

➤ Connecter RStudio à GitLab

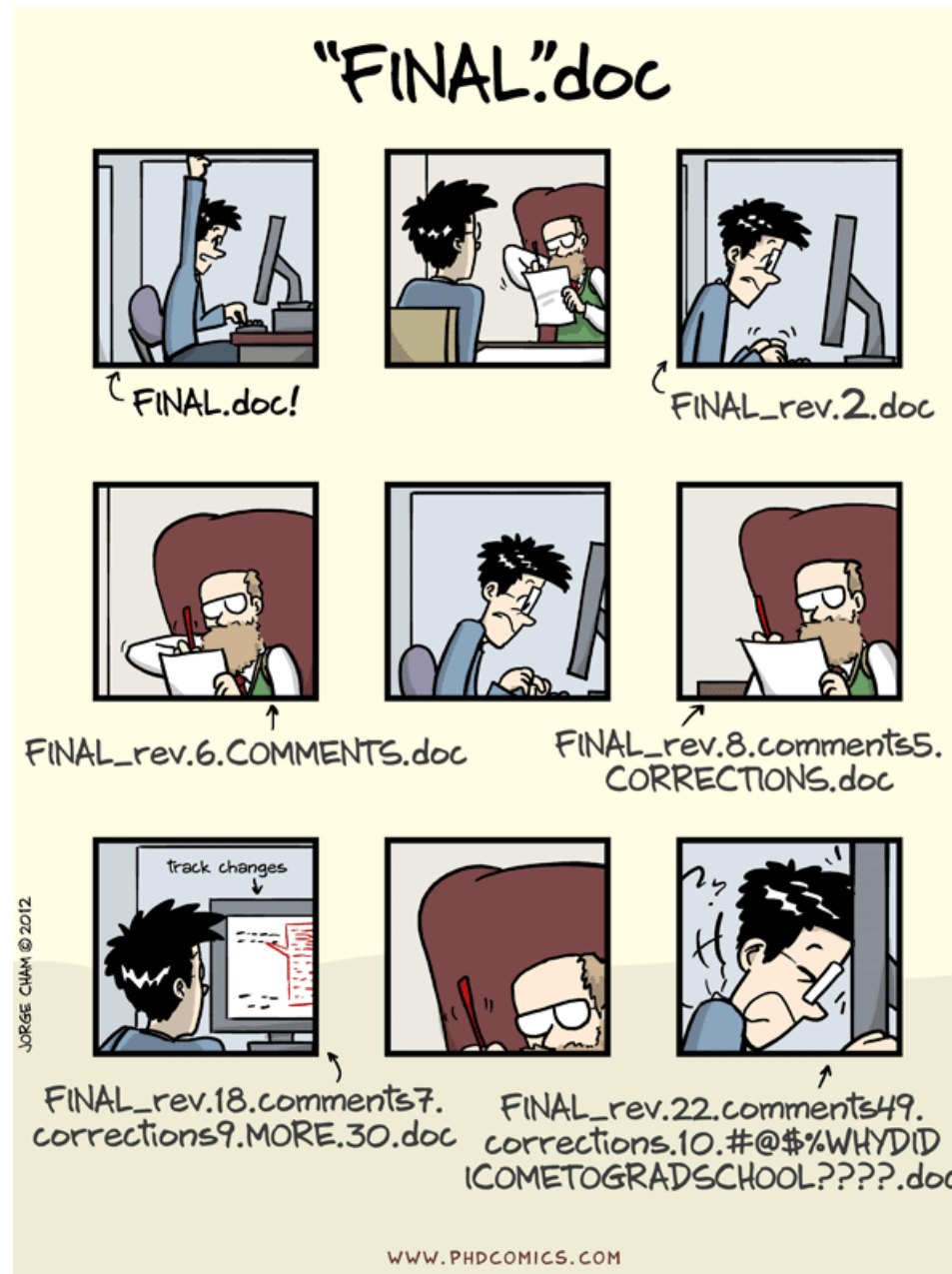
Intégrer Git à son environnement de travail



➤ Contrôle de version

Pourquoi faire ?

- Cycle de vie d'un document (script)
 - Créer de nouvelles fonctions
 - Modifier des fonctions existantes
 - Éditer le document
 - Sauver ces nouveautés/modifications
- Qui/Quand/Pourquoi/Comment ont été faites ces modifications ?
 - Historique du document
- Travail collaboratif facilité



➤ Git : qu'est-ce que c'est ?

- Logiciel de gestion de version
 - Le plus populaire et le plus rapide actuellement
- Basé sur des "snapshots"
 - Seuls les fichiers modifiés sont tracés
 - Mini système de fichier
 - On ne sauve pas directement les modifications
- Pas de serveur requis : tout est en local
 - On peut sauver (*commit*) ses modifications même si on n'a pas de connexion internet/réseau.
- intégrité des fichiers
 - *Checksum* : absolument toutes les modifications seront vues par git
 - Impossible de corrompre un fichier sans s'en rendre compte



➤ Git : client en ligne de commande

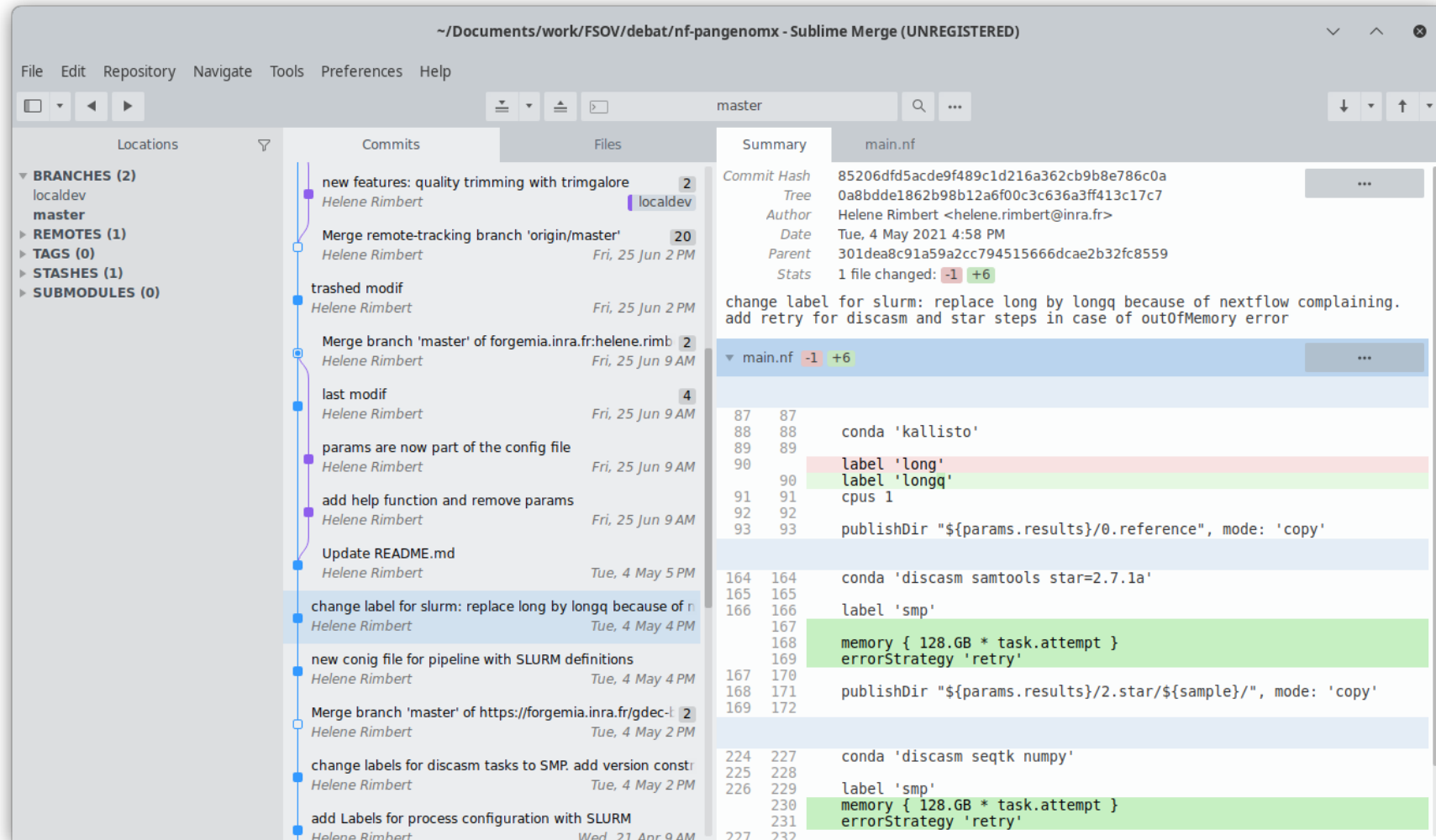
Installation

- Linux (debian-based, Ubuntu)
\$ sudo apt install git-all
<https://git-scm.com/download/linux>
- Mac OS
<https://git-scm.com/download/mac>
- Windows : Git for Windows
<https://gitforwindows.org/>



