

Formation Git

III - Les commits

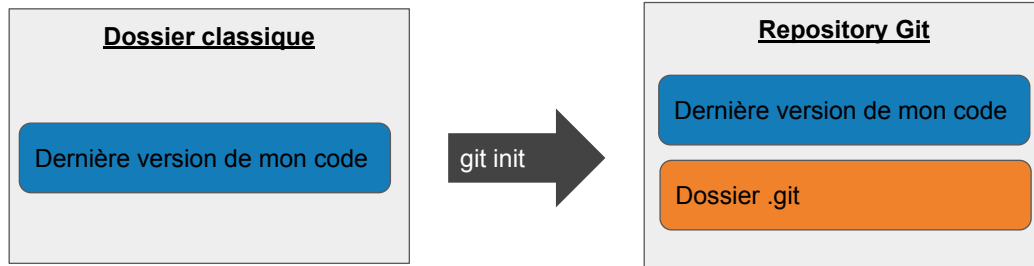


Arnaud MERCIER
arnaud.mercier@hexotech.fr



Nous allons voir la base des historiques GIT: les commits

Dépôt Git



```
git init [nom_dossier] // Créer un dossier et initialiser un dépôt dans ce dernier
```



le dossier .git contient l'historique, ne pas le supprimer !

Un dépôt Git c'est simplement un dossier du nom de .git qui contient l'ensemble des informations permettant la gestion par Git de votre code source.

Pour créer un dépôt Git en local, il y a deux possibilités:

- Créer un dépôt vide: `git init [nom_dépôt]`
- Créer un dépôt dans un dossier existant: `git init`

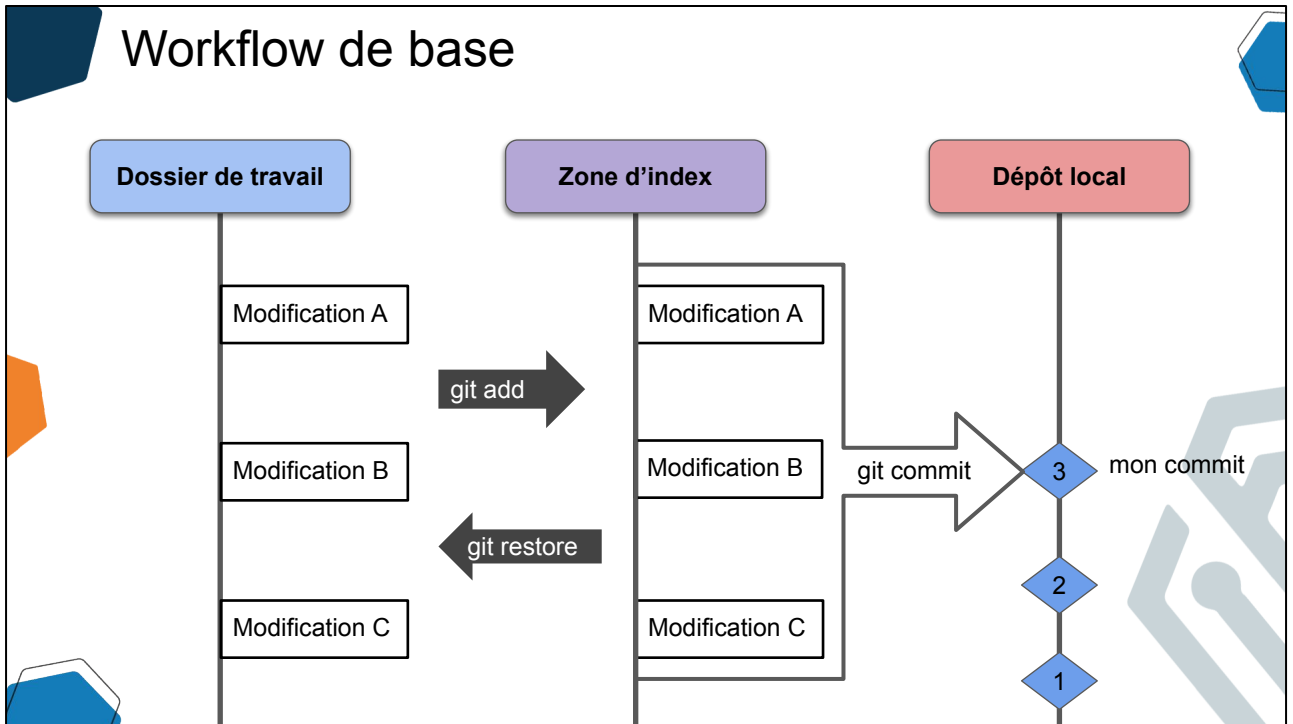
INFO: le dossier .git est un dossier caché

ATTENTION: ne pas supprimer le dossier .git au risque de perdre toute son historique

EXEMPLE:

1. `git init mon_depot` -> voir ce qu'il y a dans le dossier `mon_depot` et son `.git`
2. Créer un dossier GOT avec le contenu suivant:
 - a. `todo.txt`
 - Walder Frey
 - Ser Ilyn Payne
 - b. `done.txt`
 - Ser Meryn Trant
 - Polliver et Chiswyck
 - Le Titilleur
 - Ser Amory Lorch
3. Dans le dossier GOT, créer un dépôt: `git init`
4. Changer le `user.name` local en "Arya Stark"

