Solution de versionning

ESGI - 3ème année IW 2020

Git : Les commandes de base

git init

Git : Les commandes de base

Créer un dépôt (working copy)

```
utilisateur@machine:/demo$ git init
Initialized empty Git repository in /demo/.git/

utilisateur@machine:/demo$ ls -la
total 12
drwxr-xr-x 3 utilisateur utilisateur 4096 Sep 25 14:35 .
drwxr-xr-x 4 utilisateur utilisateur 4096 Sep 25 14:35 ..
drwxr-xr-x 7 utilisateur utilisateur 4096 Sep 25 14:35 .git

utilisateur@machine:/demo$ ls -la .git/
total 40
drwxr-xr-x 7 utilisateur utilisateur 4096 Sep 25 14:35 .
drwxr-xr-x 3 utilisateur utilisateur 4096 Sep 25 14:35 ..
drwxr-xr-x 2 utilisateur utilisateur 4096 Sep 25 14:35 branches
-rw-r--r-- 1 utilisateur utilisateur 92 Sep 25 14:35 config
-rw-r--r-- 1 utilisateur utilisateur 73 Sep 25 14:35 description
-rw-r--r-- 1 utilisateur utilisateur 23 Sep 25 14:35 HEAD
drwxr-xr-x 2 utilisateur utilisateur 4096 Sep 25 14:35 hooks
drwxr-xr-x 2 utilisateur utilisateur 4096 Sep 25 14:35 info
drwxr-xr-x 4 utilisateur utilisateur 4096 Sep 25 14:35 objects
drwxr-xr-x 4 utilisateur utilisateur 4096 Sep 25 14:35 refs
```

Créer un dépôt (bare)

```
utilisateur@machine:/demo$ git init --bare
Initialized empty Git repository in /demo/

utilisateur@machine:/demo$ ls -la
total 40
drwxr-xr-x 7 utilisateur utilisateur 4096 Sep 25 14:40 .
drwxr-xr-x 4 utilisateur utilisateur 4096 Sep 25 14:40 ..
drwxr-xr-x 2 utilisateur utilisateur 4096 Sep 25 14:40 branches
-rw-r--r-- 1 utilisateur utilisateur 66 Sep 25 14:40 config
-rw-r--r-- 1 utilisateur utilisateur 73 Sep 25 14:40 description
-rw-r--r-- 1 utilisateur utilisateur 23 Sep 25 14:40 HEAD
drwxr-xr-x 2 utilisateur utilisateur 4096 Sep 25 14:40 hooks
drwxr-xr-x 2 utilisateur utilisateur 4096 Sep 25 14:40 info
drwxr-xr-x 4 utilisateur utilisateur 4096 Sep 25 14:40 objects
drwxr-xr-x 4 utilisateur utilisateur 4096 Sep 25 14:40 refs
```

Les différents états d’un fichier

Git : Les commandes de base

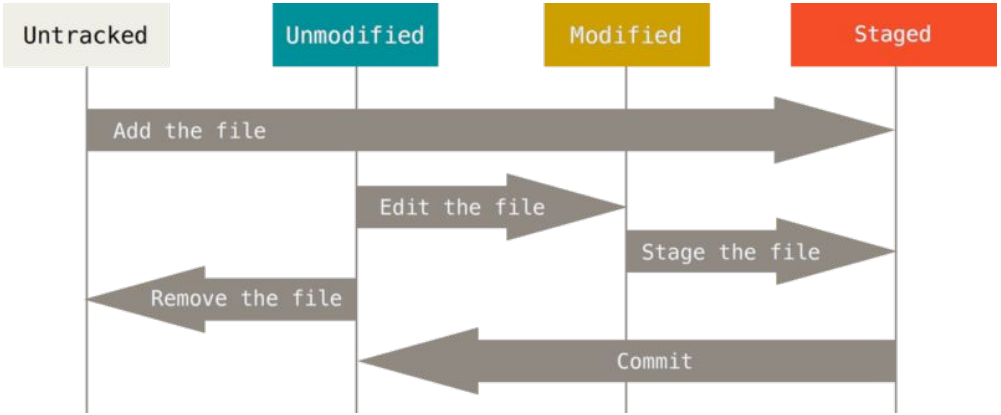
Connaître l’état d’un dépôt

```
utilisateur@machine :/demo$ git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
```

Le dépôt est dit “clean”, en opposition avec “dirty” (lorsque des fichiers traqués sont modifiés).

