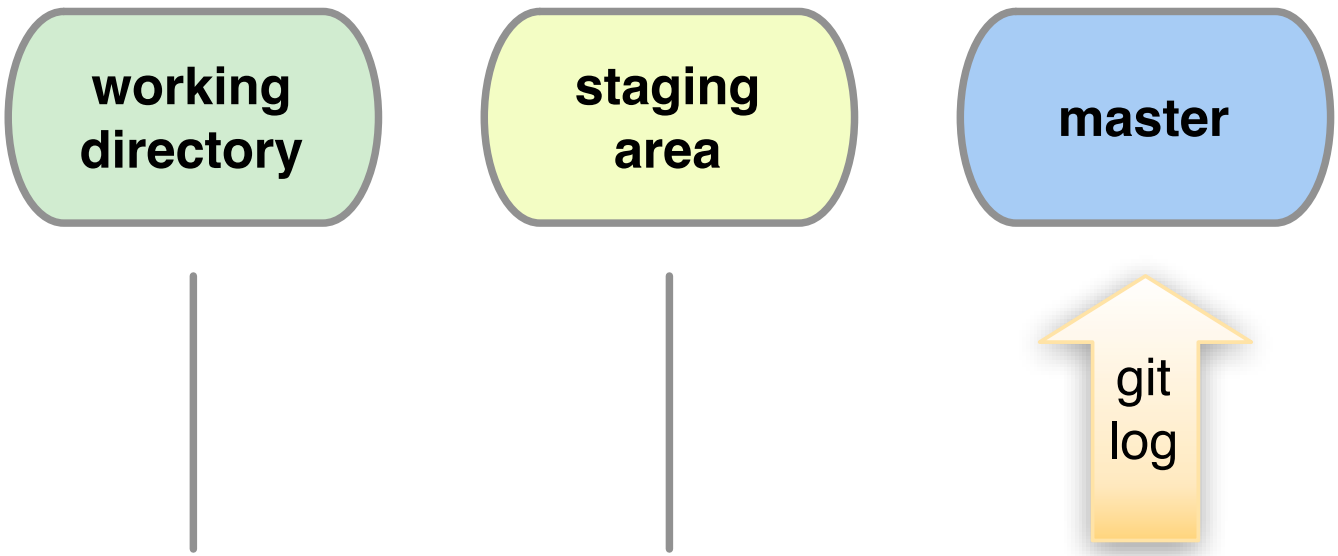
`git diff --cached` shows differences between index and last commit (HEAD).



Single+Local `git`. Logs.

`git log`

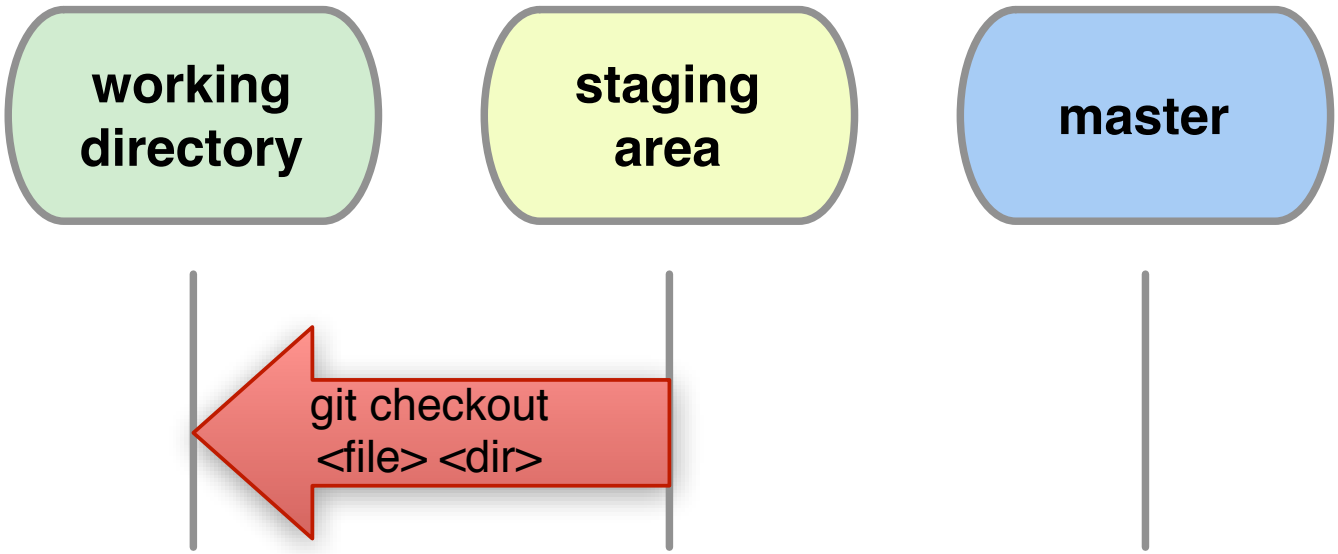
Shows details of the commits.



