Introduction à git

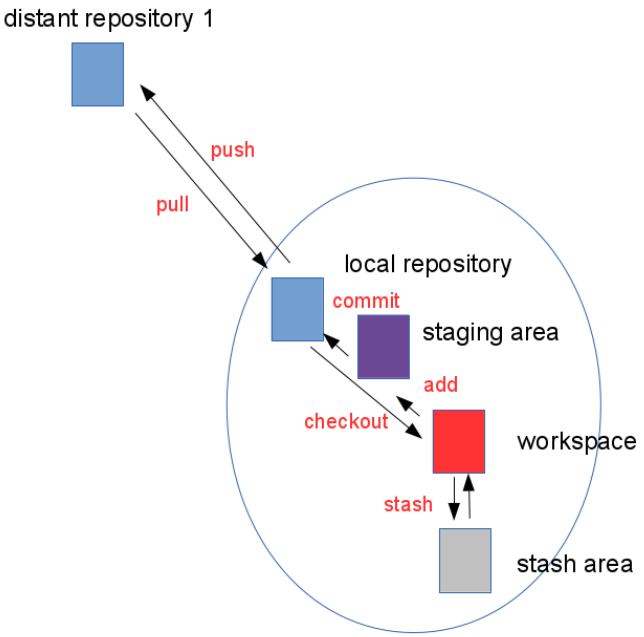
Nathalie et Marie-Jo, 10 juillet 2013

Objectif
Connaître l'environnement
Pratiquer une utilisation de base

Prérequis
Utiliser un gestionnaire de version
Amener son portable avec git installé

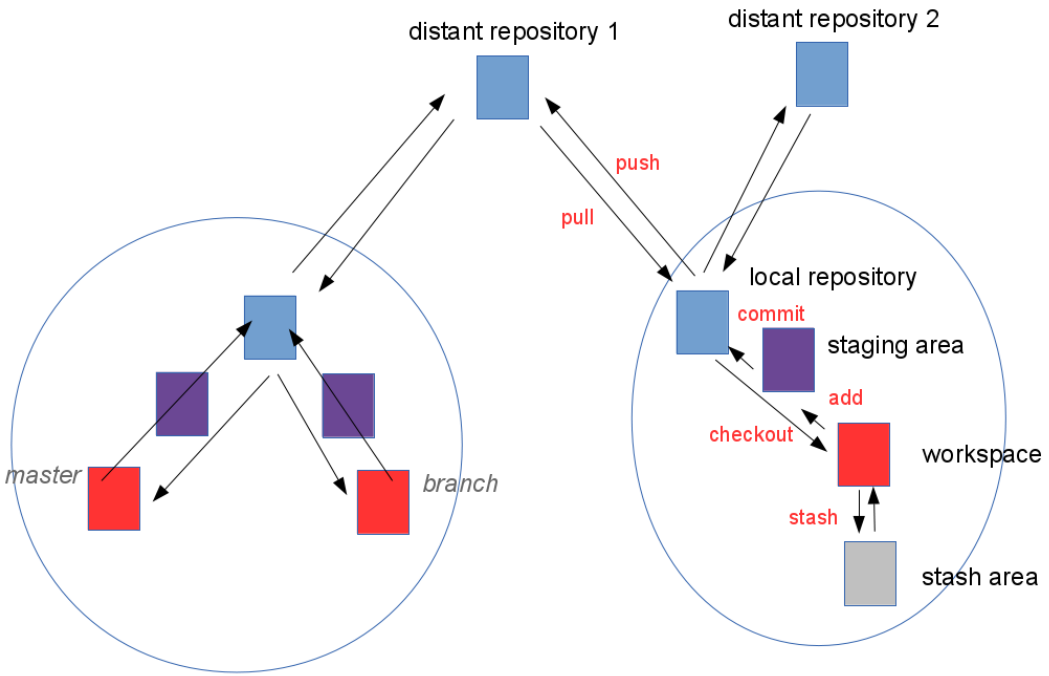
INTRODUCTION

Logiciel de gestion décentralisée de version
DSCM (Distributed Source Code Management)
→ *chaque version locale de travail a son repository*