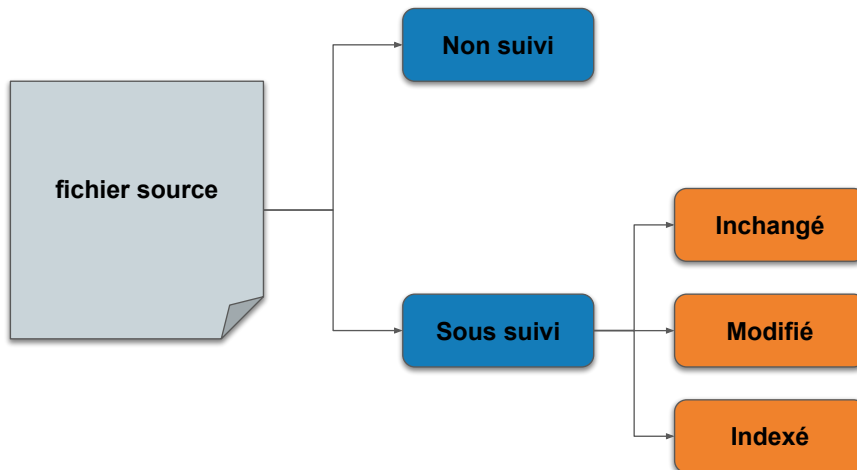
Workflow de base



Il y a 3 zones dans le workflow d'enregistrement d'une nouvelle version dans le dépôt Git.

- 1- **Dossier de travail**, qui correspond à vos modifications en cours
- 2- **Zone d'index** qui correspond aux modifications que vous souhaitez enregistrer. Il faut voir cette zone comme un sac, dans lequel on met ce que l'on souhaite enregistrer.
- 3- **Dépôt** qui correspond à l'historique de votre projet. On prend tout ce qui se trouve dans la zone d'index et on l'enregistre dans l'historique sous forme d'un commit.

Etat des fichiers sous Git



Les fichiers peuvent avoir plusieurs état d'un point de vue de Git:

1- **non suivi**: cela correspond a des fichiers dans votre espace de travail qui ne sont pas gérés par git. soit car vous ne le souhaitez pas, soit car vous ne les avez jamais enregistrés dans votre dépôt.

2- une fois que votre fichier a déjà été enregistré une fois dans votre dépôt, il sera suivi par Git:

a- **inchangé**, pas de modifications en cours sur ce fichier

b- **modifié**, des modifications sont présents sur ce fichier dans votre workspace

c- **indexé**, des modifications sont présents sur ce fichier dans votre zone d'index.

Etat du dépôt

```
MINGW64/f/codeur-pro/Formations/Git/workspace/cours_git/mon_site_web
$ git status
On branch master
Your branch is ahead of 'origin/master' by 1 commit.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   hello.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)

        deleted:    .gitignore
        modified:   style.css

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

        test.txt
        tmp/
```

Suivi et Indexé

Suivi et modifié

Non suivi

git status // Etat courant du dépôt

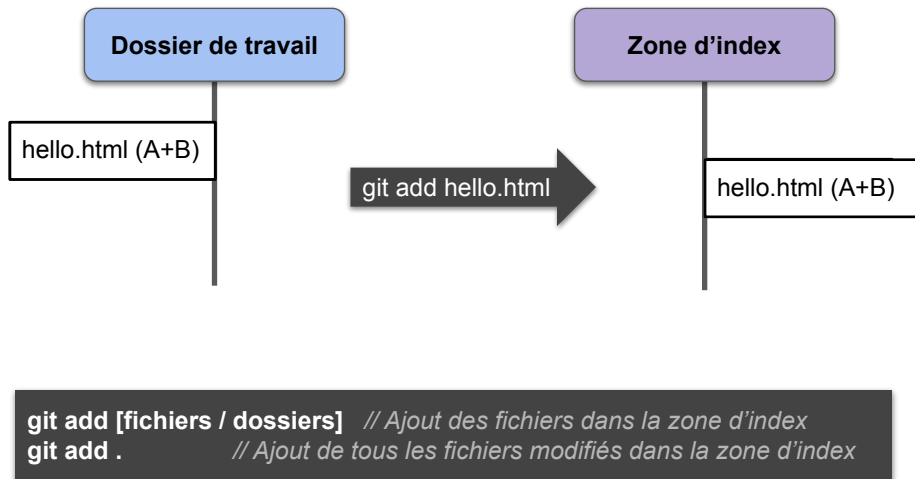
La commande Git status vous donne l'état courant des différents fichiers.

Astuce: Git est très verbeux de base et vous donne alors des indications sur les commandes à utiliser en fonction de l'état de votre dépôt. Prenez donc toujours le temps de lire ce qu'il vous écrit.

EXEMPLE (GOT):

1. git status

Indexer les modifications



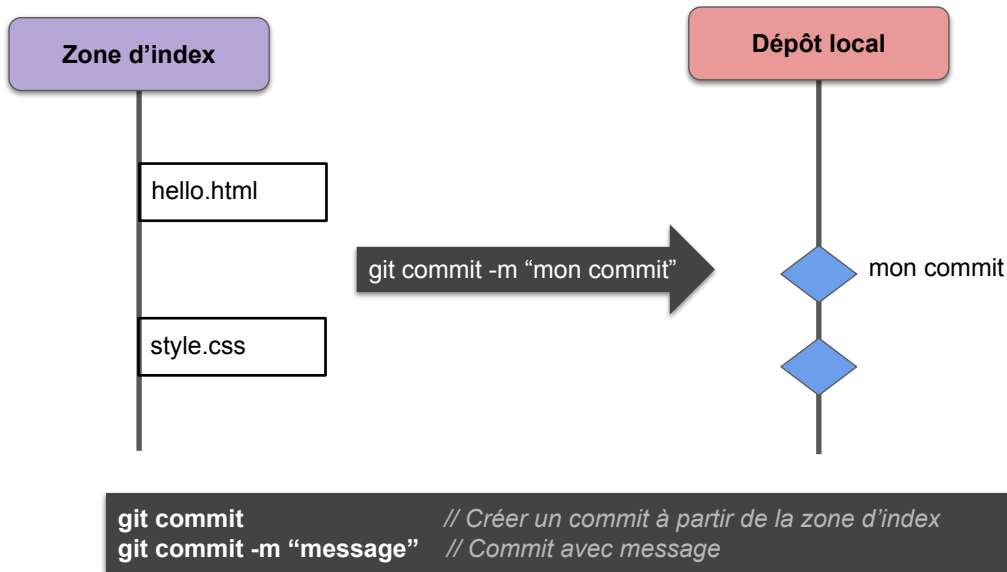
Pour ajouter ses modifications dans la zone d'index, il faut utiliser la commande git add:

- soit avec le nom des fichiers concernés
- soit avec . pour prendre tous les fichiers modifiés

EXEMPLE (GOT):

1. **git add .**
2. **git status**

Créer un commit depuis l'index



Vous avez indexer ce que vous souhaitez enregistrer. vous avez votre liste de fichiers et leurs modifications dans la zone d'index. Il ne vous reste plus qu'à réaliser un commit via la commande `git commit`.

Sans l'option `-m`, qui permet d'ajouter un message, un éditeur de texte s'ouvrira pour vous demander de rentrer un message.

Ce message est là pour indiquer ce qu'apporte votre commit au projet, quelle est la raison de la modification. Cela permet alors une relecture et compréhension de l'historique plus simple. Il est même possible d'indiquer des numéros de faits techniques pour faire un lien avec votre gestionnaire d'anomalies.

EXEMPLE (GOT):

1. `git commit -m "initial commit"`

Format du message de commit

Objet : Court résumé informatif d'une ligne (de préférence moins de 50 caractères)

Détails: Texte explicatif plus détaillé si nécessaire. Sur une ou plusieurs lignes

Référence: Index d'une ou plusieurs références, comme par exemple un ID de ticket

Format

[FEAT] Add drivers for Laser

- Add drivers for Laser management
- Create BIT for the Laser
- Improved linting
- Add doc for driver specifications

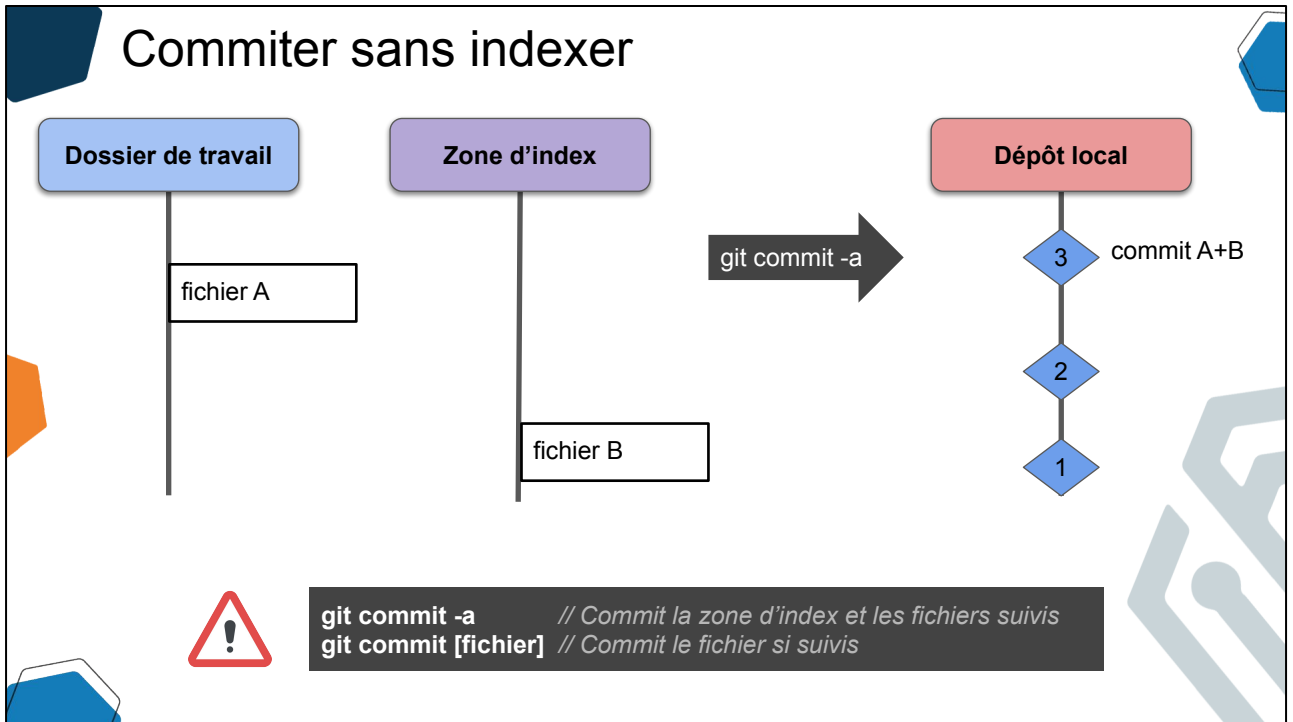
Refs #1234

Exemple

Il n'y a pas de vraie règle pour les messages de commit. En revanche il y a des conventions largement partagées:

- **Objet :** Court résumé informatif d'une ligne (de préférence moins de 50 caractères)
- **Détails:** Texte explicatif plus détaillé si nécessaire. Sur une ou plusieurs lignes
- **Référence:** Index d'une ou plusieurs références, comme par exemple un ID de ticket