➤ Git : client graphique

Sublime Merge <https://www.sublimemerge.com/>



➤ Git : notions importantes

Working directory / staging (index) / repository (database)

- 3 statuts possibles pour un fichier dans git
 - Modifié: modifications non suivies par git
 - Indexé: le fichier modifié est signalé à git avant intégration à la db
 - *Commité*: la nouvelle version du fichier est ajoutée au dépôt/db

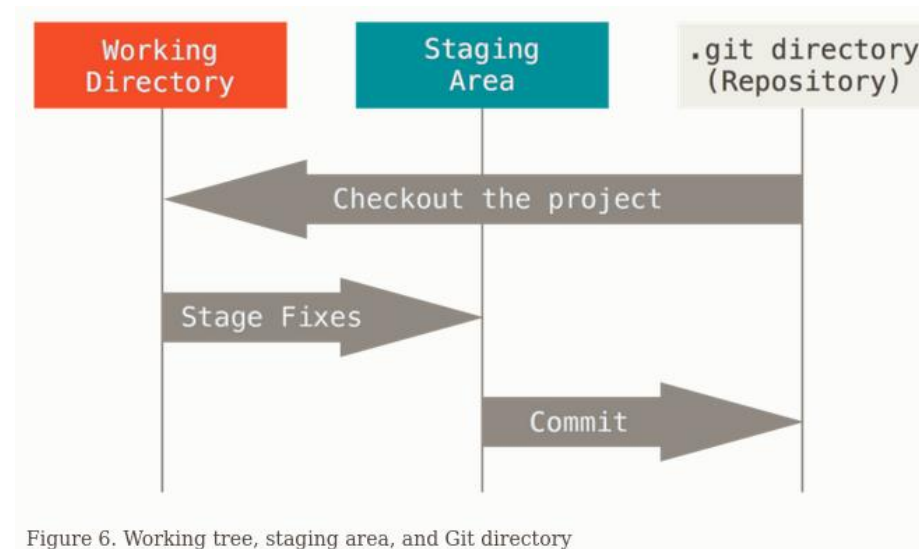


Figure 6. Working tree, staging area, and Git directory

<https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>

➤ Git : notions importantes

Working directory / staging (index) / repository (database)

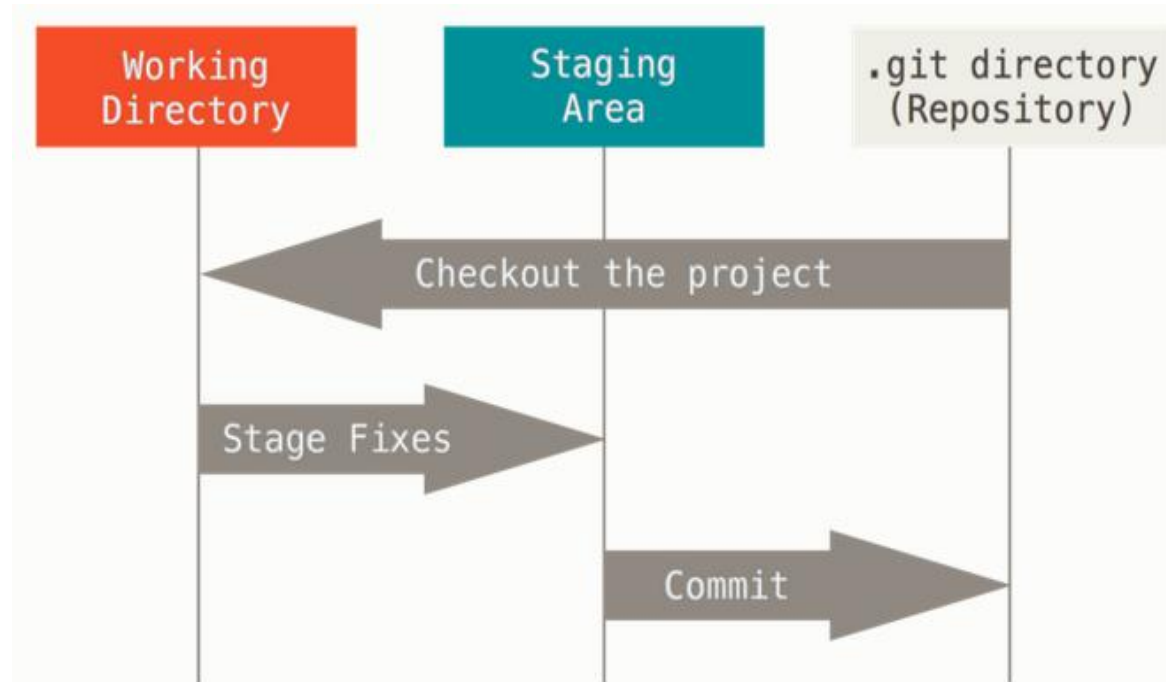


Figure 6. Working tree, staging area, and Git directory

<https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F>

1. Modification
2. Ajout dans la *staging area* des modifications que l'on va vouloir sauver
3. Sauvegarde dans la DB de fichiers modifiés (commit)



➤ Git : premiers pas

Configuration utilisateur

Définition du nom d'utilisateur:

```
$ git config --global user.name "Helene Rimbart"
```

Définition de l'adresse mail

```
$ git config --global user.email helene.rimbart@inrae.fr
```

Afficher la configuration du compte git

```
$ git config --list  
user.name=Helene Rimbart  
user.email=helene.rimbart@inra.fr  
color.ui=true
```



➤ Git : premiers pas

Initialiser un dépôt à partir d'un répertoire existant

```
$ cd /home/hrimberty/devs/my_project
```

```
$ git init
```

Nouveau dossier caché `.git/` (conf, database de suivi de version etc ...)

Indexer les fichiers que l'on veut versionner (=> *staging*)

```
$ git add *.py
```

```
$ git add README.md
```

Sauver les fichiers ajoutés à l'index

```
$ git commit -m "version initiale de mon projet"
```



➤ Git : premiers pas

Récupérer une copie d'un dépôt existant (depuis gitlab par exemple)

```
$ git clone https://gitlab.com/hrimberty/my_project
```

Crée un dossier "my_project" contenant tous les fichiers du dépôt

Si on veut cloner le projet dans un autre répertoire

```
$ git clone https://gitlab.com/hrimberty/my_project my_project_copy2
```

Plusieurs protocoles de transferts possibles:

- https://
- git://
- user@server:path/to/repository.git

➤ Git : enregistrer des modifications

\$ git status : Show the working tree status

\$ git add/ git commit

	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
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAAAANDS	2 HOURS AGO

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

➤ Git : enregistrer des modifications

\$ git status : Show the working tree status

\$ git add/ git commit

```
$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   main.nf
#       modified:   nextflow.config
#       modified:   run.sh
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       dag.html
#       singularity/
#       test_data/conta.diamondidx.dmnd
no changes added to commit (use "git add" and/or "git commit -a")
```

