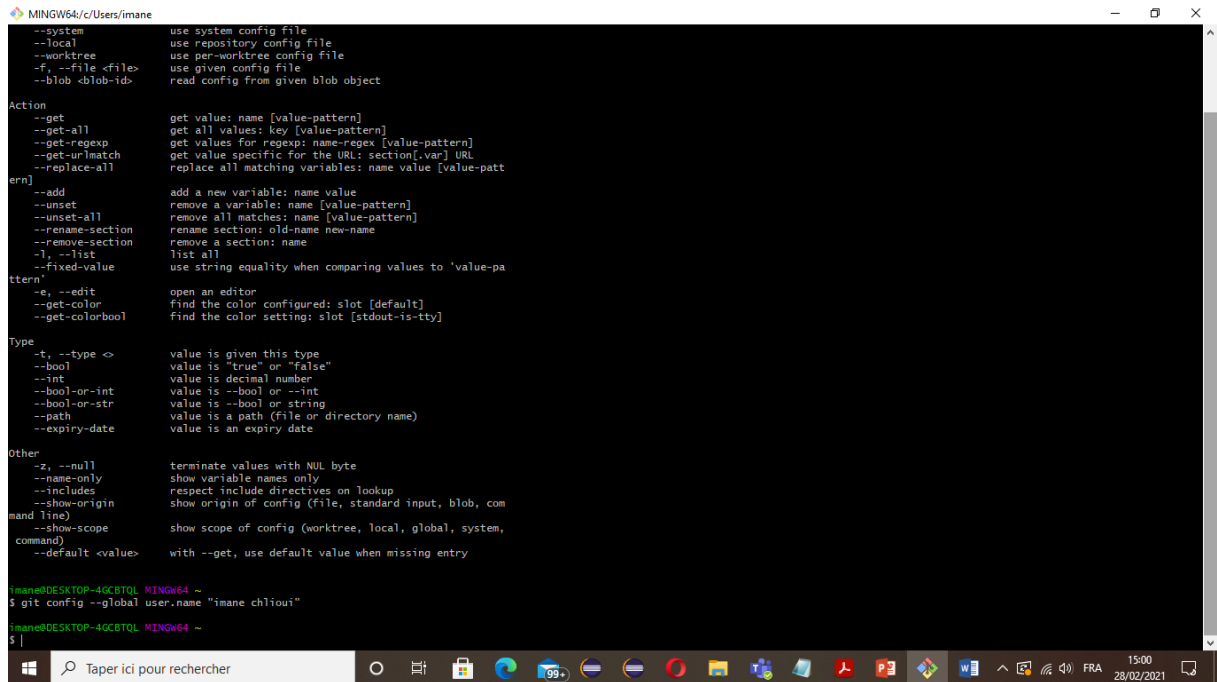


# TP1 : GIT

- Indiquer votre nom pour savoir qui a fait le commit à l'aide de la commande :

**Git config --global user.name « votre nom »**



```
MINGW64/c/Users/imane
--system      use system config file
--local       use repository config file
--worktree    use per-worktree config file
-f, --file <file>  use given config file
--blob <blob-id>  read config from given blob object

Action
--get         get value: name [value-pattern]
--get-all    get all values: key [value-pattern]
--get-regexp  get values for regexp: name-regexp [value-pattern]
--get-urlmatch get value specific for the URL: section[-var] URL
--replace-all replace all matching variables: name value [value-pattern]

ern]
--add         add a new variable: name value
--unset       remove a variable: name [value-pattern]
--unset-all  remove all matches: name [value-pattern]
--rename-section rename section: old-name new-name
--remove-section remove a section: name
-l, --list    list all
--fixed-value use string equality when comparing values to 'value-pattern'

ttern'
-e, --edit    open an editor
--get-color   find the color configured: slot [default]
--get-colorbool find the color setting: slot [stdout-is-tty]

Type
-t, --type <type> value is given this type
--bool         value is "true" or "false"
--int          value is decimal number
--bool-or-int  value is --bool or --int
--bool-or-str  value is --bool or string
--path         value is a path (file or directory name)
--expiry-date  value is an expiry date

Other
-z, --null     terminate values with NUL byte
--name-only    show variable names only
--includes     respect include directives on lookup
--show-origin  show origin of config (file, standard input, blob, command line)
--show-scope   show scope of config (worktree, local, global, system, command)
--default <value> with --get, use default value when missing entry

imane@DESKTOP-4GCR7QL MINGW64 ~
$ git config --global user.name "imane chlioui"

imane@DESKTOP-4GCR7QL MINGW64 ~
$
```

- Pour s'assurer que votre nom a bien changé vous pouvez taper :

**Git config --global user.name**

- De meme pour votre adresse mail vous pouvez la changer à l'aide de la commande :

**Git config -- global user.email « votre email »**

```

MINGW64/c/Users/imane
ttern'
-e, --edit          open an editor
--get-color         find the color configured: slot [default]
--get-colorbool     find the color setting: slot [stdout-is-tty]

Type
-t, --type <v>     value is given this type
--bool              value is "true" or "false"
--int               value is decimal number
--bool-or-int       value is --bool or --int
--bool-or-str       value is --bool or string
--path              value is a path (file or directory name)
--expiry-date       value is an expiry date

Other
-z, --null          terminate values with NUL byte
--name-only         show variable names only
--includes          respect include directives on lookup
--show-origin       show origin of config (file, standard input, blob, com
mand line)
--show-scope        show scope of config (worktree, local, global, system,
command)
--default <value>   with --get, use default value when missing entry

imane@DESKTOP-4GCBTQL MINGW64 ~
$ git config --global user.name "imane chlioui"

imane@DESKTOP-4GCBTQL MINGW64 ~
$ git config --global user.name
imane chlioui

imane@DESKTOP-4GCBTQL MINGW64 ~
$ git config --global user.email "imanechlioui@gmail.com"

imane@DESKTOP-4GCBTQL MINGW64 ~
$

```

- Avant de commencer n'importe quel projet il faut toujours l'initialiser, à l'aide de la commande : **Git init**

1) Après l'exécution de cette commande qu'est ce qui a changé ?

