Digital Academy by insy2s

QU'EST-CE QUE GIT ?

Git est un logiciel de **versioning** créé en 2005 par **Linus Torvalds**, **le créateur de Linux**.

Un logiciel de versioning, ou **gestionnaire de versions** est un logiciel qui permet de conserver un historique des modifications.

Il permet de rapidement identifier les changements effectués et de revenir à une ancienne version en cas de problème.

Les gestionnaires de versions sont quasiment incontournables aujourd'hui car ils facilitent grandement la gestion de projets et ils permettent de travailler en équipe de manière beaucoup plus efficace.

QU'EST-CE QUE GIT ?

Parmi les logiciels de gestion de versions, Git est le leader incontesté et il est donc indispensable pour tout développeur de savoir utiliser Git.

Ainsi **si vous travaillez seul**, vous pourrez conserver l'historique de vos modifications ou revenir sur une version précédente, **si vous travaillez en équipe**, le gestionnaire de versions fusionnera les modifications des personnes qui travaillent simultanément sur un même fichier, ainsi vous ne risquez plus de voir votre travail supprimé par erreur.

QU'EST-CE QUE GITHUB ?

GitHub est un service en ligne qui va héberger votre dépôt Git.

Dans ce cas nous parlerons **de dépôt distant** puisque vos fichiers ne seront pas stockés sur votre machine mais à distance.

De cette manière, votre projet ou fichier pourra être modifié ou distribué par les autres développeurs à distance et aussi cela vous permettra de garder un historique délocalisé.

Vous pourrez avoir plusieurs dépôts distants avec des droits différents (lecture seule, écriture, etc).

QU'EST-CE QUE GITHUB ?

GitHub est le service en ligne le plus populaire et le plus utilisé comparé à **GitLab** ou **BitBucket** par exemple.

Beaucoup de recruteurs demandent désormais le lien GitHub des candidats lors des processus de recrutement pour voir la qualité de leur code, leurs capacités et leur plus-value.

CRÉER UN COMPTE GITHUB

Pour créer votre compte GitHub, rendez-vous à l'adresse suivante :

https://github.com

Par défaut GitHub est gratuit mais si vous souhaitez passer à la vitesse supérieure, il existe également des offres payantes.

► **Votre tableau de bord** permettra de suivre les problèmes, d'extraire les demandes sur lesquelles vous travaillez ou que vous suivez, accéder à vos repositories et rester à jour sur les activités récentes des organisations et des repositories auxquels vous êtes abonné.

► **L'interface repositories** est l'emplacement où vous pourrez créer et retrouver vos dépôts existants.

CRÉER UN COMPTE GITHUB

► **Votre profil** pour éditer vos informations, voir vos contributions sur différents projets ou repositories, que ce soient les vôtres, ceux d'autres personnes ou sur des repositories publics.

► **Le Pull requests** permet de faire des demandes de modifications réalisées sur le code

► **L'explore** permet de trouver de nouveaux projets open source sur lesquels travailler

INSTALLATION DE GIT SUR VOTRE MACHINE

Pour télécharger Git sur votre ordinateur, rendez-vous à l'adresse suivante :
https://git-scm.com/downloads