INTRODUCTION

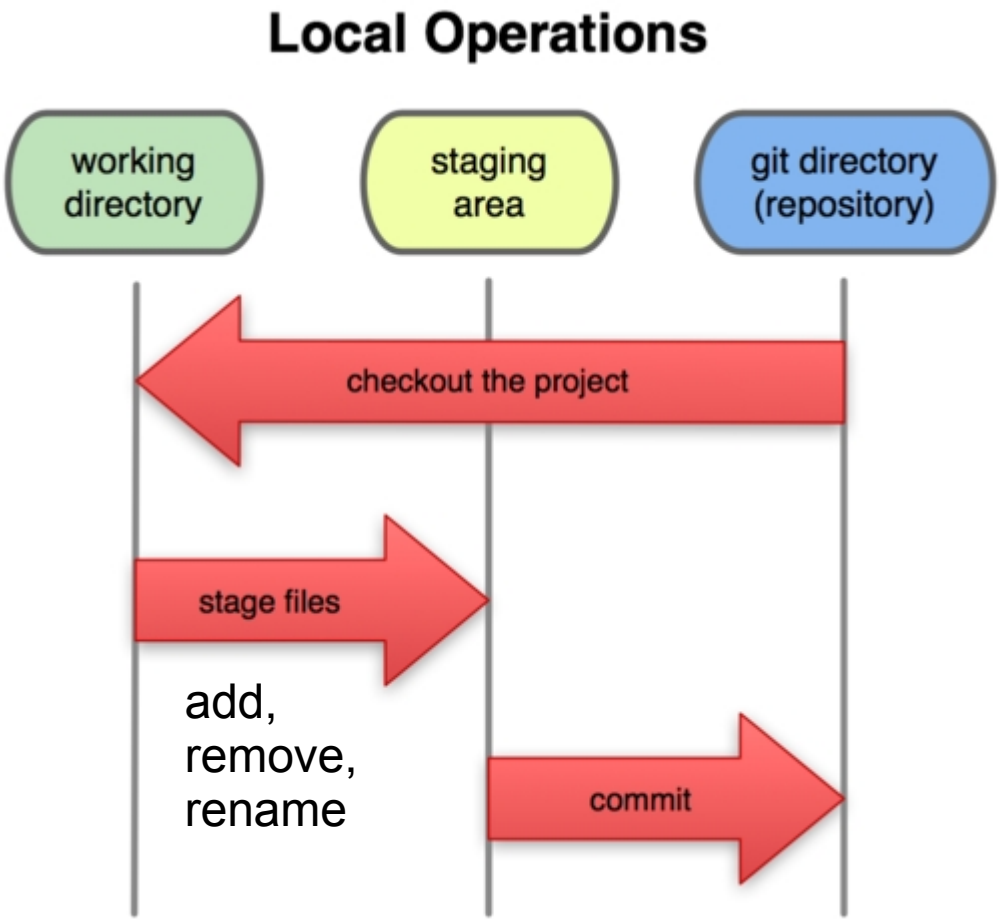
Logiciel de gestion décentralisée de version
DSCM (distributed Source Code Management)
→ *chaque version locale de travail a son repository*



Développé en 2005 par Linus Torvald pour le noyau Linux

3 zones locales

GIT gère 3 zones : le répertoire Git, le répertoire de travail et la zone d'index.



La zone index permet de préparer progressivement un commit.

INTRODUCTION

Devient le standard actuel

CVS	SVN	GIT
centralized	centralised	Decentralized
Name based	Release based	Hash based
	Limited merge capabilities	Designed for merge
	Bad ancestry (each commit increase version number)	Steep learning curve

Sous Ubuntu, installer le meta-paquet git-all

INTRODUCTION

Pourquoi cette difficulté d'apprentissage alors même que l'on connaît SVN ?

- une difficulté supplémentaire à gérer avec la gestion de différents repositories
- Une zone nouvelle à gérer localement : 'staging area'
- des commandes avec des noms pas très explicites (exemple checkout = switch, add pour ajouter un fichier ou résoudre un conflit, reset …)
- l'impression que le retour en arrière n'est pas si facile

➢ amène à appréhender différemment le changement :
passer d'une vision chronologique à une vision topographique
(graphe de commits manipulable)
« You'll know you have reach a Zen plateau of branching wisdom when your mind contains only commit topologies »

➢ amène aussi à changer sa façon de travailler :
- faire de petits commits que l'on peut modifier et qui génèrent des conflits plus simples à gérer
- retravailler l'historique pour simplifier le graphe
- utiliser les branches

