Introduction à Git

Alexandre Ancel

alexandre.ancel@ihu-strasbourg.eu
IHU Strasbourg - Institut de Chirurgie Guidée par l'Image / IRCAD

29/05/2017

Outline

Plan

Pourquoi les (C)VCS/DVCS ?

Gestion de versions :

- Centralisée: (C)VCS ou décentralisée: DVCS

But :

- Conserver l'historique d'un ensemble de fichiers en les versionnant
  - Travail sur les changesets (groupes de diffs)
  - Retourner à des versions antérieures (détection de bugs, ...)
  - Documenter les modifications (auteurs, ...)
  - Plus nécessaire : Conserver des copies de fichiers

Exemples :

- Code source d'une application (C/C++, scripts, html ...)
- Fichiers de configuration (dotfiles ...)
- Articles au format TEX
- ...

Collaboration

Aspect collaboratif :

- Possibilité de n'utiliser le (D)VCS qu'en mode mono-utilisteur
- Permet l'édition collaborative de documents avec gestion des conflits de fusion avancée

Aspect organisationnel / Gestion de projet :

- Plateformes construites autour (web)
- Tickets/Issues/Milestones
- Boards (Scrum, Kanban, . . .)

Aspect participatif / Communautaire :

- Merge/Pull requests (Github/Gitlab)

Différence (C)VCS/DVCS

VCS:

- Une seule copie centralisée d'un dépot
- Commiter = Enregistrer ses changements dans le système central

DVCS:

- Chaque "clone" a une copie complète du dépôt distant
  - Mode déconnecté : Les changements peuvent rester locaux
    - Tout sauf l'envoi (push) et la récupération (pull) peut etre fait localement.
  - Création de sauvegardes multiples
  - Rapidité d'accès aux modifications: Pas besoin d'accéder à un serveur, juste au disque
- Possibilité d'envoyer des changements indifférement vers plusieurs dépots distants
  - Faire tester une modification à quelqu'un avant de proposer les changements à tous
- Englobe la fonctionnalité "centralisée"

(C)VCS/DVCS Principaux

VCS :

- cvs (The CVS Team, Open Source, 1990),
  - Dernière release stable : 05/08
- Subversion (Apache, Open Source, 2000),
  - Dernière release stable : 11/16
- Perforce Helix (Perforce software, Propriétaire, 1995, DVCS depuis),
  - Dernière release stable : 06/16

DVCS :

- Mercurial, hg (Open Source, 2005),
  - Dernière release stable: 05/17
  - Facebook, W3C, Mozilla, Nginx, OpenJDK, Rhodecode, ...
- bazaar (Canonical/Community, Open Source, 2005),
  - Dernière release stable: 02/16
  - Ubuntu, Inkscape, ...

Git

DVCS:

- 1ère release: 07/04/2005 (12 ans)
- Dernière release stable: 10/05/2017
- Créé pour le kernel linux par Linus Torvalds, après retrait de l'utilisation gratuite de BitKeeper

Utilisateurs de git:

- git, kernel Linux, Google, Kitware, Communauté Open-Source (Github) ...

Plateforme d'hébergements:

- Web: Github, Bitbucket, Gitlab ...
- Auto-hébergeables:
  - Gitlab, Rhodecode, Gogs, ...
  - `git init --bare`

Fichiers binaires

Cas particulier des fichiers binaires: changesets

- Commit d'un fichier binaire : tout le contenu dans l'historique
- A chaque commit, le dépôt augmente de la taille du nouveau fichier

Solutions pour git :

- Fichier .gitignore : Ignorer des fichiers par regexp

```
bin/
*.o
```

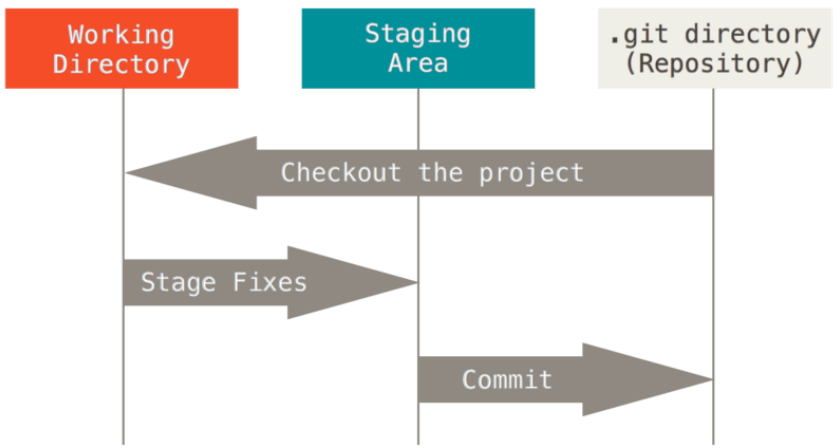
- Extensions: git lfs (Gitlab, Github), git annex