➤ Git : enregistrer des modifications

\$ git status : Show the working tree status

\$ git add/ git commit

```
$ git add dag.html
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#       new file:   dag.html
#
# Changes not staged for commit:
#   (use "git add/rm <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   main.nf
#       modified:   nextflow.config
#       modified:   run.sh
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       singularity/
#       test_data/conta.diamondidx.dmnd
no changes added to commit (use "git add" and/or "git commit -a")
$ git commit -m "new dag.html diagram"
```

➤ Git : enregistrer des modifications

\$ git status : Show the working tree status

\$ git add/ git commit

```
$ git commit -m "new dag.html diagram"
[master 6eeb1f8] new dag.html diagram
1 file changed, 268 insertions(+)
create mode 100644 dag.html
```

On peut regarder l'historique des commits

```
$ git log
commit 6eeb1f88fe86002de70c6b4efadf082d47f6e5bd
Author: Helene Rimbart <helene.rimbart@inra.fr>
Date:   Fri Sep 3 14:51:03 2021 +0200
```

new dag.html diagram

```
commit 38d005c1fd6b0fd1b3adf2c066aa97115be44c40
Merge: d7ecff9 4c90cd5
Author: Helene Rimbart <helene.rimbart@inra.fr>
Date:   Thu Jul 15 13:28:53 2021 +0200
```

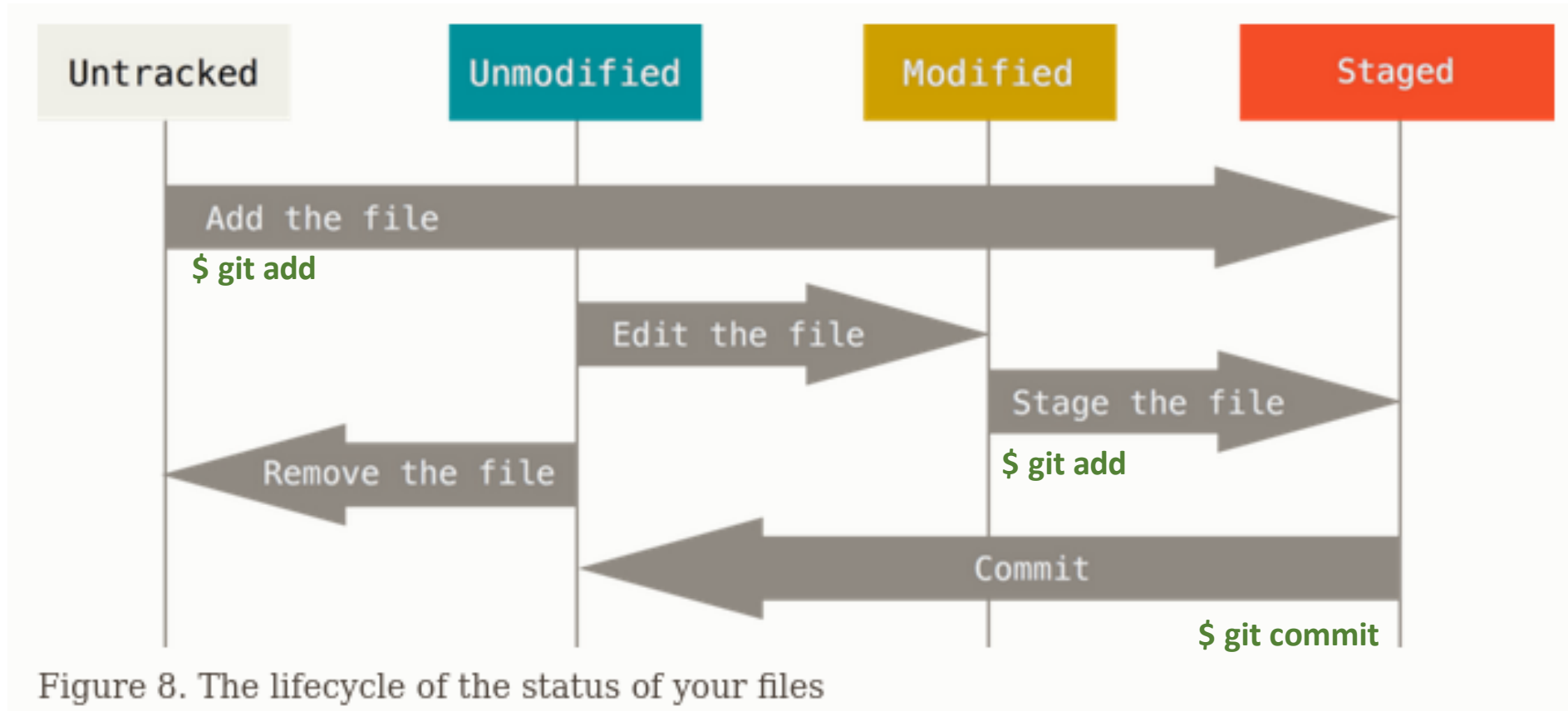
Merge branch 'master' of <https://forgemia.inra.fr/gdec-bioinfo/nf-pantranscriptomix>

```
$ git log --pretty=oneline
6eeb1f88fe86002de70c6b4efadf082d47f6e5bd new dag.html diagram
38d005c1fd6b0fd1b3adf2c066aa97115be44c40 Merge branch 'master' of https://forgemia.inra.fr/gdec-bioinfo/nf-pantranscriptomix
```

Checksum : ID du commit

➤ Git : enregistrer des modifications

Cycle de vie d'un fichier



<https://git-scm.com/book/en/v2/Git-Basics-Recording-Changes-to-the-Repository>

A circular icon with a teal border containing a black diamond with a white Git branching diagram inside. A teal line curves from the top left towards the bottom left of the slide.

➤ Git : ignorer des fichiers

Fichier .gitignore

- Possibilité d'ignorer des fichiers : **.gitignore**

```
$ cat .gitignore  
work*  
.nextflow*  
slurm*.out  
data/  
test_data/results
```

➤ Git : quelles différences entre le dépôt et ma copie de travail ?

\$ git diff

```
$ git diff main.nf
diff --git a/main.nf b/main.nf
index 4b59b36..2248ef4 100644
--- a/main.nf
+++ b/main.nf
@@ -23,7 +23,7 @@ def helpMessage(){
     --trim_length [INT]

    DISCASM/Trinity contig selection
-    --seqtk_length [INT]
+    --trinity_length [INT]
-    300bp)
+    300bp)

    Decontamination (blast-plus)
    --deconta_blastdb [PATH]
@@ -73,7 +75,7 @@ if (params.fasta){
    Channel
    .fromPath(params.fasta)
    .ifEmpty{exit 1, "Fasta Reference genome not found: ${params.fasta}"}
-    .into { ch_ref_bcftools; ch_faidx}
+    .into { ch_ref_bcftools; ch_faidx; ch_ref_starindex}
}
```

Minimum read length to keep for a read pair (bp, default: 50)

Minimum contig length of Trinity results to keep (default:

Minimum contig length of Trinity results to keep (default:

BLAST index for decontamination (should be ncbi-nr)



➤ Git : ajouter des modifications oubliées au dernier commit

```
$ git commit --amend
```

- Exemple: un bug a nécessité la modification de main.nf et bin/mapping.py

```
$ git add main.nf  
$ git commit -m "bug fix"  
$ git add bin/mapping.py  
$ git commit --amend
```

➤ Git : revenir à la version précédente d'un fichier

```
$ git checkout -- <file>
```



```
$ git status
```

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

```
    modified:   CONTRIBUTING.md
```

```
$ git checkout -- CONTRIBUTING.md
```

```
$ git status
```

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

```
    renamed:    README.md -> README
```

git checkout -- <file> va supprimer toutes les modifications faites localement par la version du dernier commit !



➤ Git : revenir à la version précédente d'un fichier (git >=2.23.0)

```
$ git restore --staged <file>
```

```
$ git restore <file>
```



```
$ git add *
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   CONTRIBUTING.md
    renamed:    README.md -> README

$ git restore --staged CONTRIBUTING.md
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:    README.md -> README

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:   CONTRIBUTING.md
```

git restore (--staged) <file> va supprimer toutes les modifications faites localement par la version du dernier commit !



