

Git

au quotidien

THIS IS GIT. IT TRACKS COLLABORATIVE WORK
ON PROJECTS THROUGH A BEAUTIFUL
DISTRIBUTED GRAPH THEORY TREE MODEL.

COOL. HOW DO WE USE IT?

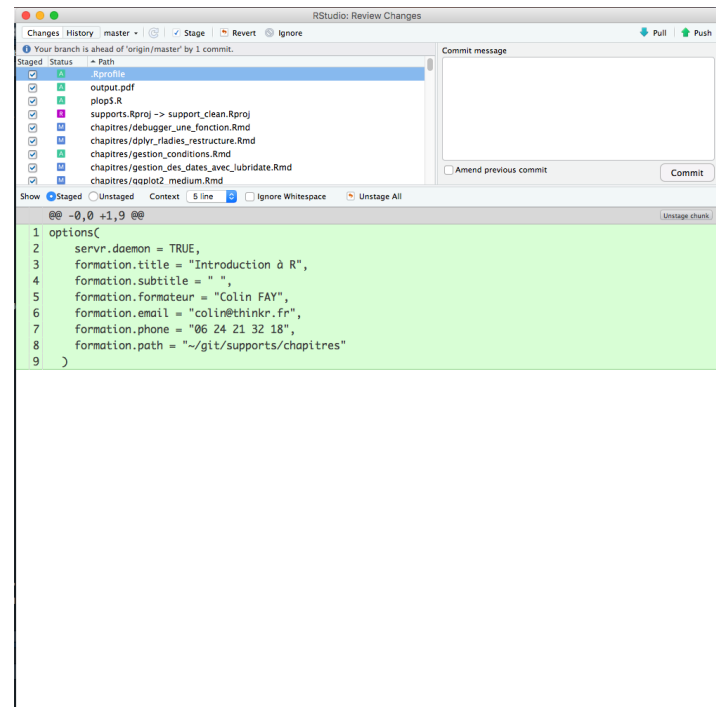
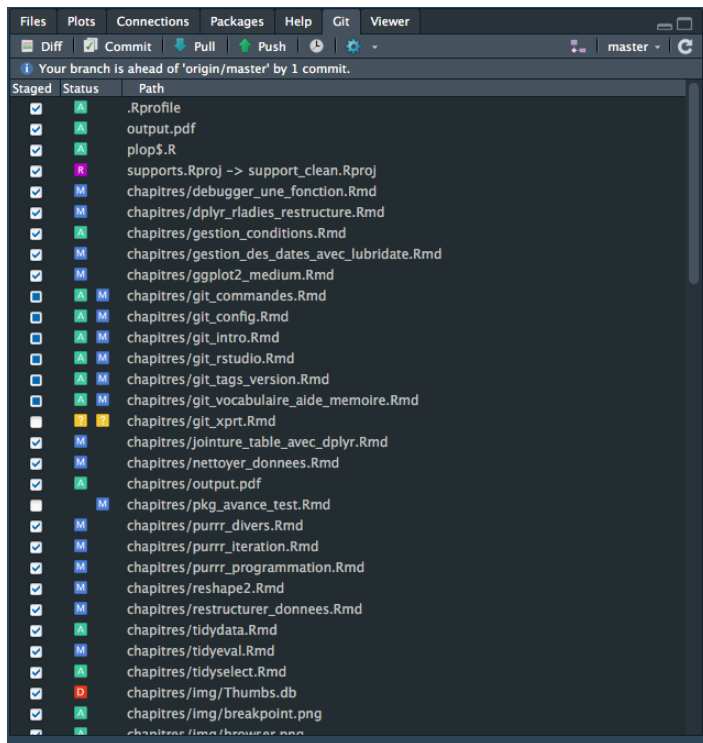
NO IDEA. JUST MEMORIZE THESE SHELL
COMMANDS AND TYPE THEM TO SYNC UP.
IF YOU GET ERRORS, SAVE YOUR WORK
ELSEWHERE, DELETE THE PROJECT,
AND DOWNLOAD A FRESH COPY.



Git et RStudio

Git est nativement intégré dans RStudio.

Si vous êtes allergique à la ligne de commande, la fenêtre "Git" vous permet de réaliser les opérations courantes en "point and clic".



Usage de Git

Au quotidien vous utiliserez ces 3 instructions principales

commit

Une fois des fichiers de votre projet modifiés, utiliser le panneau "Git" et le bouton **commits** pour figer un état. Un commentaire expliquant vos modifications devra être saisi.

Push

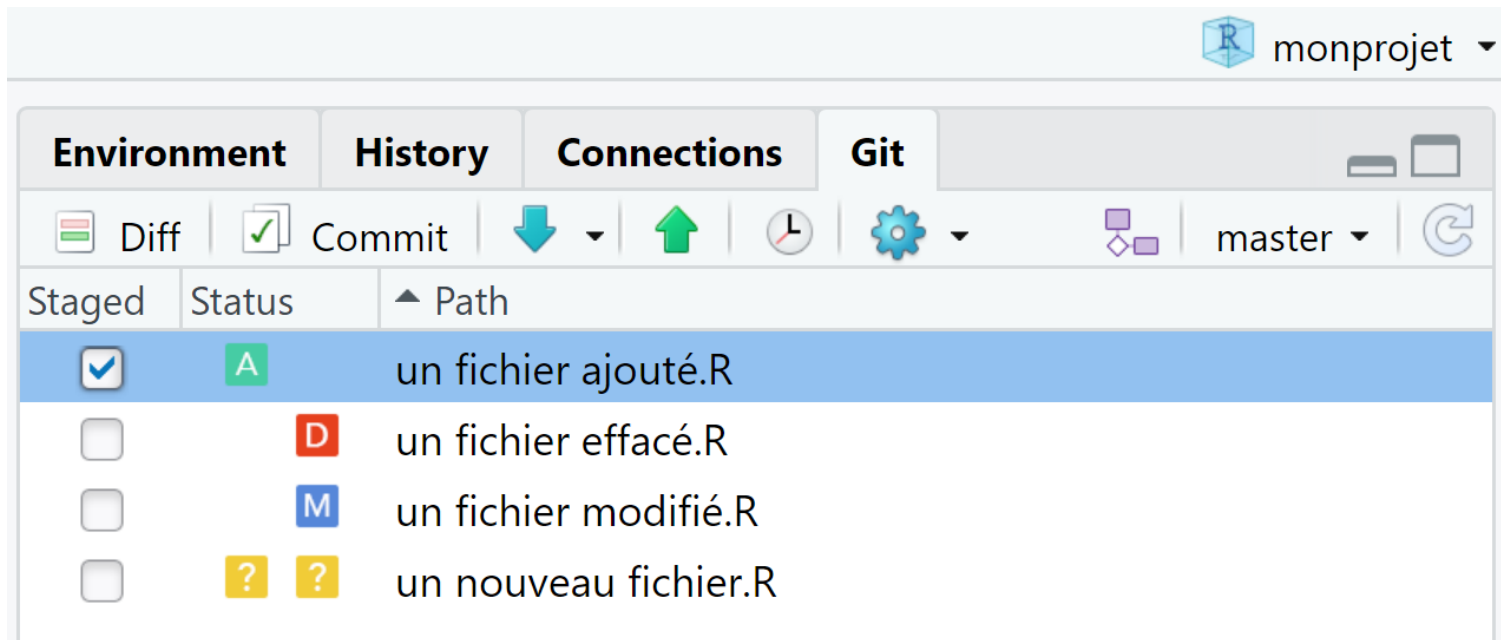
Une fois qu'un certain nombre de **commit** ont été effectués, afin de les envoyer sur le **remote**, il faut effectuer un **push**. Pour cela, utiliser la fenêtre "Git" et le bouton **push** (représenté sous la forme d'une flèche montante).

