Outils de développement et compilation

Git : gestion de code source et versionnement

Timothée Ravier

LIFO, INSA-CVL, LIPN

1re année cycle ingénieur STI
2013 – 2014

Plan global

Plan

Ce que l'on veut éviter

- cp -a mon-projet mon-projet.old
- cp -a mon-projet mon-projet.backup-03-04-2014
- cp -a mon-projet mon-projet.backup-2014-03-04

Concepts

- Se souvenir de l'état d'un projet à un moment donné ;
- Comparer le code d'un projet entre deux versions ;
- Revenir à une version précédente ;
- Faciliter la collaboration sur un projet ;
- Se souvenir de qui a modifié quel fichier, à quel moment et pourquoi ;
- Faciliter la sauvegarde de ces informations ;
- Automatiser la gestion de ces informations.

Gestion des versions d'un logiciel

- Comparaison des différences dans le code entre deux versions ;
- Utilitaire `diff` pour créer des patchs :

```
$ diff pjt-orig/file.c pjt-fixed/file.c > fix.patch
```

- Qui sont appliqués avec `patch` :

```
$ cd pjt-orig && patch -p1 < fix.patch
```

- Contribution du patch à un projet open source :
  - Envoi des patchs par mail sur une mailing list ou soumission d'une « pull request » (cf partie Git).

Gestion des versions d'un logiciel

- Si le code change il faut refaire le patch ;
- Outils pour gérer des suites de patchs (`quilt`) ;
- Aucune sauvegarde automatique ;
- Difficile de garder l'historique ;
- Difficile de travailler sur plusieurs patchs à la fois ;
- Difficile de suivre quel patch a été appliqué et pourquoi ;
- Il faut plusieurs copies du projet pour travailler.

Concurrent Versions System (CVS)

- Un des premiers gestionnaires de versions de code source ;
- Chaque fichier a sa propre version ;
- Stockage centralisé sur un serveur : dépôt ou « repository ».

Subversion (SVN)

- Principe du commit : regroupe plusieurs changements quelque soit le nombre de fichiers modifiés ;
- Commit indexés par des numéros strictement croissants ;
- Gestion centralisée : l'historique est stocké sur le serveur ; Seul le serveur possède toutes les versions du projet.
- Pour ajouter un commit il faut avoir accès au serveur ;
- Principe des branches : regroupe un ensemble de commits en leur donnant un nom.
- La branche principale s'appelle `trunk`.

Subversion (SVN) : usage

- `$ svn checkout http://projet.com/svn/trunk/ projet -username moi@mail.com`
- `$ svn status`
- `$ svn add <fichiers>`
- `$ svn update`
- `$ svn commit`

Bazaar (bzr), Mercurial (hg)...

- Gestion décentralisée : tout le monde possède une copie complète du projet, avec toutes versions et l'historique complet ;
- Il n'est plus nécessaire de communiquer en permanence avec le serveur ;
- Branches locales.