- Alors maintenant on va ajouter 2 fichiers à notre dossier (par exp test1.txt et test2.txt) et les ajouter à Git pour suivre les changements

**Git add text1.txt**

**Git add text2.txt**

**Git status**

} Ou bien la commande **Git add -A**  
pour ajouter tous les fichiers

```
MINGW64:/i/teaching/Devops/tps
Initialized empty Git repository in I:/teaching/Devops/tps/.git/

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git add test1.tx
fatal: pathspec 'test1.tx' did not match any files

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git add test1.txt

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git add testé.txt
fatal: pathspec 'testé.txt' did not match any files

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git add test2.txt

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git status
On branch master

No commits yet

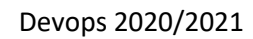
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   test1.txt
        new file:   test2.txt

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$
```

2) Modifier l'un des fichiers et retaper la commande : **Git status** : Qu'est ce qui s'est passé ?

- Pour capturer un instantané des changements actuellement stagés du projet on utilise la commande **Git commit**. Cette commande nous permet de créer une version sure du projet.

Un nouvel éditeur est ouvert avec les détails de votre commit :



—      □      ×

```
I:/teaching/Devops/tps/.git/COMMIT_EDITMSG [unix] (15:40 28/02/2021) 1,0-1 All
"I:/teaching/Devops/tps/.git/COMMIT_EDITMSG" [unix] 12L, 255B
```

- ## Git commit -m « message »

```
MINGW64:/i/teaching/Devops/tps
(use "git rm --cached <file>..." to unstage)
new file:   test1.txt
new file:   test2.txt

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:   test2.txt

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git add test2.txt

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git commit
Aborting commit due to empty commit message.

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git commit
Aborting commit due to empty commit message.

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git commit -m "initial commit"
[master (root-commit) 6b3ae22] initial commit
2 files changed, 2 insertions(+)
create mode 100644 test1.txt
create mode 100644 test2.txt

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$
```

3) Qu'est que la commande Git status vous affiche après le commit ?

- Pour faire le suivie des commit effectués, il suffit de taper la commande :

### Git log

```
MINGW64:/i/teaching/Devops/tps
imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git commit
Aborting commit due to empty commit message.

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git commit
Aborting commit due to empty commit message.

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git commit -m "initial commit"
[master (root-commit) 6b3ae22] initial commit
2 files changed, 2 insertions(+)
create mode 100644 test1.txt
create mode 100644 test2.txt

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git status
On branch master
nothing to commit, working tree clean

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$ git log
commit 6b3ae223b6a3f2101835ed33694571a3651515ac (HEAD -> master)
Author: imane chlioui <imanechlioui@gmail.com>
Date:   Sun Feb 28 15:43:19 2021 +0100

    initial commit

imane@DESKTOP-4GCBTQL MINGW64 /i/teaching/Devops/tps (master)
$
```

**Exercice à faire :**

Exercice à faire par groupe de 2 ou 3 : Pour tout le monde :

- Créez un compte sur GitHub.
- Installez GitHub Desktop, puis ouvrez-le.

Une personne du groupe :

- Créez un dépôt nommé TP Git

Une personne :

1. Créez un fichier avec NotePad++ (ou autre) dans le dépôt sur votre ordinateur.
  2. Remplissez le fichier avec un bout de code (n'importe quel langage, il peut être un fichier texte).
  3. Sauvegardez le fichier.
  4. Reprenez GitHub Desktop et ajoutez le fichier.
  5. Faites un commit.
  6. Faites un push vers le dépôt GitHub en ligne.
  7. Publiez votre dépôt et ajoutez les autres en collaborateurs sur le dépôt. Ils reçoivent une invitation par mail, ils doivent l'accepter.
- Ensuite, les autres, chacun à son tour :
8. Faites un pull
  9. Regardez l'historique pour vérifier que les changements sont là.
  10. modifier/ajouter dans le fichier.
  11. Refaites les étapes 3 à 6 ci-dessus.

N'hésitez pas à répéter cela plusieurs fois, pour être sûrs de bien comprendre !

Partie à faire individuellement (chacun sur son ordinateur, simultanément) :

- Ouvrez GitHub Desktop et clonez le dépôt précédemment créé si ce n'est déjà fait.
- Ajoutez/modifiez dans le fichier. Ne pas oublier de sauver le fichier.



- Ajoutez le fichier dans GitHub Desktop.
- Faites un commit. (pas de push)
- Observez et comparez les historiques de chacun.

Une personne fait un push. Les autres personnes ne font rien.

- Une des 2 autres personnes fait le pull sur son ordinateur (cliquez sur "push" si "pull" n'est pas affiché, et cliquez "close" sur le message d'erreur qui s'affiche).
- Résolvez le conflit de merge ensemble (il faut éditer le fichier).
- Une fois le merge terminé, faites un commit et un push.
- La dernière personne fait le pull sur son ordinateur.
- Résolvez le conflit de merge ensemble pour avoir toutes les blagues.
- Les autres peuvent faire un pull pour récupérer toutes les blagues.
- Comparez à nouveau vos historiques.