9 octobre 2018
Formation Picasoft : La gestion de version avec Git (niveau 1)

Thibaud Duhautbout
Rémy Huet

Association Picasoft

Mardi 9 octobre 2018

Question avant de commencer

« Est-ce qu'on peut s'en servir pour donner de l'élan à un pigeon ? »
Yvain, chevalier au lion
Kaamelott Livre III ep. 95 L'Étudiant

Pourquoi la gestion de version ?

- sauvegarde incrémentale du travail ;
- suivi des modifications et retour en arrière ;
- partage des modifications avec d'autres personnes ;
- centralisation des sources ;
- collaboration simplifiée ;
- possibilité de maintenir plusieurs versions.

Les différents logiciels de version

git

Et plein d'autres !

37 systèmes recensés sur Wikipedia

(`https://en.wikipedia.org/wiki/Comparison_of_version_control_software`)

Petite histoire de git...

- Créé en 2005 par les développeurs du noyau Linux ;
- Système de gestion de version distribué ;
- Rapide ;
- Possibilité de développements non-linéaires (branches) ;
- Outil très populaire chez les développeurs (GitHub, GitLab, ...)

On passe à la partie pratique !

Dans la suite, on considère que git est installé pour toutes les machines !

En cas de problème, n'hésitez pas à demander de l'aide aux gentils animateurs munis d'une pancarte « HELP » :)

On ouvre un terminal ou Git Bash !

Configuration de git et du dépôt

Création d'un nouveau répertoire de travail

```
$ mkdir formation_git
$ cd formation_git
```

Initialisation du dépôt git :

```
$ git init
Dépôt Git vide initialisé dans /home/user/formation_git/.git
```

Configuration de l'identité de l'utilisateur :

- Permet d'identifier l'auteur des mises à jour ;
- `global` : configuration au niveau du système ;
- `local` : configuration au niveau du répertoire courant.

```
$ git config --global/local user.name "<prénom nom>"
$ git config --global/local user.email "<adresse email>"
```

Voir l'état du répertoire local

La commande

```
$ git status
```

Utilité

Permet de connaître à tout moment l'état d'un répertoire git :

- La branche sur laquelle on se situe ;
- Les fichiers suivis ou non suivis ;
- Les fichiers modifiés depuis la dernière validation ;
- Les fichiers qui seront validés et ceux qui ne le seront pas.

En résumé

Le Saint Graal des commandes git !

git status à la loupe

Pour le moment...

```
$ git status
Sur la branche master

Aucun commit

rien à valider (créez/copiez des fichiers et utilisez "git add" pour les suivre)
```

On va créer un fichier !

```
$ echo "J'apprends à utiliser git" > formation.txt

$ git status
Sur la branche master

Aucun commit

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

        formation.txt

aucune modification ajoutée à la validation mais des fichiers non suivis sont présents
(utilisez "git add" pour les suivre)
```

Un commit, c'est quoi ?

Pour faire court

Un commit est un **point de sauvegarde**

À quoi ça sert ?

- Les commits se suivent ;
- Sauvegarde incrémentale ;
- Possibilité de revenir à une version donnée.

Working Directory vs. Staging Area vs. Repository

Repository [dépôt]

Le Repository (ou dépôt) correspond aux fichiers dans l'état de la dernière validation connue par git.

Working Directory [répertoire de travail]

Le Working Directory correspond à l'état actuel du répertoire git :

- nouveaux fichiers pas encore ajoutés au Repository (fichiers non suivis) ;
- fichiers modifiés depuis la dernière version.

Staging Area [zone de préparation / zone d'index]

Zone intermédiaire entre le Working Directory et le Repository. Elle contient les modifications apportées dans le Working Directory que git va ajouter au Repository.

