Versioning Git

Sommaire

- Les bases de Git
- Git remote
- GitHub
- Pratiques du code

Prérequis

- Univers UNIX
- Système de fichiers UNIX

Qu’est ce que Git?

- open source
- versionning

Parents


SUBVERSION


mercurial


BitKeeper
Scalable Version Control

A quoi ça sert?

- repository
- gestion des anciennes versions
- parallélisation (branches)

Installation de Git

```
git version
```

```
# Debian/Ubuntu
apt install git
#Arch
pacman -S git
#Alpine
apk add git
#MacOS
brew install git
```

Initialisation d’un repository

```
mkdir myProject
cd myProject
git init
```

Suppression d’un repository

git config

Il existe 3 fichiers git config sur votre machine:

- [chemin]/gitconfig aussi appelé system, [chemin] étant équivalent à l'endroit où git est installé
- ~/.gitconfig aussi appelé global
- .git/config aussi appelé local

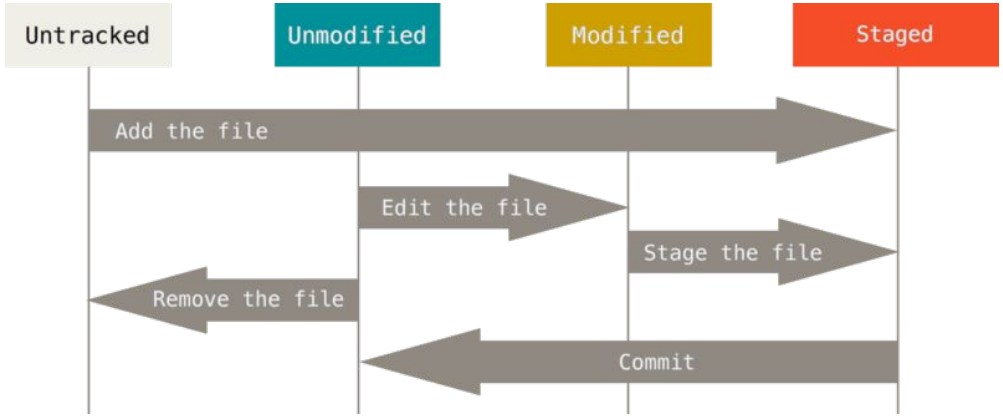
git config

```
git config --global user.name "John Doe" # Déclare votre nom dans la config global
git config --global user.email johndoe@example.com # Déclare votre mail dans la config global
```

Le principe de statut dans Git

Les différents statuts:

- Untracked
- Unmodified
- Modified
- Staged



Le principe de statut dans Git

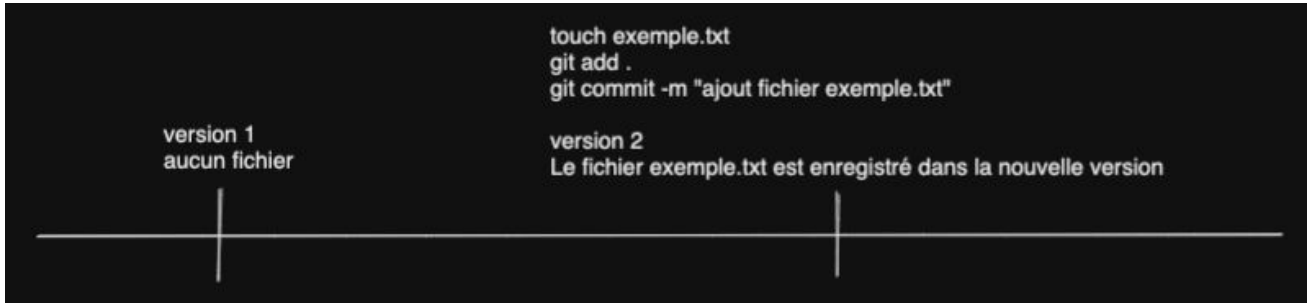
```
~/Documents/cours  git init
Initialized empty Git repository in /Users/quentin/Documents/cours/.git/
~/Documents/cours  touch exemple.txt
~/Documents/cours  git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        exemple.txt

nothing added to commit but untracked files present (use "git add" to track)
~/Documents/cours  git add exemple.txt
~/Documents/cours  git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   exemple.txt

~/Documents/cours  git commit -m "mon premier commit"
[master (root-commit) d15c1d0] mon premier commit
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 exemple.txt
~/Documents/cours  git status
On branch master
nothing to commit, working tree clean
~/Documents/cours  vim exemple.txt
~/Documents/cours  git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   exemple.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

git add

```
git add exemple.txt # je veux ajouter les modifications de mon fichier exemple.txt
git add . # je veux ajouter toutes les modifications de tous mes fichiers untracked ou modified
```

git commit

```
git commit -m "mon commit" # je sauvegarde les nouvelles versions de tous les fichiers staged avec le message de commit "mon commit"
git commit -am "mon commit" # je sélectionne tous les fichiers au statut modified pour les mettre au statut staged puis je sauvegarde les nouvelles versions de tous les fichiers staged avec le message de commit "mon commit"
```



git commit --amend

```
# Modifie le précédent commit
git commit --amend
```

```
mon premier commit

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Sat Aug 28 13:27:34 2021 +0200
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   exemple.txt
#
```

Écrire un bon message de commit

Référence : https://greg0ire.fr/git-gud/

Première ligne : sujet, jusqu'à 50 caractères

Seconde ligne : vide <= obligatoirement

Troisième ligne et plus : description longue des changements

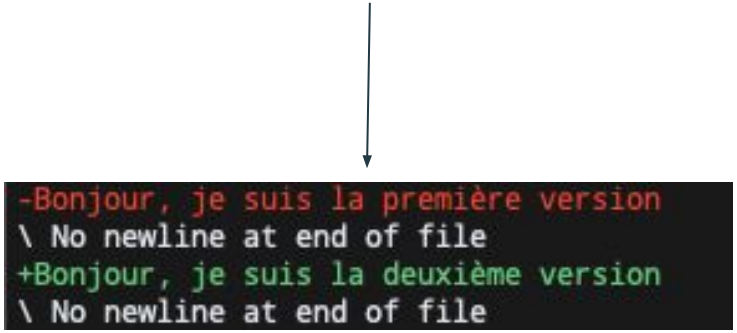
Exemple de message de commit

Référence : https://chris.beams.io/posts/git-commit/

```
Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72
characters or so. In some contexts, the first line is treated as the
subject of the commit and the rest of the text as the body. The
blank line separating the summary from the body is critical (unless
you omit the body entirely); various tools like `log`, `shortlog`
and `rebase` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you
are making this change as opposed to how (the code explains that).
Are there side effects or other unintuitive consequences of this
change? Here's the place to explain them.

Further paragraphs come after blank lines.

 - Bullet points are okay, too

 - Typically a hyphen or asterisk is used for the bullet, preceded
   by a single space, with blank lines in between, but conventions
   vary here

If you use an issue tracker, put references to them at the bottom,
like this:

Resolves: #123
    See also: #456, #789
```

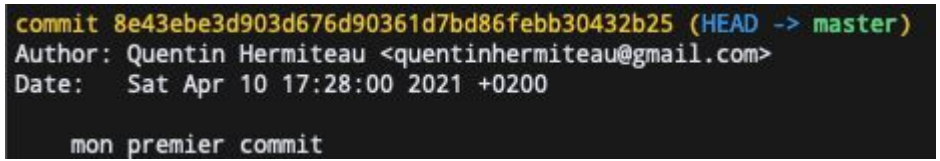
git diff

```
# git diff montre montre les différences entre la version de git et la vôtre ...
git diff exemple.txt # .sur le fichier exemple. txt
git diff # sur tous les fichiers au statut modified
```



git log

```
git log # Affiche l'historique des commits sur notre repository
```



Options de git log

L’intérêt d’écrire des messages de commit conformes aux bonnes pratiques est de pouvoir profiter de différents affichages de l’historique:

Afficher les commits de toutes les branches avec --all:
git log --all

Afficher uniquement les titres des commits avec --oneline:
git log --oneline

Afficher un graph représentant les branches avec --graph:
git log --all --oneline --graph

Afficher le résumé des changements de chaque commit avec --stat:
git log --all --oneline --stat

