

Cours IUT current version 2016-09

An Introduction to GIT

arnaud.nauwynck@gmail.com

This document:

<http://arnaud.nauwynck.free.fr/CoursIUT/CoursIUT-IntroGIT.pdf>

Outline

- What is GIT ?
- History: from SVN centralised to Github social
- Links, Documentation
- Principles
- Commands
- Cheat Sheets – Visual Commands

What is GIT ? ... Google It (not a play on word?)

I will use Google before asking dumb questions. I will use Google before asking dumb questions.

Google

Recherche

Environ 239 000 000 résultats (0,29 secondes)

Web

Images

Maps

Vidéos

Actualités

Shopping

Plus

Nanterre

Changer le lieu

Le Web

Pages en français

Pays : France

Pages en langue étrangère traduites

Plus d'outils

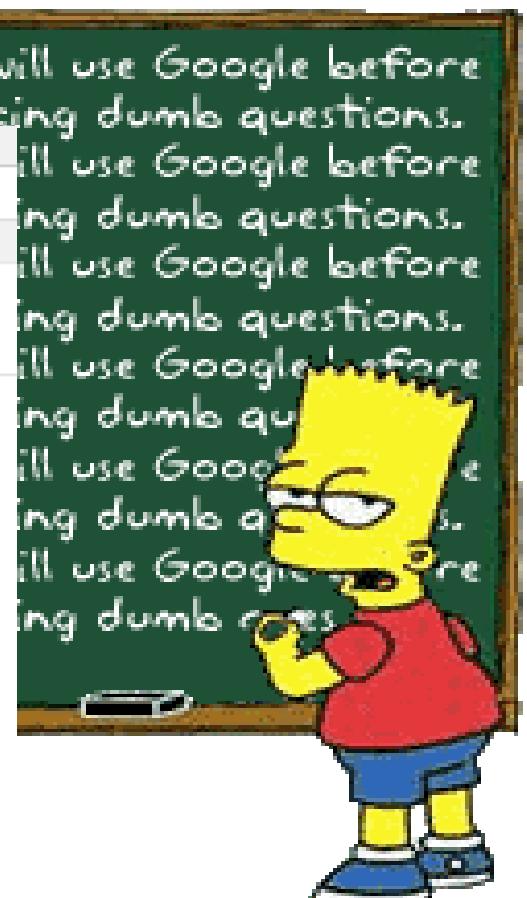
git

[Git](#)
[git-scm.com/](#) - Traduire cette page
Official website. Provides documentation, tutorials and download links.
[Downloads](#) · [Documentation](#) · [Book](#) · [Windows](#)
Vous avez consulté cette page 5 fois. Dernière visite : 22/08/11

[Git - Wikipédia](#)
[fr.wikipedia.org/wiki/Git](#)
Git est un logiciel de gestion de versions décentralisé. C'est un logiciel libre créé par Linus Torvalds, le créateur du noyau Linux, et distribué selon les termes de ...
[Particularités techniques](#) · [Fonctionnement](#) · [Quelques commandes](#) · [Interfaces](#)

[Git Extensions | Free software downloads at SourceForge.net](#)
[sourceforge.net/projects/gitextensions/](#) - Traduire cette page
Git Extensions is a toolkit to make working with **Git** under Windows more intuitive. The shell extension will intergrate in Windows Explorer and presents a nice ...
Vous avez consulté cette page 2 fois. Dernière visite : 16/11/11

[GitHub · Social Coding](#)
<https://github.com/>
Online project hosting using **Git**. Includes source-code browser, in-line editing, wikis, ticketing and more. Free for public open-source code. Commercial closed ...
Vous avez consulté cette page 5 fois. Dernière visite : 18/02/12



[http://en.wikipedia.org/wiki/Git\(software\)](http://en.wikipedia.org/wiki/Git_(software))



WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia

Interaction
Help
About Wikipedia
Community portal

Article [Talk](#)

[Read](#) [Edit](#) [View history](#)

[Search](#)

Git (software)

From Wikipedia, the free encyclopedia

In software development, **Git** (/gɪt/) is a distributed revision control and source code management (SCM) system with an emphasis on speed.^[3] Git was initially designed and developed by **Linus Torvalds** for Linux kernel development; it has since been adopted by many other projects. Every Git working directory is a full-fledged repository with complete history and full revision tracking capabilities, not dependent on network access or a central server. Git is free

The image shows the official Git logo on the left, which consists of a red diamond shape containing a white 'g' with a diagonal line through it, followed by the word "git" in a bold, lowercase sans-serif font. To the right of the logo is a screenshot of a GitHub commit history page. The page title is "projects / devliblist / commitdiff". It shows a single commit from "author: Linus Torvalds" dated "Mon, 1 Feb 2005 07:30:00 -0800 (PST)". The commit message is "Changed do... while to while, and added a line to test.c to test it." Below the commit details is a table showing file changes: "diffstat" and "diff". At the bottom of the screenshot, there is a section titled "diff --git" with some explanatory text about what diffstat and diff output mean.

GIT in few words

- GIT is a **SCM tool** (Source Code Management)
- The Successor of Subversion (SVN)
- Replace all backup-20120903.zip, tar.gz ...
when you **work alone locally**
- Replace all Svn, Ftp, Http, G:\shared\,
/mnt/usb/
when you **work in a network team**

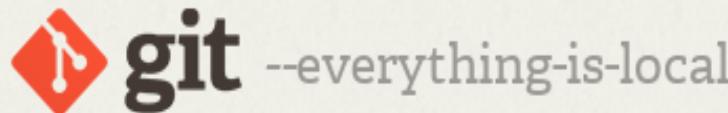
SVN (aka: Subversion)

- Problems with subversion
- Slow and bugs
- No Branches, only directories... no merge
- Centralised server + Connection needed
 - Commits are serialised identified by incremented version number
- Not Social: You want to give new ideas in branch?
 - You have to create an account on server
 - You have to create branch on server
- No workflow

GIT Solves

- FAST, rock solid
- Branches, merge
- Offline
- No centralised server
- Run locally or distributed mode
- Social
- Flexible workflows

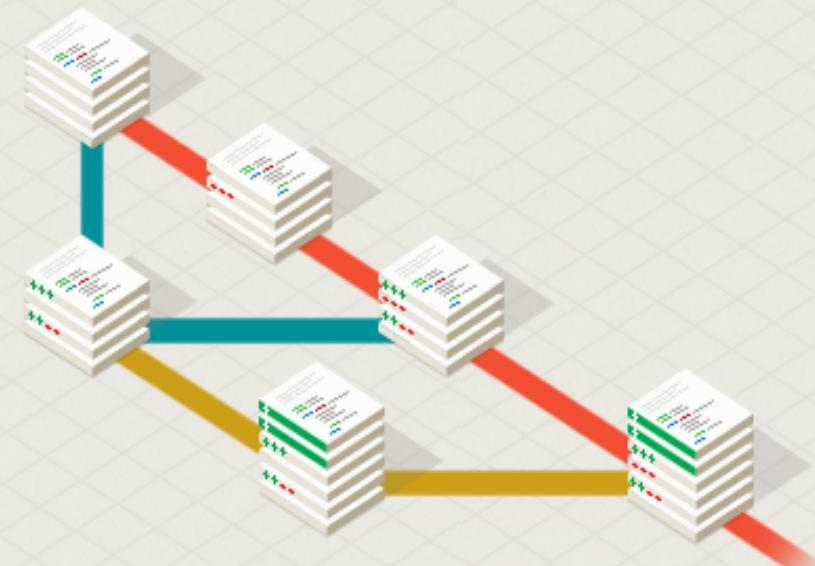
<http://git-scm.com/>



Search entire site...

Git is a **free and open source** distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is **easy to learn** and has a **tiny footprint with lightning fast performance**. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like **cheap local branching**, convenient **staging areas**, and **multiple workflows**.



About



The advantages of Git compared to other source control systems.



Documentation

Command reference pages, Pro Git book content, videos and other material.

Latest stable release

1.7.12

[Release Notes \(2012-08-20\)](#)

[Download for Linux](#)

GIT History

- Linus Torvalds
 - hated svn (non distributed, poor merge / branches...)
 - Was using “BitKeeper” for Linux source code
 - Has written “git” in few months as the replacement !
- "git," British slang meaning "a rotten person,"
Linus Torvalds: "I'm an egotistical bastard, so I
name all my projects after myself. First Linux,
now git."

Who is using it ? ~ALL & ME

Who is not using it ? (poor svn guys)

Companies & Projects Using Git

Google

facebook

Microsoft

twitter

LinkedIn

NETFLIX



PostgreSQL



This open sourced site is hosted on GitHub.
Patches, suggestions, and comments are welcome.

Git is a member of the Software Freedom Conservancy

github



2,028,332 people hosting over 3,555,467 repositories

sourceforge

<https://github.com/>

github

[Signup and...](#)

Outdated..
in 2016 : +38 M

[Sign In](#)

2,028,332 people hosting over 3,555,467 repositories

jQuery, reddit, Sparkle, curl, Ruby on Rails, node.js, ClickToFlash, Erlang/OTP, CakePHP, Redis, and [many more](#)

Find any repository



Microsoft

vmware



redhat

LinkedIn

mozilla

git /'git/

Git is an extremely fast, efficient, distributed version control system ideal for the collaborative development of software.

git·hub /'git,hAb/

GitHub is the best way to collaborate with others. Fork, send pull requests and manage all your **public** and **private** git repositories.

Plans, Pricing and Signup

Unlimited public repositories are free!

Free public repositories, collaborator management, issue tracking, wikis, downloads, code review, graphs and much more...

Create GitHub Repos

Screenshot of a GitHub user profile page for Arnaud-Nauwynck.

The top navigation bar includes the GitHub logo, a search bar, and links for Explore, Gist, Blog, Help, and the user's profile (Arnaud-Nauwynck). To the right are icons for creating a new repository, fork, and settings.

The main navigation bar below shows the user's activity: News Feed (selected), Your Actions, Pull Requests, Issues, and Stars.

A banner at the top of the feed area reads "GitHub Bootcamp" with the subtext "If you are still new to things, we've provided a few walkthroughs to get you started." It features four numbered steps:

- Set Up Git**: A quick guide to help you get started with Git. (Icon: GitHub cat with a box of Git logos.)
- Create A Repository**: Create the place where your commits will be stored. (Icon: GitHub cat inside a shipping container.)
- Fork a Repository**: Copy a repo to create a new, unique project from its contents. (Icon: GitHub cat and a blue cat standing together.)
- Be social**: Follow a friend. Watch a project. (Icon: GitHub cat surrounded by a network of smaller GitHub cats.)

Git Repository On GitHub

GitHub, Inc. (US) | https://github.com/github/gitscm-next

github Search... Explore Gist Blog Help

github / gitscm-next Watch

Code Network Pull Requests 3 Issues 37

next version of the git-scm.com site — [Read more](#)
<http://git-scm.com/>

ZIP HTTP Git Read-Only git://github.com/github/gitscm-next.git Read-Only access

branch: master Files Commits Branches 21

Latest commit to the **master** branch

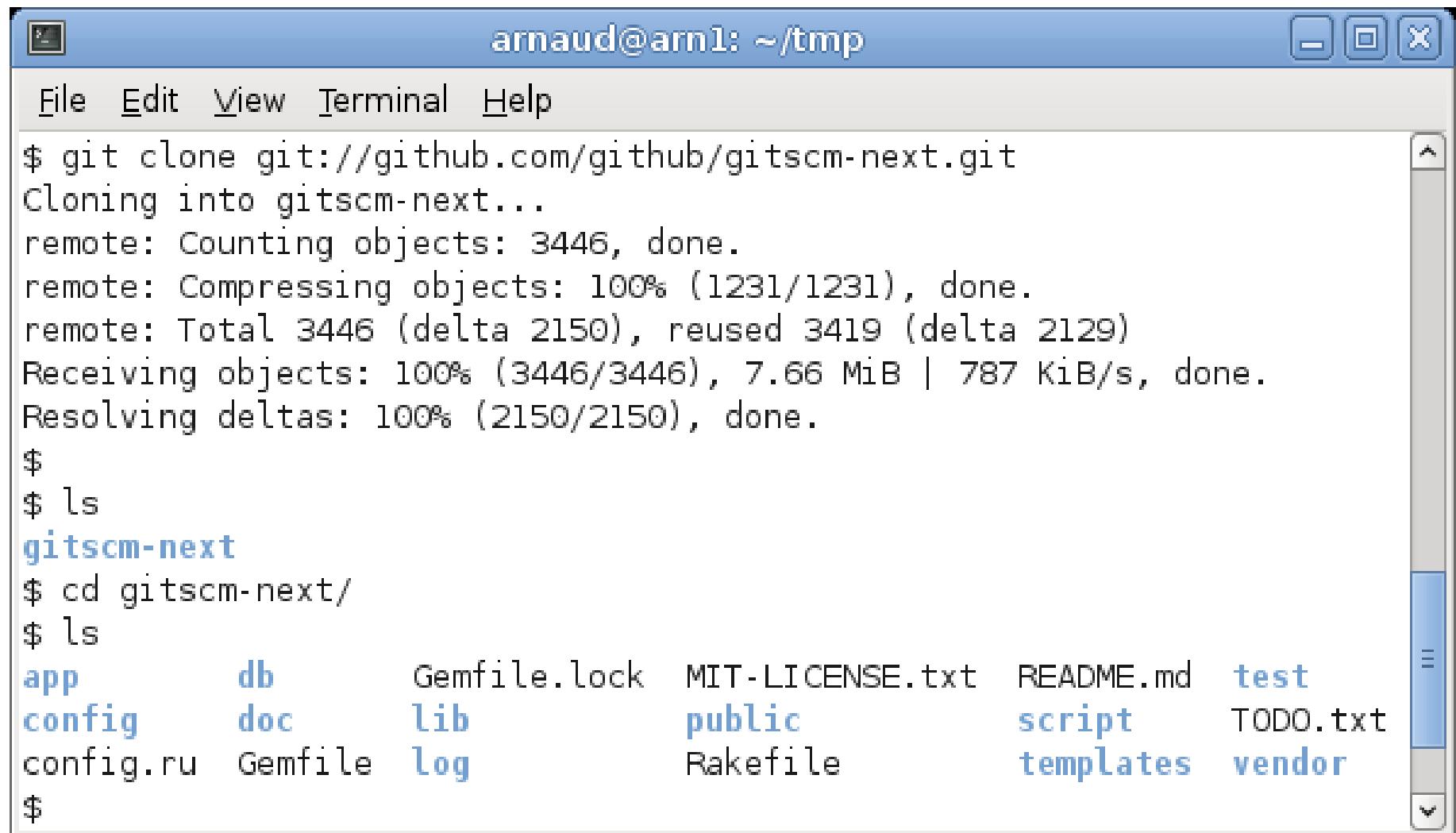
Merge pull request #150 from waitingalday/master ...

bry4n authored 10 days ago

gitscm-next /

Clone (=Download) from GIT

\$git clone <<url>> or \$ git clone --depth 1 <<url>>

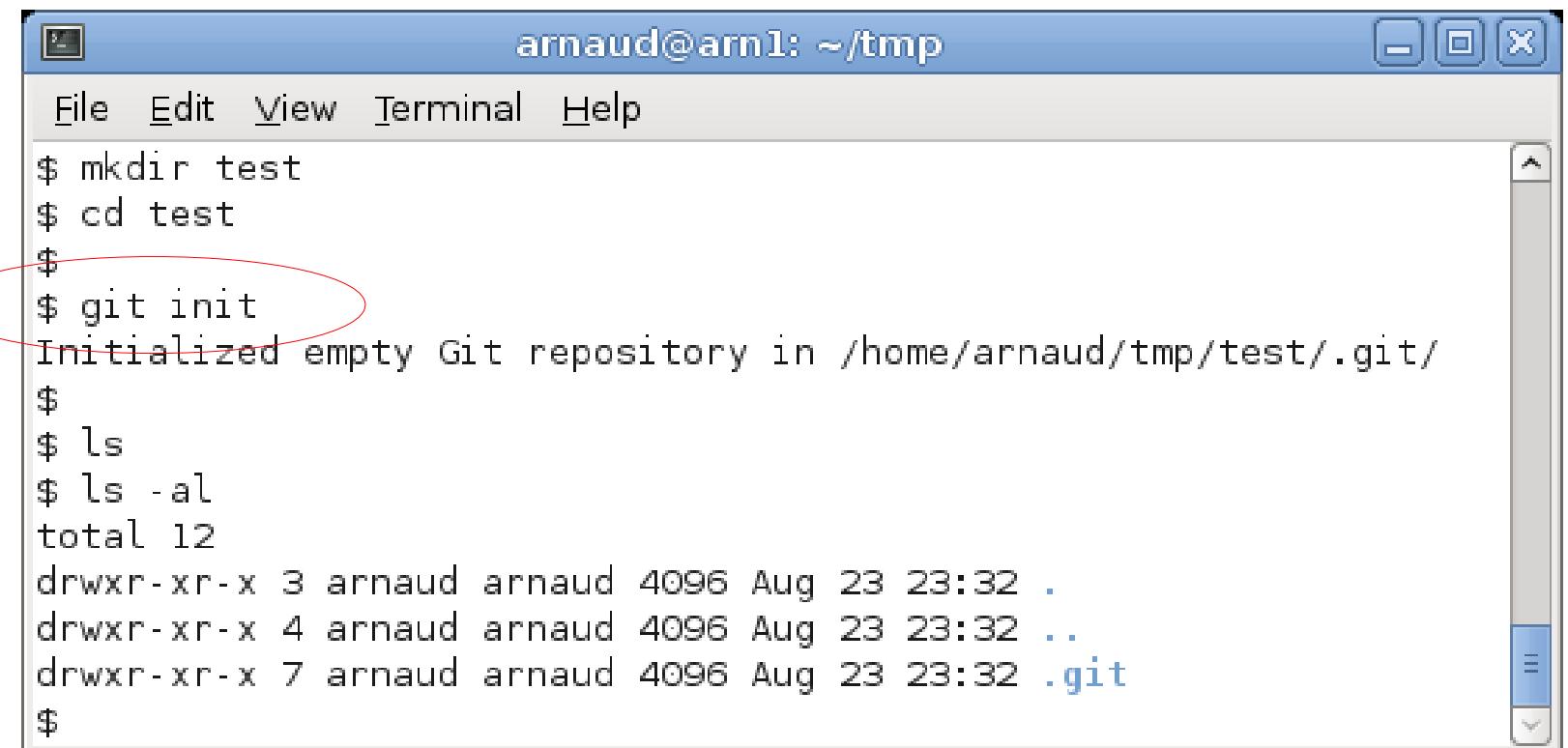


The screenshot shows a terminal window with a blue title bar. The title bar displays the user's name, arnaud@arnl, followed by the directory ~/tmp. Below the title bar is a menu bar with options: File, Edit, View, Terminal, and Help. The main area of the terminal contains the following text:

```
$ git clone git://github.com/github/gitscm-next.git
Cloning into gitscm-next...
remote: Counting objects: 3446, done.
remote: Compressing objects: 100% (1231/1231), done.
remote: Total 3446 (delta 2150), reused 3419 (delta 2129)
Receiving objects: 100% (3446/3446), 7.66 MiB | 787 KiB/s, done.
Resolving deltas: 100% (2150/2150), done.
$
$ ls
gitscm-next
$ cd gitscm-next/
$ ls
app           db          Gemfile.lock   MIT-LICENSE.txt   README.md    test
config        doc          lib            public          script      TODO.txt
config.ru     Gemfile     log            Rakefile       templates   vendor
$
```

Creating a Local Git Repo

- **\$ git init ... THAT'S ALL !!!**
- **\$ git add -a**
- **\$ git commit -m "init comment"**



The screenshot shows a terminal window titled "arnaud@arn1: ~/tmp". The window contains the following command history:

```
File Edit View Terminal Help
$ mkdir test
$ cd test
$ git init
Initialized empty Git repository in /home/arnaud/tmp/test/.git/
$ ls
$ ls -al
total 12
drwxr-xr-x 3 arnaud arnaud 4096 Aug 23 23:32 .
drwxr-xr-x 4 arnaud arnaud 4096 Aug 23 23:32 ..
drwxr-xr-x 7 arnaud arnaud 4096 Aug 23 23:32 .git
$
```

A red oval highlights the command `$ git init`, which is the focus of the slide's title.