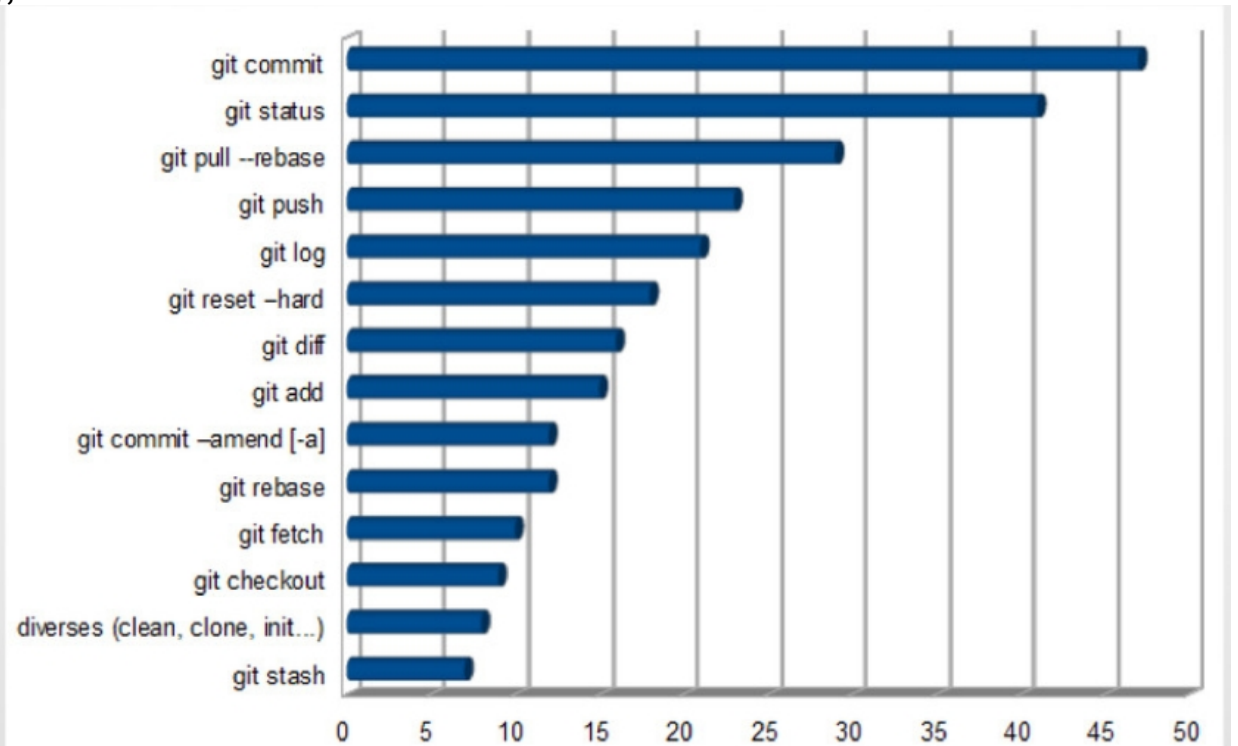
FRONT-END : git <command> [<args>]

```
Configuration
$ git config --global user.name "Your Name Comes Here"
$ git config --global user.email you@yourdomain.example.com

Bob importe un projet
$ tar xzf project.tar.gz
$ cd project
$ git init
$ git add .
$ git commit

ou Bob clone le projet d'Alice
$ git clone /url/alice/project project

Faire des modifications
$ git add file1 file2 file3
$ git status
$ git commit -a

Voir l'historique
$ git log

Gérer des branches
$ git branch experimental
$ git branch  (liste toutes les branches existantes)
$ git checkout experimental
(edit file)
$ git commit -a
$ git checkout master
(edit file)
$ git commit -a
$ git merge experimental
$ git diff  (to see conflict)
(edit file to resolve conflict)
$ git commit -a
$ git branch -d experimental (ensure that the changes are
          in the current branch)

Alice récupère le projet
alice$ git pull /url/bob/project master
```

Un aperçu des commandes

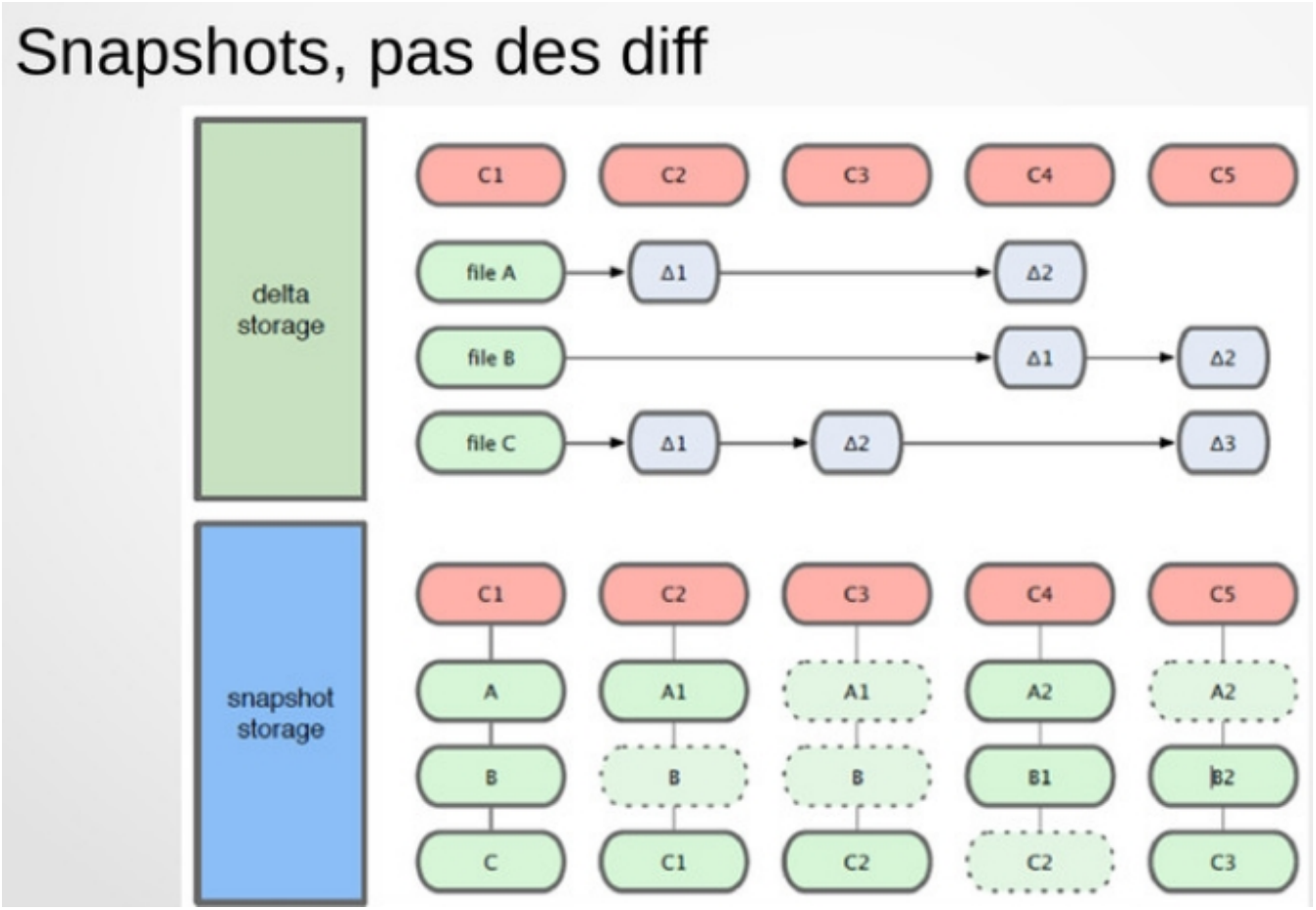
. configuration : config
. initialisation : init, clone
. travail local : add, remove, rename, status, diff, commit, show
. gérer le graphe : merge, rebase, squash, cherry-pick, revert, checkout (se déplacer)
. diversifier le développement : branch, merge
. référencer : tag
. communiquer avec d'autres repositories : push, pull, fetch
. revenir en arrière : reflog, reset
. doc : help