```
commit 26da02c54053047f2a767e1104d1c86fc8f6a29e (HEAD -> master)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 10 17:36:30 2021 +0200

    mon deuxième commit

commit 8e43ebe3d903d676d90361d7bd86febb30432b25
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 10 17:28:00 2021 +0200

    mon premier commit
```

```
~/Documents/cours > git log
~/Documents/cours > git checkout 8e43ebe3d903d676d90361d7bd86febb30432b25
Note: switching to '8e43ebe3d903d676d90361d7bd86febb30432b25'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 8e43ebe mon premier commit
~/Documents/cours > cat exemple.txt
Bonjour, je suis la première version
```

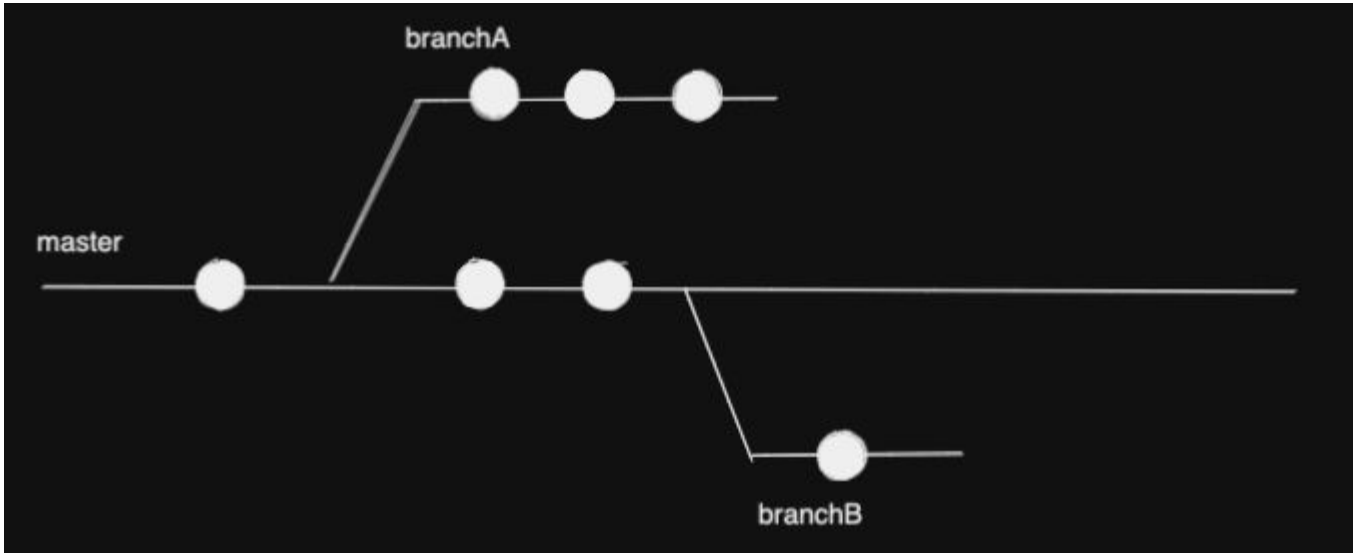
git checkout

```
commit 8e43ebe3d903d676d90361d7bd86febb30432b25 (HEAD)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 10 17:28:00 2021 +0200

    mon premier commit
```

```
~/Documents/cours git checkout master
Previous HEAD position was 8e43ebe mon premier commit
Switched to branch 'master'
```

```
commit 26da02c54053047f2a767e1104d1c86fc8f6a29e (HEAD -> master)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 10 17:36:30 2021 +0200

    mon deuxième commit

commit 8e43ebe3d903d676d90361d7bd86febb30432b25
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 10 17:28:00 2021 +0200

    mon premier commit
```

git branch



git branch

```
git branch # Donne la liste des branches qui ont été crées
git branch branchA # Créé la branche branchA si elle n'existe pas déjà
git checkout branchA # Se déplacer sur la branche branchi si elle existe
git checkout -b branchA # Créé la branche branchA et se déplace dessus
git checkout - B branchA # Créé ou reset la branche branchA et se déplace
dessus
```

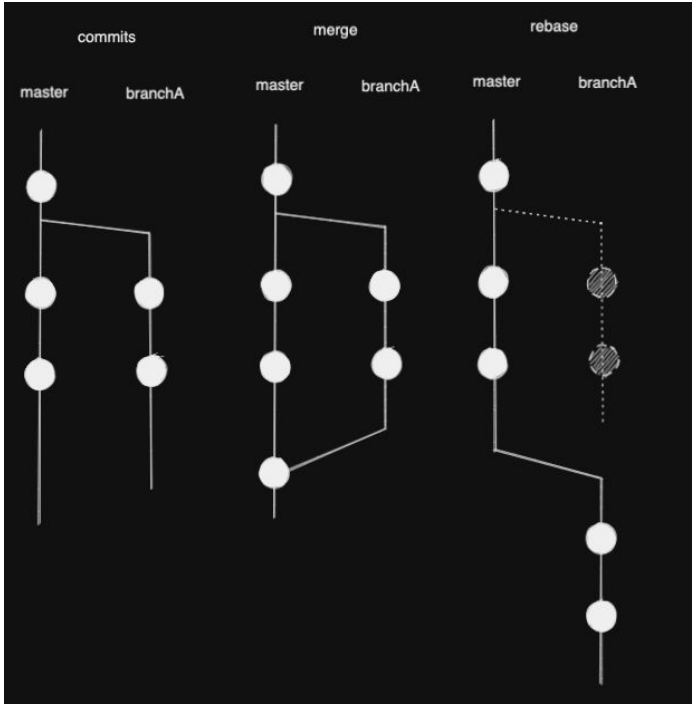
git branch

```
git branch brachA # Erreur dans le nom de ma branche
git branch -m brachA branchA # Correction du nom de la branche
git branch -d branchA # Supprime la branche
git branch -D branchA # Force la suppression de la branche
```

C'est à vous

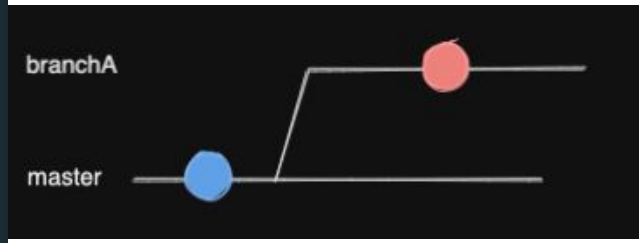
Manipuler les commandes qu'on vient de voir

La différence entre git merge et git rebase



git merge

```
git checkout master # je me dirige sur la branche dans laquelle je veux merge des modifications
git merge branchA # J'indique depuis quelle branche je veux récupérer les modifications
```

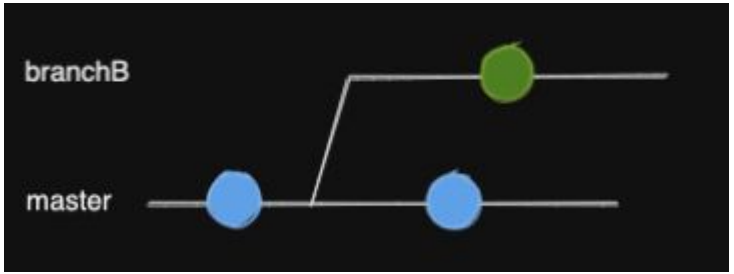



```
commit 26da02c54053047f2a767e1104d1c86fc8f6a29e (HEAD -> master)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 10 17:36:30 2021 +0200

    mon deuxième commit

commit 8e43ebe3d903d676d90361d7bd86febb30432b25
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 10 17:28:00 2021 +0200

    mon premier commit
```

```
~/Documents/cours  git merge master branchA
Updating 61b2df1..c1542da
Fast-forward
 test.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test.txt
```

```
commit c1542da75318b708d626ad16878b8ceb31e218a4 (HEAD -> master, branchA)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 11:25:33 2021 +0200

    mon commit depuis la branche branchA

commit 61b2df15f1cd27b2d7d982bed641a9941d8e63e9
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 11:25:05 2021 +0200

    mon deuxième commit

commit 3f25a8203f0b4636ee35d6b51a8c52724fc90596
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 11:24:38 2021 +0200