.git : hidden local repo files

The image displays three terminal windows side-by-side, illustrating the structure of a local Git repository.

- Top-left Terminal:** Shows the command `$ ls` output, which includes `README.txt`.
- Top-right Terminal:** Shows the command `$ ls -a` output, which includes `..`, `.git`, `.gitignore`, and `README.txt`.
- Bottom Terminal:** Shows the command `$ cd .git` followed by `$ ls`, listing the hidden repository sub-directories: `branches`, `COMMIT_EDITMSG`, `config`, `description`, `HEAD`, `hooks`, `index`, `info`, `logs`, `objects`, `refs`, and `update.sample`.
- Rightmost Terminal:** Shows the command `$ tree -a -L 3` output, providing a detailed tree view of the repository structure, including `HEAD`, `branches`, `COMMIT_EDITMSG`, `config`, `description`, `hooks`, `index`, `info`, `logs`, `objects`, `refs`, `update.sample`, and various sample files like `applypatch-msg.sample` and `commit-msg.sample`.

Basic Git Commands

\$ git status

\$ git add -a

\$ git commit -m "comment"

\$ git diff

\$ git checkout

\$ git branch

\$ git merge

\$ git log

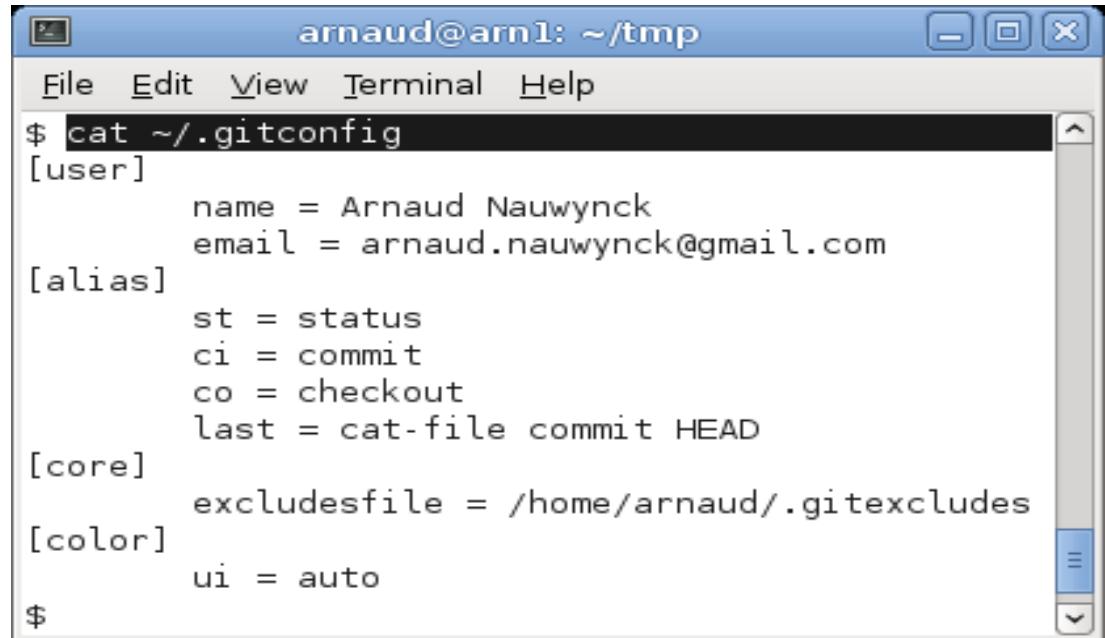
\$ git pull / push

Git Config, Git Alias ...

- Git is highly configurable
 - In <>repo>>/.git
 - <>repo>>/.gitignore
 - In ~/.gitconfig
- sample alias:
“ci” = “commit”



```
target
.classpath
.project
.settings
.svn
.gitignore (END)
```

```
arnaud@arn1: ~/tmp
$ cat ~/.gitconfig
[user]
    name = Arnaud Nauwynck
    email = arnaud.nauwynck@gmail.com
[alias]
    st = status
    ci = commit
    co = checkout
    last = cat-file commit HEAD
[core]
    excludesfile = /home/arnaud/.gitexcludes
[color]
    ui = auto
$
```

GIT Documentation(sss)

Documentation

- Reference
- Book
- Videos
- External Links

Downloads

Community

The entire [Pro Git book](#) written by Scott Chacon is available to [read online for free](#). Dead tree versions are available on [Amazon.com](#).

Documentation

Reference

[Reference Manual](#)
The official and comprehensive **man pages** that are included in the Git package itself.

Quick reference guides: [Heroku Cheat Sheet \(PDF\)](#) | [Visual Git Cheat Sheet \(SVG | PNG\)](#)

Book

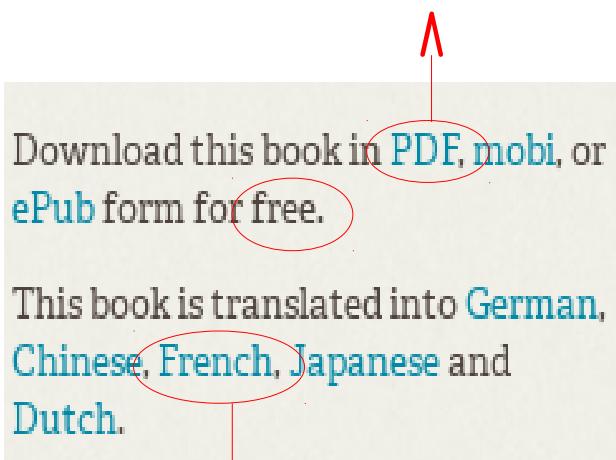
Pro Git
Scott Chacon

- 1. [Getting Started](#)
- 2. [Git Basics](#)
- 6. [Git Tools](#)
- 7. [Customizing Git](#)

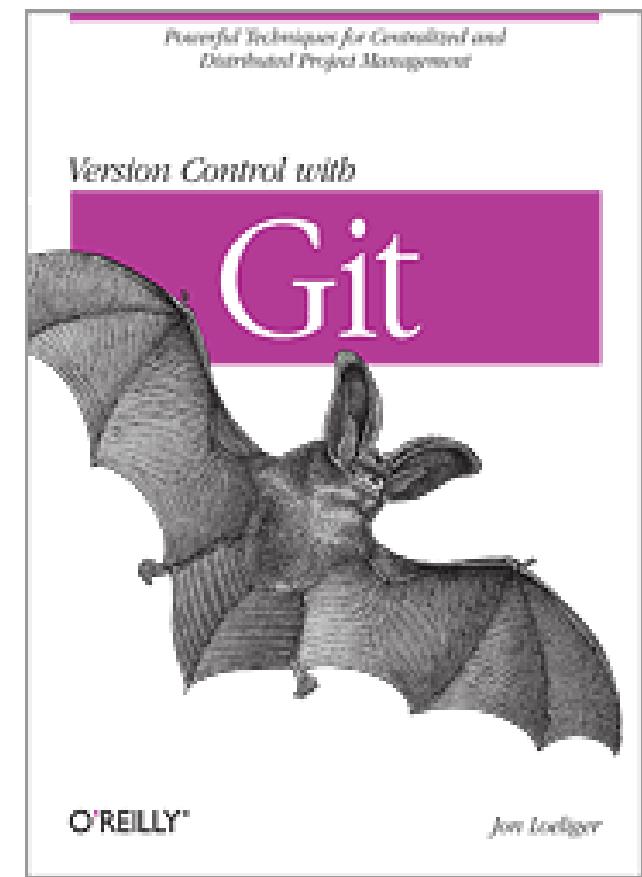
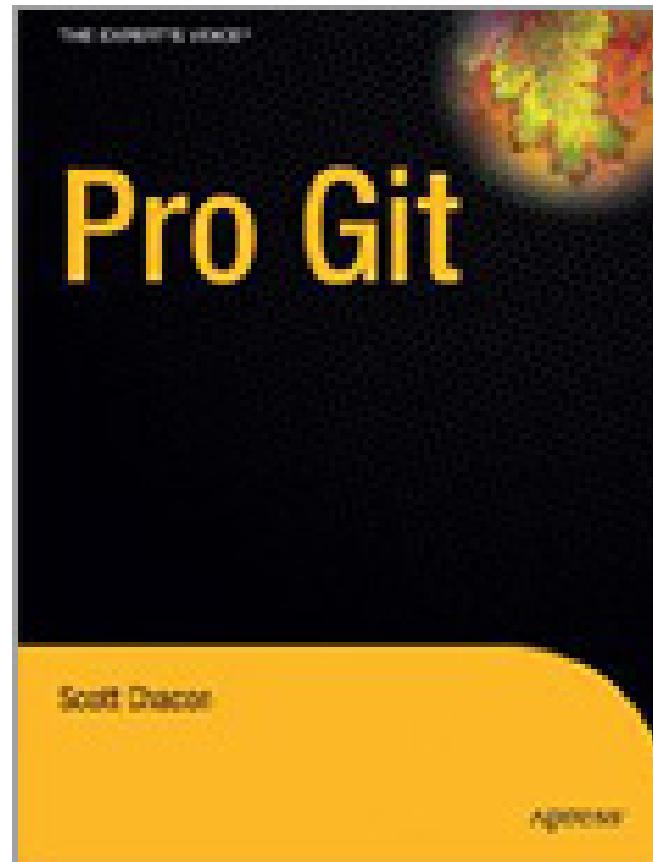
Sample Git Books

Free from Scott Chacon

<https://github.s3.amazonaws.com/media/progit.en.pdf>



<http://git-scm.com/book/fr>



Man Git ... Git Help

```
arnaud@arn1: ~
File Edit View Terminal Help

NAME
git - the stupid content tracker

SYNOPSIS
git [--version] [--exec-path[=GIT_EXEC_PATH]] [--html-path]
[-p|--paginate|--no-pager] [--no-replace-objects]
[--bare] [--git-dir=GIT_DIR] [--work-tree=GIT_WORK_TREE]
[-c name=value]
[--help] COMMAND [ARGS]

DESCRIPTION
Git is a fast, scalable, distributed revision control
system with an unusually rich command set that provides
both high-level operations and full access to internals.

See gittutorial(7) to get started, then see Everyday Git[1]
for a useful minimum set of commands, and "man
git-commandname" for documentation of each command. CVS
users may also want to read gitcvs-migration(7). See the
Git User's Manual[2] for a more in-depth introduction.

The COMMAND is either a name of a Git command (see below)
or an alias as defined in the configuration file (see git-
config(1)).

Manual page git(1) line 2
```

Git Command List : \$ git help

```
arnaud@arn1: ~
File Edit View Terminal Help
$ git help
usage: git [--version] [--exec-path[=GIT_EXEC_PATH]] [--html-path]
           [-p|--paginate|--no-pager] [--no-replace-objects]
           [--bare] [--git-dir=GIT_DIR] [--work-tree=GIT_WORK_TREE]
           [-c name=value] [--help]
           COMMAND [ARGS]
```

The most commonly used git commands are:

add	Add file contents to the index
bisect	Find by binary search the change that introduced a bug
branch	List, create, or delete branches
checkout	Checkout a branch or paths to the working tree
clone	Clone a repository into a new directory
commit	Record changes to the repository
diff	Show changes between commits, commit and tree state
fetch	Download objects and refs from another repository
grep	Print lines matching a pattern
init	Create an empty git repository or reinitialize an existing one
log	Show commit logs
merge	Join two or more development histories together
mv	Move or rename a file, a directory, or a tag
pull	Fetch from and merge with another repository's head(s)
push	Update remote refs along with associated objects
rebase	Forward-port local commits to the updated upstream tip
reset	Reset current HEAD to the specified state
rm	Remove files from the working tree and stage them for commit
show	Show various types of objects
status	Show the working tree status
tag	Create, list, delete or verify a tag

See 'git help COMMAND' for more information on a specific command

Even more ...
\$ git help --all

```
arnaud@arn1: ~
File Edit View Terminal Help
$ git help --all
usage: git [--version] [--exec-path[=GIT_EXEC_PATH]] [--html-path]
           [-p|--paginate|--no-pager] [--no-replace-objects]
           [--bare] [--git-dir=GIT_DIR] [--work-tree=GIT_WORK_TREE]
           [-c name=value] [--help]
           COMMAND [ARGS]

available git commands in '/usr/lib/git-core'
-----
add           hash-object          reflog
add-interactive   help             relink
am            http-backend        remote
annotate       http-fetch         remote-ftp
apply          http-push          remote-ftps
archive        imap-send          remote-http
bisect         index-pack         remote-https
bisect--helper  init             remote-testgit
blame          init-db            repack
branch         instaweb           replace
bundle         log               repo-config
cat-file       lost-found        request-pull
check-attr     ls-files          rerere
check-ref-format  ls-remote        reset
checkout       ls-tree           rev-list
checkout-index  mailinfo          rev-parse
cherry         mailsplit         revert
cherry-pick    merge            rm
clean          merge-base        send-pack
clone          merge-file        shell
commit         merge-index       shortlog
commit-tree    merge-octopus    show
```

\$ Man git-<<cmd>>
= ... \$ Git Help <<cmd>>

arnaud@arn1: ~

File Edit View Terminal Help

GIT-ADD(1) Git Manual GIT-ADD(1)

NAME

git-add - Add file contents to the index

SYNOPSIS

```
git add [-n] [-v] [--force | -f] [--interactive | -i] [--patch | -p]
         [--edit | -e] [--all | [--update | -u]] [--intent-to-add | -N]
         [--refresh] [--ignore-errors] [--ignore-missing] [--]
         <filepattern>...
```

DESCRIPTION

This command updates the index using the current content found in the working tree, to prepare the content staged for the next commit. It typically adds the current content of existing paths as a whole, but with some options it can also be used to add content with only part of the changes made to the working tree files applied, or remove paths that do not exist in the working tree anymore.

The "index" holds a snapshot of the content of the working tree, and it is this snapshot that is taken as the contents of the next commit. Thus after making any changes to the working directory, and before running the commit command, you must use the add command to add any new or modified files to the index.

This command can be performed multiple times before a commit. It only adds the content of the specified file(s) at the time the add command is run; if you want subsequent changes

Manual page git-add(1) line 1

Example :
\$ git help add
\$ man git-add

Understand Git from Commands?



```
$ git reset origin/some-branch^2{4}^^ --soft  
$ git commit -a --fixup=HEAD^  
detached HEAD  
conflict  
...
```

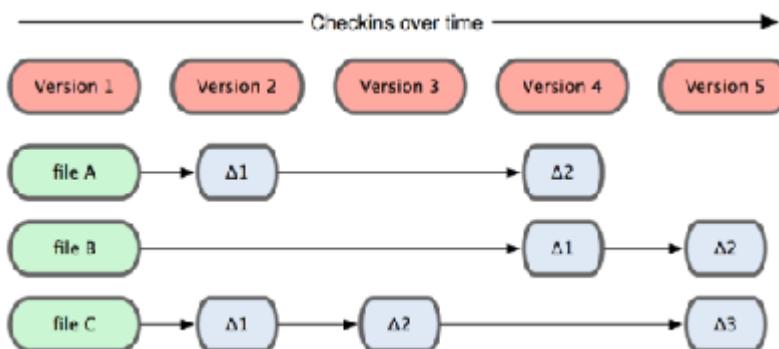
You have to understand Git from Internals



Git : Content Tracking, not DIFF

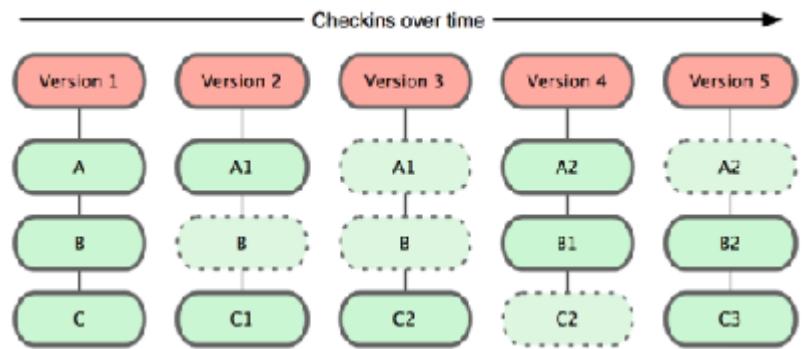
Subversion

Store DIFFs between versions



GIT

Store Full SNAPSHOTS at each point
... BLOB may be repeated from version to version



Git SHA1, Blob

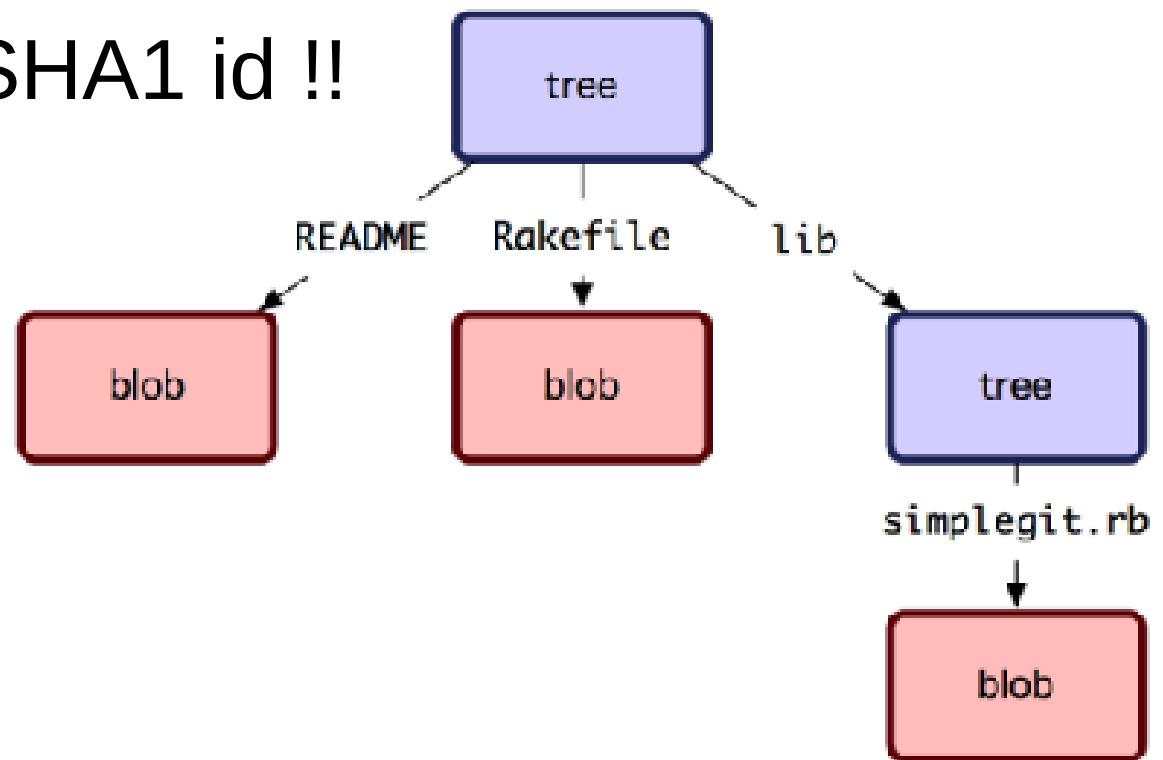
911e7..

blob	size
The MIT License Copyright (c) <year> <copyright holder> Permission is hereby granted,	

- A BLOB (content file) has a SHA1
- Independently of its file name
- SHA1 = cryptographic hash-code (256 bytes)
 - identical content => same SHA1
 - Same SHA1 => same content
(otherwise probability $1/8^{256}$ = never ever)
- File are indexed per SHA1: in .git/objects/...