Plan

Quelques précisions [1]

- A chaque changement dans l'historique de git, un checksum unique SHA-1 est calculé (associé au commit, changement atomique dans l'index)
- Dossier .git dans le repertoire = votre dépôt local

- Les 3 états du dépôt local git:



[1] Source: https://git-scm.com, Licence: https://creativecommons.org/licenses/by/3.0, non modifié

Commande git

Apprendre par la ligne de commande (mécanismes sous-jacents),
Puis utiliser git avec des applications

```
$ git <command> option
```

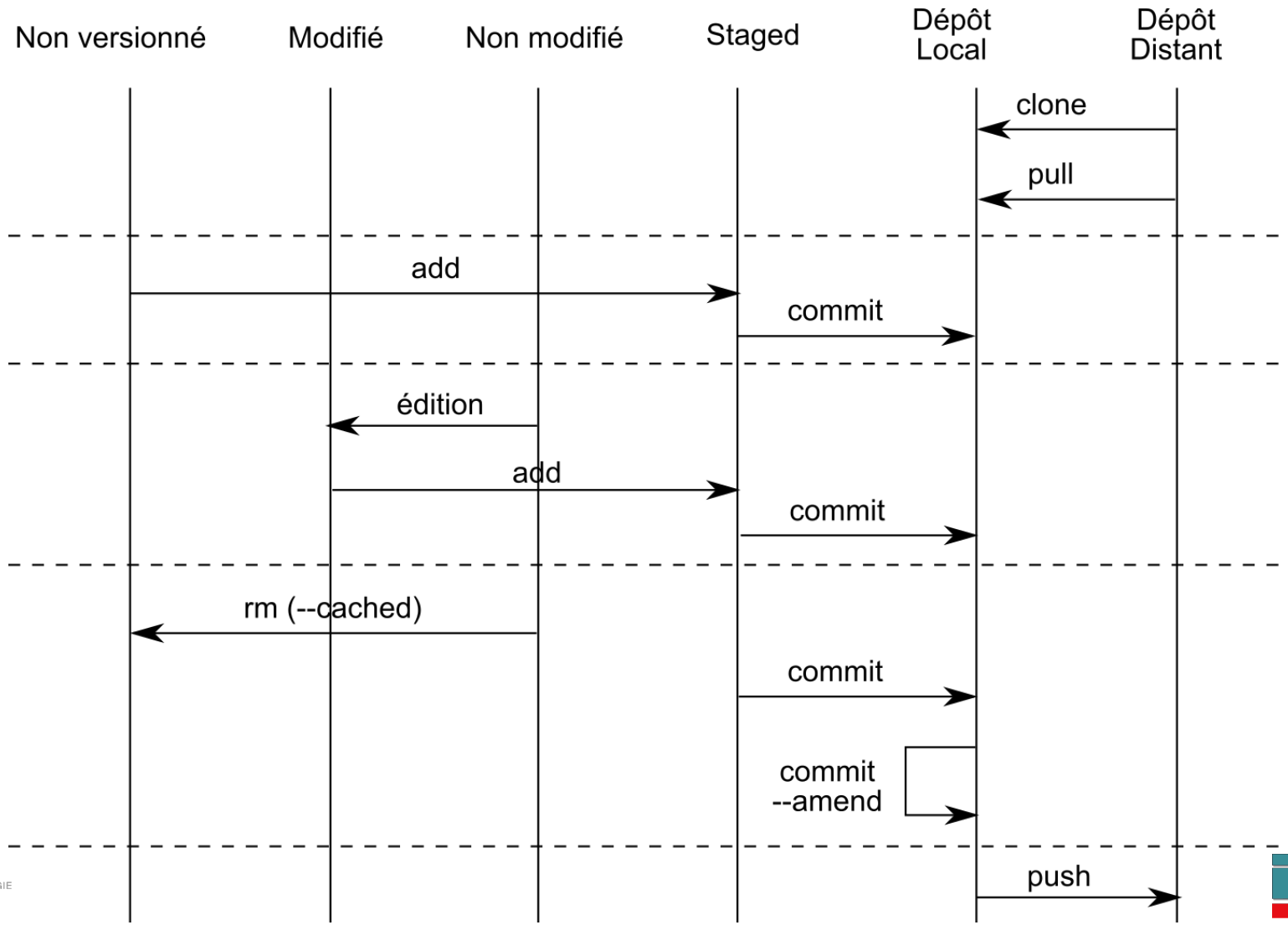
Commandes à voir:
`clone`, `pull`, `add`, `rm`, `commit`, `push`, `status`, `log`, `diff`, `revert`, `reset`, `branch`, `fetch`, `checkout`, `merge`, `rebase`, `remote`, `init`

