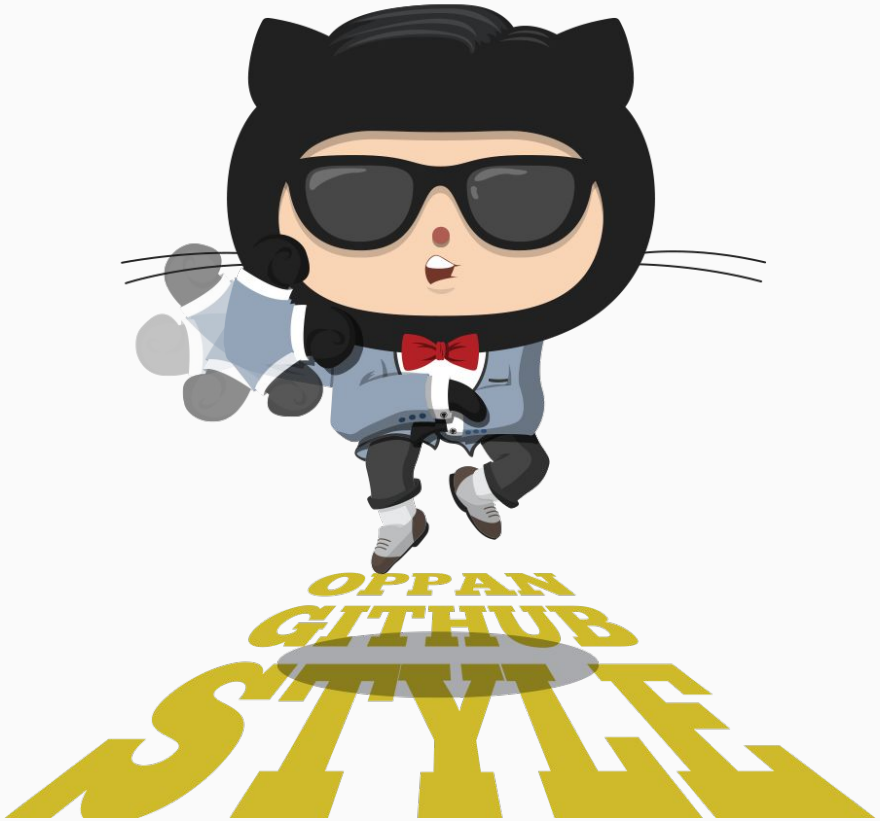
GIT Ignition

Guide de GIT pour briller en société



GIT = Con / Conne… Car même le grand Linus Torvalds peut se tromper :)

Disclaimer

- Ultra-Condensé à partir de la formation GitTotal de Christophe Porteneuve.
- @porteneuve est l'un des spécialistes de GIT en France. Le suivre sur Twitter me semble une excellente ideé.
- GitTotal c'est 3 jours de formation intensive qui vous colle dans le buffer. En 1h à 2h de présentation vous n'aurez donc quasiment rien d'autre qu'un très rapide aperçu.

Pourquoi c’est génial ?

- Pas de criticité d'une seule machine
- Les contributeurs peuvent travailler déconnectés du « serveur »
- On peut participer à un projet sans nécessiter les permissions par un responsable du projet (les droits de commit/soumission peuvent donc être donnés après avoir démontré son travail et non pas avant); inestimable pour la contribution aux projets open-source.
- La plupart des opérations sont (beaucoup) plus rapides, car locales !
- On peut faire du travail « privé » : des brouillons, des essais… sans risque de gêner les collaborateurs.
- Rien n’empêche de garder un dépôt de référence, avalisé, sur lequel seuls certains peuvent écrire.