Committer sans indexer



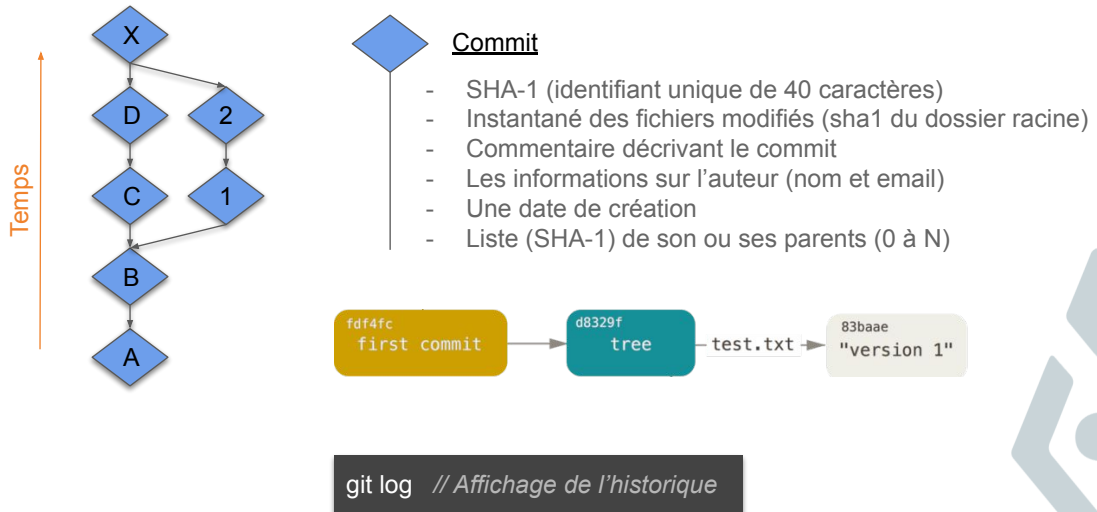
Il est possible de faire un commit sans passer par la zone d'index, mais attention, cette action est risquée, car vous pouvez enregistrer dans votre commit des modifications non souhaitées. La zone d'index est justement une zone tampon qui permet de vérifier le future contenu du commit.

Lors de l'utilisation de l'option -a, l'ensemble des fichiers suivis et modifiés qui sont indexés ou non, seront pris en compte dans le commit si suivis)

EXEMPLE (GOT):

1. `git commit -am"ajout Melisandre"`

Commits et historique



Un commit est constitué d'un ensemble d'informations:

- le sha1: identifiant unique de 40 caractères qui est également le checksum du commit
- l'instantané de la zone d'index au moment du commit
- Commentaire décrivant le commit
- Les informations sur l'auteur (nom et email)
- Une date de création
- Liste (SHA-1) de son ou ses parents

EXEMPLE (GOT):

1. `git log`

Désindexer les modifications

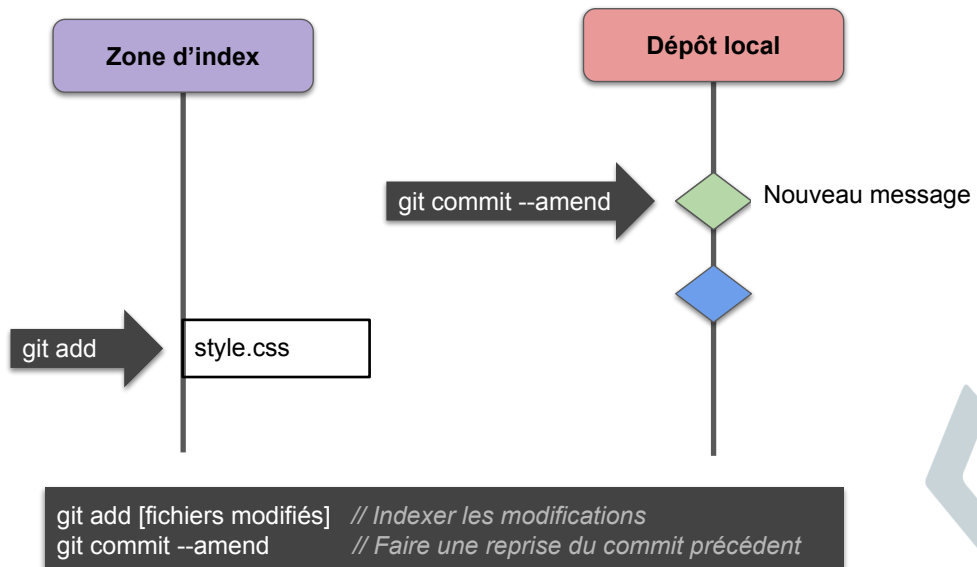


Pour désindexer des modifications, il faut utiliser la commande “git reset HEAD” ou git restore --staged”. Les modifications sont toujours dans l’espace de travail mais ne sont plus dans la zone d’index

EXEMPLE (GOT):

1. Ajouter “Melisandre” dans todo.txt
2. `git add .`
3. `git status`
4. `git restore --staged .`
5. `git status`

Modifier le dernier commit d'une branche



Vous souhaitez revoir votre dernier commit pour y ajouter des modifications oubliés ou simplement modifier le message de commit? et bien c'est possible avec la commande "git commit --amend"

