YouCode

POWERED BY SIMPLON

Rapport Scenario 1

En Carde par : Hanane Jabane

Realise par :

Khadija Amardoul

Khadija Melouane

Majda Fannan

Marouan Khadiri

Abdelali El Ouadi

Adam Khairi

SOMMAIRE

1-Immertion :

Cette étape contient une définition bref de git, le fonctionnement de fichier « .git » et la création de 2 répertoires l'une dans l'autre :

- Définition de Git :

Git est un Logiciel de contrôle de version qui permet de tracer l'évolution de projet et d'y apporter des modifications sereinement. Créé par Linus Torvalds.Il est utilisé pour gérer des codes sources.

- Fonctionnement de fichier « .git » :

Quand on initialise le projet, un dossier caché « .git » est automatiquement créé. Ce dossier git contient toutes les informations nécessaires à votre projet dans le contrôle de version. Les informations sur les validations, et l'adresse du référentiel. Il contient également un journal qui stocke votre historique de validation afin que vous puissiez revenir à l'historique.

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop
$ mkdir Projects

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop
$ cd Projects/
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects
$ mkdir Demo

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects
$ cd Demo/


youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo
$ git init
Initialized empty Git repository in C:/Users/youcode/Desktop/Projects/Demo/.git/

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git status
On branch master

No commits yet

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

→Ce dernier commentaire signifie qu'il y a aucune modification au niveau de répertoire 'Demo'

Adding and commiting :

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ touch README.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ code .

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ ls
README.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git add .

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git commit -m "Adding readme.md file"
[master (root-commit) 7d9e493] Adding readme.md file
```

2-Découverte :

Dans cette phase, on découvre d'abord les fichiers de dossier. git.

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ cd .git

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo/.git (GIT_DIR!)
$ ls -al
total 13
drwxr-xr-x 1 youcode 197121   0 nov.  26 11:40 ./
drwxr-xr-x 1 youcode 197121   0 nov.  26 11:37 ../
-rw-r--r-- 1 youcode 197121  22 nov.  26 11:40 COMMIT_EDITMSG
-rw-r--r-- 1 youcode 197121 130 nov.  26 11:36 config
-rw-r--r-- 1 youcode 197121  73 nov.  26 11:36 description
-rw-r--r-- 1 youcode 197121  23 nov.  26 11:36 HEAD
drwxr-xr-x 1 youcode 197121   0 nov.  26 11:36 hooks/
-rw-r--r-- 1 youcode 197121 137 nov.  26 11:40 index
drwxr-xr-x 1 youcode 197121   0 nov.  26 11:36 info/
drwxr-xr-x 1 youcode 197121   0 nov.  26 11:40 logs/
drwxr-xr-x 1 youcode 197121   0 nov.  26 11:40 objects/
drwxr-xr-x 1 youcode 197121   0 nov.  26 11:36 refs/
```

Ainsi en définit quelques clauses de .git :

- HEAD : les informations sur la branche et les commit qu'il contienne etc.
- LOG : l'historique des commits
- BRANCH : un moyen de demander un nouveau répertoire de travail, une nouvelle zone de stagging et un nouvel historique de projet.

Finalement nous avons créé dans la branche master un fichier nommé 'licence.md', puis nous avons commité ce fichier et afficher les fichiers traqués :

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo/.git (GIT_DIR!)
$ cd ..

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ touch Licence.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git commit -m "Adding Licence.md file"
On branch master
Untracked files:
        Licence.md

nothing added to commit but untracked files present

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git add .

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git commit -m "Adding Licence.md file"
[master 1284a5a] Adding Licence.md file
1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Licence.md
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git log --pretty=format: --name-only
Licence.md

README.md
```

3-Historique :

Pour commencer cette étape, nous avons tout d'abord Affichez le dernier commit, toute en ajoutant l'option d'affichage de la hiérarchie de la branche, avec les commit et leur branche aussi. Nous avons fait ça avec la commande suivante :

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git log --all --graph --decorate
* commit 1284a5ac37ad1e969231711b5e43e56efb1d56b9 (HEAD -> master)
| Author: majda FANNAN <fannanmajda123@gmail.com>
| Date:   Tue Nov 26 12:02:14 2019 +0100
|
|     Adding Licence.md file
|

* commit 7d9e493bf93b34bc8803df0ec3f37a99d51a17ce
  Author: majda FANNAN <fannanmajda123@gmail.com>
  Date:   Tue Nov 26 11:40:01 2019 +0100

      Adding readme.md file
```

Ensuite nous avons utilisé l'alias pour remplacer le nom 'long' de cette commande.

