Cours IUT
current version 2016-09

An Introduction to GIT

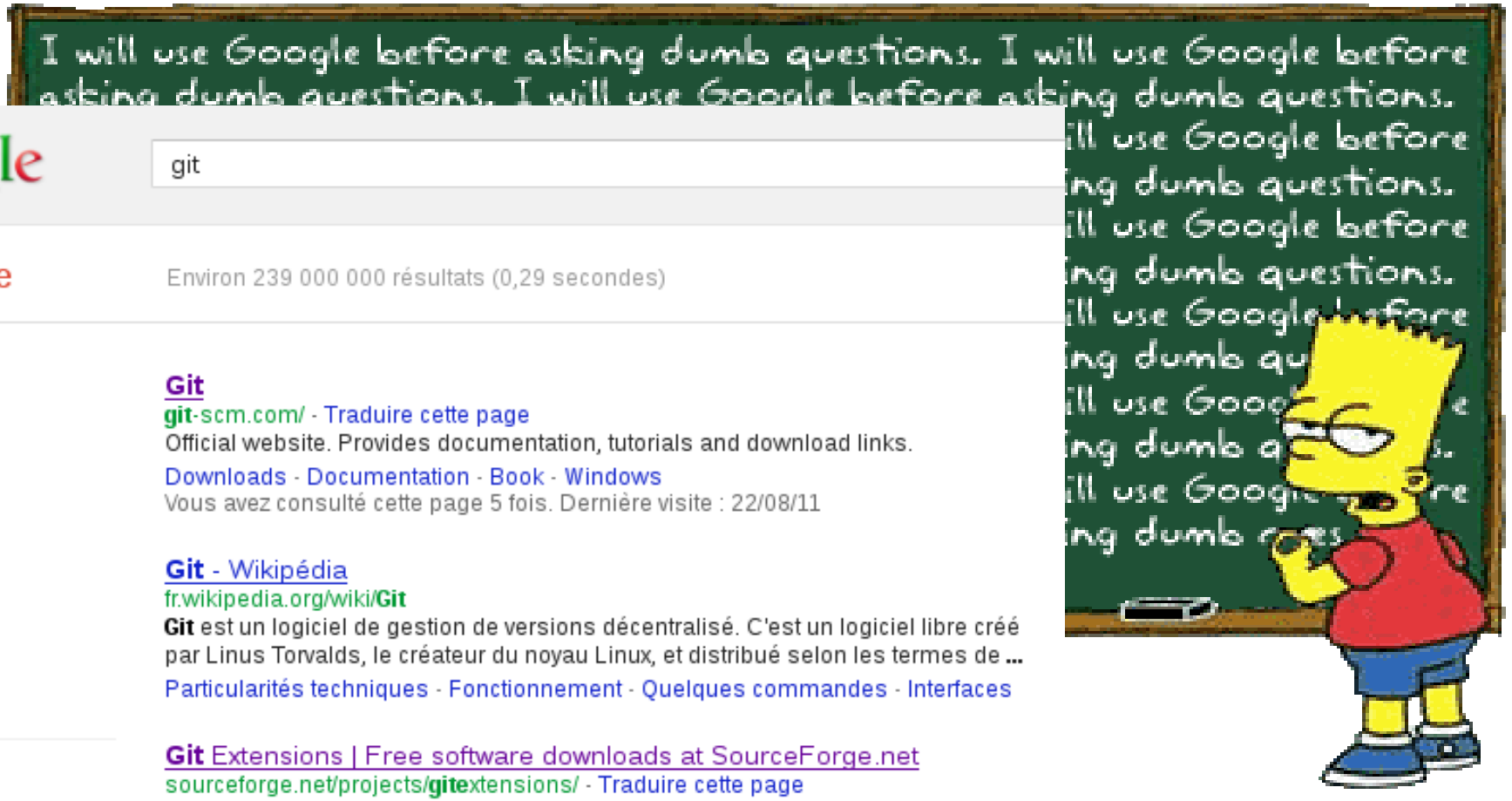
arnaud.nauwynck@gmail.com

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?)



Google

git

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-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/**git**extensions/ - 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://**git**hub.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)

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

Git logo

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Interaction
Help
About Wikipedia
Community portal

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/

git --everything-is-local

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."

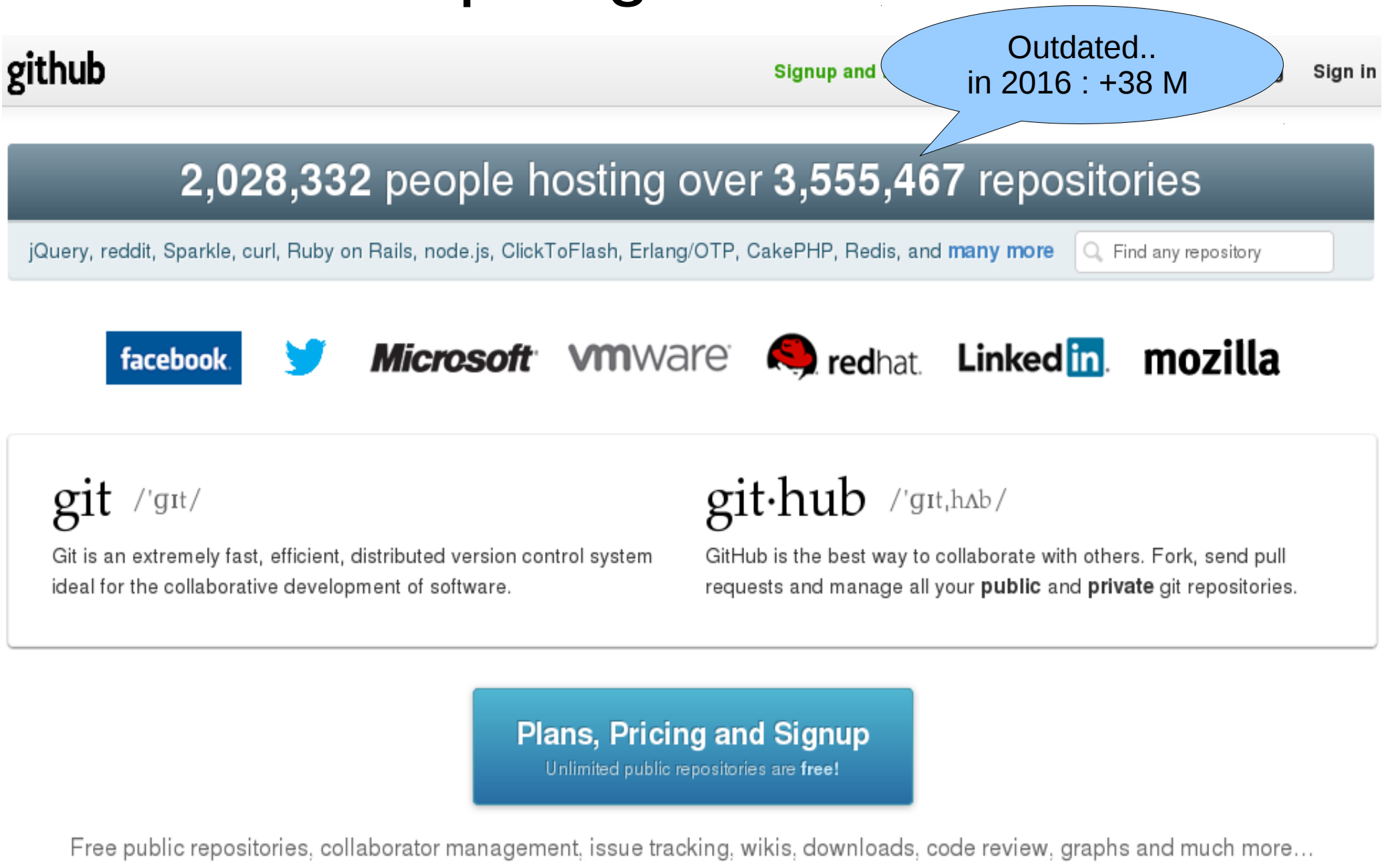
Who is using it ? ~ALL & ME
Who is not using it ? (poor svn guys)

github

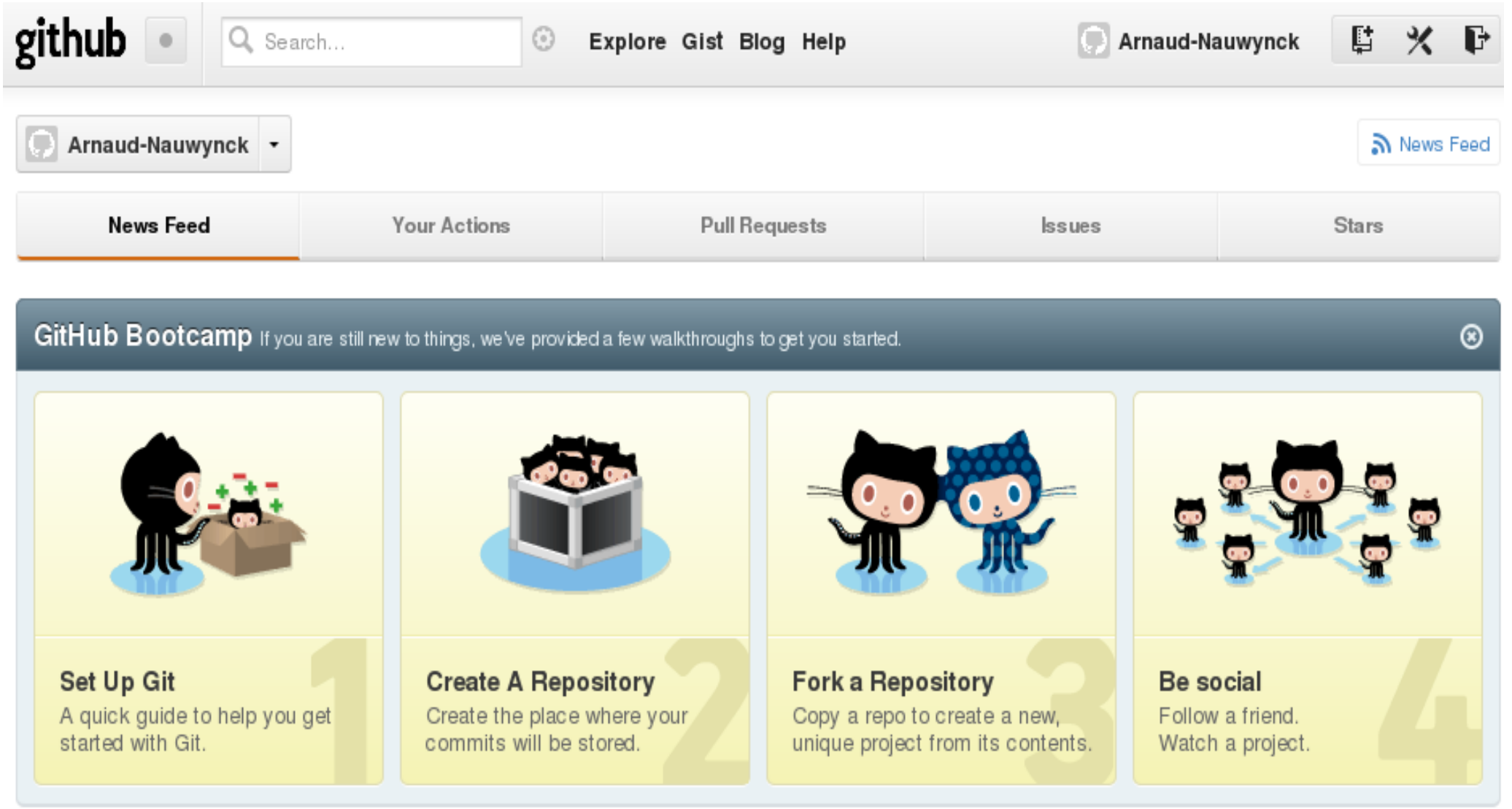
2,028,332 people hosting over 3,555,467 repositories