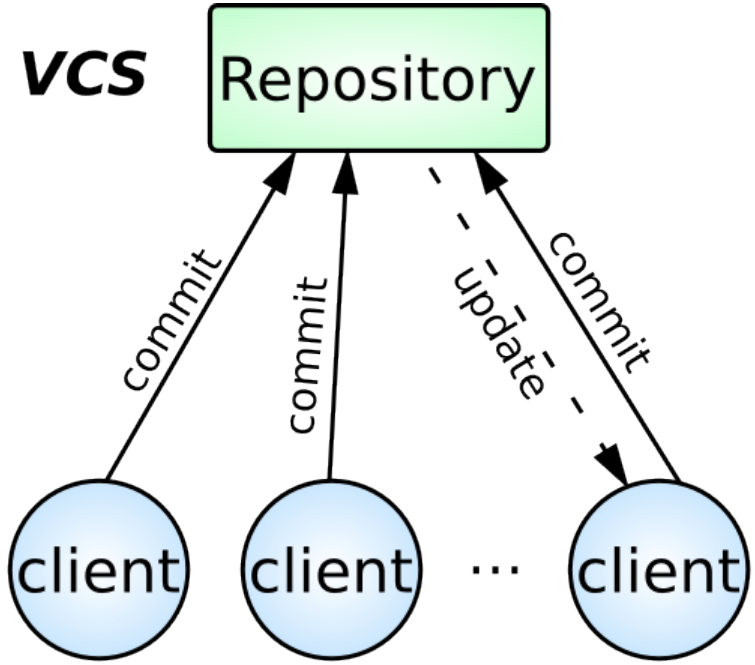
Plan

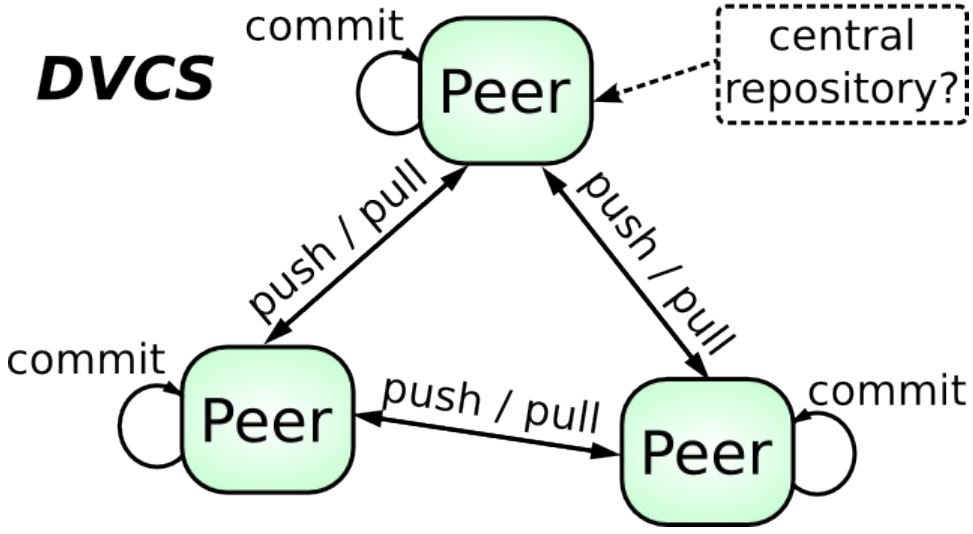
Caractéristiques

- Logiciel libre (GNU GPL 2) ;
- Crée par Linus Torvalds, le créateur du noyau Linux ;
- Conçu dès le départ pour fonctionner avec le noyau ;
- Gestion de source et de versions ;
- Décentralisé / distribué (ne nécessite pas de connexion avec un serveur pour fonctionner) ;
- Branches locales gratuites ;
- Rapide et puissant ;
- ...