Git Tree

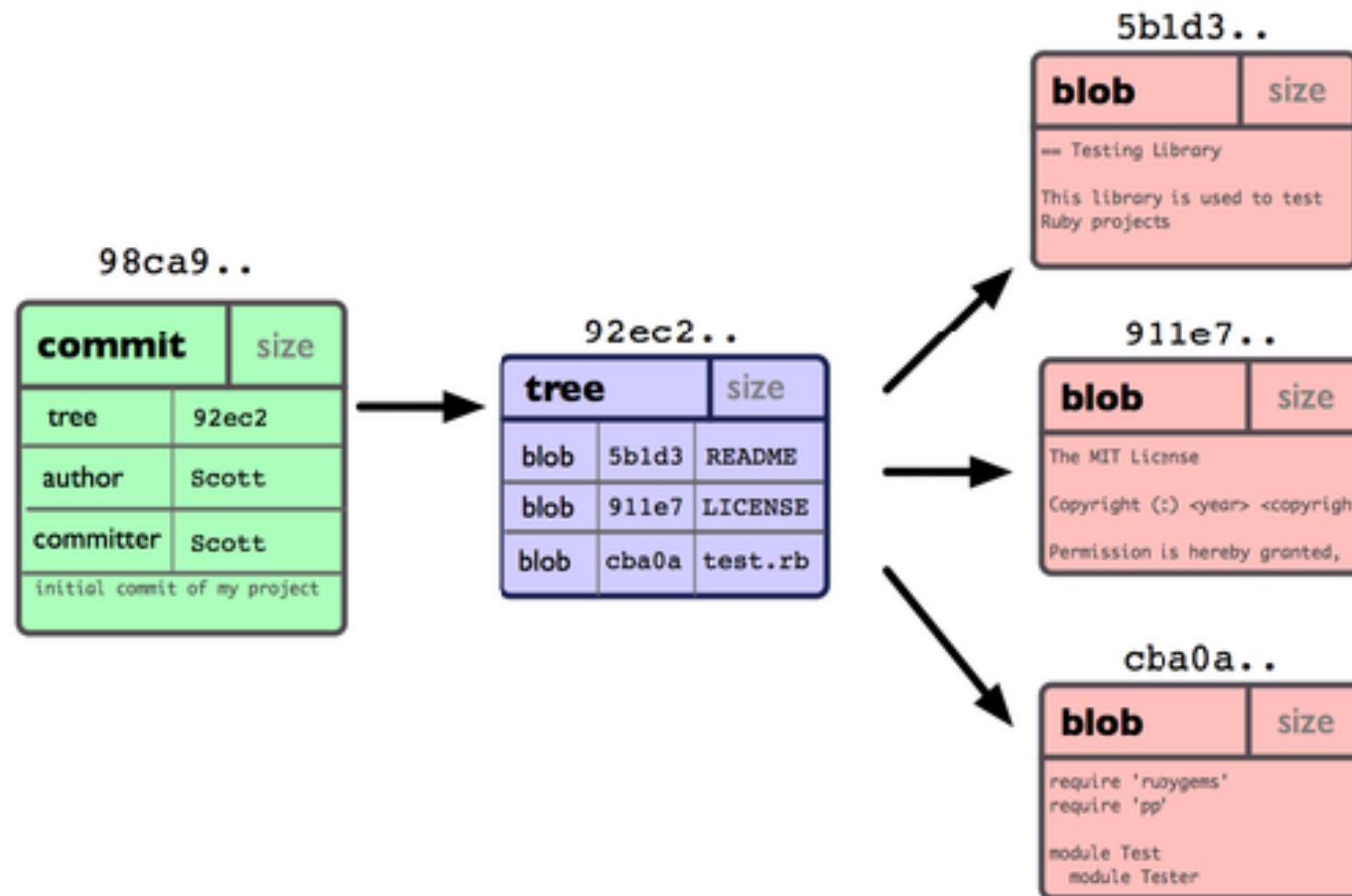
- A “Tree” object represent a Directory
- It is a list of name->Tree and name->Blob (list name->SHA1)
- Tree also have SHA1 id !!



Tree / DAG

- Tree is like an Inode in a filesystem : allow recursion of dir/sub-dir...
- BUT same Trees and same Blobs are immutable and can be shared by pointers in the graph
- DAG = Directed Acyclic Graph
- VERY efficient compression of data, and versionning

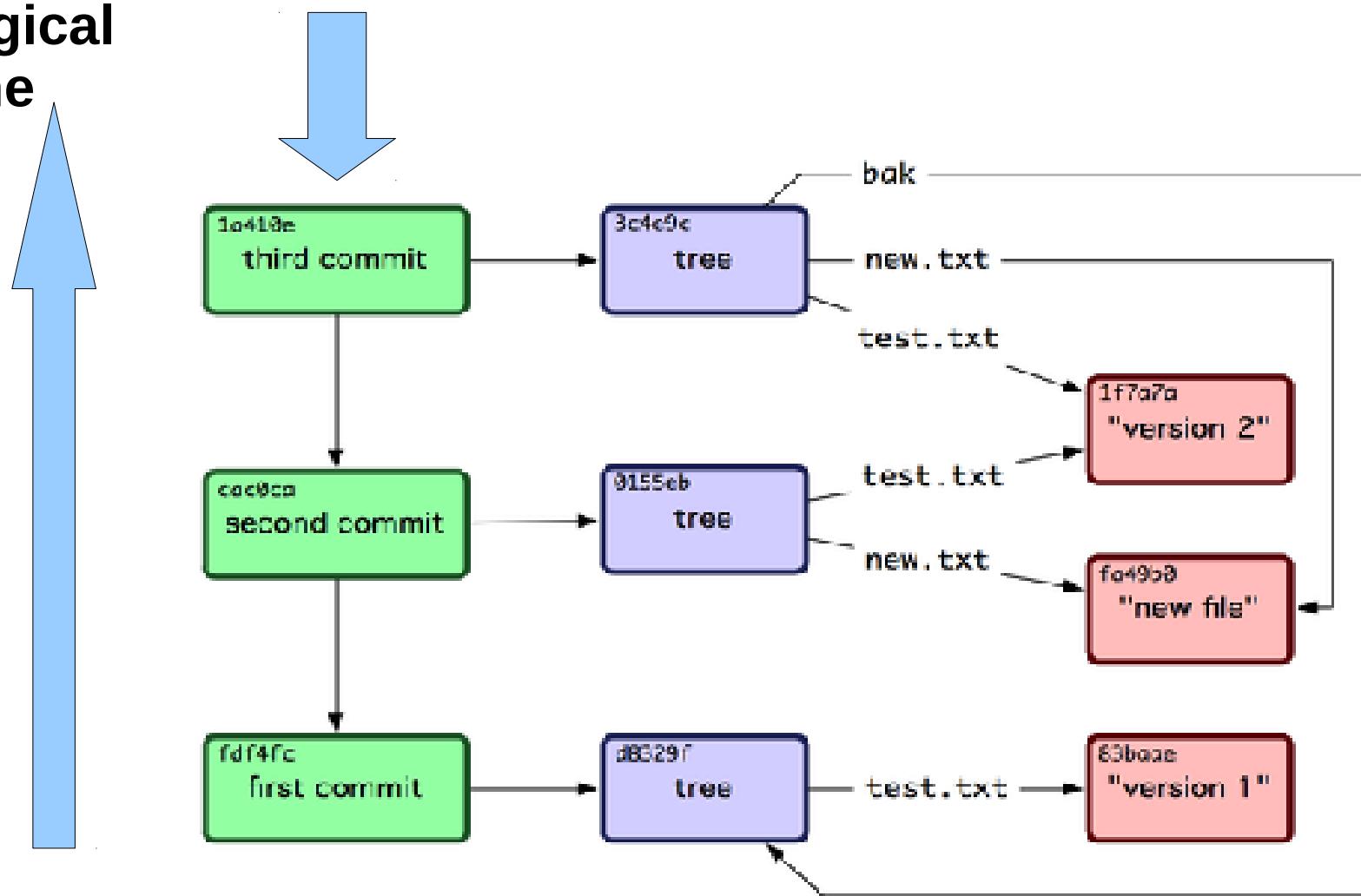
Git Commit → Tree → Blobs



Git Commit History

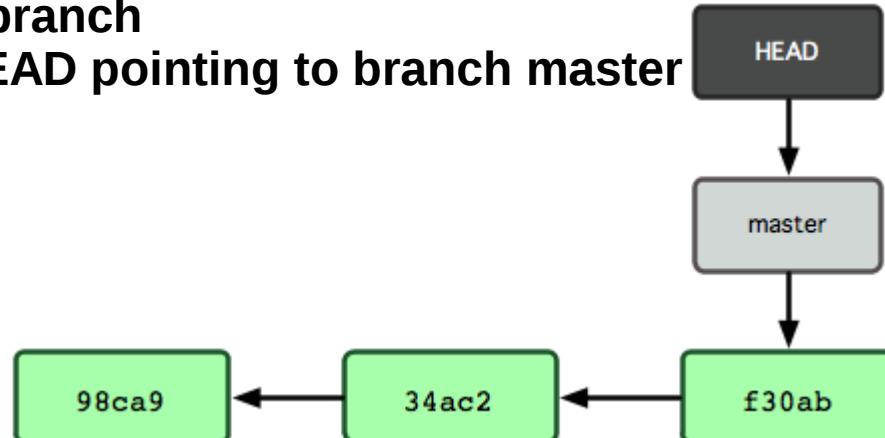
Ref: HEAD (insertion of new commits)

Logical
time

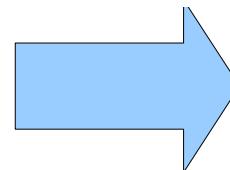


Git Branches ...

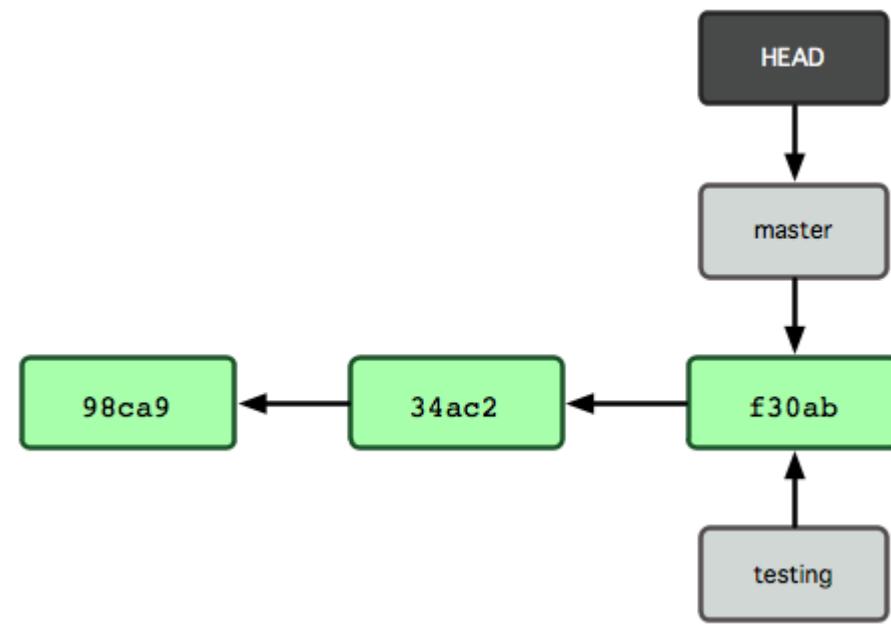
**1 branch
HEAD pointing to branch master**