sourceforge

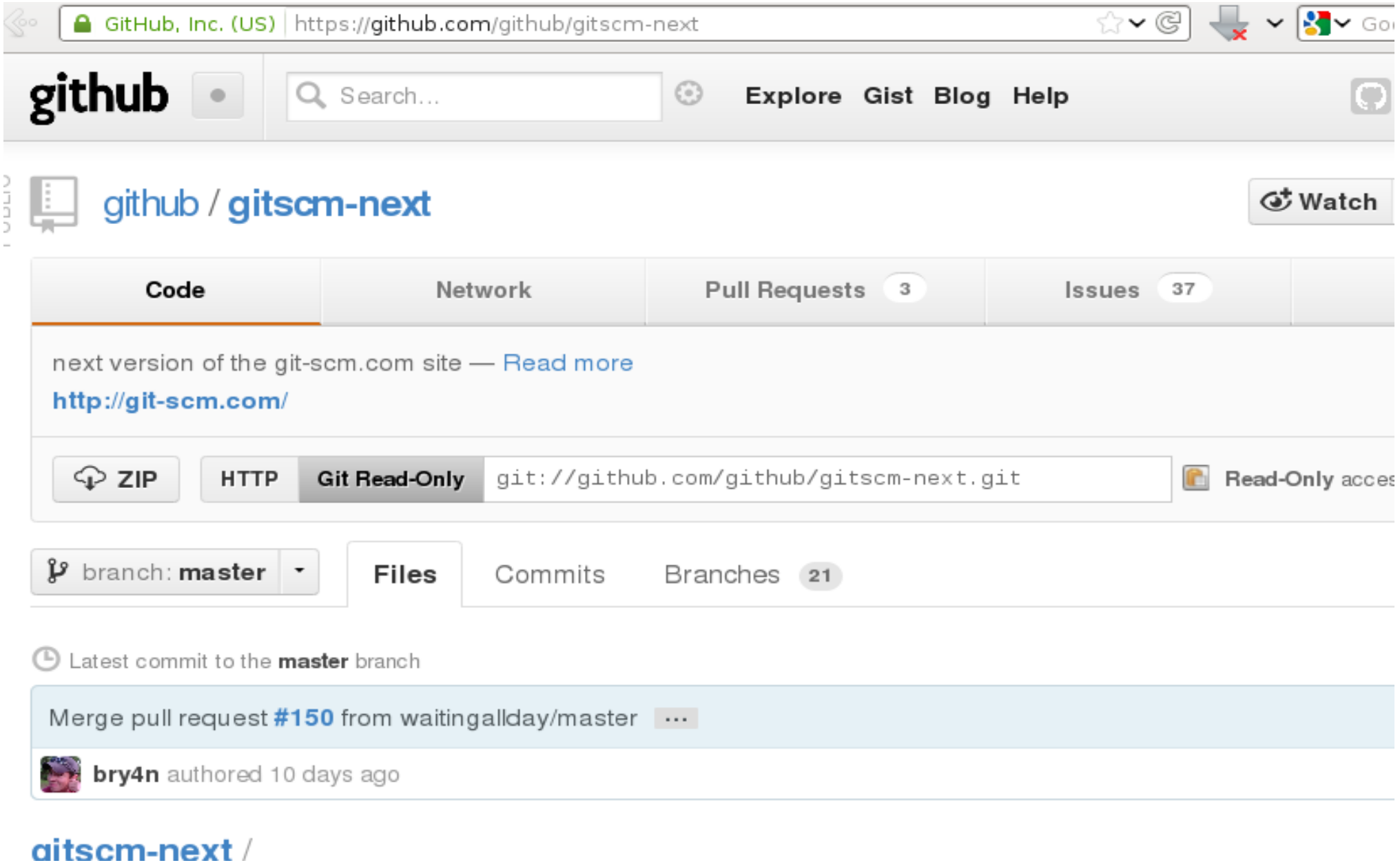
https://github.com/



Create GitHub Repos

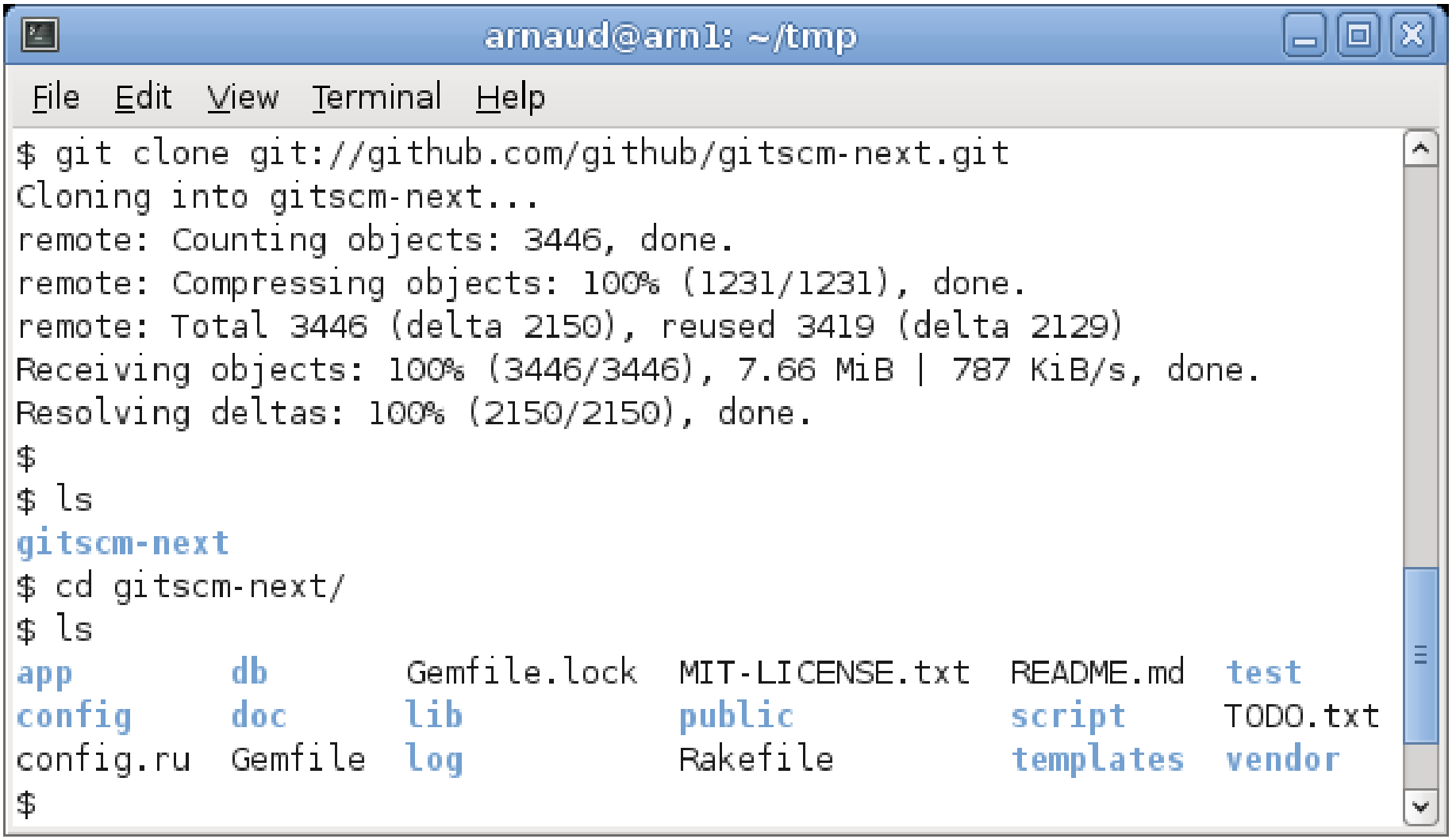


Git Repository On GitHub



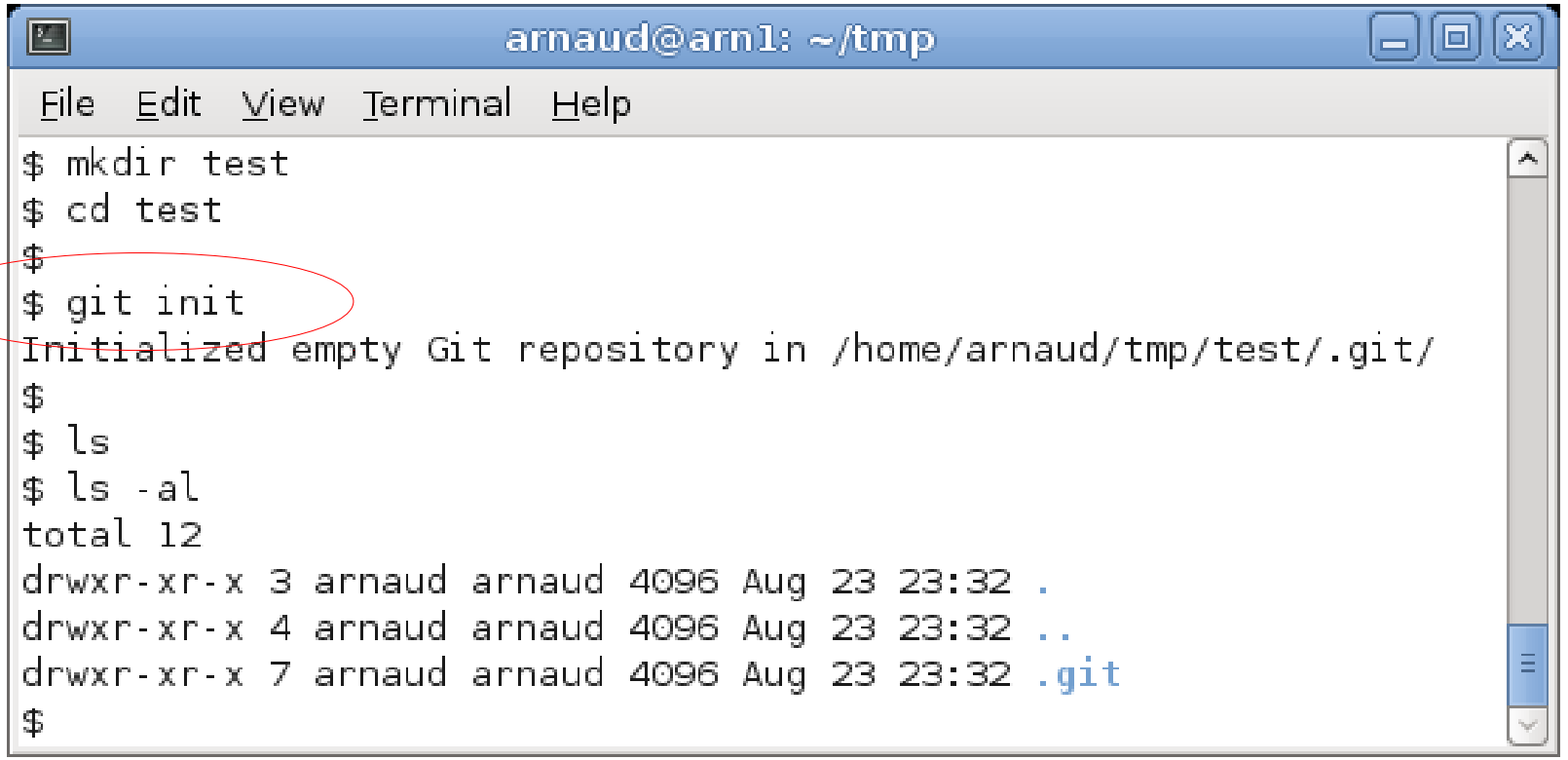
Clone (=Download) from GIT

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

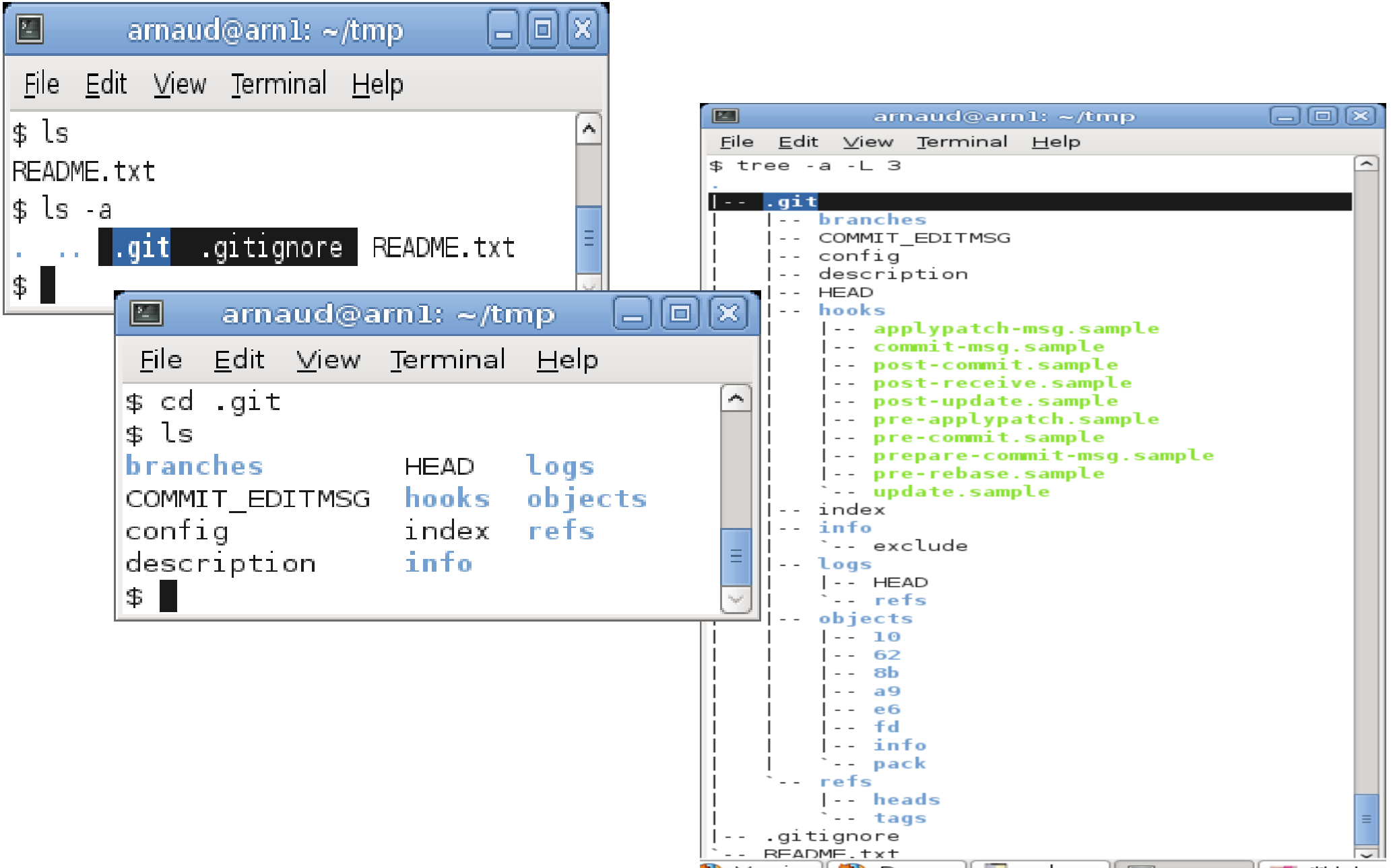


Creating a Local Git Repo

- $ **git init** … THAT'S ALL !!!
- $ git add -a
- $ git commit -m “init comment”



.git : hidden local repo files



Basic Git Commands

$ git status

$ git add -a

$ git commit -m “comment”

$ git diff

$ git checkout

$ git branch

$ git merge

$ git log

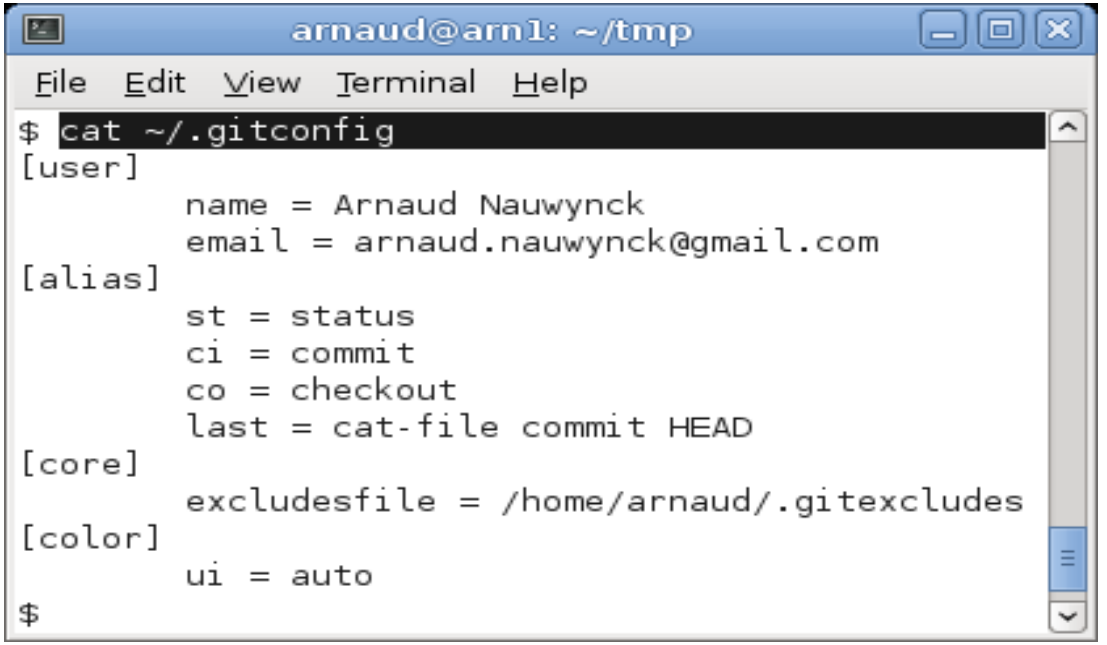
$ git pull / push

Git Config, Git Alias ...

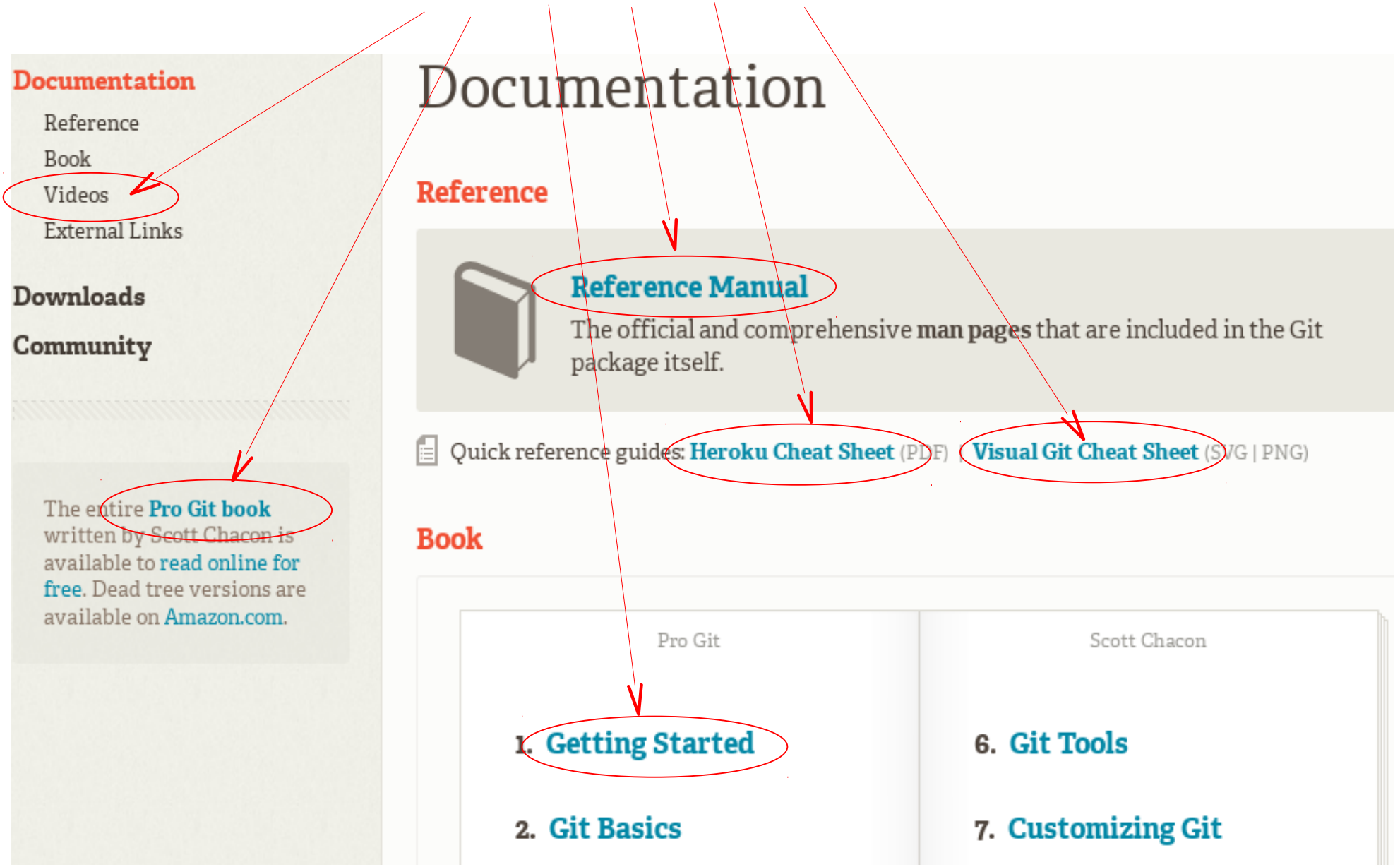
- Git is highly configurable
- In <<repo>>/.git

- <<repo>>/.gitignore
- In ~/.gitconfig

  sample alias:
  “ci” = “commit”



GIT Documentation(sss)



Sample Git Books

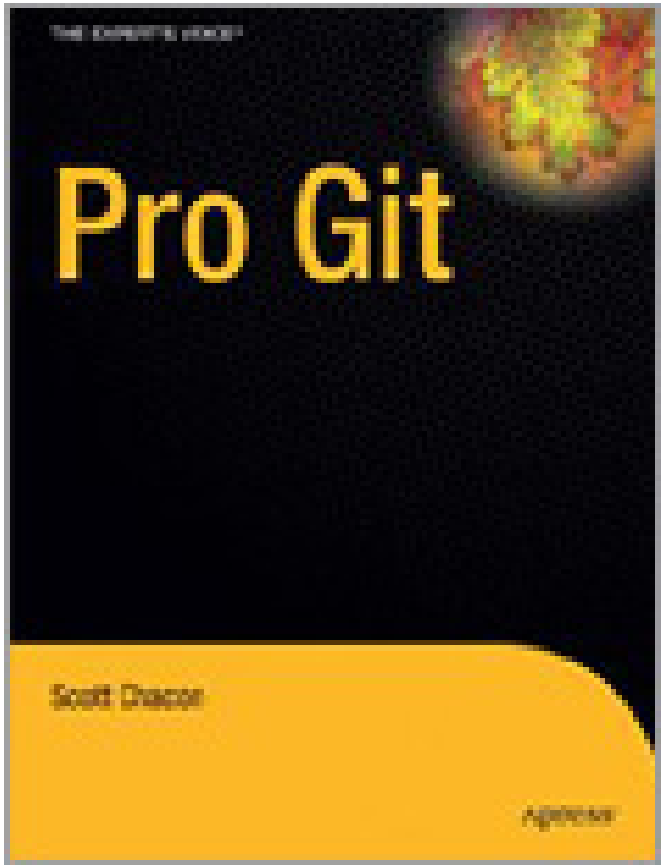
Free from Scott Chacon

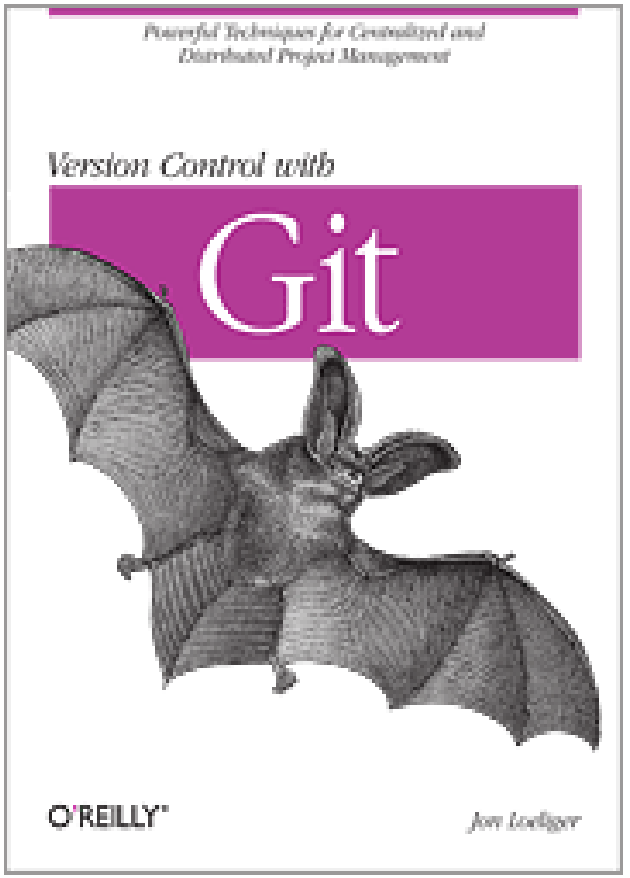
https://github.s3.amazonaws.com/media/progit.en.pdf

Download this book in PDF, mobi, or ePub form for free.

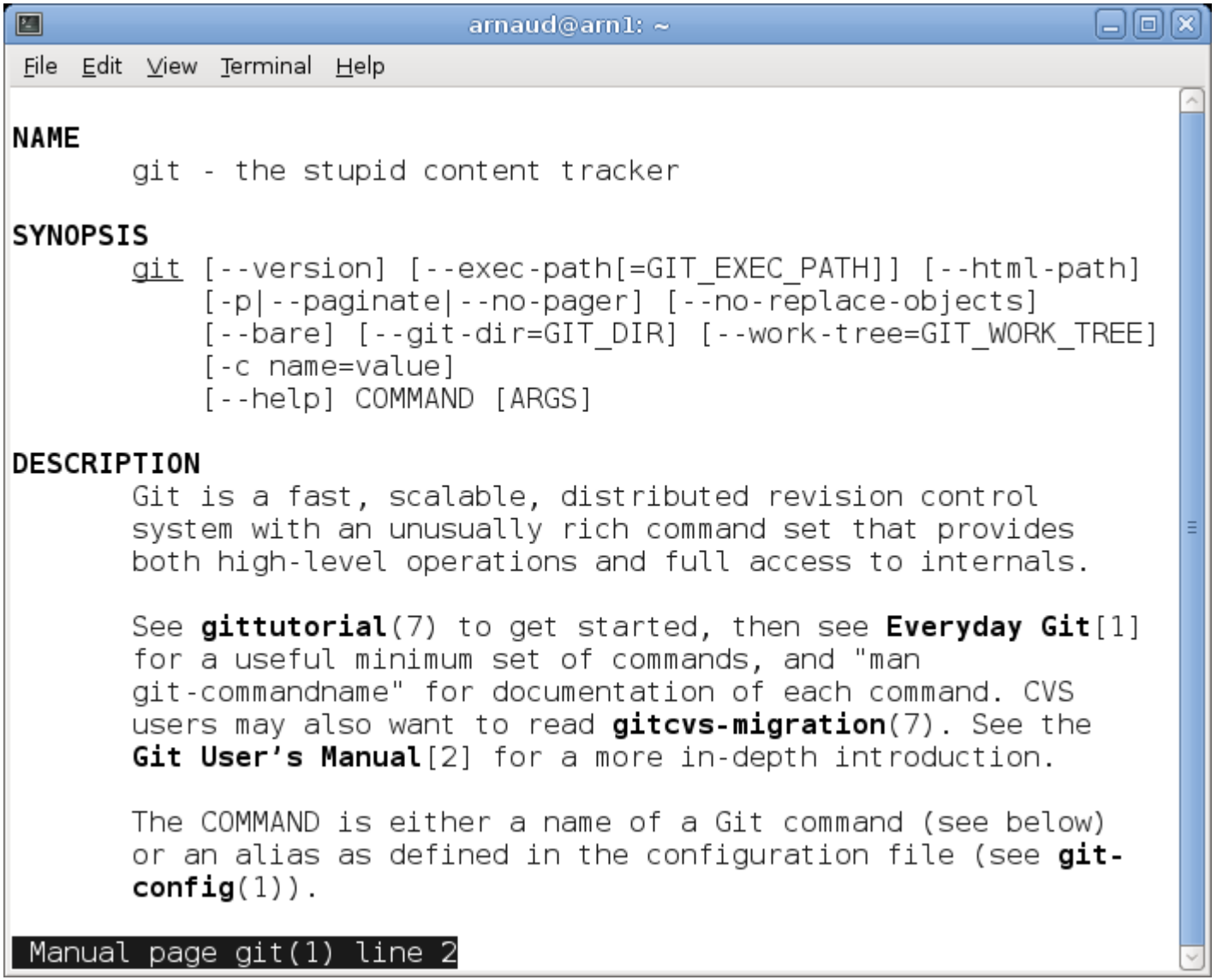
This book is translated into German, Chinese, French, Japanese and Dutch.

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





Man Git … Git Help



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 wor
   clone       Clone a repository into a new directo
   commit      Record changes to the repository
   diff        Show changes between commits, commit
   fetch       Download objects and refs from anothe
   grep        Print lines matching a pattern
   init        Create an empty git repository or rei
   log         Show commit logs
   merge       Join two or more development historie
   mv          Move or rename a file, a directory, o
   pull        Fetch from and merge with another rep
   push        Update remote refs along with associa
   rebase      Forward-port local commits to the upd
   reset       Reset current HEAD to the specified s
   rm          Remove files from the working tree an
   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 sp
$
```

Even more …

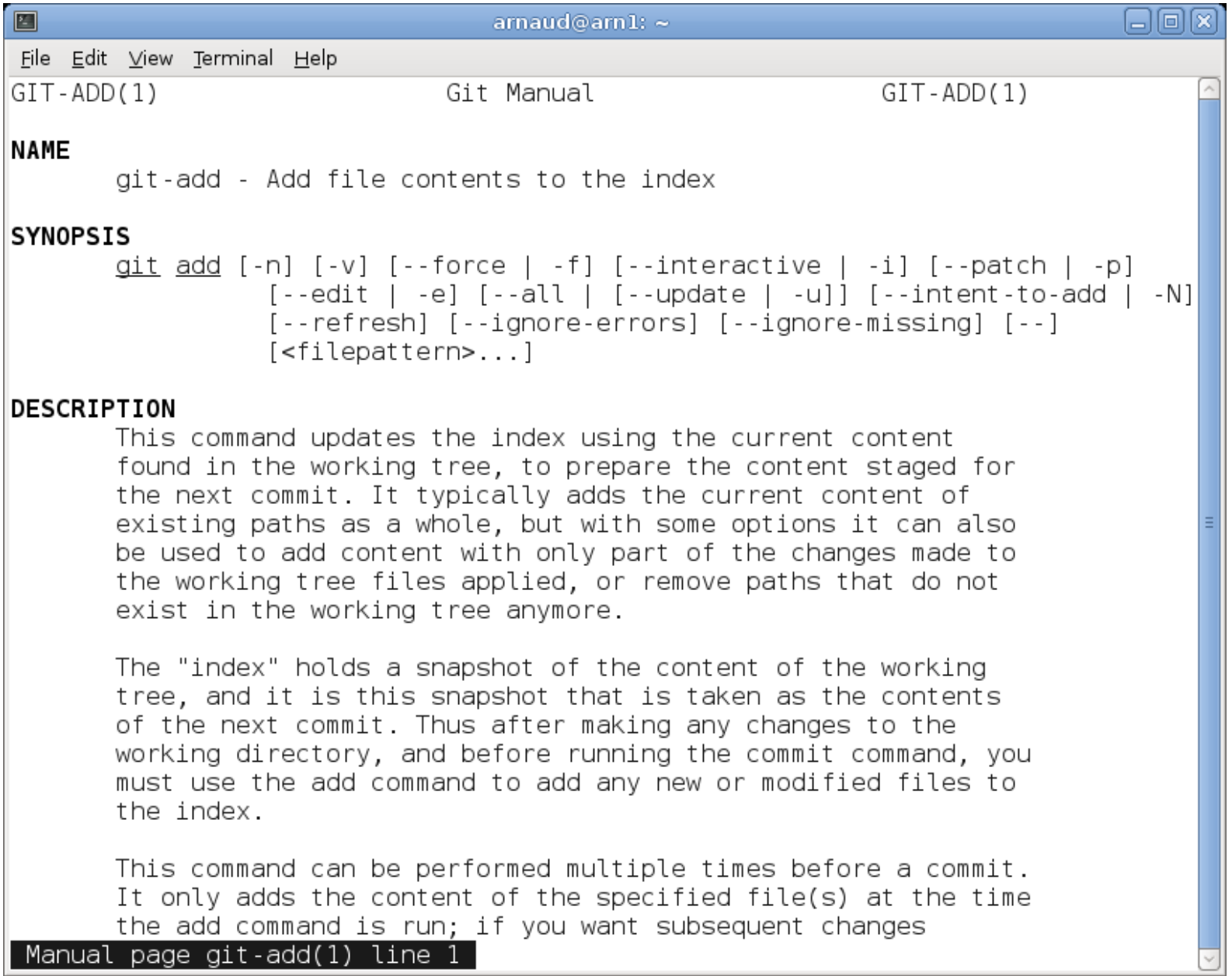
$ git help –all

```
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>>

Example :
$ git help add
$ man git-add



Understand Git from Commands?

