

Basics Overview



- Starting a Project
 - Fresh (no source yet)
 - Existing source locally
 - GitHub project (Fork and clone)
- Basic Workflow (add, commit, push & pull)
- Working with Files (rename, move & delete)
- History and Aliases
- Ignoring Unwanted Files

Créer un environnement neuf pour git (Completely Fresh)

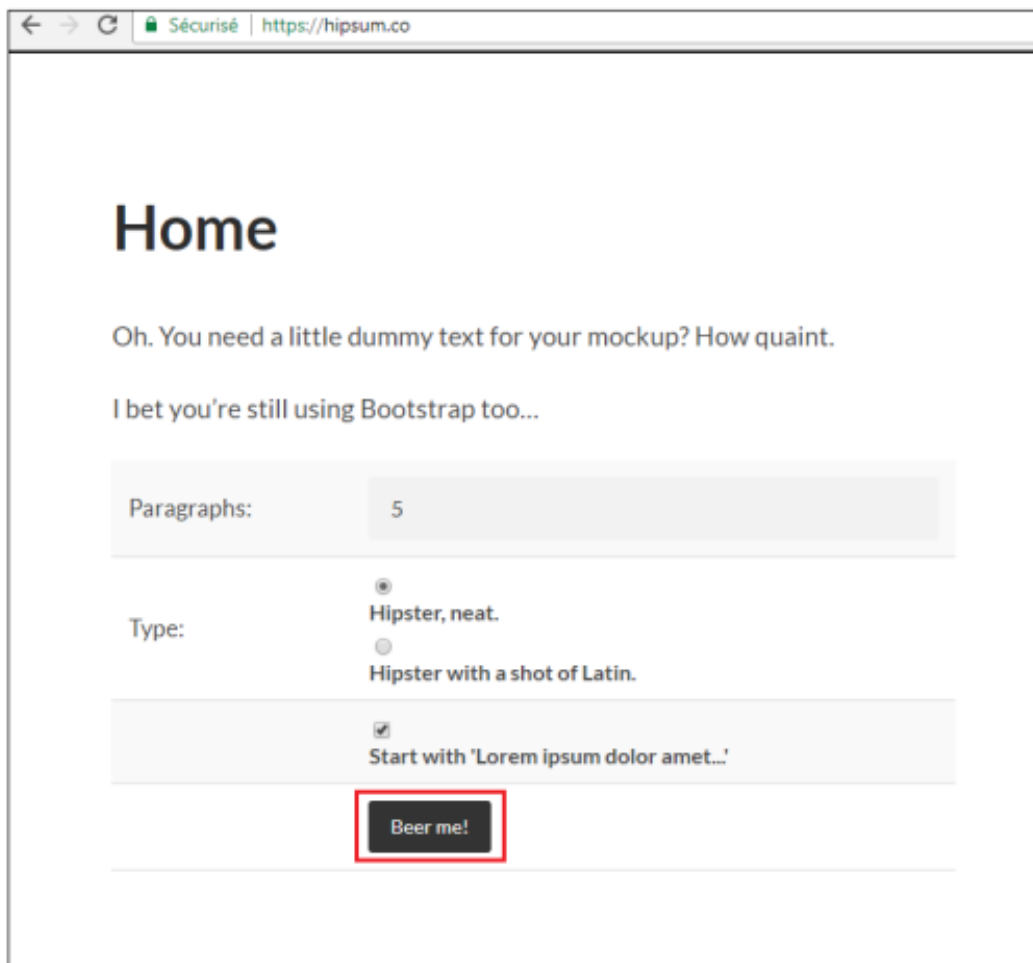
Ouvrir dans windows un terminal **git** dans le répertoire de votre choix ex : **c :\space**

```
1 git init fresh-project
2 ls
3 cd fresh-project
4 ls
5 ls -al
6 cd .git
7 ls
8 cd ..
9 pwd
10 git status
```

Pour récupérer du texte , on peut utiliser n'importe quel générateur de texte.

Par exemple sur <https://hipsum.co/> .

Ouvrir le site dans un navigateur



The screenshot shows a web browser window with the address bar displaying "Sécurisé | https://hipsum.co". The page content includes a heading "Home", a paragraph "Oh. You need a little dummy text for your mockup? How quaint.", and another paragraph "I bet you're still using Bootstrap too...". Below these are form controls: a "Paragraphs:" label with a value of "5", a "Type:" label with two radio button options "Hipster, neat." (selected) and "Hipster with a shot of Latin.", and a checkbox labeled "Start with 'Lorem ipsum dolor amet...'" which is checked. A "Beer me!" button is highlighted with a red rectangle.