Accès au dépôts: https ou ssh

Première chose à faire:

```
git config --global user.name "Prénom Nom"
git config --global user.email "nom.prenom@mail.com"
```

Commandes de base



Vérifier l'état d'un dépôt:

```
git status
```

```
Sur la branche develop
Votre branche est en retard sur 'origin/develop' de 3 commits, et peut être mise à jour en avance rapide.
  (utilisez "git pull" pour mettre à jour votre branche locale)
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git checkout -- <fichier>..." pour annuler les modifications dans la copie de travail)

        modifié :         applications/crb/CMakeLists.txt
        modifié :         applications/crb/heat3d/CMakeLists.txt
        modifié :         applications/crb/linearelasticity3d/CMakeLists.txt
        modifié :         applications/databases/pod_database.cpp
        modifié :         cmake/modules/Feel++Config.cmake.in

Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)

        applications/crb/cmake/

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
```

Vérifier l'état d'un dépôt:

git diff

```
diff --git a/CMakeLists.txt b/CMakeLists.txt
index 51319d6..ce22928 100644
--- a/CMakeLists.txt
+++ b/CMakeLists.txt
@@ -464,7 +464,7 @@ include(feelpp.directive)

 # The configuration File needs to be done at the very end otherwise we won't have the correc
 # The add_subdirectory is way too soon to get those values
-set(config_file "cmake/modules/Feel++Config.cmake")
-configure_file( ${config_file}.in ${config_file} @ONLY)
-INSTALL(FILES ${CMAKE_BINARY_DIR}/${config_file} DESTINATION share/feel/cmake/modules COMPON
-
+# The install command of this file is located in cmake/module/CMakeLists (because of feelpp-
+set(FEELPP_CONFIG_LIB_FILE ${FEELPP_BINARY_DIR}/cmake/modules/Feel++Config.cmake )
+configure_file(${FEELPP_SOURCE_DIR}/cmake/modules/Feel++Config.cmake.in ${FEELPP_CONFIG_LIB_
+configure_file(${FEELPP_SOURCE_DIR}/cmake/modules/feelpp.install.config.cmake.in ${FEELPP_BI
diff --git a/cmake/machines/CMakeLists.txt b/cmake/machines/CMakeLists.txt
index bf7ce71..1f5ad52 100644
--- a/cmake/machines/CMakeLists.txt
+++ b/cmake/machines/CMakeLists.txt
@@ -24,4 +24,4 @@
 #
 FILE(GLOB files "${CMAKE_CURRENT_SOURCE_DIR}/*.cmake")

-INSTALL(FILES ${files} DESTINATION share/feel/cmake/machines COMPONENT Devel)
+INSTALL(FILES ${files} DESTINATION share/feelpp/feel/cmake/machines COMPONENT Devel)
diff --git a/cmake/modules/CMakeLists.txt b/cmake/modules/CMakeLists.txt
index 5628584..30b6b7b 100644
--- a/cmake/modules/CMakeLists.txt
+++ b/cmake/modules/CMakeLists.txt
@@ -50,3 +50,4 @@ FILE(GLOB modules "${CMAKE_CURRENT_SOURCE_DIR}/*.cmake")

 INSTALL(FILES ${modules} DESTINATION share/feelpp/feel/cmake/modules COMPONENT Devel)
 INSTALL(FILES ${CMAKE_CURRENT_BINARY_DIR}/Feel++Config.cmake DESTINATION share/feelpp/feel/c
+INSTALL(SCRIPT ${CMAKE_CURRENT_BINARY_DIR}/feelpp.install.config.cmake COMPONENT Devel)
```