Pull

Pour rapatrier des modifications depuis le **remote**, il faut faire un **pull** en utilisant la fenêtre "Git" et le bouton **pull** (représenté sous la forme d'une flèche descendante). Votre projet Rstudio est alors modifié.

Usage de Git

Le panneau git de Rstudio va vous indiquer en temps réel l'état de votre projet. Le statut des différents fichiers et dossiers seront affichés.

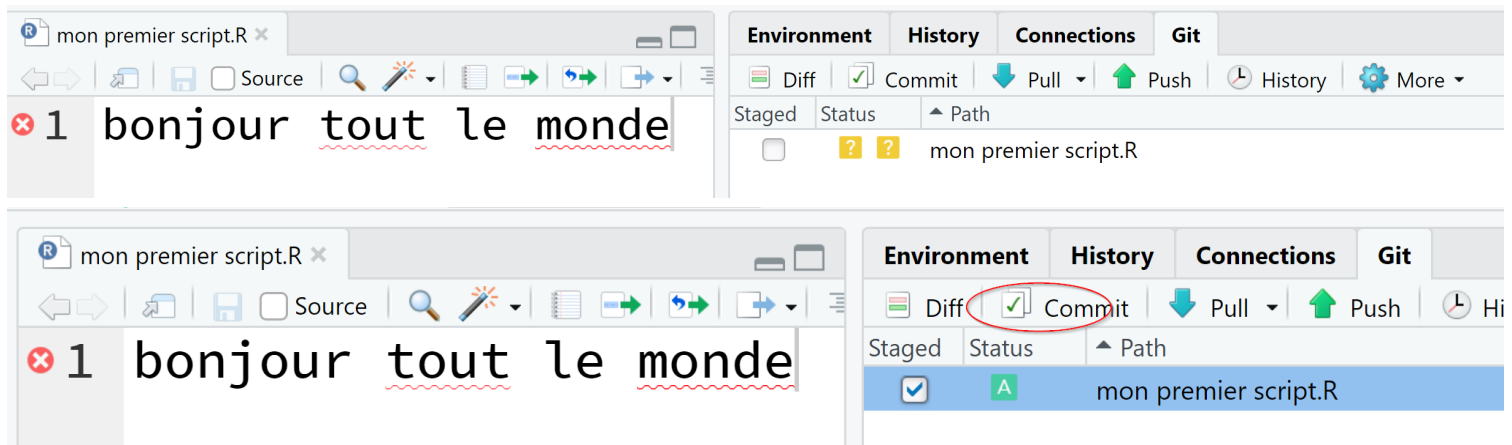


- Un nouveau fichier sera associé a une icone orange contenant un ?
- Ce nouveau fichier sera associé un une icone verte contenant un A une fois que vous l'aurez coché (dans la colonne 'staged').
- Un fichier modifié sera associé a une icone bleue contenant un M
- Un fichier effacé sera associé a une icone rouge contenant un D

Usage de Git

Premier commit

- Avec File > New File > Rscript (ou ctrl + shift + N) ouvrez un nouveau fichier .R .
- Enregistrez le (ctrl + S) dans votre projet. Il va apparaitre en tant que nouveau fichier dans le panneau Git. Choisissez un nom explicite, idéalement sans majuscules ni caracteres spéciaux.
- Avant de faire votre Premier commit, écrivez quelque chose dans ce fichier avant de l'enregistrer à nouveau.
- Cochez le dans la colonne staged (il passe avec une icone verte A)
- Cliquez sur le bouton **commit**