Copier par exemple le premier paragraphe et le coller dans le fichier `hipster.txt` que l'on crée avec `npp`.

Lorem ipsum dolor amet flannel put a bird on it plaid deep v scenester. Cornhole meditation snackwave palo santo jean shorts typewriter blue bottle iceland etsy keffiyeh ugh pug crucifix flannel. Four loko swag cloud bread hoodie heirloom literally aesthetic pabst skateboard enamel pin bespoke flexitarian raclette pop-up. Etsy fam butcher bicycle rights. Unicorn lomo drinking vinegar crucifix 90's, tumeric vaporware DIY copper mug chillwave vape yr coloring book semiotics. Health goth everyday carry narwhal, microdosing tote bag pour-over drinking vinegar bitters truffaut scenester. Direct trade etsy hexagon kombucha coloring book schlitz.



```
1 npp hipster.txt
2 git status
3 git add hipster.txt
4 git status
5 git commit
6 git status
7 cd ..
8 pwd
9 ls
10 rm -rf fresh-project/
11
```

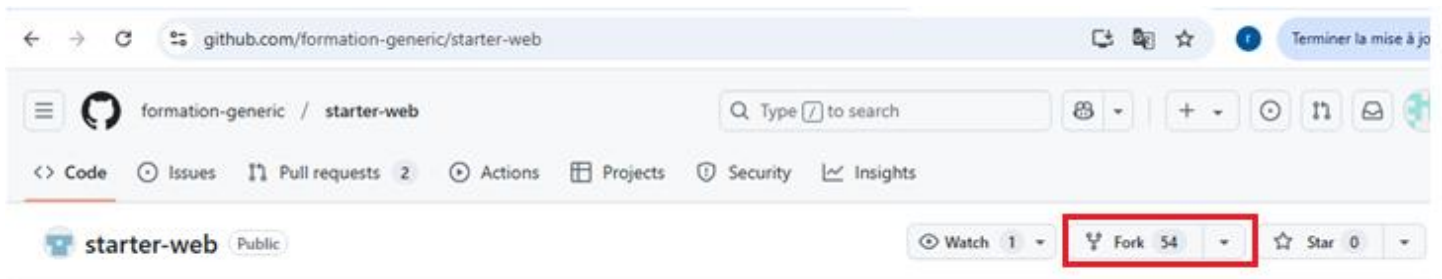
Le fork permet de récupérer une repository d'un autre utilisateur dans votre compte github.
Un fois le projet dans votre repository , vous pouvez y apporter vos modifications , l'importer localement, le réexporter etc...

Pour créer un fork:

Ouvrir une fenêtre de navigation et se connecter à son compte github

Ouvrir une seconde fenêtre de navigation et coller l'url de la repository que vous souhaitez récupérer.

<https://github.com/formation-generic/starter-web.git>




Appuyer sur fork

Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Required fields are marked with an asterisk (*).

Owner *
 /


✓ starter-web is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

☒ Copy the `master` branch only

Contribute back to formation-generic/starter-web by adding your own branch. [Learn more.](#)

 You are creating a fork in your personal account.

Create fork

Appuyer sur Create fork : la repository est créée dans votre compte gitub.

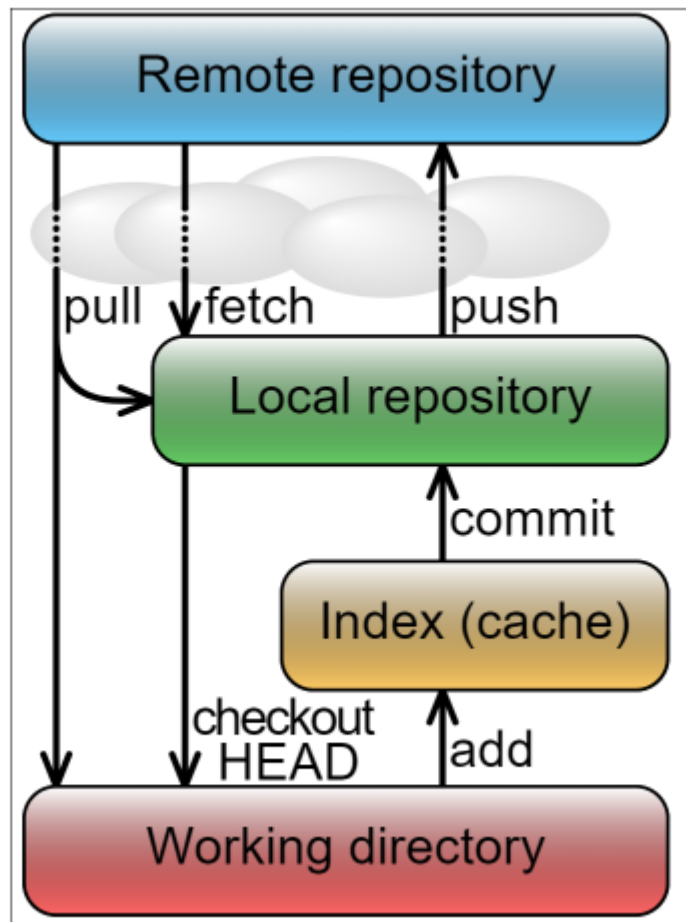
Remote (GitHub)

