

GIT

Installation

[Site officiel de git](#)

Installer git sur Windows

Téléchargez *Git* sur le site officiel et installez le sur votre poste de travail. Gardez les paramètres par défauts pour l'installation.

Pour savoir si *Git* a été correctement installé :

- ouvrir l' **invite de commandes**
- taper `git help`
- si vous voyez apparaître la liste des commandes *Git*, c'est que *Git* est bien installé sur votre machine

Installer git sur Mac

Téléchargez *Git* sur le site officiel et installez le sur votre poste de travail. Gardez les paramètres par défauts pour l'installation.

Pour savoir si *Git* a été correctement installé

- ouvrir le **terminal**
- taper `git help`
- si vous voyez apparaître la liste des commandes *Git*, c'est que *Git* est bien installé sur votre machine

Configuration de *Git*

Il faut commencer par renseigner votre nom et votre email, ceux-ci apparaîtront dans toutes les validations de *Git* et permettront d'identifier ce que vous avez fait :

```
git config --global user.name "Mon Nom"
git config --global user.email "votreemail@votreemail.com"
```

⚠ Si vous ne configurez pas le nom et l'email, vous ne pourrez pas interagir avec GitHub.

Il est recommandé d'activer les couleurs afin d'améliorer la lisibilité, passez ces quatre lignes dans *Git Bash* :

```
git config --global color.ui true
git config --global color.diff auto
git config --global color.status auto
git config --global color.branch auto
```

Afin de vérifier que vos paramètres soient bien ajoutés à *Git*, vous pouvez utiliser la commande suivante pour lister les configurations :

```
git config --list
```

Utilisation de base de *Git*

Initialiser un dossier avec *Git*

Dans un premier temps, il faut nous positionner sur le dossier qui accueillera notre dépôt *Git*. Deux méthodes pour cela :

Méthode 1 :

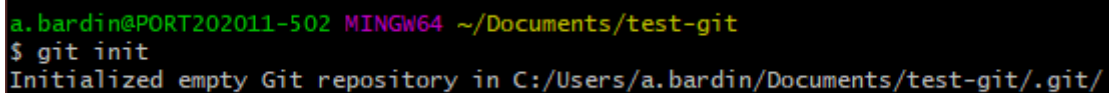
- Ouvrir *Git Bash*
- Se déplacer grâce aux lignes de commandes jusqu'au dossier qui nous intéresse ([Ligne de commande Windows](#) - [Ligne de commande Mac](#))

Méthode 2 :

- Grâce à l'explorateur de fichiers, se placer dans le répertoire du dossier qui nous intéresse
- Faire un clic droit sur le dossier
- Dans la liste des commandes, sélectionnez "***Git Bash Here***"

Maintenant que vous êtes positionnés sur le dossier que vous allez utiliser comme dépôt *Git*, il faut l'initialiser. Pour cela taper la ligne suivante :

```
git init
```



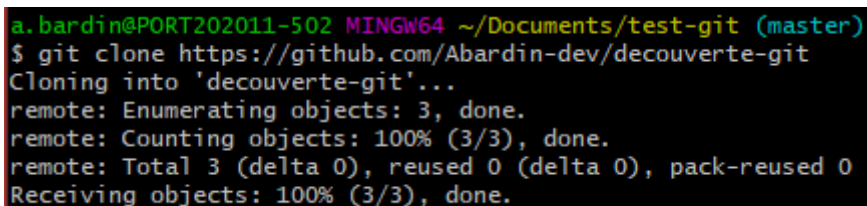
```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git
$ git init
Initialized empty Git repository in C:/Users/a.bardin/Documents/test-git/.git/
```

Un dossier `.git` (caché) a été ajouté dans notre dossier, cela signifie que notre dossier est bien initialisé avec *Git*.

Récupérer un dépôt distant

Si vous souhaitez récupérer un projet déjà hébergé sur *GitHub* pour collaborer dessus, il faut utiliser la commande suivante :

```
git clone url_du_dépôt_Git_sur_Github
```



```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git (master)
$ git clone https://github.com/Abardin-dev/decouverte-git
Cloning into 'decouverte-git'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Vous avez maintenant une copie du dépôt distant en local sur votre poste de travail. Vous pouvez à présent commencer à modifier les fichiers que vous avez récupérés (cela n'impactera pas ce qui se trouve sur le dépôt pour l'instant puisque vous avez effectué une copie de ce qui s'y trouvait).

Vérifier le statut

Git possède une commande pour obtenir des informations sur l'état de notre dépôt local :

```
git status
```

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

```
$ git status
On branch master
Your branch is up to date with 'origin/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:   programme-reel.md