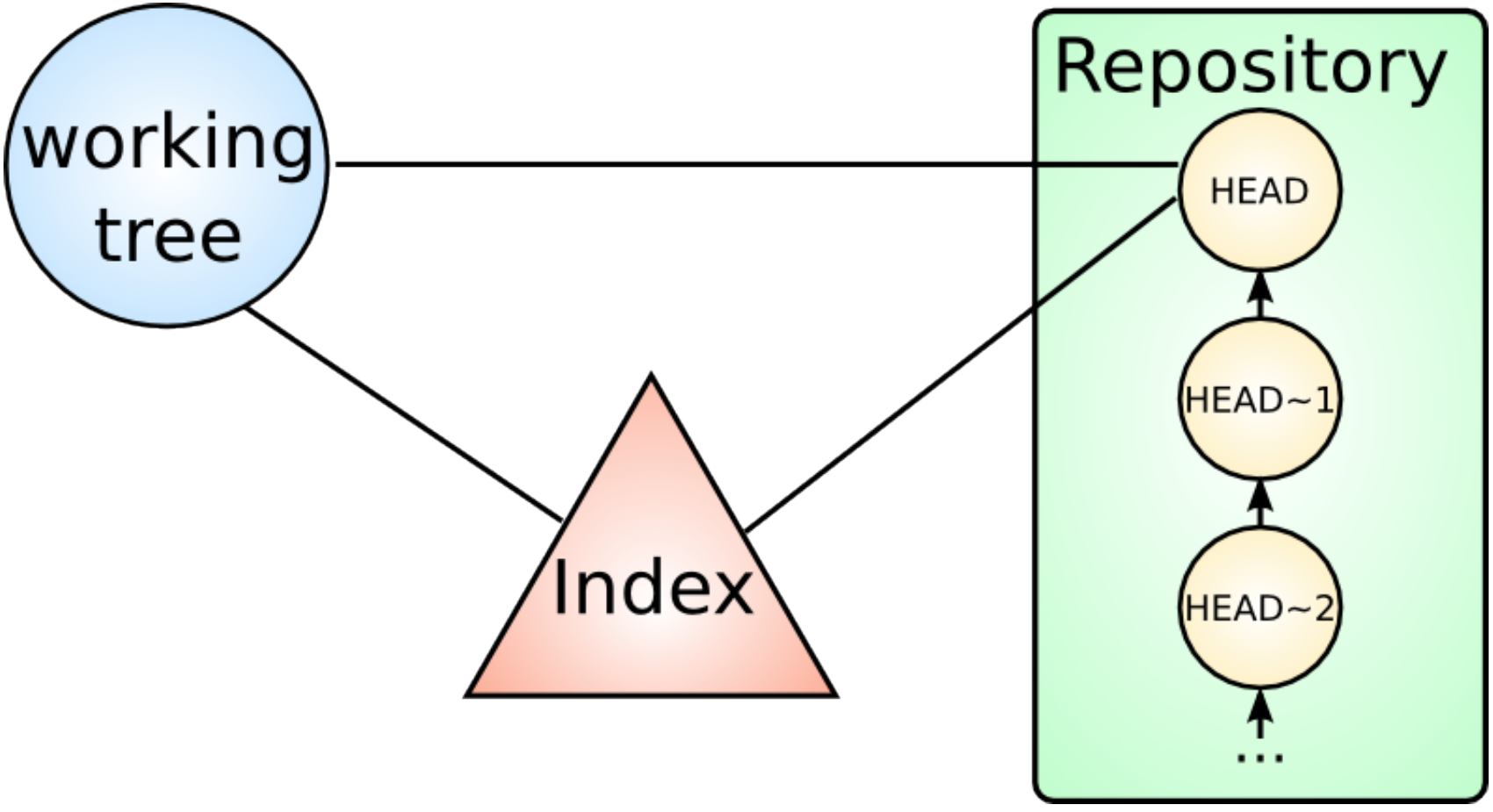
Plus qu'une sauvegarde

- Sauvegarde facile sur un serveur distant ;
- Propose une attitude saine de développement ;
- Conçu pour le développement coopératif ;
- Sûr : les hashs des commits correspondent au code.

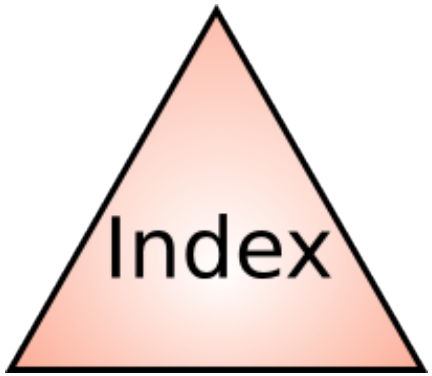

VCS
Repository
commit
commit
update
commit
client
client
...
client


DVCS
commit
Peer
central repository?
push / pull
push / pull
commit
Peer
push / pull
Peer
commit