Cloner la repository starter-web



```
1 pwd
2 cd space
3 git clone https://github.com/votre_id_github/starter-web.git
4 ls
5 cd starter-web
6 ls
7 ls -al
8 cd .git
9 ls
10 cd ..
11 clear
12 git status
13
```

Basic Git Workflow / First Commit





```
1 pwd
2 cd space
3 ls
4 cd starter-web
5 ls
6 git status
7 npp hipster.txt # ajouter un paragraphe à partir de https://fr.lipsum.com/
8 ls
9 git status
10 git add hipster.txt
11 git status
12 git commit # Add message in notepad++
13 git status
14 git pull origin master # Make sure that the origin branch is up to date with master
15 clear
16 git push origin master
17
```

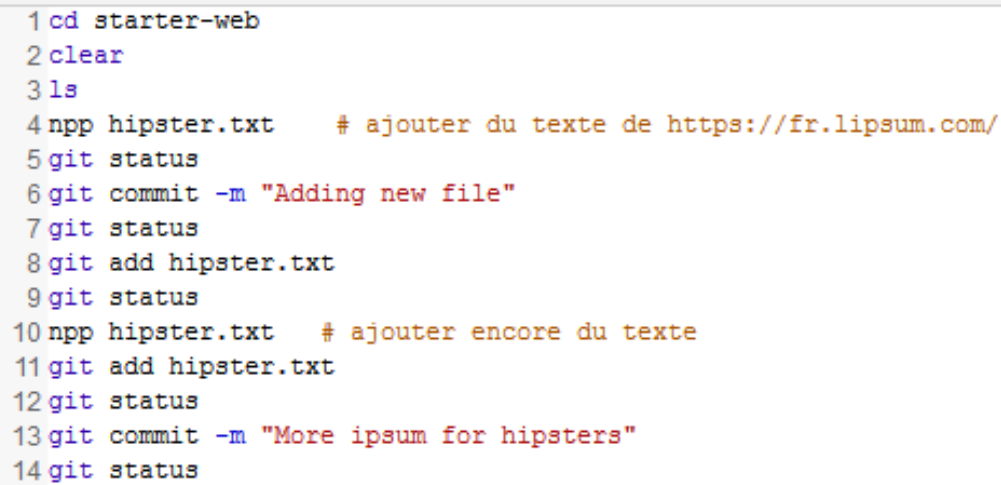
Tracked Files / Express Commit

List not tracked file



```
1 cd starter-web
2 ls
3 git status
4 npp hipster.txt # Ajouter du texte de
5 git commit -am "Adding more ipsum text"
6 git ls-files # all tracked files
7 npp newfile.txt # ajouter du texte de https://fr.lipsum.com/
8 git status
9 git ls-files # the newfile.txt is not listed
10 git add newfile.txt
11 git status
12 git ls-files # newfile.txt is listed
```

Editing Files



```
1 cd starter-web
2 clear
3 ls
4 npp hipster.txt    # ajouter du texte de https://fr.lipsum.com/
5 git status
6 git commit -m "Adding new file"
7 git status
8 git add hipster.txt
9 git status
10 npp hipster.txt   # ajouter encore du texte
11 git add hipster.txt
12 git status
13 git commit -m "More ipsum for hipsters"
14 git status
```

Attention : il y a qq chose qui a été volontairement oublié dans ces lignes. Un indice : avant un commit , il faut toujours placer les modifications dans la staging area.

git add -A : place dans la staging area TOUS les fichiers et répertoires de TOUT l'espace de travail , qu'ils soient déjà suivis ou qu'ils soient nouveaux.