    mon premier commit
```

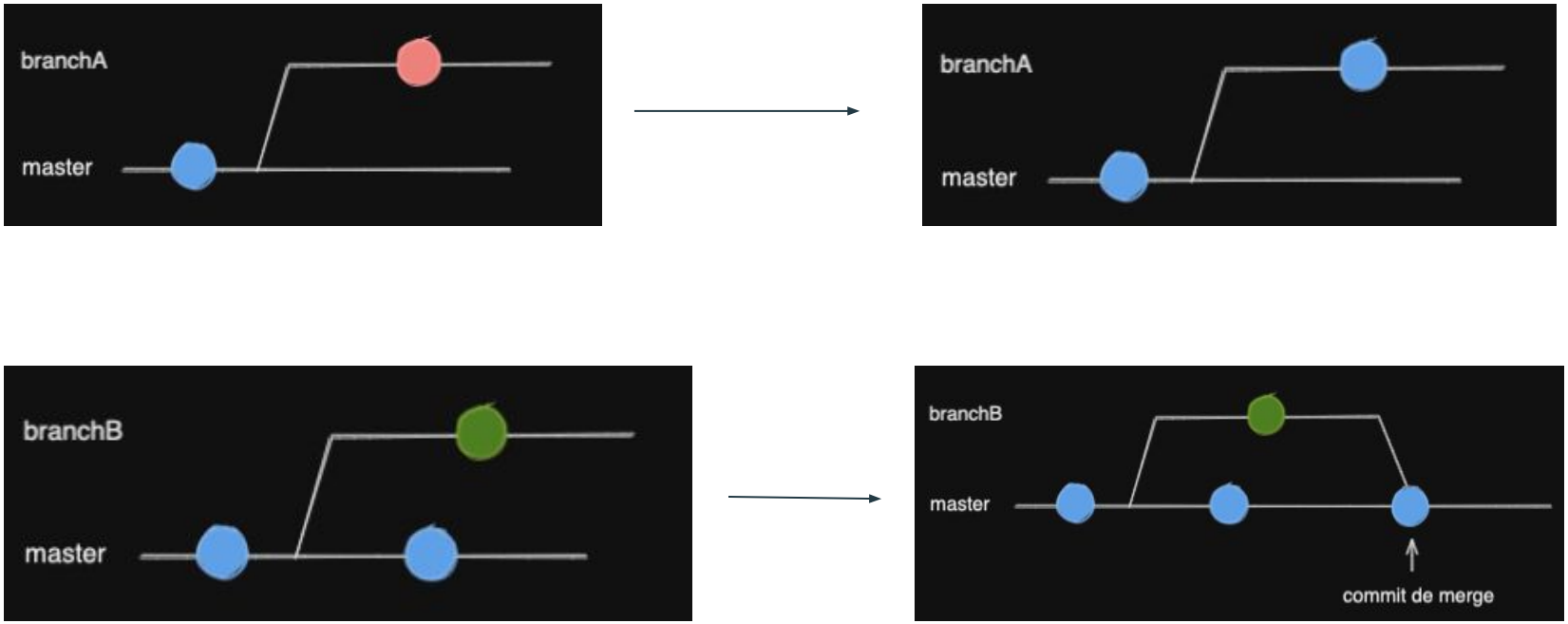
git merge




```
Merge branch 'branchB'
Ajout des modifications de branchB sur master
# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
```

```
commit e33145ff052e2e9950e03812e78fd90e56e4c7ea (HEAD -> master)
Merge: db2d9b8 da2dd2d
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 16:40:08 2021 +0200

    Merge branch 'branchB'
    Ajout des modifications de branchB sur master

commit da2dd2de69ec7a8f087002f5233d5febe36ea24f (branchB)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 16:39:50 2021 +0200

    mon commit depuis la branche branchB

commit db2d9b837a70a1a993ed63d8eabe1f2cb61a7c23 (branchA)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 16:39:30 2021 +0200

    mon commit depuis la branche branchA

commit 5a2f1e23cac33969d5969e0b72513180c60b188d
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 16:39:11 2021 +0200

    mon deuxième commit

commit 0b58fbc200f6b0d1bf0d8a9a85ede82a245c969f
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 16:38:48 2021 +0200

    mon premier commit
```

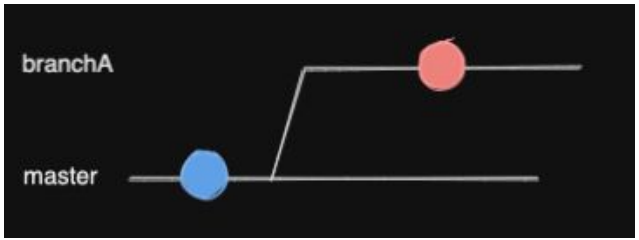
git merge


branchA
master
branchA
master
branchB
master
branchB
master
commit de merge

git rebase

```
git checkout branchA # je me dirige sur la branche dans laquelle je veux rebase des modifications
git rebase main # J'indique depuis quelle branche je veux récupérer les modifications
```

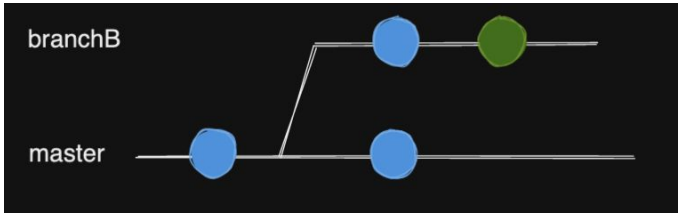
git rebase




```
commit d61677c27df939dcfd0f6bb5e9ca65daf8b29067 (HEAD -> master, branchA)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 17:09:38 2021 +0200

    mon commit depuis la branche branchA

commit b7a5763372719c555ec90523bbbffd71d8488ed4
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 17:09:10 2021 +0200

    mon deuxième commit

commit cebb36fce41b42b78b81fd7428c0fbaa53a9bf1e
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 17:08:52 2021 +0200

    mon premier commit
```

```
~/Documents/cours git rebase branchA
First, rewinding head to replay your work on top of it...
Fast-forwarded master to branchA.
```

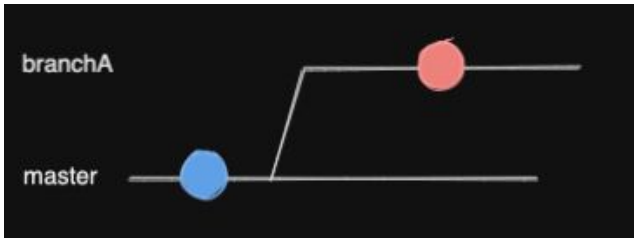
git rebase

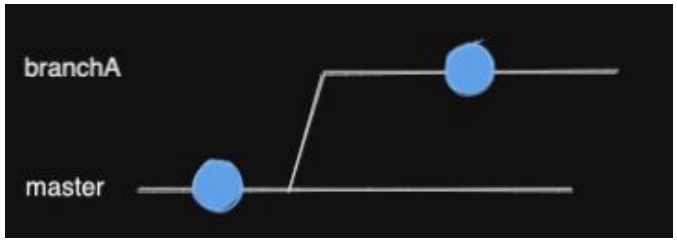


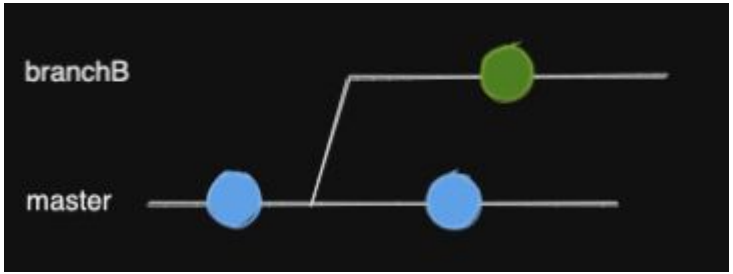

```
~/Documents/cours git rebase branchB
First, rewinding head to replay your work on top of it...
Applying: mon commit depuis la branche branchA
```

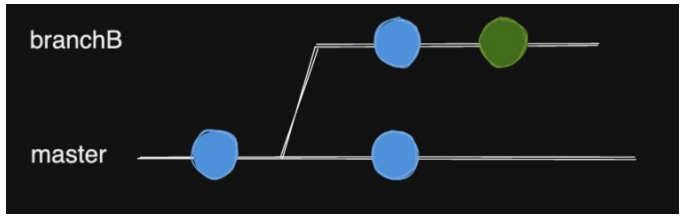
```
commit a7b892987c28e15d8d5060beeaa22b2c4b74a083 (HEAD -> master)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 17:09:38 2021 +0200

    mon commit depuis la branche branchA

commit 90ac150cd09ec18100dac3762f7dc8390dccb5f3 (branchB)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 17:09:51 2021 +0200

    mon commit depuis la branche branchB

commit b7a5763372719c555ec90523bbbffd71d8488ed4
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 17:09:10 2021 +0200

    mon deuxième commit

commit cebb36fce41b42b78b81fd7428c0fbaa53a9bf1e
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Sat Apr 17 17:08:52 2021 +0200

    mon premier commit
```

git rebase


branchA
master


branchA
master


branchB
master


branchB
master

git rebase intéractif

```
git rebase -i master # lance une session de rebase intéractif
git rebase --interactive master # autre façon d'écrire la
commande
```

git rebase intéractif

```
pick 7f9d889 ajout d'un fichier
pick 31f0f6a modi d'un fichier
pick 5cc6959 fix d'un fichier

# Rebase c224850..5cc6959 onto c224850 (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
```

```
pick 7f9d889 ajout d'un fichier
reword 31f0f6a modi d'un fichier
drop 5cc6959 fix d'un fichier

# Rebase c224850..5cc6959 onto c224850 (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
```

git rebase intéractif

```
commit 39d8767f0678dd60f6e6e1cdaaa39bc862105b5c (HEAD -> branchC)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Apr 19 16:39:17 2021 +0200

    modification d'un fichier

commit 858415229b10ff9c7092956bb59491313f0aefd5
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Apr 19 16:39:07 2021 +0200

    ajout d'un fichier

commit c22485037ab5c9d33ace7407a0fc36bf23b8e1eb (master)
Merge: 9c155f5 512d761
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Apr 19 16:24:28 2021 +0200

    Merge branch 'branchB'
```

git rebase intéractif

```
pick 7f9d889 ajout d'un fichier
squash 31f0f6a modi d'un fichier
squash 5cc6959 fix d'un fichier

# Rebase c224850..5cc6959 onto c224850 (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
~
```

```
# This is a combination of 3 commits.
# This is the 1st commit message:

ajout d'un fichier

# This is the commit message #2:

modification d'un fichier

# This is the commit message #3:

fix d'un fichier

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# Date:      Mon Apr 19 16:58:42 2021 +0200
#
# interactive rebase in progress; onto c224850
# Last commands done (3 commands done):
#    squash 889737c modi d'un fichier
#    squash 53af5cf fix d'un fichier
# No commands remaining.
# You are currently rebasing branch 'branchC' on 'c224850'.
#
# Changes to be committed:
#       new file:   tutu.txt
#
~
~
```

git rebase intéractif