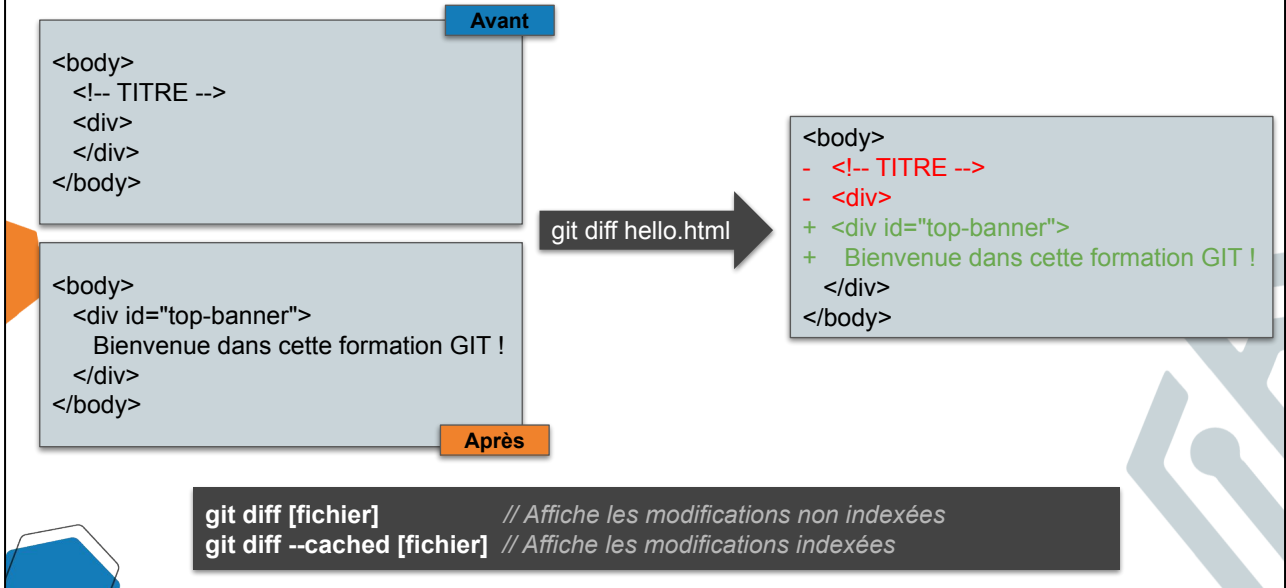
Pour ajouter des modification, utilisez "git add". une fois la zone d'index prête, utilisez la commande "git commit --amend". si vous souhaitez uniquement modifier le message de commit, taper simplement la commande sans modifier la zone d'index. Git vas alors prendre la zone d'index et l'ajouter aux autres modification comprises dans le commit que l'on corrige.

si vous n'indiquez pas de nouveau message, git utilisera celui du commit précédent.

EXEMPLE (GOT):

1. modifier "Melisandre" dans **todo.txt** en "Melisandre (pretresse rouge)"
2. **git add**
3. **git commit -amend**

Voir les modifications



la commande `git diff` permet de voir les différences entre deux versions d'un fichier. La commande est utilisable pour les fichiers dans la zone d'index ou l'espace de travail

les lignes qui commencent par - représentent une suppression
les lignes qui commencent par + représentent un ajout
les lignes sans + ou - sont inchangées

Remarque: `git` fonctionne en delta de lignes. si vous modifiez une partie de ligne uniquement, `git` vas vous afficher deux lignes dans le diff:

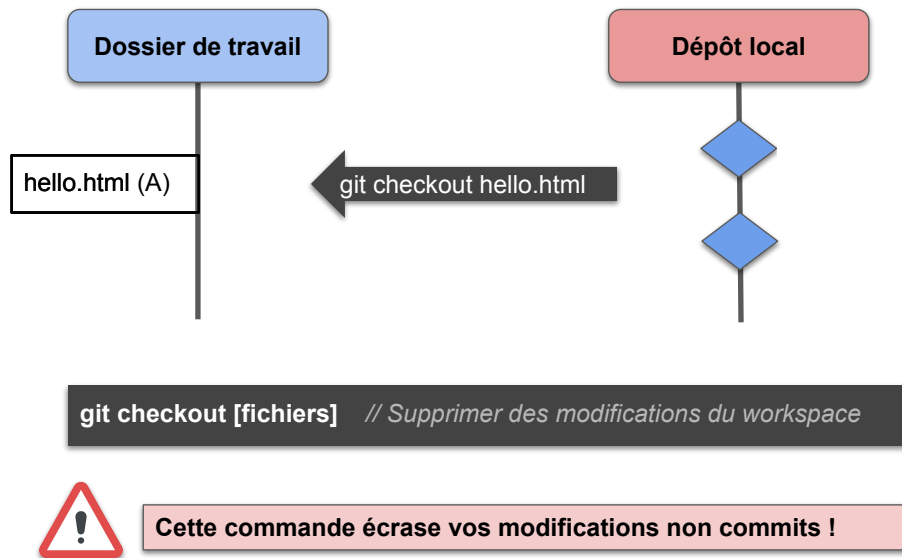
- une suppression (ancienne ligne)
- un ajout (nouvelle ligne)

EXEMPLE (GOT):

1. **Ajouter dans le fichier `todo.txt`**
 - Cersei Lannister
 - Joffrey Lannister
 - Tywin Lannister
2. **`git status`**
3. **`git diff`**
4. **`git add .`**
5. **`git status`**
6. **`git diff`**
7. **`git diff --cached`**
8. **`git restore --cached .`**

1. **git status**
2. **git add .**
3. **git commit -> "Ajout de la famille lannister"**
4. **git status**