git add . : place dans la staging area TOUS les fichiers du répertoire courant et de ses sous-répertoires à partir duquel la commande est exécutée. Ceci concerne les fichiers et répertoires qu'ils soient déjà suivis ou qu'ils soient nouveaux. Si **git add .** est exécuté à partir du répertoire racine du projet, cela produit un effet équivalent à git add -A

git add -u : place dans la staging area UNIQUEMENT les fichiers modifiés du projet QUI SONT DEJA SUIVIS

Recursive Add

```
1 cd starter-web
2 git status
3 mkdir -p level1/level2/level3 # construire une structure de répertoire
4 ls
5 cd level1
6 pwd
7 npp level1-file.txt # ajouter du texte de https://fr.lipsum.com/
8 ls
9 cd level2
10 npp level2-file.txt # ajouter du texte de https://fr.lipsum.com/
11 ls
12 cd level3
13 npp level3-file.txt # ajouter du texte de https://fr.lipsum.com/
14 ls
15 clear
16 cd ../../..
17 ls
18 git status
19 git add .
20 git status
21 git commit
```

Annuler les modifications apportées dans le contenu de fichiers .

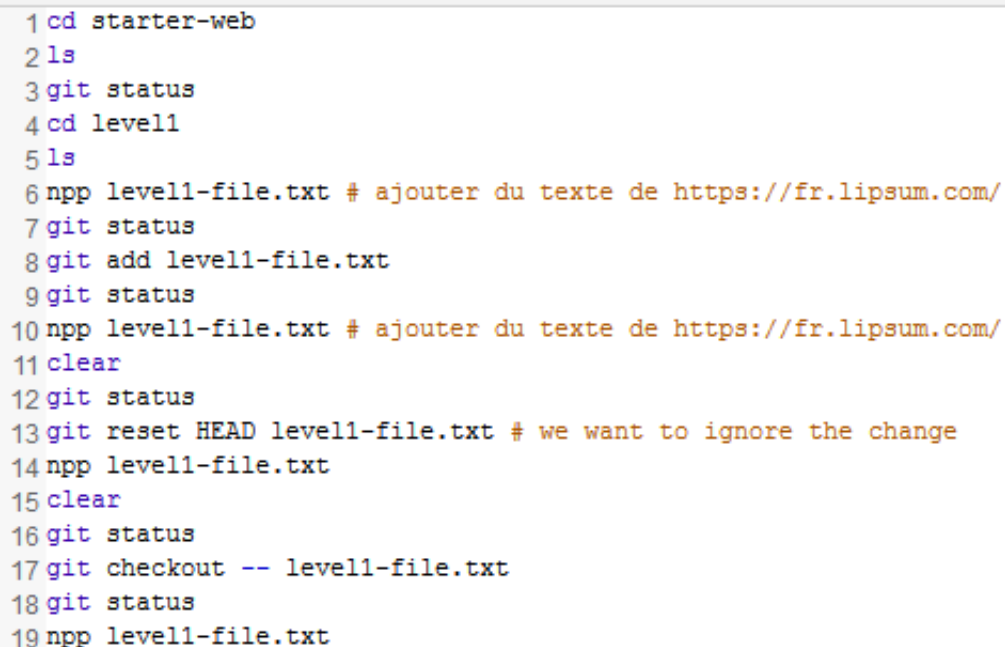
En fonction du cas d'utilisation, plusieurs possibilités : **git reset** ou **git checkout**

git reset --hard pour annuler TOUS les changements sur tout le projet et remettre la working copy dans sa situation où elle était lors du dernier commit.

git reset -- nom_fichier pour retirer de la staging area un fichier ou des fichiers donnés (exemples : * , *.txt , */*.txt pour atteindre tous les sous répertoires, etc..) et le/les remettre dans la working copy dans l'état où il/ils était/étaient lorsqu'il/ils a/ont été placé/placés dans la staging area

git checkout -- nom_fichier pour annuler des modifications en cours sur un fichier ou des fichiers donnés (exemples : * , *.txt , */*.txt pour atteindre tous les sous répertoires, etc..) qui ne sont pas encore dans la staging area et le/les remettre dans la working copy dans l'état où il/ils était/étaient lors du dernier commit

Backing Out Changes



```
1 cd starter-web
2 ls
3 git status
4 cd level1
5 ls
6 npp level1-file.txt # ajouter du texte de https://fr.lipsum.com/
7 git status
8 git add level1-file.txt
9 git status
10 npp level1-file.txt # ajouter du texte de https://fr.lipsum.com/
11 clear
12 git status
13 git reset HEAD level1-file.txt # we want to ignore the change
14 npp level1-file.txt
15 clear
16 git status
17 git checkout -- level1-file.txt
18 git status
19 npp level1-file.txt
```

Renommer un fichier ou répertoire .