Single+Local `git`. “How to clean this mess??” OPT.

```
git checkout <filename>
```

Get rid of what changed in **`<filename>`** (between working dir and staging area).



Single+Local **git**. Time travelling. OPTIONAL

Back to the past when you did commit `dbb49293790b84...`

```
git checkout dbb4929
```

...and now, *back to the present!*

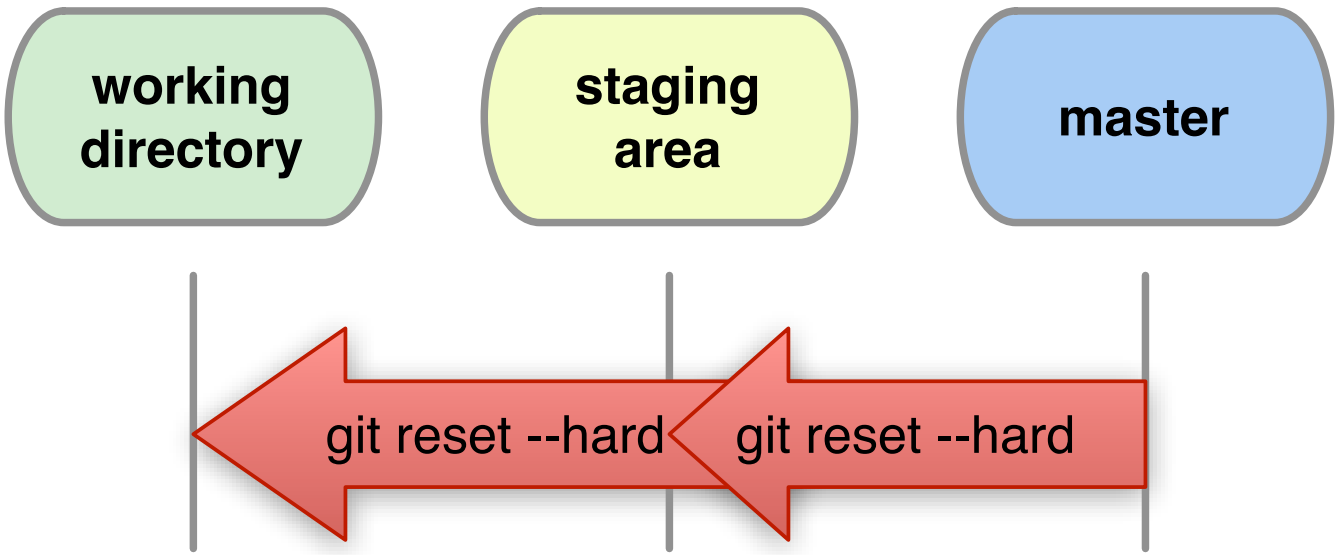
```
git checkout master
```

Single+Local `git`. “How to clean this mess??”. OPT.

First read *carefully* **`git status`**. If you panic:

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

Restore all files as in the last commit.



Warning: `reset` can destroy history!

Single+Local **git**. (Re)move. OPTIONAL

Warning: whenever you want to remove, move or rename a tracked file use **git**:

```
git rm <filename>
```

```
git mv <oldname> <newname>
```

Remember to **commit** these changes!

```
git commit -m "File (re)moved."
```