Statut d’un fichier

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



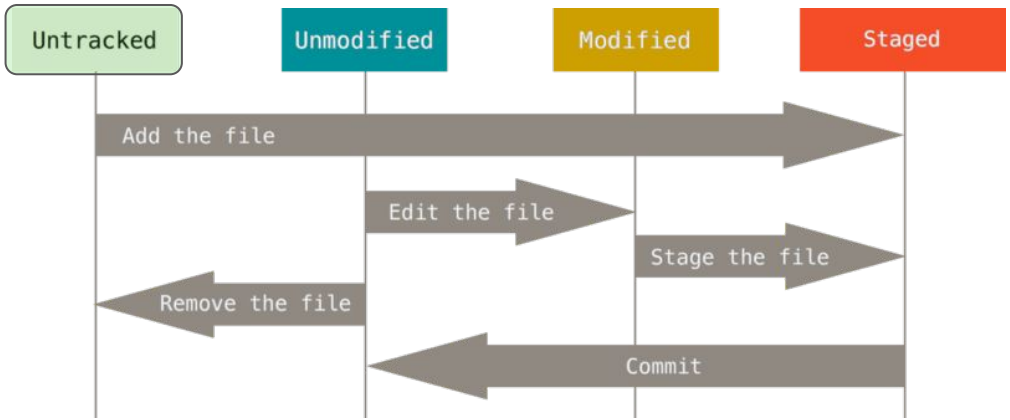
Fichier non traqué

```
utilisateur@machine:/demo$ touch README.md
utilisateur@machine:/demo$ git status
On branch master

No commits yet

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

        README.md

nothing added to commit but untracked files present (use "git add" to track)
```

Le dépôt est dit “dirty”, le fichier “README.md” n’est pas traqué.

Statut d’un fichier

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



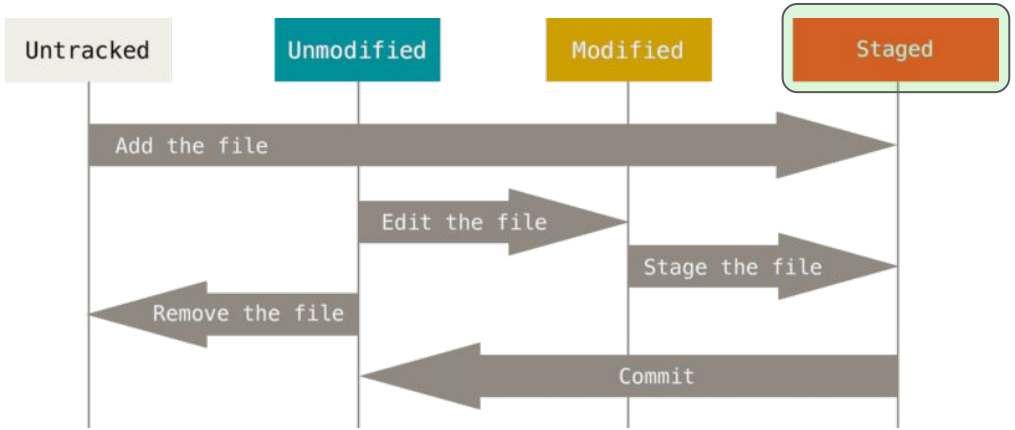
Fichier ajouté

```
ubuntu@machine:/demo$ git add README.md
ubuntu@machine:/demo$ git status
On branch master

No commits yet

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

       new file:   README.md
```

Le dépôt est dit “dirty”, le fichier “README.md” est ajouté au “staged”.