Usage de Git

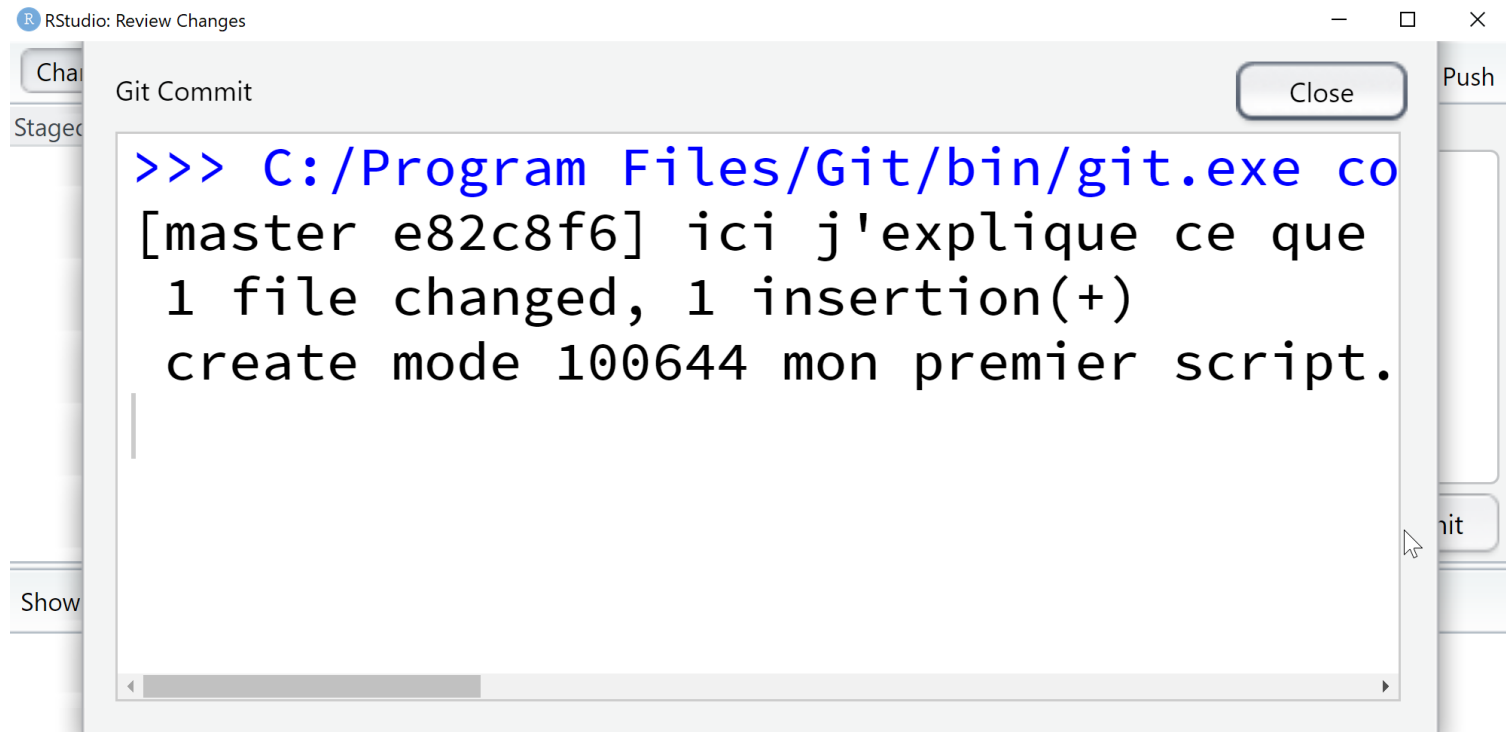
Premier commit

Vous devez écrire un message de `commit`. Ce message est important car il raconte l'histoire de votre projet. Cliquez ensuite sur le bouton `commit` de cette fenêtre. Vous pouvez ensuite cliquer sur close et fermer la fenêtre de `commit`.

Usage de Git

Premier commit

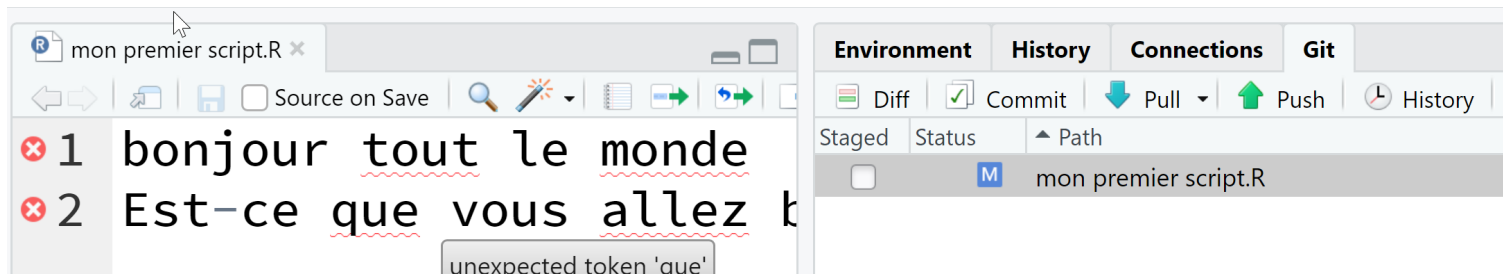
Vous devez écrire un message de **commit**. Ce message est important car il raconte l'histoire de votre projet. Cliquez ensuite sur le bouton **commit** de cette fenêtre.



Usage de Git

Premier commit

Faites à présent d'autres modifications sur ce fichier, enregistrez-les. git detecte la modification et passe votre fichier au statut "modifié" (icone bleue avec un M)



Faites un nouveau **commit** de cette modification :

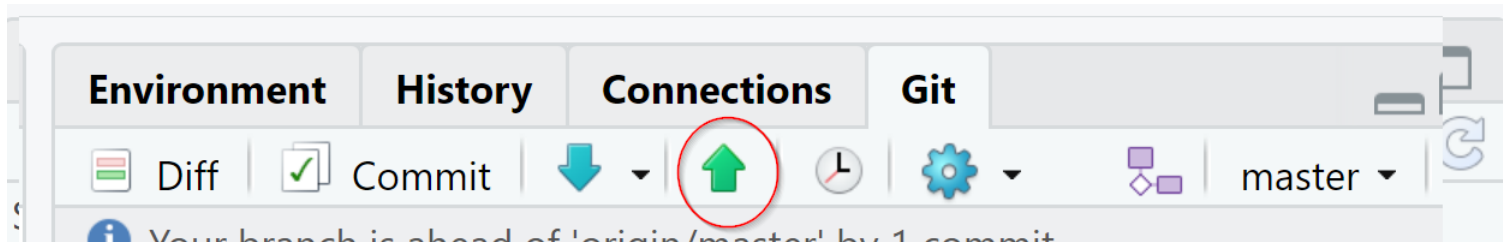
1. on coche le fichier
2. on clique sur **commit**
 - notez la fenêtre qui précise les modifications effectuées
3. on précise le message de **commit**
4. on clique sur **commit**
5. on ferme la fenêtre de **commit**

Usage de Git

Premier push

Pour l'heure les **commit**s que vous avez fait ne sont présents qu'en local sur votre ordinateur, le **remote** n'est pas au courant des modifications que vous avez faites.