Ajouter des modifications pour validation

Ajouter les modifications d'un fichier pour validation :

```
$ git add <fichier(s)>
```

Ajouter toutes les modifications pour validation (tous les fichiers) :

```
$ git add -A
```

Enlever les modifications d'un fichier de la validation :

```
$ git reset <fichier>
```

(ne change pas le contenu du fichier mais indique juste à git d'ignorer ses modifications pour la validation)

```
$ git status
Sur la branche master
Aucun commit
Fichiers non suivis:

        formation.txt

$ git add formation.txt

$ git status
Sur la branche master

Aucun commit

Modifications qui seront validées :
  (utilisez "git rm --cached <fichier>..." pour désindexer)

        nouveau fichier : formation.txt

$ git reset formation.txt

$ git status
Sur la branche master
Aucun commit
Fichiers non suivis:

        formation.txt

$ git add formation.txt
```

Valider les modifications

Valider les changements qui ont été ajoutés au staging area :

```
$ git commit
```

Le commit doit contenir un message qui résume le contenu des modifications. Pour l'entrer directement :

```
$ git commit -m "<message>"
```

```
$ git commit -m "Ajout du premier fichier"
[master (commit racine) 6b6799b] Ajout du premier fichier
 1 file changed, 1 insertion(+)
 create mode 100644 formation.txt
```

Dissection d'un commit

À propos du commit

- Chaque commit possède un identifiant unique ;
- Un commit est asocié à une unique personne ;
- L'historique des commits est incrémental. Tout commit (excepté le premier) a un commit « père » ;
- Un commit correspond à une version figée du projet ;
- On peut naviguer dans les commits.

git log

Afficher l'historique des commits

```
$ git log
commit 2487fdd243542146f15a8e6bb00a94a39117ea1b ( HEAD -> master )
Author: huetremy <remy.huet@etu.utc.fr>
Date:   Mon Sep 24 09:54:14 2018 +0200

            Ajout du premier fichier
```

On peut voir ici :

- L'identifiant unique du commit ;
- L'auteur (et son mail) ;
- La date du commit ;
- Le message qui a été mis lors du commit.

git diff

Permet de voir les modifications apportées au dépôt :

- Depuis l'état du staging area : `$ git diff` ;
- Depuis le dernier commit : `$ git diff HEAD` ;
- Depuis un commit quelconque : `$ git diff <id_commit>` ;
- Entre deux commits quelconques :
  `$ git diff <id_commit_départ> <id_commit_arrivée>`.

```
$ echo "J'ajoute une ligne à mon fichier" >> formation.txt

$  git diff

 diff --git a/formation.txt b/formation.txtindex 951923e..bbbb145 100644--- a/formation.txt+++ b/formation.txt
@@ -1 +1,2 @@
 J'apprends à utiliser git
+J'ajoute une ligne à mon fichier

$  git add formation.txt

$  git diff

$  git diff HEAD

 diff --git a/formation.txt b/formation.txtindex 951923e..bbbb145 100644--- a/formation.txt+++ b/formation.txt
@@ -1 +1,2 @@
 J'apprends à utiliser git
+J'ajoute une ligne à mon fichier

$  git commit -m "Second commit"

[master e788cc0] Second commit
 1 file changed, 1 insertion(+)
```

```
$ git log

commit e788cc09da2de56b9dd530a78c2f610b94bea356 (HEAD -> master)
Author: huetremy <remy.huet@etu.utc.fr>
Date:   Mon Sep 24 11:39:21 2018 +0200

    Second commit

commit 2487fdd243542146f15a8e6bb00a94a39117ea1b
Author: huetremy <remy.huet@etu.utc.fr>
Date:   Mon Sep 24 09:54:14 2018 +0200

    Ajout du premier fichier

$  git diff 2487d e788c

 diff --git a/formation.txt b/formation.txtindex 951923e..bbbb145 100644--- a/formation.txt+++ b/formation.txt
@@ -1 +1,2 @@
 J'apprends à utiliser git
+J'ajoute une ligne à mon fichier
```

Enregistrer les modifications locales

```
$ echo "travail en cours..." >> formation.txt
```

– « Tiens, tu pourrais m'envoyer le rapport ? »
– « Euh, en fait je travaille dessus et j'ai changé des choses... »
– « Bah fais un git stash ! »

`$ git stash` : enregistre les modifications locales et restaure le working directory à l'état du dernier commit.

```
$ git status
Sur la branche master
Modifications qui ne seront pas validées :

        modifié :         formation.tex

$  git stash
Copie de travail et état de l'index sauvegardés dans WIP on master: 9a7302c Second commit

$  git status
Sur la branche master
rien à valider, la copie de travail est propre
```