Annuler les modifications du Workspace



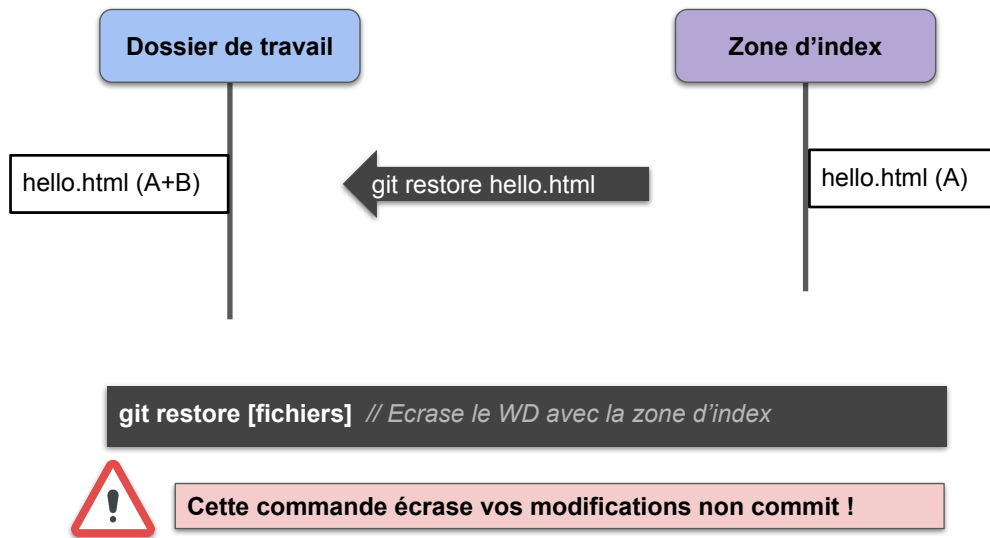
Si vous avez des modifications dans votre espace de travail et souhaitez les annuler, vous pouvez utiliser la commande “git checkout”

ATTENTION: les modifications seront définitivement perdus. Avec Gi il est possible de récupérer facilement des modifications à l'exception de quelques commandes comme par exemple “git checkout”

EXEMPLE (GOT):

1. Ajouter dans le fichier todo.txt
- Sansa stark
2. `git status; git diff`
3. `git checkout todo.txt`
4. `git status; git diff`

Annuler les modifications du Workspace



Contrairement au “git checkout”, le git restore écrase l’espace de travail non pas avec le contenu du dernier commit mais de la zone d’index.

Dans le cas où aucune modification d’un fichier n’est indexé, alors git checkout et git restore auront le même résultat

Ignorer des fichiers avec **.gitignore**

```
# pas de fichier .a
*.a

# mais suivre lib.a malgré la règle précédente
!lib.a

# ignorer uniquement le fichier TODO à la racine du projet
/TODO

# ignorer tous les fichiers dans le répertoire build
build/

# ignorer doc/notes.txt, mais pas doc/server/arch.txt
doc/*.txt

# ignorer tous les fichiers .txt sous le répertoire doc/
doc/**/*.txt
```

.gitignore



Le fichier .gitignore est un fichier caché

Le fichier .gitignore doit être ajouté à votre gestion de version

Dans votre espace de travail, vous pouvez avoir du code source mais aussi d'autres fichiers comme par exemple des binaires générés après compilation ou encore des fichiers permettant de tester votre code (ex fiches clients pour un logiciel de gestion). Ces fichiers ne doivent pas se retrouver dans votre gestion de configuration. Le problème c'est que "git status" vas les détecter comme de nouveaux fichiers.

Pour remédier à cela, il est possible d'ajouter à votre projet un fichier ".gitignore" qui vas lister les fichiers et dossiers à ne pas prendre en compte.

Attention: ce fichier doit être ajouté à la gestion de version pour qu'il soit partagé avec les autres développeur et pour suivre son évolution.

EXEMPLE (GOT):

1. **Créer le le fichier créer stark.txt:**
 - Arya Stark
 - Sansa Stark
 - Robb Stark
 - Eddard Stark
 - Catelyn Stark
 - Rickon Stark
 - Bran Stark
2. **git status; git diff**
3. **créer .gitignore:**
build/

***.exe**

stark.txt

- 1. git status; git diff**
- 2. git add .gitignore**
- 3. git commit -m "Ajout .gitignore"**

Indexation sélective

Dossier de travail

```
<body>
- <!-- TITRE -->
- <div>
+ <div id="top-banner">
+   Bienvenue dans cette formation GIT !
</div>
</body>
```

Zone d'index

```
<body>
- <div>
+ <div id="top-banner">
+   Bienvenue dans cette formation GIT !
</div>
</body>
```

```
git add --patch [fichier] // Indexation interactive
```

Parfois, on ne souhaite pas indexer ou désindexer l'intégralité d'un fichier mais uniquement des modifications précises. par exemple ne pas indexer les printf de debug. Pour cela on ajoute l'option "--path" à la commande "git add". A ce moment là, on passe en mode interactif.

Stage this hunk [y,n,q,a,d,s,e,?]?

y - stage this hunk

n - do not stage this hunk

q - quit; do not stage this hunk or any of the remaining ones

a - stage this hunk and all later hunks in the file

d - do not stage this hunk or any of the later hunks in the file

s - split the current hunk

e - manually edit the current hunk

? - print help

EXEMPLE (GOT):

1. **Modifier todo.txt:**
 - La montagne
 - Le limier
2. **git status; git diff**
3. **git add --patch todo.txt**
4. **git status; git diff --cached**
5. **git commit -m "Ajout de la montagne"**

1. **git add .**
2. **git status; git diff --cached**
3. **git commit -m "Ajout du limier"**

Désindexation sélective

Dossier de travail

```
<body>
- <!-- TITRE -->
- <div>
+ <div id="top-banner">
+   Bienvenue dans cette formation GIT !
+ </div>
</body>
```