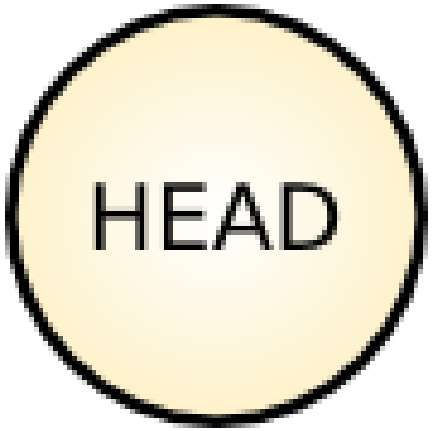
What's git



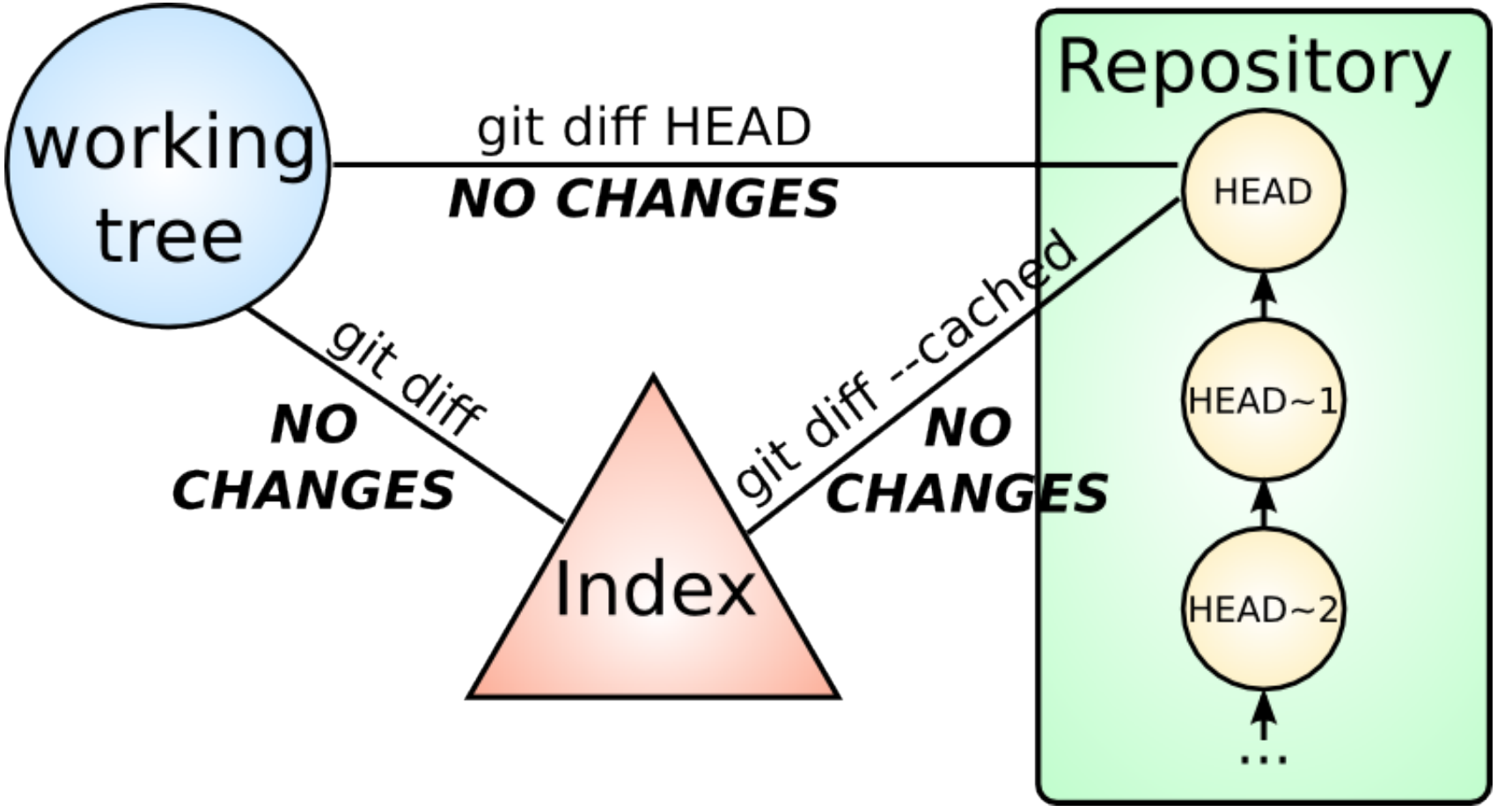
It’s the sandbox