Statistiques d'utilisation
des commandes
par un utilisateur averti



BACKEND : orienté contenu

CVS et SVN orienté 'delta storage files'
GIT orienté snapshot storage (à chaque commit, tout le contenu est stocké)



ORGANISATION DES DONNEES

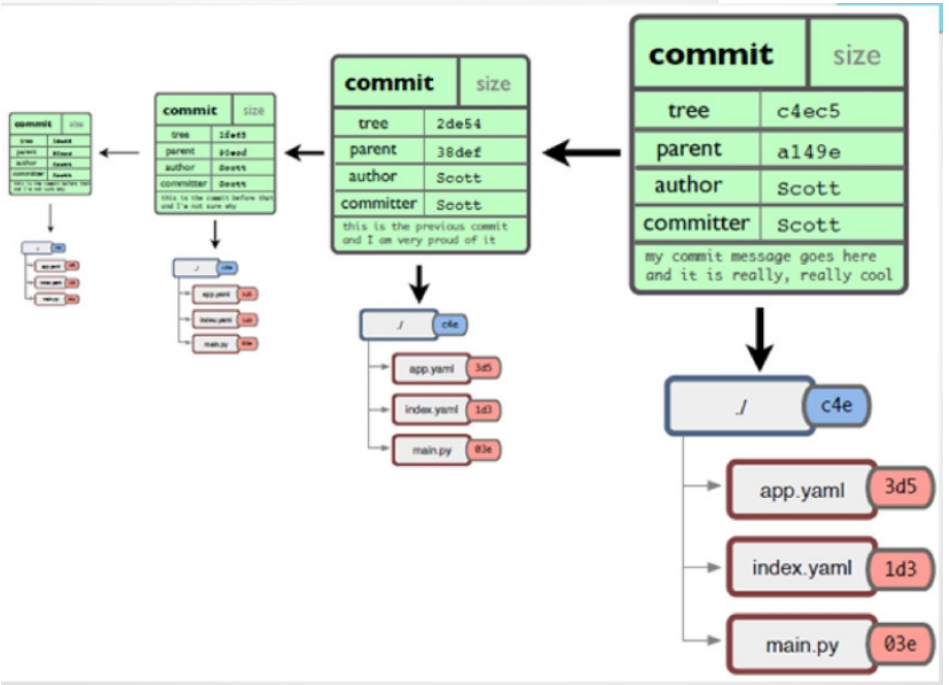
4 types d'objets dans Git :

- **Blob** (Binary Large Object) : contenu d'un fichier
- **Tree** : liste de références vers sous-arbres et blobs
- **Commit** : pointe sur un tree + références vers parents (au moins un) + métadonnées
- **Tag** : pointeur sur un commit + métadonnées