git log

```
* 6b5dd76 (HEAD -> develop, origin/develop, origin/HEAD) move back to 0.104.0 [ci skip]
* a36fc7a (tag: v0.103.2, origin/master) v0.103.2 [ci skip]
*   17915d0 Merge branch 'develop'
|\
| * 0075fdc up docker tags
| * 4e9b58a use file( INSTALL )  : tentative to fix make install (@Trophime)
| * fec268d remove FEELPP_BINARY_BUILD_DIR and FEELPP_SOURCE_BUILD_DIR of Feel++Config.cmake
| * be8e5cf add feelpp subdir
| * c5f8d76 fix feel++ installation with make install
| * ba23cd1 add FEELPP_PARAVIEW_DIR or FEELPP_VTK_DIR in Feel++Config.cmake
| *   a2845c1 Merge branch 'master' into develop
| |\
| * | dc6147c (tag: v0.103.0) bump up version from v0.103.0 to v0.104.0 [ci skip]
* | | 489d9ce up [ ci skip]
| |/
|/|
* | f2a2a4a up github relase and updload scripts [ci skip]
* | e1b63a9 (tag: v0.103.1) do not add testsuite deps if it does not exist fixed #903
* | e4cd3c2 up release script [ci skip]
|/
* ade3f1c fix inconsistent override compilation warning
* 5a115db fix sub-projects compilation
* 420e107 minimum is 3.1.3 now
*   81e1715 Merge branch 'develop' of https://github.com/feelpp/feelpp into develop
|\
| *   759ce85 Merge branch 'develop' of https://github.com/feelpp/feelpp into develop
| |\
| | * d327af9 fix compilation of integrator::evaluate(  std::vector<Eigen::Matrix<T, M,N> ) w
| | * 4809a67 Update pipeline.yml
| * | 2394ae8 install headers of toolboxes/feel/modelvf/
| * | 306b0e8 up gls stab :
| |/
| * f291f0d use const* instead of const& : allow to have a copy assignment operator which are
* | 8f85ea1 use CM_MAKE_CXX_STANDARD to define clang/gcc -std option
|/
```

Annuler des changements [2]

Reset:

- `git reset <fichier>`: Annuler l'état "Staged" d'un fichier

Checkout:

- `git checkout <fichier>`: Récupère la dernière version commitée d'un fichier (Modifications perdues)

Revert:

- Annule un commit (avec un nouveau commit)
- `git revert <commit>`
- Ne change pas l'historique : OK pour les commits publics

[2] Source: https://www.atlassian.com/git, Licence: https://creativecommons.org/licenses/by/2.5/au/, non modifié

Plan

Branches dans git [3]

Facilité de créations de branches

- Développement en parallèle de la branche principale
- Réintégration des modifications avec opérateur de fusion

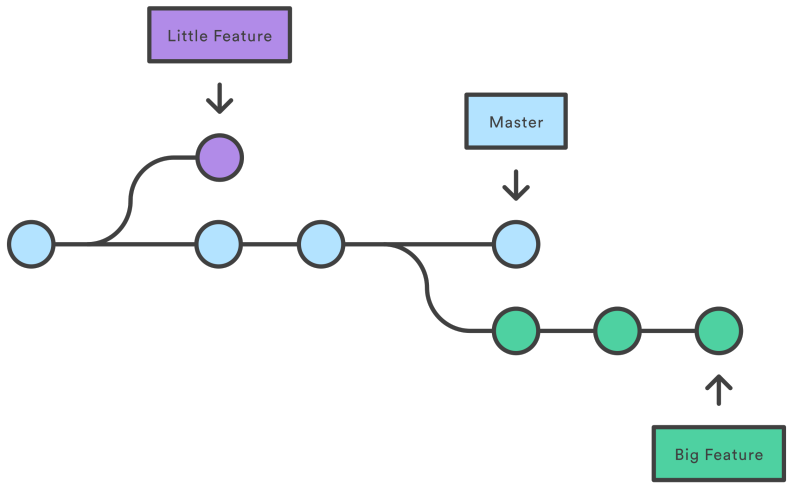
Modèles de branches:

- master, dev(evelop)
- Conventions de nommage: feat/newExporter, fix/bug ...
- Branches de sprint (Une histoire par branche)

[3] Source: https://git-scm.com, Licence: https://creativecommons.org/licenses/by/3.0, non modifié

Gestion des branches [4]

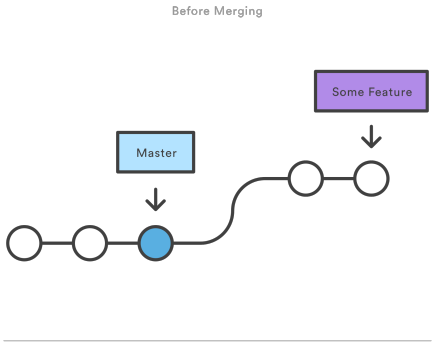
- git branch (-vvv): Lister/créer/supprimer des branches
- git fetch: Télécharger des branches depuis un autre dépôt
- git checkout: Changement de branche
  - git checkout -b <branch>: Créer et changer de branche
- git log –oneline –graph –decorate –all: Afficher les branches

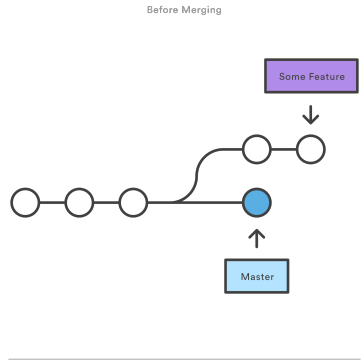


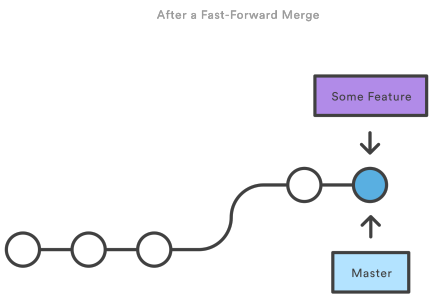
[4] Source: https://www.atlassian.com/git, Licence: https://creativecommons.org/licenses/by/2.5/au/, non modifié

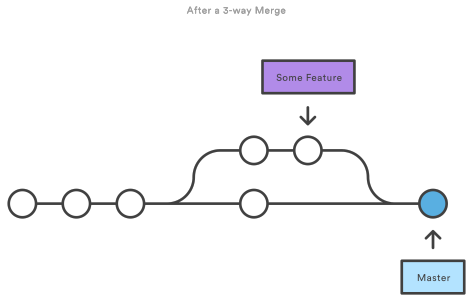
Fusions et type de fusion [5]

- git merge (git pull = git fetch + git merge)
  - Fusion transparente: Fast forward (par défaut)
  - Fusion marquée: Merge classique









~~Fast-forward~~ 3-way merge

[5] Source: https://www.atlassian.com/git, Licence: https://creativecommons.org/licenses/by/2.5/au/, non modifié

Fusions et gestion des conflits

- Lors des étapes de fusion: possibilité de conflits de fusion

```
Auto-merging main.c
CONFLICT (content): Merge conflict in main.c
Automatic merge failed; fix conflicts and then commit the result.
```

- Editer les fichiers en conflits

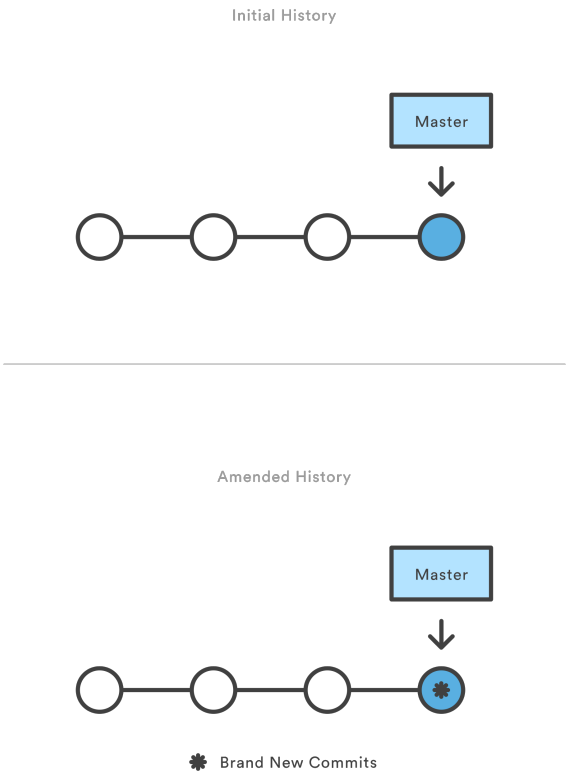
```
#include <mpi.h>

<<<<<<< HEAD
void main()
=======
int main(int argc, char ** argv)
>>>>>>> branch-a
```