.gitconfig (à copier / coller dans votre ~)

```
[color]
  ui = true
[user]
  name = Votre Nom
  email = votre.email@votre.domaine
[alias]
  ck = checkout
  st = status
  ci = commit
  lg = log --graph --pretty=tformat:'%Cred%h%Creset -%C(auto)%d%
Creset %s %Cgreen(%an %ar)%Creset'
[core]
  pager = cat
  #
  # Out of luck: on Windows w/o msysGit? You may have Notepad++…
  # editor = 'C:/Program Files/Notepad++/notepad++.exe' -multiInst -
notabbar -nosession -noPlugin
  #
  # If you want to use Sublime Text 2's subl wrapper:
  editor = nano
  whitespace = -trailing-space
  autocrlf = input
[diff]
  mnemonicPrefix = true
  wordRegex = .
[fetch]
  recurseSubmodules = on-demand
[grep]
  extendedRegexp = true
[log]
  abbrevCommit = true
[merge]
  conflictStyle = diff3
  tool = p4merge
[mergetool]
  keepBackup = false
  keepTemporaries = false
  prompt = false
```

```
(...)
[pull]
  # This is GREAT… when you know what you're doing and are
careful
  # not to pull --no-rebase over a local line containing a true
merge.
  # rebase = true
  # WARNING! This option, which does away with the one gotcha of
  # auto-rebasing on pulls, is only available from 1.8.5 onwards.
  rebase = preserve
[push]
  default = upstream
[rerere]
  # If, like me, you like rerere, decomment these
  # autoupdate = false
  # enabled = true
[status]
  submoduleSummary = true
[merge "theirs"]
  driver = true
```

Jargon

Commit / Révision / Version

Série de modifications faisant l'objet d'une soumission unique, idéalement thématique et atomique. Reçoit un identifiant unique

Head

Dernière version du projet dans la branche en cours

Tag / Label

Étiquette d'un commit

Dépôt / Repository

Ensemble de commit du projet + méta-données spécifiques à GIT.

Branche

Ensemble de commits. Un même dépôt peut avoir plein de branches, comme un arbre.

Tronc / Master

La branche de référence / principale.

Fusion / Merge

Fusion des fichiers en un ensemble cohérent et coordonné.