Il faut distinguer 2 cas d'utilisation

1- renommer un fichier ou un répertoire par l'Operating System : dans ce cas, git ne suit pas le fichier nouvellement nommé.

2- renommer un fichier ou un répertoire par git avec la commande git mv : git suit le fichier nouvellement nommé.

git mv : renomme un fichier suivi par git

```
git mv nom_fichier nom_fichier_nouveau
```

git mv provoque la suppression de nom_fichier et la création de nom_fichier_nouveau. Il place les 2 opérations dans la staging area en attente du prochain commit.

Tant que git commit n'est pas effectué, il est possible d'annuler la modification.

Pour annuler les effets de git mv :

- Soit git reset --hard

- Soit git reset -- nom_fichier + git checkout -- nom_fichier
git reset -- nom_fichier_nouveau
supprimer à la main nom_fichier_nouveau

Rename

```
1 cd starter-web
2 ls
3 cd level1/level2/level3
4 pwd
5 clear
6 ls
7 git status
8 git mv level3--file.txt level3.txt # renommer à partir de la commande git
9 ls
10 git status
11 git commit -m "renaming level3 file"
12 cd ..
13 clear
14 pwd
15 ls
16 mv level2-file.txt level2.txt # renommer à partir de l'os
17 ls
18 git status
19 git add -A
20 git status
21 git commit
22 clear
```

```
23 ls
24 git mv level2.txt 2.txt
25 ls
26 git status
27 git mv 2.txt level2.txt
28 ls # git considère 2 action: * suppression d'un fichier et création d'un
nouveau
29 git add -A # Recursivly add file and update file that was added, renamed or
deleted
30 git status
31 git commit -m "renaming level2 file"
32 clear
33 ls
34 git mv level2.txt 2.txt
35 ls
36 git mv 2.txt level2.txt # idem à un reset
37 git status
38 git mv level2.txt level3/
39 ls
```

```
40 cd level3
41 ls
42 git status
43 cd ..
44 git status
45 git commit
46 ls
47 cd level3
48 ls
49 mv level2.txt .. # without git
50 ls
51 cd ..
52 pwd
```

```
53 ls
54 git status
55 git add -A
56 git status
57 git commit
58 clear
59 cd ..
60 pwd
61 # file renamed File explorer
62 git status
63 git add .
64 git status
65 git commit # renaming level1 file
```

Suppression de fichier ou de répertoire

git rm : enregistrement dans la staging area de la suppression d'un fichier suivi par git.
Cela fonctionne aussi pour les répertoires

```
git rm nom_fichier  
git rm -r nom_repertoire
```

Tant que git commit n'est pas effectué, il est possible d'annuler la modification.

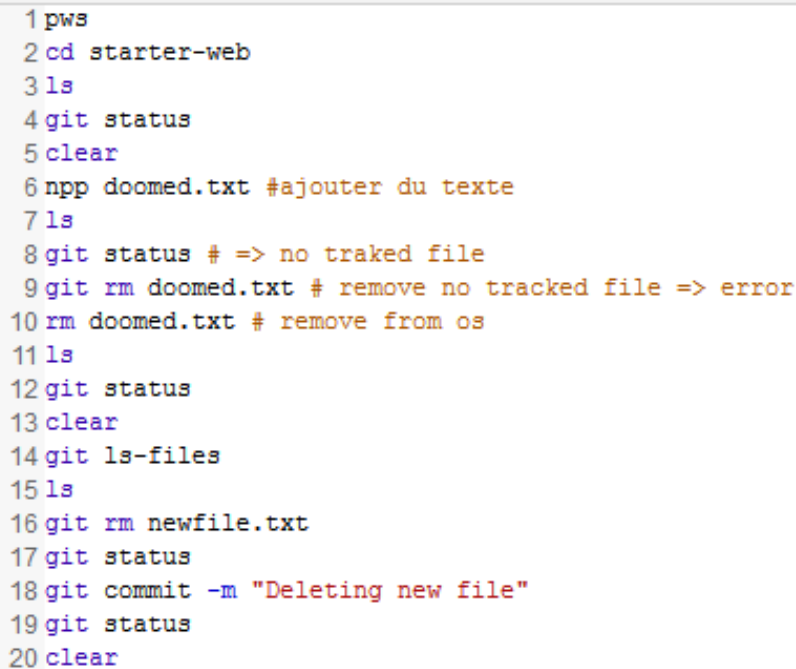
Pour annuler les effets de git rm :