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

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
```

Votre premier *commit*

Une fois vos modifications effectuées, il va vous falloir ajouter votre fichier modifié à la gestion des versions (on dit que l'on envoie notre fichier dans l'*Index* - une "zone de transit" qui contiendra les modifications à envoyer). Il vous faut avant vous placer sur le dossier du dépôt avec *Git Bash*.

Ensuite, on utilisera la commande :

```
git add le_nom_de_votre_fichier.extension
```

Exemple : `git add test.txt`

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git add README.md
```

Il est également possible, si l'on doit envoyer de nombreux fichiers, d'utiliser la commande :

`git add .` ou `git add --all` (ces deux commandes produisent le même résultat, elles vont ajouter à l'*Index* tous les fichiers qui ont été modifiés).

Valider les modifications

Une fois vos modifications effectuées et vos fichiers ajoutés à l'*Index*, il va vous falloir valider ces modifications, c'est-à-dire les ajouter au dépôt local (également appelé *HEAD*). La commande pour cela est `git commit`.

Mais cela n'est pas suffisant, il faut passer des paramètres à notre commande pour pouvoir l'utiliser.

```
git commit -a -m "le nom de votre commit"
```

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git commit -m "initial commit"
[main 1f82103] initial commit
1 file changed, 2 insertions(+)
```

Le nom que l'on donne à notre commit représentera ce qui a été modifié dans nos fichiers (ajout de telle fonctionnalité, corrections de tel bug...).

Voilà, vos fichiers modifiés sont maintenant présents dans le *HEAD* (pour une approche imagée, nous avons enregistré les modifications de nos fichiers dans *Git*).

[i] Pour pouvoir commit sur un dépôt dont vous n'êtes pas le propriétaire, il faut au préalable que celui-ci vous ait invité en tant que collaborateur du dépôt.

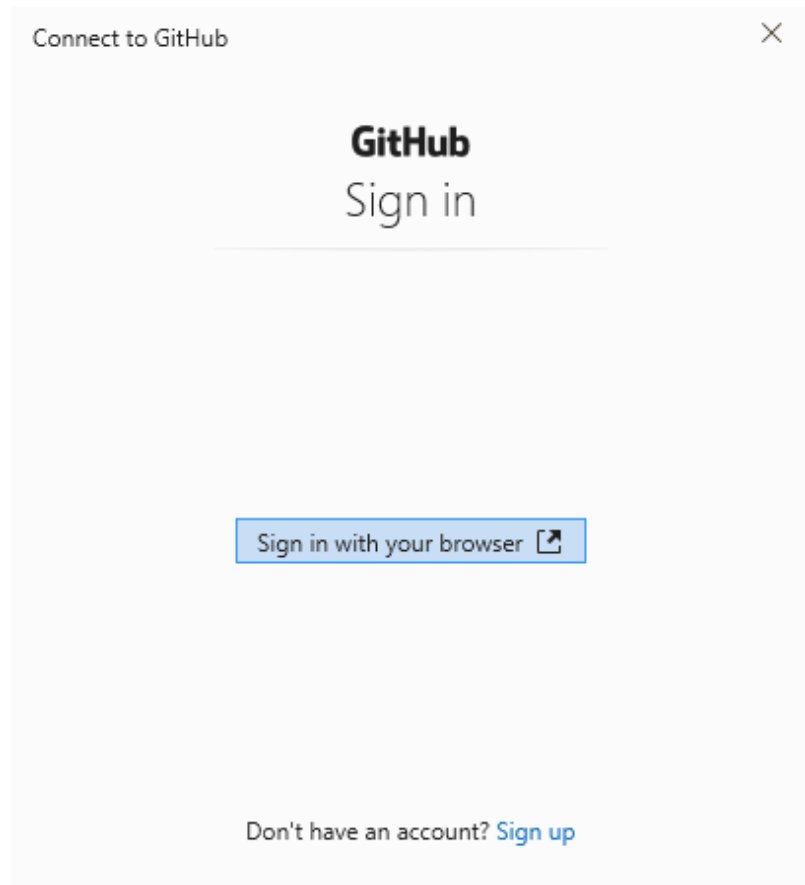
Envoyer ses modifications sur le dépôt distant

Pour envoyer ses modifications sur le dépôt distant (*GitHub*), il faut utiliser la commande :

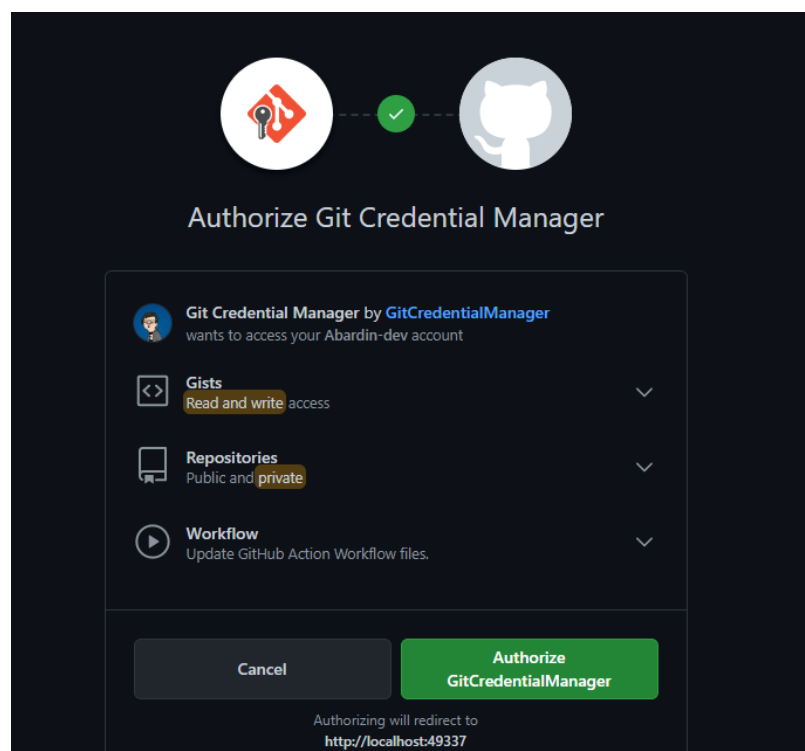
```
git push
```

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 269 bytes | 269.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Abardin-dev/decouverte-git
5136e30..1f82103  main -> main
```