Il vous faut envoyer ces **commit**s sur le serveur distant, vous allez faire un **push**. Pour cela il faut cliquer sur la flèche verte montante du panneau git.



Usage de Git

Premier push

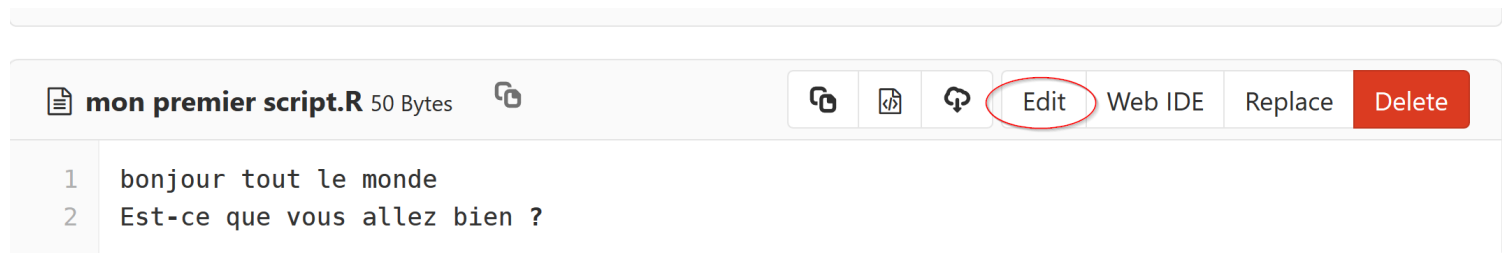
Rendez-vous à nouveau sur gitlab à l'adresse de la forme https://git.thinkr.fr/votre_nom/monprojet , actualisez la page. Vous verrez que votre projet est accessible en ligne.

Usage de Git

Premier pull

Pour le moment vous travaillez seul sur votre projet, donc le **pull** est sans effet : vous êtes "already up to date". Nous allons simuler une modification d'un fichier en intervenant directement dans l'interface web de gitlab.

cliquer sur le fichier `.R` que vous avez versionné, puis sur 'edit'



Usage de Git

Premier pull

Utiliser l'interface web pour modifier le fichier et cliquer sur 'commit change'

Edit file

Write

Preview changes

🔑 master

mon premier script.R

≡ Soft wrap

text ▼

1

bonjour tout le monde

2

j'intercale cette ligne ici

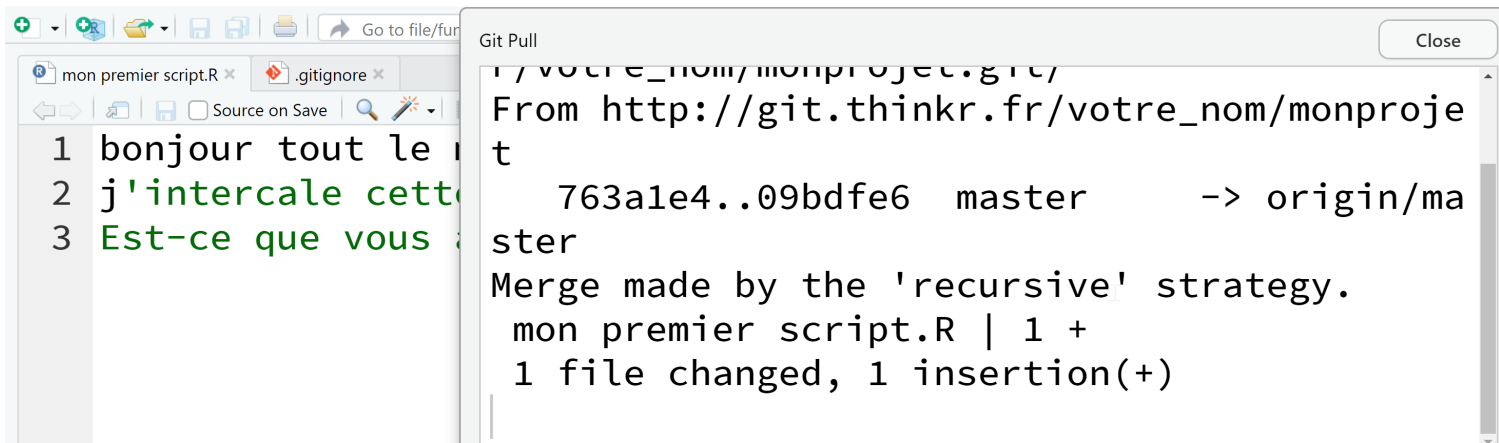
3

Est-ce que vous allez bien ?

Usage de Git

Premier pull

Rendez vous dans Rstudio pour cliquer sur **pull** (flèche verticale descendante), vous verrez alors le fichier `.R` se mettre à jour dans votre projet local



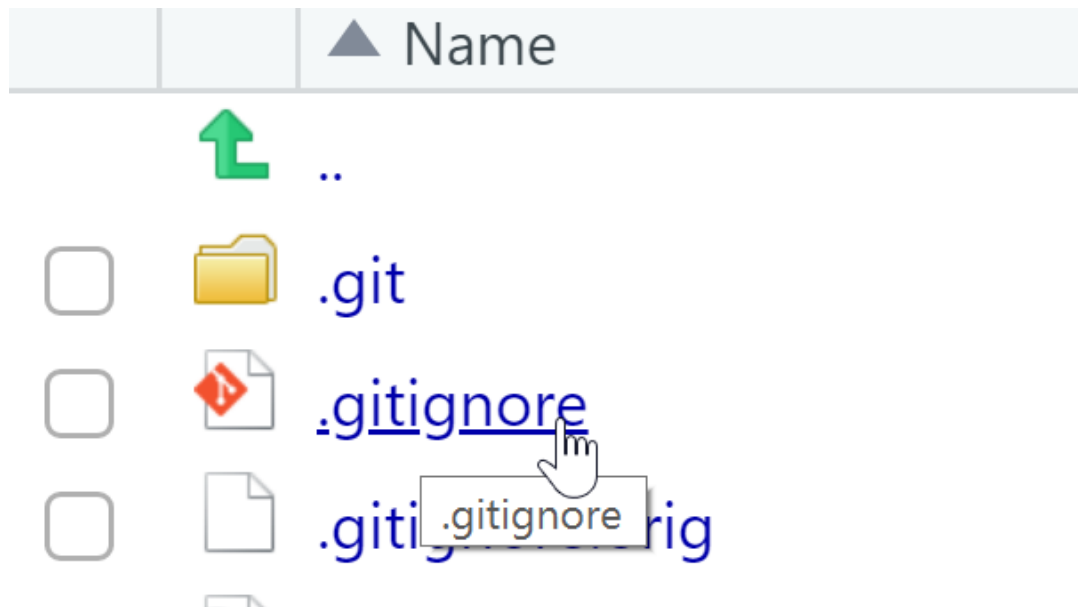
Usage de Git

le .gitignore

Il n'est pas nécessaire de versionner l'intégralité des fichiers de votre projet. Seuls ceux "cochés" seront associés à des **commits**.