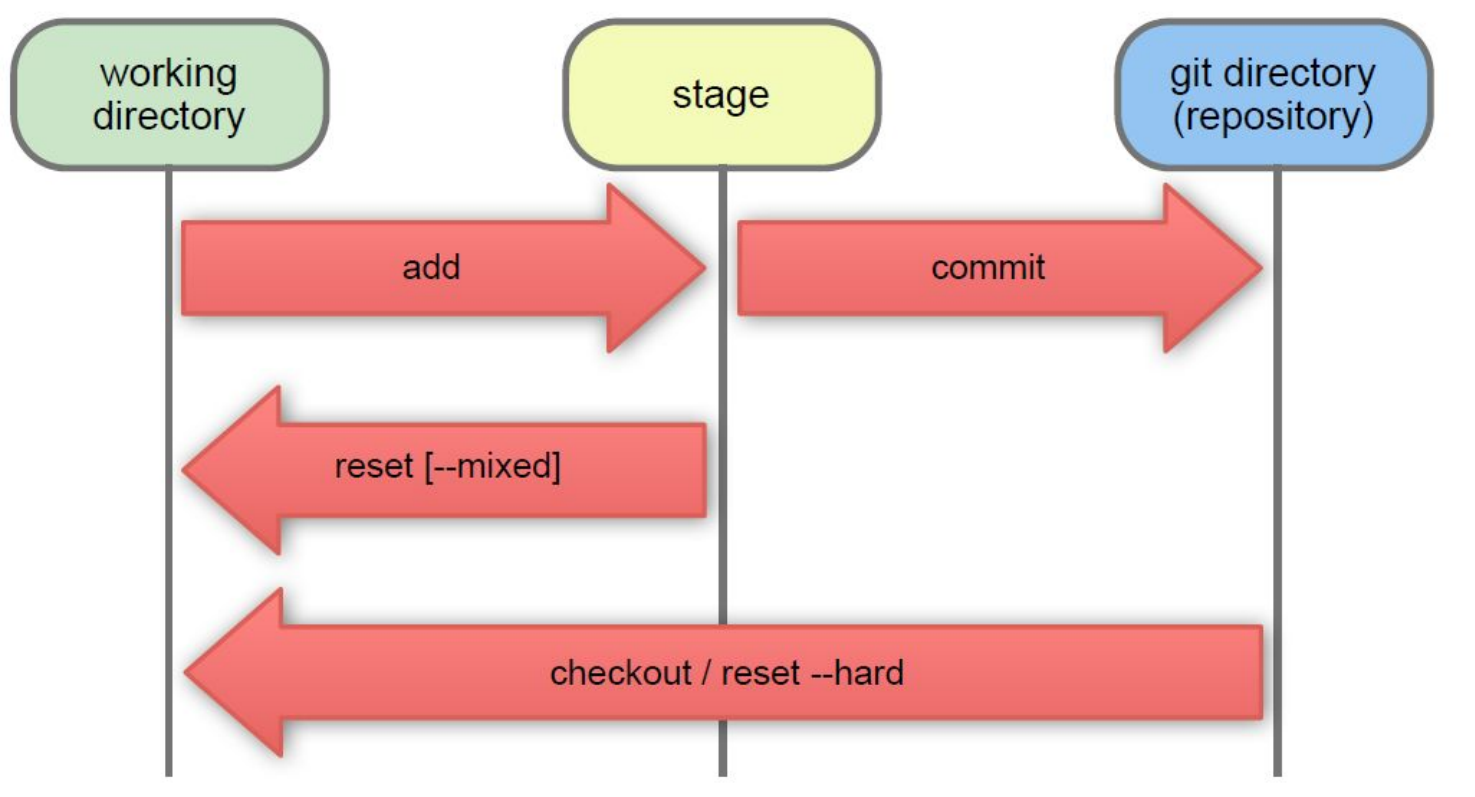
Conflit

Situation empêchant une fusion automatique ; se résout manuellement

Working Directory/Tree/Copy

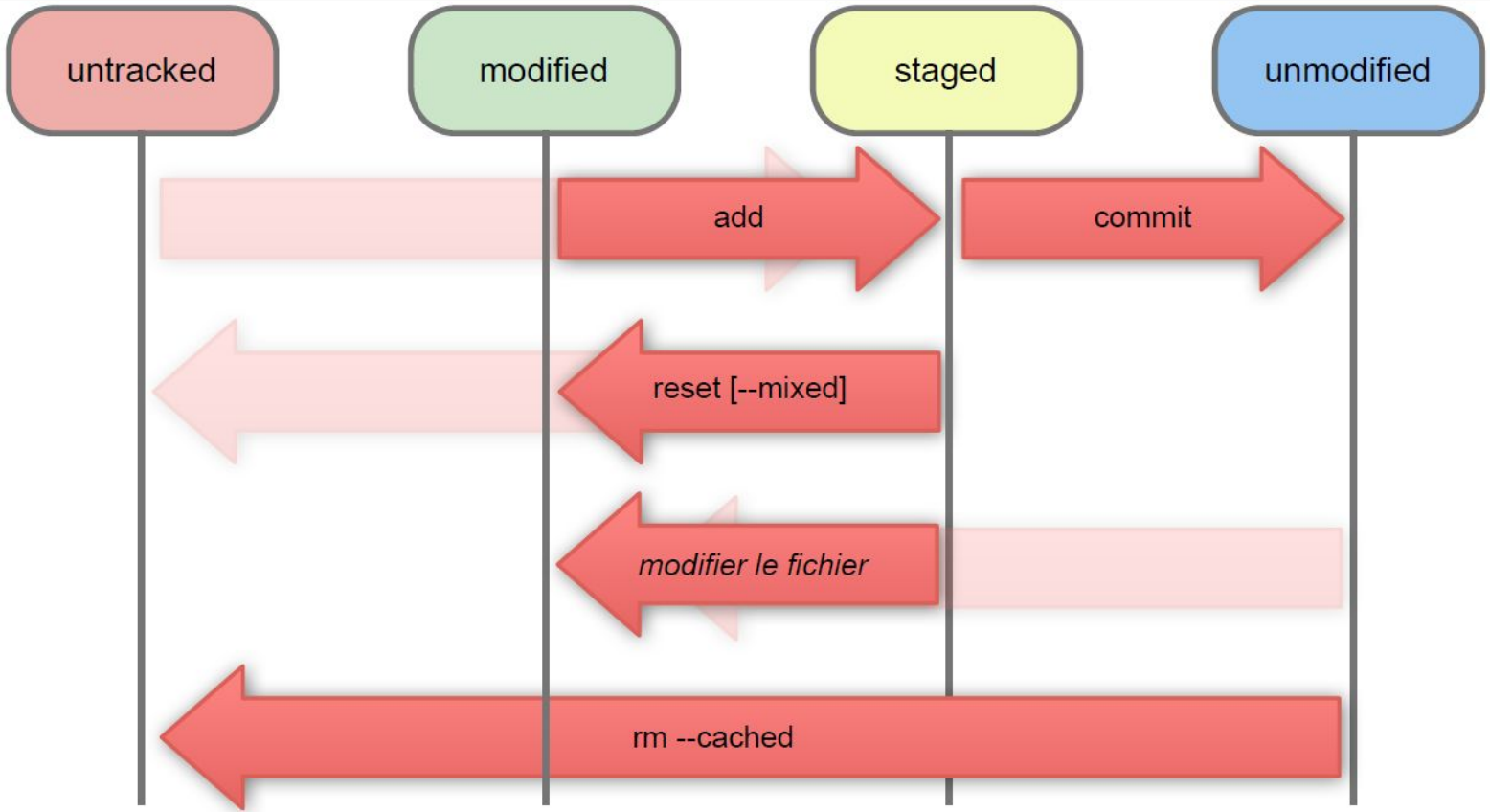
L'état actuel sur le disque des dossiers et fichiers