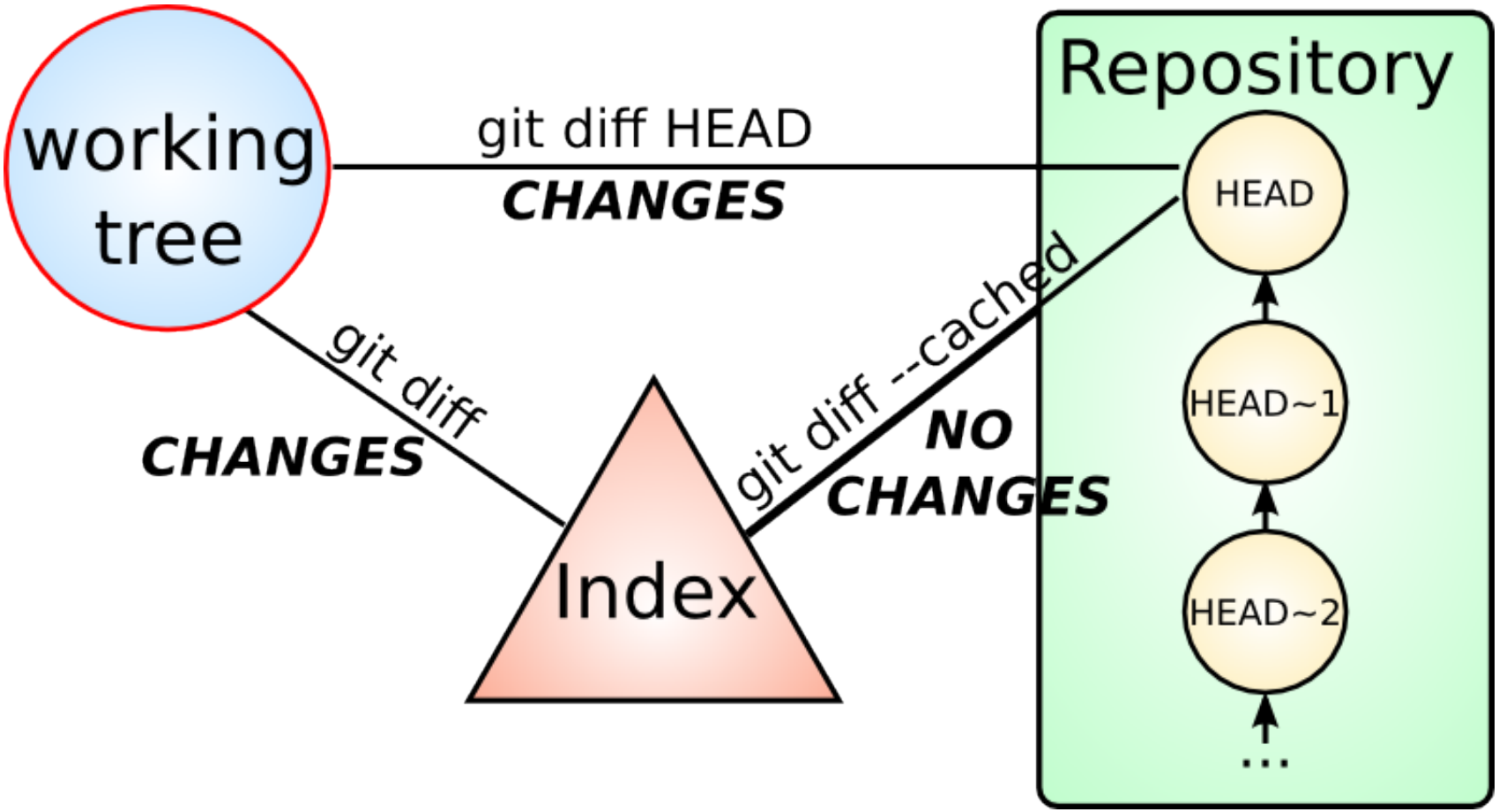
Holds the new changes to be committed

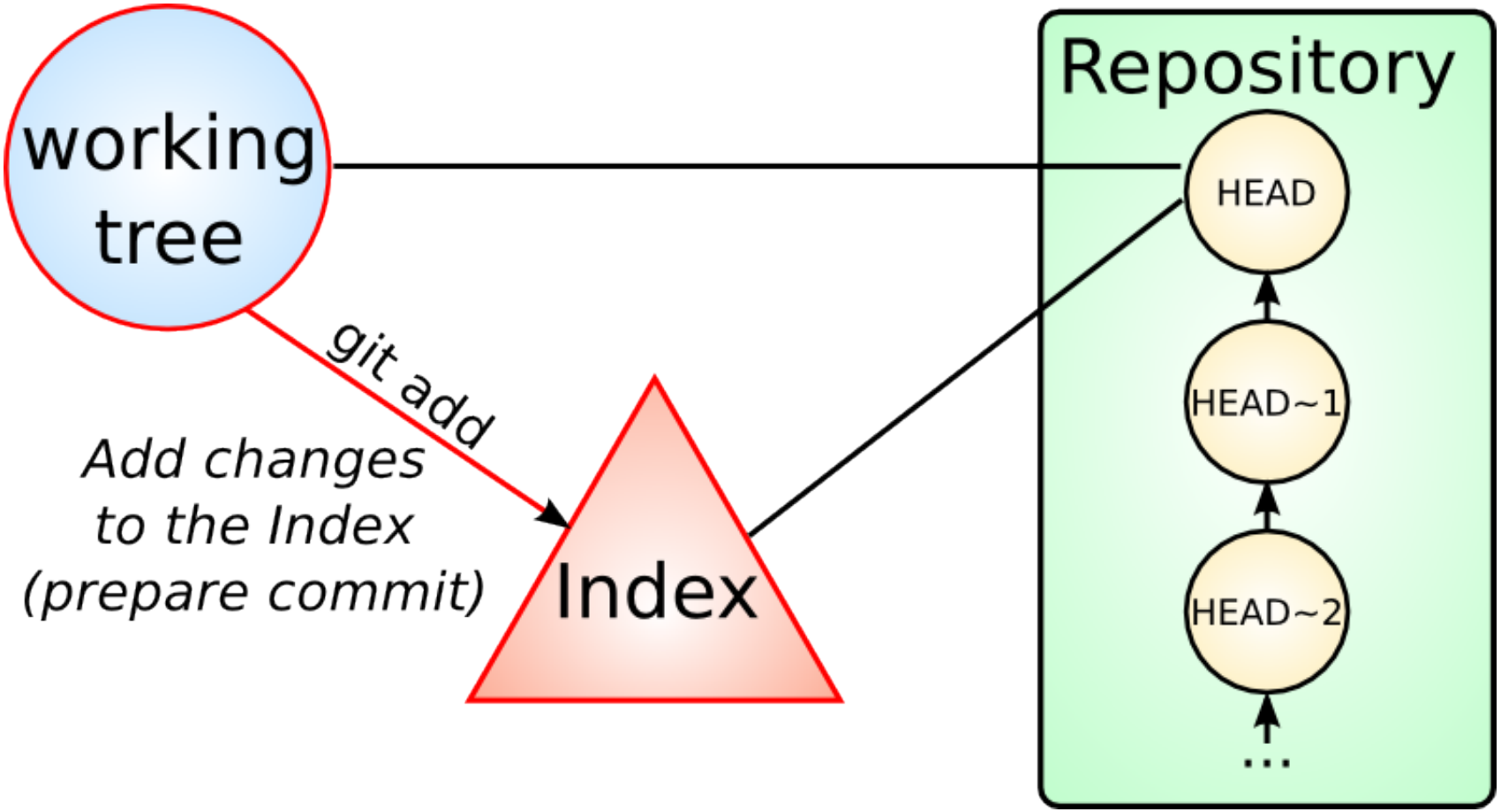