- Soit git reset --hard

- Soit git reset -- nom_fichier + git checkout -- nom_fichier

Pour un répertoire : Soit git reset -- nom_repertoire + git checkout -- nom_repertoire

Deleting Files



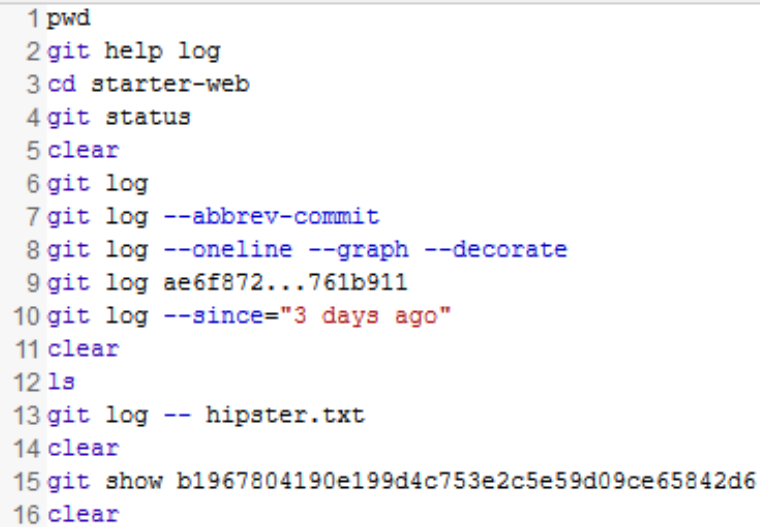
```
1 pws  
2 cd starter-web  
3 ls  
4 git status  
5 clear  
6 npp doomed.txt #ajouter du texte  
7 ls  
8 git status # => no tracked file  
9 git rm doomed.txt # remove no tracked file => error  
10 rm doomed.txt # remove from os  
11 ls  
12 git status  
13 clear  
14 git ls-files  
15 ls  
16 git rm newfile.txt  
17 git status  
18 git commit -m "Deleting new file"  
19 git status  
20 clear
```

```
21 git status
22 git ls-files
23 git rm hipster.txt
24 ls
25 git status
26 git reset HEAD hipster.txt
27 git status
28 git checkout -- hipster.txt
29 ls
30 git status
31 clear
32 ls
33 rm hipster.txt
34 ls
35 git status
36 git add -A
37 git status
38 git commit -m "deleting hipeter.txt"
39 git status
40 clear

41 ls
42 rm -rf level1
43 ls
44 git status
45 git add -A
46 git status
47 git commit -m "deleting level1 and all children"
48 git status
```

git log : visualise l'historique

History



```
1 pwd
2 git help log
3 cd starter-web
4 git status
5 clear
6 git log
7 git log --abbrev-commit
8 git log --oneline --graph --decorate
9 git log ae6f872...761b911
10 git log --since="3 days ago"
11 clear
12 ls
13 git log -- hipster.txt
14 clear
15 git show b1967804190e199d4c753e2c5e59d09ce65842d6
16 clear
```


alias : création d'un raccourci pour exécuter des commandes

Git Alias



```
1 pwd
2 cd starter-web
3 ls
4 git status
5 clear
6 git log --all --graph --decorate --oneline
7 git hist
8 git config --global alias.hist "log --all --graph --decorate --oneline"
9 git hist
10 clear
11 npp ~/.gitconfig # verification
```

.gitignore : exclure des fichiers ou des répertoires du suivi de git

Ignoring Unwanted Files and Folders

```
1 pwd
2 cd starter-web
3 ls
4 git status
5 ls -al
6 npp .gitignore
7 ls -al
8 git status
9 git add .gitignore
10 git status
11 git commit
12 git status
13 ls
14 npp access.log #ajouter du texte
15 ls
16 git status
17 npp .gitignore # ajouter fichier à exclure
18 clear
19 git status
20 mkdir log

21 mv access.log log
22 ls
23 cd log
24 ll
25 cp access.log access.2014-11-04
26 ll
27 cd ..
28 git status
29 npp .gitignore # exclure le répertoire i.e log/
30 git status
31 git commit -am "Excluding log file directory"
32 git status
```