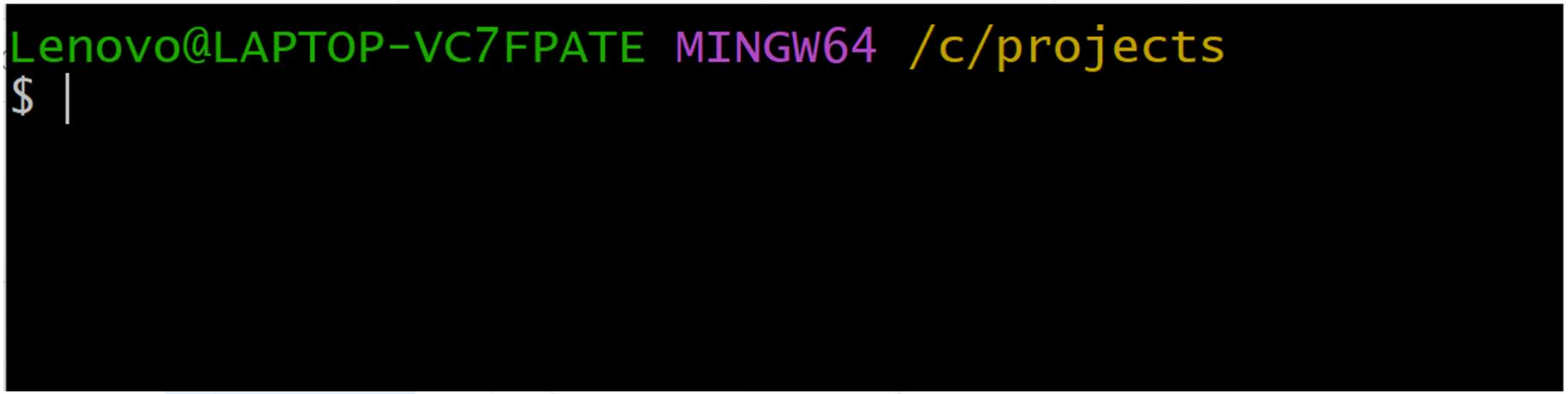
Choisissez la dernière version de Git correspondant au système d'exploitation que vous utilisez (Windows, MacOS ou Linux/Unix) puis cliquez sur download.

Une fois téléchargé, exécutez le fichier récupérer pour procéder à l'installation de Git sur votre ordinateur en laissant tout par défaut.

Sur Windows, si on vous le demande, décochez la case **"Launch Git Bash"** pour ne pas le lancer à la fin de l'installation et décochez également la case **"View Release Note".**

INITIALISER & CRÉER LE DÉPÔT LOCAL

Nous allons configurer Git (***git config***), mais pour cela nous allons avoir besoin du **terminal Git Bash**.



INITIALISER & CRÉER LE DÉPÔT LOCAL

Configurer votre identité

La première étape va être de renseigner votre nom et adresse mail.
Pour cela, tapez les commandes suivantes dans Git Bash :

```
git config --global user.name "Prénom Nom"
git config --global user.email adresse@example.com
```

--global permet d'appliquer la configuration à tous vos projets, sinon cela s'appliquera uniquement au projet en cours.

Pour vérifier si vos paramètres ont bien été pris en compte, tapez la commande :

```
git config --list
```

INITIALISER & CRÉER LE DÉPÔT LOCAL

Créer et initialiser le dépôt local

Pour créer un dossier de dépôt sur votre ordinateur en ligne de commande, taper la commande :

```
mkdir nomdudossier
```

Pour entrer dans ce nouveau dossier taper :