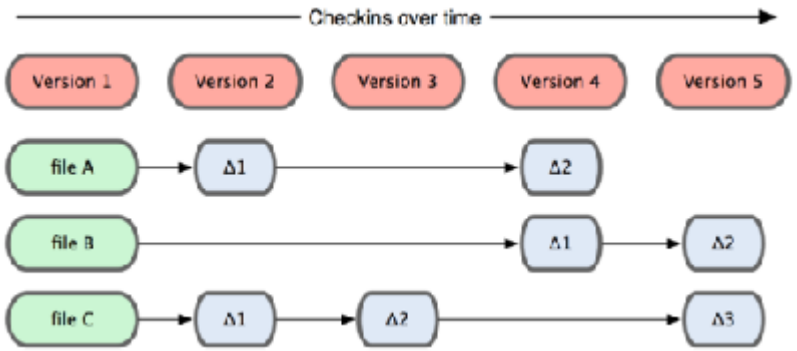

```
$ git reset origin/some-branch^2{4}^^ –-soft
$ git commit -a –fixup=HEAD^
detached HEAD
conflict
...
```

You have to understand Git from Internals

Git : Content Tracking, not DIFF

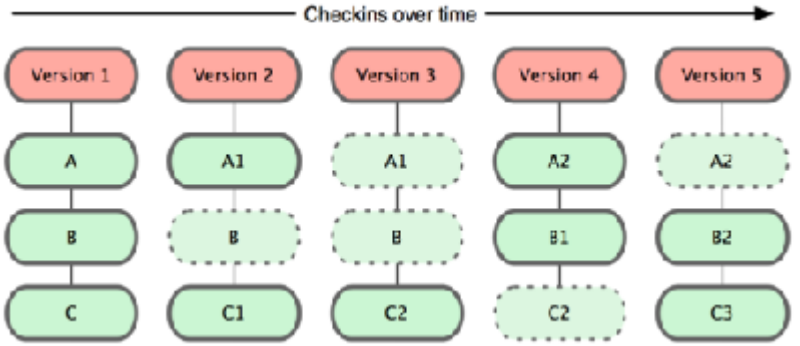
Subversion

Store DIFFs between versions

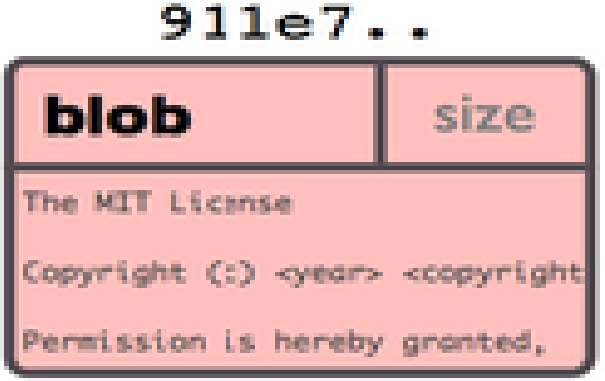


GIT

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



Git SHA1, Blob

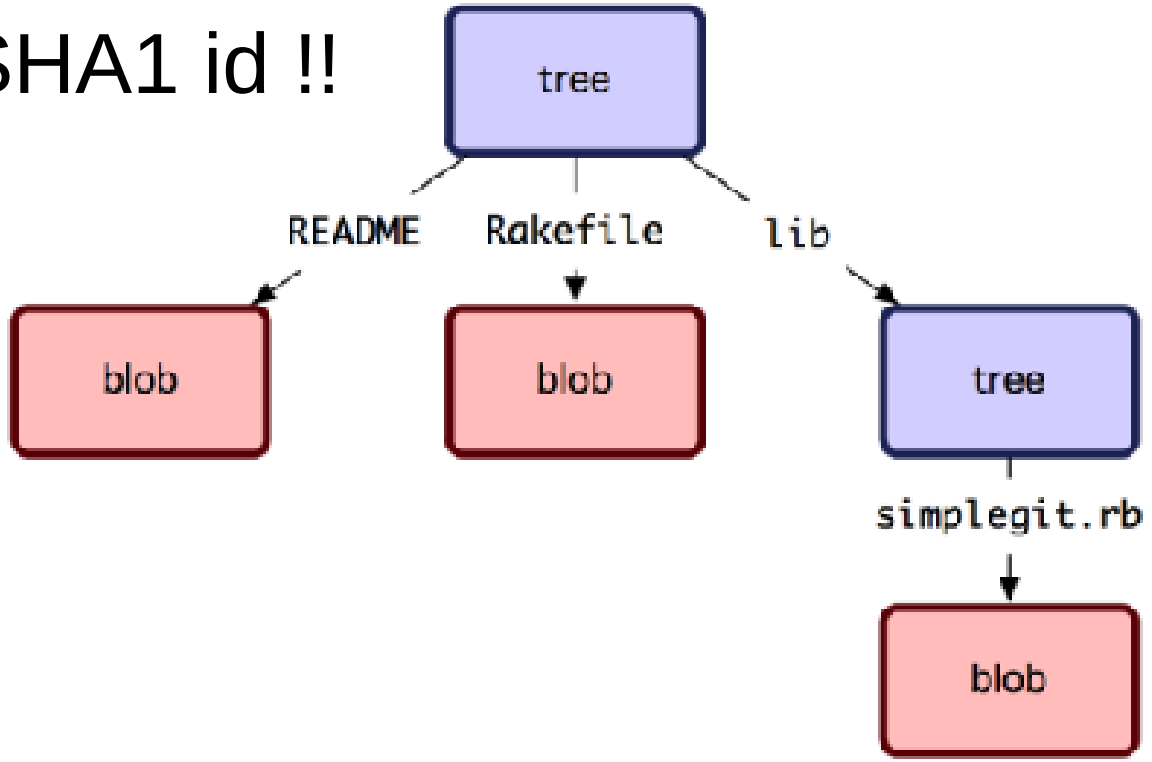


- A BLOB (content file) has a SHA1
- Independently of its file name
- SHA1 = cryptographic hash-code (256 bytes)
  - identical content => same SHA1
  - Same SHA1 => same content
    (otherwise probability 1/8^256 = never ever)

- File are indexed per SHA1: in .git/objects/...

Git Tree

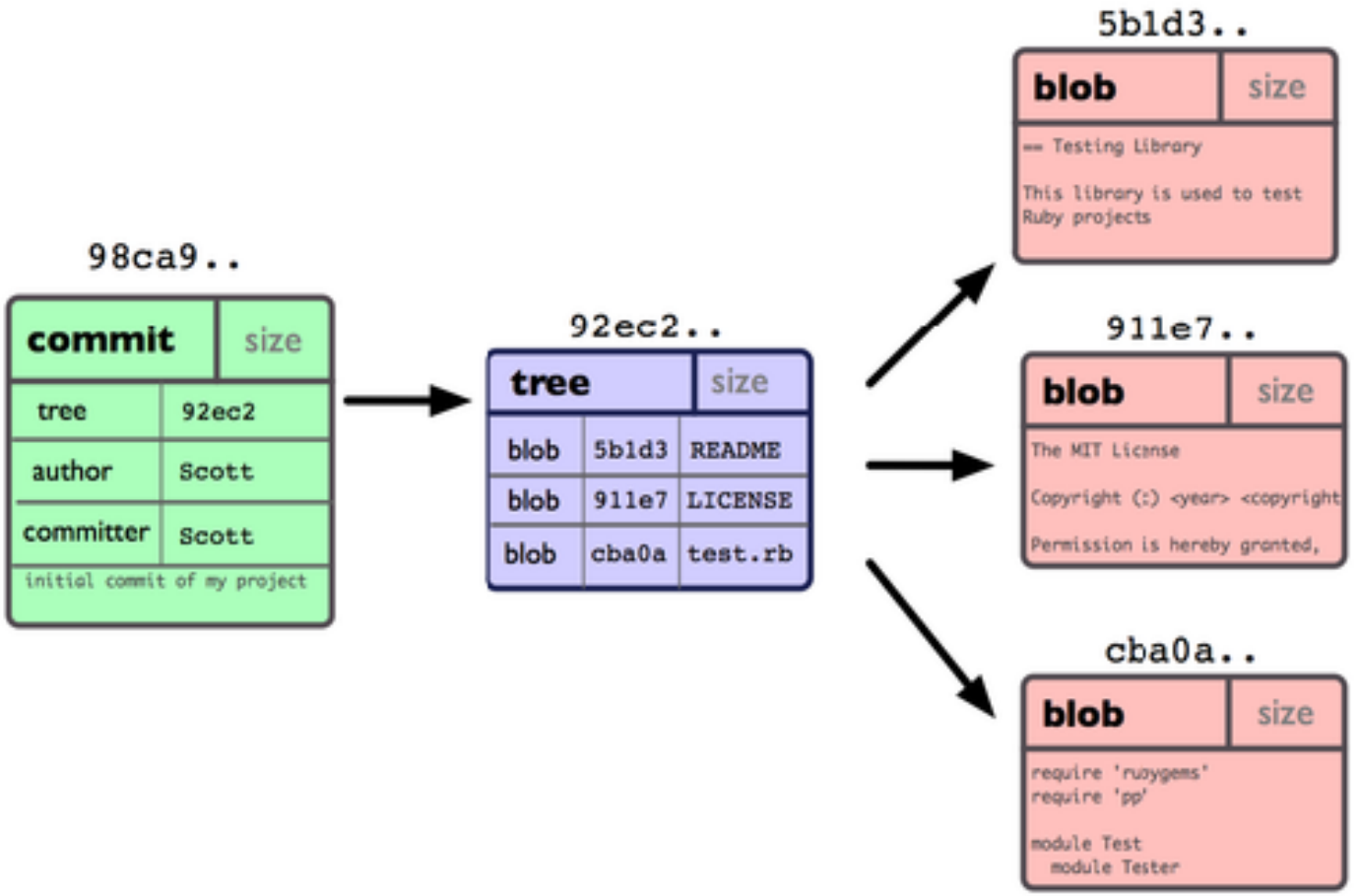
- A “Tree” object represent a Directory
- It is a list of name->Tree and name->Blob (list name->SHA1)
- Tree also have SHA1 id !!



Tree / DAG

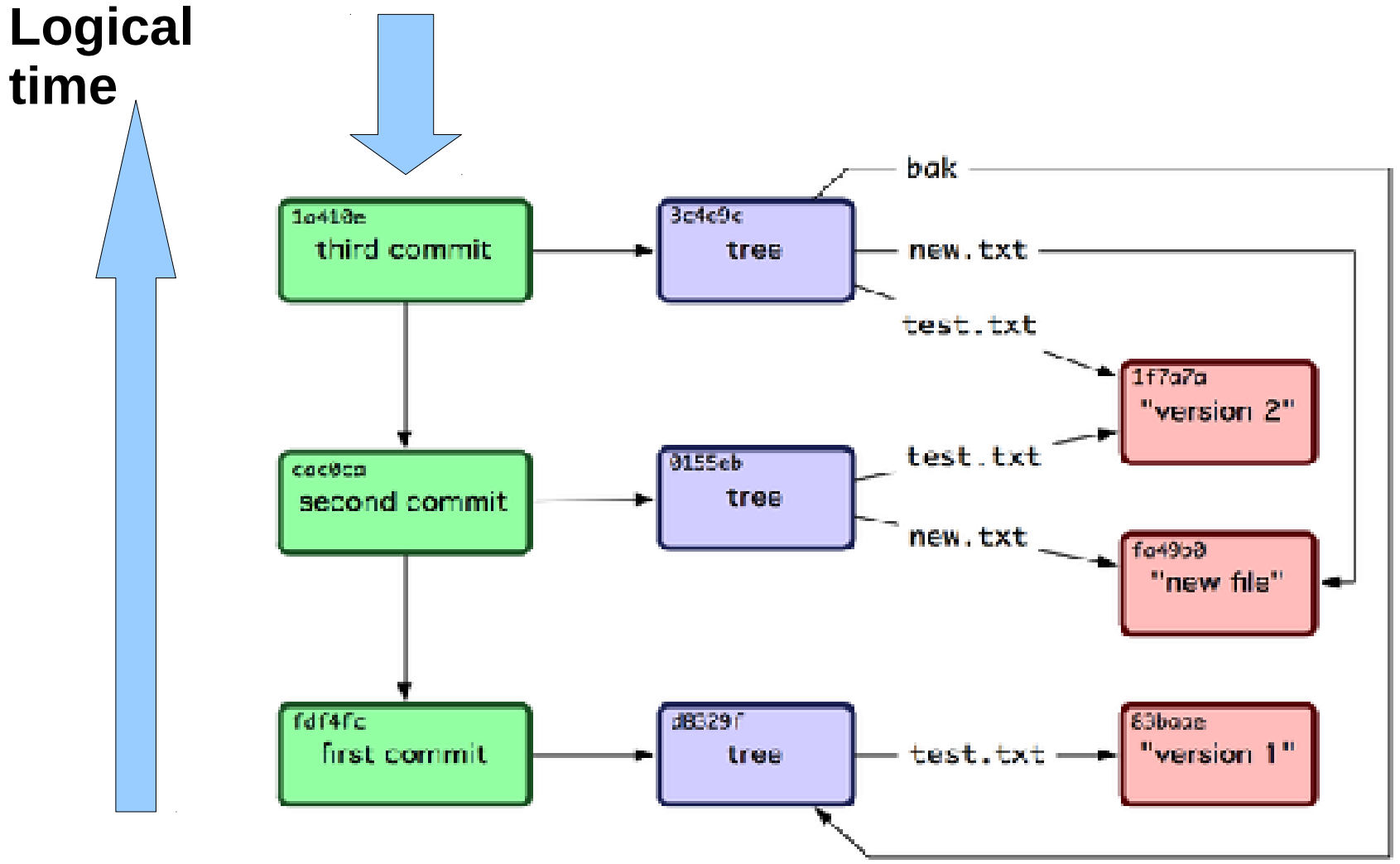
- Tree is like an Inode in a filesystem : allow recursion of dir/sub-dir...
- BUT sames Trees and sames Blobs are immutable and can be shared by pointers in the graph
- DAG = Directed Acyclic Graph
- VERY efficient compression of data, and versionning

Git Commit → Tree → Blobs



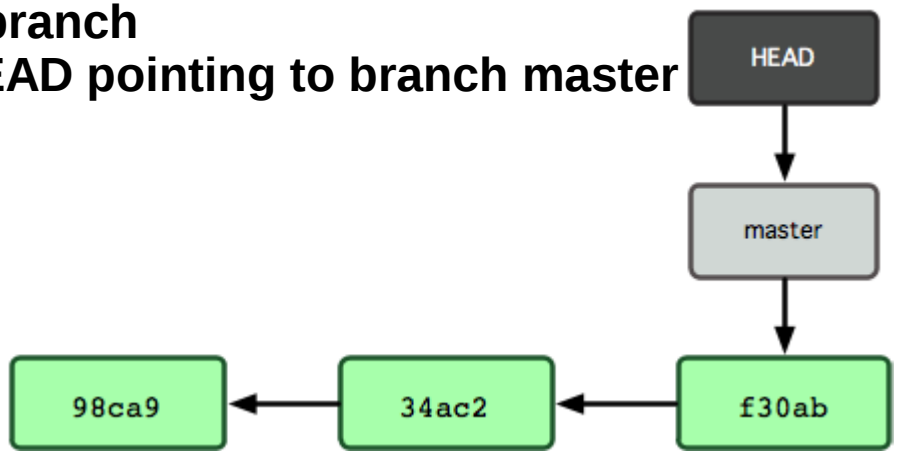
Git Commit History

Ref: HEAD (insertion of new commits)

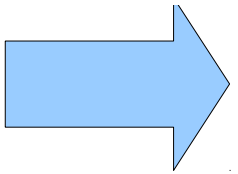


Git Branches ...

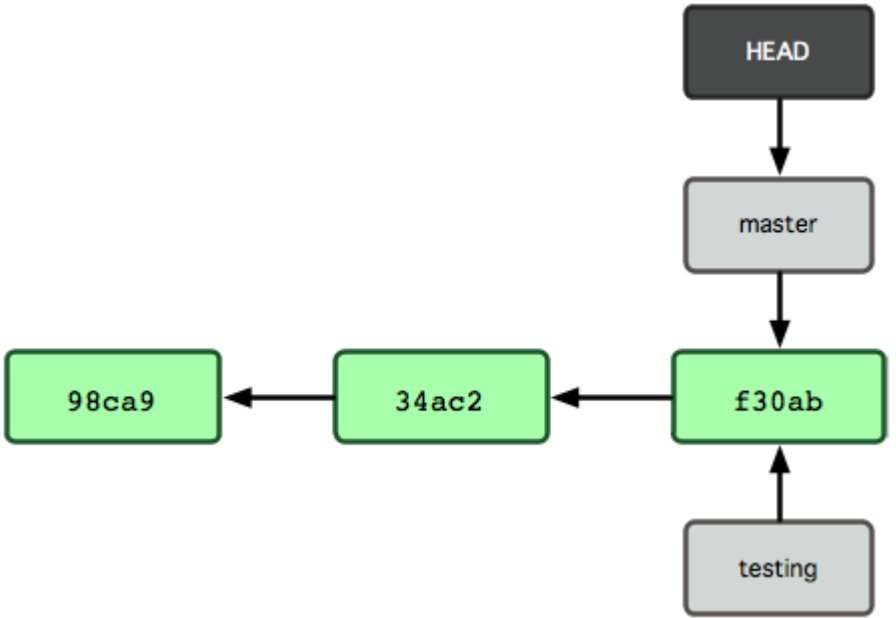
1 branch
HEAD pointing to branch master




```
$ git branch testing
```

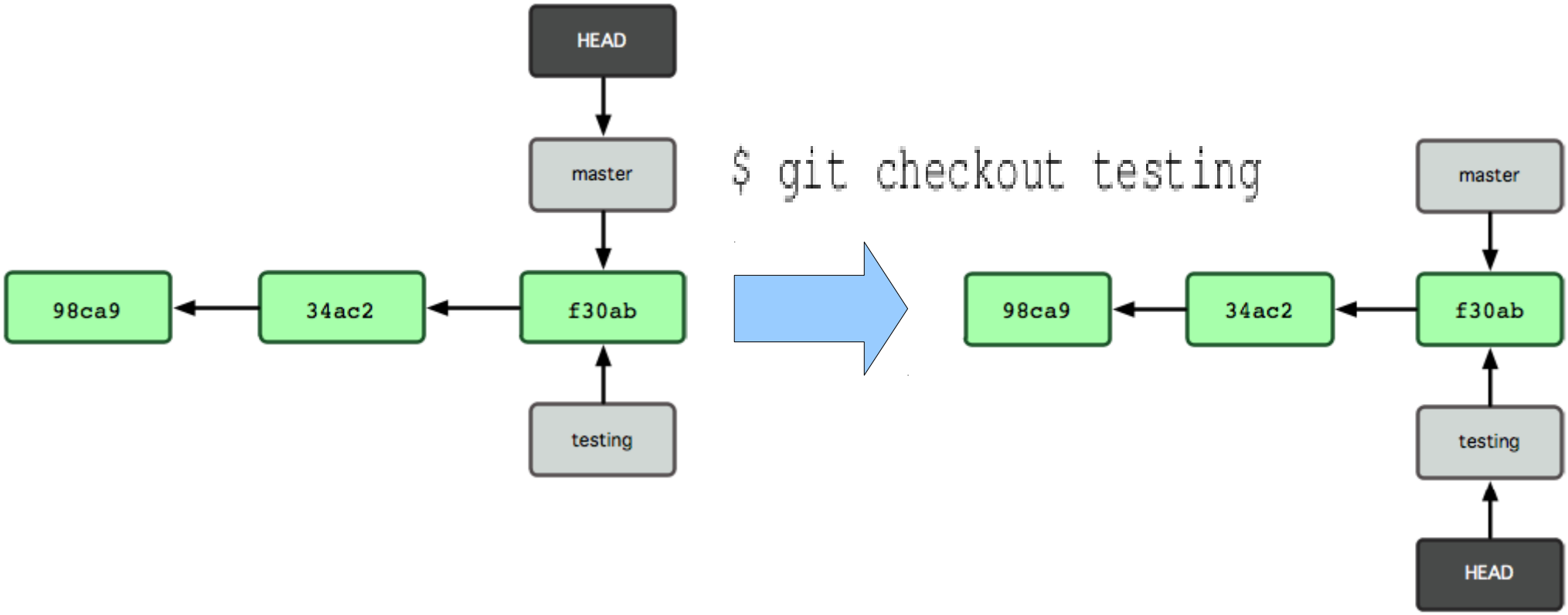


2 branches pointing to the same commit
(=> same Tree)
HEAD pointing to branch master