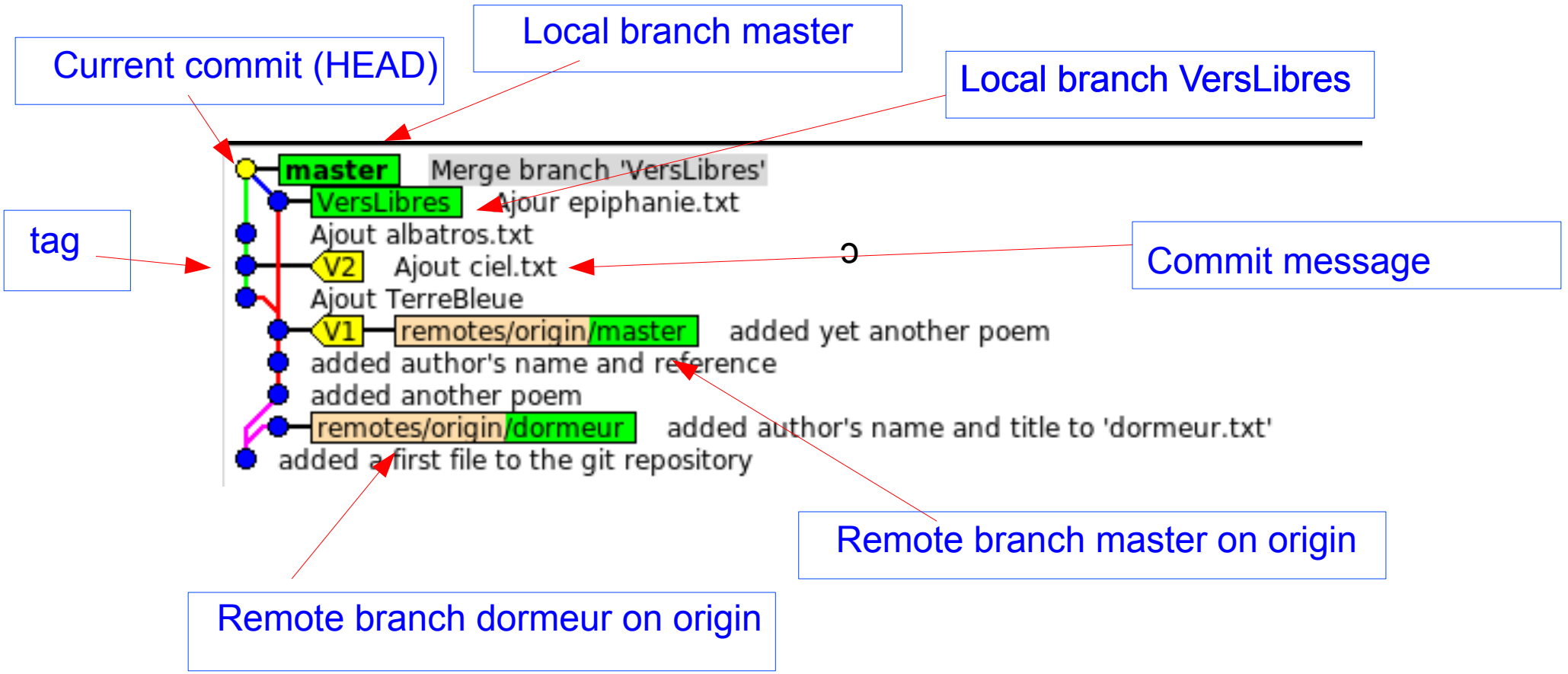
  Certains pointeurs sont déplacés par Git (master, branches), d'autres sont fixes (tag)

Chaque objet est identifié par un hash code : somme de contrôle SHA1 (40 caractères)

53b89fc **=** 53b89fc7bb117aee396285f9bc6ce913599a6574



GRAPHE DE COMMIT



DESIGNATION DES COMMITS

Possibilités de nommer les commits dans les commandes de différentes façons

- <nom de branche> : alias pour le commit le plus récent de la branche
- <tag> : alias de branche, jamais déplacé
- <SHA-1> : si non ambigu, le début du code suffit
- <nom>^ : père du commit
- <nom>^n : $n^{ème}$ parent (si parents multiples)
- <nom>~ : ancètre du dernier commit
- <nom>~n : $n^{ème}$ ancêtre (HEAD~2=HEAD^^)
- ...

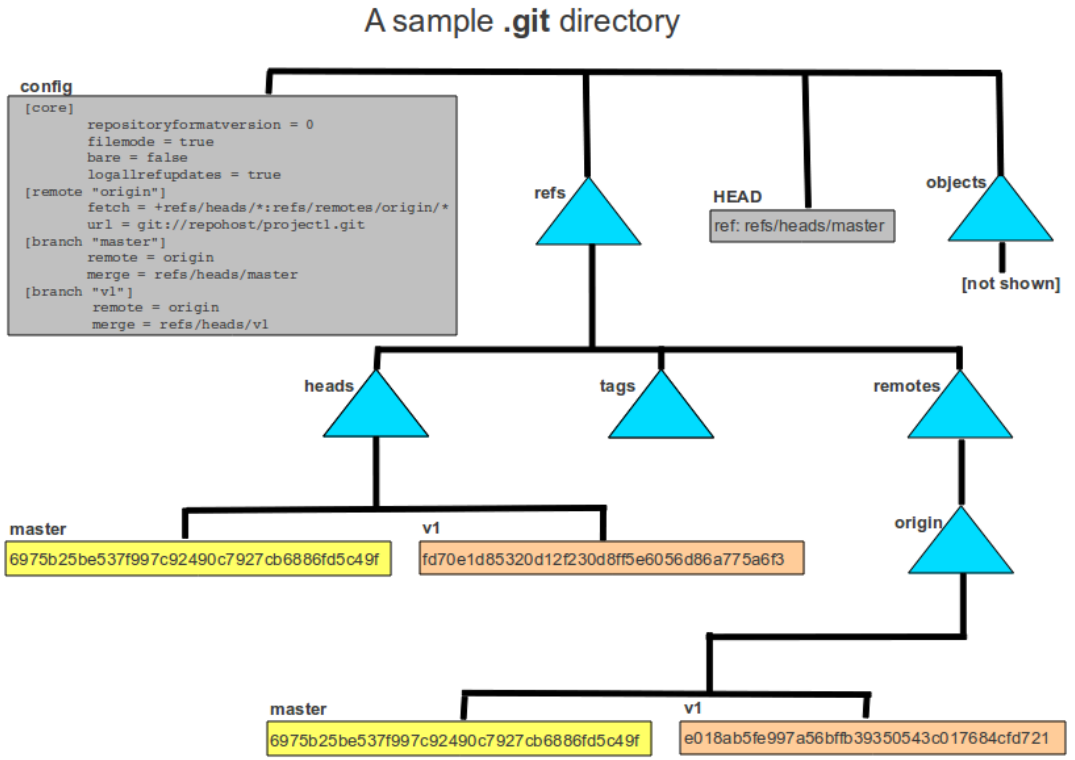
Tout commit non référencé (dangling ou detached HEAD) est détruit par le garbage collector.