- `git add <fichier> && git commit`

Réécrire l'histoire ? [6]

N'utilisez ces commandes que sur des commits locaux !

git reset, git commit –amend

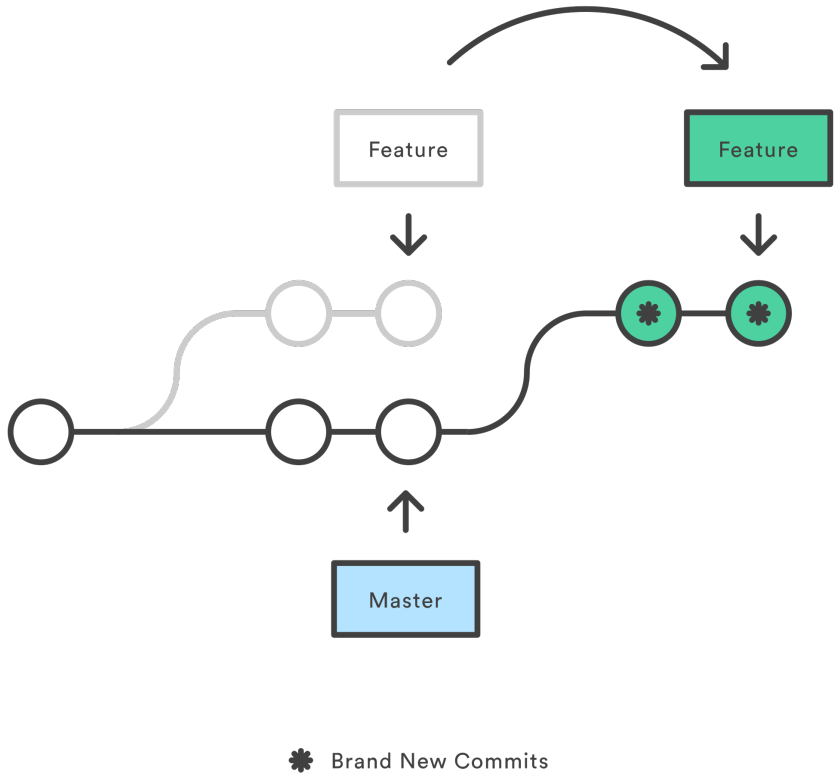


[6] Source: https://www.atlassian.com/git, Licence: https://creativecommons.org/licenses/by/2.5/au/, non modifié

Réécrire l'histoire ? [7]

Mais surtout: git rebase

- Rejouer une suite de commit sur le bout d'une branche



[7] Source: https://www.atlassian.com/git, Licence: https://creativecommons.org/licenses/by/2.5/au/, non modifié

Tags

Git permet de tagger des commits:

- git tag: Liste les tags disponibles
- git tag <tagname> <commit>: Créé un nouveau tag sur un commit
- git push <tagname>: Envoyer le tag vers le dépôt distant
- git push –tags: Envoyer tous les tags

Utile pour la CI, permet d'automatiser la création de releases

Plan

Remotes

Remote = version du dépôt stocké dans un dossier local, une autre machine ou Internet

- https: `https://github.com/fw4spl-org/fw4spl.git`
- ssh: git@github.com:fw4spl-org/fw4spl.git
- Repertoire local: /home/aancel/git/fw4spl-bare

Gestion des remotes

- git remote -v: Liste des remote disponibles
- git remote add: Ajout un nouveau remote pour le dépôt
- git push <remote> <branch>: Pousser une branche vers un remote

Dépôt "bare"

- git init: Créer un dépôt de travail
- git init –bare: Créer un dépôt de partage

Pratique

Exemple: Vous utilisez un cluster de calcul qui n'autorise que l'accès SSH depuis l'extérieur.

Comment accéder à votre code versionné sur cette machine ?