Statut d’un fichier

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



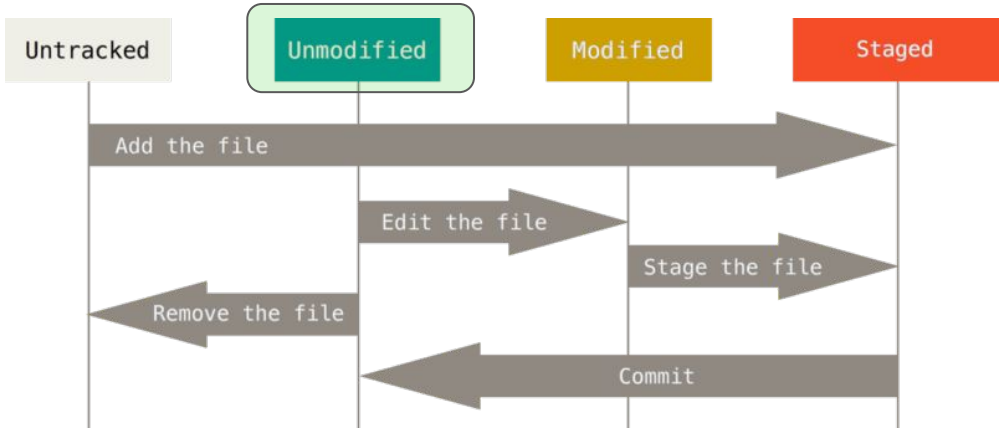
Fichier commité

```
utilisateur@machine:/demo$ git commit -m "Commit README"
[master (root-commit) 2e2ccd5] Commit README
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
utilisateur@machine:/demo$ git status
On branch master
nothing to commit, working tree clean
```

Le dépôt est dit “clean”, le fichier “README.md” commité, il n’y a pas de changements.

Statut d’un fichier

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



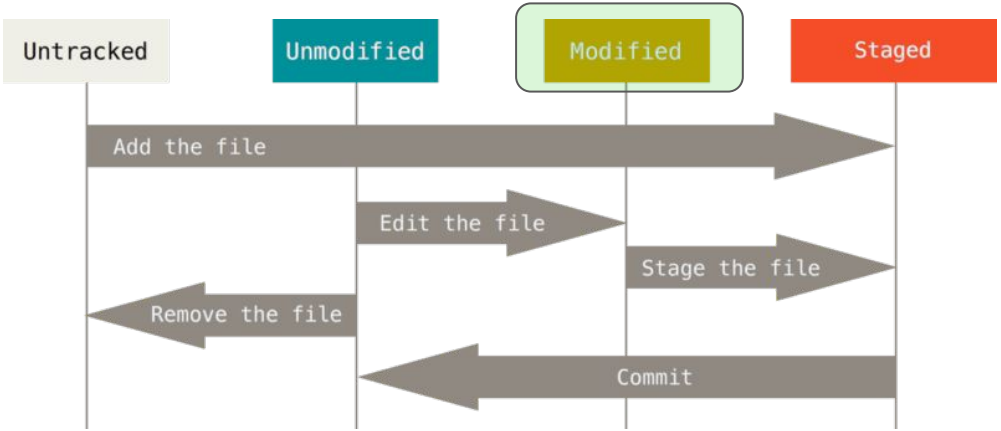
Fichier modifié

```
utilisateur@machine:/demo$ echo 'Demo' >> README.md; echo '====' >> README.md
utilisateur@machine:/demo$ git status
On branch master
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:   README.md

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

Le dépôt est dit “dirty”, le fichier “README.md” est modifié.

Statut d’un fichier

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



Ajouter un dossier

```
utilisateur@machine:/demo$ mkdir demo-folder
utilisateur@machine:/demo$ git status
On branch master
nothing to commit, working tree clean
utilisateur@machine:/demo$ touch demo-folder/.gitkeep
utilisateur@machine:/demo$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        demo-folder/

nothing added to commit but untracked files present (use "git add" to track)
```

Les “dossiers” n’existent pas, ce sont des chemins de fichier. Il faut ajouter un fichier (même vide ou/et caché) dans le dossier pour le versionner).

git commit

Git : Les commandes de base

Première exécution : paramètrage git

```
*** Please tell me who you are.

Run

  git config --global user.email "you@example.com"
  git config --global user.name "Your Name"

to set your account's default identity.
Omit --global to set the identity only in this repository.

fatal: unable to auto-detect email address (got 'root@9bf2a8919a5c.(none)')
```

Message de commit

Commit message avec -m pour message court :

```
git commit -m "Description de ce que le commit apporte"
```

Commit sans aucune option :

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#       new file:   README.md
#
```