Il est possible d'explicitement demander à git de ne pas surveiller tel ou tel fichier.

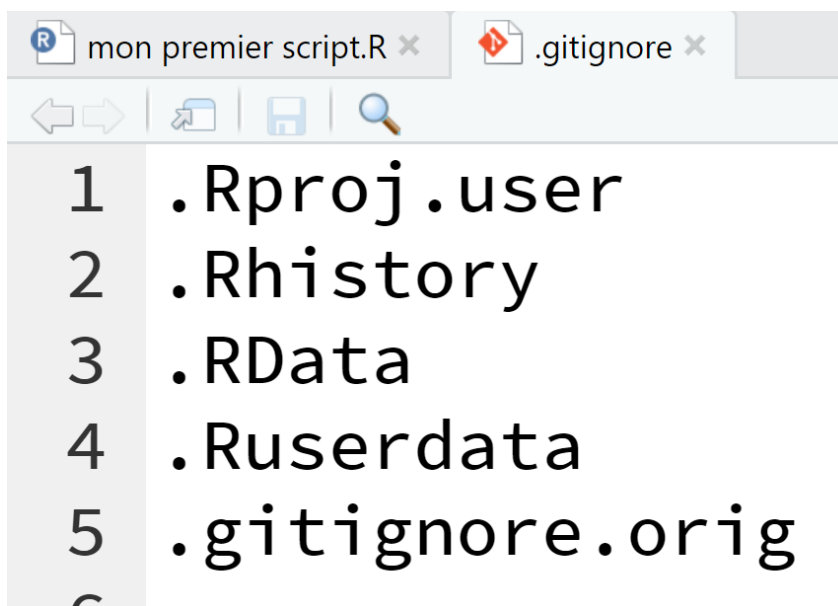
C'est le rôle du fichier `.gitignore` qui se trouve à la racine de votre projet.



Usage de Git

le .gitignore

Il s'agit d'un fichier texte qui accepte les expressions régulières et permet de définir des règles qui correspondent à plusieurs fichiers.



```
1 .Rproj.user
2 .Rhistory
3 .RData
4 .Ruserdata
5 .gitignore.orig
```

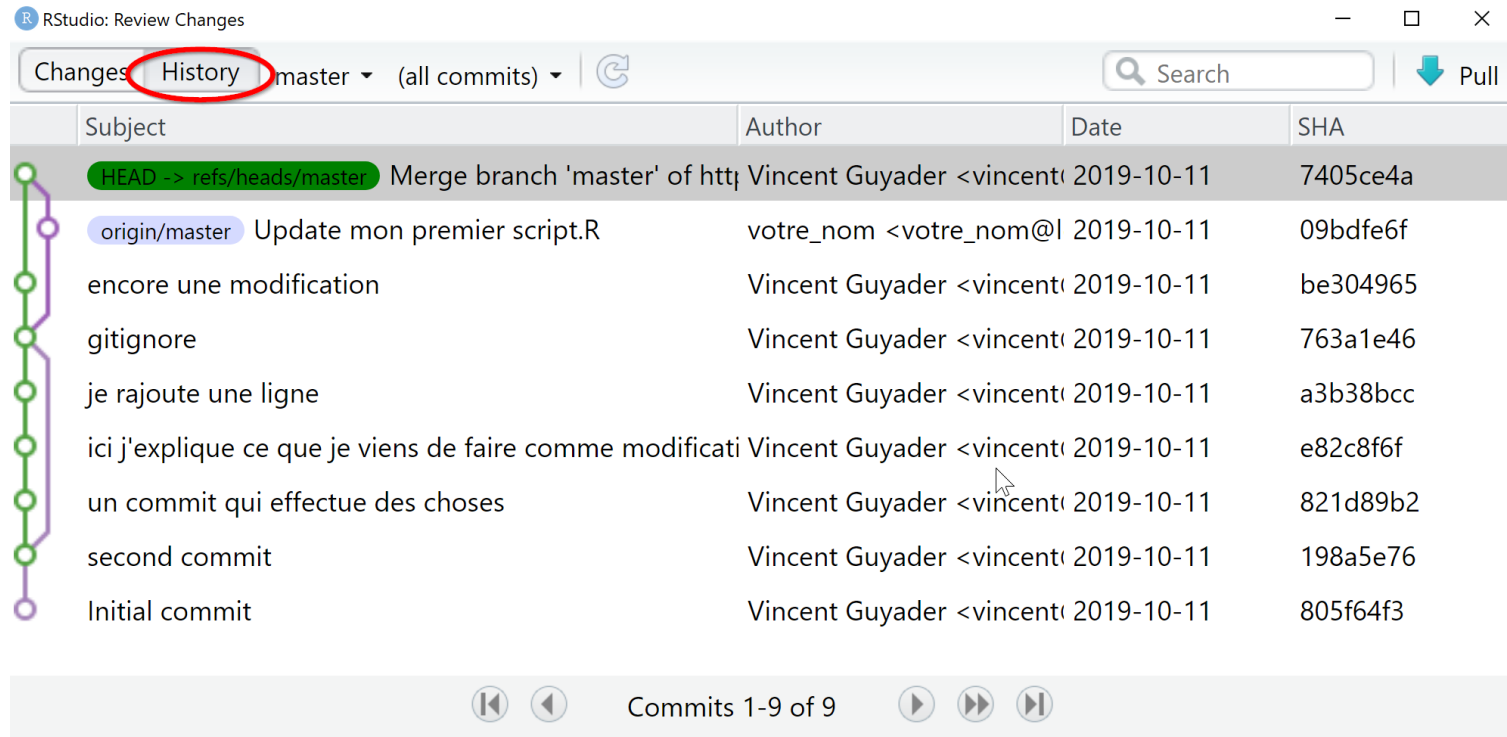
Vous pouvez éditer ce fichier directement ou utiliser dans R la commande :

```
usethis::use_git_ignore("fichier.R")
```


Usage de Git

Voir l'historique

En cliquant sur **commit** puis history, vous verrez la succession de **commits** réalisés.