\$ git branch testing

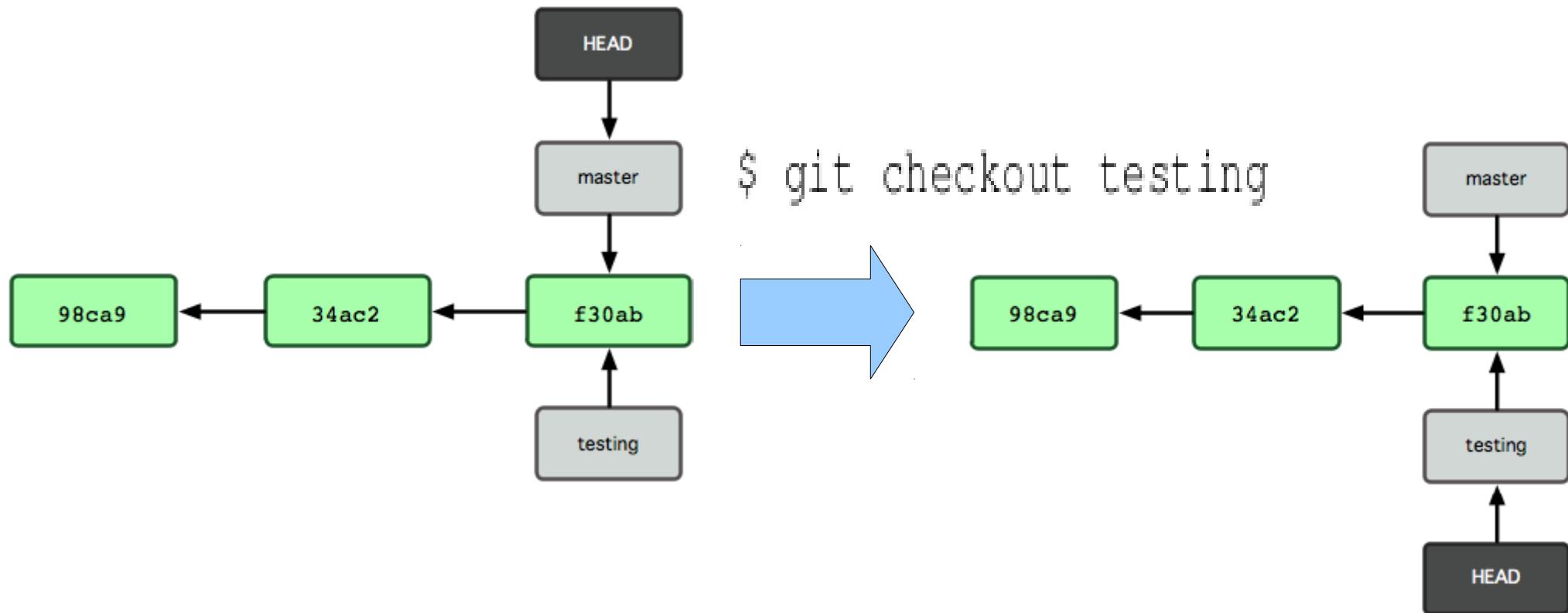


**2 branches pointing to the same commit
(=> same Tree)
HEAD pointing to branch master**



Checkout Branch

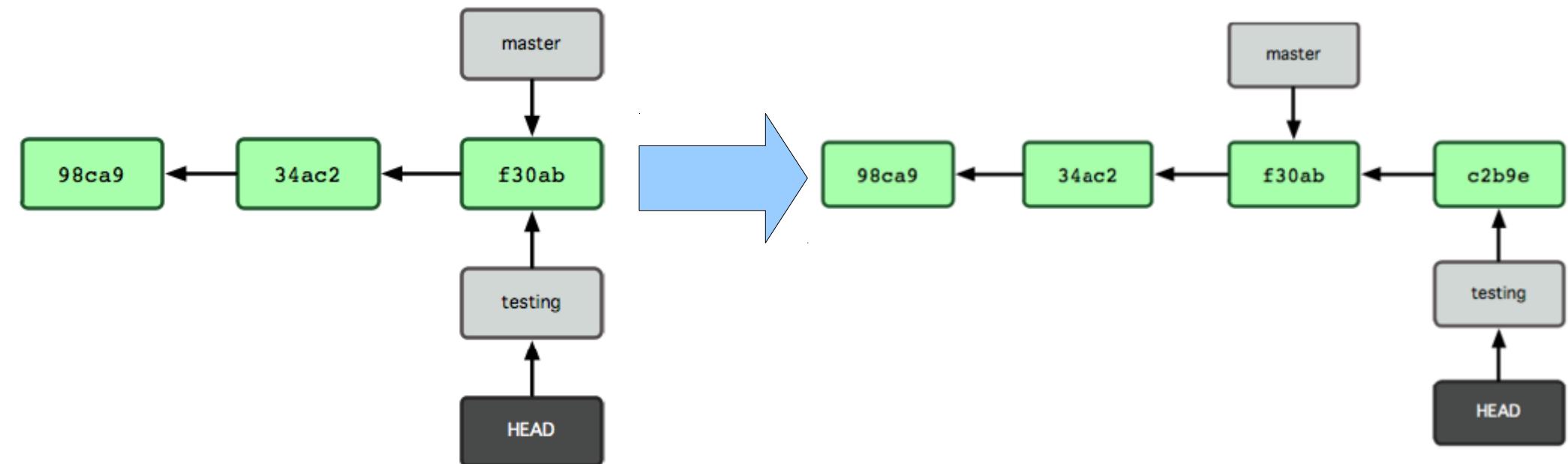
HEAD pointing to branch master



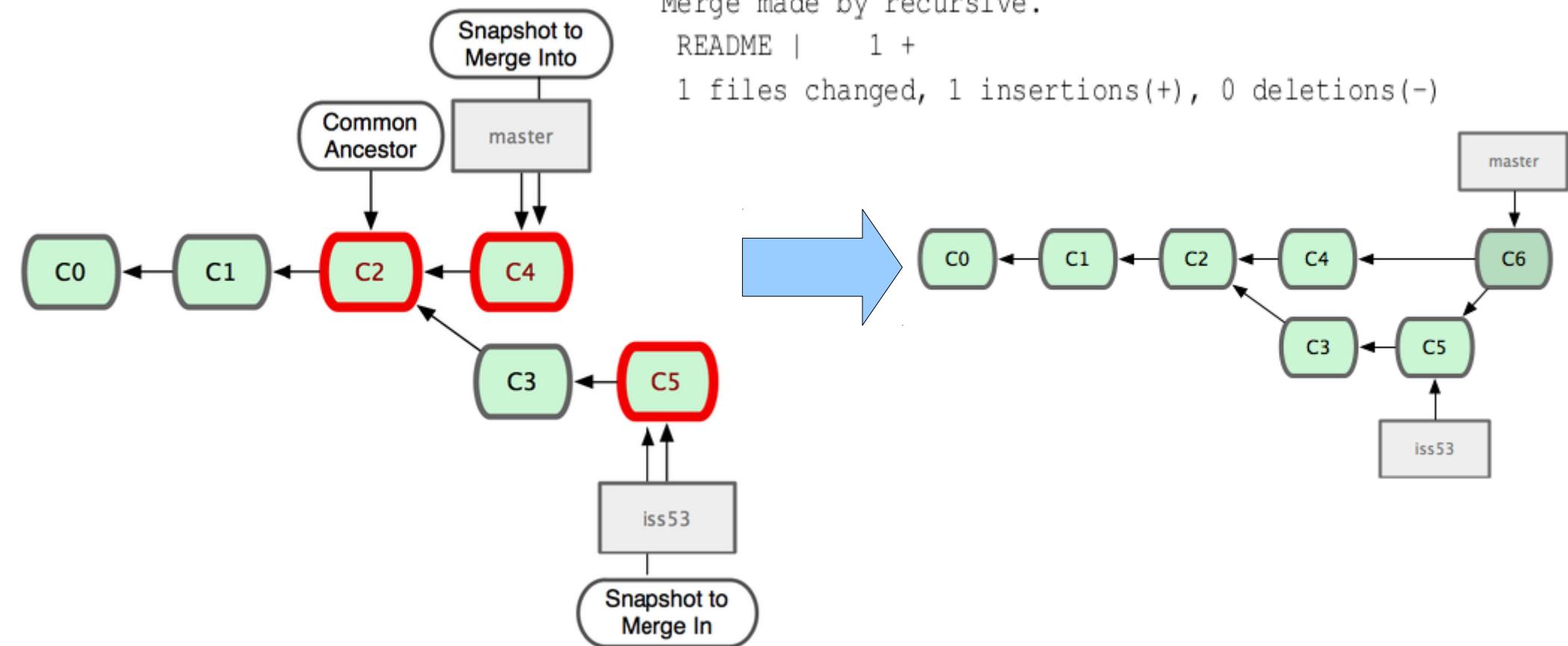
HEAD pointing to branch testing

Git Commit (from HEAD branch)

```
$ git commit -a -m 'made a change'
```



Git Merge



Cheat Sheets...

- Some standards
 - .. to print and keep at your desk
- some others very interesting and surprising:
 - Interactive Workspace/Index/Repo arrows
<http://ndpsoftware.com/git-cheatsheet.html>
 - Interactive moving
<http://onlywei.github.io/explain-git-with-d3/>
 - graphical representation of commands internals
<https://marklodato.github.io/visual-git-guide/index-en.html>

Cheat Sheet from Git Home page

https://na1.salesforce.com/help/doc/en/salesforce_git_developer_cheatsheet.pdf

The screenshot shows the top navigation bar of the Git Cheat Sheet page. A red oval highlights the 'Heroku Cheat Sheet (PDF)' link, which is located next to the 'Visual Git Cheat Sheet (SVG | PNG)' link.

Git Cheat Sheet

Overview
When you first setup Git, set up your user name and email address so your first commits record them properly.
`git config --global user.name "My Name"
git config --global user.email "user@email.com"`

About Git, GitHub and Heroku.
Git is a free & open source, distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

GitHub is the best way to collaborate around your code. Fork, send pull requests and manage all your public and private git repositories.

Heroku is a cloud application platform that supports a number of different programming languages including Java, Ruby, Node.js, and Clojure - it's a new way of building and deploying web apps.

Basic Git Workflow Example

Initialize a new git repository, then stage all the files in the directory and finally commit the initial snapshot.

```
$ git init  
$ git add .  
$ git commit -m 'initial commit'
```

Create a new branch named featureA, then check it out so it is the active branch. then edit and stage some files and finally commit the new snapshot.

```
$ git branch featureA  
$ git checkout featureA
```

Branch & Merge

Working with Git branches and the stash.

<code>git branch</code>	list your branches. a * will appear next to the currently active branch
<code>git branch [branch-name]</code>	create a new branch at the current commit
<code>git checkout [branch]</code>	switch to another branch and check it out into your working directory
<code>git checkout -b [branch]</code>	create a branch and immediately switch to it
<code>git merge [branch]</code>	merge another branch into your currently active one and record the merge as a commit
<code>git log</code>	show commit logs
<code>git stash</code>	stash away the currently uncommitted modifications in your working directory temporarily
<code>git stash apply</code>	re-apply the last stashed changes

Git Cheat Sheet

<http://git.or.cz/>

Remember: `git command --help`

Global Git configuration is stored in `$HOME/.gitconfig` ([git config --help](#))

Create

From existing data

```
cd ~/projects/myproject  
git init  
git add .
```

From existing repo

```
git clone ~/existing/repo ~/new/repo  
git clone git://host.org/project.git  
git clone ssh://you@host.org/proj.git
```

Show

Files changed in working directory
`git status`

Changes to tracked files
`git diff`

What changed between \$ID1 and \$ID2
`git diff $id1 $id2`

History of changes
`git log`

History of changes for file with diffs
`git log -p $file $dir/ec/tory/`

Who changed what and when in a file
`git blame $file`

A commit identified by \$ID
`git show $id`

A specific file from a specific \$ID
`git show $id:$file`

All local branches
`git branch`
(star '*' marks the current branch)

Cheat Sheet Notation

\$id : notation used in this sheet to represent either a commit id, branch or a tag name
\$file : arbitrary file name
\$branch : arbitrary branch name

Concepts

Git Basics

master : default development branch
origin : default upstream repository
HEAD : current branch
HEAD^ : parent of HEAD
HEAD-4 : the great-great grandparent of HEAD

Revert

Return to the last committed state
`git reset --hard`

⚠ you cannot undo a hard reset

Revert the last commit
`git revert HEAD` Creates a new commit

Revert specific commit
`git revert $id` Creates a new commit

Fix the last commit
`git commit -a --amend`
(after editing the broken files)

Checkout the \$id version of a file
`git checkout $id $file`

Branch

Switch to the \$id branch
`git checkout $id`

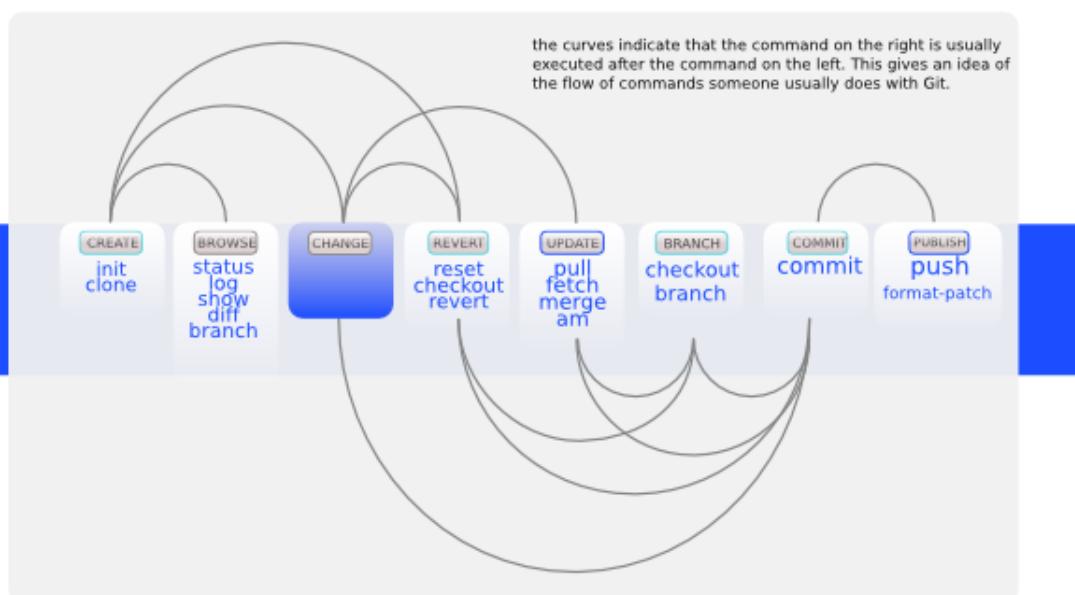
Merge branch1 into branch2
`git checkout $branch2
git merge branch1`

Create branch named \$branch based on the HEAD
`git branch $branch`

Create branch \$new_branch based on branch \$other and switch to it
`git checkout -b $new_branch $other`

Delete branch \$branch
`git branch -d $branch`

Commands Sequence



Update

Fetch latest changes from origin
`git fetch`
(but this does not merge them).

Pull latest changes from origin
`git pull`
(does a fetch followed by a merge)

Apply a patch that some sent you
`git am -3 patch.mbox`
(in case of a conflict, resolve and use
`git am --resolved`)

Publish

Commit all your local changes
`git commit -a`

Prepare a patch for other developers
`git format-patch origin`

Push changes to origin
`git push`

Mark a version / milestone
`git tag v1.0`

Useful Commands

Finding regressions

`git bisect start` (to start)
`git bisect good $id` (\$id is the last working version)
`git bisect bad $id` (\$id is a broken version)

`git bisect bad/good` (to mark it as bad or good)
`git bisect visualize` (to launch gitk and mark it)
`git bisect reset` (once you're done)

Check for errors and cleanup repository

`git fsck`
`git gc --prune`

Search working directory for foo()
`git grep "foo()"`

Resolve Merge Conflicts

To view the merge conflicts
`git diff` (complete conflict diff)
`git diff --base $file` (against base file)
`git diff --ours $file` (against your changes)
`git diff --theirs $file` (against other changes)

To discard conflicting patch

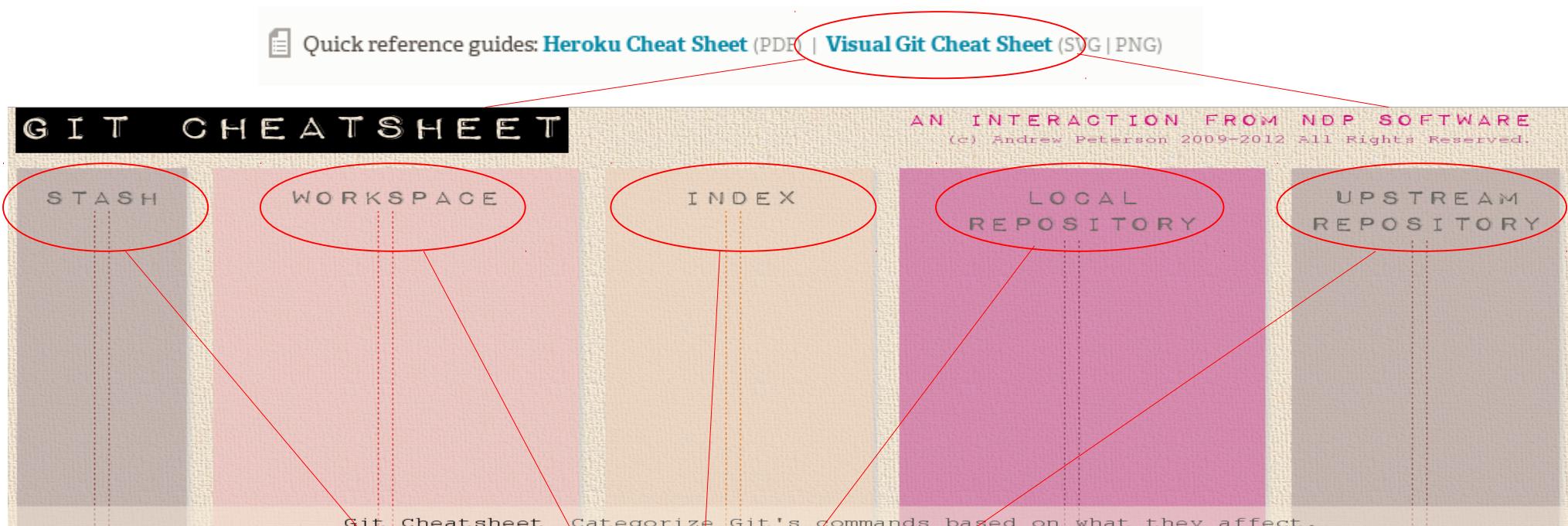
`git reset --hard`
`git rebase --skip`

After resolving conflicts, merge with

`git add $conflicting_file` (do for all resolved files)
`git rebase --continue`

Interactive Graphical Cheat Sheet from Git Home page

<http://ndpsoftware.com/git-cheatsheet.html>
CLICK to SEE



See Next Slides

Cheat Sheet Workspace

WORKSPACE

INDEX

LOCAL
REPOSITORY

UPSTR
REPOSI

status

diff

diff <commit or branch>

add <file... or dir...>

add -u

rm <file...>

mv <file...>

commit -a -m 'msg'

checkout <file...> or
<dir...>

reset --hard

checkout <branch>

checkout -b <name of new branch>

merge <commit or branch>

rebase <upstream>

cherry-pick <sha>

revert <sha>

clone <repo>

pull <remote> <refspec>

reset --hard <remote>/<branch>

clean

Cheat Sheet Index

WORKSPACE

INDEX

LOCAL
REPOSITORY

status

diff

add <file... or dir...>

add -u

rm <file...>

mv <file...>

checkout <file...> or
<dir...>

reset HEAD <file1>
<file2> ...

reset --soft HEAD^

diff --cached [<commit>]

commit -m 'msg'

commit --amend

Cheat Sheet Local Repo

WORKSPACE	INDEX	LOCAL REPOSITORY	UPSTREAM
diff <commit or branch>			
commit -a -m 'msg'		reset --soft HEAD^	
reset --hard			
checkout <branch>			
checkout -b <name of new branch>			
merge <commit or branch>			
rebase <upstream>			
cherry-pick <sha>			
revert <sha>			
	diff --cached [<commit>]		
	commit -m 'msg'		
	commit --amend		
		log	
		diff <commit> <commit>	
		branch	
		branch -d <branch>	
		branch --track <new> <remote/branch>	
		fetch <remote> <refspec>	
		push	
		push <remote> <branch>	
		push <remote> <branch>:<branch>	

Cheat Sheet Upstream Repo

<SPACE	INDEX	LOCAL REPOSITORY	UPSTREAM REPOSITORY
			branch --track <new> <remote/branch>
clone <repo>			
pull <remote> <refspec>			
reset --hard <remote>/<branch>			
		fetch <remote> <refspec>	
		push	
		push <remote> <branch>	
		push <remote> <branch>:<branch>	
			branch -r
			push <remote> :<branch>

Cheat Sheet Stash

GIT CHEATSHEET

AN INTERACTION FROM NDP SOFTWARE

(c) Andrew Peterson 2009-2012 All Rights Reserved.

STASH

WORKSPACE

INDEX

LOCAL REPOSITORY

UPSTREAM REPOSITORY

```
stash save [<msg>]
```

```
stash apply [<name>]
```

```
stash pop
```

```
stash list
```

```
stash show [<stash>]
```

```
stash drop [<name>]
```

```
stash clear
```

```
stash branch <branchname> [<stash>]
```

Github explain-git-with-d3

onlywei.github.io/explain-git-with-d3/#rebase

Visualizing Git Concepts with D3

This website is designed to help you understand some basic git concepts visually. This is my first attempt at using both SVG and D3. I hope it is helpful to you.