Checkout Branch

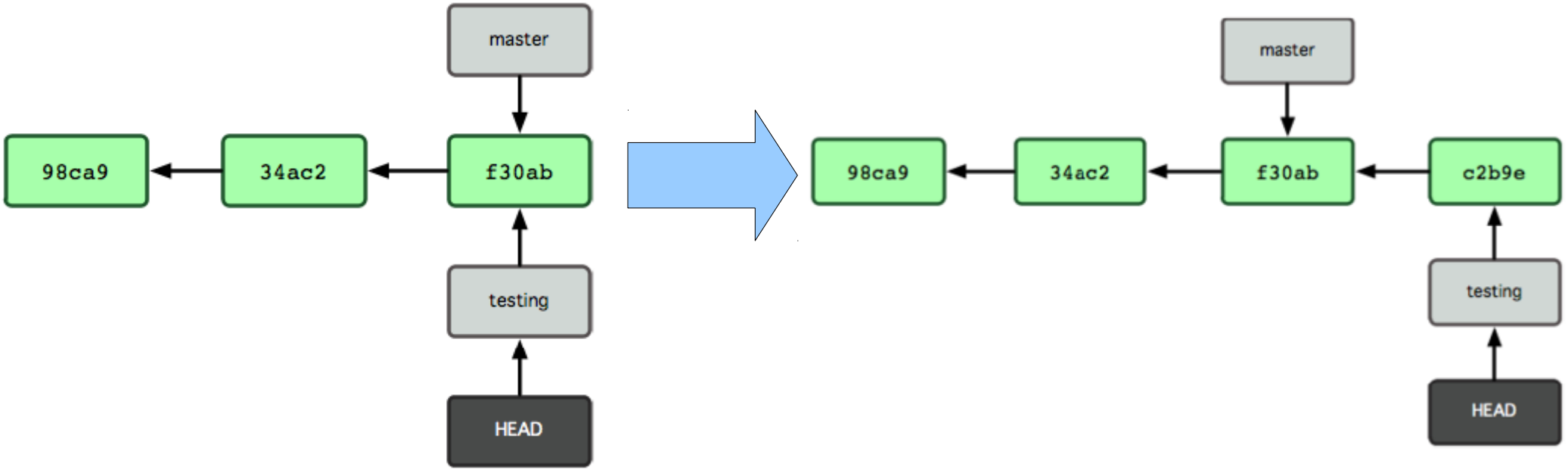
HEAD pointing to branch master



HEAD pointing to branch testing

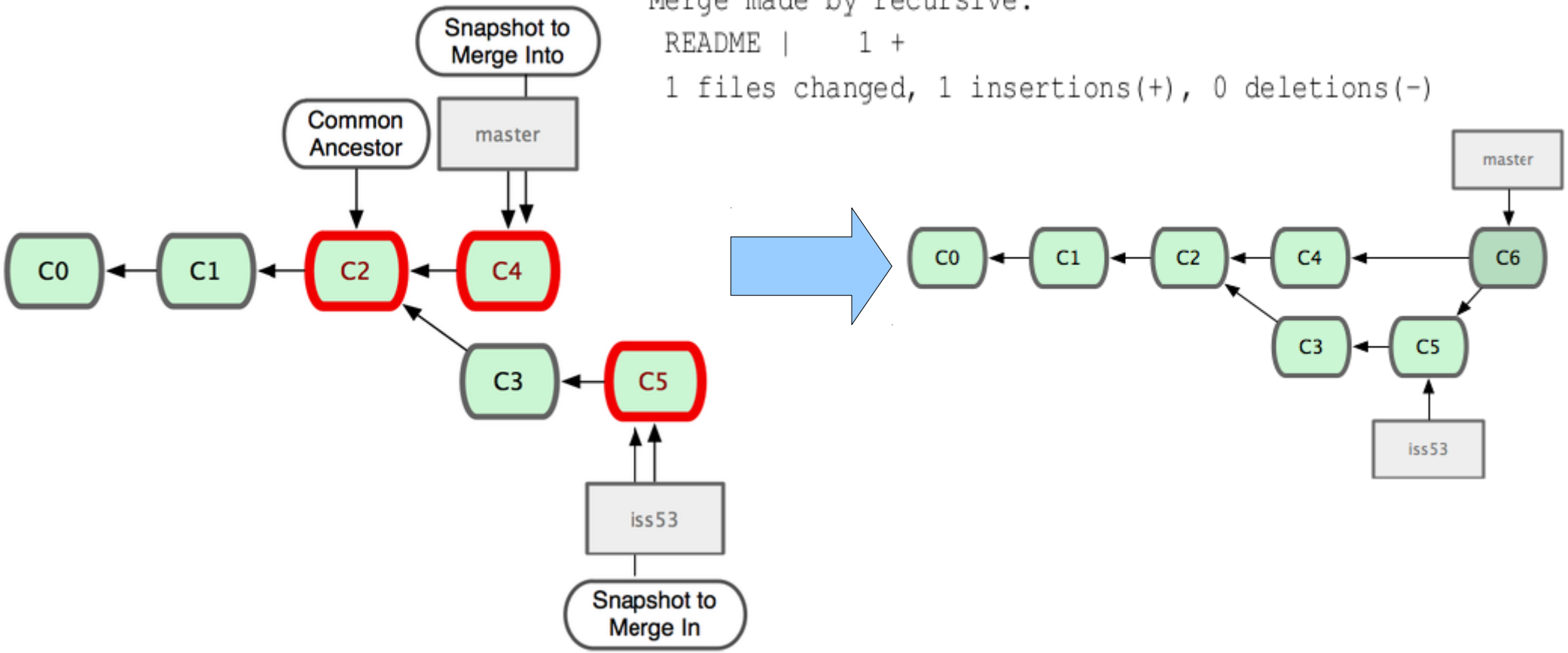
Git Commit (from HEAD branch)

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



Git Merge

```
$ git checkout master
$ git merge iss53
Merge made by recursive.
 README |    1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
```



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

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.

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

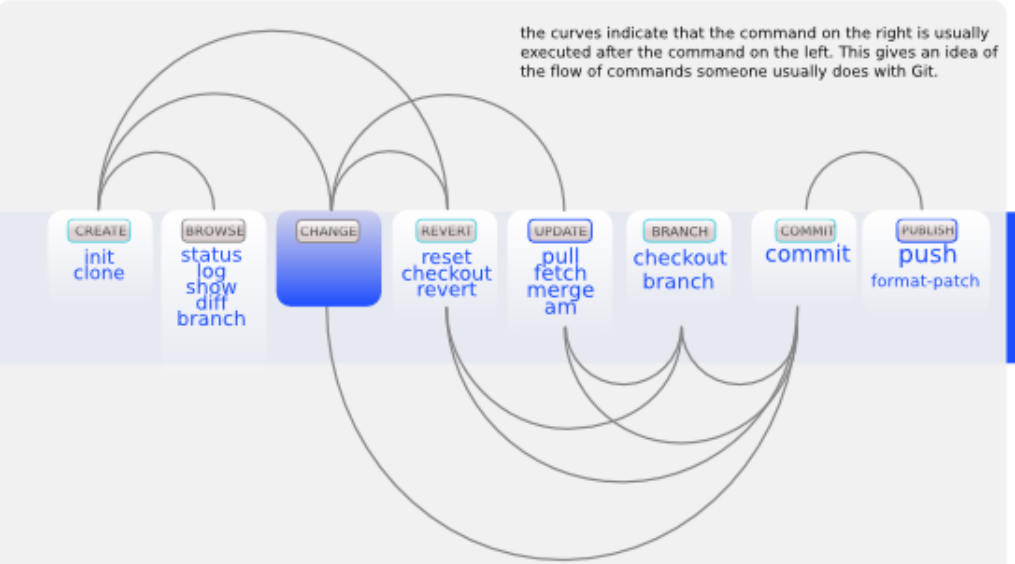
Git Cheat Sheet

http://git.or.cz/

Remember: git command --help

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

Commands Sequence



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)

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
```

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 conclicts

```
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

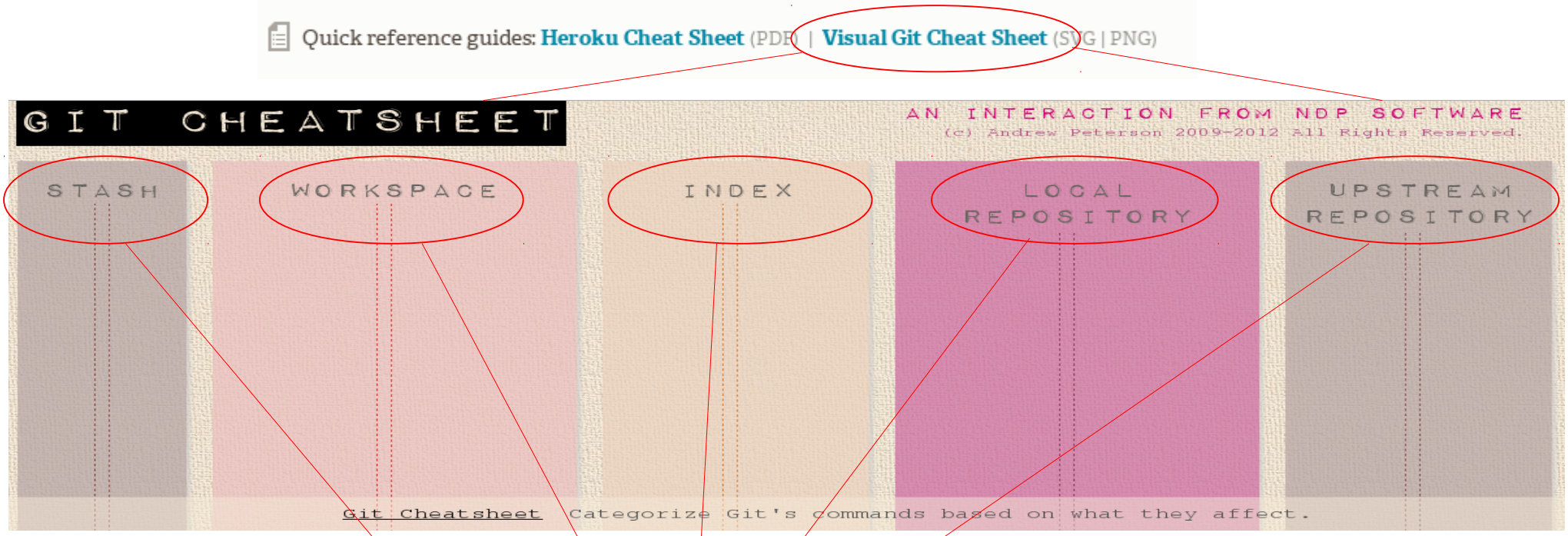
```
git add $conflicting_file  (do for all resolved files)
git rebase --continue
```

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

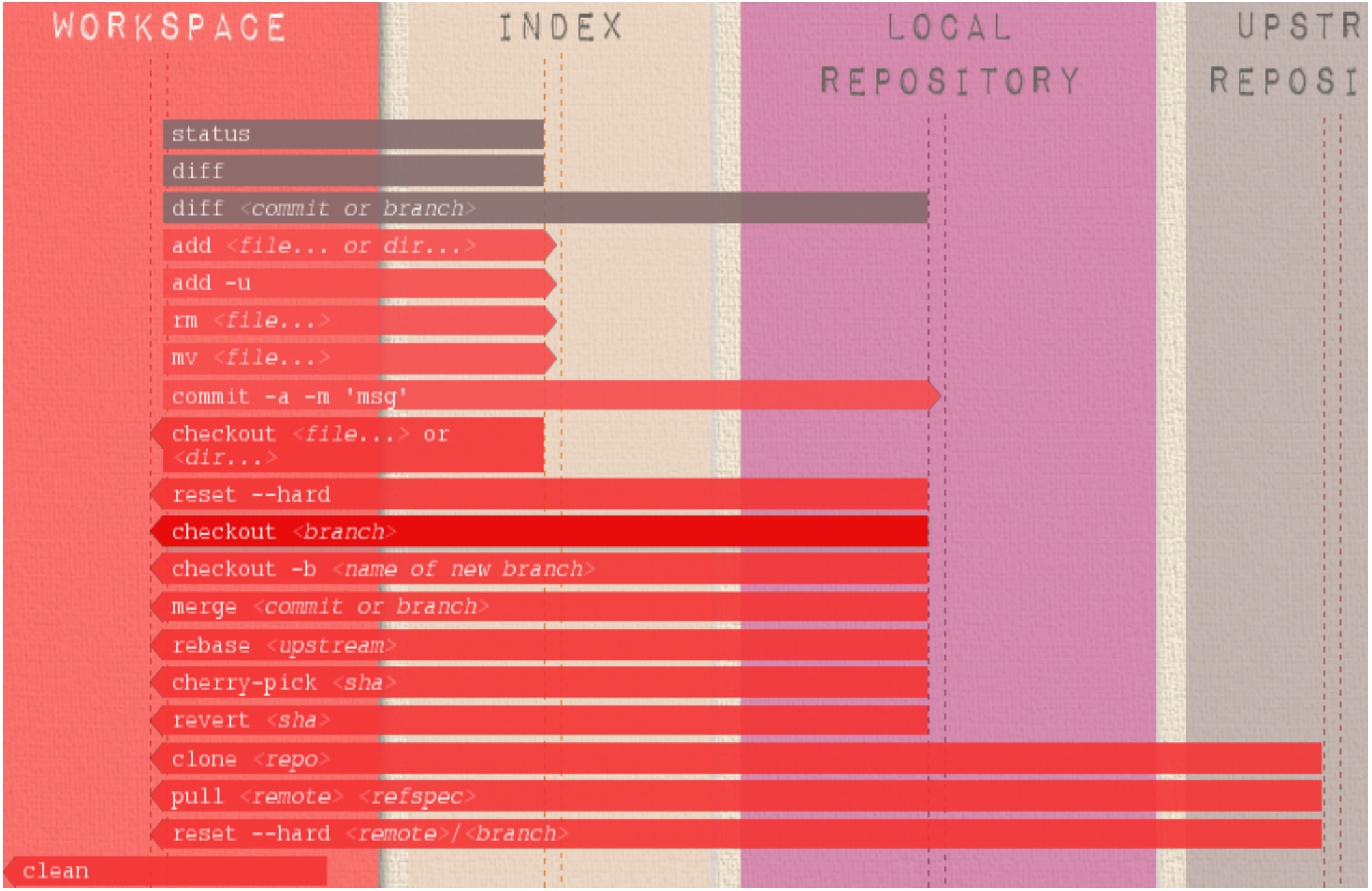
Interactive Graphical Cheat Sheet
from Git Home page