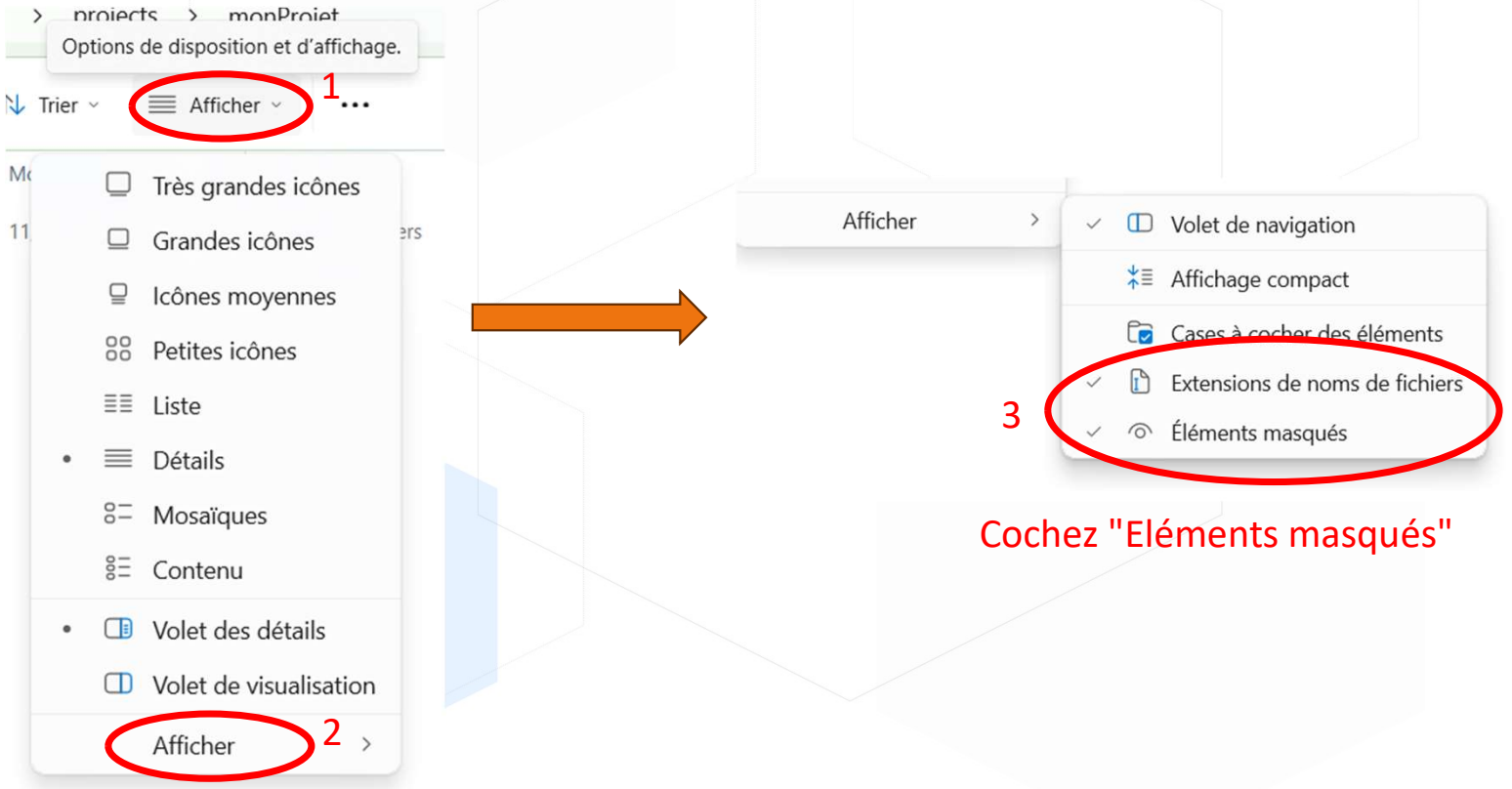
```
cd nomdudossier
```

Pour initialiser le dépôt (cela créera un fichier caché .git dans le dossier créé), taper :

```
git init
```