➤ Git : revenir à la version précédente d'un fichier (git >=2.23.0)

```
$ git restore --staged <file>
```

```
$ git restore <file>
```



```
$ git status
```

```
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:    CONTRIBUTING.md
```

```
$ git restore CONTRIBUTING.md
```

```
$ git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git restore --staged <file>..." to unstage)
```

```
renamed:    README.md -> README
```

git restore (--staged) <file> va supprimer toutes les modifications faites localement par la version du dernier commit !



➤ Git : utiliser un dépôt distant (Gitlab par exemple)

\$ git remote / git remote add <URL> / git pull / git push

Connecter son dépôt local à un dépôt distant

```
$ git remote add origin git@gitlab.com:hrimbert/my_project.git
$ git remote -v
origin      https://gitlab.com/hrimbert/my_project.git (fetch)
origin      https://gitlab.com/hrimbert/my_project.git (push)
```

Télécharger les modifications du remote dans le dépôt local

```
$ git fetch <remote>
```

Récupérer les modifications du remote dans l'espace de travail

```
$ git pull <remote>
```

Envoyer ses modifications sur le dépôt distant (après commit)

```
$ git push
```



➤ Git : créer des tags

```
$ git tag
```

Pointeur sur un commit particulier (par défaut le dernier)

Lister les tags

```
$ git tag
```

```
V1.0
```

```
$ git tag -l "V1.5*"
```

```
V1.5.0
```

```
V1.5.1
```

Créer des tags annotés ou non

```
$ git tag -a v1.0
```

```
$ git tag -a v1.0 -m "stable version 1.4"
```

Voir le détail d'un tag particulier

```
$ git show v1.5.0
```

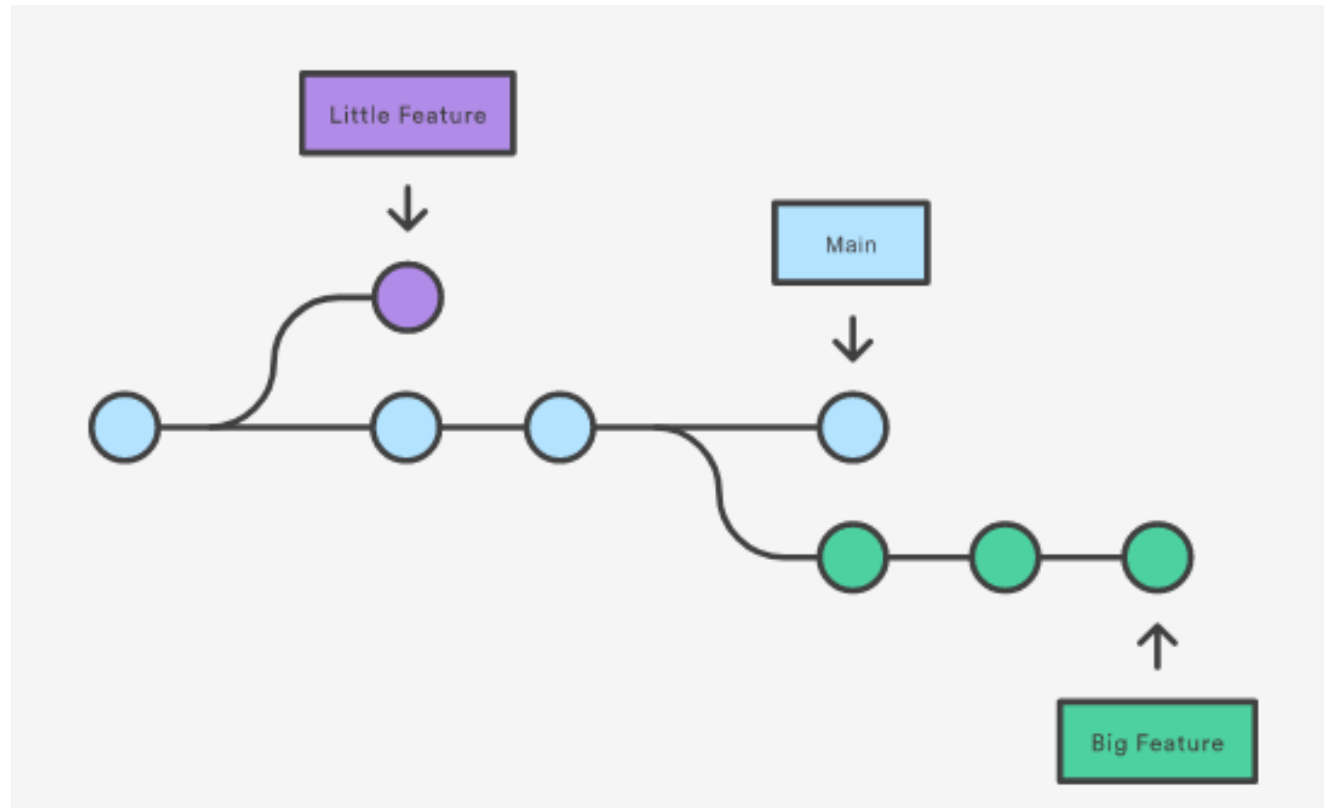
Ajouter un tag à posteriori: ajouter le checksum (ou juste les quelques premiers caractères, voir *git log*)

```
$ git tag -a v1.1 0b7434d86859
```


➤ Git : les branches

\$ git branch / git checkout / git fetch / git merge

Utilisé pour gérer les nouveaux développements, corrections de bugs



<https://www.atlassian.com/git/tutorials/using-branches>



➤ Git : les branches

\$ git branch / git checkout / git fetch / git merge

Utilisé pour gérer les nouveaux développements, correction

Créer une nouvelle branche

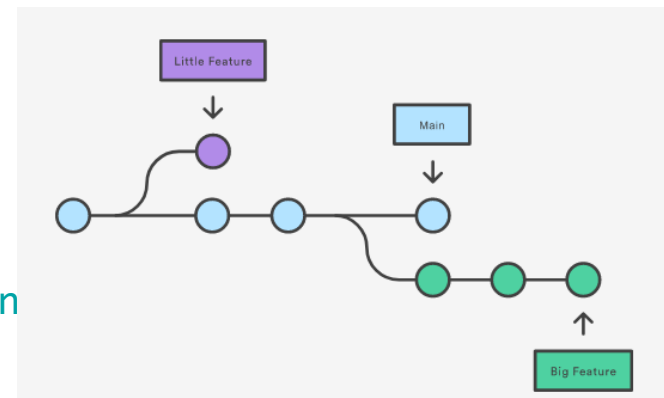
```
$ git branch demo
```

Lister les branches du projet

```
$ git branch --list
```

```
demo
```

```
* master ← Branche actuelle
```



<https://www.atlassian.com/git/tutorials/using-branches>

➤ Git : les branches

\$ git branch / git checkout / git fetch / git merge

Naviguer entre les différentes branches
Mettre à jour les fichiers dans le répertoire de travail
\$ git checkout

Lister les branches du projet

```
$ git branch --list
```

```
demo
* master
```

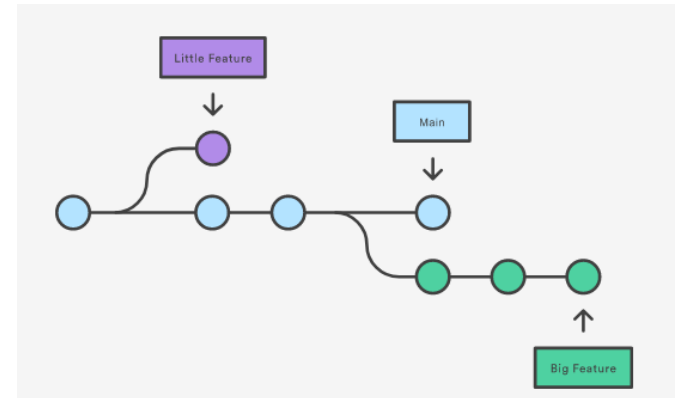
Changer de branche

```
$ git checkout demo
```

```
D bck.nextflow.config
M main.nf
M nextflow.config
M run.sh
M run_test.sh
M test_data/conta.fa.nin
D test_data/conta.fa.nsd
D test_data/conta.fa.nsi
T test_data/ref.fa
Switched to branch 'demo'
```

```
$ git branch --list
```

```
* demo
master
```