L’idée de Git



http://ndpsoftware.com/git-cheatsheet.html

La vie d’un fichier dans GIT



Commande de base

git init (puis création du .gitignore)
git add .
git commit -am "Import initial" ou git clone adress_du_depot
git commit -m
git commit -a (attention, pas les untracked !)

Commandes de Fun

git add -i (Interactif !)
git add -p (Patches)
git status
git diff (--staged)
git log (ou version aliasé git lg)
git show
git commit --amend
git oops
git reset
git rm --cached

Modification :
^HEAD : n-1 si n=HEAD
^^
^^^
…
HEAD~2

NE PAS COMMITER TOUT ET N'IMPORTE QUOI ! On est pas chez mémé !

Tu touches à un truc tu commit ! (sinon !)

- Tirer parti du git add -p
- Retirer du stage « trop enthousiaste » avec git reset ou git rm --cached (pour du non versionné jusqu'ici)
- Faire une série de commits atomiques

C’est René chef... Il a touché au truc, ça a fait pchit et y’a plus rien qui marche....

T’as paumé un truc ?
git reflog

- purgé au GC
- gc.reflogExpire
- purement local traine pas trop quand même

T’as complétement tout cassé, tu veux revenir au dernier commit avant tes modifications ?
git reset --hard

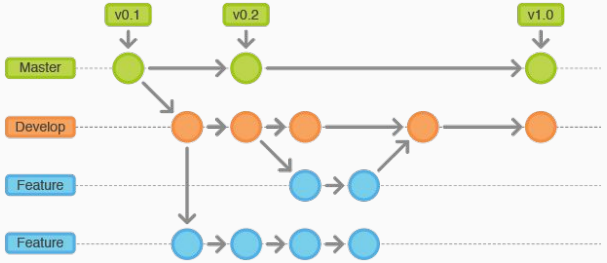
T’as pas le temps de branch pour aller éteindre un incendie ?
git stash
git stash apply

Tu veux faire semblant d’avoir tout bien commité comme il faut du premier coup ?
git rebase -i [sha1]

Branches (On est bien dans sa branche ouéch !)

Pourquoi faire ?

- Isoler du code et pas tout péter
- Travailler à plusieurs sans jamais se gêner
- Tenter un truc badass sans souiller son slip
- Si ta modif va te prendre plus de ½ journée.