```
commit 97d581cce6fdd48945b151a5d1c7244074064ef7 (HEAD -> branchC)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Apr 19 16:58:42 2021 +0200

    ajout d'un fichier

    modification d'un fichier

    fix d'un fichier

commit c22485037ab5c9d33ace7407a0fc36bf23b8e1eb (master)
Merge: 9c155f5 512d761
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Apr 19 16:24:28 2021 +0200

    Merge branch 'branchB'
```

git rebase intéractif

```
reword 7f9d889 ajout d'un fichier
fixup 31f0f6a modi d'un fichier
fixup 5cc6959 fix d'un fichier

# Rebase c224850..5cc6959 onto c224850 (3 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
# Note that empty commits are commented out
```

```
commit 85412d1006274176c881b1f3835b29a33ac67bdd (HEAD -> branchC)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Apr 19 17:19:34 2021 +0200

    ajout d'un fichier

commit c22485037ab5c9d33ace7407a0fc36bf23b8e1eb (master)
Merge: 9c155f5 512d761
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Apr 19 16:24:28 2021 +0200

    Merge branch 'branchB'
```

Régler des conflits

```
~/Documents/esgi/cours/solution de versionning  branchB  git rebase master
Auto-merging exemple.txt
CONFLICT (content): Merge conflict in exemple.txt
error: could not apply b200c80... Modification fichier exemple branchB
Resolve all conflicts manually, mark them as resolved with
"git add/rm <conflicted_files>", then run "git rebase --continue".
You can instead skip this commit: run "git rebase --skip".
To abort and get back to the state before "git rebase", run "git rebase --abort".
Could not apply b200c80... Modification fichier exemple branchB
~/Documents/esgi/cours/solution de versionning  branchB | rebase-i
```

Régler des conflits

```
<<<<<<< HEAD
Bonjour
=======
Aurevoir
>>>>>>> b200c80 (Modification fichier exemple branchB)
```

Régler des conflits

```
Aurevoir
~
```

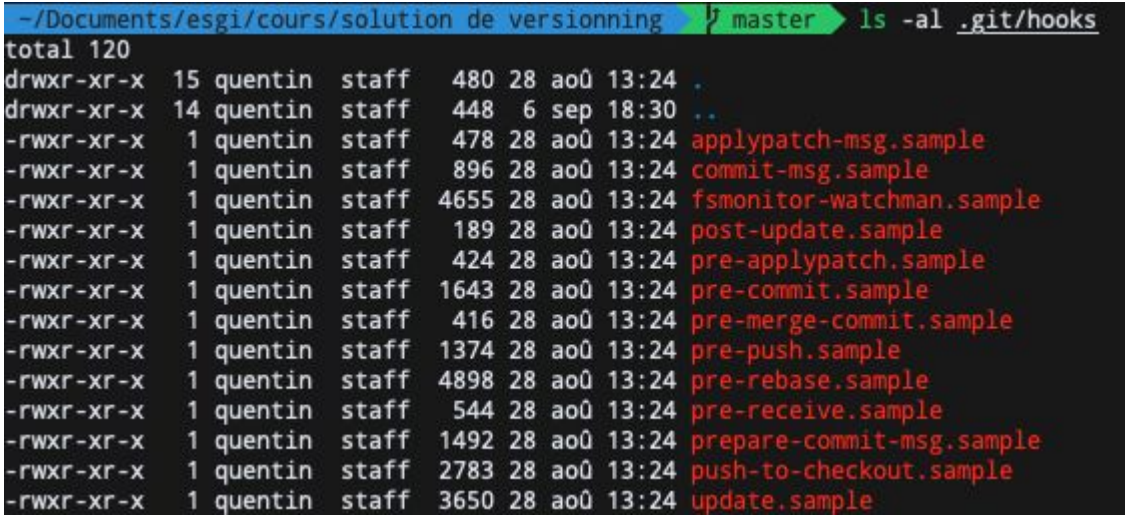
```
~/Documents/esgi/cours/solution de versionning  branchB | rebase-i  vim exemple.txt
~/Documents/esgi/cours/solution de versionning  branchB | rebase-i  git add exemple.txt
~/Documents/esgi/cours/solution de versionning  branchB | rebase-i  git rebase --continue
[detached HEAD 468dabb] Modification fichier exemple branchB
 1 file changed, 1 insertion(+), 1 deletion(-)
Successfully rebased and updated refs/heads/branchB.
~/Documents/esgi/cours/solution de versionning  branchB
```

TP

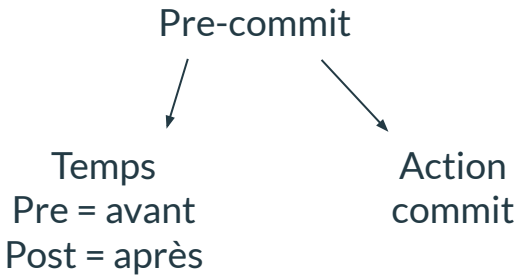
- git clone le repo https://github.com/quentinhermiteau/tp-merge-rebase
- Récupérer les différentes branch avec git checkout
- Récupérer les modifs de feature/hello sur main
- Récupérer les modifs de feature/calc sur main
- Obtenir l'historique suivant

```
commit 821f4c5f920ff7613cecbf364a01954cf114836f (HEAD -> main, feature/calc)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Jan 24 12:18:38 2022 +0100

    Ajout de la fonction pull

commit 87d4ba0e0f0157ddfc7d1b22ffe5d9c90a0998f4
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Jan 24 12:17:25 2022 +0100

    Ajout du fichier utils

    Ajout de la fonction add

commit dafe9ddd095c485c85d770148bcf529674eb1745
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Jan 24 12:02:27 2022 +0100

    update index function

commit c466bfd0acf749be4e2f3341ad6b17c925be49ef (feature/hello)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Jan 24 12:00:58 2022 +0100

    Ajout de la fonction hello

commit 375dcf913d120bb272832464960c768d211865c4 (origin/main, origin/HEAD)
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Jan 24 12:02:08 2022 +0100

    Création de la classe MyClass

commit f1b8872b7fc6d175962ae6d0157aff65da81d657
Author: Quentin Hermiteau <quentinhermiteau@gmail.com>
Date:   Mon Jan 24 11:59:56 2022 +0100

    :tada: initial commit
```

Git hooks


~/Documents/esgi/cours/solution de versionning
master
ls -al .git/hooks
total 120
drwxr-xr-x 15 quentin staff 480 28 aoû 13:24 .
drwxr-xr-x 14 quentin staff 448 6 sep 18:30 ..
-rwxr-xr-x 1 quentin staff 478 28 aoû 13:24 applypatch-msg.sample
-rwxr-xr-x 1 quentin staff 896 28 aoû 13:24 commit-msg.sample
-rwxr-xr-x 1 quentin staff 4655 28 aoû 13:24 fsmonitor-watchman.sample
-rwxr-xr-x 1 quentin staff 189 28 aoû 13:24 post-update.sample
-rwxr-xr-x 1 quentin staff 424 28 aoû 13:24 pre-applypatch.sample
-rwxr-xr-x 1 quentin staff 1643 28 aoû 13:24 pre-commit.sample
-rwxr-xr-x 1 quentin staff 416 28 aoû 13:24 pre-merge-commit.sample
-rwxr-xr-x 1 quentin staff 1374 28 aoû 13:24 pre-push.sample
-rwxr-xr-x 1 quentin staff 4898 28 aoû 13:24 pre-rebase.sample
-rwxr-xr-x 1 quentin staff 544 28 aoû 13:24 pre-receive.sample
-rwxr-xr-x 1 quentin staff 1492 28 aoû 13:24 prepare-commit-msg.sample
-rwxr-xr-x 1 quentin staff 2783 28 aoû 13:24 push-to-checkout.sample
-rwxr-xr-x 1 quentin staff 3650 28 aoû 13:24 update.sample

Git hooks



Git hooks

```
# it wants to stop the commit.
#
# To enable this hook, rename this file to "pre-commit".

if git rev-parse --verify HEAD >/dev/null 2>&1
then
        against=HEAD
else
        # Initial commit: diff against an empty tree object
        against=$(git hash-object -t tree /dev/null)
fi

# If you want to allow non-ASCII filenames set this variable to true.
allownonascii=$(git config --type=bool hooks.allownonascii)

# Redirect output to stderr.
exec 1>&2

# Cross platform projects tend to avoid non-ASCII filenames; prevent
```

```
#!/bin/sh
echo "hook pre-commit"
~
```

```
~/Documents/esgi/cours/solution de versionning  master  chmod 744 .git/hooks/pre-commit
```

```
~/Documents/esgi/cours/solution de versionning  master +  git commit -m "test hook pre-commit"
hook pre-commit
[master 4e6179d] test hook pre-commit
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Git reflog

Toute action entreprise sur le dépôt local est enregistré dans un historique consultable avec la commande: git reflog

Les actions ont un sha1 de référence ainsi qu'un numéro selon leur ordre d'exécution

Il est possible de revenir à l'état du dépôt à l'exécution d'une action avec:

git reset --hard Id_du_commit

git reset --hard HEAD@{numéro}

Possibilité d'annuler une suppression de branche, un merge, un rebase etc en local

Revenir sur une action est aussi une action en soit => possibilité de l'annuler aussi

Git stash

Lorsqu'on travaille et que l'on doit synchroniser sa branche ou effectuer une autre opération, une commande existe pour sauvegarder ses modifications sans devoir faire un commit:

git stash

git stash save zone_de_remise

Une fois la tâche terminée, pour réappliquer ces changements:

git stash pop

git stash pop zone_de_remise

On peut aussi lister les modifications remisées avec: git stash list

Git stash

Un code “stashé” est encodé comme des commits, avec l’ensemble des ajouts et suppressions des caractères dans les fichiers.

git/refs/stash fait référence au stash le plus récent.