Points to the current commit

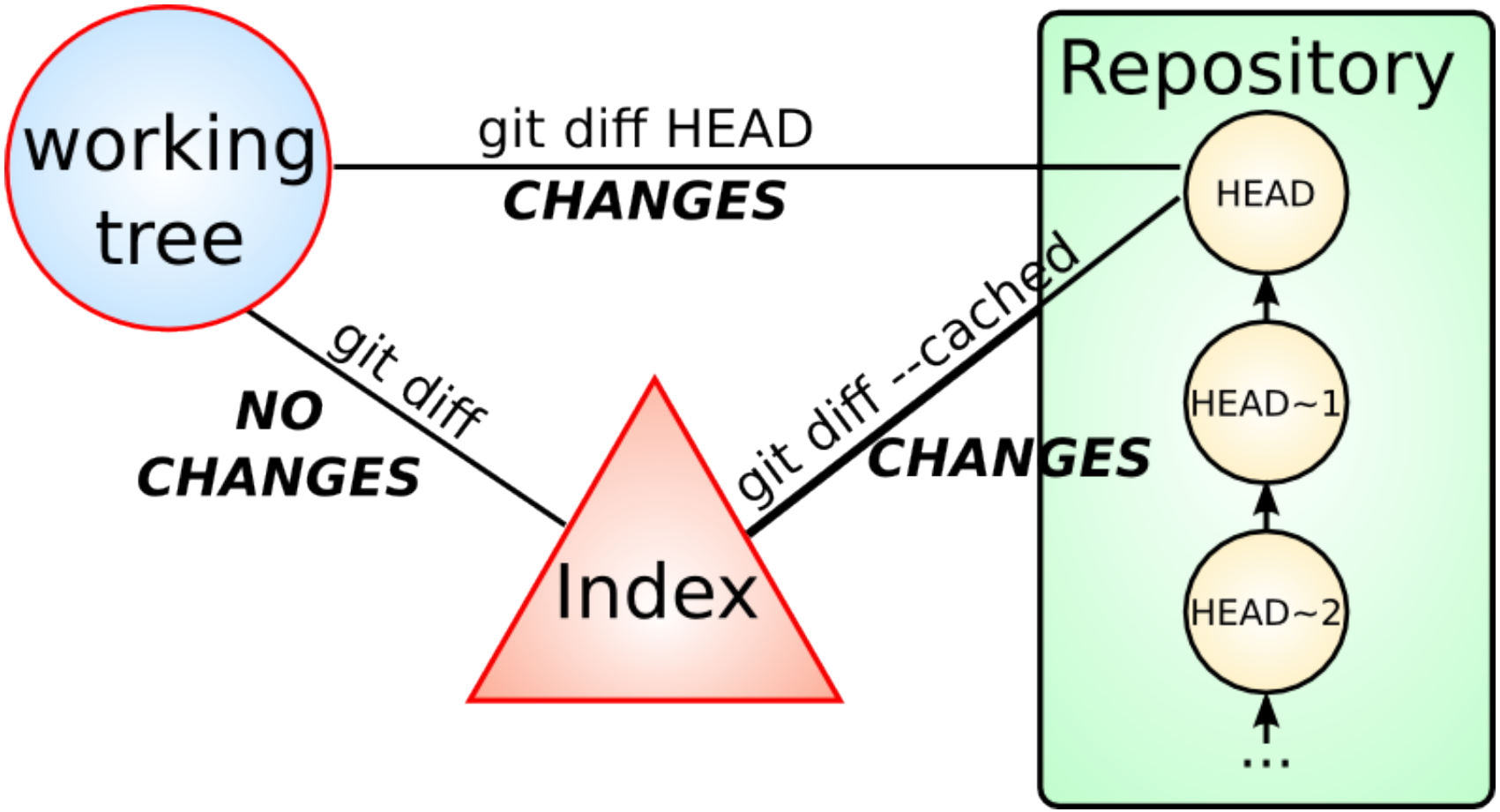

working tree
git diff HEAD
NO CHANGES
Repository
HEAD
HEAD~1
HEAD~2
...
git diff
NO CHANGES
Index
git diff --cached
NO CHANGES