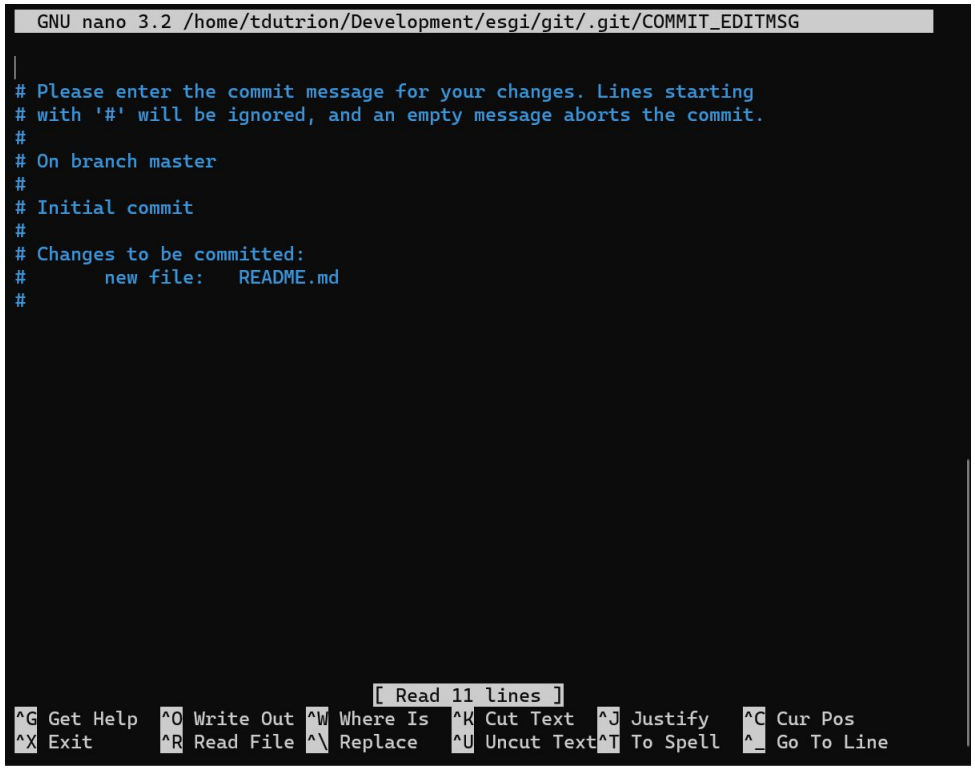
Configurer l’éditeur de message de commit

```
git config --global core.editor nano
```

ou

```
git config core.editor nano
```

Commit avec Nano



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

Contrairement aux plateformes de gestion de projet externe, l'historique git suivra forcément l'intégralité du projet, bien le documenter offre des garanties.

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 log

Git : Les commandes de base

Exemple de message de commit

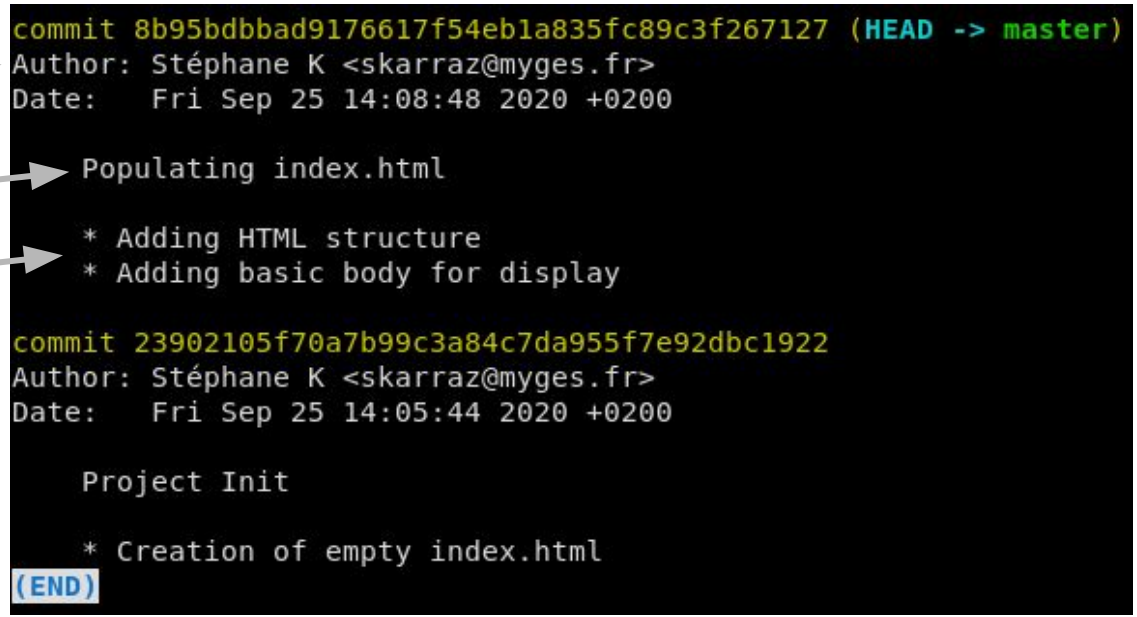
```
utilisateur@machine:~/Development/esgi/git$ git log
commit 3a33969b82661de34bdbea27d683d0d48d90af84 (HEAD -> master)
Author: Thomas Dutrion <thomas@engineor.com>
Date:   Fri Sep 25 14:15:23 2020 +0200

    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
```