<https://www.atlassian.com/git/tutorials/using-branches>

➤ Git : les branches

\$ git branch / git checkout / git fetch / git merge

Si plusieurs branches sur un remote:

Récupérer toutes les branches distantes

```
$ git fetch --all
```

On peut ensuite se positionner sur la branche voulue

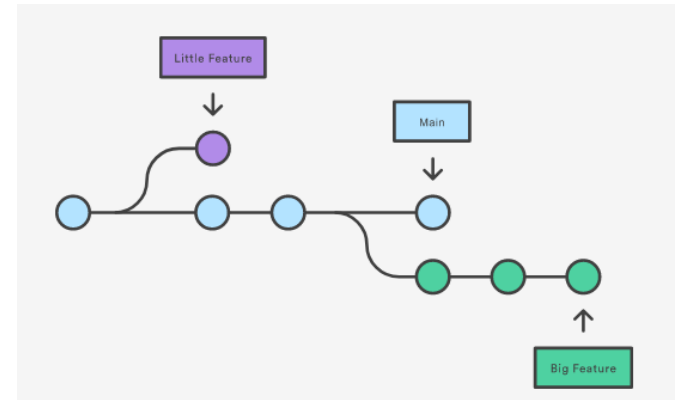
```
$ git checkout remotebranch
```

```
$ git status
```

```
# On branch demo
```

```
# Changes not staged for commit:
```

```
[]...
```



<https://www.atlassian.com/git/tutorials/using-branches>

➤ Git : les branches

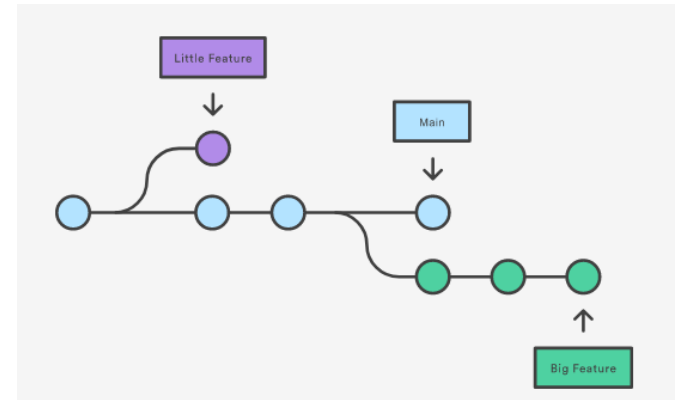
\$ git branch / git checkout / git fetch / git merge

A faire avant de merger 2 branches:

- mettre à jour les deux branches
\$ git fetch -all # récupère toutes les modifs des branches
\$ git pull # récupère toutes les modifs de la branche courante
- Se positionner sur la branche "receveuse":
Si on intègre la branche démo à master, on se positionne sur master
\$ git checkout master

On peut maintenant merger la branche démo sur master

\$ git merge demo



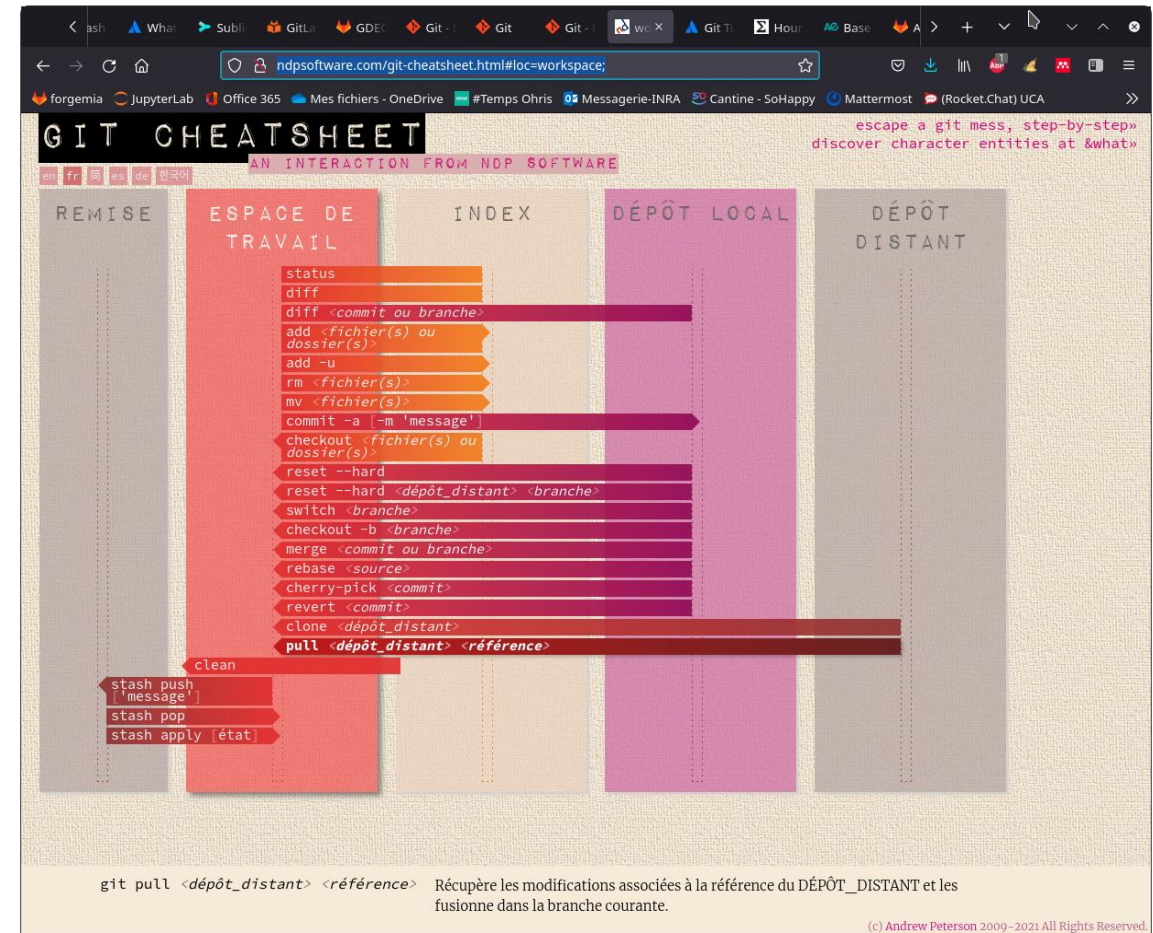
<https://www.atlassian.com/git/tutorials/using-branches>

➤ Git : Ressources

<https://git-scm.com/book/en/v2>

<https://www.atlassian.com/git/tutorials>

<http://ndpsoftware.com/git-cheatsheet.html#loc=workspace;>



➤ Clé SSH

1) Créer une clé :