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ gir config -global historique "git log -all -graph -decorate
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git config --list | grep alias
alias.historique=log --graph --oneline
alias.last=log -1 HEAD
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git historique README.md
* 7d9e493 Adding readme.md file
```

4-excluding files :

A cet étape, il est demandé de renommer le fichier 'licence.md',et faire le staging avec mise à jour. Mais sans le faire avec la commande « git add . ». Par conséquence nous avons décidé de stagger avec « git add 'nom de fichier' » (il est possible aussi d'utilisé l'alias).

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ mv Licence.md Licence.txt

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ ls
Licence.txt  README.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git add Licence.txt

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git commit -m "Rename Licence.md to Licence.txt"
[master 55fdf8c] Rename Licence.md to Licence.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Licence.txt
```

Maintenant il est le temps d'exclure des fichiers du répertoire local

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ touch application.log

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ touch .gitignore

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ code .

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git add .

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git commit -m "Creation de fichier application.log"
[master ebfd32e] Creation de fichier application.log
 2 files changed, 1 insertion(+)
commit ebfd32eb5ffc6ba8a69ca95f6b19b77dba410685 (HEAD -> master)
Author: majda FANNAN <fannanmajda123@gmail.com>
Date:   Tue Nov 26 15:24:55 2019 +0100
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git ls-files . --exclude-standard --others

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git ls-files . --ignored --exclude-standard --others
application.log

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ code README.md
```

Constat :

Après qu'on a fait le staging/commiting on a constaté que le fichier application.log n'est pas traqué.et c'est à cause de fichier. gitignore qui a pour rôle d'ignorer les fichiers que on veut pas visionner dans Git

5-Branching and Merging :

C'est bien évidemment la phase merging. Nous avons fait des modifications sur le fichier readme.md au niveau de branche master et au niveau d'une nouvelle branche nommé 'updates'. Cependant nous ne nous avons pas commiter qu'après avoir basculé à la branche 'updates'

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ code README.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git branch updates

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
```

```
$ git checkout updates
Switched to branch 'updates'
M       README.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (updates)
$ ls
application.log  Licence.txt  README.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (updates)
$ git add .

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (updates)
$ git commit -a -m "creation branch updates"
[updates 971cc3a] creation branch updates
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Voilà le résultas de ce que nous avons fait :

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (updates)
$ git historique
* 971cc3a (HEAD -> updates) creation branch updates
* ebfd32e (master) Creation de fichier application.log
* 55fdf8c Rename Licence.md to Licence.txt
* 1284a5a Adding Licence.md file
* 7d9e493 Adding readme.md file
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (updates)
$ git checkout master
Switched to branch 'master'

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ cat readme.md
#Demo project un simple fichier
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git historique
* ebfd32e (HEAD -> master) Creation de fichier application.log
* 55fdf8c Rename Licence.md to Licence.txt
* 1284a5a Adding Licence.md file
* 7d9e493 Adding readme.md file

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git merge updates
Updating ebfd32e..971cc3a
Fast-forward
 README.md | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Constat :

- Au niveau de la branche :

On a constaté que la modification qu'on fait avant la création de la branche updates, se trouve dans cette dernière.

- Retournant à la branche master

On a constaté que aucune modification n'affecté la branche master.

6-conflit résolution :

Tout d'abord on a créé une branche qui s'appelle "BAD", Apres on a modifier le fichier README.md en ajoutant la ligne trouble. Et on a fait le adding et le commiting on tapent la commande : « git commit –am «'message'»

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git branch BAD

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ code README.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git commit -am "creation branche BAD"
[master 3e354e3] creation branche BAD

 1 file changed, 2 insertions(+), 1 deletion(-)
```

Retournant vers la branche principale Master, maintenant on va faire une modification sur le même fichier README.md en ajoutant la ligne troubleshooting . Ensuite le staging et le commiting avec un message 'branche bien faite'.