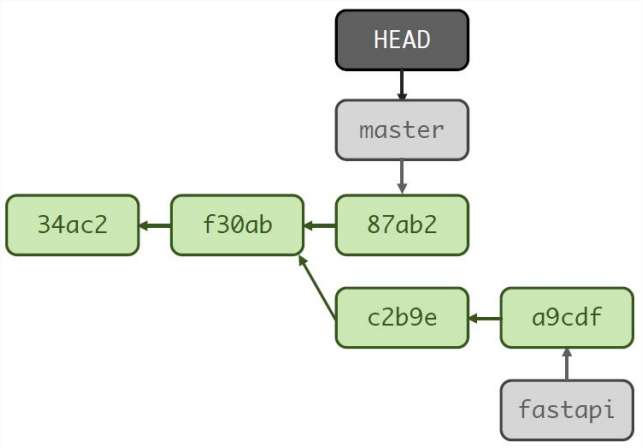


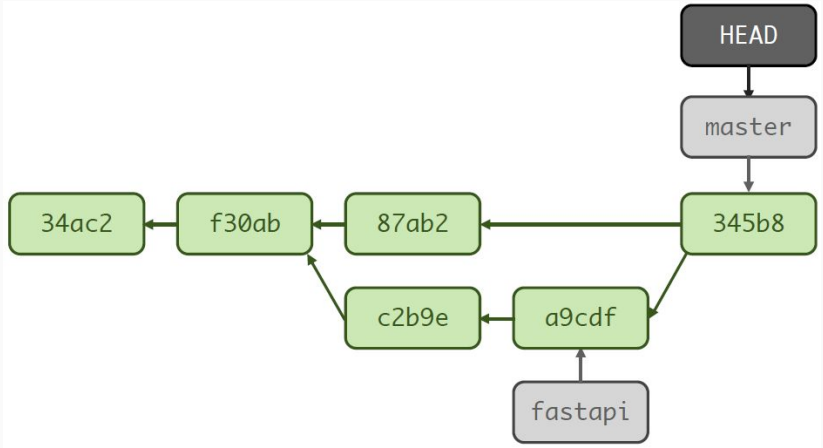
Commandes :
git branch nomBranche
git branch -d nomBranche (supprimer fusionnée)
git branch -D nomBranche (supprimer non fusionnée)
git checkout -b new_branch (crée et bascule)

Etiquetter une branche
git tag nomDuTag
git push --tags

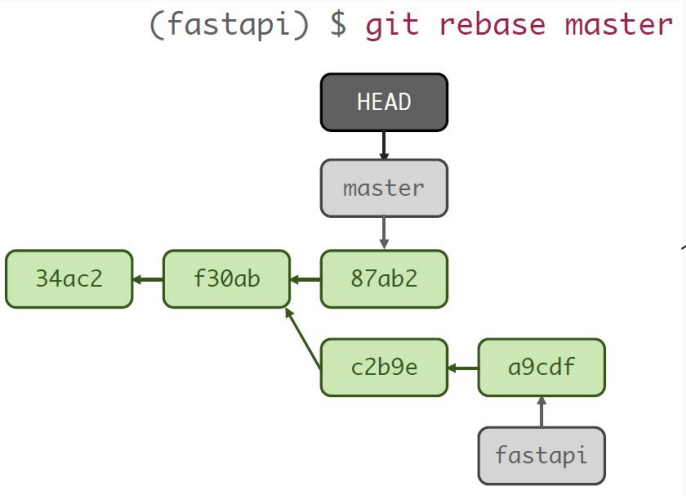
Dupliquer une modification
git cherry-pick sha1