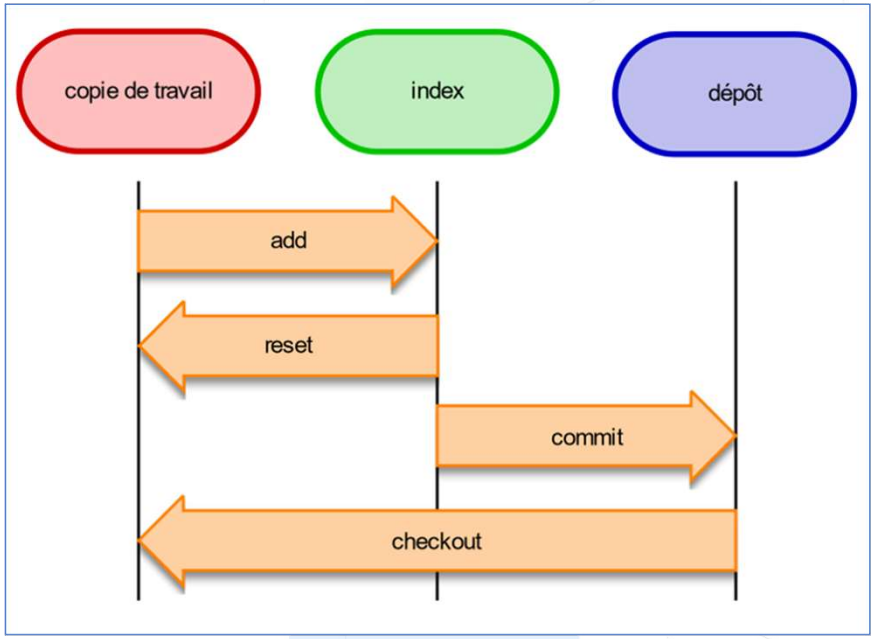
INITIALISER & CRÉER LE DÉPÔT LOCAL

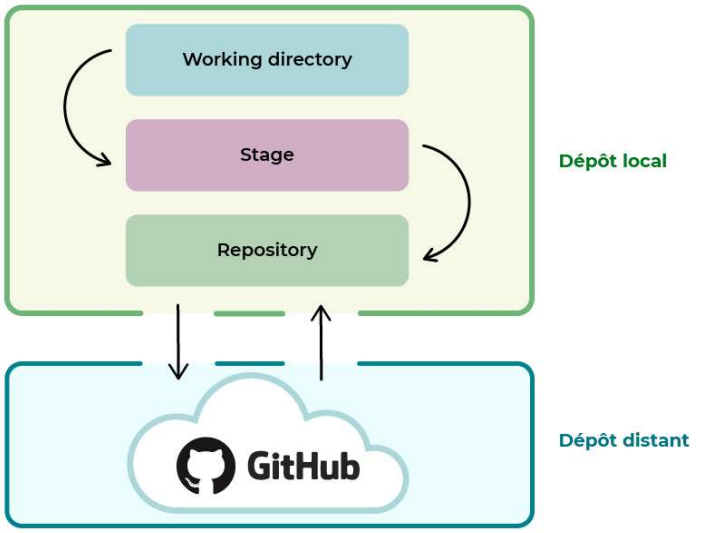
Si vous ne voyez pas le dossier .git dans le dossier de votre projet, faites cette manipulation



Cochez "Eléments masqués"

SCHÉMAS DU DÉROULEMENT





Source : https://openclassrooms.com/fr/

INITIALISER & CRÉER LE DÉPÔT LOCAL

Créer et initialiser le dépôt local

Pour créer un fichier comme par exemple un fichier html taper la commande :

```
touch index.html
```

Pour lister tous les nouveaux fichiers et les fichiers modifiés à commiter, taper :

```
git status
```

Pour ajouter un fichier à l'**index** (***stage***), taper :

```
git add index.html
```

Si le projet est composé de plusieurs dossiers ou fichiers, vous pouvez les ajouter ensemble en tapant la commande :

```
git add .
```

INITIALISER & CRÉER LE DÉPÔT LOCAL

Créer et initialiser le dépôt local

Pour enregistrer des instantanés de fichiers de façon permanente dans l'historique des versions, taper :

```
git commit -m "ajout du fichier index.html"
```

L'option ***-m*** permet de mettre un commentaire au projet directement en ligne de commande, sinon il faudra l'ajouter manuellement avec l'éditeur de texte, faites-le régulièrement afin de savoir exactement ce qui a été fait à chaque modification, ce sera très utile pour suivre l'évolution du projet.

Envoyer le dépôt local vers le dépôt distant

Pour relier le repo local avec le distant, taper ([…] correspond à votre repo):

```
git remote add origin https://github.com/[...].git
```

Puis :

```
git branch -M main
```

INITIALISER & CRÉER LE DÉPÔT LOCAL

Envoyer le dépôt local vers le dépôt distant

Enfin, pour envoyer tous les commits de la branche locale vers GitHub :

```
git push
```

À chaque **mise à jour** ou **nouvelle version** de votre projet en local il faudra penser à reproduire la même procédure à partir du ***add*** jusqu'au ***push.***

LE SYSTÈME DE BRANCHES

Ce que nous avons vu jusqu'à présent c'est bien beau, mais le souci est que chaque modification est directement reporté sur **la branche principale**, la branche **main,** ce qui *"écrase"* la version précédente.