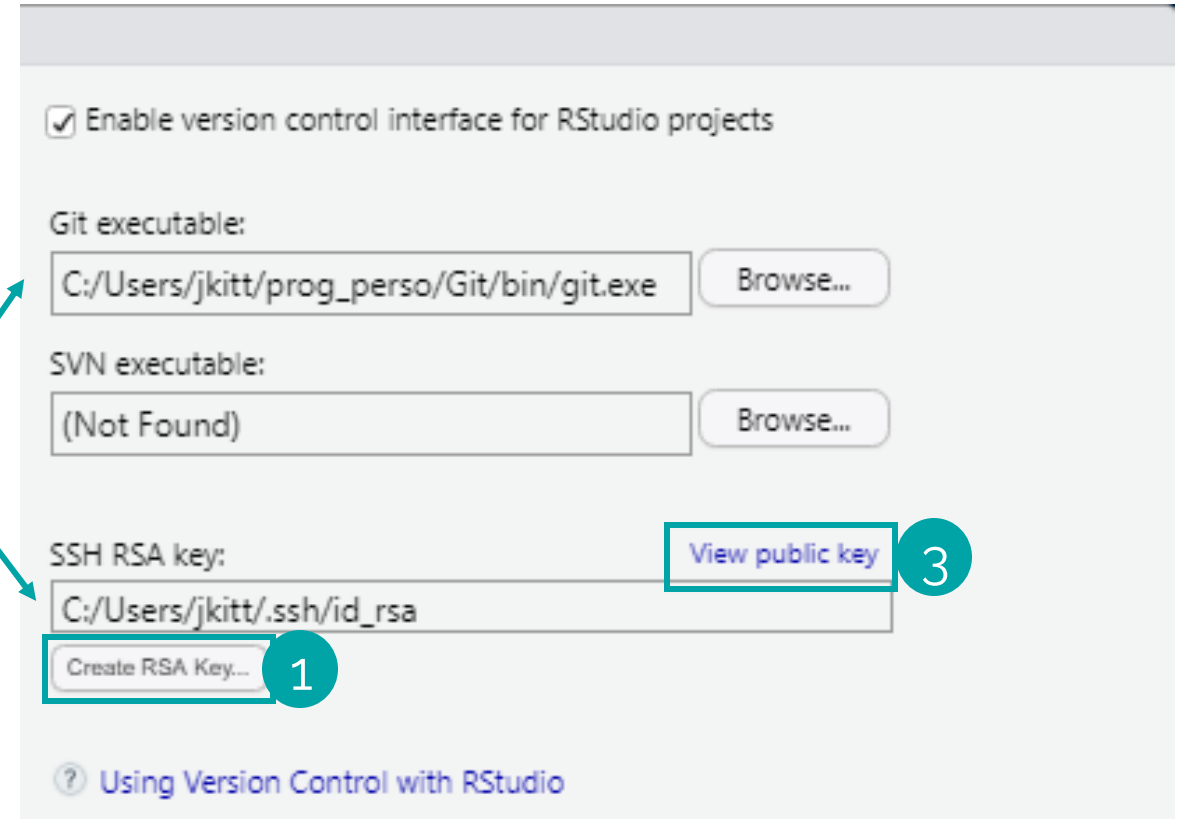
- *Tools > Global options > Git/SVN*
- Create RSA key

2) Vérifier :

- Git executable
- SSH RSA key

3) Copier la clé :

- View public key
- Ctrl + C



➤ Configurer GitLab

- Forgemia : <https://forgemia.inra.fr>
 - Connexion SSO
 - INRAE dans le menu déroulant
 - Identifiants LDAP
- Configuration :
 - *Profile > Preferences > SSH Keys*
 - Coller la clé SSH
 - Identifiants LDAP



Add an SSH key

To add an SSH key you need to [generate one](#) or use an [existing key](#).

Key

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Do not paste your private SSH key, as that can compromise your identity.

Typically starts with "ssh-ed25519 ..." or "ssh-rsa ..."

Title

e.g. My MacBook key

Expires at

jj / mm / aaaa

Give your individual key a title. This will be publicly visible.

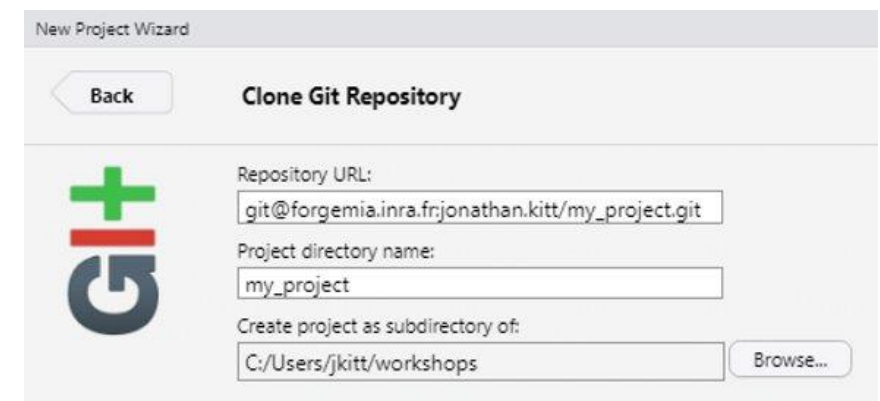
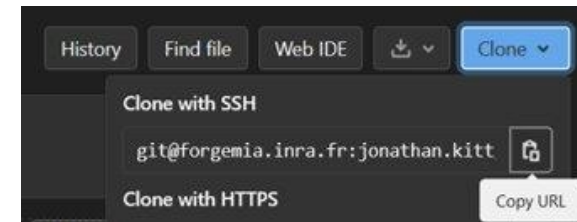
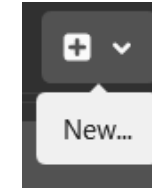
Key can still be used after expiration.

[Add key](#)



➤ GitLab vers RStudio

- Créer un nouveau projet sur GitLab :
 - *New > New project/repository > Create blank project*
 - ☒ *Initialize repository with a README*
 - *Create project*
- Cloner le projet :
 - *Clone > Clone with SSH > Copy URL*
- Dans RStudio :
 - *File > New Project > Version Control > Git*
 - Coller le lien vers le répertoire GitLab



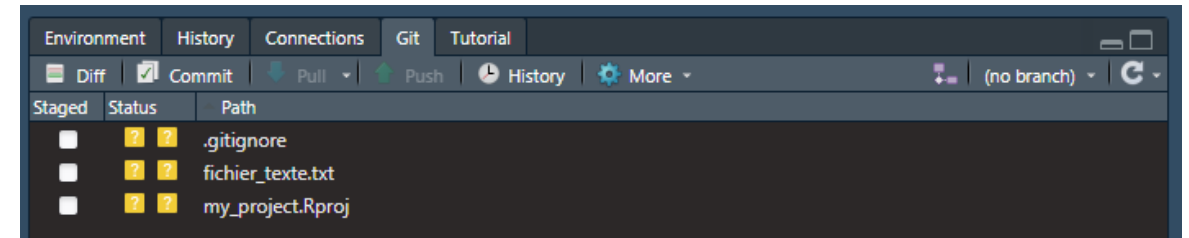
➤ RStudio vers GitLab (1)

- Associer un dossier existant à un nouveau projet :

- *File > New Project > Existing Directory*
- Indiquer le chemin vers le dossier
- *Open in new session*

- Options du projet :

- *Tools > Project Options > Git/SVN*
- *Version control system : Git*
- *Confirm New Git Repository : Yes*
- *Confirm Restart RStudio : Yes*

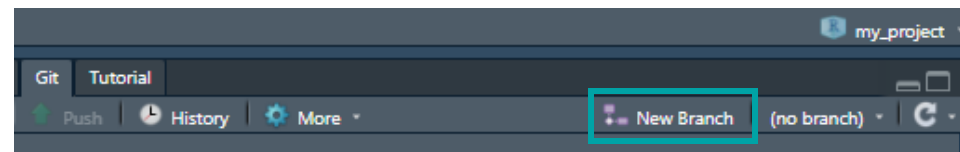
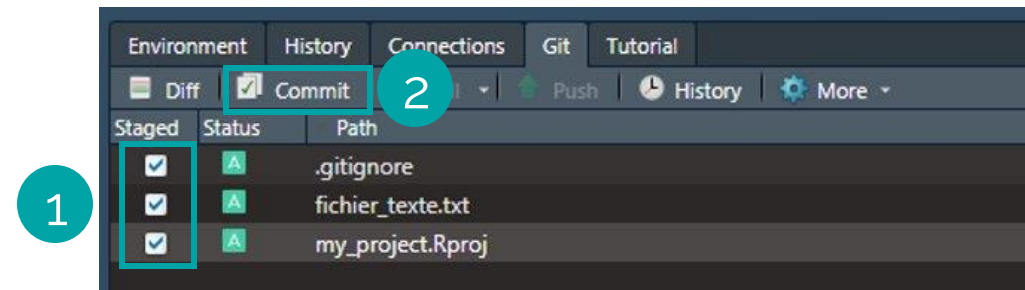