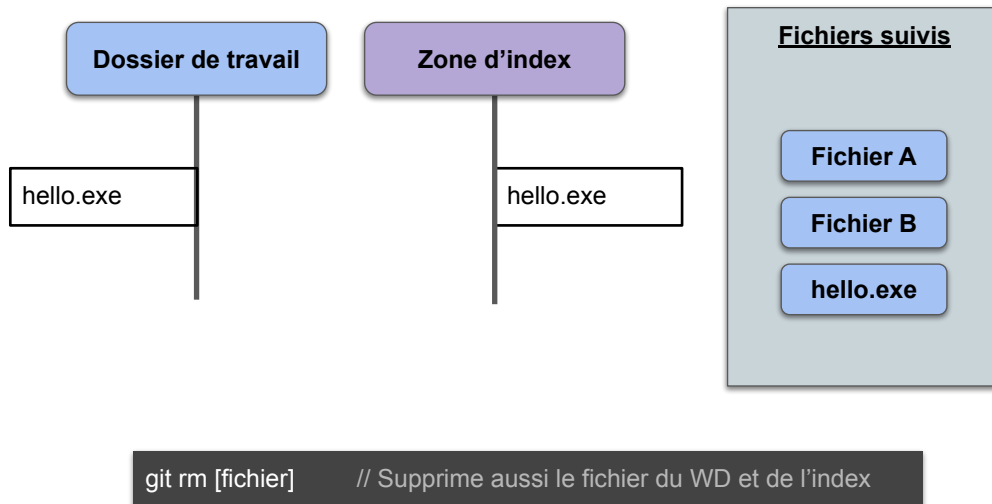
Zone d'index

```
<body>
- <div>
+ <div id="top-banner">
+   Bienvenue dans cette formation GIT !
+ </div>
</body>
```

```
git restore --staged --patch [fichier] // désindexation interactive
```

Il est également possible de désindexer de manière sélective via un procédé identique à l'indexation

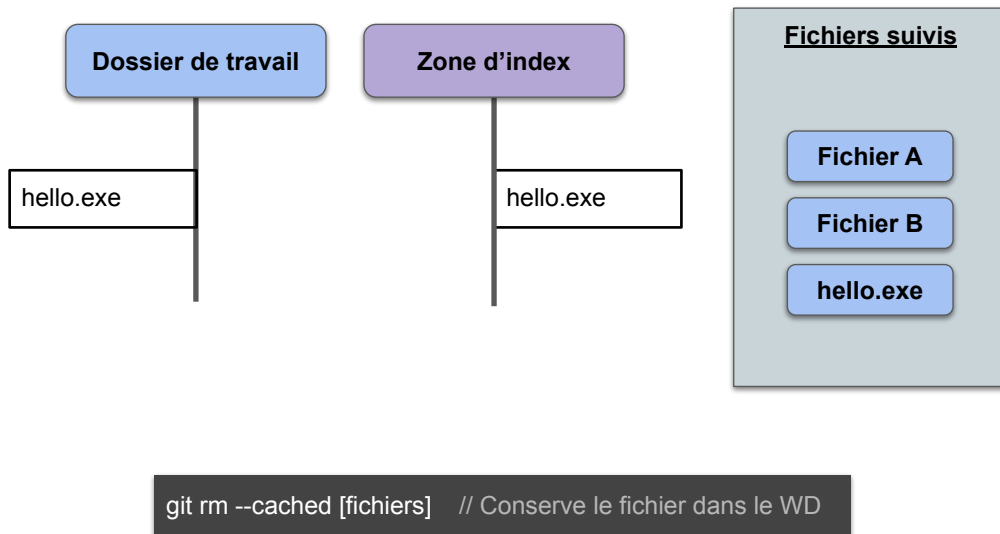
Retirer un fichier de la liste de suivi



Si vous souhaitez retirer un fichier du suivi de Git, vous pouvez utiliser la commande "git rm"

Attention, cela supprime également le fichier dans le workspace et la zone d'index

Retirer un fichier de la liste de suivi [cached]

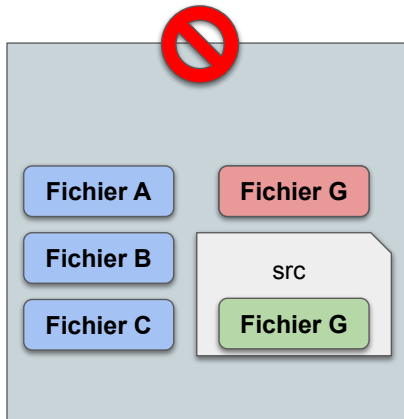


Si vous souhaitez retirer un fichier du suivi de Git alors que celui-ci se trouve dans la zone d'index, vous pouvez utiliser la commande "git rm --cached" qui fera d'une pierre deux coup:

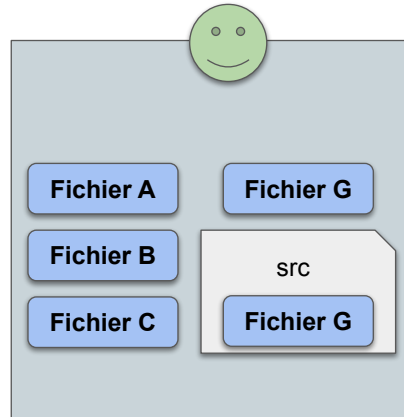
- 1- désindexer le fichier
- 2- retirer le fichier du suivi

en revanche, le fichier sera toujours présent dans l'espace de travail

Déplacer / Renommer un fichier



```
mv [origine] [cible]
```



```
git mv [origine] [cible]
```

Vous réorganisez l'arborescence de votre projet et devez alors déplacer des fichiers? Utilisez la commande `git mv` plutôt que la commande `mv` seule. car dans ce dernier cas, git considère que vous avez supprimé le fichier d'origine et créé un autre fichier.