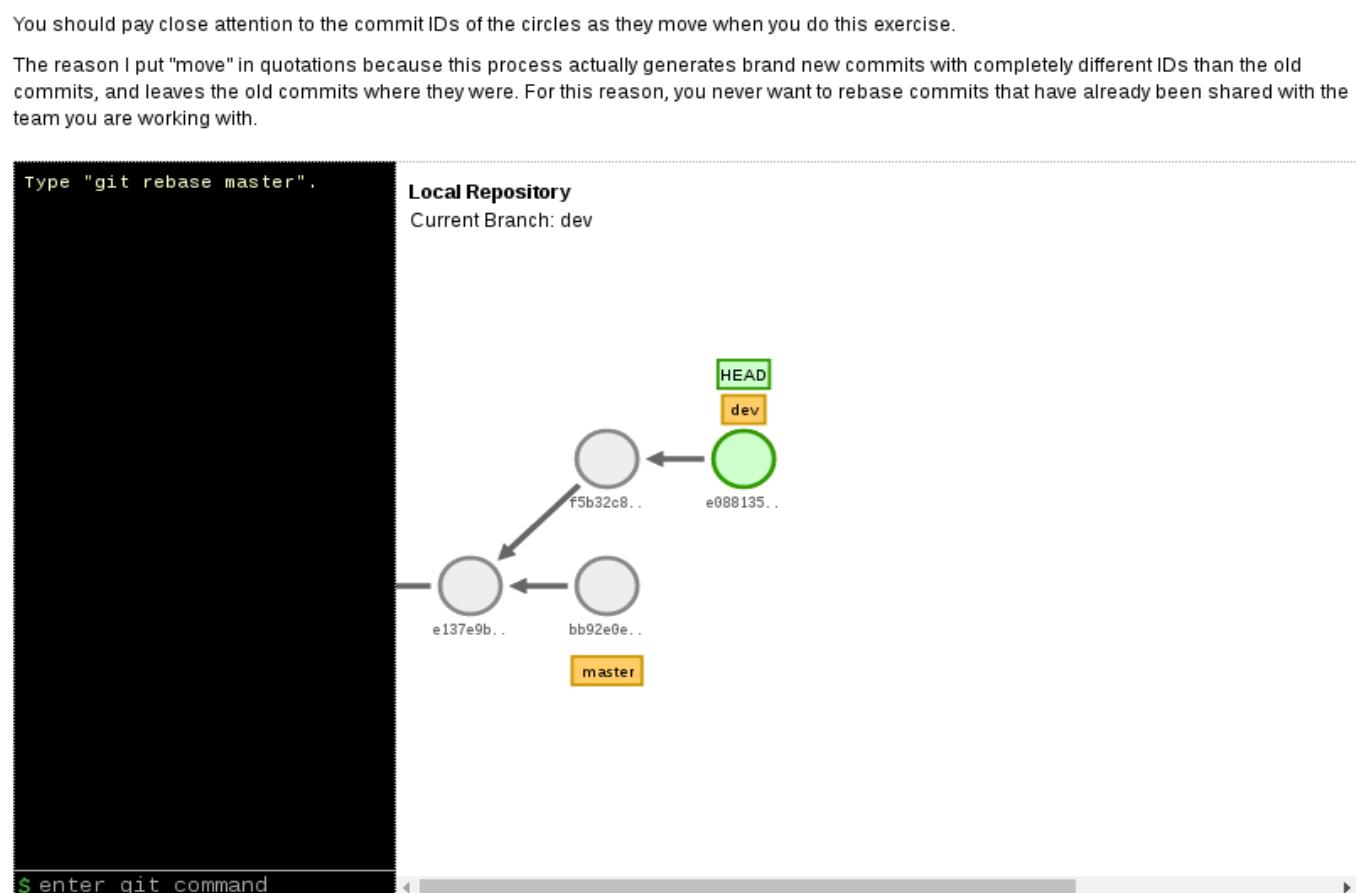
Adding/staging your files for commit will not be covered by this site. In all sandbox playgrounds on this site, just pretend that you always have files staged and ready to commit at all times. If you need a refresher on how to add or stage files for commit, please read [Git Basics](#).

Sandboxes are split by specific git commands, listed below.

Basic Commands	Undo Commits	Combine Branches	Remote Server
git commit	git reset	git merge	git fetch
git branch	git revert	git rebase	git pull
			git tag

<http://onlywei.github.io/explain-git-with-d3/>

Type command
interactive
&
See interactive
drawing
moving!!



Visual Git Guide

<https://marklodato.github.io/visual-git-guide/index-en.html>



Other Languages: [Deutsch](#) [Español](#) [Français](#) [Italiano](#) [日本語](#) [中文](#) [Português](#) [Русский](#) [Slovenčina](#) [Tiếng Việt](#) [简体中文](#) [正體中文](#)

A Visual Git Reference

If the images do not work, you can try the [Non-SVG](#) version of this page.

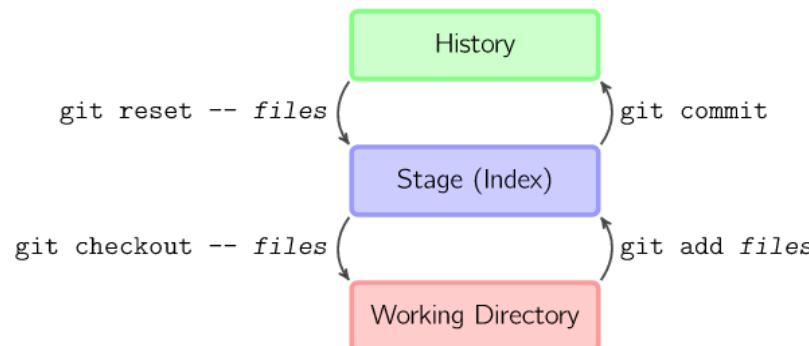
This page gives brief, visual reference for the most common commands in git. Once you know a bit about how git works, this site may solidify your understanding. If you're interested in how this site was created, see my [GitHub repository](#).

Also recommended: [Visualizing Git Concepts with D3](#)

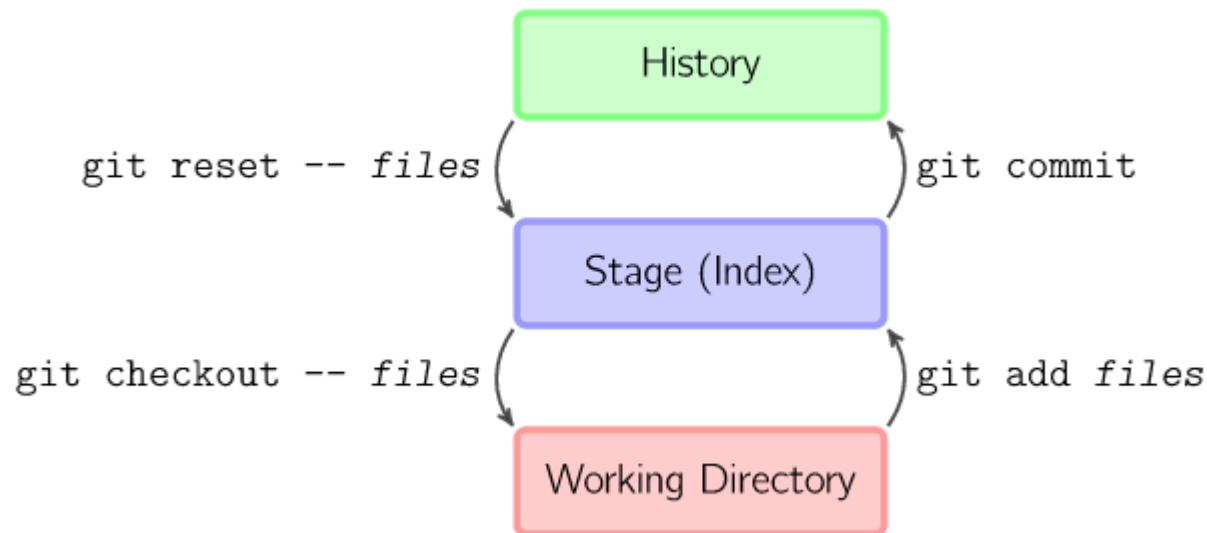
Contents

1. [Basic Usage](#)
2. [Conventions](#)
3. [Commands in Detail](#)
 - a. [Diff](#)
 - b. [Commit](#)
 - c. [Checkout](#)
 - d. [Committing with a Detached HEAD](#)
 - e. [Reset](#)
 - f. [Merge](#)
 - g. [Cherry Pick](#)
 - h. [Rebase](#)
4. [Technical Notes](#)

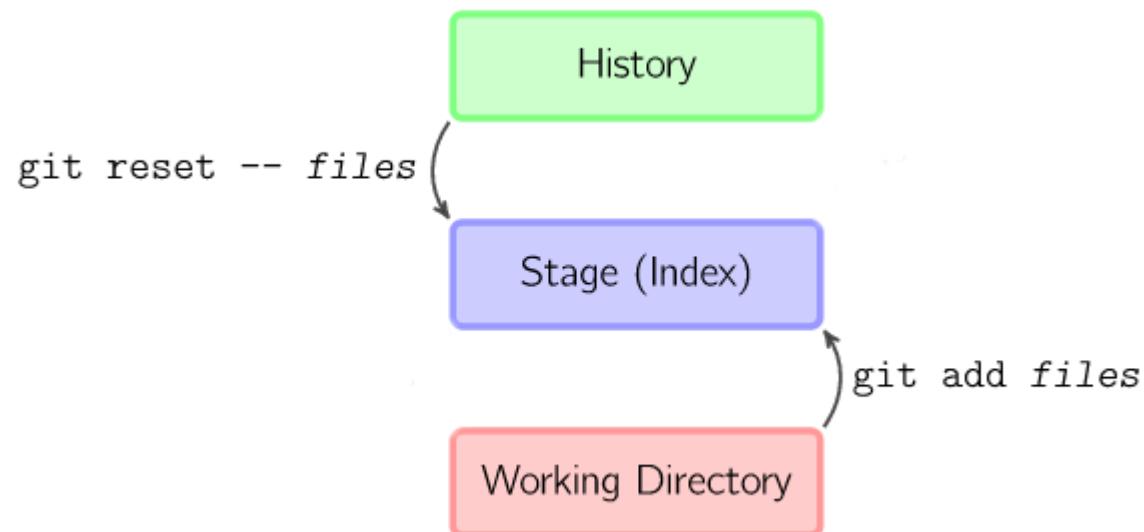
Basic Usage



Visual Git Guide : Basic Commands



Opposite of “git add” ? “git reset -- ...” DON’T “git rm”

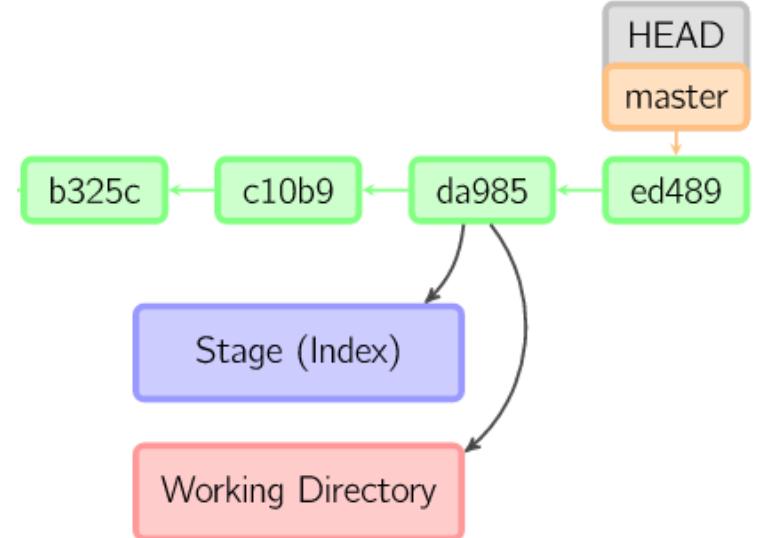
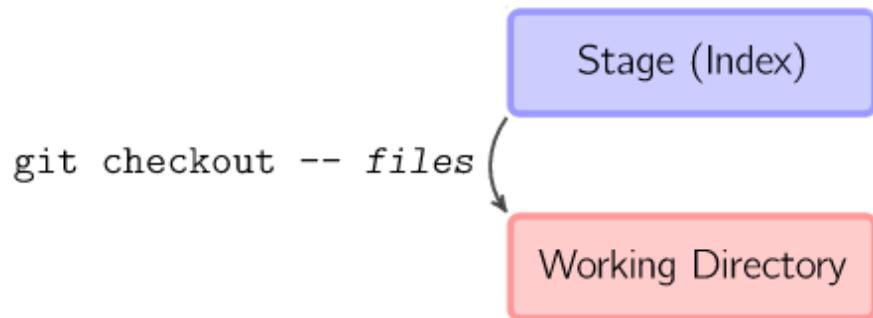


Git Checkout -- file

Usefull to **undo local file modifications**
= restore
idem restore from HEAD.. assuming index~HEAD

Usefull to **restore file content from history** = as in past commits

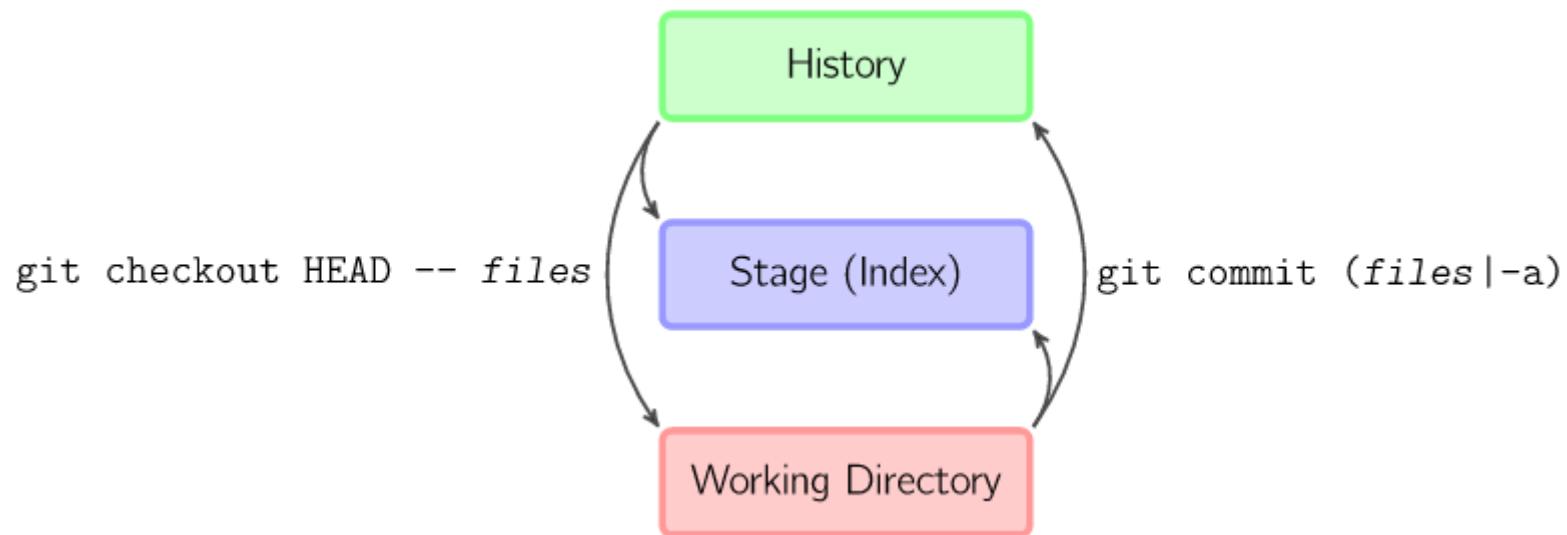
`git checkout HEAD~ files`



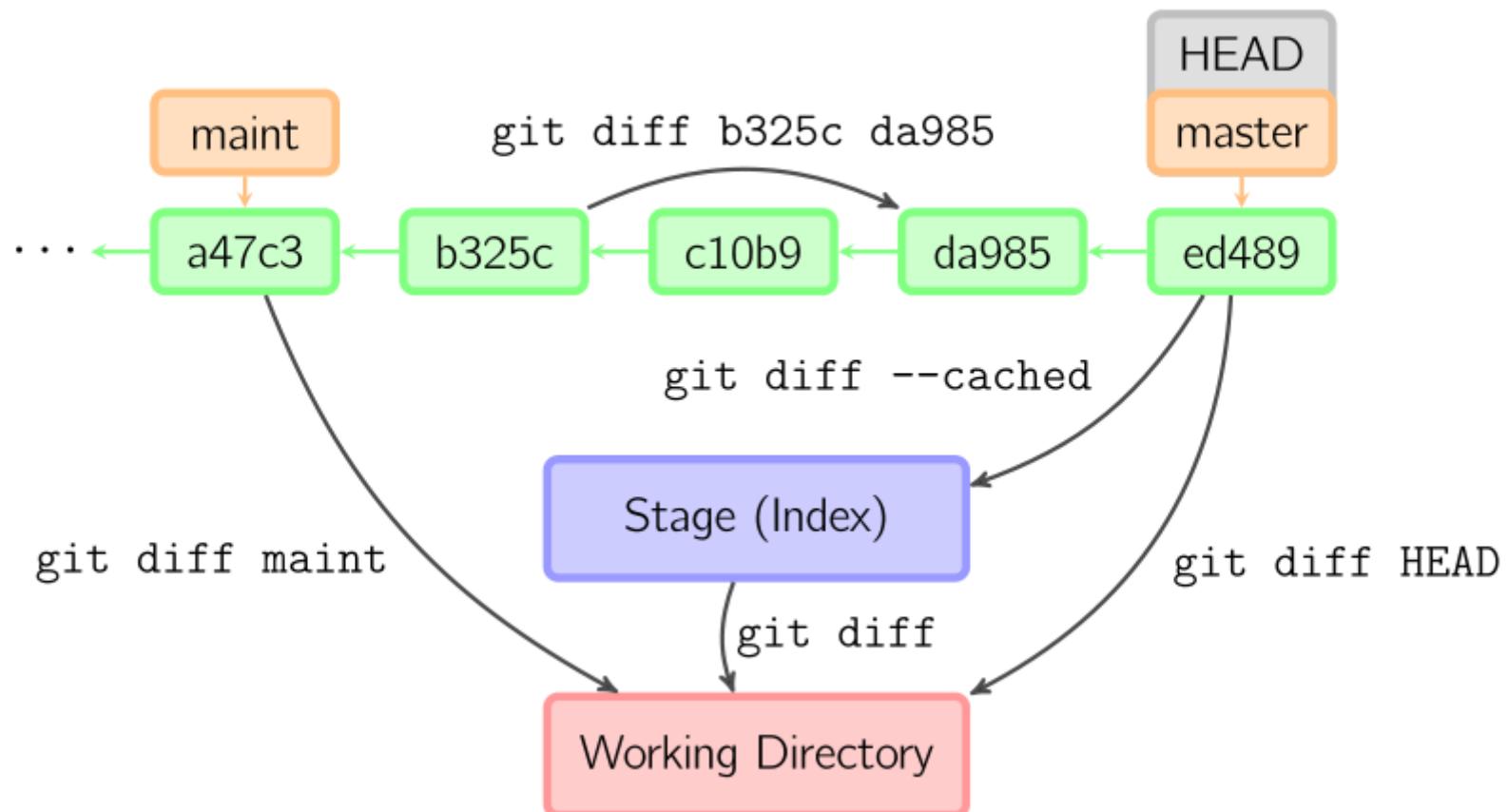
Remarks : use git argument escape “- -“ ... image you have a file named dash “-rm”

Basic Command ..