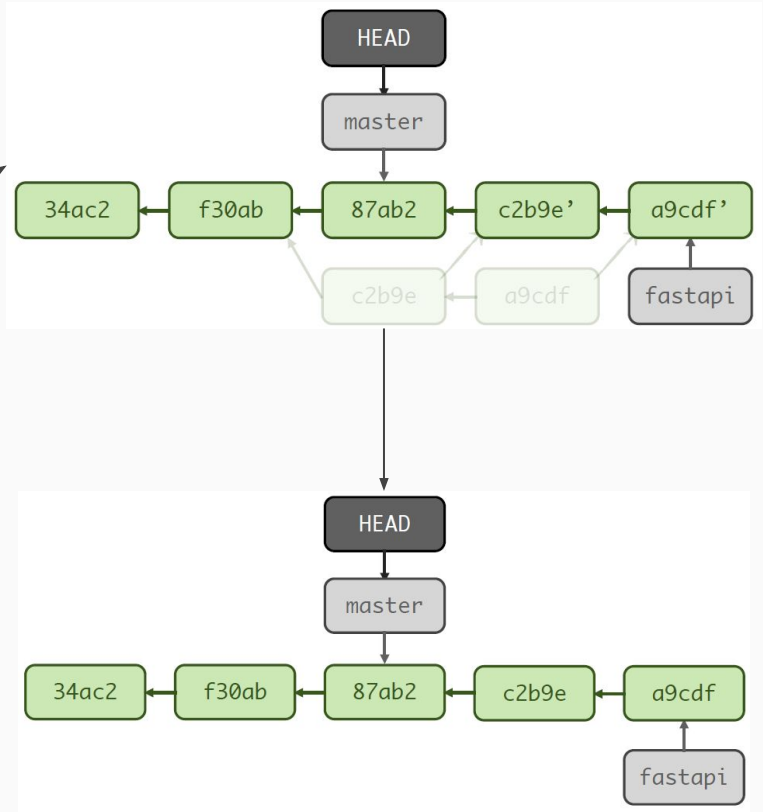
Git Merge pur (no-ff)





Git Rebase + Merge (fast forward)