Lors de votre tout premier `git push`, *Git* vous demandera de vous authentifier.



Si vous êtes déjà connectés sur votre navigateur, *GitHub* vous demandera si vous acceptez de lier votre compte *GitHub* à *Git*.



Sinon, vous devrez vous identifier à l'aide de vos identifiants *GitHub*.

Sign in to GitHub
to continue to Git Credential
Manager

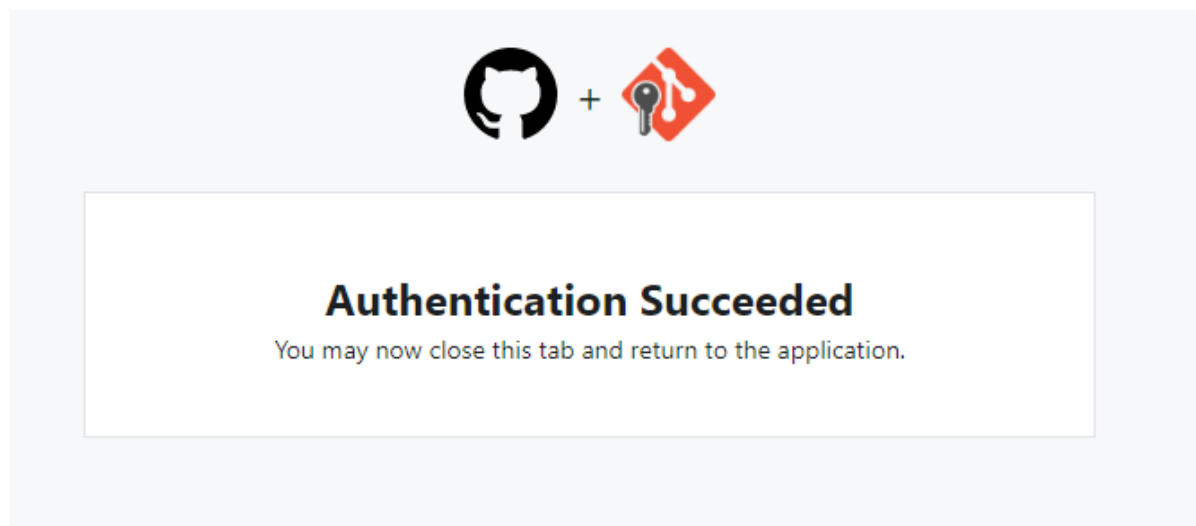
Username or email address

Password [Forgot password?](#)

[Sign in](#)

New to GitHub? [Create an account.](#)

Un message de confirmation vous confirmera la liaison.






Dans les paramètres de votre compte *GitHub*, vous retrouverez dans l'onglet "Applications" tous les systèmes auxquels *GitHub* est lié. Ici, on retrouve la liaison avec notre *Git*.