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

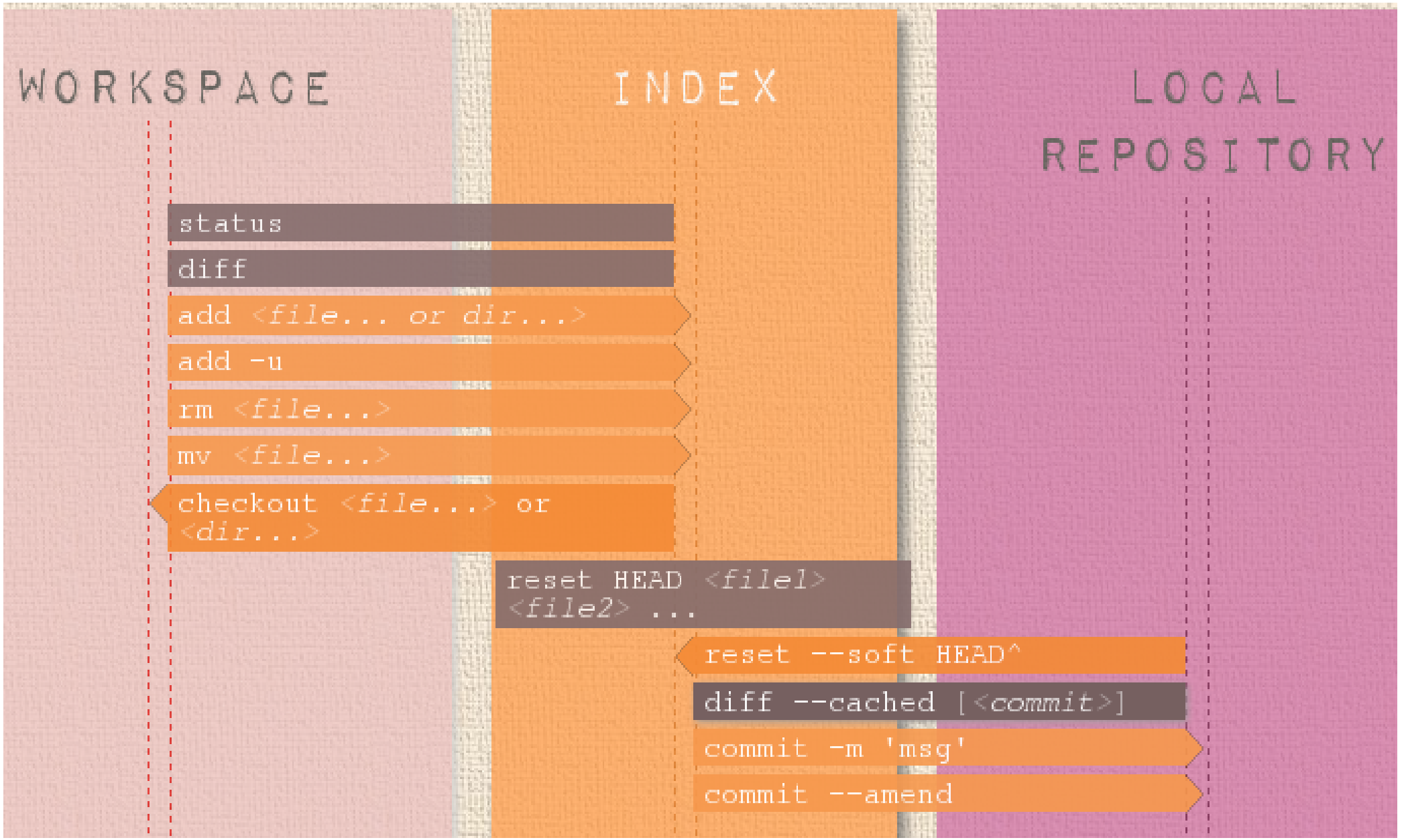


See Next Slides

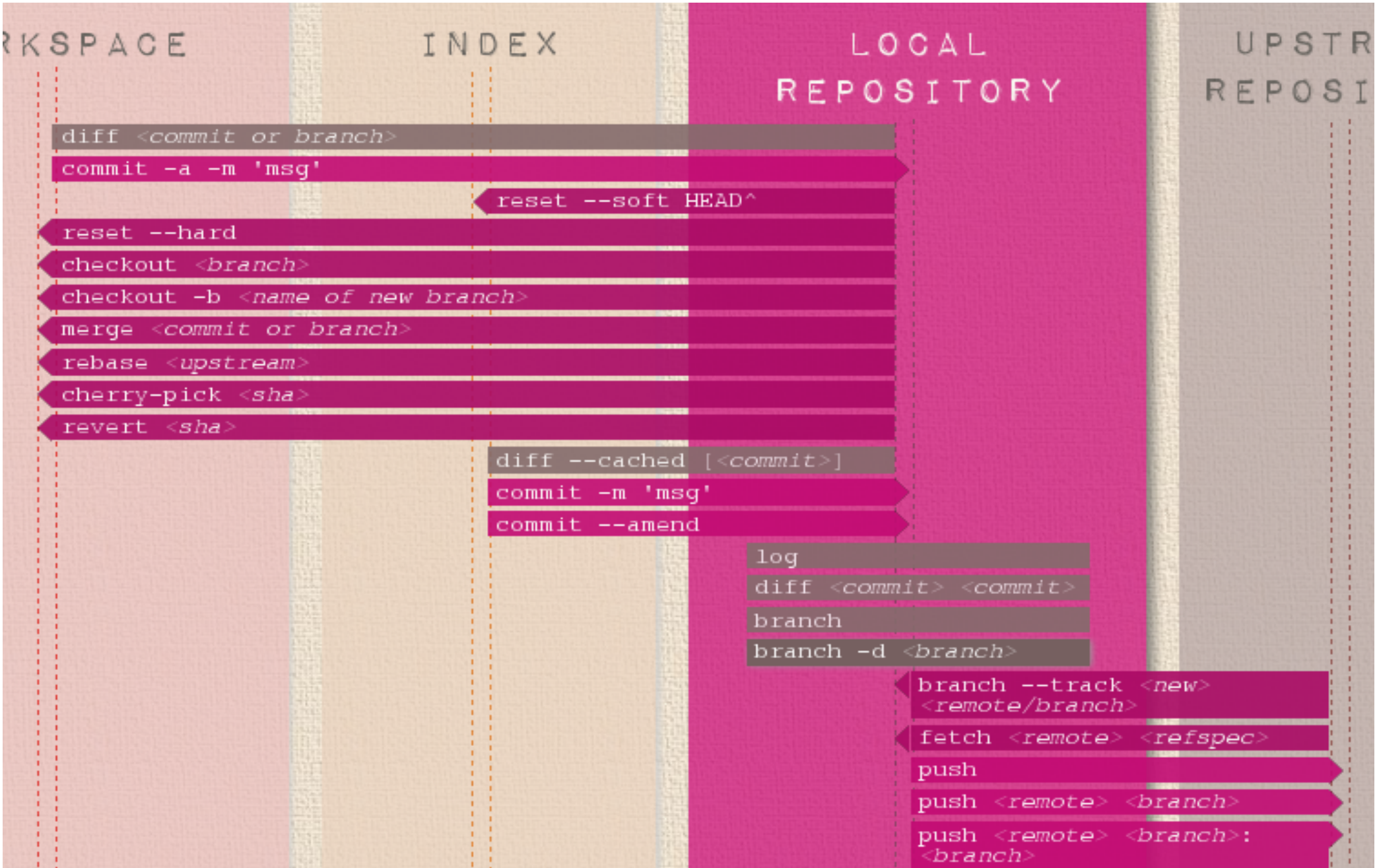
Cheat Sheet Workspace



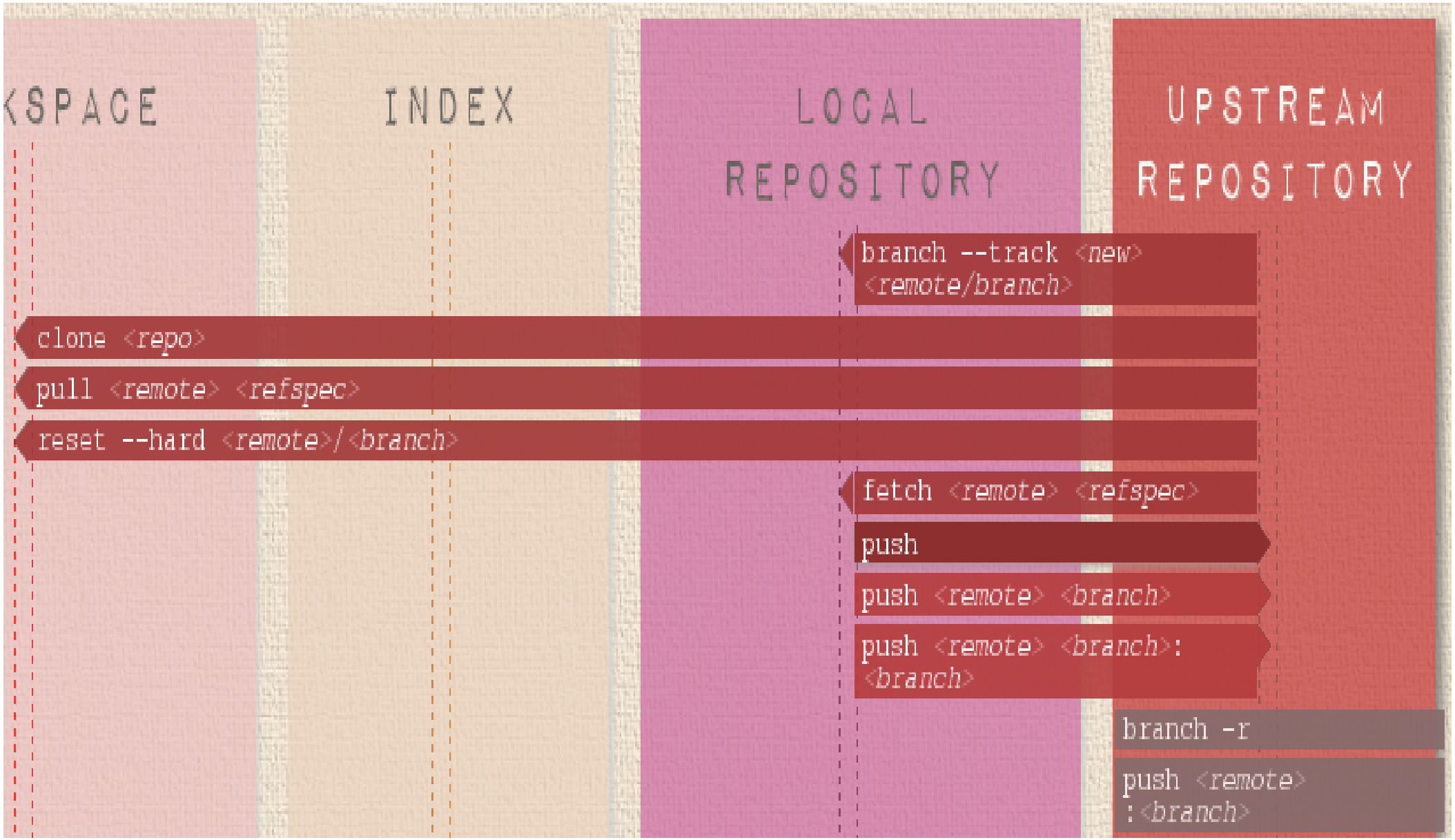
Cheat Sheet Index



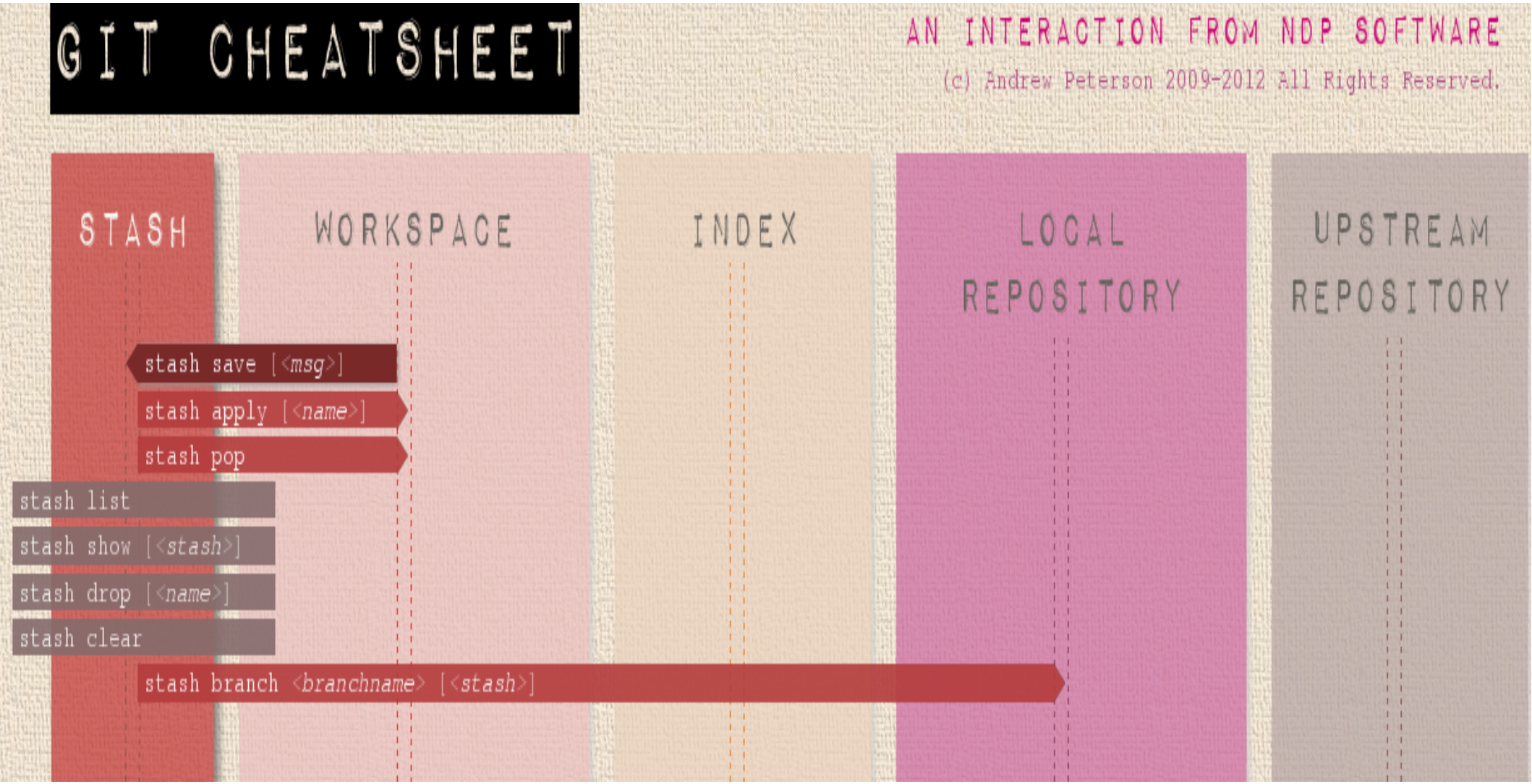
Cheat Sheet Local Repo



Cheat Sheet Upstream Repo



Cheat Sheet Stash



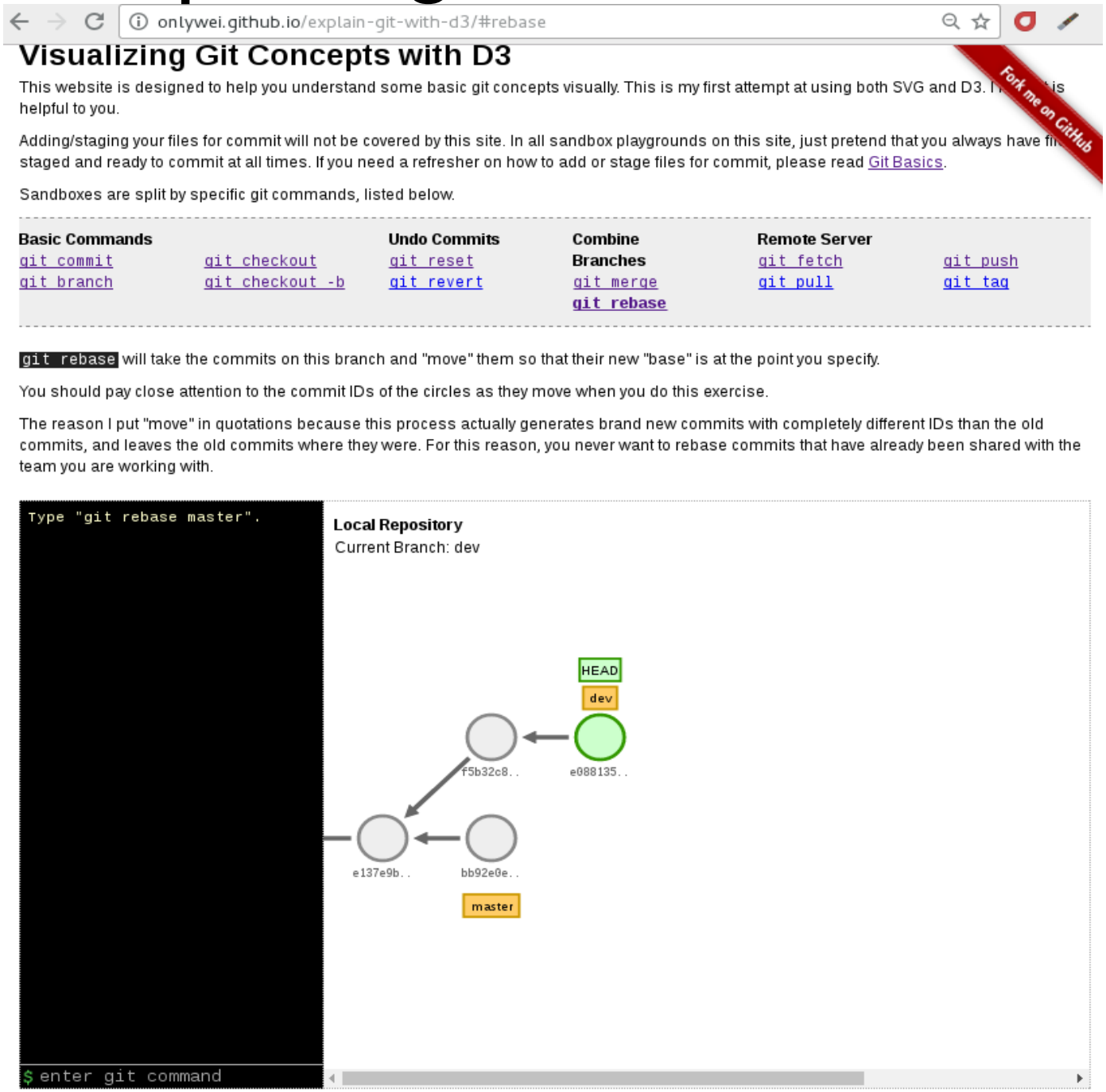
Github explain-git-with-d3

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

Type command interactive

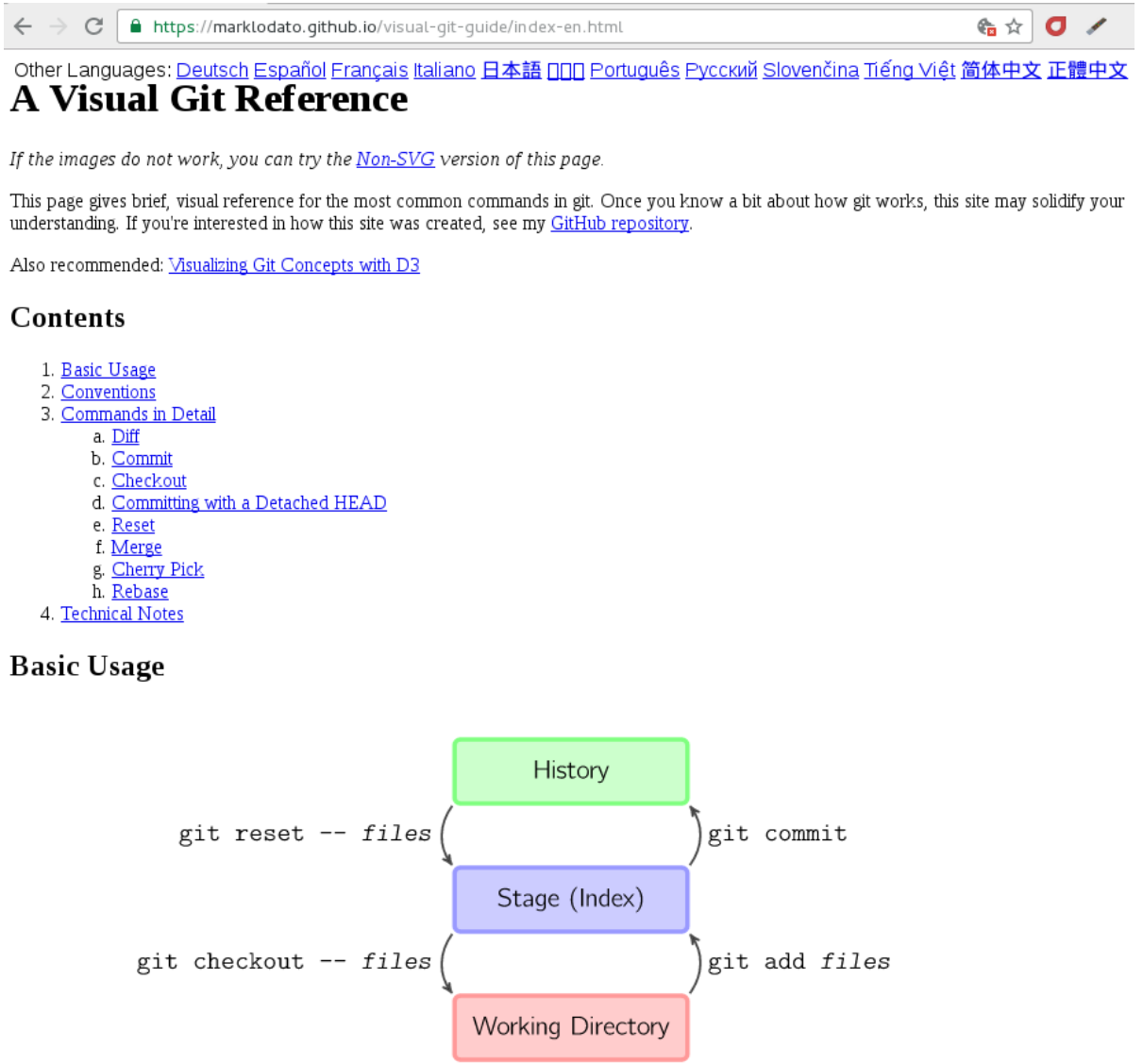
&

See interactive draving moving!!



Visual Git Guide

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



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

Other Languages: Deutsch Español Français Italiano 日本語 □□□ 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.

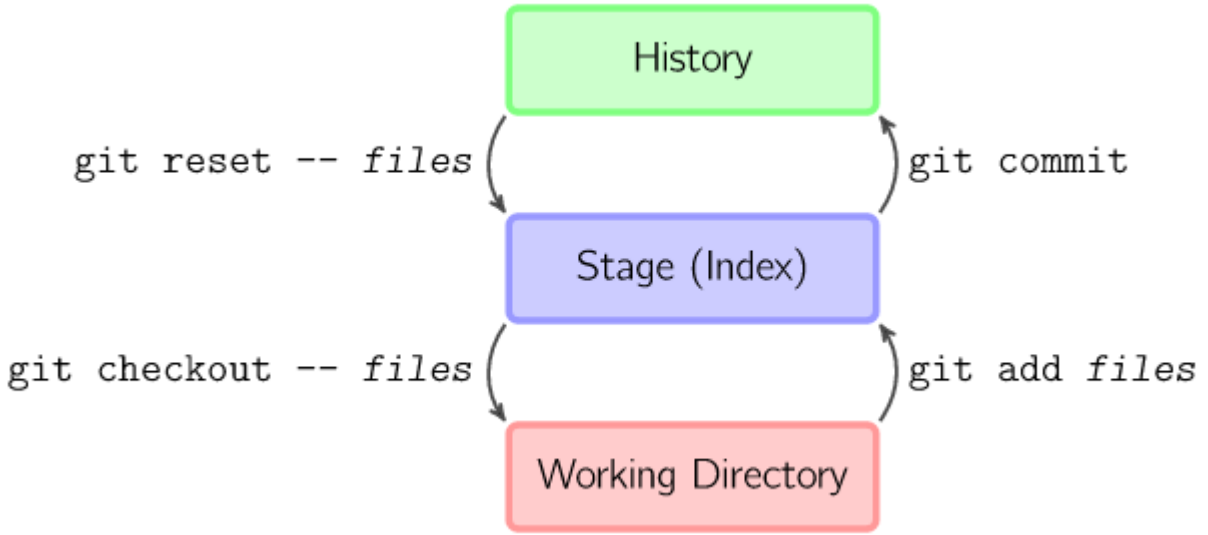
This page gives brief, visual reference for the most common commands in git. Once you know a bit about how git works, this site may solidify your understanding. If you're interested in how this site was created, see my GitHub repository.

Also recommended: Visualizing Git Concepts with D3

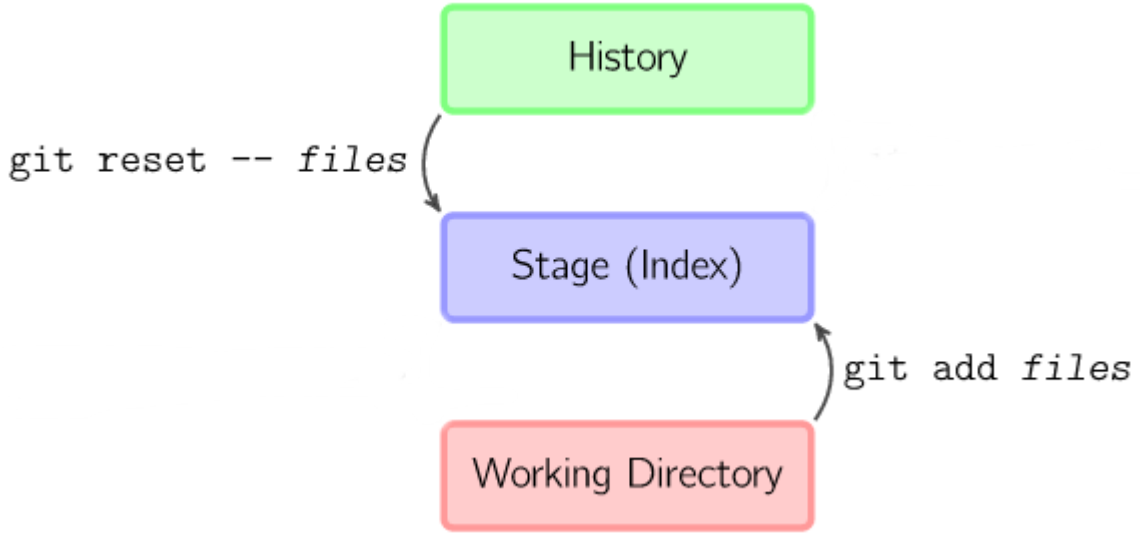
Contents

Basic Usage

Visual Git Guide : Basic Commands

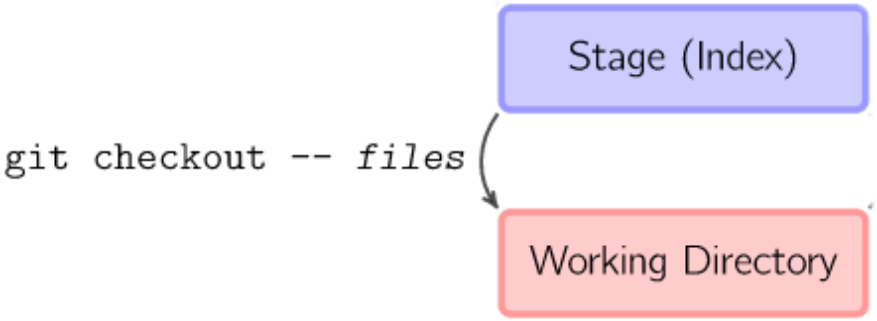


Opposite of “git add” ?
“git reset -- ”… DON'T “git rm”