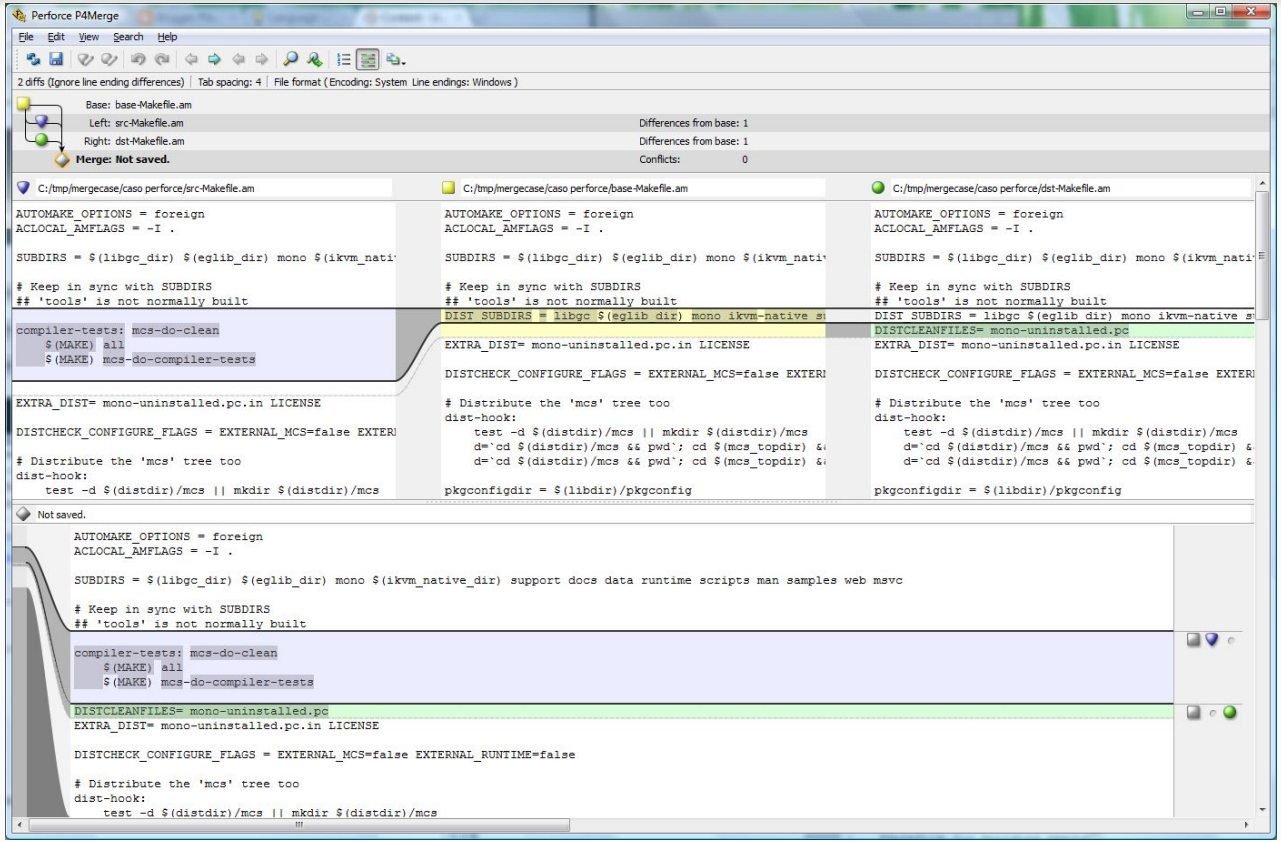


Ca marche pas… J’ai un conflit :(

git mergetool
resoudre les conflits puis
git add .
git rebase --continue

done.

Si tout le temps le même conflit, activer le “rerere”



https://www.perforce.com/product/components/perforce-visual-merge-and-diff-tools

Dépots distants

Lier une branche locale avec le dépot distant
git remote add name url
ex. git remote add origin git@github.com:lalmat/nRelay.git
git remote rename old new

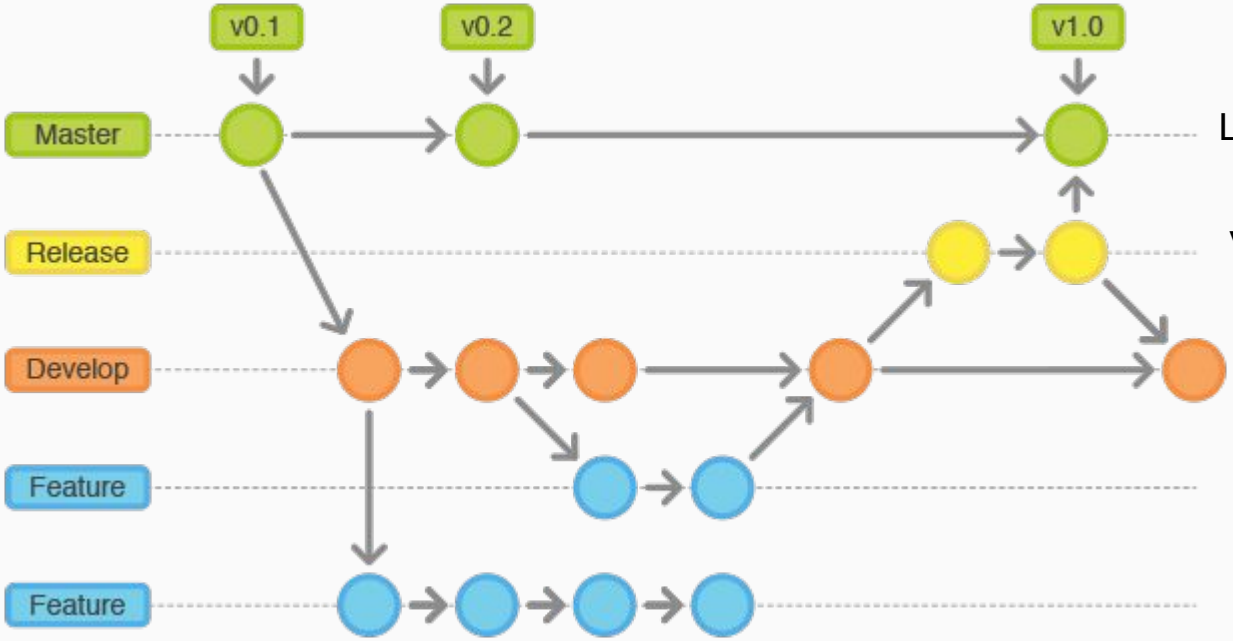
Récupérer les modifications distantes
git fetch (récupère sans appliquer)
git fetch origin master
git pull (= git fetch + git merge)
git pull --rebase (idem + rebase)

Voir un peu se qui se passe
git branch -a (voir tout)
git branch -t laBranche origin/laBranche
(la récupérer en local + la suivre)
git checkout laBranche

Envoyer ses modifications
git push -u origin laBranche (au début)
git push (après)
git push origin :branch-to-delete (supprimer une branche à distance)

git push -f c'est le mal !