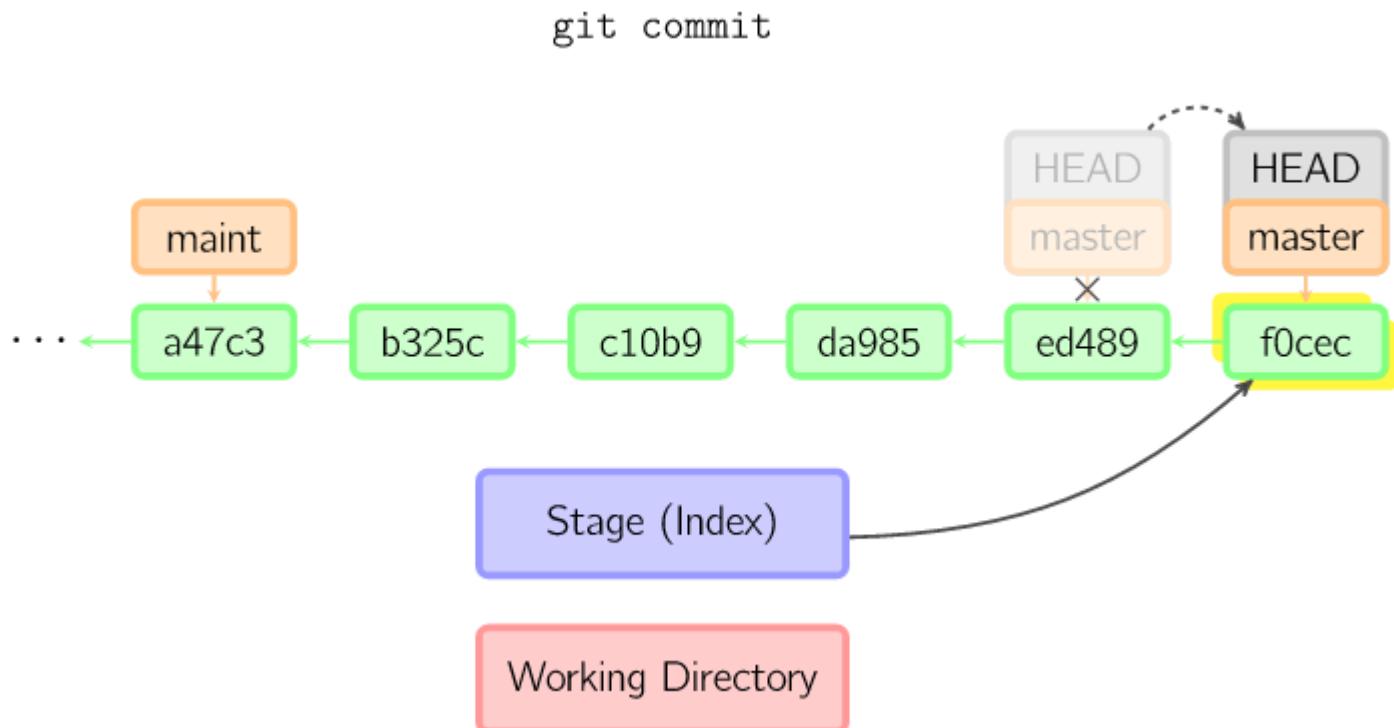
Shortcut Repo<->Workspace



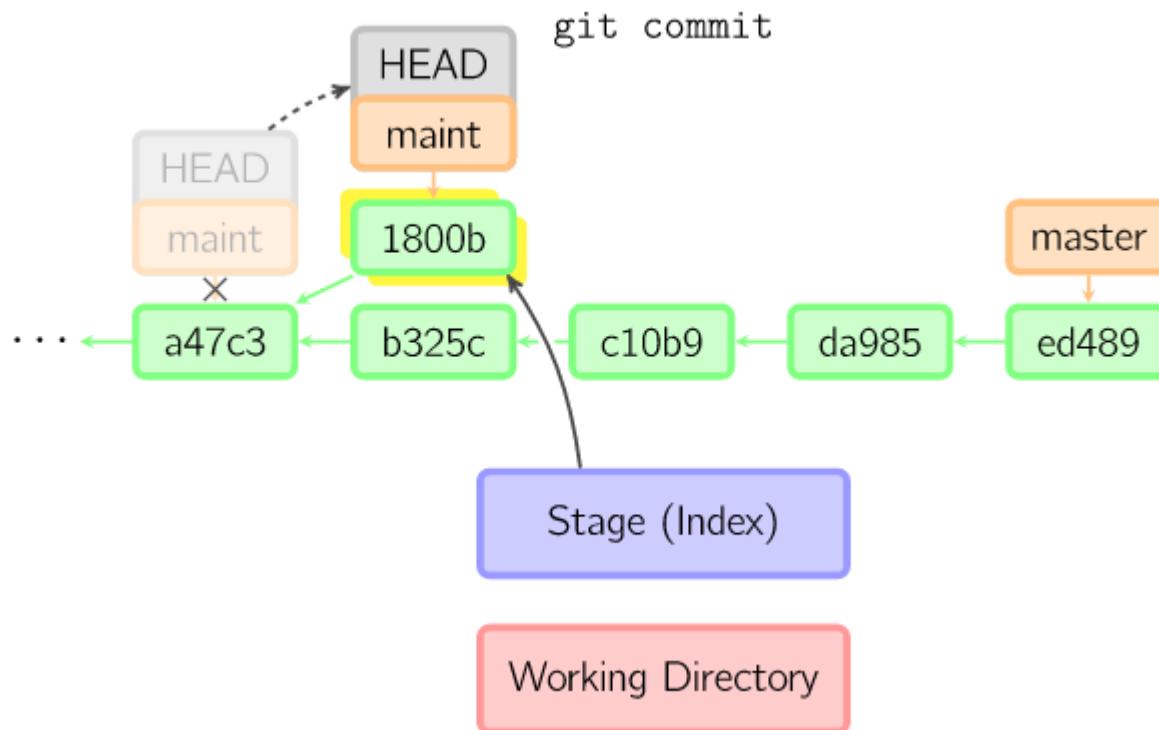
Git Diff



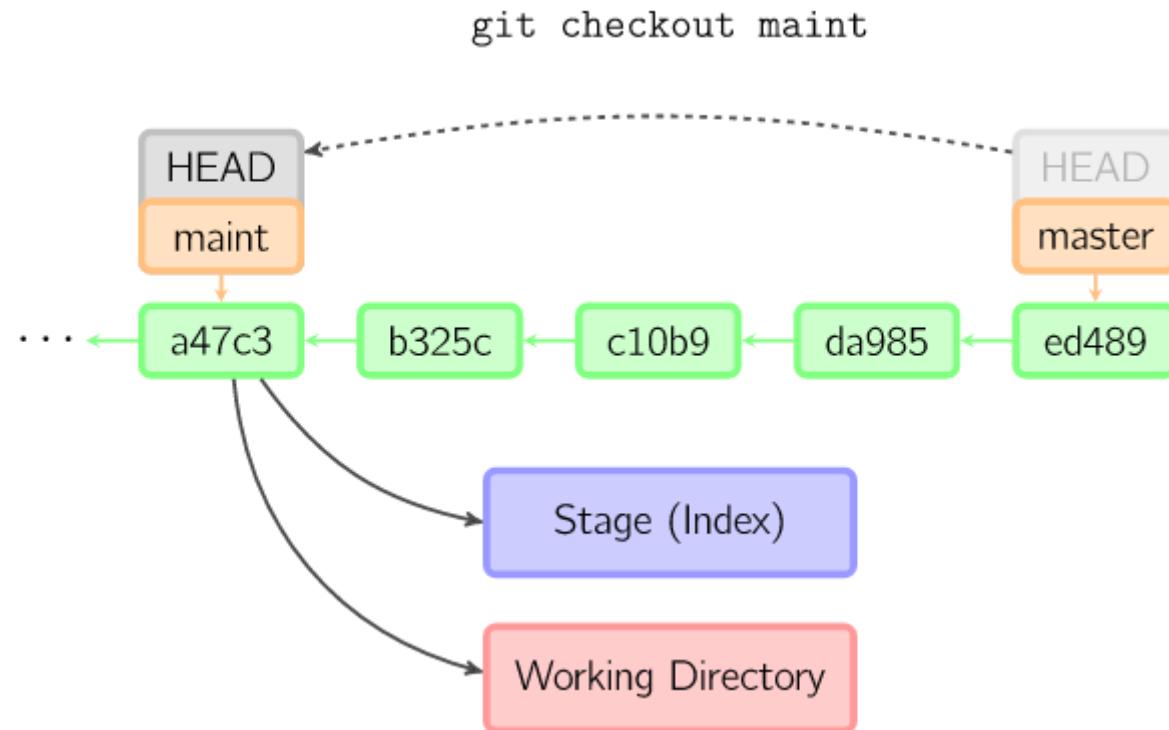
Git Commit (where HEAD=master)



Git Commit where HEAD = branch

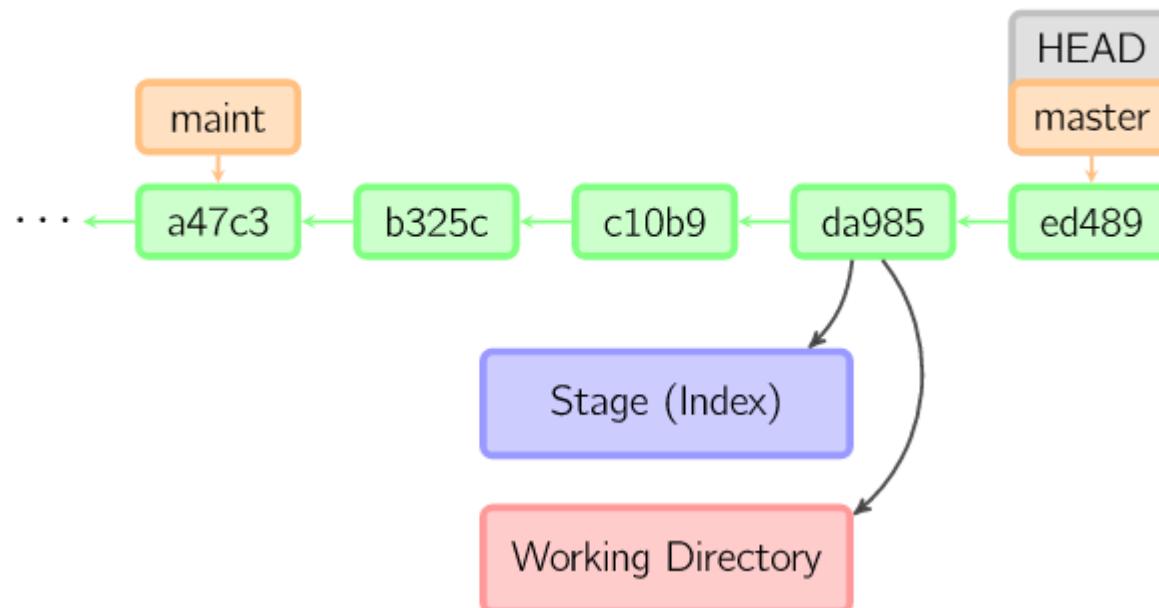


Checkout (1/3): git co branch (=change current branch)

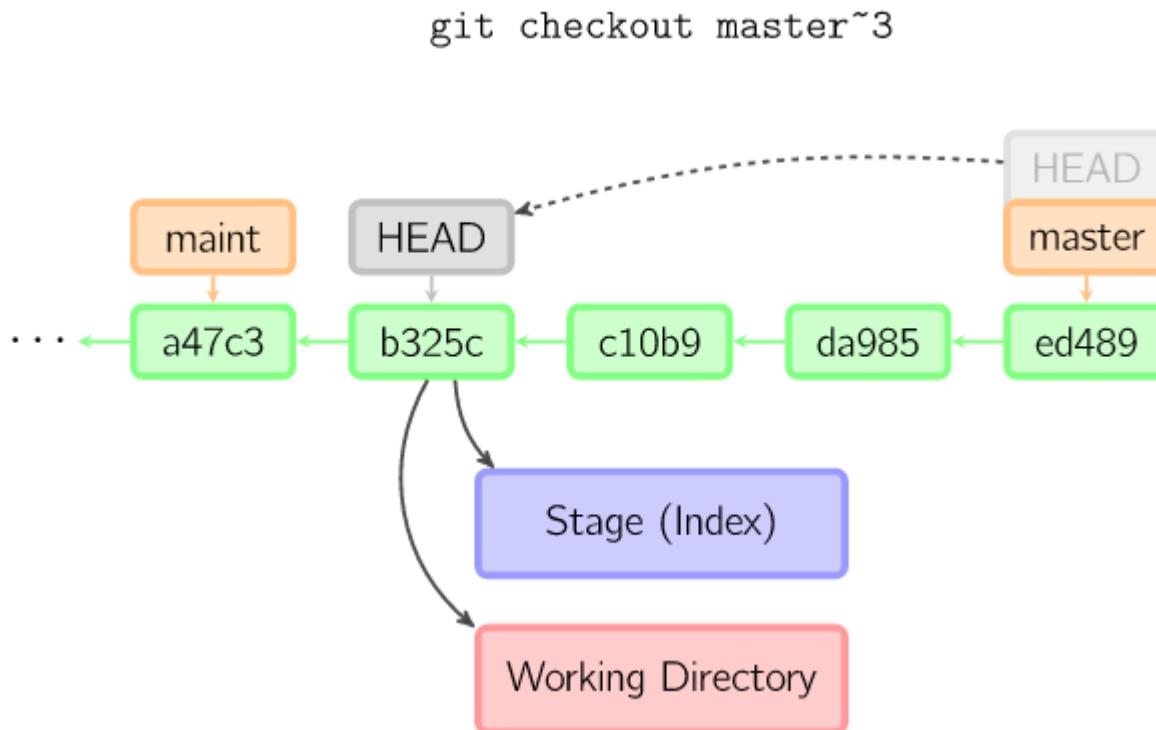


Checkout (2/3) : git co -- file1 file2 = restore a file

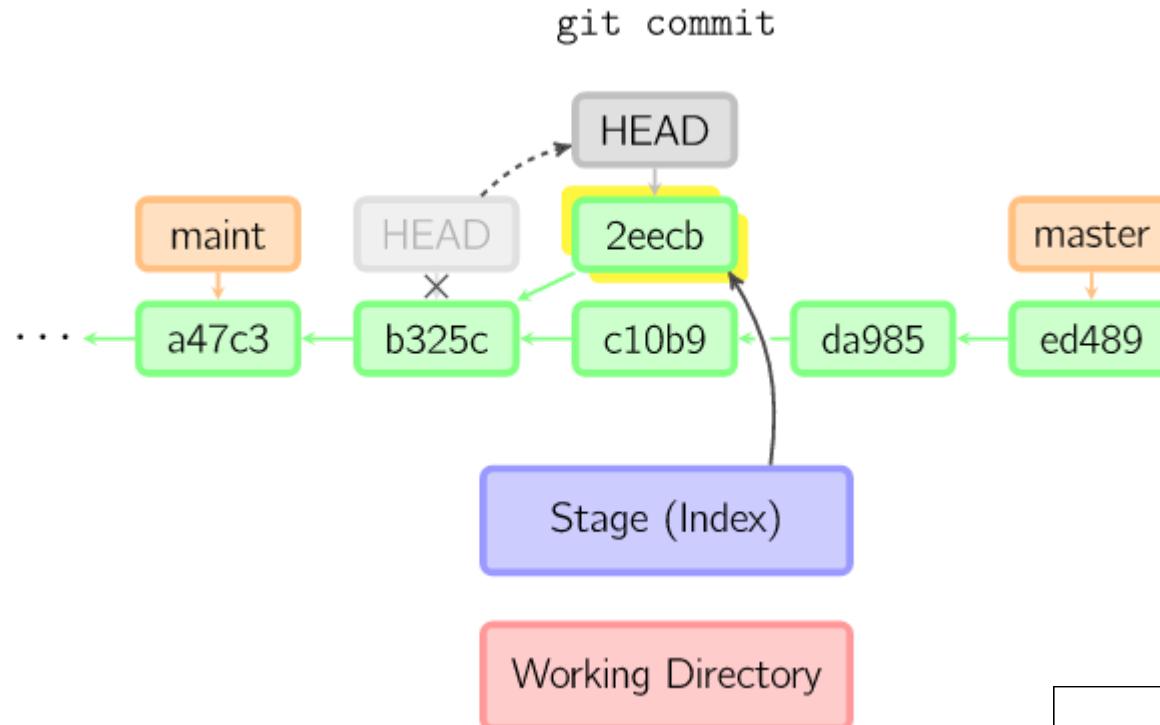
git checkout HEAD~ files



Checkout (3/3) : git co head~ 3 “detached HEAD” (=anonymous branch)!

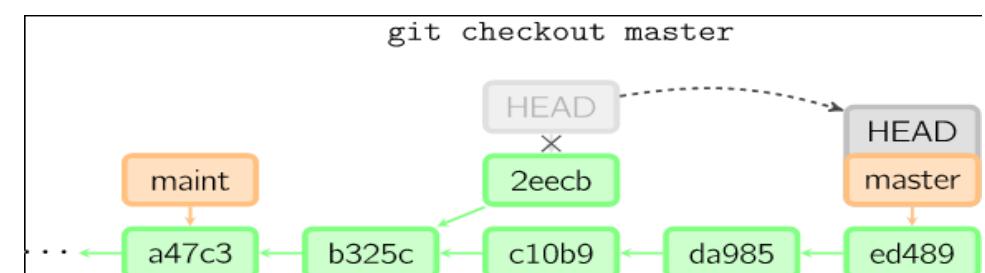


Detached HEAD ??