RStudio: Review Changes

Changes **History** master (all commits) Search Pull

Subject	Author	Date	SHA
HEAD -> refs/heads/master Merge branch 'master' of http	Vincent Guyader <vincent@	2019-10-11	7405ce4a
origin/master Update mon premier script.R	votre_nom <votre_nom@l	2019-10-11	09bdf6f
encore une modification	Vincent Guyader <vincent@	2019-10-11	be304965
gitignore	Vincent Guyader <vincent@	2019-10-11	763a1e46
je rajoute une ligne	Vincent Guyader <vincent@	2019-10-11	a3b38bcc
ici j'explique ce que je viens de faire comme modificati	Vincent Guyader <vincent@	2019-10-11	e82c8f6f
un commit qui effectue des choses	Vincent Guyader <vincent@	2019-10-11	821d89b2
second commit	Vincent Guyader <vincent@	2019-10-11	198a5e76
Initial commit	Vincent Guyader <vincent@	2019-10-11	805f64f3

Commits 1-9 of 9

Usage de Git

Voir l'historique

Les message de `commits` sont très importants



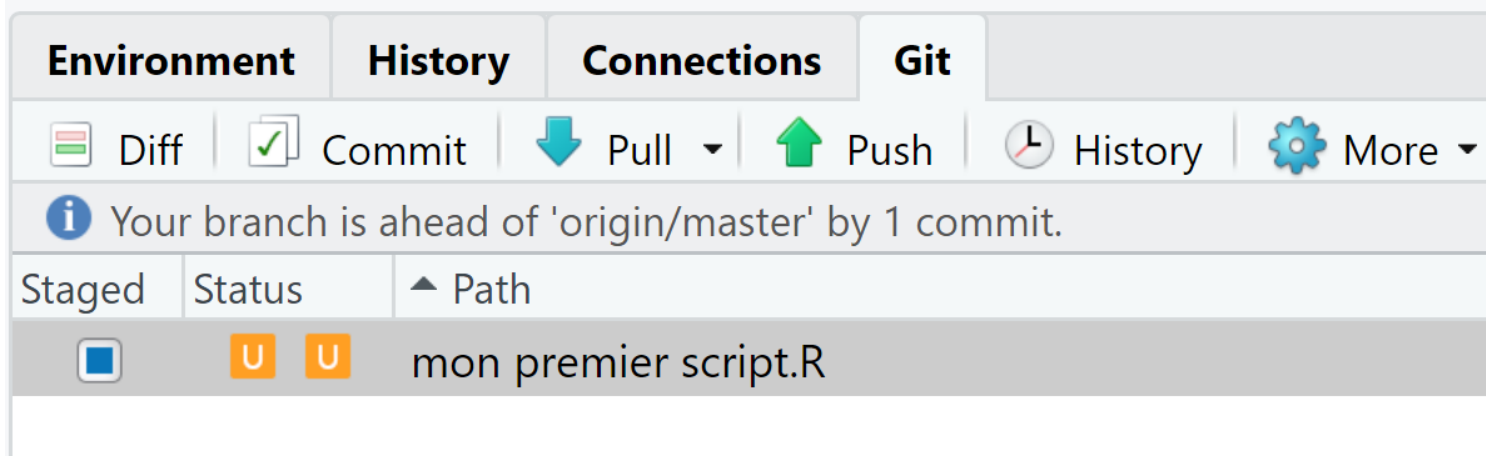
	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFT	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Usage de Git

Premier conflit

Ecrire à **plusieurs** en **même temps** sur le **même fichier**, même avec un outil dédié reste quelque chose de "problématique". Parfois les modifications que vous allez récupérer avec un **pull** contredisent des modifications que vous avez effectuées de votre côté en local. Il se produit alors un **conflit**.



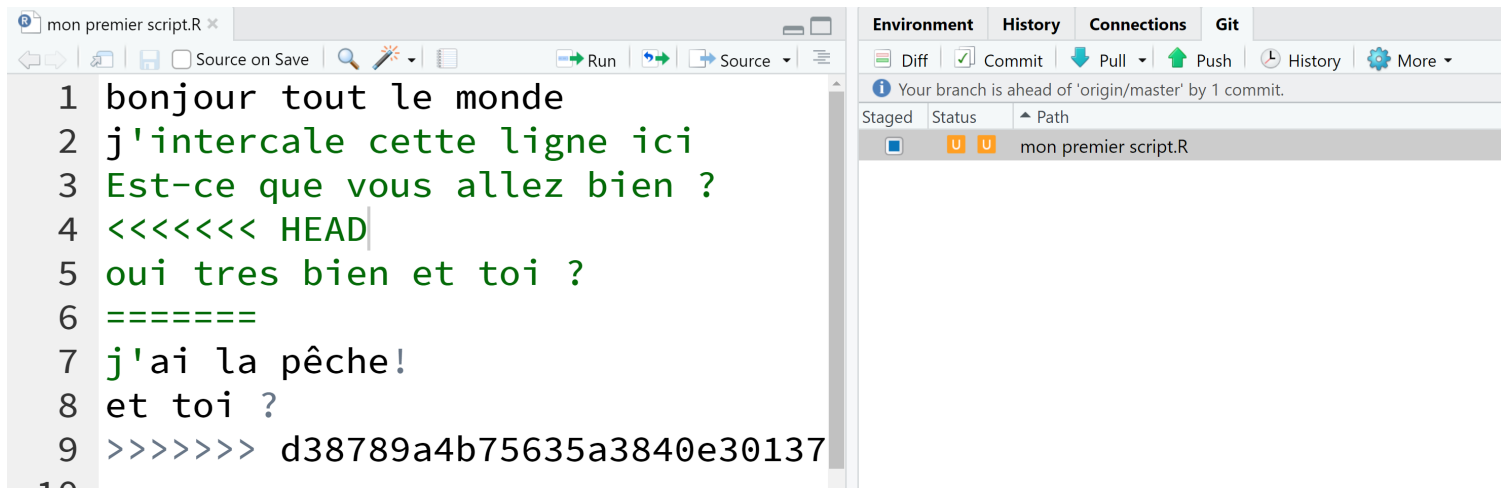
Le fichier est associé à une icône orange avec un **U**, et son contenu peut être un peu déroutant... Les conflits sont des lignes de codes qui diffèrent d'une version à l'autre. La résolution des conflits est **à la charge de l'utilisateur**.