- Créer un nouveau projet sur GitLab :

- *New > New project/repository > Create blank project*
- ☐ *Initialize repository with a README*
- *Create project*

➤ RStudio vers GitLab (2)

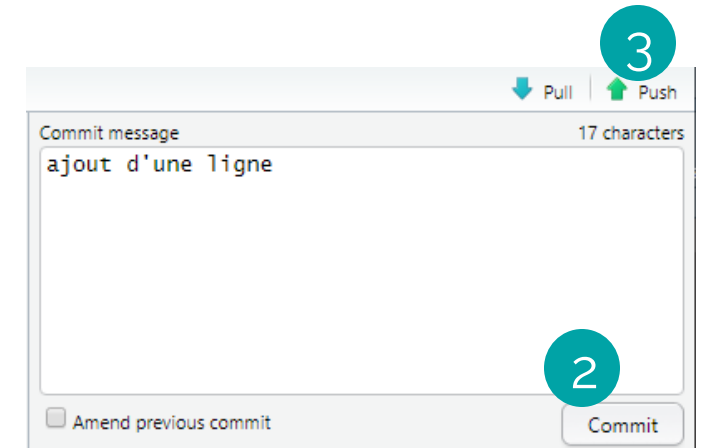
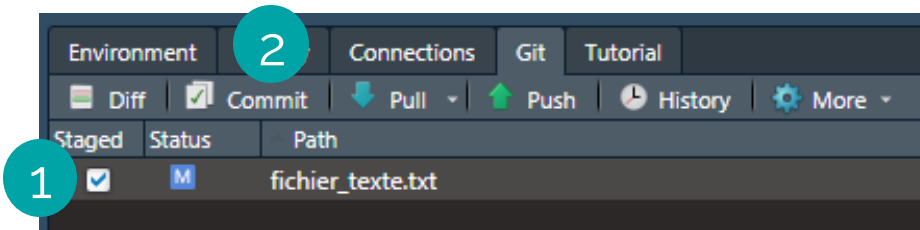
- Cloner le projet :
 - *Clone > Clone with SSH > Copy URL*
- Onglet Git dans RStudio :
 - (1) Sélectionner les fichiers
 - (2) Cliquer sur Commit
 - *Commit message > Commit*
- Créer une nouvelle branche :
 - *New Branch > Add Remote*
 - *Remote Name : origin*
 - *Branch Name : master*
 - *Local Branch Already Exists : Overwrite*



➤ Modifier un fichier en local

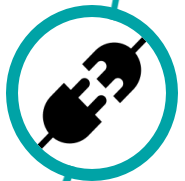
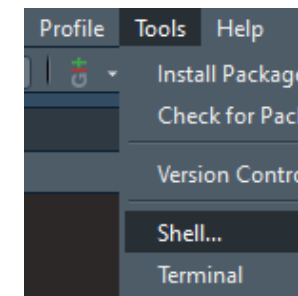
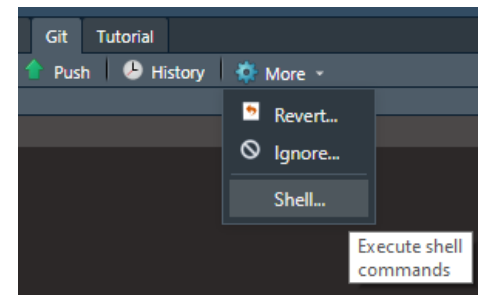
- Editer le fichier
- Workflow "clique-bouton" :

- (1) Stage
- (2) Commit
- (3) Push



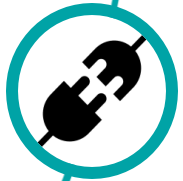
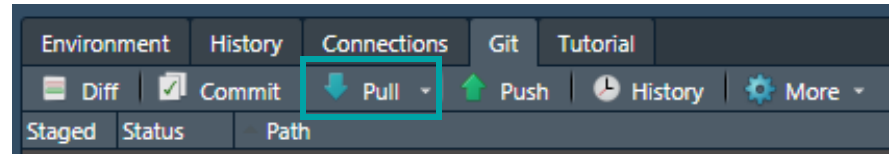
- Workflow shell :

```
> git status  
> git add fichier_texte.txt  
> git commit -m "workflow shell"  
> git push origin master
```

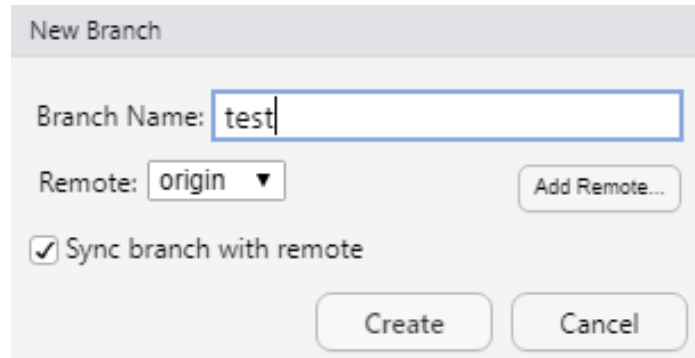


➤ Modifier un fichier sur GitLab

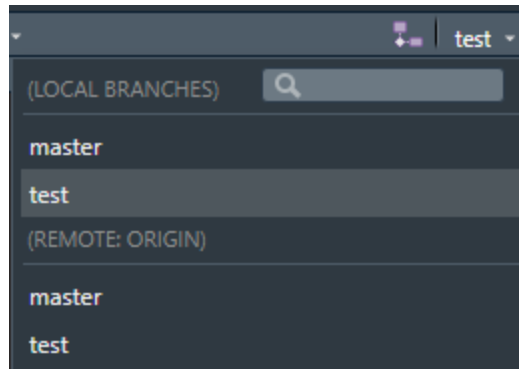
- Editer le fichier :
 - *Edit > ... > Commit changes*
- Workflow "clique-bouton" :
Pull
- Workflow shell :
> `git pull`



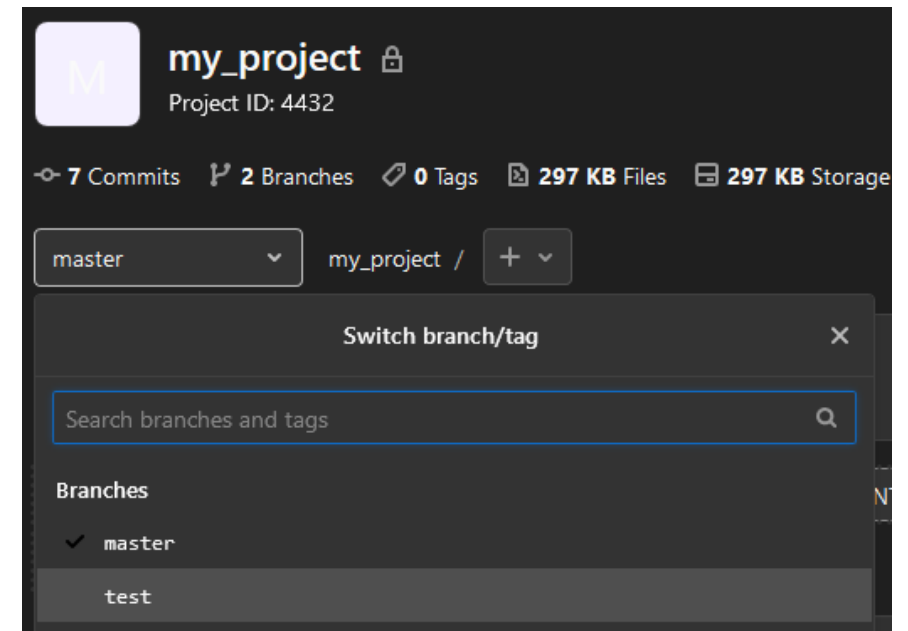
➤ Créer une nouvelle branche (1)