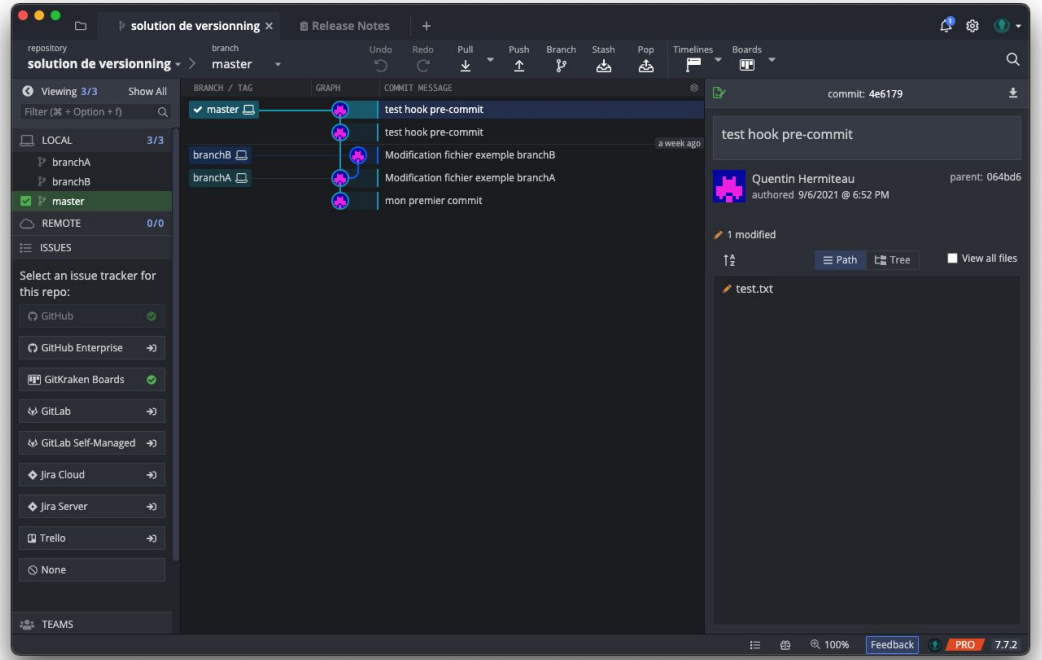
Un code “stashé” peut être réparti sur plusieurs commit, on peut consulter ceux du dernier stash avec: git log stash ou git log stash@{0}

Git ignore

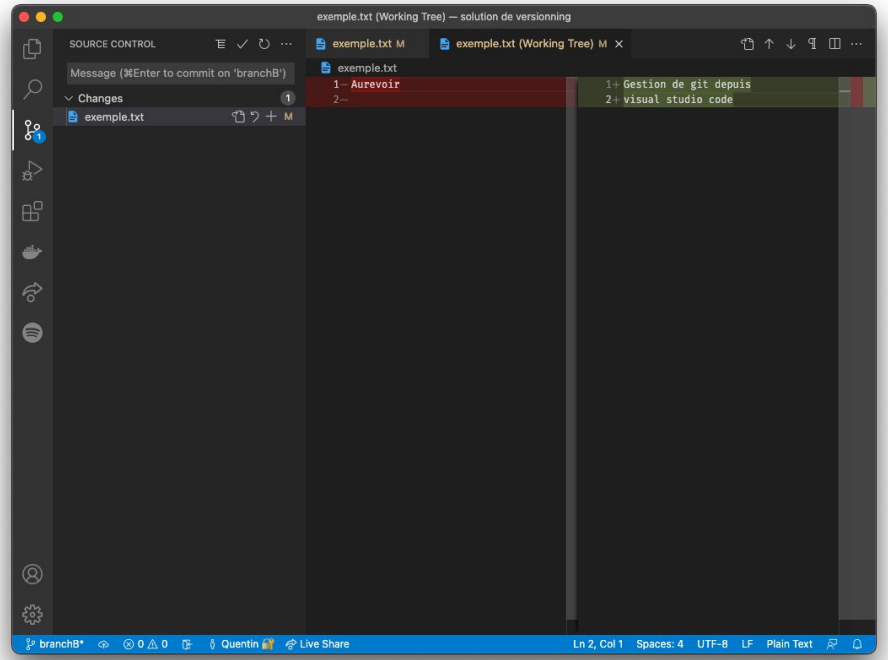
https://github.com/internetwache/GitTools

```
# See https://help.github.com/ignore-files/ for more about ignoring files.

# config
config/base/setupProxy.js

# dependencies
/node_modules

# testing
/coverage
/tests/e2e/reports/*
/tests/e2e/screenshots/*

# production
/build

# misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

Makefile
npm-debug.log*
yarn-debug.log*
yarn-error.log*
.idea
debug.log

# cypress
/cypress/videos
/cypress/screenshots
```

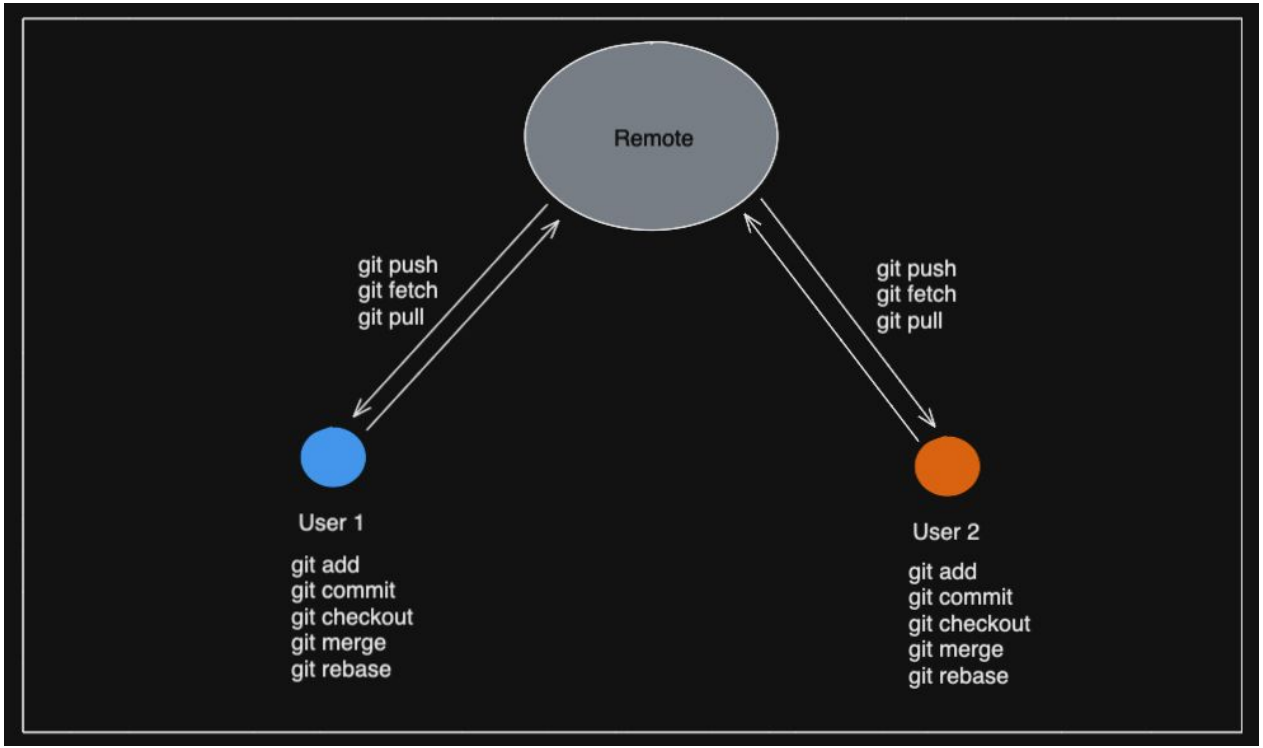
Git kraken


solution de versionning
Release Notes
repository
solution de versionning
branch
master
Undo
Redo
Pull
Push
Branch
Stash
Pop
Timelines
Boards
Viewing 3/3
Show All
Filter (⌘ + Option + f)
LOCAL 3/3
branchA
branchB
master
REMOTE 0/0
ISSUES
Select an issue tracker for this repo:
GitHub
GitHub Enterprise
GitKraken Boards
GitLab
GitLab Self-Managed
Jira Cloud
Jira Server
Trello
None
TEAMS
BRANCH / TAG
GRAPH
COMMIT MESSAGE
master
test hook pre-commit
test hook pre-commit
a week ago
branchB
Modification fichier exemple branchB
branchA
Modification fichier exemple branchA
mon premier commit
commit: 4e6179
test hook pre-commit
Quentin Hermiteau
authored 9/6/2021 @ 6:52 PM
parent: 064bd6
1 modified
Path
Tree
View all files
test.txt
100%
Feedback
PRO
7.7.2