Usage de Git

Premier conflit

Pour générer un conflit :

1. Depuis Rstudio modifier un fichier, faire un **commit**
2. Depuis l'interface web modifier ce même fichier, en racontant autre chose et faire 'commit change'
3. Depuis Rstudio faire un **pull**



The screenshot shows the RStudio interface. The script editor on the left contains the following text:

```
1 bonjour tout le monde
2 j'intercale cette ligne ici
3 Est-ce que vous allez bien ?
4 <<<<<< HEAD
5 oui tres bien et toi ?
6 =====
7 j'ai la pêche!
8 et toi ?
9 >>>>>> d38789a4b75635a3840e30137
```

The Git pane on the right shows the 'Environment' tab. It displays the message: 'Your branch is ahead of 'origin/master' by 1 commit.' Below this, the 'Staged' tab is selected, showing a file named 'mon premier script.R' with a status of 'U' (Unmerged).

Usage de Git

Premier conflit

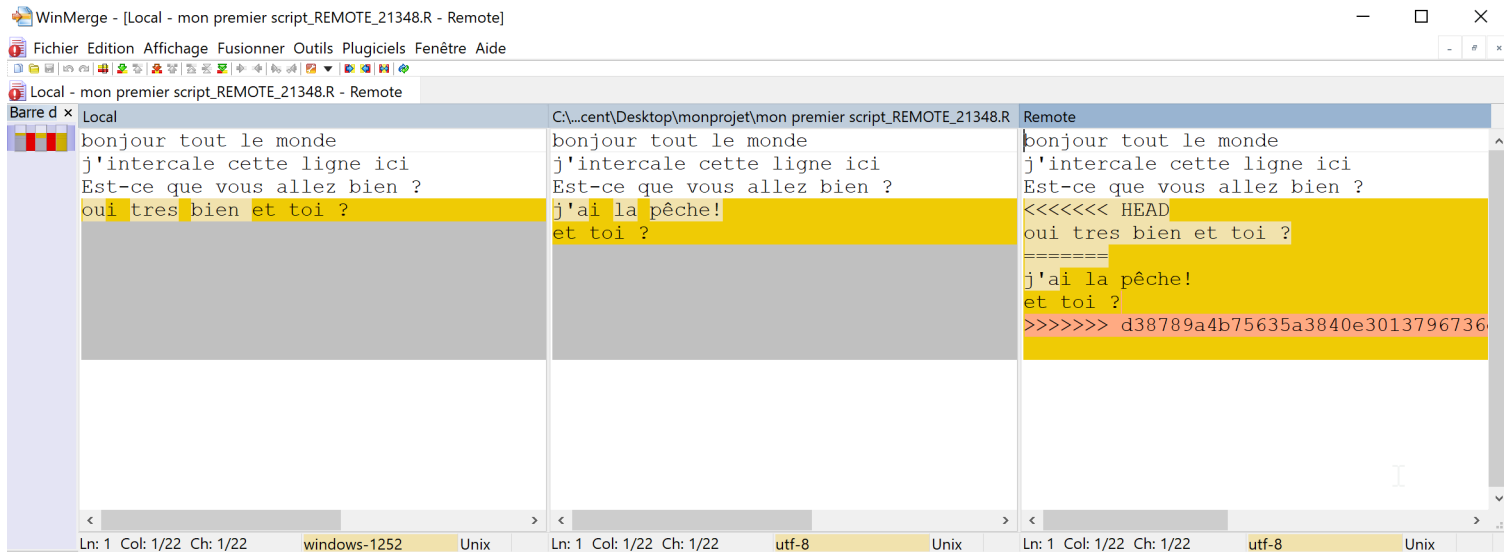
Dans le fichier on distingue des parties délimitées par des `<<<<` et des `>>>>`, et vont s'entrecroiser les lignes qui viennent des `commits` locaux et des `commits` distants.

Pour résoudre un conflit, 2 solutions :

1. Pour les cas simples, on peut éditer directement, à la main le fichier en prenant soin de bien supprimer les `<<<<` et `>>>>` (*ctrl + D permet de supprimer une ligne dans RStudio*). Une fois la modification effectuée, il suffit de recocher le fichier et de refaire un `commits`.
2. Pour les cas plus complexes, un outil peut être nécessaire. Nous allons utiliser winmerge. Pour le lancer depuis le terminal il faut lancer `git mergetool`