Restaurer les modifications enregistrées

– « Eeeeeh mais je fais comment maintenant ? »
– « Tranquille, git stash pop ! »

`$ git stash pop` : applique les modifications enregistrées par le **dernier** `stash` sur le working directory (attention aux conflits en cas de modifications qui se recoupent !)

```
$ git stash pop
Sur la branche master
Modifications qui ne seront pas validées :

        modifié :         formation.txt

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
refs/stash@{0} supprimé (849fdcc28bb60b9196099a567958cd8408b599dc)
```

Et pour faire bonne mesure...

```
$ git add formation.txt

$ git commit -m "travail en cours"
```

Changer de version

« Dis, t'aurais encore la version du projet qu'on a envoyé au prof la semaine dernière ? »

```
$ git log
commit 2624c90dbc8f28be29f7cbd8ea497eaef8832f44 (HEAD -> master)
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sun Sep 30 20:38:59 2018 +0200

    travail en cours

commit 9a7302c06628ef69a5e1c9cebc2a1c2904e7d41f
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sun Sep 30 20:34:08 2018 +0200

    Second commit <--- C'est cette version-là qu'il veut !

commit 6b6799b3209de6cb00c69b2afb490abb0f5481e9
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sat Sep 22 22:18:26 2018 +0200

    Ajout du premier fichier
```

`HEAD` = position actuelle du Working Directory dans le Repository

Changer de version (suite)

`$ git checkout <sha commit>` : rétablit le HEAD au commit indiqué

Attention, l'identifiant du commit ne sera peut-être pas le même chez vous, pensez à le remplacer par celui qui est affiché dans votre log !

```
$ cat formation.txt
J'apprends à utiliser git
J'ajoute une ligne à mon fichier
travail en cours...

$  git checkout 9a7302c06628ef69a5e1c9cebc2a1c2904e7d41f
Note : extraction de '9a7302c06628ef69a5e1c9cebc2a1c2904e7d41f'.

Vous êtes dans l'état « HEAD détachée ». Vous pouvez visiter, faire des modifications
expérimentales et les valider. Il vous suffit de faire une autre extraction pour
abandonner les commits que vous faites dans cet état sans impacter les autres branches

[...]

HEAD est maintenant sur 9a7302c Second commit
```

Changer de version (suite)

```
$ git status
HEAD détachée sur 9a7302c
rien à valider, la copie de travail est propre

$  cat formation.txt
J'apprends à utiliser git
J'ajoute une ligne à mon fichier

$  git log --all
commit 2624c90dbc8f28be29f7cbd8ea497eaef8832f44 (master)
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sun Sep 30 20:38:59 2018 +0200

    travail en cours

commit 9a7302c06628ef69a5e1c9cebc2a1c2904e7d41f (HEAD)
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sun Sep 30 20:34:08 2018 +0200

    Second commit

commit 6b6799b3209de6cb00c69b2afb490abb0f5481e9
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sat Sep 22 22:18:26 2018 +0200

    Ajout du premier fichier
```

Changer de version (suite)

« Et comment je reviens où j'étais avant ça ? »

```
$ git checkout master
La position précédente de HEAD était sur 9a7302c Second commit
Basculement sur la branche 'master'

$ git log
commit 2624c90dbc8f28be29f7cbd8ea497eaef8832f44 (HEAD -> master)
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sun Sep 30 20:38:59 2018 +0200
    travail en cours

commit 9a7302c06628ef69a5e1c9cebc2a1c2904e7d41f
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sun Sep 30 20:34:08 2018 +0200
    Second commit

commit 6b6799b3209de6cb00c69b2afb490abb0f5481e9
Author: Thibaud Duhautbout <thibaud@duhautbout.ovh>
Date:   Sat Sep 22 22:18:26 2018 +0200
    Ajout du premier fichier

$ cat formation.txt
J'apprends à utiliser git
J'ajoute une ligne à mon fichier
travail en cours...
```