Git dans Visual Studio Code



Git remote

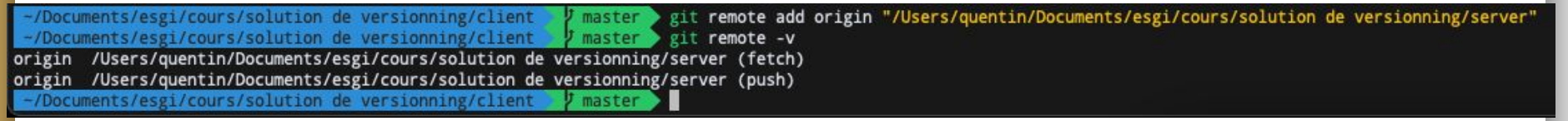

Remote
git push
git fetch
git pull
git push
git fetch
git pull
User 1
git add
git commit
git checkout
git merge
git rebase
User 2
git add
git commit
git checkout
git merge
git rebase

Mise en place d'un remote

```
mkdir server
cd server
git init --bare #Initialise un repository git sans répertoire de
travail
```

Mise en place d'un remote

```
git remote add [name] [path]
```



Git push

```
git push #pousse sur la branche remote
git push --set-upstream origin master #créé la branche en remote et pousse
dessus
```

GitHub

- GitHub students
- Gratuit pour les projets open source et les repo publiques
- Version self-hosted payantes
- Ambiance plus communautaire, découvertes de projets, suivi de projets
- Kanban intégré (GitHub projects)
- CI/CD avec GitHub actions depuis fin 2019
- Protection des branches, authentification, gestion des droits des contributeurs

GitLab

- Gratuit pour les repo publiques et privés
- Version self-hosted gratuite et payante
- Plus axé entreprise
- Kanban intégré (Board)
- CI/CD avec GitLab CI
- Protection des branches, authentification, gestion des droits des contributeurs

Générer une paire de clés SSH

SSH fonctionne avec un échange de clés (https://www.youtube.com/watch?v=y2SWzw9D4RA).

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

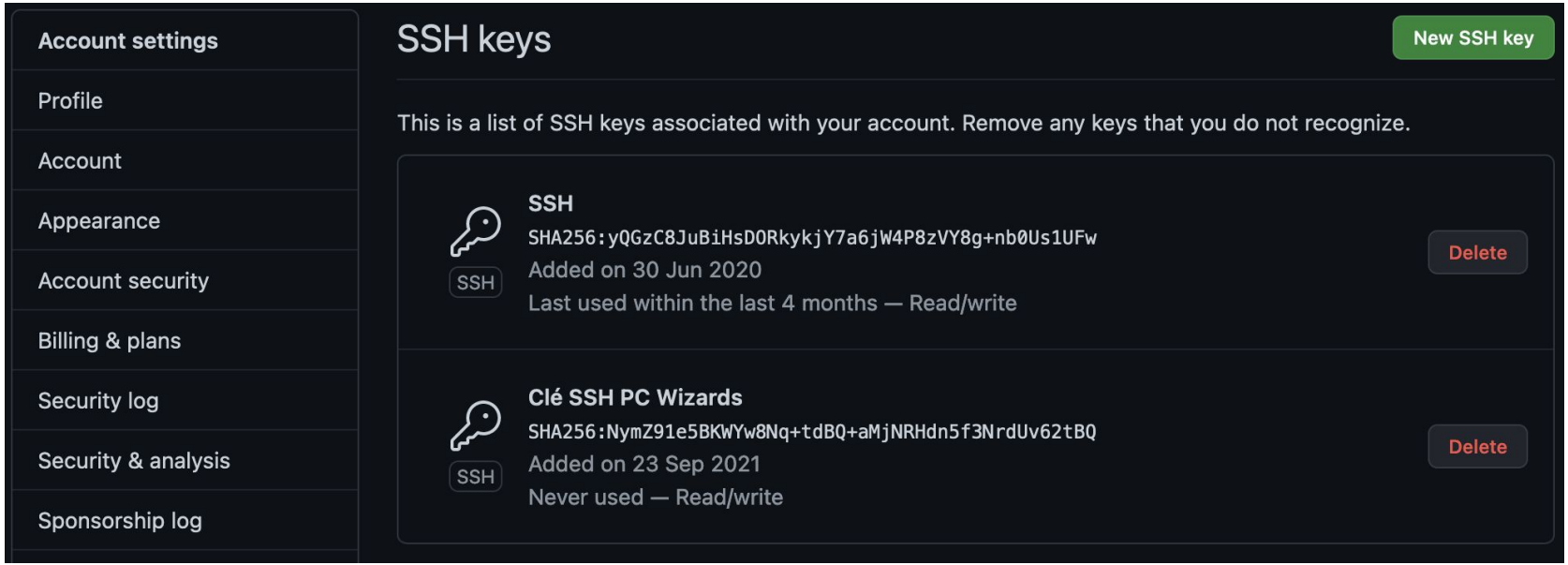
`-t` : algorithme de chiffrement à utiliser
`-C` : commentaire, souvent l'adresse email ou le `utilisateur@machine` , mais peut être n'importe quelle chaîne de caractères
`-f` : fichiers de sortie, ici `/root/.ssh/id_ed25519` et `/root/.ssh/id_ed25519.pub`
`-N` : passphrase

Les permissions sont importantes

```
utilisateur@machine:~/$ ls -la ~/
total 12
drwxr-xr-x    3 root     root          4096 Sep 26 07:54 .
drwxr-xr-x    1 root     root          4096 Sep 26 08:01 ..
drwx------    2 root     root          4096 Sep 26 07:54 .ssh
utilisateur@machine:~/$ ls -la ~/.ssh
total 16
drwx------    2 root     root          4096 Sep 26 07:54 .
drwxr-xr-x    3 root     root          4096 Sep 26 07:54 ..
-rw-------    1 root     root          3434 Sep 26 07:54 id_ed25519
-rw-r--r--    1 root     root           745 Sep 26 07:54 id_ed25519.pub
```

- `.ssh` : chmod 700 : seul le propriétaire peut utiliser ce dossier
- `.ssh/id_ed25519` : chmod 600 : clé privée, seul le propriétaire peut écrire/lire
- `.ssh/id_ed25519.pub` : chmod 644 : clé publique, tout le monde peut lire, seul le propriétaire peut modifier

Clé SSH


Account settings
Profile
Account
Appearance
Account security
Billing & plans
Security log
Security & analysis
Sponsorship log
SSH keys
New SSH key
This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.
SSH
SHA256:yQGzC8JuBiHsDORkykjY7a6jW4P8zVY8g+nb0Us1UFw
Added on 30 Jun 2020
Last used within the last 4 months — Read/write
SSH
Delete
Clé SSH PC Wizards
SHA256:NymZ91e5BKWYw8Nq+tdBQ+aMjNRHdn5f3NrdUv62tBQ
Added on 23 Sep 2021
Never used — Read/write
SSH
Delete

Clé GPG

https://www.gnupg.org/download/

Générer une clé PGP

```
utilisateur@machine:~/$ gpg --full-gen-key
gpg (GnuPG) 2.2.23; Copyright (C) 2020 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/home/utilisateur/.gnupg' created
gpg: keybox '/home/utilisateur/.gnupg/pubring.kbx' created
Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
  (14) Existing key from card
Your selection? 1
```

Générer une clé PGP

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (3072) 4096
Requested keysize is 4096 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 0
Key does not expire at all
Is this correct? (y/N) y

GnuPG needs to construct a user ID to identify your key.