Usage de Git

Premier conflit



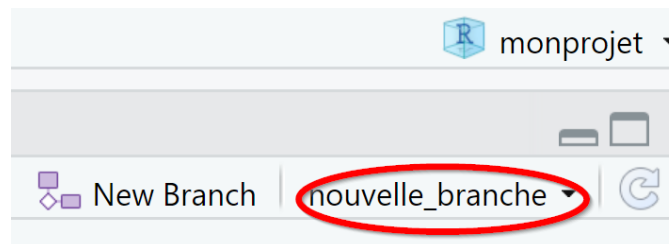
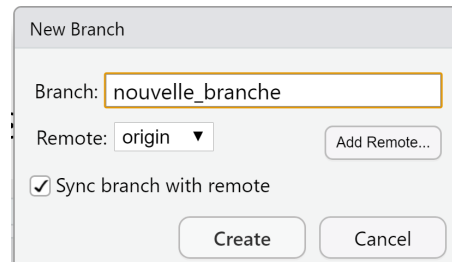
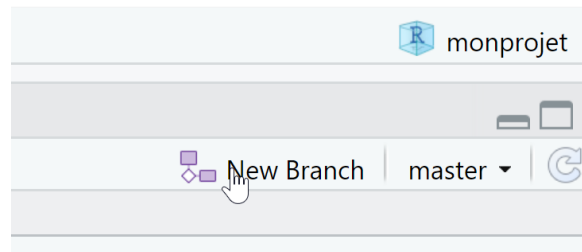
The screenshot shows the WinMerge application window titled "WinMerge - [Local - mon premier script_REMOTE_21348.R - Remote]". The menu bar includes "Fichier", "Edition", "Affichage", "Fusionner", "Outils", "Plugiciels", "Fenêtre", and "Aide". The toolbar contains various icons for file operations. The main window displays three panes: "Local", "C:\...cent\Desktop\monprojet\mon premier script_REMOTE_21348.R", and "Remote". The "Local" pane shows the text: "bonjour tout le monde", "j'intercale cette ligne ici", "Est-ce que vous allez bien ?", and "oui tres bien et toi ?". The "Remote" pane shows the text: "bonjour tout le monde", "j'intercale cette ligne ici", "Est-ce que vous allez bien ?", "oui tres bien et toi ?", "j'ai la pêche!", and "et toi ?". The "Remote" pane also shows a commit hash "d38789a4b75635a3840e3013796736". The status bar at the bottom shows "Ln: 1 Col: 1/22 Ch: 1/22" for each pane, with "windows-1252" and "utf-8" encodings and "Unix" line endings.

Winmerge permet de voir simultanément les 3 fichiers, le local, le distant et celui qui sera conservé. Il dispose d'outil graphique permettant de choisir ligne à ligne les modifications à accepter ou non.

Usage de Git

Première branche

Pour l'heure nous travaillons directement dans l'unique branche 'master'. Nous allons créer une branche depuis Rstudio. Cette branche permettra de travailler sur le projet sans toucher, pour le moment, à la branche 'master'.

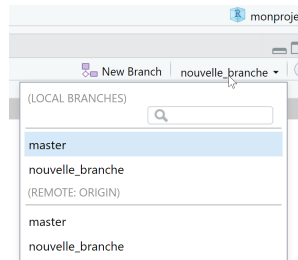


Usage de Git

Première branche

Dans cette 'nouvelle branche', effectuez des modifications et des **commits**. Modifiez des fichiers rajoutez-en des nouveaux et faites un **push**.

Retournez sur la branche master



Vous remarquerez que le contenu de votre projet local a été remis dans le même état qu'au moment où vous avez créé la 'nouvelle branche'.

Profitez-en pour rajouter un nouveau fichier dans la branche **master**, faites un **commits** et un **push**.

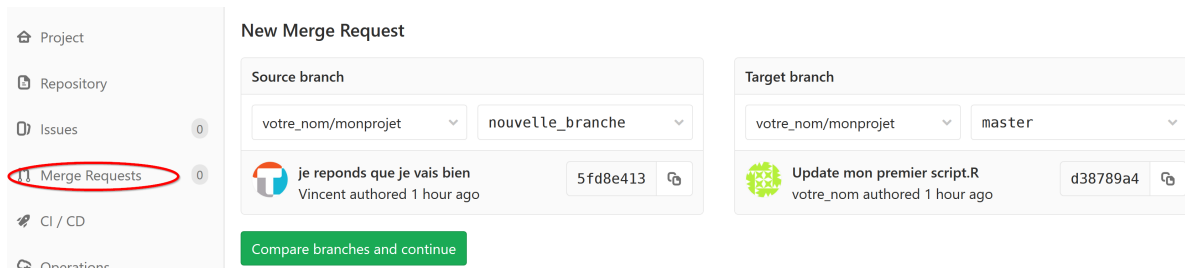
Usage de Git

Premier merge

Les 2 branches 'master' et 'nouvelle branche' diffèrent de plus en plus, nous allons maintenant verser dans master les modifications effectuées dans 'nouvelle branche' : nous allons faire un **merge**

2 façons de faire :

1. Via l'interface web de gitlab, en cliquant sur 'Merge Request'



Une merge request ne devrait jamais être validée par l'auteur de la modification. Il s'agit d'une bonne occasion pour qu'une autre personne dans le projet vérifie puis accepte la modification effectuée sur 'master'. gitlab rend ce processus 'user friendly'.

Dans gitlab on parle de 'merge request', dans github de 'pull request'. Il s'agit de la même chose.

Usage de Git

Premier merge

1. Depuis le terminal

```
git checkout master # pour se positionner dans master si ce n'est pas  
git merge nouvelle_branche # pour rappatrier le contenu de nouvelle_b
```

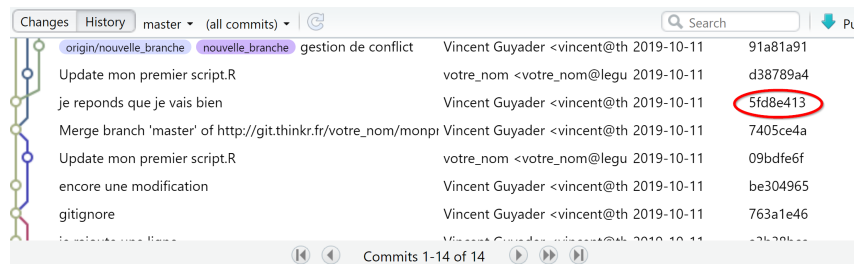
Des conflits peuvent là-aussi avoir lieu, ils sont à régler avant d'aller plus loin.

Usage de Git

Premier checkout

Git vous permet de revenir en arrière et de **voyager dans l'histoire de votre projet**. Il est tout à fait possible de revenir à l'état du projet d'il y a quelques jours pour créer une nouvelle branche à cet endroit.

Pour vous repositionner sur un **commit** particulier, il vous faut trouver son **hash**. il est visible dans la colonne 'SHA1' dans le volet history



On peut alors dans le terminal effectuer un **checkout**

```
git checkout 5fd8e
```

Si l'on souhaite effectuer des modifications à partir de ce **commit** il convient d'y créer une branche, afin d'éviter de perdre le contact avec les autres **commits** plus récents.

Ressources et crédits

- <https://xkcd.com/>
- <http://geek-and-poke.com/>