TP 3.1

Scénario

Vous arrivez sur un projet qui n'utilise pas encore de gestion de configuration... remédions à cela avec Git !

Objectifs

1. Créer un dépôt git dans le dossier du **tp3_1_enonce**.
2. **Faire un commit** des fichiers présents dans le workspace avec comme **message "initial commit"**
3. Editer le fichier hello.html et modifier le **titre** de la page web pour **"Formation Git"** et le **contenu de la page** pour **"Bienvenue dans cette formation GIT"**.
4. Diviser ces modifications en deux commits.
 - a. commit uniquement la modification du titre
 - b. commit le reste

CORRECTION:

1. le projet existe déjà, se déplacer dans le répertoire
 - a. git init
 - b. git status
2. Il faut penser à indexer les modifications puis faire le commit.
 - a. git add .
 - b. git commit -m"Initial commit"
3. Modifier le fichier hello.html
 - a. <title>Formation Git</title>
 - b. <body><div>Bienvenue dans cette formation GIT</div></body>
4. Découper la modification en 2 commits via l'indexation sélective:
 - a. git add --patch hello.html -> s , y , n
 - b. git status; git diff --cached
 - c. git commit -m"Modification du titre"
 - d. git add .
 - e. git commit -m"Modification du contenu"

TP 3.2

Scénario

Votre collègue souhaite faire un commit de ses dernières modifications mais malheureusement il se retrouve bloqué. Aidez le.

- Il a mis dans la zone d'index toutes ses modifications via un "git add ." alors qu'il ne voulait commit que sa modification sur le fichier html. Sa modification du fichier css est à réserver pour un prochain commit.
- Il a supprimé le dossier image sans faire exprès.

Objectifs

1. Faire en sorte de commit uniquement le fichier html. Aidez le à faire le commit sans l'option -m pour lui montrer. Indiquez comme message "correction du chemin du fichier style".
2. Aidez le à récupérer le dossier image.

CORRECTION:

1. Il faut soit désindexer ce qui ne doit pas être commit puis faire le commit pour sauvegarder au plus vite le travail terminé:
 - a. git restore --cached .
 - b. git add hello.html
 - c. git status
 - d. git commit -m"correction du chemin du fichier style"
2. La suppression d'un fichier est une modification qui "supprime toutes les ligne" il faut alors annuler les modifications:
 - a. git restore images/
 - b. git status

TP 3.3

Scénario

Votre stagiaire a fait de mauvais commits et se retrouve avec les fichiers binaires sous suivi Git.

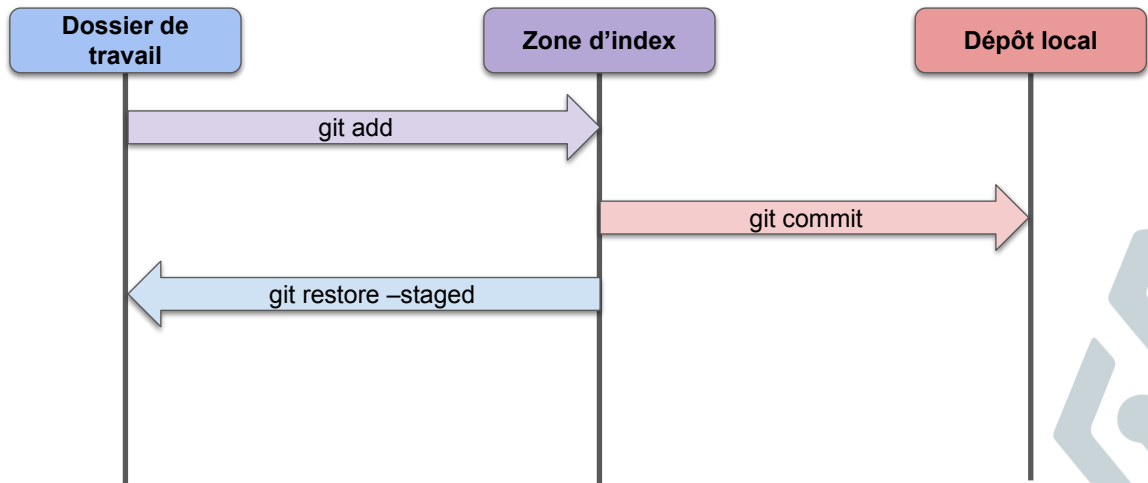
Objectifs

1. Déplacer le fichier main.c dans src.
2. Supprimer le dossier bin de la gestion mais le conserver dans le workspace.
3. Mettre en place un système permettant de ne plus faire de commit du dossier bin par erreur.
4. Faire le commit de toutes ces modifications avec le message "remise au propre du dépôt"
5. Mince!, il a oublié de changer le texte du printf pour "Formation Git <codeur-pro>". Modifiez le commit que vous venez de faire pour ajouter cette modification.

CORRECTION:

1. Attention à utiliser une commande Git pour conserver l'historique du fichier:
 - a. `mkdir src`
 - b. `git mv main.c src/main.c`
2. Faire en sorte que le dossier bin ne soit plus suivi par Git:
 - a. `git rm --cached bin/hello_world`
3. Il faut utiliser un fichier .gitignore:
bin/
4. `git commit -m"Remise au propre du dépôt"`
5. Modifier le fichier src/main.c:
 - a. `git commit --amend`

Bilan



Nous avons vu ensemble comment utiliser la zone d'index afin de préparer notre commit. Nous avons également vu comment créer des commits et comment visualiser l'historique constitué. Nous allons maintenant voir plus en détail comment utiliser un historique dans Git