Applications

Installed GitHub Apps Authorized GitHub Apps **Authorized OAuth Apps**

You have granted **4 applications** access to your account. Sort ▾ Revoke all

-  **CodePen** ...
Last used within the last 3 months · Owned by [codepen](#)
-  **diagrams.net** ...
Last used within the last 3 months · Owned by [jgraph](#)
-  **Git Credential Manager** ...
Last used within the last week · Owned by [GitCredentialManager](#)

Vous pouvez aller voir sur l'interface de *GitHub*, dans votre *repository*, vos modifications sont à présent visible.

Travail collaboratif

La méthodologie `add - commit - push` fonctionne pour une utilisation personnelle de *Git* mais qu'en est-il lorsque l'on souhaite collaborer sur un projet ?

Pour permettre cela, il existe la commande suivante :

```
git pull
```

Elle va nous permettre de mettre à jour notre espace de travail à partir du dépôt distant (On va aller récupérer les modifications qu'il y a pu avoir sur le dépôt distant par les autres collaborateurs).

La commande nous indiquera si notre dépôt local est à jour par rapport au dépôt distant (*Already up to date*)

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git pull
Already up to date.
```

Ou si les deux versions divergent, dans ce cas, elle indiquera le nombre de fichiers mis à jour ainsi que leur nom.

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 646 bytes | 80.00 KiB/s, done.
From https://github.com/Abardin-dev/decouverte-git
   1f82103..5c15c9c  main      -> origin/main
Updating 1f82103..5c15c9c
Fast-forward
 README.md | 2 --
 1 file changed, 2 deletions(-)
```

⚠ L'utilisation de `git pull` devra toujours intervenir avant un `git push` sous peine de voir la version du dépôt distant écrasé par notre version locale, c'est-à-dire supprimer le travail des autres collaborateurs

Résolution des conflits

Dans la plupart des cas, lors d'un `git pull`, *Git* se chargera automatiquement de fusionner nos modifications locales avec les modifications qui proviennent du dépôt distant.

Dans ce cas, une interface va s'ouvrir.

Git vous indique les fichiers concernés par le problème avec la ligne *CONFLICT* et passe en mode *MERGING*.

Ouvrez les fichiers en erreur à l'aide de votre éditeur de code favori. Vous devriez voir dans votre code les éléments suivants :

```
<<<<<<< HEAD
Hello World
=====

Test merge commit
e5de6b358ac1cf14f99284f1cf04244c04df2529
```

Représenté par `<<<<<<< HEAD` `=====` `>>>>>>>`. Ce qui se trouve au-dessus des `=====` représente vos modifications effectuées en local tandis que ce qui se trouve en dessous représente les modifications sur le dépôt distant. Chaque partie concerne la même ligne initiale. À vous de choisir si vous souhaitez garder vos modifications, celles du dépôt distant où bien encore garder les deux. (Vous devez retirer les 3 lignes de délimitations `<<<<<<< HEAD`, `=====` et `>>>>>>>`).

Une fois les fichiers en erreur corrigés. Il faut ajouter ces modifications à l'*Index*. Comme dans le cycle standard, vous devrez commencer par un `git add` :

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main|MERGING)
$ git add --all
```

Puis un `git commit` pour valider vos modifications. *Git* vous indiquera que la fusion entre votre dépôt local et le dépôt distant c'est bien exécuté.

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main|MERGING)
$ git commit -m "merge"
[main b91ee8a] merge
```

Si vous essayez de nouveau un `git pull`, *Git* vous indiquera que votre version est déjà à jour. Vous pouvez passer à la dernière étape : `git push` pour publier vos modifications.

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git pull
Already up to date.

a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git push
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 608 bytes | 608.00 KiB/s, done.
Total 6 (delta 0), reused 2 (delta 0), pack-reused 0
To https://github.com/Abardin-dev/decouverte-git
e5de6b3..b91ee8a main -> main
```

Historique des *commits*

La commande suivante permet d'obtenir la liste de l'ensemble des *commits* réalisé sur le dépôt. On retrouvera pour chaque commit le propriétaire, la date et le message de commit trier du plus récent au plus ancien.