STOCKAGE

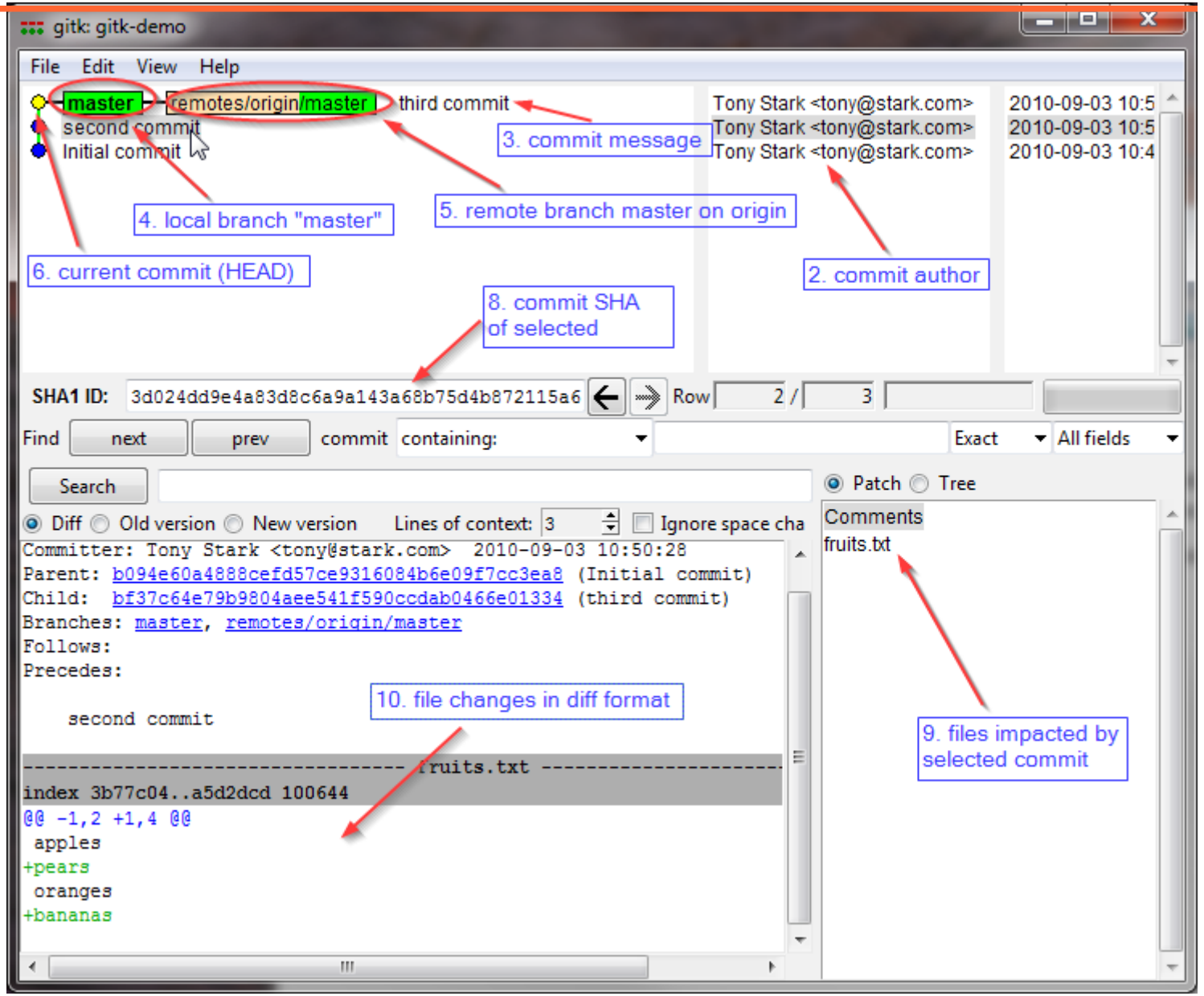
Configuration globale utilisateur (user, email, editor, alias, merge tool …)
→ ~/.gitconfig
$ git config --global color.ui true

```
diff --git a/epiphanie.txt b/epiphanie.txt
index 2b415c3..89f16e9 100644
--- a/epiphanie.txt
+++ b/epiphanie.txt
@@ -1,2 +1,6 @@
-Je rêvais depuis l'aube. Mon destin s'égouttait, en proie au doute, et goûtant
+Mon destin s'égouttait, en proie au doute, et goûtant
 le repos des clameurs neuves,
+
+Modification ajoutee
```