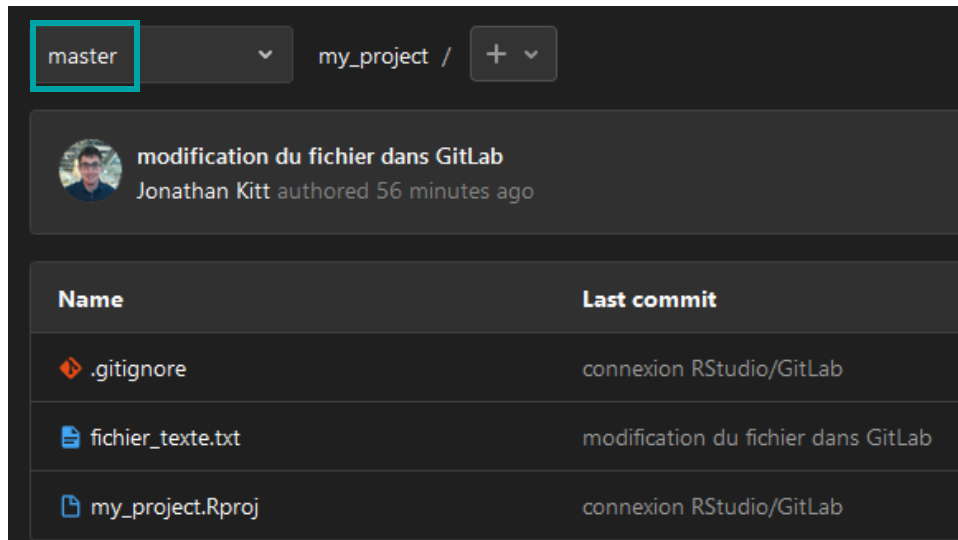
- RStudio






- GitLab

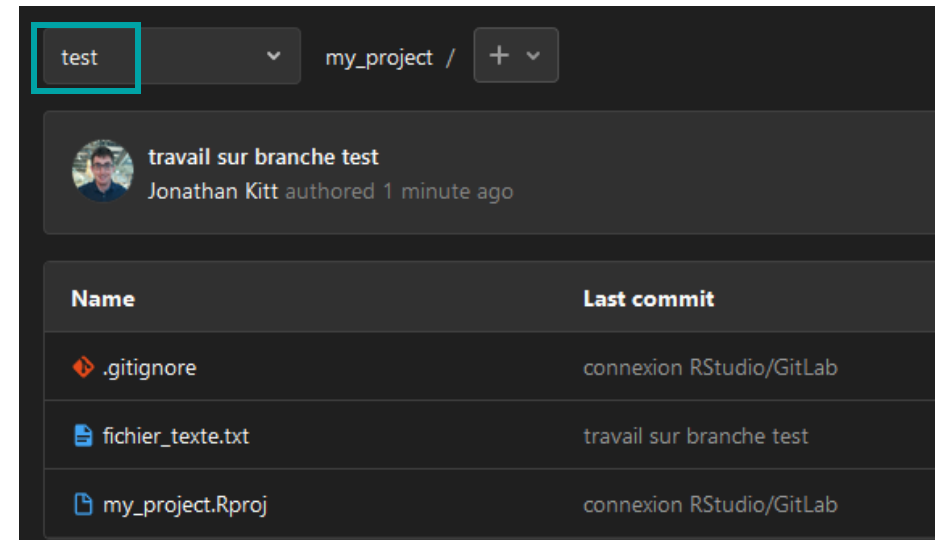


➤ Créer une nouvelle branche (2)






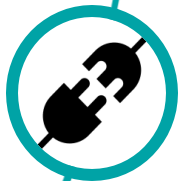
The screenshot shows the GitLab web interface for a repository named 'my_project'. The 'master' branch is selected in the top left dropdown menu. Below the branch name, a commit by Jonathan Kitt is shown with the message 'modification du fichier dans GitLab', authored 56 minutes ago. A table below lists the files in the repository:

Name	Last commit
 .gitignore	connexion RStudio/GitLab
 fichier_texte.txt	modification du fichier dans GitLab
 my_project.Rproj	connexion RStudio/GitLab



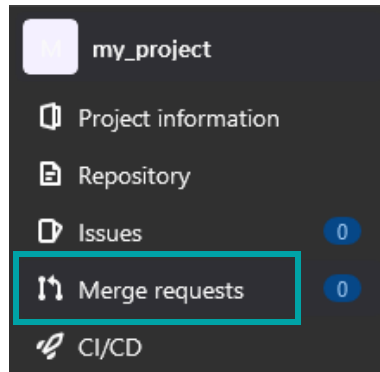
The screenshot shows the GitLab web interface for the same repository 'my_project', but now the 'test' branch is selected in the top left dropdown menu. A new commit by Jonathan Kitt is shown with the message 'travail sur branche test', authored 1 minute ago. The table of files is updated to reflect the changes on this branch:

Name	Last commit
 .gitignore	connexion RStudio/GitLab
 fichier_texte.txt	travail sur branche test
 my_project.Rproj	connexion RStudio/GitLab

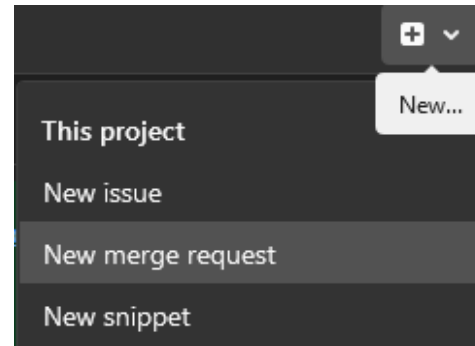


➤ Merge

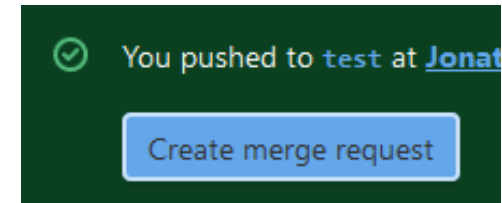
- *Create merge request*



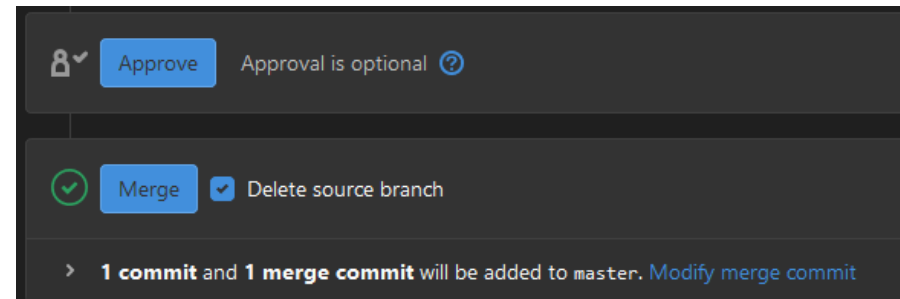
ou



ou



- *Approve merge request*



➤ Package *usethis*

<https://usethis.r-lib.org/>

- Installer et charger le package

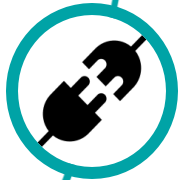
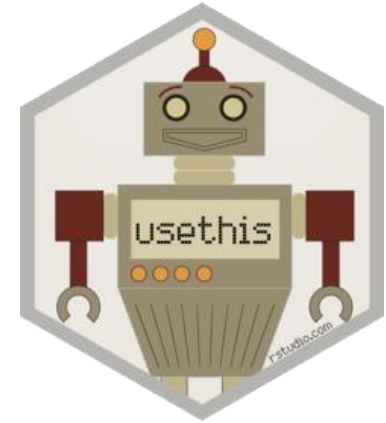
```
> install.packages("usethis")  
> library(usethis)
```

- Associer un projet à un répertoire existant

```
> usethis::create_project(".')
```

- Initier la connexion à git

```
> usethis::use_git()
```





IN CASE OF FIRE



Git Commit



Git Push



Git Out