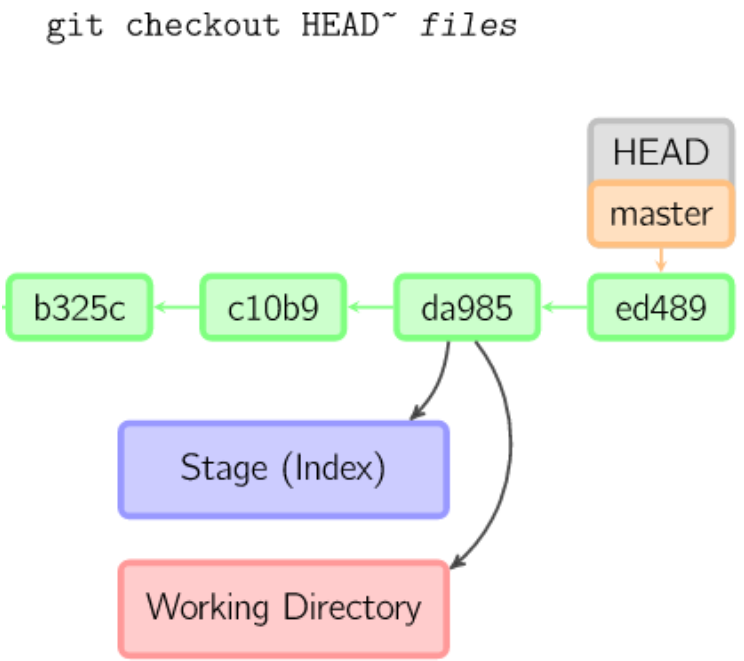


Git Checkout -- file

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

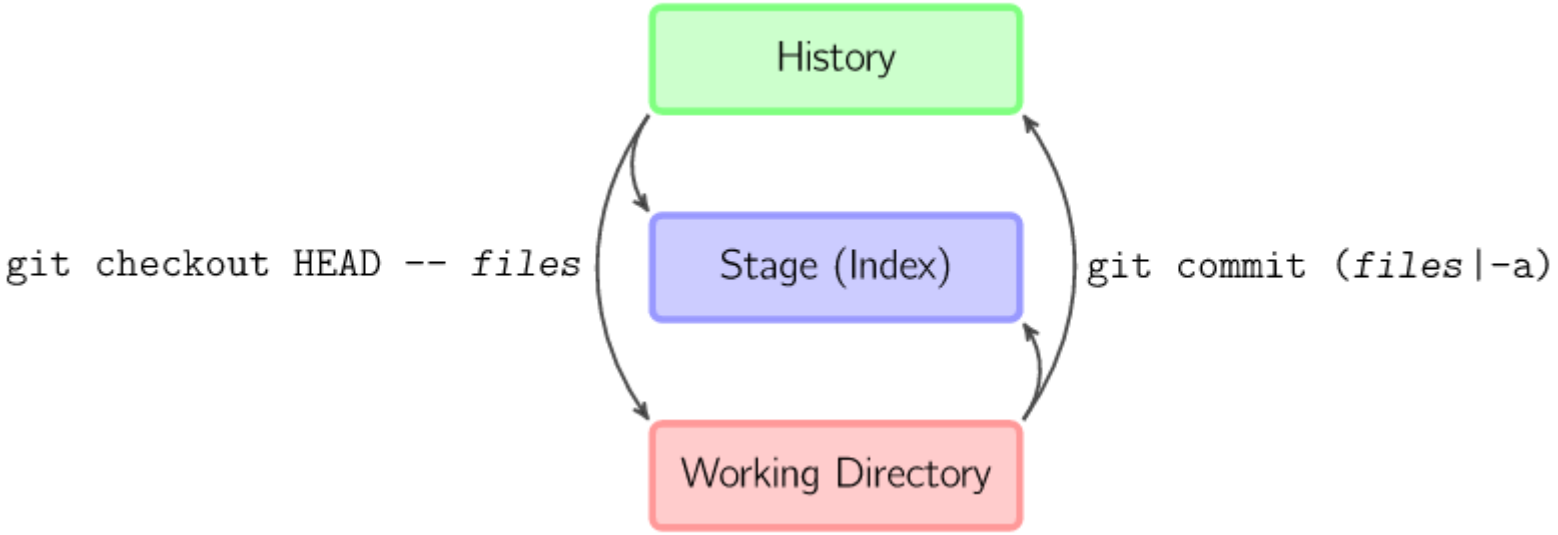


Usefull to **restore file content from history** = as in past commits

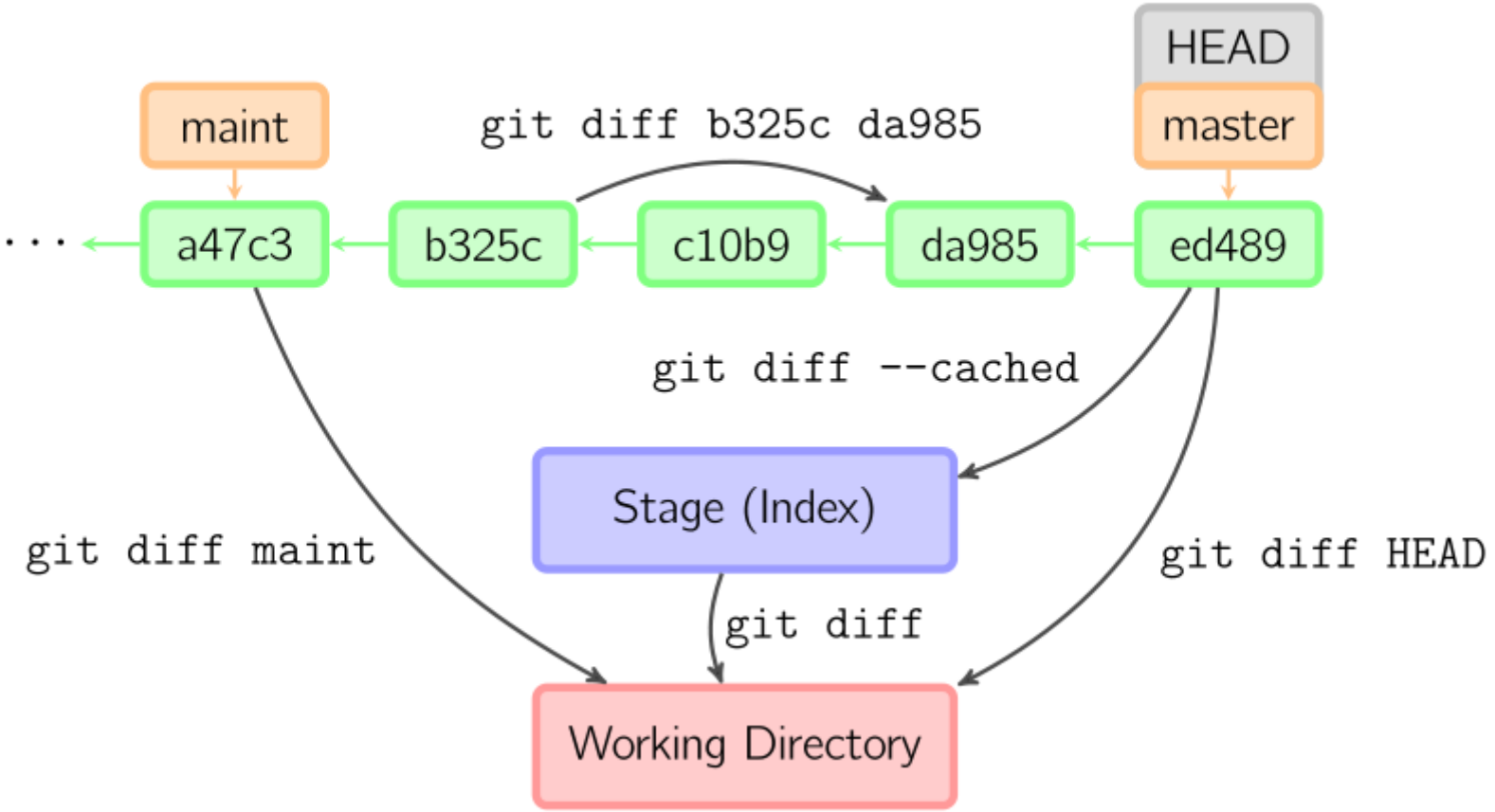


Remarks : use git argument escape “- -“ … image you have a file named dash “-rm”

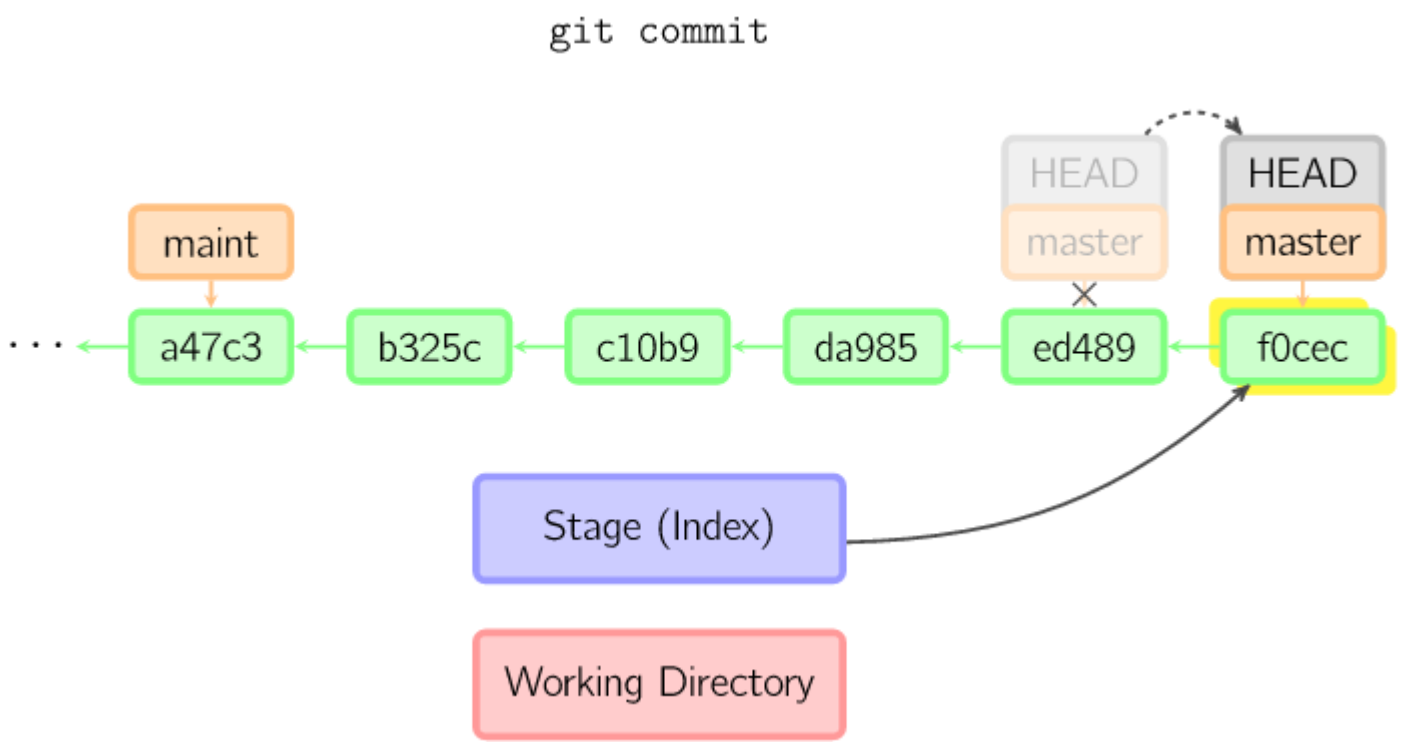
Basic Command ..
Shortcut Repo<->Workspace



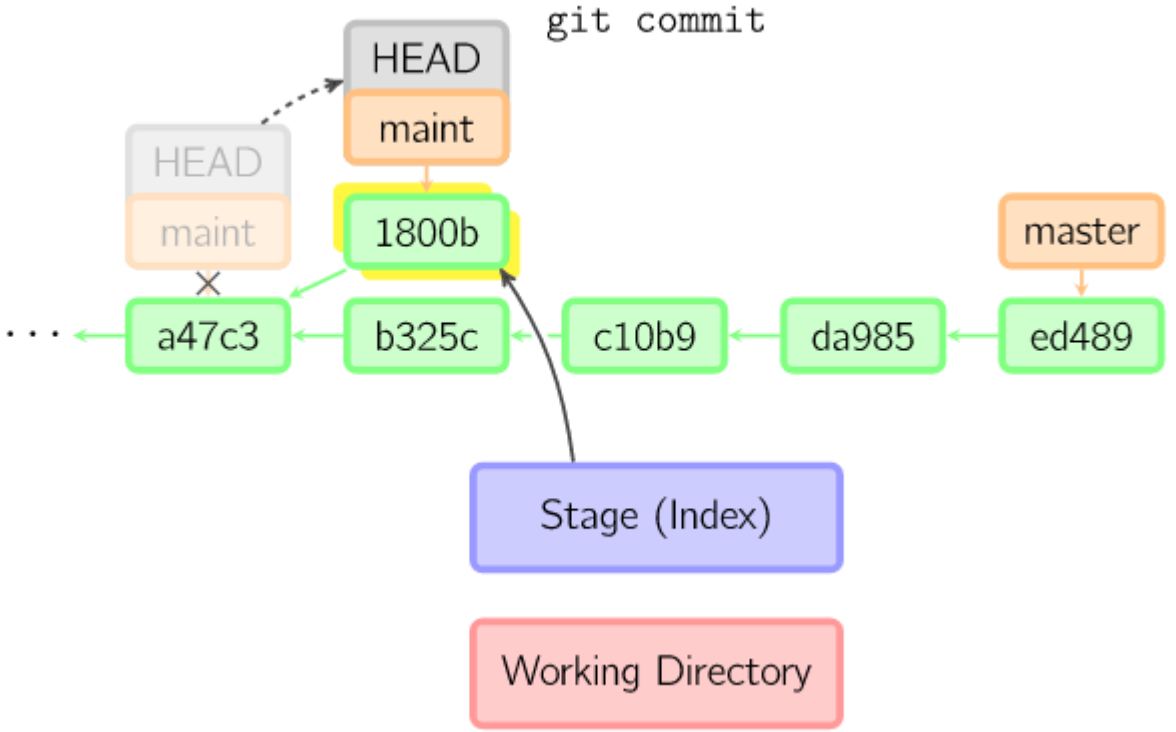
Git Diff


HEAD
maint
git diff b325c da985
master
...
a47c3
b325c
c10b9
da985
ed489
git diff --cached
Stage (Index)
git diff maint
git diff HEAD
git diff
Working Directory

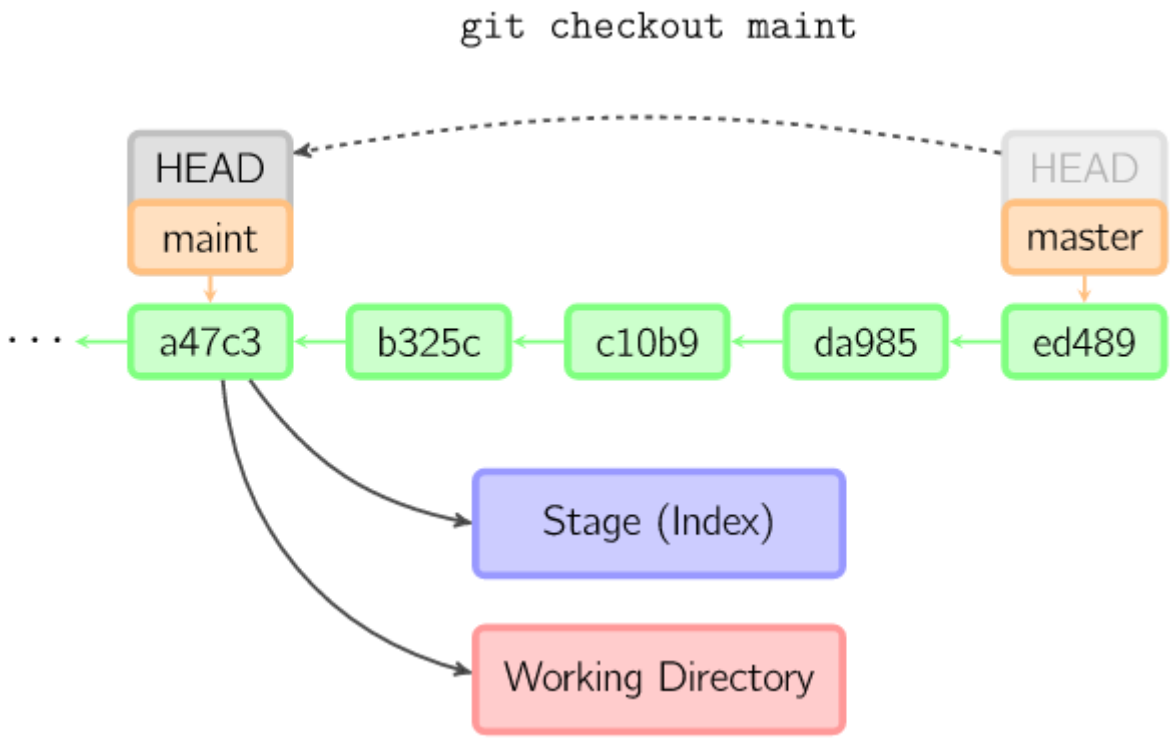
Git Commit
(where HEAD=master)



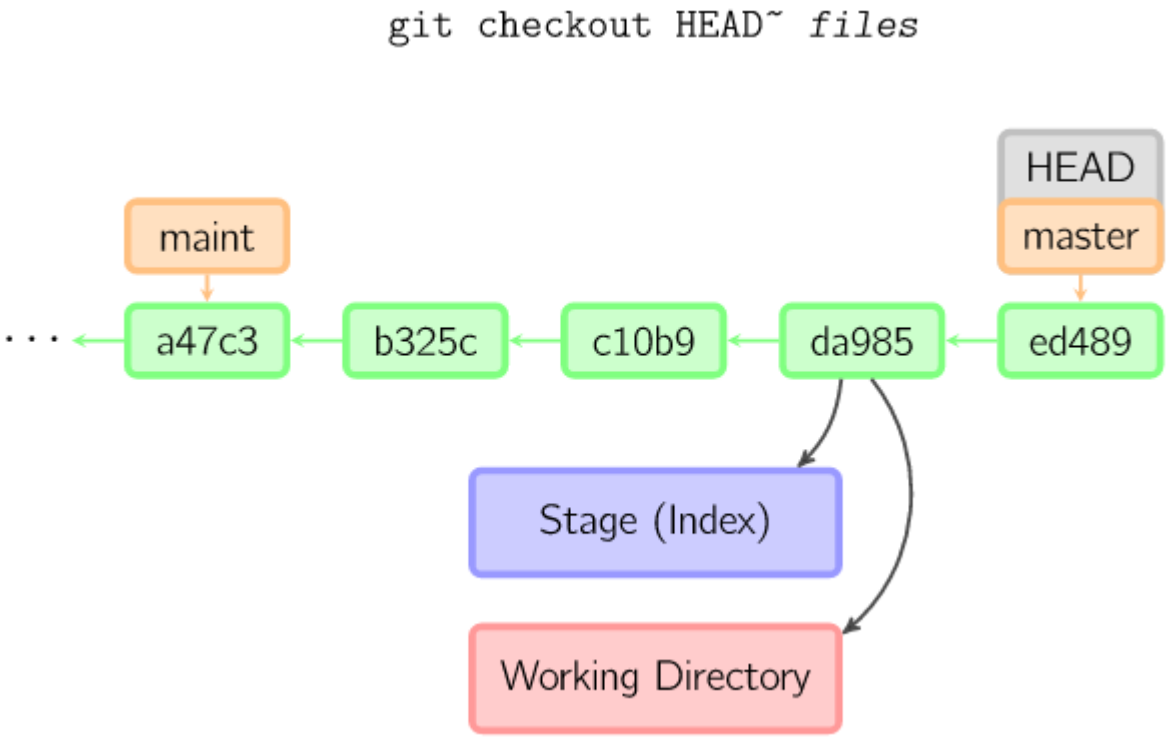
Git Commit
where HEAD = branch



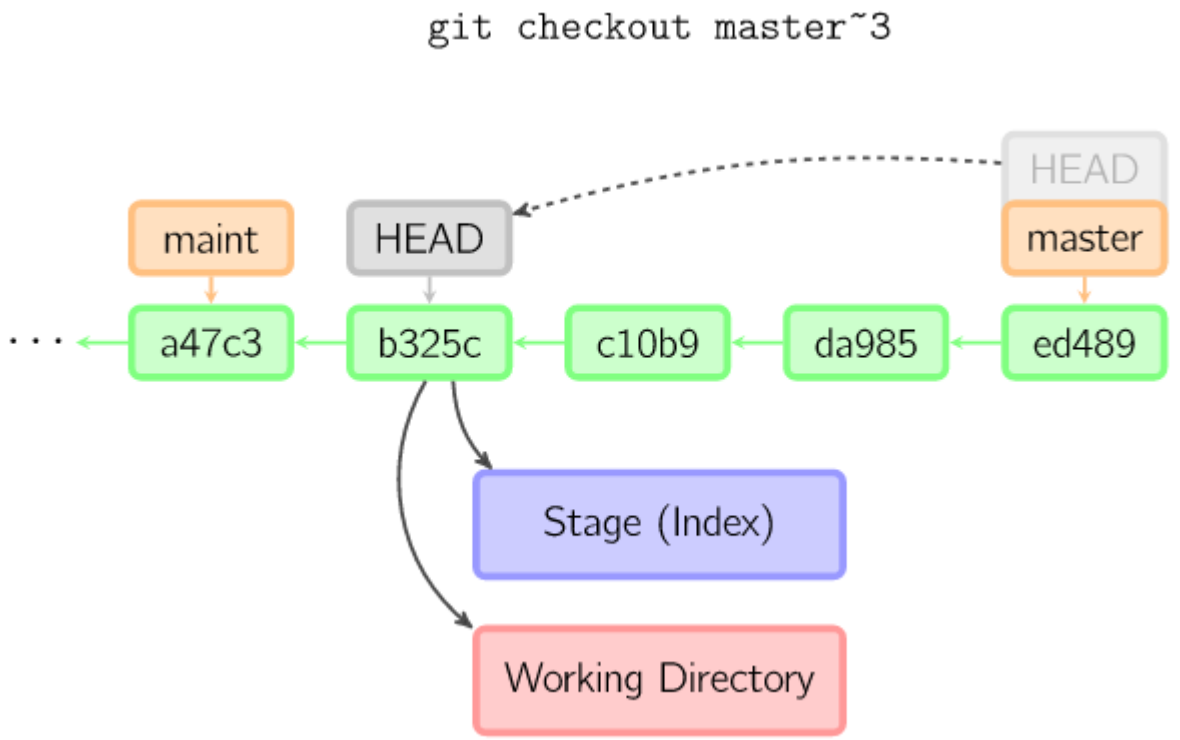
Checkout (1/3): git co branch (=change current branch)



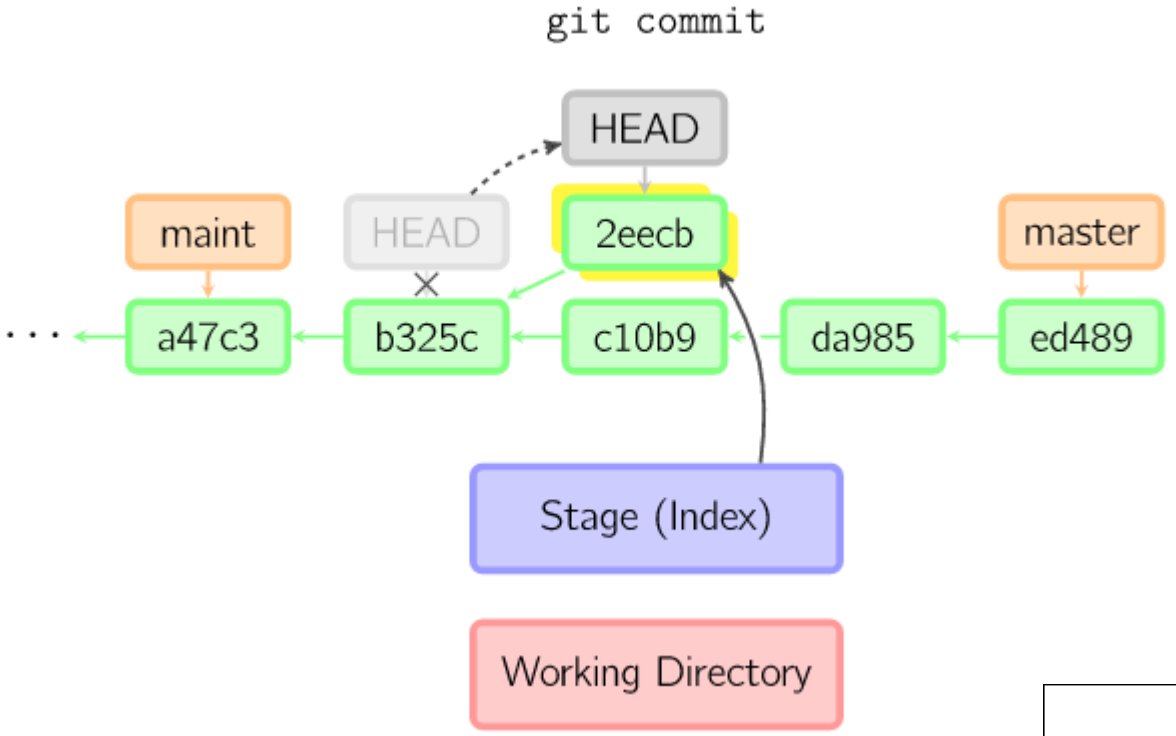
Checkout (2/3) : git co -- file1 file2 = restore a file



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

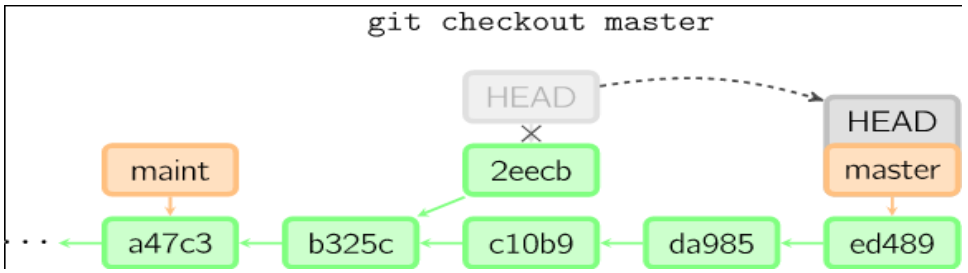


Detached HEAD ??