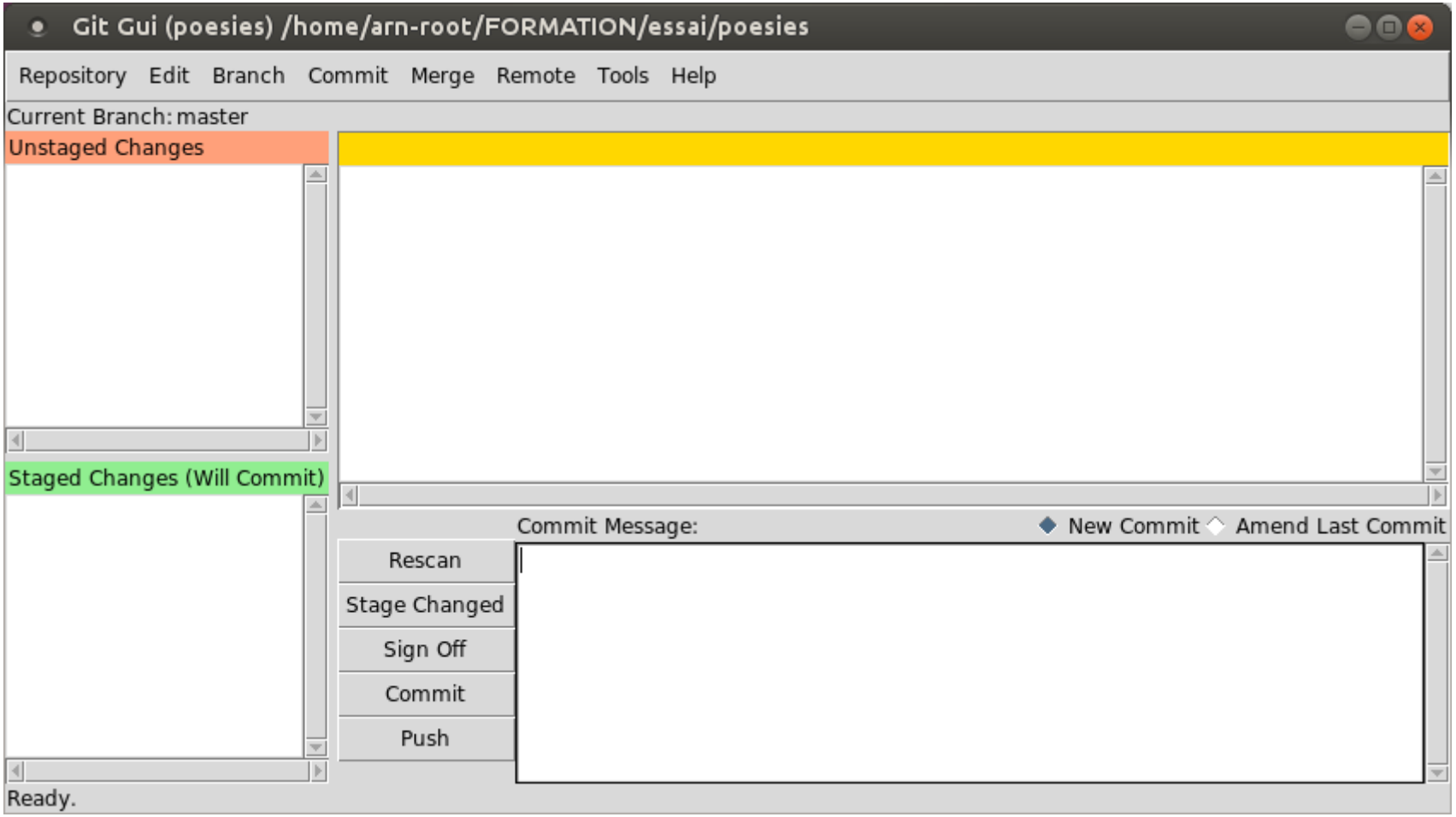
Par projet, un répertoire à la racine de l'arborescence du projet contient tout !
→ <PROJECT>/.git



OUTILS ASSOCIES : gitk



OUTILS ASSOCIES : git gui



Unlike gitk, git gui focuses on commit generation and single file annotation and does not show project history

OUTILS ASSOCIES

D'autres environnements existent : git-cola, smartgit, gitg, giggle ...

Github.com : service web d'hébergement et de gestion de développement de logiciels, utilisant le programme Git

WORKFLOW

Git flow

GitHub flow (beaucoup plus simple)

→ anything in the master branch is deployable
→ to work on something new, create a descriptively named branch off of master
→ commit to that branch locally and regularly push your work to the same names branch on the server
→ when you need feedback or help, or you think the branch is ready for merging, open a pull request
→ after someone else has reviewed and signed off on the feature you can merge it into the master
→ once it is merged and pushed to 'master', you can and should deploy immediately

Que du bon ?