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (BAD)
$ git checkout master
Switched to branch 'master'

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ code readme.md

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git commit -am "branche bien faite"
[master 845e6ef] branche bien faite
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Cette commande permet de corriger une erreur dans un message de commit Précédente.

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git commit --amend -m "ajout de ligne trouble shooting"
[master dbfad01] ajout de ligne trouble shooting
 Date: Tue Nov 26 16:54:34 2019 +0100
 1 file changed, 2 insertions(+), 1 deletion(-)
```

L'historique des commit :

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git log
commit dbfad01ef562738fcc822ad416603538daccae54 (HEAD -> master)
commit dbfad01ef562738fcc822ad416603538daccae54 (HEAD -> master)
commit dbfad01ef562738fcc822ad416603538daccae54 (HEAD -> master)
Author: majda FANNAN <fannanmajda123@gmail.com>
Date:   Tue Nov 26 16:54:34 2019 +0100

    ajout de ligne trouble shooting

commit 971cc3a9dd5d2329aef5a0974146441b24ae2953 (updates)
Author: majda FANNAN <fannanmajda123@gmail.com>
Date:   Tue Nov 26 16:18:07 2019 +0100

    creation branch updates

commit ebfd32eb5ffc6ba8a69ca95f6b19b77dba410685
Author: majda FANNAN <fannanmajda123@gmail.com>
Date:   Tue Nov 26 15:24:55 2019 +0100

    Creation de fichier application.log

commit 55fdf8c56271ed88189fb86d4f0289505b0d0eca
Author: majda FANNAN <fannanmajda123@gmail.com>
Date:   Tue Nov 26 15:19:42 2019 +0100

    Rename Licence.md to Licence.txt

commit 1284a5ac37ad1e969231711b5e43e56efb1d56b9
Author: majda FANNAN <fannanmajda123@gmail.com>
Date:   Tue Nov 26 12:02:14 2019 +0100

    Adding Licence.md file

commit 7d9e493bf93b34bc8803df0ec3f37a99d51a17ce
Author: majda FANNAN <fannanmajda123@gmail.com>
Date:   Tue Nov 26 11:40:01 2019 +0100
```

On a constaté que le message de commit était changer.

- **Explication de ce petite scenario :**

On va fusionner la branche ‘ BAD ‘ avec la branche principale ‘master ‘.

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master)
$ git merge BAD
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master|MERGING)
$ cat README.md
#Demo project un simple fichier (modifications1)

<<<<<<< HEAD
troubleshooting
=======
trouble
>>>>>>> BAD
```

```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master|MERGING)
$ git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
opendiff kdiff3 tkdiff xxdiff meld tortoisemerge gvimdiff diffuse diffmerge ecmerge
p4merge araxis bc codecompare smerge emerge vimdiff
Merging:
README.md