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

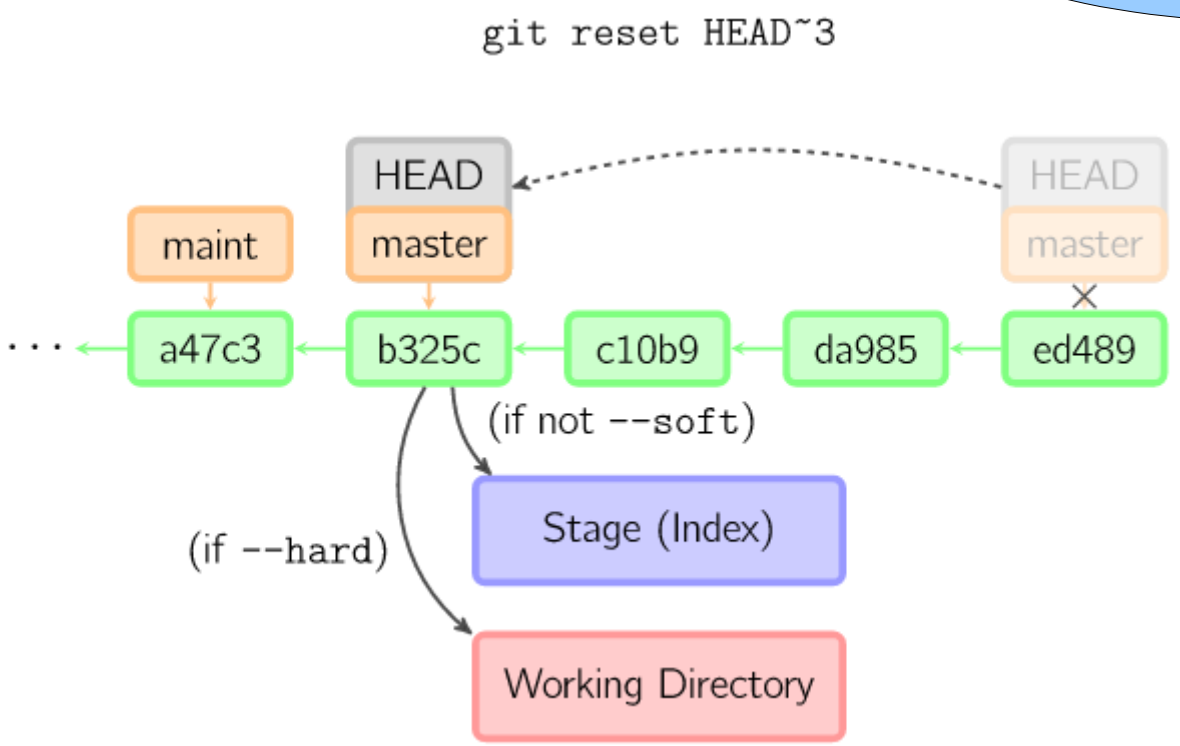
RISK... If change to branch or master..
$ git co master



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

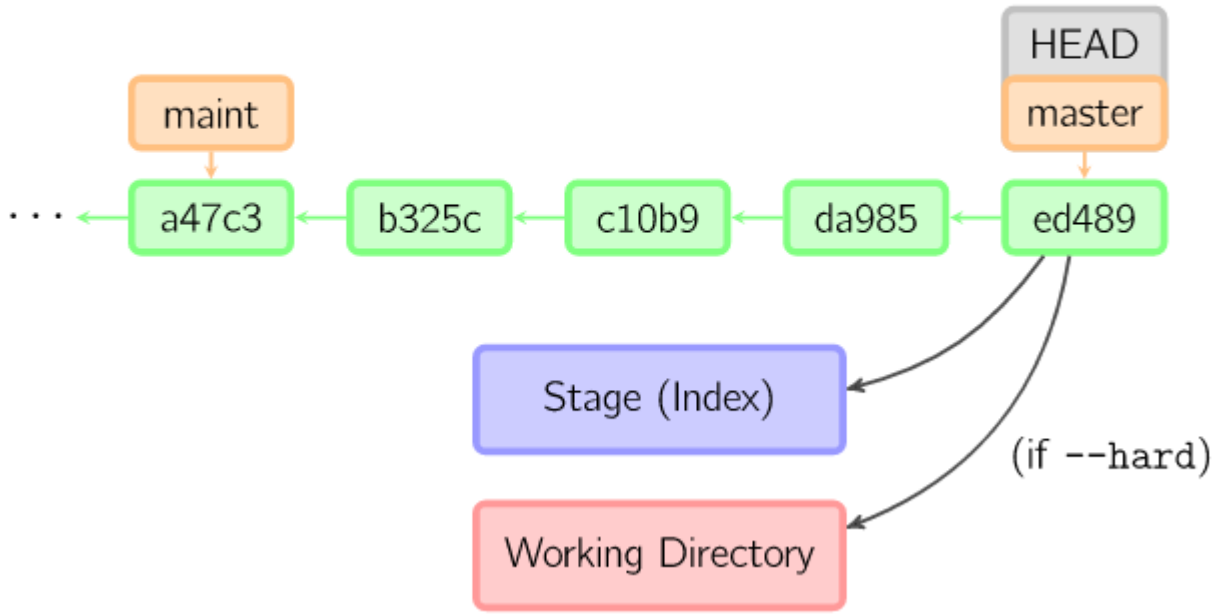
Git Reset ~~~
… another way to
“Shoot yourself in the FOOT”

… Like Detached “**HEAD**” ?



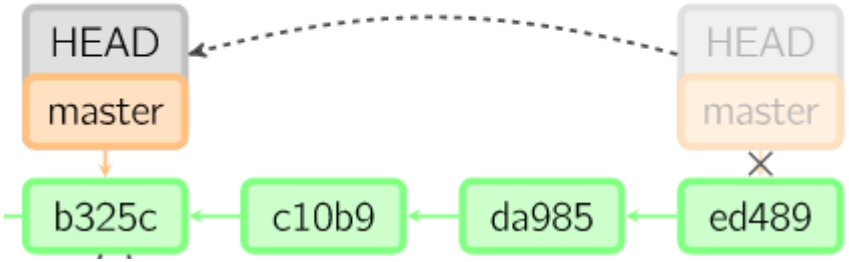
Git Reset
= implicit : Git reset HEAD

… reset nothing = OK
restore “**HEAD**” content



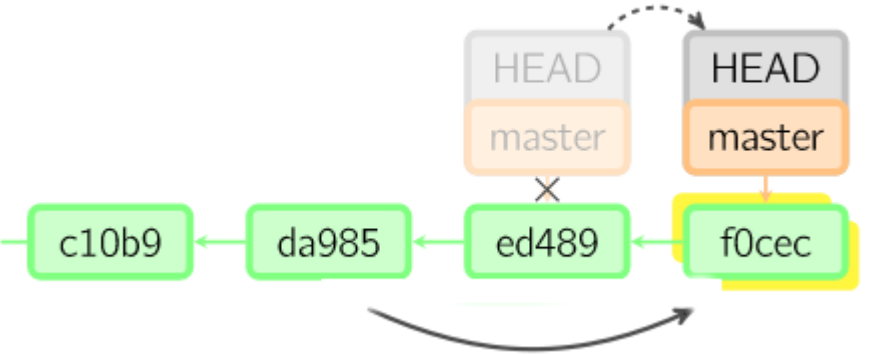
reset head~ … vs revert head~

git reset HEAD~3



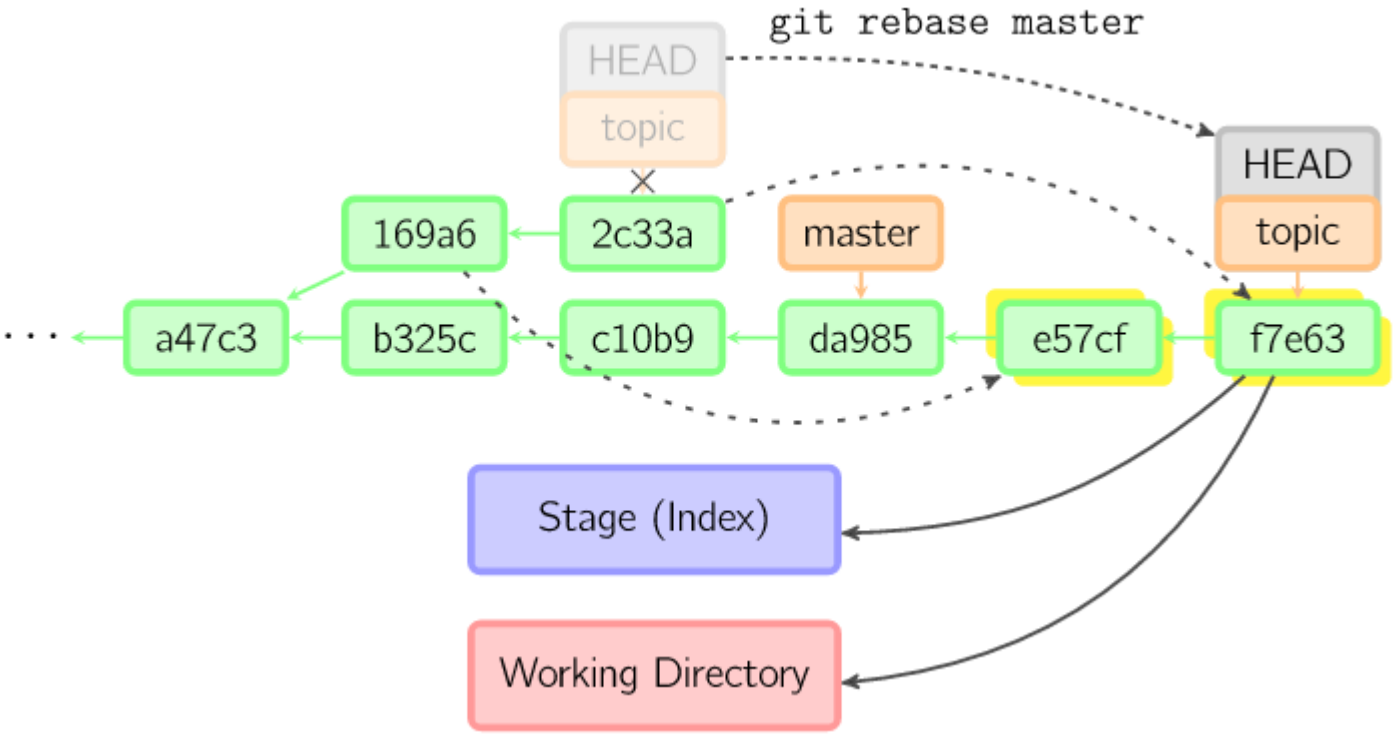
Reset = undo content & undo commits

git revert HEAD~



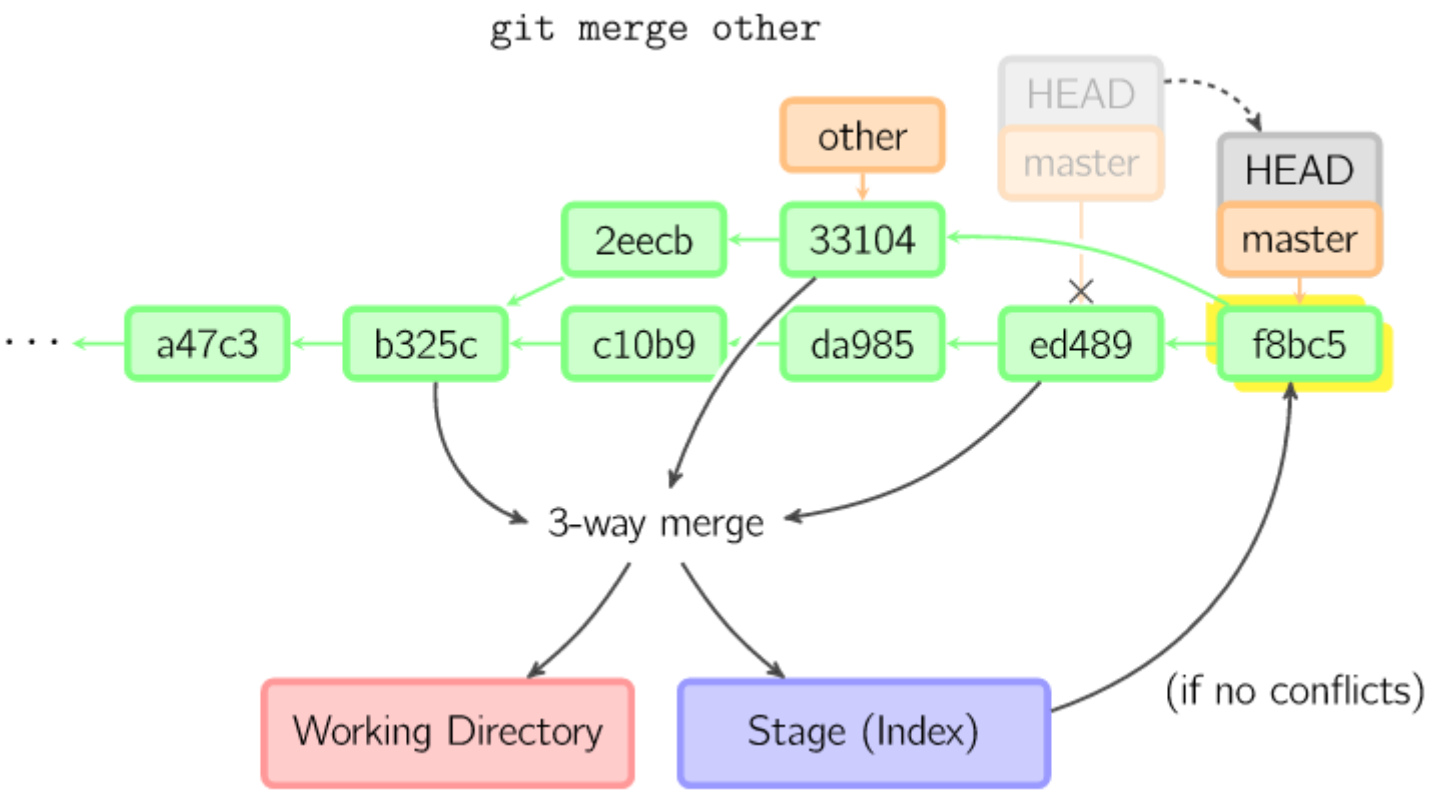
Revert = undo content
BY APPLYING inverse of delta
then commit forward
(no brain wash history, only overwrite to fix it)

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

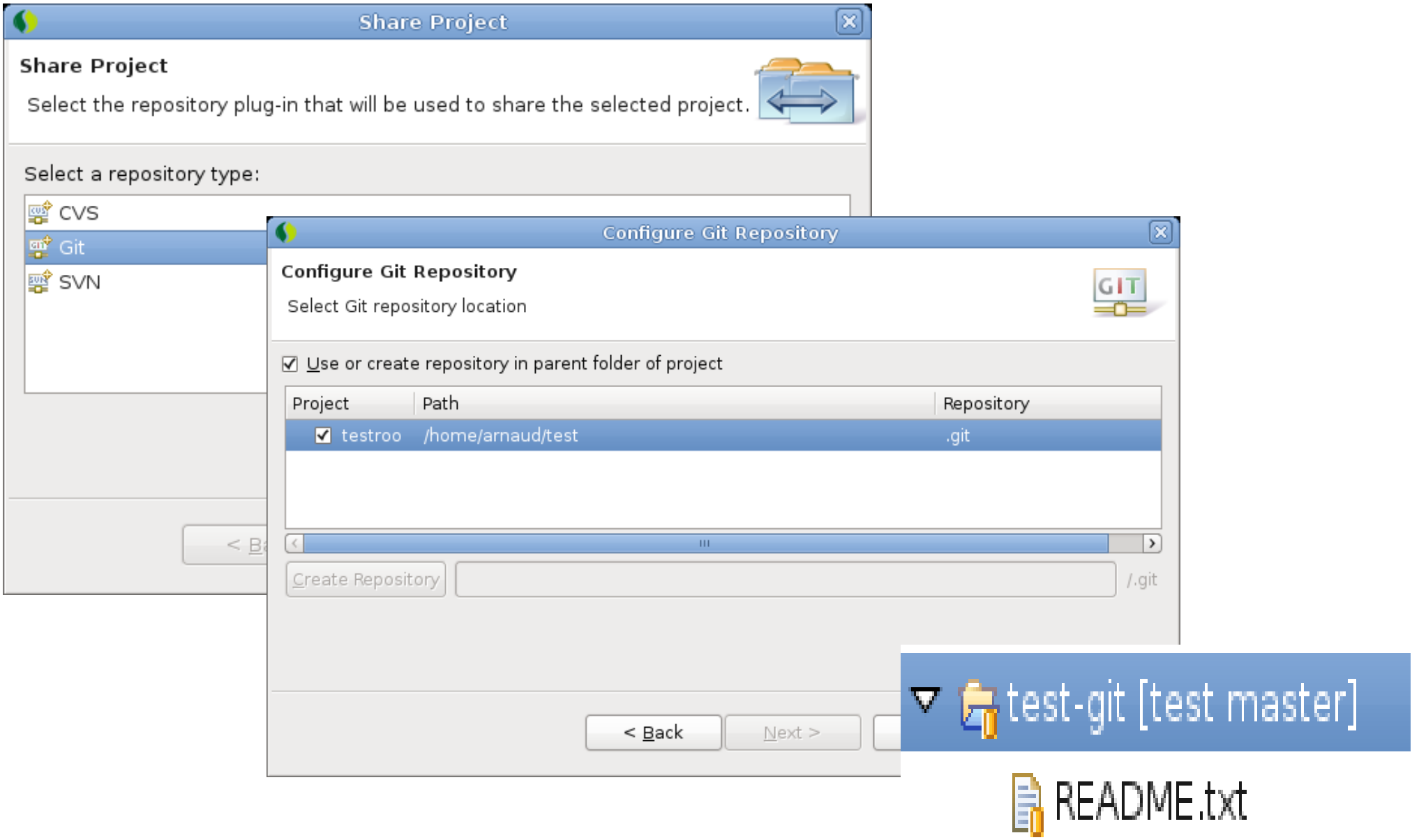
merge


git merge other
other
HEAD
master
HEAD
master
2eecb
33104
a47c3
b325c
c10b9
da985
ed489
f8bc5
3-way merge
Working Directory
Stage (Index)
(if no conflicts)