Quelques points faibles

- Git encombrant sur Microsoft Windows
- l'historique peut grandir très vite et très fort (attention avec fichiers binaires)
- les sous-répertoire vides ne peuvent pas être suivis
- noms ou utilisation de commandes parfois déroutantes pour démarrer
- marquer qui modifie quoi

Les utilisateurs semblent enthousiastes !

Communauté d'utilisateurs vivantes, beaucoup de tutoriels sur le net.

Quelques références

Site officiel : http://git-scm.com/

Livre 'Pro Git' de Scott Chacon : http://git-scm.com/book/fr/

Apprendre Git en interactif en 15 mn : http://try.github.io/levels/1/challenges/1

Apprendre Git : http://www.codeschool.com/courses/try-git ,
http://www.codeschool.com/courses/git-real ,
http://www.codeschool.com/courses/git-real-2

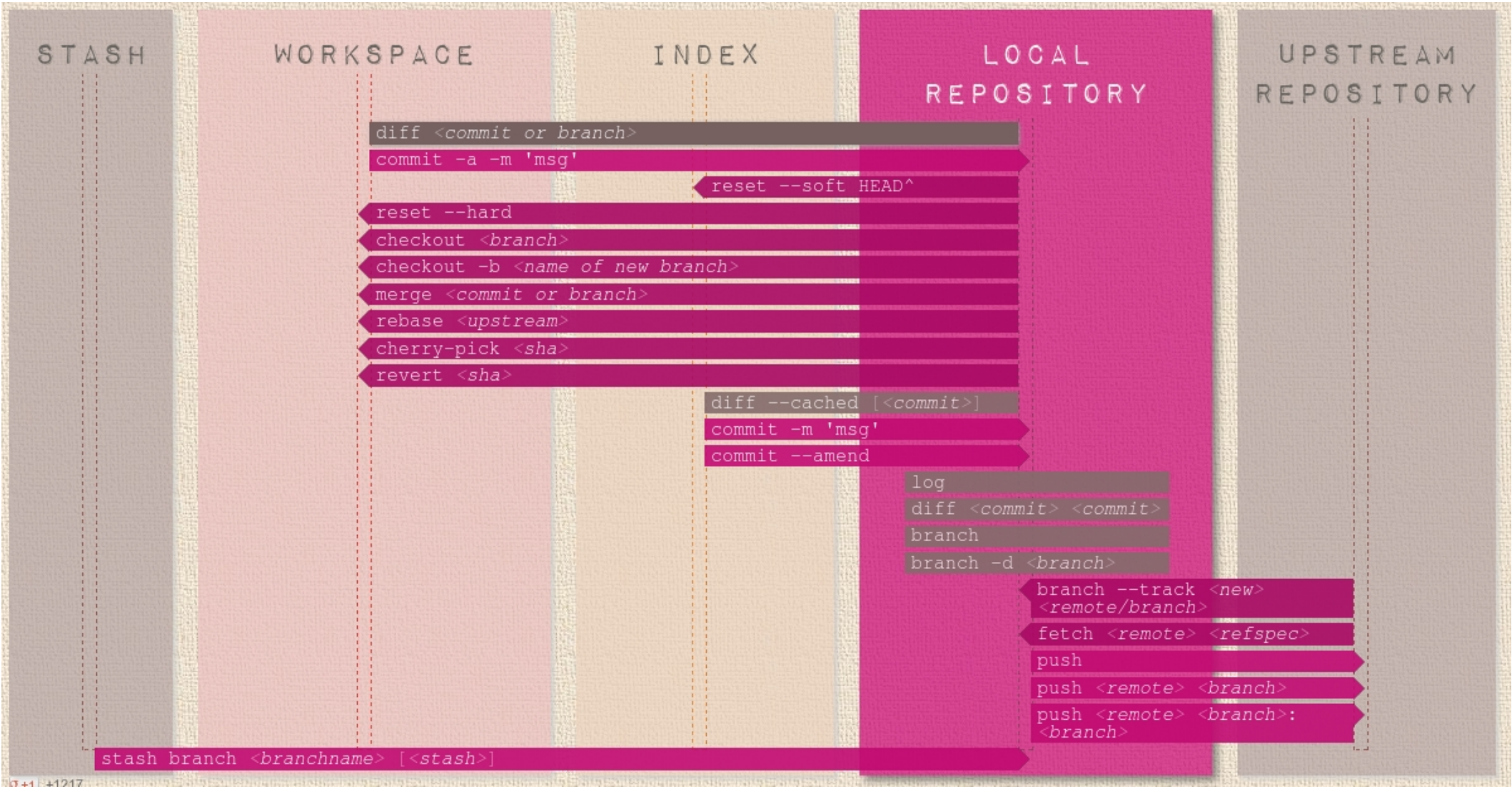
Guide tour : http://gitimmersion.com/

Référence des commandes : http://gitref.org/

Description interactive des commandes :
http://ndpsoftware.com/git-cheatsheet.html#loc=local_repo;

TP : http://nathalievilla.org/spip.php?article91

Commandes



http://ndpsoftware.com/git-cheatsheet.html#loc=local_repo;