Real name: Utilisateur De La Machine
Email address: utilisateur@machine.fr
Comment: démo
```

Générer une clé PGP

```
You selected this USER-ID:
    "Utilisateur De La Machine (démo) <utilisateur@machine.fr>"

Change (N)ame, (C)omment, (E)maila or (O)kay/(Q)uit? O
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.
gpg: /home/utilisateur/.gnupg/trustdb.gpg: trustdb created
gpg: key 6AA8CF9F1DFEB1DD marked as ultimately trusted
gpg: directory '/home/utilisateur/.gnupg/openpgp-revocs.d' created
gpg: revocation certificate stored as
'/home/utilisateur/.gnupg/openpgp-revocs.d/1472CA3B3A97E01D42372838A2A1B12FB3EBFA87.rev'
public and secret key created and signed.
```

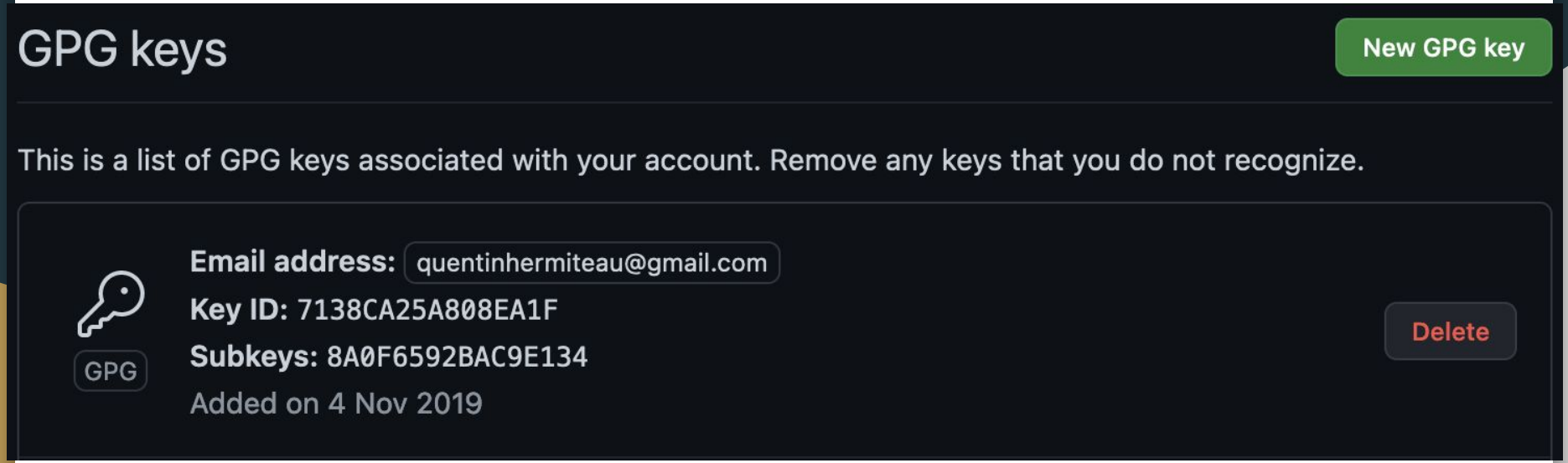
Générer une clé PGP

```
pub   rsa4096 2020-09-27 [SC]
      1472CA3B3A97E01D42372838A2A1B12FB3EBFA87
uid                      Utilisateur De La Machine (démo)
<utilisateur@machine.fr>
sub   rsa4096 2020-09-27 [E]
```

Clé GPG

```
~/Documents/cours/git/4IW4/test-github git:(main) (0.074s)
gpg --list-secret-keys --keyid-format=long
/Users/quentin/.gnupg/pubring.kbx
---------------------------------
sec   rsa4096/86FBBB824B32C79C 2021-12-21 [SC]
      C4DC8EA5225AB32C368DFFAA86FBBB824B32C79C
uid                 [ultimate] Quentin Hermiteau (Clé GPG PC Wizards) <quentinhermiteau@gmail.com>
ssb   rsa4096/920240160F0E4216 2021-12-21 [E]

sec   rsa4096/5139CEC2F80FFF43 2022-05-23 [SC]
      E0E66E55AE3CD420618B5D6C5139CEC2F80FFF43
uid                 [ultimate] Quentin Hermiteau (Clé GPG Total) <qhermiteau@wedigital.garden>
ssb   rsa4096/46147312FF577654 2022-05-23 [E]


~/Documents/cours/git/4IW4/test-github git:(main)
gpg --armor --export 5139CEC2F80FFF43
```

Clé GPG


GPG keys
New GPG key
This is a list of GPG keys associated with your account. Remove any keys that you do not recognize.
Email address: quentinhermiteau@gmail.com
Key ID: 7138CA25A808EA1F
Subkeys: 8A0F6592BAC9E134
Added on 4 Nov 2019
GPG
Delete

Signer un commit

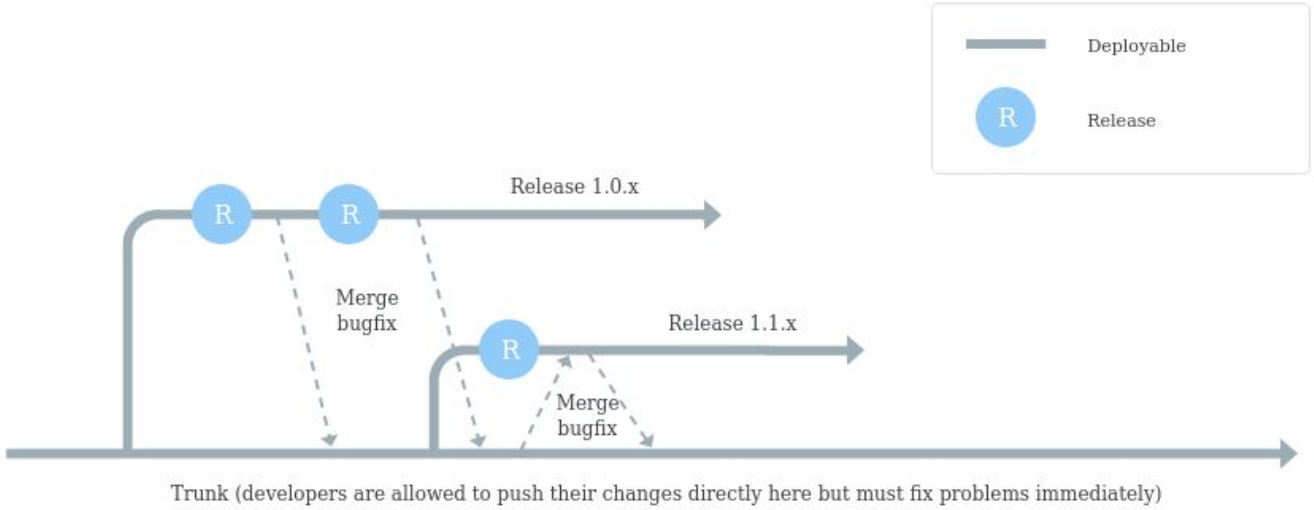
```
utilisateur@machine:~/$ git config --global user.signingkey A2A1B12FB3EBFA87

utilisateur@machine:~/$ # git config --global gpg.program gpg2 # facultatif

utilisateur@machine:~/$ # export GPG_TTY=$(tty) # facultatif

utilisateur@machine:~/$ git commit -S -m "Signed commit"