En effet, **la branche principale** portera l'**intégralité** des modifications effectuées.

Le but n'est donc pas de réaliser les modifications directement sur cette branche, mais de les réaliser sur d'autres branches, et après divers tests, de les intégrer sur la branche principale.

Ces branches secondaires correspondront à autant de dossiers différents.

Les différentes branches correspondent à des copies de votre code principal à un instant T, où vous pourrez tester toutes vos idées sans que cela impacte votre code principal.

LE SYSTÈME DE BRANCHES

Pour connaître les branches présentes dans votre projet, tapez :

```
git branch
```

Pour créer une nouvelle branche, tapez :

```
git branch nomdelabranche
```

Pour basculer d'une branche à l'autre :

```
git checkout nomdelabranche
```

FUSIONNER LES BRANCHES

Une fois que vous avez terminé votre travail sur **la branche secondaire**, il va falloir intégrer l'évolution de votre projet à **la branche principale**.

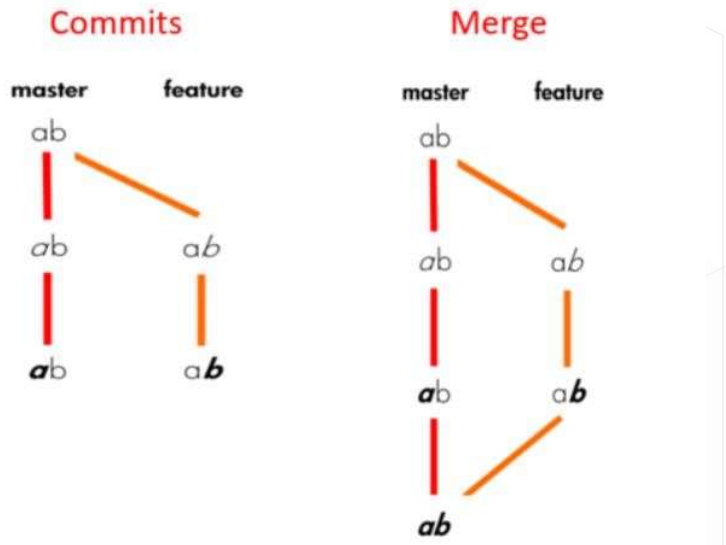
Via le **terminal Git Bash,** il faudra d'abord **switcher** sur **la branche principale** :

```
git checkout main
```

Ensuite, pour **fusionner** votre **branche secondaire** à celle-ci, tapez :

```
git merge nomdelabranche
```

FUSIONNER LES BRANCHES



CRÉER VOTRE DÉPÔT SUR GITHUB

Pour pouvoir déposer votre projet sur GitHub, vous devez créer un repository.

Cliquez sur le **"+"** en haut, dans le coin supérieur droit, puis choisissez l'option **"New repository"**.

Donnez-lui un nom ***(à retenir pour le reporter plus tard dans Git)***, choisissez un dépôt public, vous pourrez ensuite cocher ou décocher les options proposés plus bas (**README** = indique les informations clés de votre projet, **.gitignore** = permet d'ignorer certains fichiers de votre projet Git).

Cliquez sur **"Create repository"**, votre dépôt est créé.

Ce ***repo distant*** nous servira pour récupérer ***le repo local*** que vous créerez sur votre machine via Git.

ACCÉDER À UN DÉPÔT DISTANT

Si vous travaillez à plusieurs sur un projet et que vous souhaitez **récupérer un repository sur GitHub** mis en place par un membre du groupe pour apporter vos modifications, vous devez d'abord récupérer **l'URL du dépôt distant**.

Rendez-vous sur GitHub et allez sur le projet.
Cliquez ensuite sur le bouton "Code" en vert pour copier l'URL.
Retournez sur **Git Bash** et tapez :

```
git clone urlcopiée
```

METTRE À JOUR LE DÉPÔT LOCAL

Pour fusionner toutes les modifications présentes sur le dépôt distant dans le répertoire de travail local, la commande pull est utilisée.

```
git pull
```

Cela mettra à jour l'ensemble des fichiers et dossiers du repo dans votre répertoire courant en local.