Working Tree with Changes

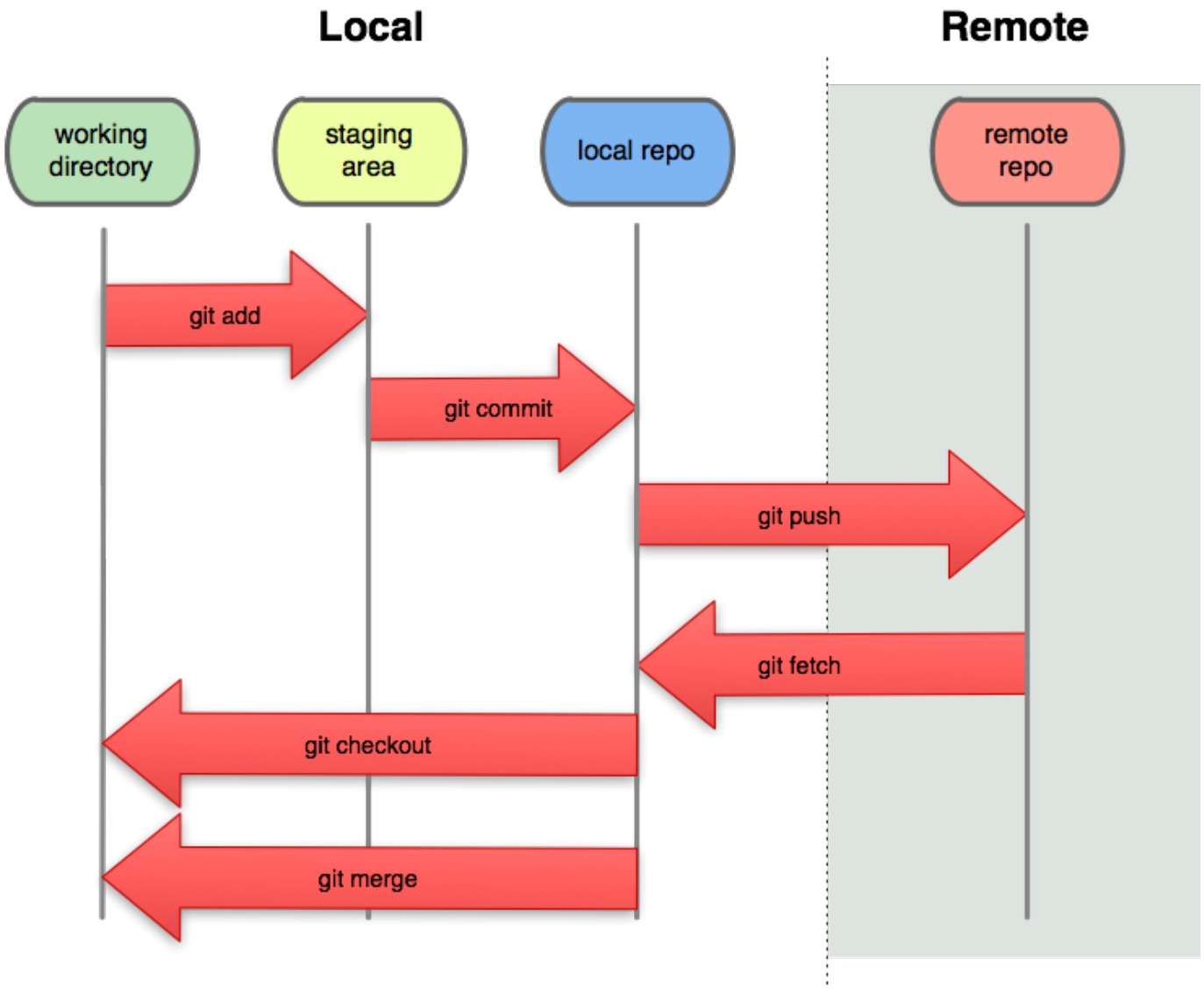



working tree
git add
Add changes to the Index (prepare commit)
Index
Repository
HEAD
HEAD~1
HEAD~2
...

Index with Changes



Architecture



Travailler avec Git

- `git add` : ajoute des fichiers au dépôt et au prochain commit (index) ;
- `git rm` : supprime des fichiers d'un dépôt ;
- `git commit` : crée un commit en local ;
- `git pull` : récupère les changements présents sur le dépôt distant ;
- `git push` : envoie les changements sur le dépôts distant ;
- `git reset` : annule certaines opérations (notamment add) ;
- `git checkout` : effectue des modifications sur la copie de travail.

Tout est une branche

Travailler avec les branches

- `git branch` : créer une nouvelle branche ;
- `git checkout` : mets à jour la copie de travail, donc potentiellement change de branche ;
- `git merge` : regroupe les changements de plusieurs branches ;
- `git rebase`, `git pull -rebase` : déplace les commits de la branche courante à la suite des commits d'une autre branche ;
- `git abort` : annule une opération de merge ou de rebase en cours.

Tout est récupérable, ou l'historique tout puissant

- `git log` : affiche l'historique ;
- `git diff` : affiche les différences entre deux commits, l'index ou le dossier de travail ;
- `git revert` : annule un commit ;
- `git stash [list|save|pop]` : place les changements courant dans une zone de stockage temporaire.

Plan

Documentation

- Lire le tutoriel officiel et le Git Pro Book [3].

Pensez à vos messages de commit !

http://xkcd.com/1296/

Références I

1. Git Pro Book : http://git-scm.com/book/
2. Vue interactive d'un dépôt : http://pcottle.github.io/learnGitBranching/
3. Why Git is better than X : http://thkoch2001.github.io/whygitisbetter/