Exemple de message de commit

```
utilisateur@machine:/demo$ git log --oneline
3a33969 (HEAD -> master) Summarize changes in around 50 characters or less
```

Référence dans l'historique

Sha1 entier



Auteur: nom et email

Date de commit

Titre

Texte explicatif de l'intention derrière le code

Affichage personnalisé

```
git log --graph \
        --abbrev-commit \
        --decorate \
        --format=format:'%C(bold blue)%h%C(reset) - %C(bold green)(%ar)%C(reset) %C(white)%s%C(reset) %C(dim
white)- %an%C(reset)%C(bold yellow)%d%C(reset)' \
        --all
```

git push / git remote

Git : Les commandes de base

Dépôt distant

Un dépôt distant est :

- Un dépôt git (working copy ou bare)
- Accessible via un chemin de système de fichier, une uri (http/https/ssh…)
- Identifié sur le dépôt local avec un nom (par exemple “origin”)

Un dépôt peut avoir zéro, un ou plusieurs dépôts distants avec le(s)quel(s) il peut se synchroniser.

Initialisation de l’espace de travail

```
utilisateur@machine:/demo$ mkdir -p {local,distant}

utilisateur@machine:/demo$ git -C local/ init
Initialized empty Git repository in /demo/local/.git/

utilisateur@machine:/demo$ git -C distant/ init --bare
Initialized empty Git repository in /demo/distant/

utilisateur@machine:/demo$ touch local/README.md

utilisateur@machine:/demo$ git -C local/ add README.md

utilisateur@machine:/demo$ git -C local/ commit -m "Add readme file"
[master (root-commit) b4c71b1] Add readme file
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md

utilisateur@machine:/demo$ git -C local/ log --oneline
b4c71b1 (HEAD -> master) Add readme file

utilisateur@machine:/demo$ git -C distant/ log --oneline
fatal: your current branch 'master' does not have any commits yet
```

Ajout de dépôt distant et synchronisation

```
utilisateur@machine:/demo$ git -C local/ remote -v
origin  ../distant (fetch)
origin  ../distant (push)

utilisateur@machine:/demo$ git -C local/ push origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 873 bytes | 873.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To ../distant
* [new branch]      master -> master

utilisateur@machine:/demo$ git -C local/ log --oneline
b4c71b1 (HEAD -> master, origin/master) Add readme file

utilisateur@machine:/demo$ git -C distant/ log --oneline
b4c71b1 (HEAD -> master) Add readme file
```

git clone

Git : Les commandes de base

Git clone

Commande pour cloner un projet distant:

```
git clone URI [nom du dossier local]
```

Exemples :

```
git clone ../un-chemin-relatif/depot mon-dossier-de-travail-local

git clone /un-chemin-absolu/depot mon-dossier-de-travail-local

git clone https://mon-serveur-git/utilisateur/depot.git mon-dossier-de-travail-local

git clone git@machine:utilisateur/depot.git mon-dossier-de-travail-local
```

Mise en application

Git : Les commandes de base

Mise en application

- Créer deux dossier de repository (un workdir et un bare)
- Ajouter un readme dans le workdir
- Enregister (commit) le readme (`* ea74851 (HEAD -> master) Add initial documentation`)
- Ajouter un remote (le bare)
- Pousser (push) sur le remote
- Git log sur le bare pour voir le changement
- Ajouter une licence (`curl -o LICENSE.txt https://www.gnu.org/licenses/gpl-3.0.txt`)
- Enregister (commit) la licence
- Git log sur le remote : pas de commit licence (`* ea74851 (HEAD -> master) Add initial documentation`)
- Pousser (push) sur le workdir -> remote puis log dans le bare, on voit le changement