Normal merge conflict for 'README.md':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
```

Après le fusionnement on voit un message de conflit à cause de modification du même fichier README .md dans les deux branches

7-mergetools :

C'est le temps pour les mergetools. Nous avons fait le choix d'installer le programme p4merge. Après l'installation, On a configuré git on ajoutant p4merge comme un mergetool. Ainsi on a édité la configuration de prompt au niveau de fichier. gitconfig :

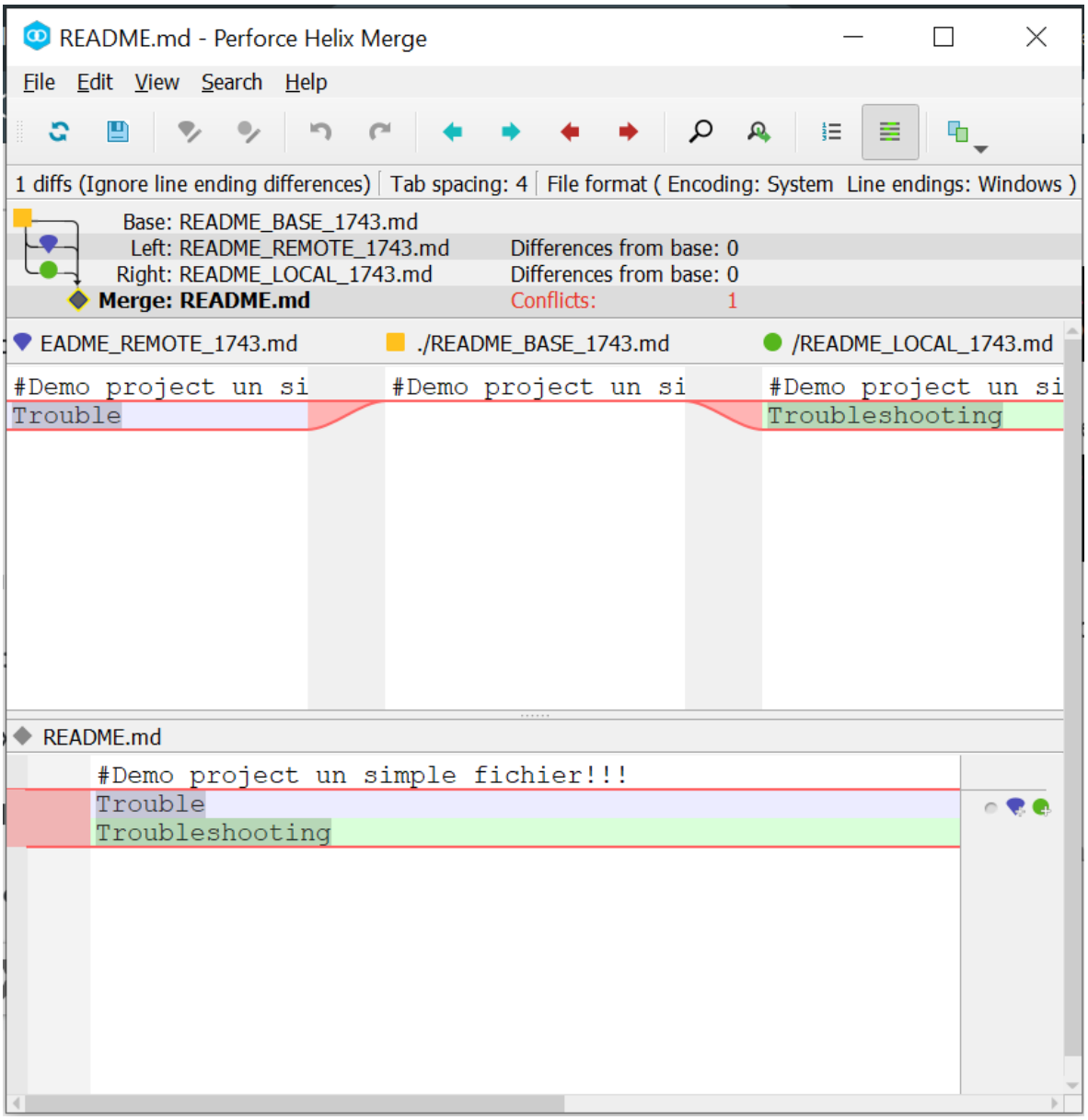
```
youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop
$ cd Projects/Demo/

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master|MERGING)
$ git config --global --list
user.name=majda FANNAN
user.email=fannanmajda123@gmail.com
alias.historique=log --graph --oneline
alias.last=log -1 HEAD