Annuler les modifications sur un fichier particulier

```
$ echo "ligne qui casse tout..." >> formation.txt
```

« Je sais pas ce que j'ai fait, mais ce fichier il était bien avant que j'y touche... On peut revenir en arrière non ? »

`$ git checkout -- <nom du fichier>` : rétablit le fichier indiqué dans la version du dernier commit (**les modifications locales seront perdues !**)

Annuler les modifications sur un fichier précis

```
$ git status
Sur la branche master
Modifications qui ne seront pas validées :

        modifié :         formation.txt

$ git diff formation.txt
diff --git a/formation.txt b/formation.txt
index b1e9ea1..06fc038 100644
--- a/formation.txt
+++ b/formation.txt
@@ -1,3 +1,4 @@
 J'apprends à utiliser git
 J'ajoute une ligne à mon fichier
 travail en cours...
+ligne qui casse tout

$ git checkout -- formation.txt

$ git status
Sur la branche master
rien à valider, la copie de travail est propre

$ cat formation.txt
J'apprends à utiliser git
J'ajoute une ligne à mon fichier
travail en cours...
```

GitLab

Principe

Un serveur git en ligne pour sauvegarder et partager des dépôts. Les deux plus connus sont GitHub et GitLab.

Application

Création d'un dépôt sur GitLab

git clone

« Dis, il paraît qu'on peut récupérer le dépôt de la présentation, comment faire ? »

```
$ git clone https://gitlab.utc.fr/picasoft/formations/A18/git-v1
Clonage dans 'git-v1'...
warning: redirection vers https://gitlab.utc.fr/picasoft/formations/A18/git-v1.git/
remote: Counting objects: 74, done.
remote: Compressing objects: 100% (55/55), done.
remote: Total 74 (delta 35), reused 40 (delta 16)
Dépaquetage des objets: 100% (74/74), fait.

$ cd git-v1

$ git status

Sur la branche master
Votre branche est à jour avec 'origin/master'.

rien à valider, la copie de travail est propre
```

git pull

« Dis, comment je récupère tes modifications sur mon PC ?
– T'as essayé git pull ?
– Ah, non... »

```
$ git pull
warning: redirection vers https://gitlab.utc.fr/picasoft/formations/a18/git-v1.git/
remote: Counting objects: 3, done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0)
Dépaquetage des objets: 100% (3/3), fait.
Depuis https://gitlab.utc.fr/picasoft/formations/a18/git-v1
   00938d6..3f31ae0  master     -> origin/master
Mise à jour 00938d6..3f31ae0
Fast-forward
 presentation.tex | 29 ++++++++++++++++++++++++++++++-
 1 file changed, 28 insertions(+), 1 deletion(-)
```

git push

« Dis, comment je peux envoyer mes modifications vers la remote ?
– Facile ! Tu utilises git push ! »

```
$ cd
$ cd Documents
$ git clone https://gitlab.utc.fr/mon_login/fomation-git
...
$ cd formation-git
$ echo "Hello World" > text.txt
$ git add -A
$ git commit -m "Ajout fichier text.txt"
...
$ git push
Username for 'https://gitlab.utc.fr':  huetremy
Password for 'https://huetremy@gitlab.utc.fr':
warning: redirection vers https://gitlab.utc.fr/huetremy/formation-git.git/
Énumération des objets: 5, fait.
Décompte des objets: 100% (5/5), fait.
Compression par delta en utilisant jusqu'à 8 fils d'exécution
Compression des objets: 100% (3/3), fait.
Écriture des objets: 100% (3/3), 1.19 KiB | 1.19 MiB/s, fait.
Total 3 (delta 2), réutilisés 0 (delta 0)
To https://gitlab.utc.fr/huetremy/picasoft-git.git/
   d407f20..80900d3  master -> master
```

Conclusion

Maintenant, vous savez :

- utiliser les fonctions basiques de git !
- versionner un projet local ;
- sauvegarder votre projet sur un GitLab distant !

Mais il reste plein de choses à voir !

La prochaine fois on verra :

- les branches ;
- le merge, rebase et la joie des conflits ;
- gestion plus fine des fichiers du dépôt.

Et maintenant, à vos questions !