COLLABORER SUR GITHUB

Exemple

Notre dépôt local vient d'être mis à jour.
Vérifions les branches de notre projet :

```
git branch
```

Nous obtenons ceci :

```
*main
new-version-html
new-version-css
```

En plus de la branche principale **main**, nous avons deux autres branches.
Nos collègues ont déjà envoyé (***push***) les branches pour créer de nouvelles fonctionnalités.

COLLABORER SUR GITHUB

Exemple

Dans un contexte professionnel, vous ne pourrez pas fusionner (***merge***) les modifications des branches avec la branche principale puisque celle-ci sera bloquée.
En effet, il doit y avoir **une vérification du code** avant de le pusher.
Vous devez faire ce que nous appelons ***une pull request*** directement sur votre interface **GitHub**.
Donc si vous créez ***une pull request***, vous aurez au préalable :

- ► Créer une nouvelle branche
- ► Envoyé votre code sur cette même branche.

Ces deux conditions rassemblées, **un bandeau** apparaîtra sur GitHub vous suggérant de ***créer la pull request.***

COLLABORER SUR GITHUB

Exemple

Vous cliquerez dessus, **mettrez un commentaire** pour expliquer les raisons de vos modifications, puis **vous validerez** en cliquant sur ***create pull request***.

Une fois votre ***pull request*** créée, elle peut être **fusionnée à la branche principale**.

Sur certains projets, il peut arriver que votre code ne puisse pas être fusionné sur la branche principale **sans être révisé et validé** par d'autres membres du projet.

Nous appelons cela une ***Code Review***.

Cela permet de prévenir de certaines erreurs éventuelles ou d'en discuter.

GESTION DES CONFLITS

Lors de l'ajout d'un code avec le code existant (lors d'un pull ou un pull request par exemple), il y aura une fusion entre les 2 codes, on a ce qu'on appelle un **Merge,**

Il y a 2 types de merges :

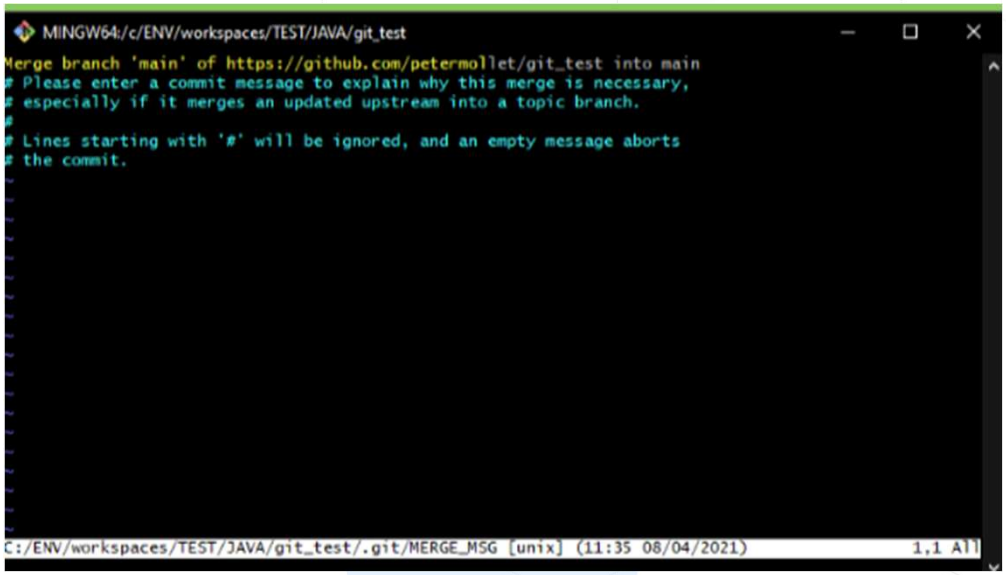
- Merge Automatique réussi
- Merge avec conflits

GESTION DES CONFLITS

Merge Automatique réussi :

Il n'y pas de conflits : les codes à ajouter et le code existant sont sur des lignes différentes.