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master|MERGING)
$ git config --global merge.tool p4merge

youcode@DESKTOP-OC51QH5 MINGW64 ~/Desktop/Projects/Demo (master|MERGING)
$ git config --global mergetool.p4merge.path 'C:\Program Files\Perforce\p4merge.exe'
```

On a constaté qu'après la configuration de prompt, si on tape la commande « git mergetool » l'interface de p4merge s'affiche. Une interface qui montre les conflits des deux branches.



8- challenge :

Finalement, après avoir résolu les conflit entre les deux branches, il est convenables de rejeter les fichies indésirables en laissant que les fichiers : readme.md /licence.txt et .gitignore

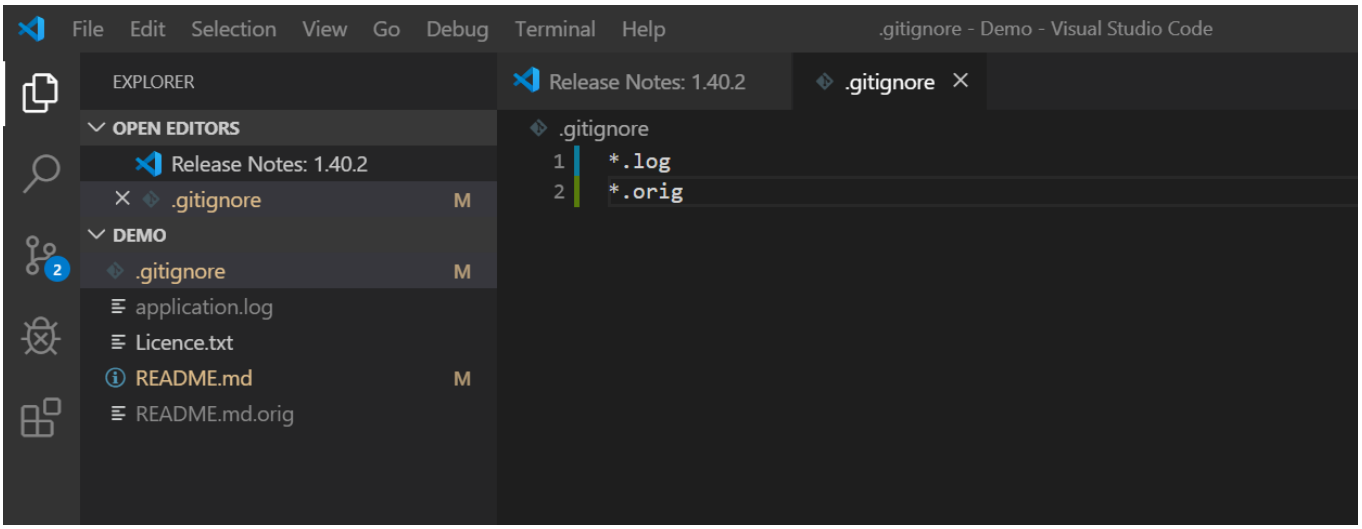


Table de commandes :

Command	Explication
ls	Lister le contenu du dossier actuel
Cd Dossier1	Ouvrir le dossier avec le nom « Dossier1 »
Cd ..	Remonter d'un dossier
Mkdir "New Folder"	"Make Directory" Créer un dossier
Touch « New File.* »	Créer un ficher
Start filename.*	Ouvrir un fichier
Git init	Initializer le dossier actuel
Git add .	Indexer le contenu de dossier
Git commit –m "message"	Sauvgarger une commit avec le commentaire "message"
Git status	Indiquer l'état des fichiers dans le dépôt Git, ainsi que les fichiers non suivie
Vim	Editeur de texte
Git log	Afficher la liste des commits qui ont été effectués
Git push origin master	Mettre à jour les commits effectuée sur la copie local vers le référence
Git reset --hard	Supprime toutes les modifications du dernier commit
Git checkout	Permet de naviguer entre les branches
Git clone	Cloner un fichier ou un dossier en local
Git remote set-url « lien du dossier»	Change le url pour le remote par le lien donnée
Git pull	Mise à jour du dépôt local à partir du/des dépôt(s) distant(s)
Git merge	Incorporation des modifications d'une branche dans la branche courante

git config --global --list	Afiicher la liste de configuration
git mergetool	Affiche l'interface de merge pour la gestion des conflits
git log --all --graph --decorate	Affiche la hiérarchie des commit
gir config –global alia.name " nom de commande "	Donner un alias à une commande