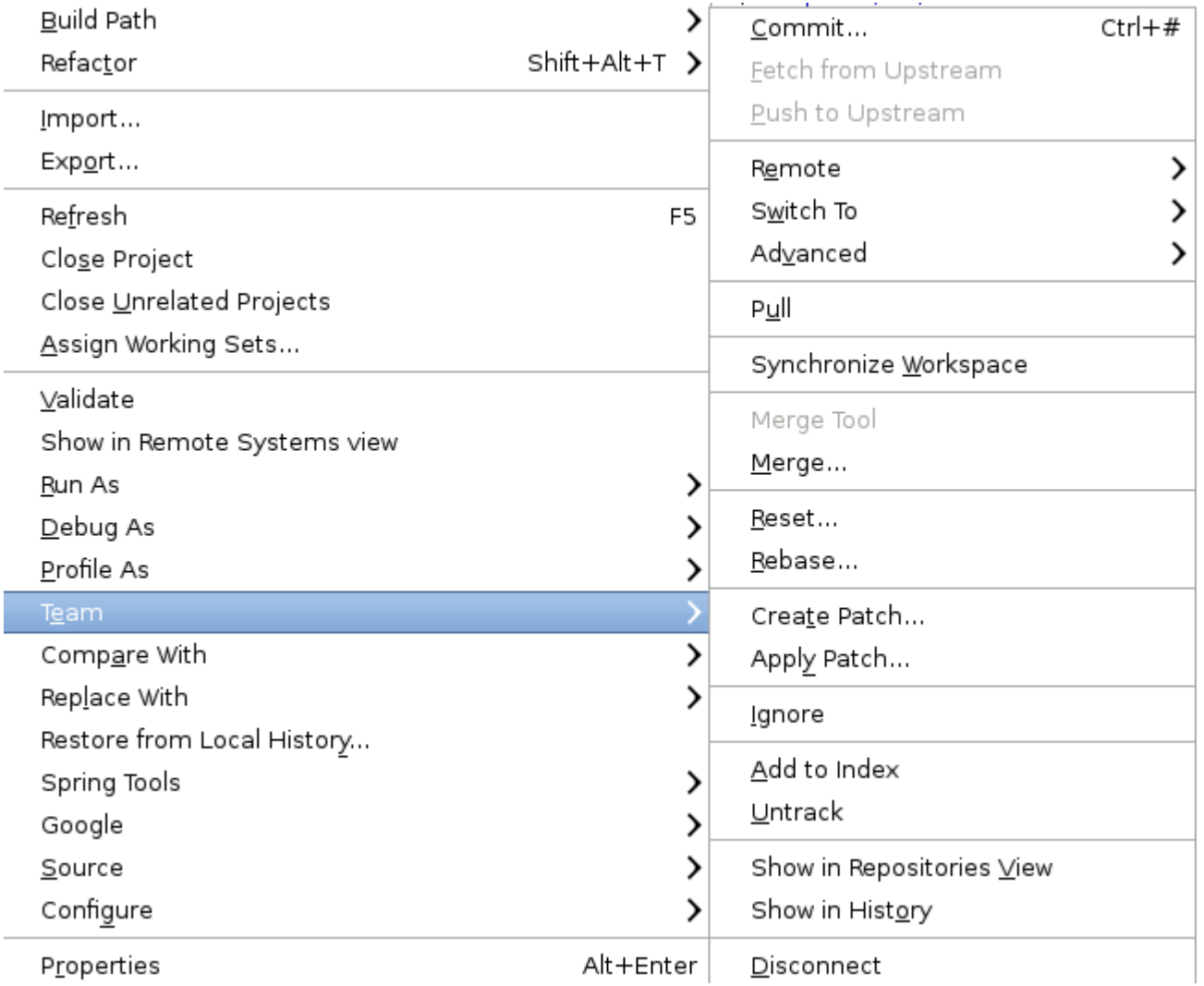
Git Tools

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

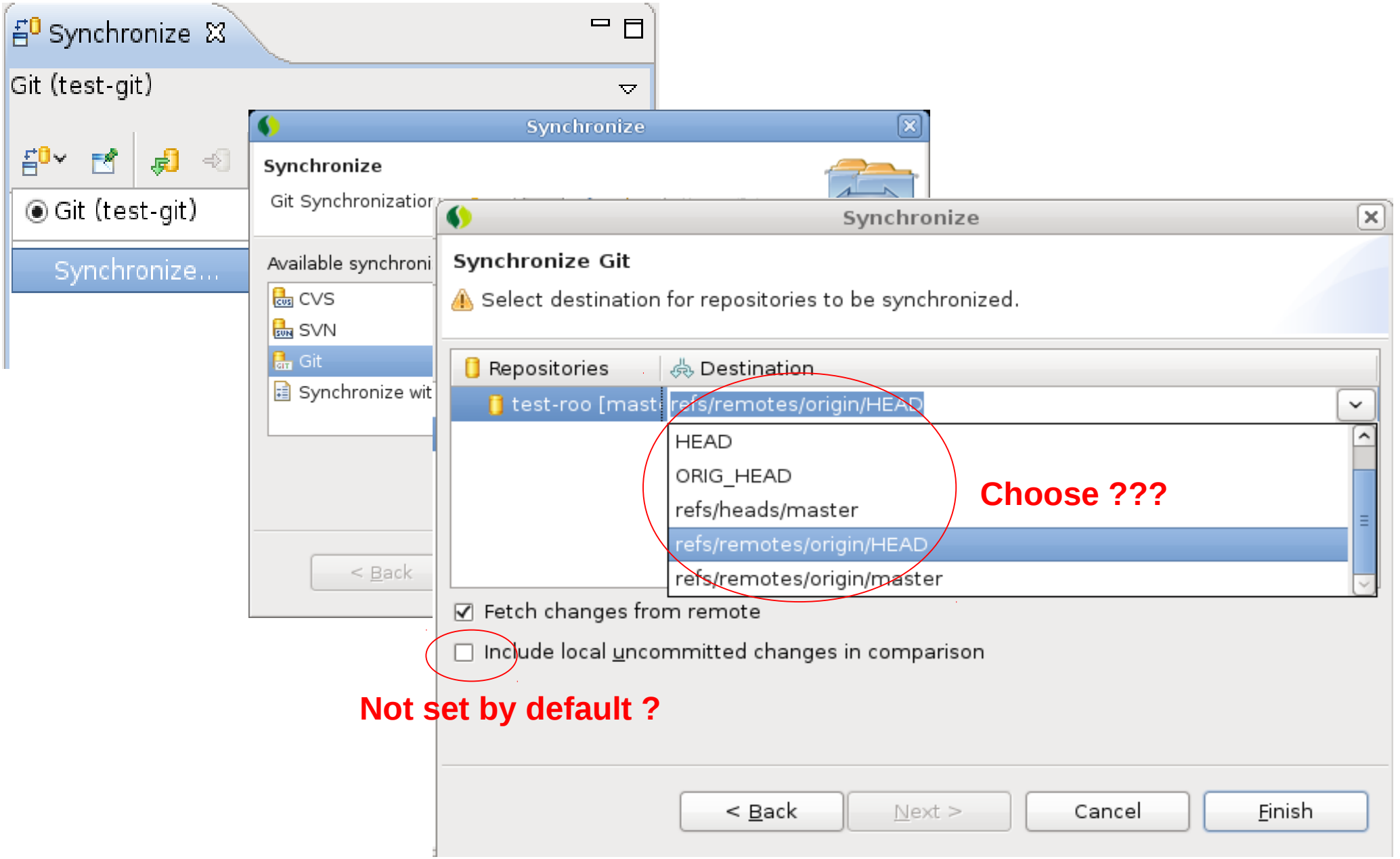
Egit : Git in Eclipse



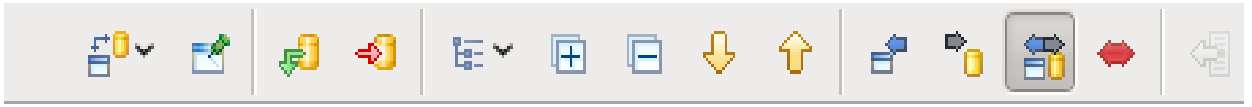
Eclipse Team Menu / Perspective / History View ...



Eclipse eGit Synchronize View

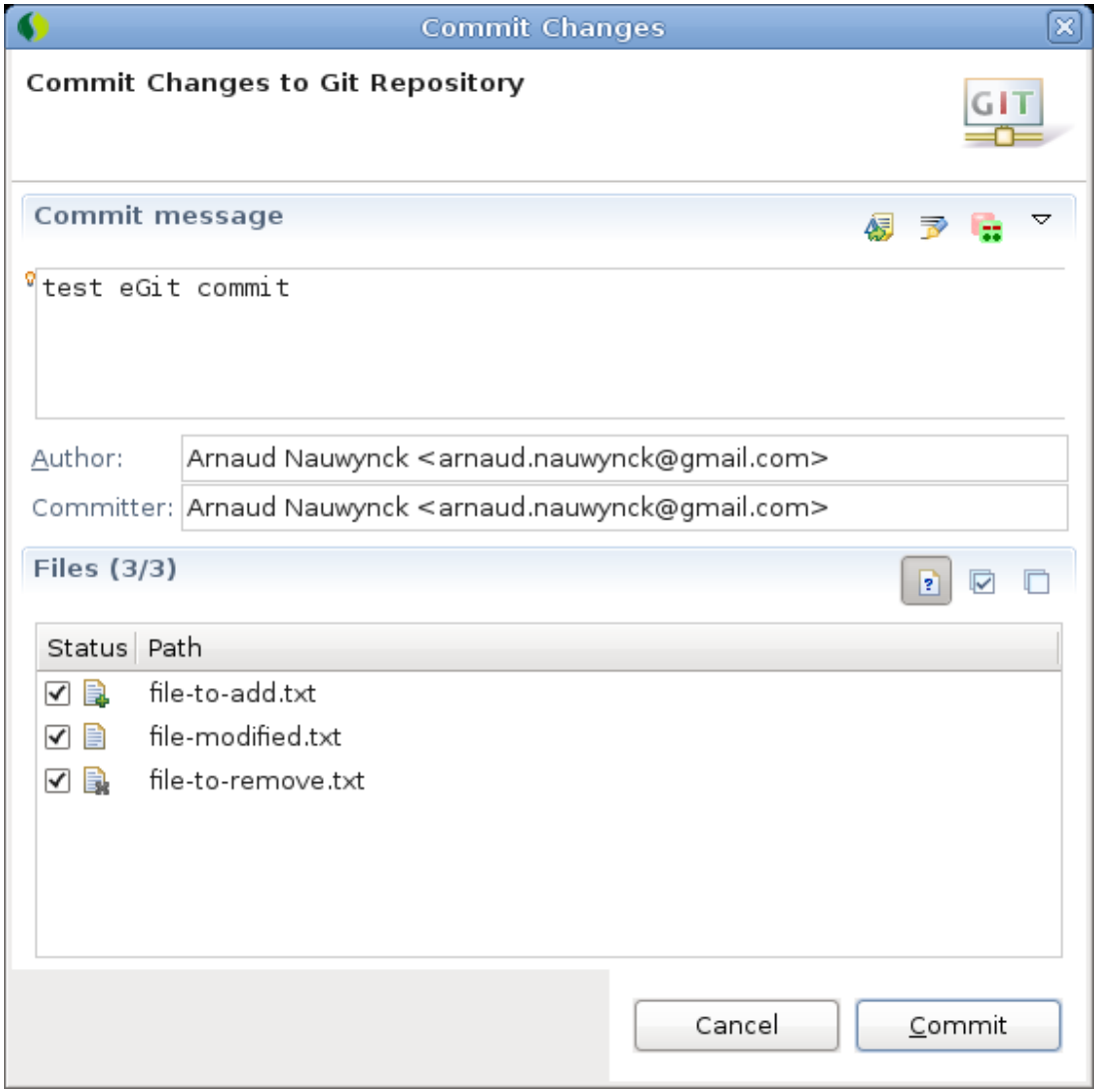


eGit Commit

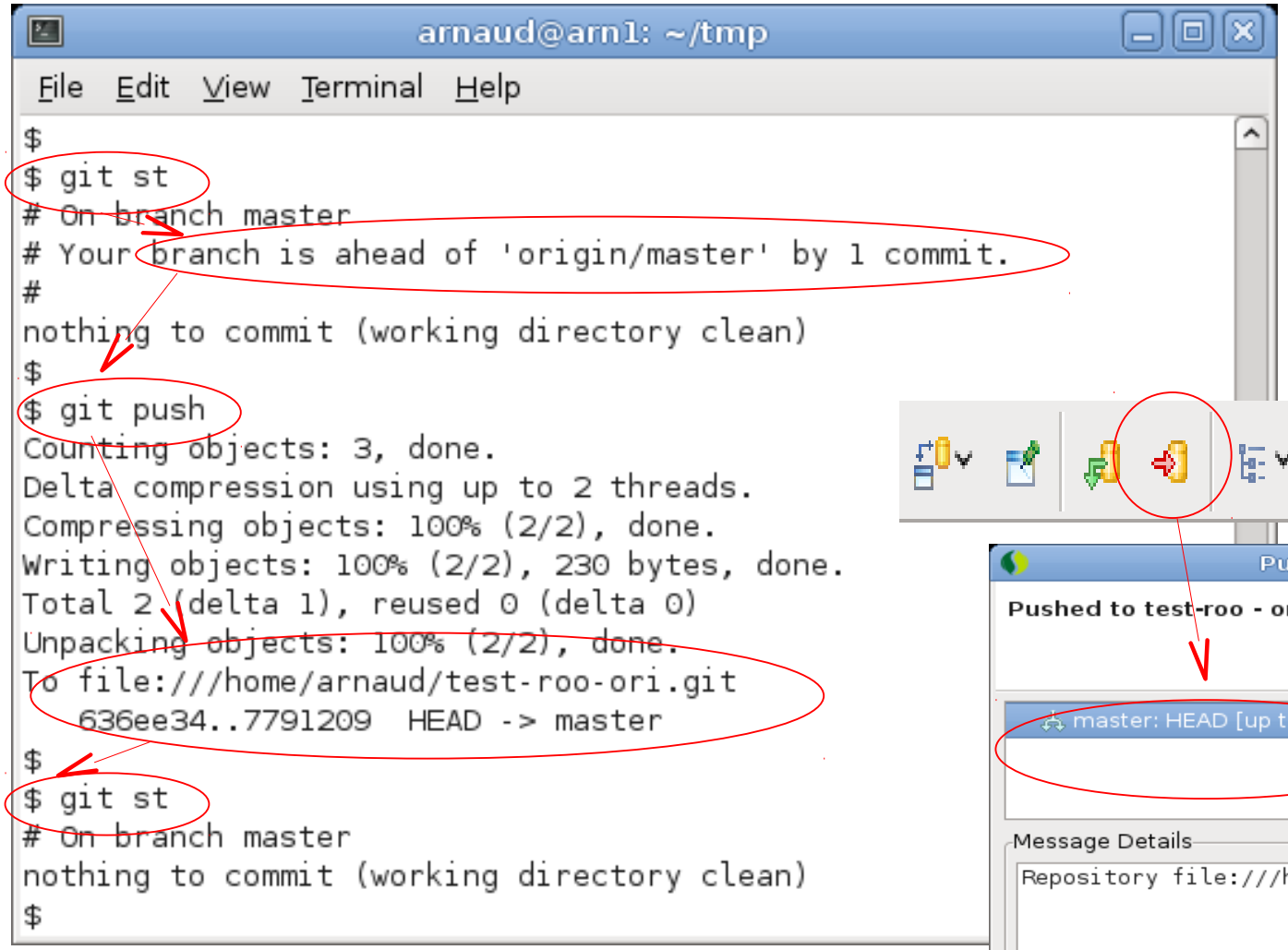


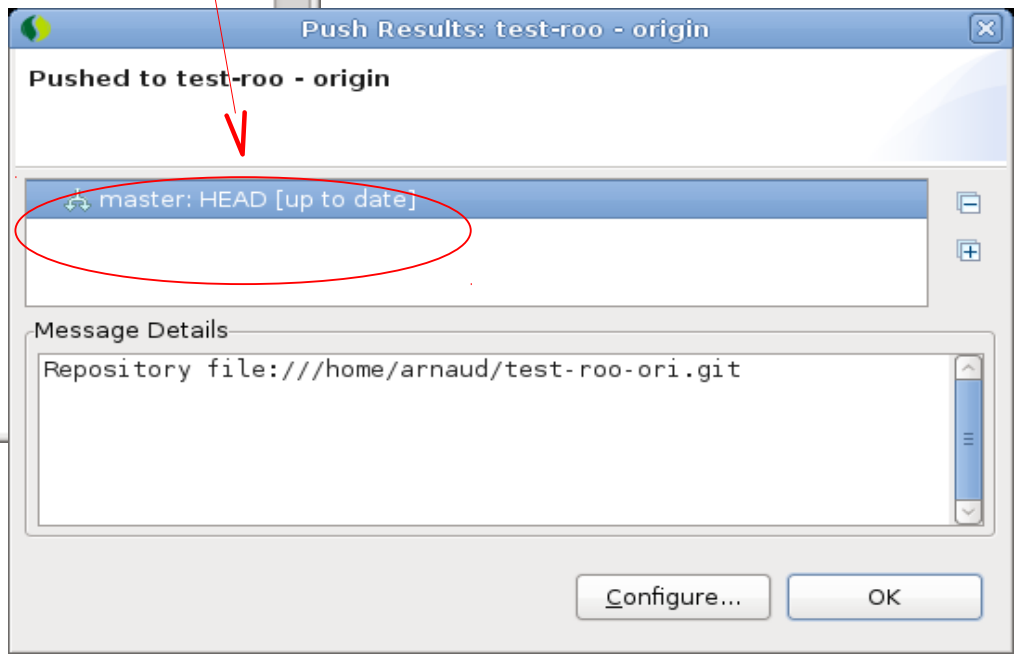
testroo [test-roo master ↑2]

- file-modified.txt
- file-to-add.txt
- file-to-remove.txt

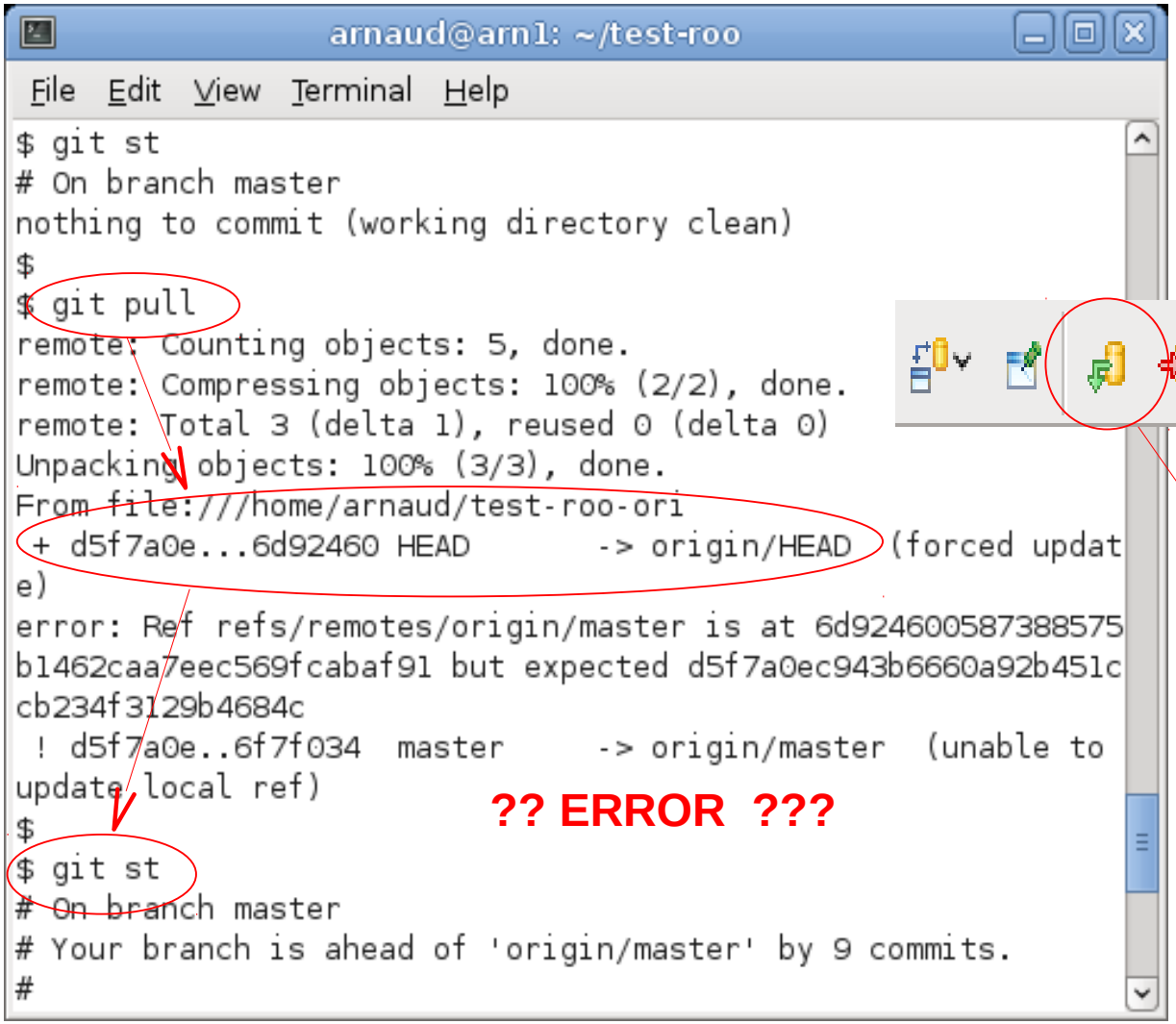


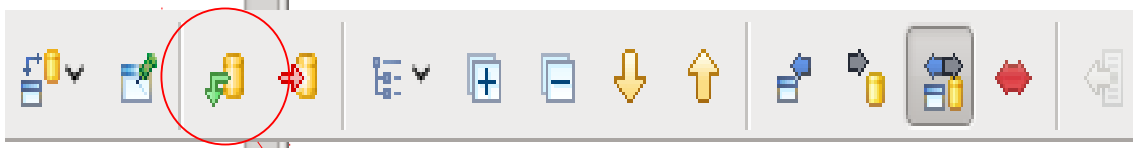
Git Push


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)
$


Push Results: test-roo - origin
Pushed to test-roo - origin
master: HEAD [up to date]
Message Details
Repository file:///home/arnaud/test-roo-ori.git
Configure...
OK

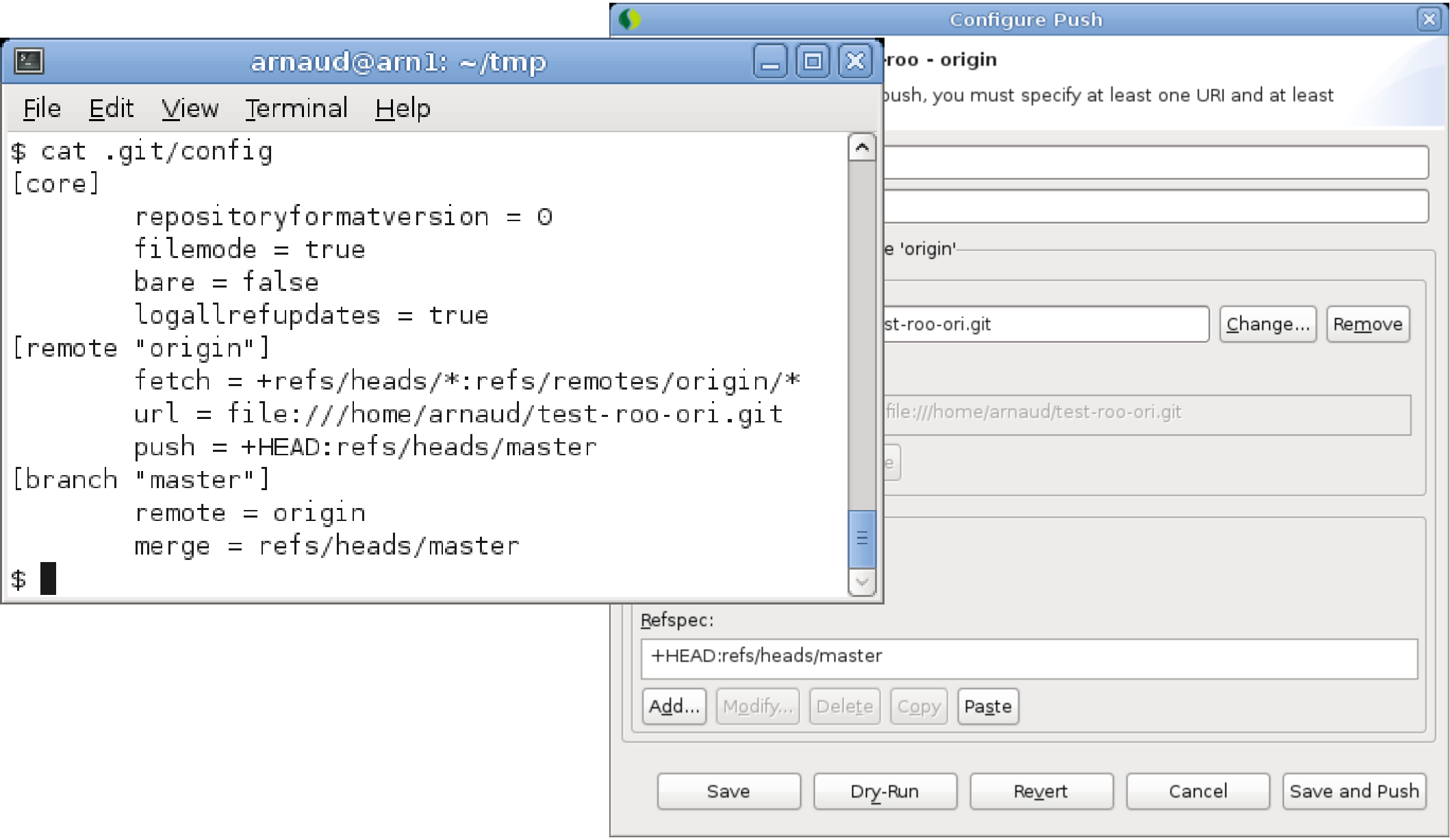
Git Pull (=Fetch+Merge)





SILENT UPDATE in ECLIPSE??

Egit Fetch/Push Config



Git / eGit …

- Errors are often difficult to Understand
- IMPORTANT to understant **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