utilisateur@machine:~/$ git commit -m "Signed commit"
```

Troubleshooting

```
utilisateur@machine:~/$ echo "test" | gpg --clearsign
utilisateur@machine:~/$ echo "test" | gpg2 --clearsign
```

https://www.gnupg.org/(it)/documentation/manuals/gnupg/Common-Problems.html

Workflow : trunk based development



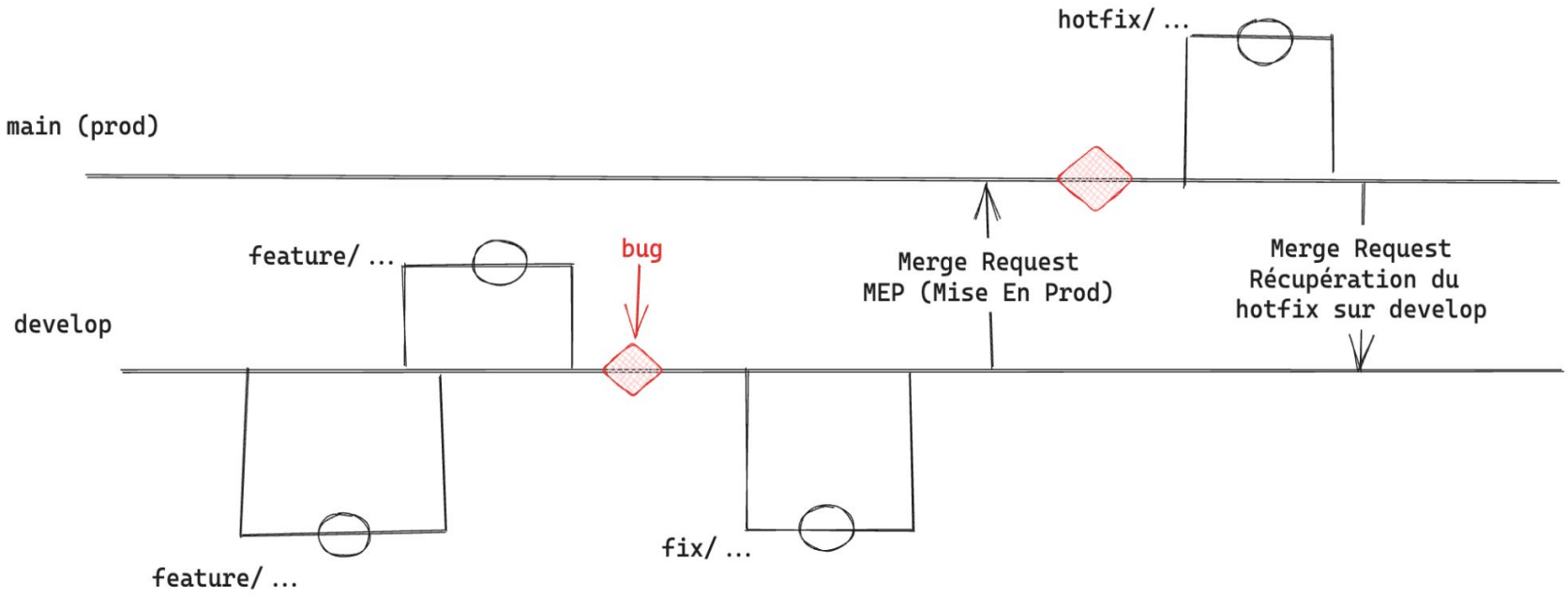
Workflow : Github flow

- Entre gitflow et le trunk based development
- Une branche principale (main) permanente
- Une branche par feature, fix etc.
- Ouverture d'une Pull Request (ou Merge Request sur gitlab) pour ouvrir une discussion sur la branche (le code peut encore évoluer)
- Si la PR est validée, une pipeline est éventuellement lancée
- Une fois la pipeline terminée sans erreur, la branche est alors fusionnée sur main et la PR close
- https://guides.github.com/introduction/flow/

Workflow : Git flow

Branche	Nombre	Branche d'origine	Durée de vie	Fonction
master	Unique		Permanente	Code stable, testé et validé potentiellement éligible pour une MEP (Mise En Production)
feature	Plusieurs	develop	Développement d'une feature	Code en cours de développement qui implémente une fonctionnalité à embarquer dans la prochaine version de l'application
develop	Unique	master	Permanente	Code de la prochaine version de l'application. Une fois que le développement d'une fonctionnalité (*feature*)est fini, le code est fusionné sur cette branche
release	Unique	develop	Recette	Branche sur laquelle on corrigera les bugs détectés pendant la phase de recette
hotfix	Aucune/Plusieurs	master	Correction d'un bug	Branche où on fait les corrections des bugs sur le code présent sur la branche `master` (production)

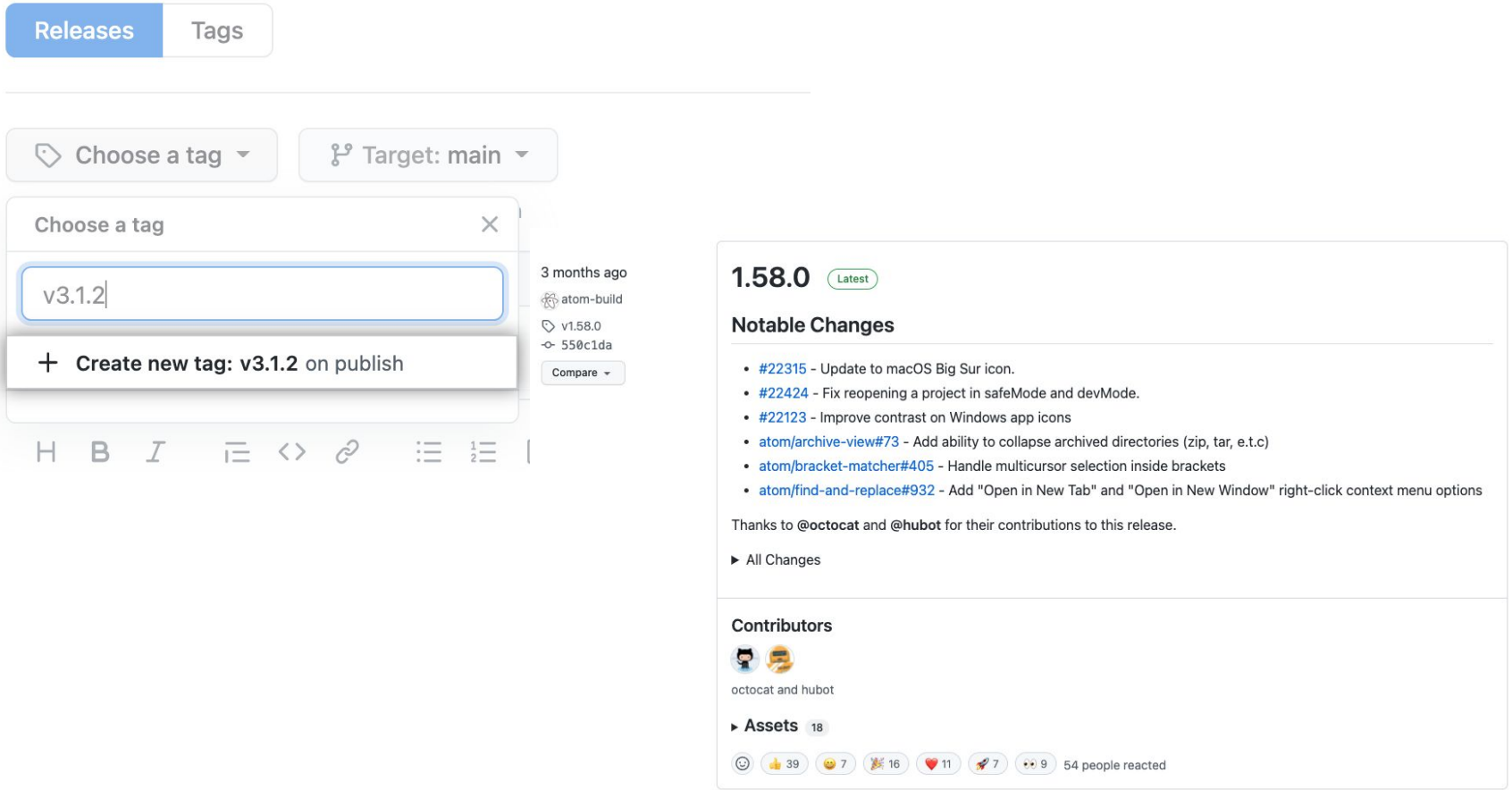
Workflow : Git flow



Grand tour de GitHub

- Création d'un repository
- Création d'une issue
- Création du projet
- Création de templates
- Pull requests
- Les fichiers Markdown

SemVer
Semantic Versioning


Releases
Tags
Choose a tag
Target: main
Choose a tag
v3.1.2
Create new tag: v3.1.2 on publish
3 months ago
atom-build
v1.58.0
550c1da
Compare
1.58.0
Latest
Notable Changes
#22315 - Update to macOS Big Sur icon.
#22424 - Fix reopening a project in safeMode and devMode.
#22123 - Improve contrast on Windows app icons
atom/archive-view#73 - Add ability to collapse archived directories (zip, tar, e.t.c)
atom/bracket-matcher#405 - Handle multicursor selection inside brackets
atom/find-and-replace#932 - Add "Open in New Tab" and "Open in New Window" right-click context menu options
Thanks to @octocat and @hubot for their contributions to this release.
All Changes
Contributors
octocat and hubot
Assets 18
39
7
16
11
7
9
54 people reacted

Licences

Les licences open source

Les licences libres ont été créées dans le but de proposer une alternative aux licences propriétaires existantes, représentées par le Copyright© ou le TradeMark ™.

Les droits d'auteur sont détenus par le développeur qui a écrit le programme ou par l'entreprise qui l'emploie.

Le détenteur de la licence définit l'utilisation qu'il souhaite faire de son programme :

- Le garder pour lui
- Le vendre à un tiers
- L'utilisation de son droit d'auteur pour conditionner l'utilisation du programme.

Ces conditions seront définies dans les termes de la licence.

Source : https://www.diatem.net/les-licences-open-source/

Principes fondamentaux d’un logiciel open source

L’auteur des droits est libre de changer les conditions de la licence ou d’y apporter des aménagements.

Le logiciel libre se définit par le respect de libertés fondamentales :la liberté d’utiliser le logiciel à n’importe quelle fin,
la liberté de modifier le programme pour répondre à ses besoins,
la liberté de redistribuer des copies,
la liberté de partager avec d’autres les modifications apportées.

Quand une licence offre à ses utilisateurs toutes ces libertés, il peut être qualifié de logiciel libre (Open source).

Source : https://www.diatem.net/les-licences-open-source/

Comment choisir sa licence ?

https://choosealicense.com/

CODE_OF_CONDUCT.md

- Un code de conduite définit des normes sur la manière de s'engager dans une communauté.
- Il signale un environnement inclusif qui respecte toutes les contributions.
- Il décrit également les procédures à suivre pour résoudre les problèmes entre les membres de la communauté de votre projet.

Template :

- Contributor-covenant.org
- citizencodeofconduct.org

CONTRIBUTING.md

Des guidelines pour indiquer comment contribuer à votre projet.

Pour aider les contributeurs à faire du bon travail.

Exemples:

- https://github.com/atom/atom/blob/master/CONTRIBUTING.md
- https://github.com/github/opensource.guide/blob/master/CONTRIBUTING.md
- https://github.com/rails/rails/blob/master/CONTRIBUTING.md