```
git log
```

```
a.bardin@PORT202011-502 MINGW64 ~/Documents/test-git/decouverte-git (main)
$ git log
commit b91ee8a63965ee8daa1cf3a47e9f95a9e7b1eb41 (HEAD -> main, origin/main, origin/HEAD)
Merge: 7f0a009 e5de6b3
Author: Abardin-dev <a.bardin@codeur.online>
Date: Tue Feb 2 12:08:30 2021 +0100

    merge

commit e5de6b358ac1cf14f99284f1cf04244c04df2529
Author: BARDIN-TONNEL Alexandre <a.bardin@codeur.online>
Date: Tue Feb 2 12:05:22 2021 +0100

    Update README.md

commit 7f0a00991006ae626e5c5fe0e1235e8e7fbea0cc
Author: Abardin-dev <a.bardin@codeur.online>
Date: Tue Feb 2 12:05:00 2021 +0100

    initial commit

commit 5c15c9c88152867d8fa0aacd20ef93a16e278640
Author: BARDIN-TONNEL Alexandre <a.bardin@codeur.online>
Date: Tue Feb 2 12:03:51 2021 +0100
```

Initier un dépôt distant depuis un dépôt local

Si l'on souhaite partir d'un projet sur notre poste de travail qui ne posséderait donc pas de dépôt distant, il faudra opérer quelques modifications.

Pour commencer, il vous faudra créer un *repository* sur *GitHub* qui servira à accueillir votre dépôt local.

Ensuite, procéder comme vous avez l'habitude de le faire, `git init` sur votre dossier, `git add *`, `git commit -m "message"` afin d'ajouter les fichiers au `HEAD` (dépôt local).

C'est maintenant que les changements vont s'opérer. Puisque vous n'êtes pas parti d'un dépôt distant, il va falloir lier votre dépôt local au dépôt distant précédemment créer. Pour cela, il faudra synchroniser nos dépôts avec `git remote` :

```
git remote add origin https://github.com/Abardin-dev/test.git
```

Puis on ajoutera des paramètres à notre `git push` :

```
git push -u origin master
```

Voilà notre dépôt local est lié au dépôt distant, vous pouvez reprendre le cycle habituel d'utilisation de *Git*.

Autre commande utiles

```
git diff
```

Cette commande retourne toutes les différences entre la version de votre espace de travail et le dépôt distant. Taper `q` pour quitter la liste des différences.

Pour résumer

- `git init` pour initialiser
- `git clone` pour récupérer un dépôt distant
- `git add`
- `git commit`
- `git pull` pour récupérer les modifications faites sur le dépôt distant
- `git push` pour envoyer ses modifications sur le dépôt distant
- `git log` pour avoir un historique des *commits*

Si vous respectez cet ordre (`git pull` - `git add` - `git commit` - `git pull` - `git push`) vous ne devriez rencontrer aucun problème dans l'utilisation de *Git*.

Toutes les commandes vues précédemment possèdent des paramètres additionnels. Je vous invite à regarder la documentation pour en savoir plus sur les possibilités d'utilisation.

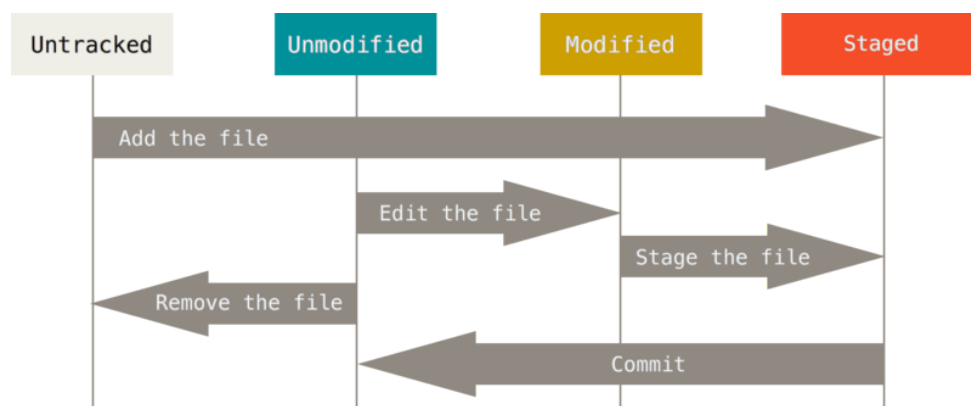
Vous savez maintenant utiliser *Git* de façon basique. Cette utilisation fonctionne bien pour gérer vos projets individuels (les envoyer sur votre *GitHub*) ou pour de petits projets en groupes dans le cadre de la formation. Cependant, dans un contexte d'entreprise, cette utilisation ne sera pas suffisante. Cela fera l'objet d'un prochain cours sur l'utilisation poussée de *Git* en suivant une méthodologie spécifique.

Liens utiles

[La documentation officiel de Git](#)

[Représentation visuel du fonctionnement de Git avec rappel des commandes possible à chaque étapes](#)

Les différents états possible pour un fichier :



[Tutoriel vidéo sur les bases de Git](#)