RISK... If change to branch or master..
\$ git co master

Commit is still POSSIBLE
in “anonymous” branch ...
(anon. = no name reference)

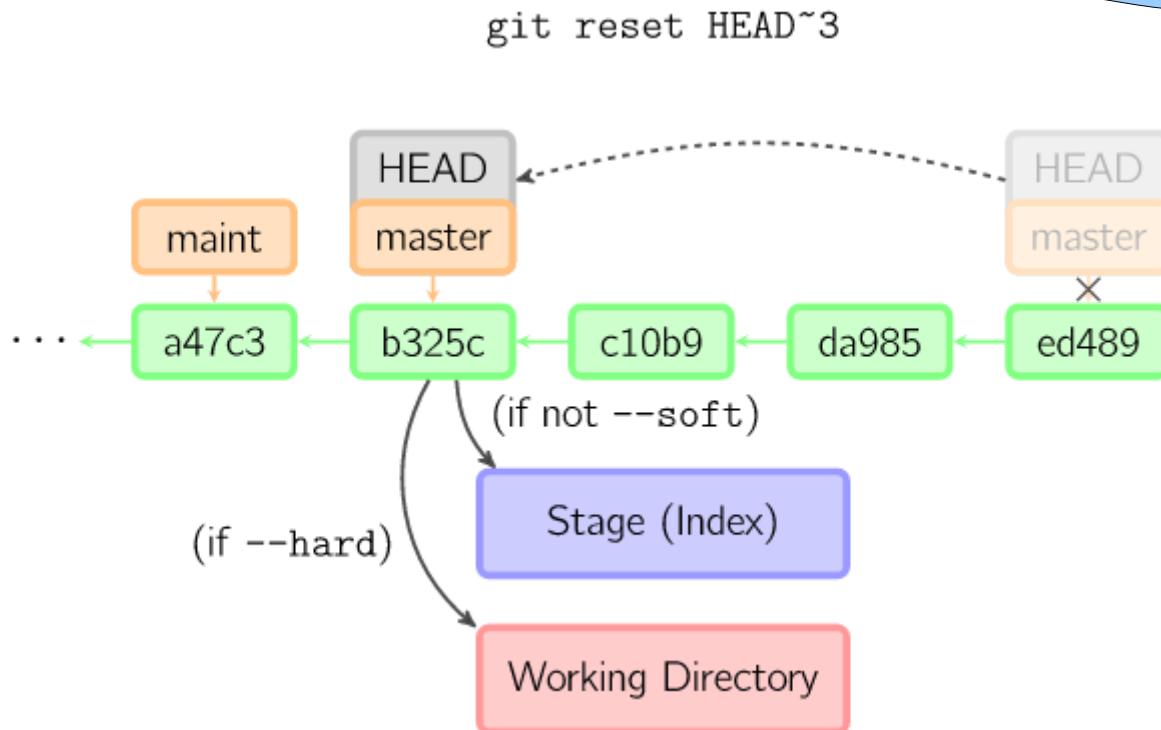


... LOSE DATA “detached HEAD” !!
commit 2eecb.. no more referenced!
(only in .git/reflog)

Git Reset ~~~

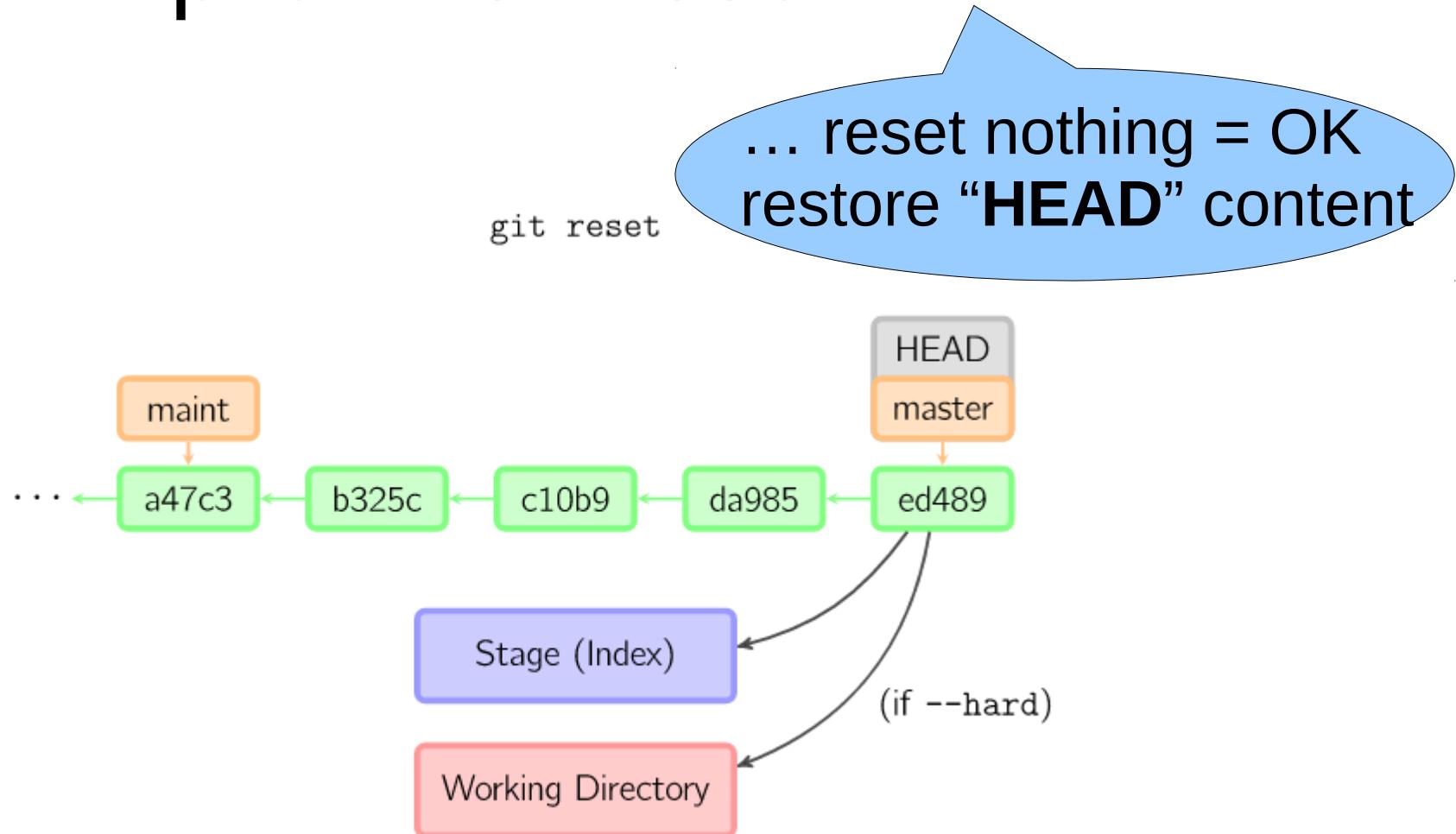
.... another way to
“Shoot yourself in the FOOT”

... Like Detached
“HEAD” ?



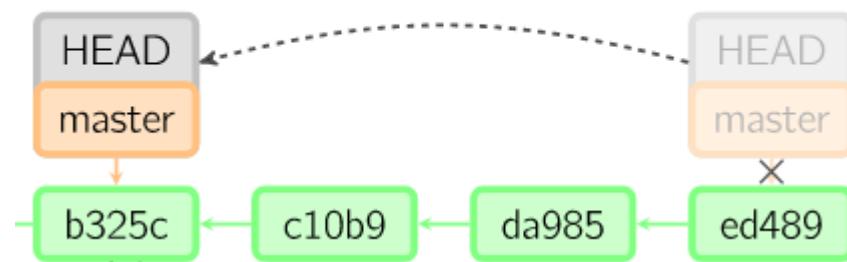
Git Reset

= implicit : Git reset HEAD



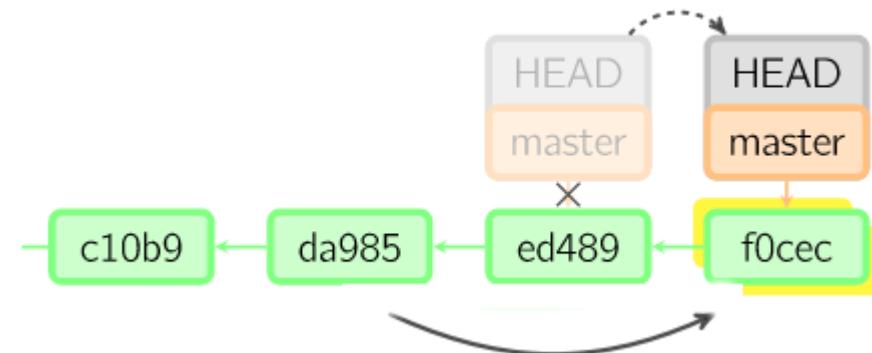
reset head~ ... vs revert head~

git reset HEAD~3



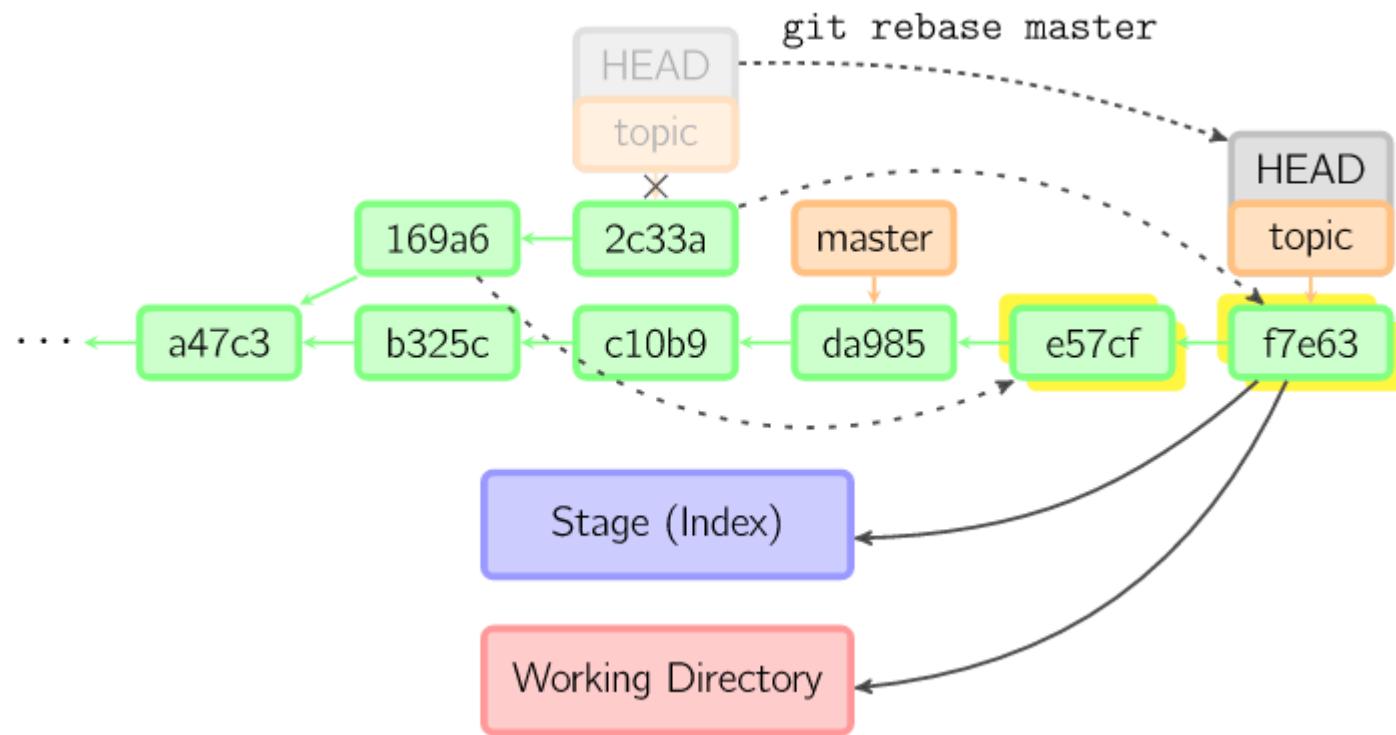
Reset = undo content & undo commits

git revert HEAD~



Revert = undo content
BY APPLYING inverse of delta
then commit forward
(no brain wash history, only overwrite to fix it)

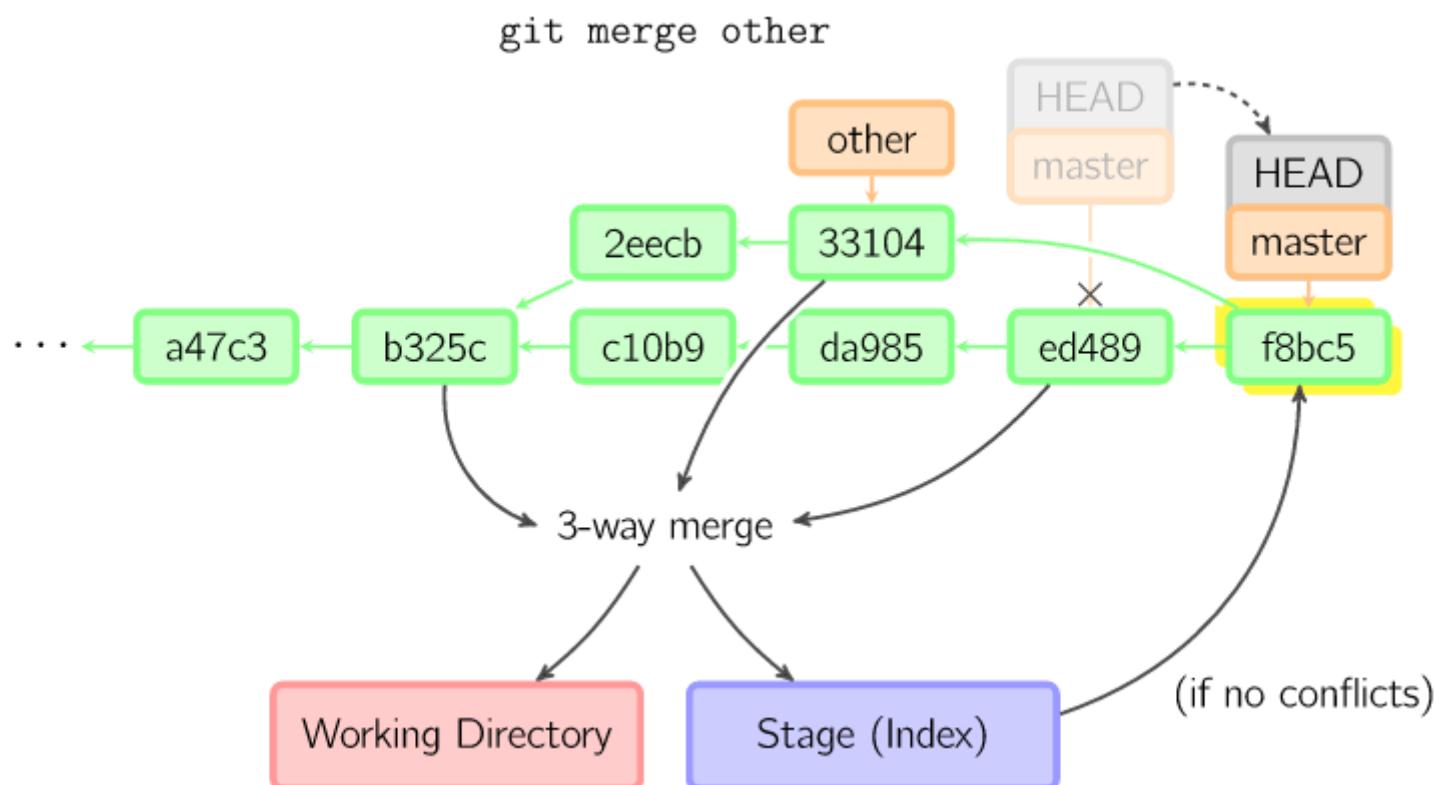
Rebase



Remarks:

- **equivalent to N times “cherry pick”**
- Commit 169a6 → e57cf applied diff to master (same diff, SHA1 & content differs)
Commit 2c33a → f7e63 applied diff to master (same diff, SHA1 & content differs)
- **LINEAR history .. simpler than merge** (graph), but same content

merge



Git Tools

- Git Command line is your first friend
- In Eclipse : Egit, in File Explorer: Tortoise..
- Github : direct Windows/Linux client
commit + push in once
- Graphical Branch representations
... many tools

Egit : Git in Eclipse

Share Project

Select the repository plug-in that will be used to share the selected project.

Select a repository type:

CVS
Git
SVN

Configure Git Repository

Select Git repository location

Use or create repository in parent folder of project

Project	Path	Repository
testroo	/home/arnaud/test	.git

< Back Next >

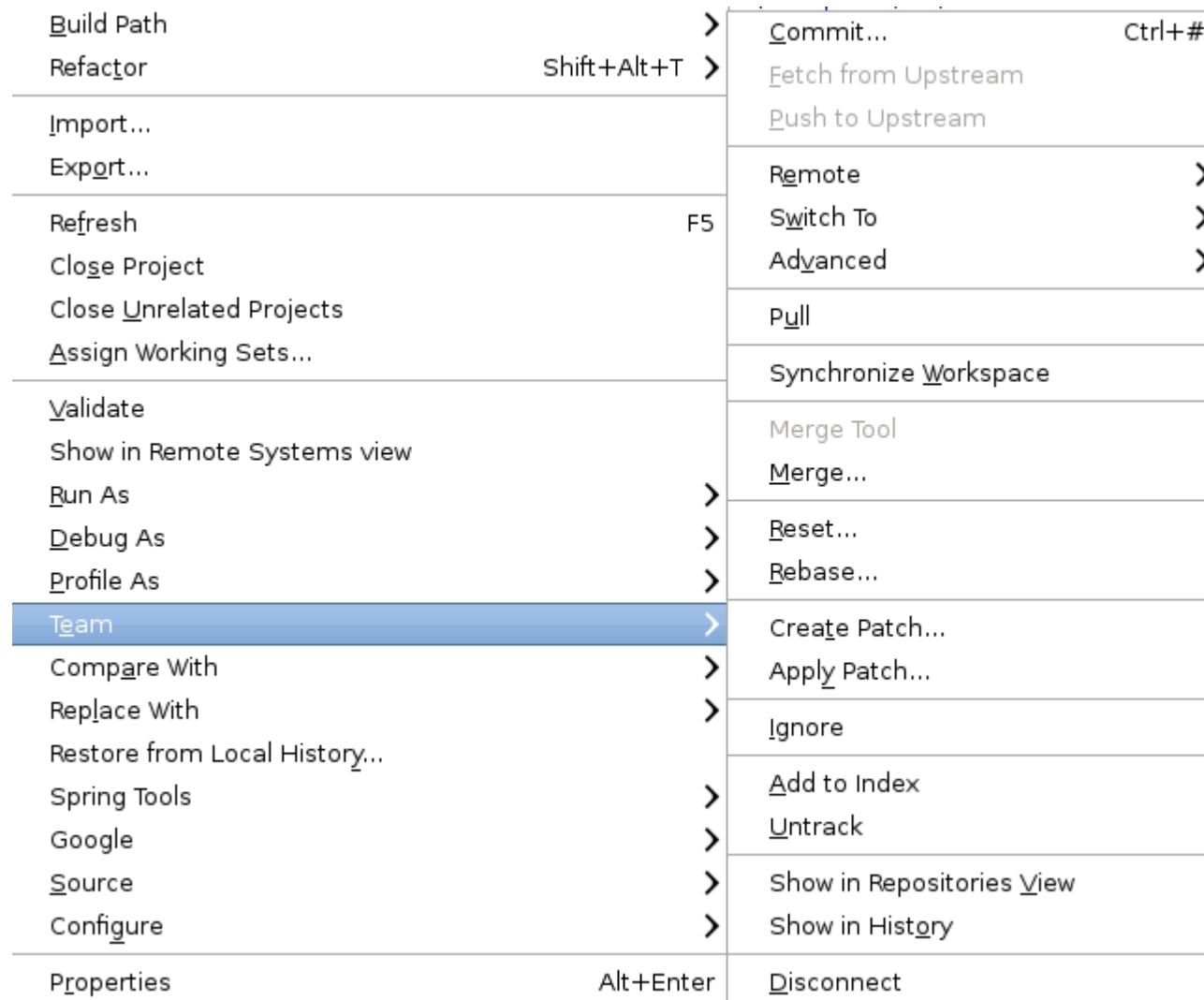
Create Repository

test-git [test master]