Le merge peut générer un message :



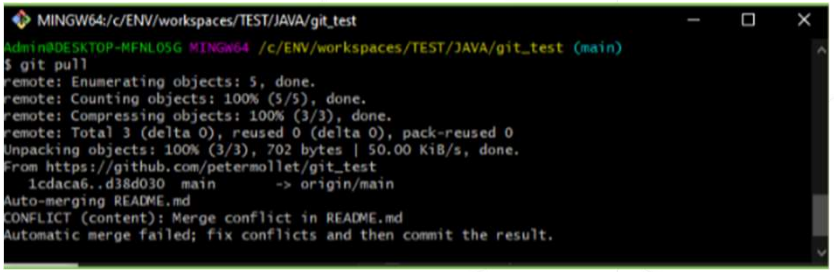
Pour passer ce message cliquer sur ESC (Echap) puis :wq et entrer

GESTION DES CONFLITS

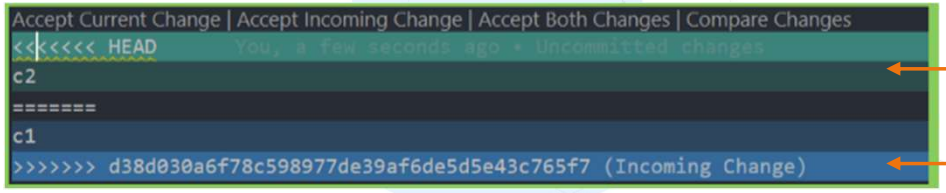
Merge avec conflits :

Il y a de conflits : les codes à ajouter et le code existant sont sur des lignes similaires.

Le merge peut générer un message :



Il faut aller sur le ou les fichier(s) et gérer les conflits



Nouveau code

Code déjà présent dans le repo

GESTION DES CONFLITS

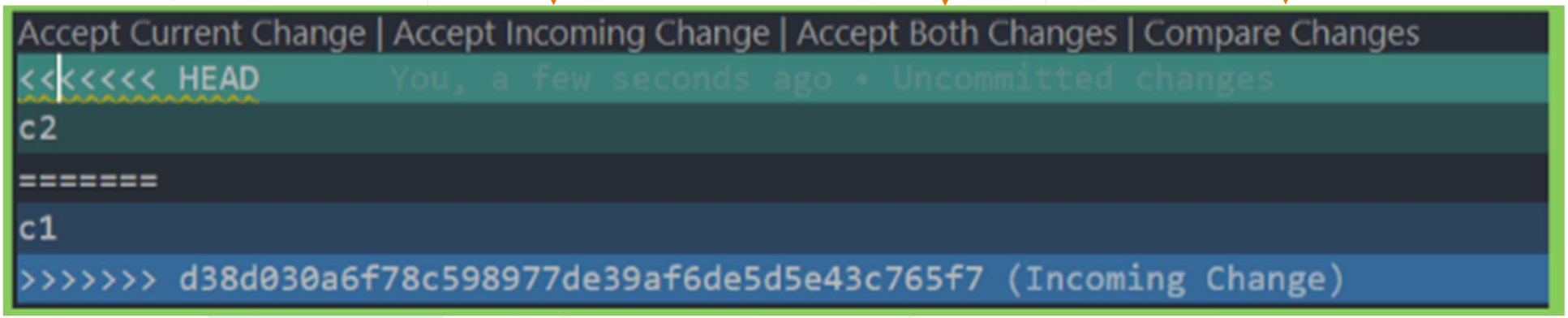
Merge avec conflits :

Accepter le Nouveau code

Accepter le code déjà présent dans le repo

Accepter les 2 codes

Comparer les changements



GESTION DES CONFLITS

Merge avec conflits :

Après votre modification, il faudra valider votre merge (tout comme toutes modifications de code

LIENS UTILES

https://training.github.com/downloads/fr/github-git-cheat-sheet.pdf

https://training.github.com/downloads/fr/github-git-cheat-sheet/

https://prezi.com/view/Zyk2WGGsJmr995jOpWjA/

https://github.com

https://git-scm.com/downloads