GitFlow Model



Livraison continue (prod)

Validation client (beta)

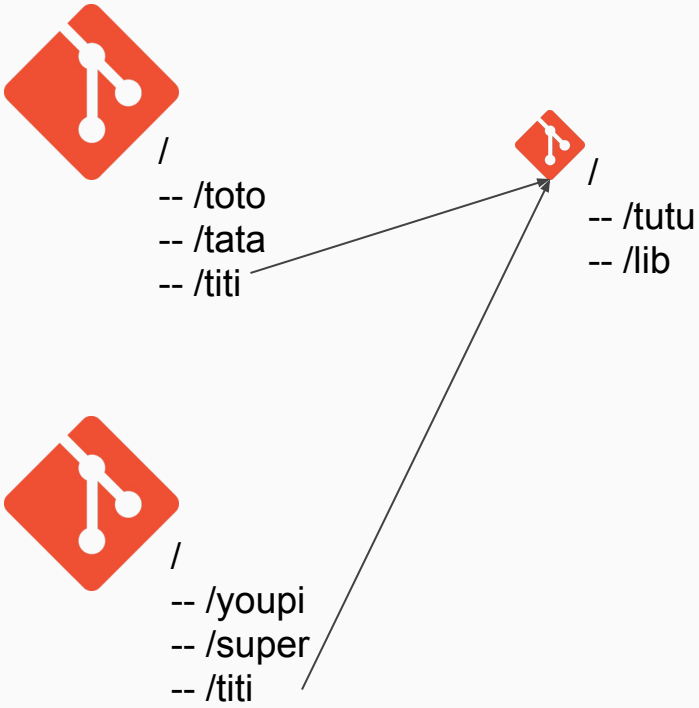
Intégration continue (alpha)

Nouvelle fonctionnalité

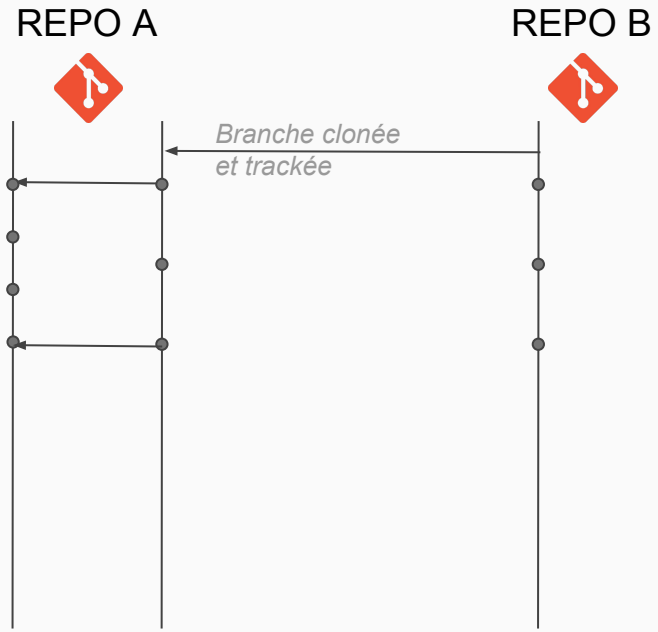
sudo apt-get install git-flow
git init
git flow init

Liens multi dépots (Rock&Roll Baby !)

SubModules *le “ln -s” de git*



SubTrees *le mirroring de Git*



Bien plus propre mais faut bien maitriser Git

GITHUB PullRequest Process

Comment ça marche chez GitHub