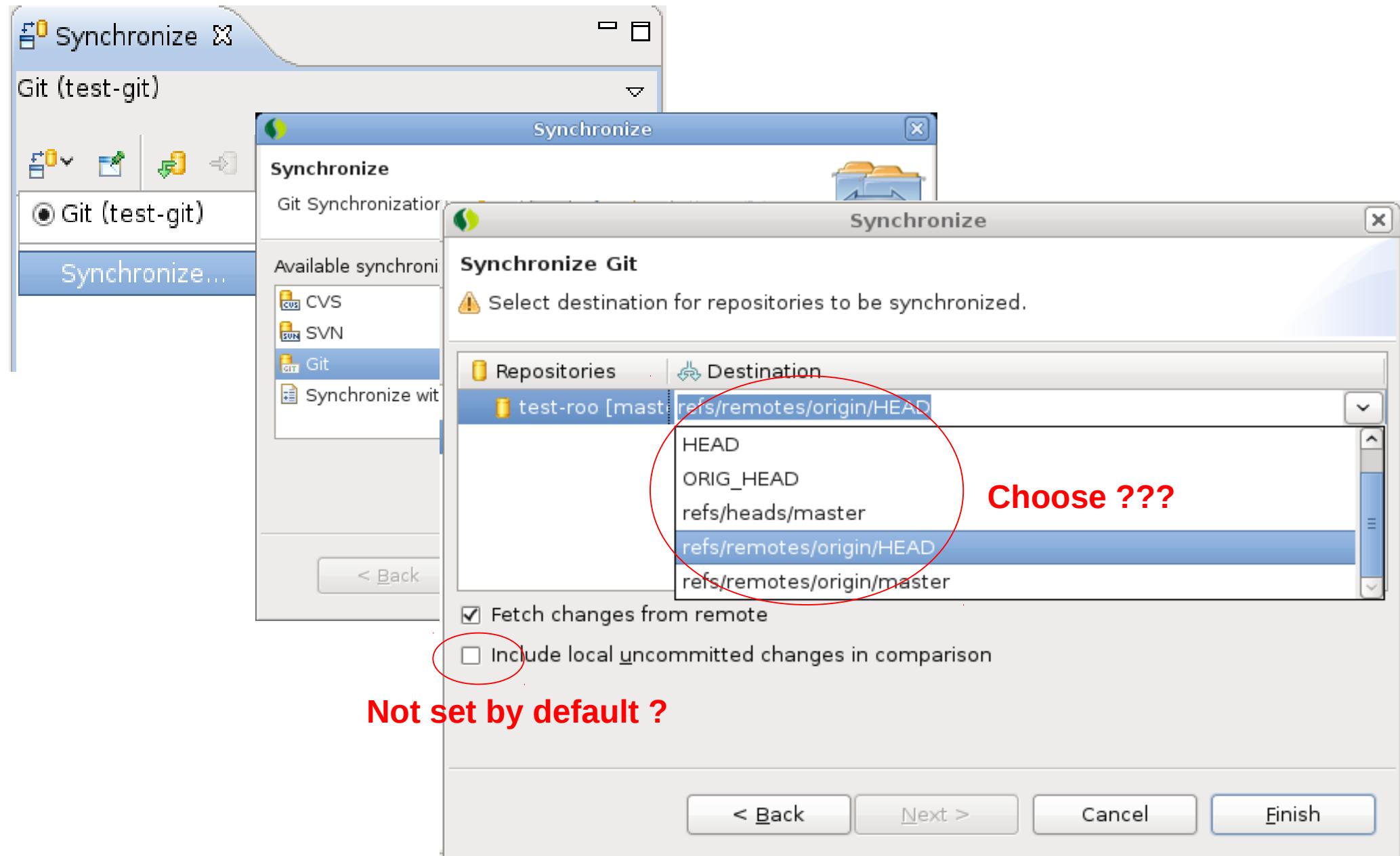
README.txt

The image shows two overlapping Eclipse dialog boxes. The top dialog is titled 'Share Project' and contains a message to select a repository plug-in. The bottom dialog is titled 'Configure Git Repository' and shows the configuration for a Git repository. It includes a table for selecting a repository location, with a row for 'testroo' at path '/home/arnaud/test' and repository '.git'. A checkbox for 'Use or create repository in parent folder of project' is checked. At the bottom of the dialog are 'Create Repository' and 'Next >' buttons. Below the dialogs, a status bar displays 'test-git [test master]' and a file icon for 'README.txt'.

Eclipse Team Menu / Perspective / History View ...



Eclipse eGit Synchronize View



eGit Commit

The screenshot shows the eGit interface within a Java IDE. On the left, a file tree displays a repository named 'testroo' with three files: 'file-modified.txt', 'file-to-add.txt', and 'file-to-remove.txt'. A context menu is open over 'file-modified.txt', with the 'Commit...' option highlighted. To the right, a 'Commit Changes' dialog is open, containing fields for the commit message ('test eGit commit'), author ('Arnaud Nauwynck <arnaud.nauwynck@gmail.com>'), committer ('Arnaud Nauwynck <arnaud.nauwynck@gmail.com>'), and a list of files ('file-to-add.txt', 'file-modified.txt', 'file-to-remove.txt').

File tree:

- testroo [test-roo master ↑ 2]
 - file-modified.txt
 - file-to-add.txt
 - file-to-remove.txt

Context menu (Commit... selected):

- Commit... Ctrl+Shift+C
- Add to Git Index
- Ignore
- Merge Tool
- Show In >
- Copy Ctrl+C
- Paste Patch Ctrl+V
- Delete Delete

Commit Changes dialog:

Commit Changes to Git Repository

Commit message:

test eGit commit

Author: Arnaud Nauwynck <arnaud.nauwynck@gmail.com>

Committer: Arnaud Nauwynck <arnaud.nauwynck@gmail.com>

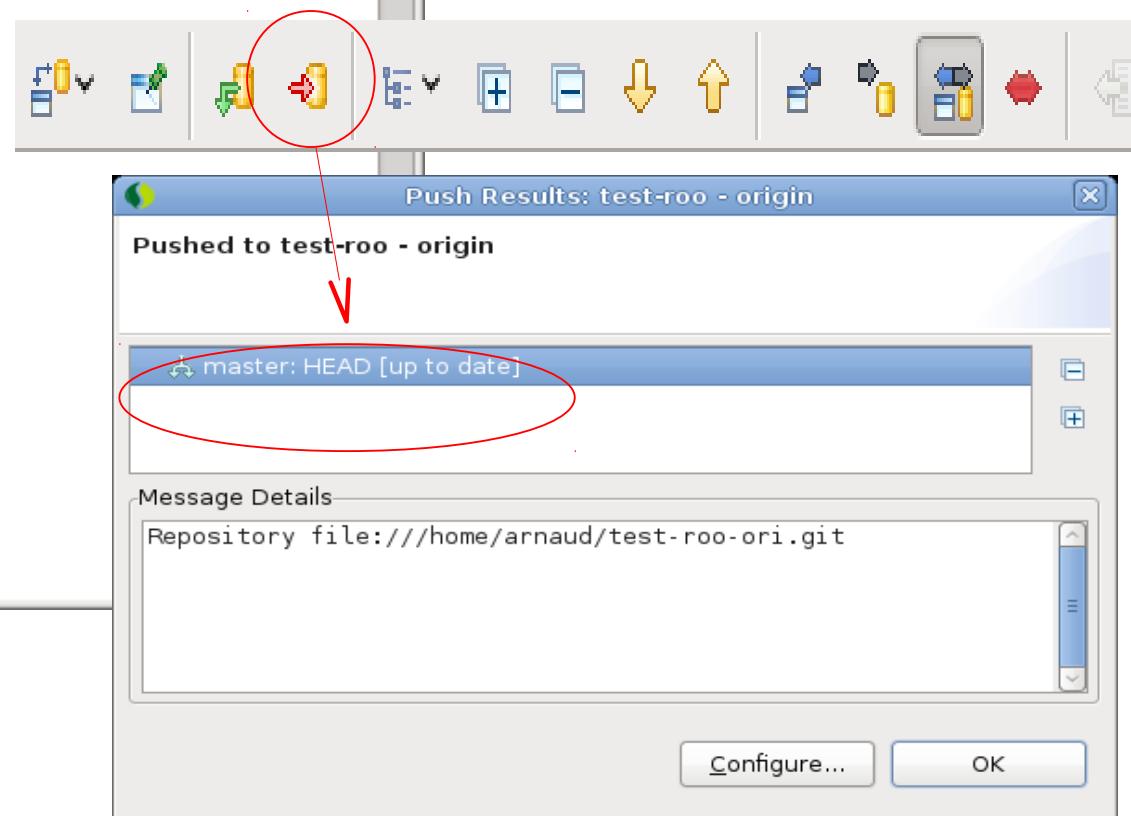
Files (3/3)

Status	Path
<input checked="" type="checkbox"/>	file-to-add.txt
<input checked="" type="checkbox"/>	file-modified.txt
<input checked="" type="checkbox"/>	file-to-remove.txt

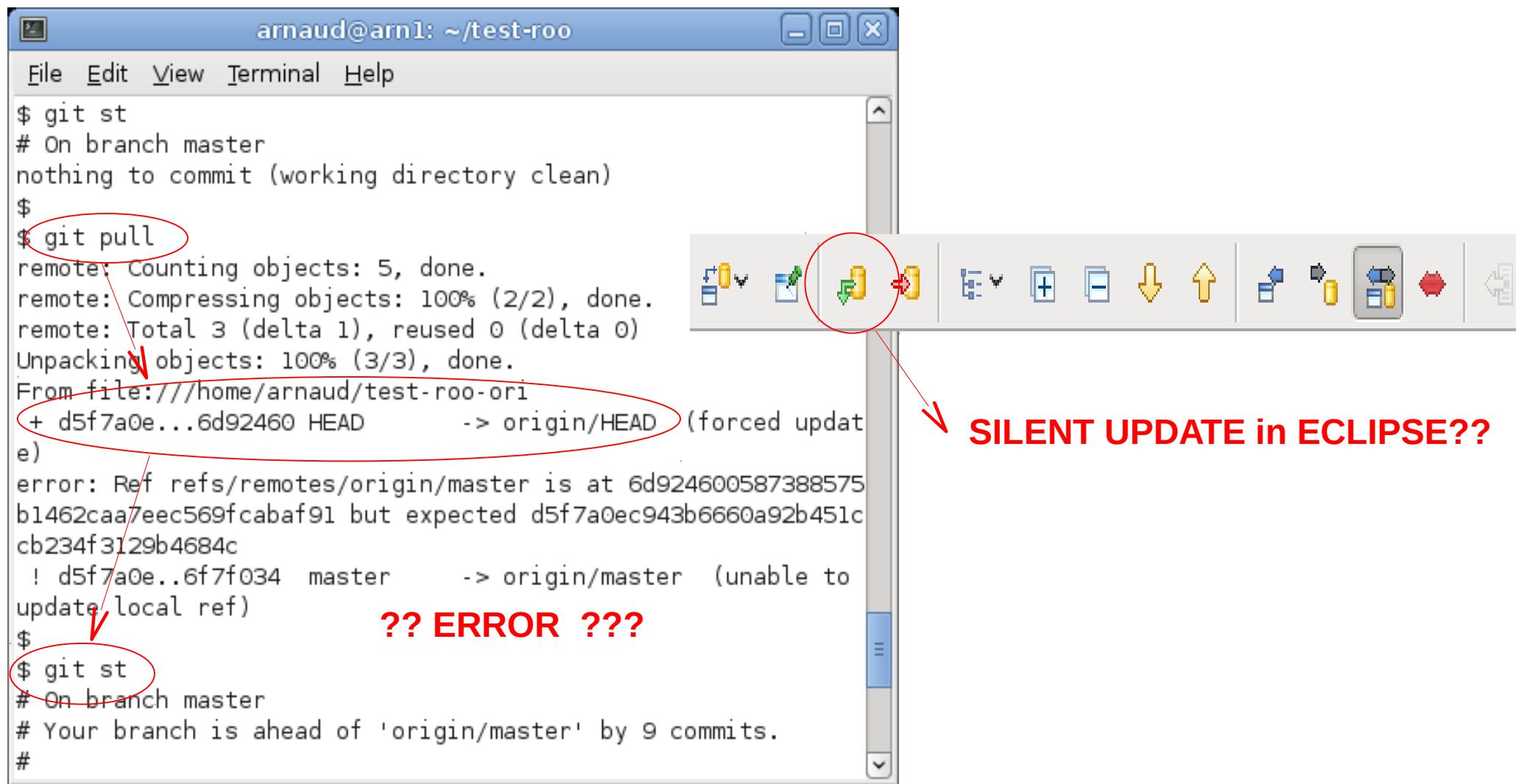
Buttons: Cancel, Commit

Git Push

```
arnaud@arn1: ~/tmp
File Edit View Terminal Help
$  
$ git st  
# On branch master  
# Your branch is ahead of 'origin/master' by 1 commit.  
#  
nothing to commit (working directory clean)  
$  
$ git push  
Counting objects: 3, done.  
Delta compression using up to 2 threads.  
Compressing objects: 100% (2/2), done.  
Writing objects: 100% (2/2), 230 bytes, done.  
Total 2 (delta 1), reused 0 (delta 0)  
Unpacking objects: 100% (2/2), done.  
To file:///home/arnaud/test-roo-ori.git  
 636ee34..7791209 HEAD -> master  
$  
$ git st  
# On branch master  
nothing to commit (working directory clean)  
$
```



Git Pull (=Fetch+Merge)



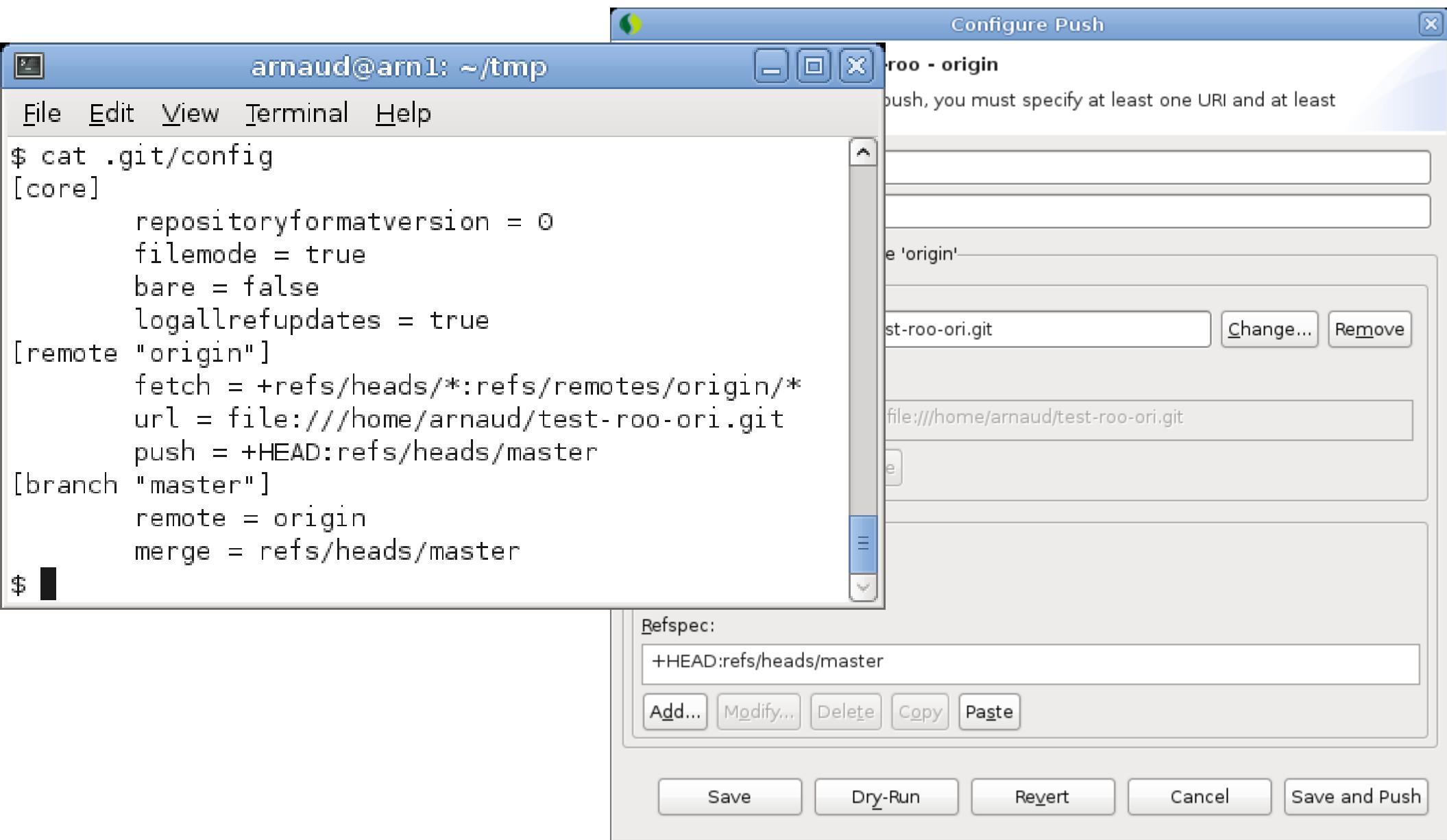
arnaud@arn1: ~/test-roo

```
$ git st
# On branch master
nothing to commit (working directory clean)
$  
$ git pull
remote: Counting objects: 5, done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0)
Unpacking objects: 100% (3/3), done.
From file:///home/arnaud/test-roo-ori
 + d5f7a0e...6d92460 HEAD      -> origin/HEAD (forced update)
e)
error: Ref refs/remotes/origin/master is at 6d924600587388575
b1462caa7eec569fcabaf91 but expected d5f7a0ec943b6660a92b451c
cb234f3129b4684c
 ! d5f7a0e..6f7f034 master      -> origin/master (unable to
update local ref)
$ ?? ERROR ???
$ git st
# On branch master
# Your branch is ahead of 'origin/master' by 9 commits.
#
```

The screenshot shows a terminal window within the Eclipse IDE interface. The terminal output displays a successful 'git pull' command, followed by an error message about a forced update and a local ref update failure. A red circle highlights the 'git pull' command, and another red circle highlights the error message. A red arrow points from the error message to the text '?? ERROR ???'. The Eclipse toolbar is visible at the top, featuring various icons including a merge icon (highlighted with a red circle) and a refresh icon.

SILENT UPDATE in ECLIPSE??

Egit Fetch/Push Config



Git / eGit ...

- Errors are often difficult to Understand
- IMPORTANT to understand **Git CONCEPTS**
- Command line shell knowledge is necessary
UI Tool optionnal (eGit, Tortoise)

Conclusion

GIT is Beautiful !

Question ?

arnaud.nauwynck@gmail.com

This document:

<http://arnaud.nauwynck.free.fr/CoursIUT/CoursIUT-IntroGIT.pdf>