- Créer un dépôt bare sur la frontale (Utilisez un dossier de votre disque pour simuler cela)
- Depuis votre dépôt local, ajouter le remote correspondant
- Pusher le code vers le dépôt bare
- Récupérer votre code dans un autre dossier à partir du dépôt bare

Plan

Merge-Request / Pull-Request

Dans quel cas utiliser les MR/PR ? (aspect participatif)

- Pas de droits d'accès en écriture (push) au dépôt
- Proposer les changements que vous avez fait

Merge-Requests (Gitlab) / Pull-Requests (Github):

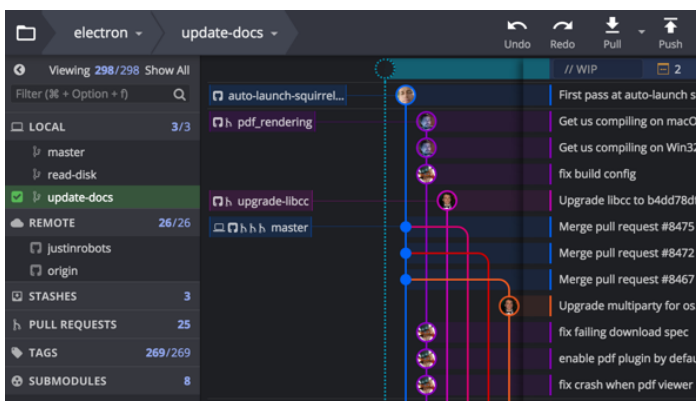
- Fork : Creation d'une copie à un instant t d'un dépot
- Travail sur le contenu (texte, fichiers sources . . .)
- (Optionnel) Resynchronisation avec le repoository d'origine
- Creation d'une Merge/Pull Request sur le dépot original
  - Discuter d'une feature, faire de la revue de code . . .
  - Référencer les futurs commits faits sur la branche pour discussion
- Merge (classique généralement)

Plan

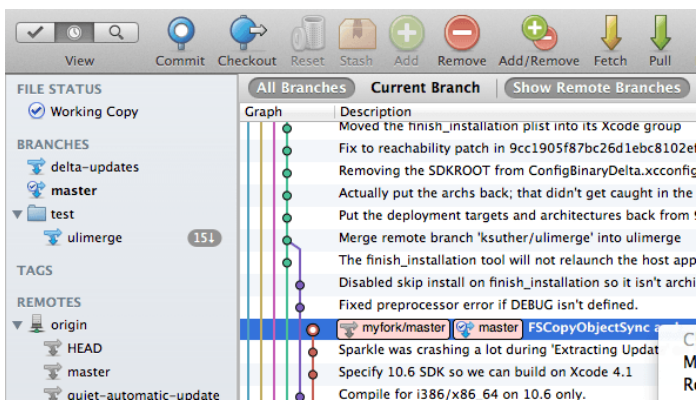
Applications dédiées [8]

`https://git-scm.com/downloads/guis`:

- tig (OS X, Linux)
- gitk (Windows, OS X, Linux)
- Github Desktop (Windows, OS X)
- SmartGit (Windows, OS X, Linux)
- TortoiseGit (Windows)



GitKraken (Windows, OS X, Linux)



SourceTree (OS X, Windows)

[8] Source: `https://git-scm.com`, Licence: `https://creativecommons.org/licenses/by/3.0`, non modifié

Ligne de commande / Intégration

- Ligne de commande : git
- Intégration :
  - Vim
    - Fugitive: `https://github.com/tpope/vim-fugitive`
    - git-gutter: `https://github.com/airblade/vim-gitgutter`
  - Emacs
    - Magit: `https://magit.vc`, `https://github.com/magit/magit`
    - git-gutter: `https://github.com/syohex/emacs-git-gutter`

Plan

Conclusion

Git:

- DVCS (déconnecté, rapidité d'accès aux méta-données, ...)
- Intégration dans de nombreux outils
- De (trop ?) nombreuses commandes sont disponibles
  - Certaines ont des utilisations simialaires, permettent de faire les même chose (potentiellement troublant)
- Attention cependant aux modifications d'historique quand vous travaillez de manière collaborative

Références:

- `https://git-scm.com`
- `https://www.atlassian.com/git/tutorials`
- `man git-<command>`

Jeux:

- `https://try.github.io`