

# LA GÉNÉRATION TRAVAILLEUR DU SAVOIR

Créer, éditer et partager des fichiers d'ordinateurs :

- textes
- images
- vidéos
- etc

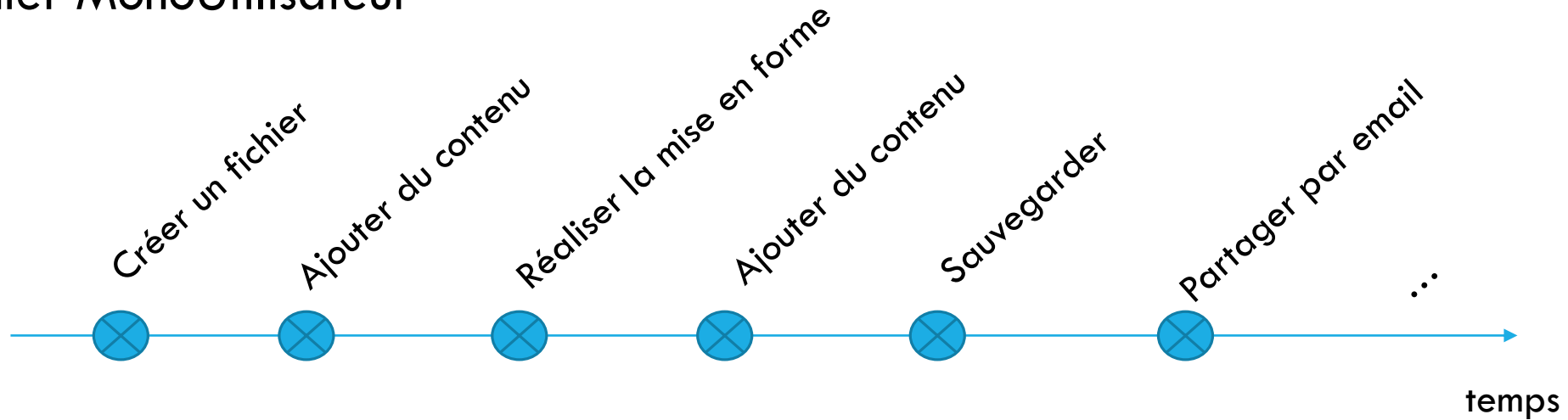


La grande partie de notre travail consiste à :

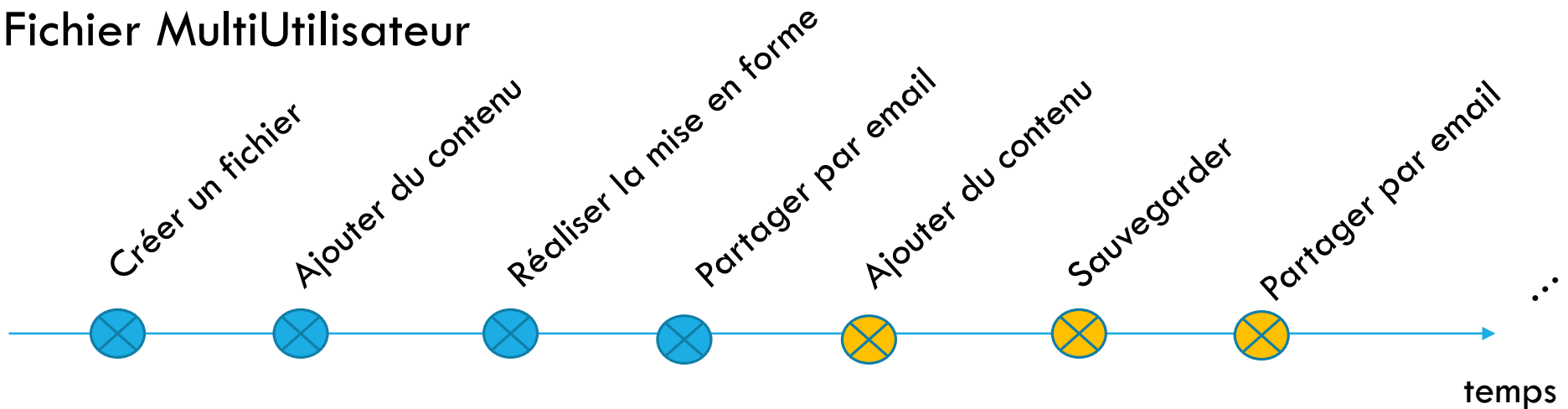
- Créer des fichiers
- Sauvegarder des fichiers
- Editer des fichiers
- Sauvegarder à nouveau ces fichiers
- Partager ces fichiers par email / Cloud
- Relire ces fichiers
- etc

# CYCLE DE VIE D'UN FICHIER

## Fichier MonoUtilisateur



## Fichier MultiUtilisateur



# EXEMPLE DE DOSSIER AVEC DES VERSIONS FINALES

Nom	Modifié le	Type	Taille
mode d'emploi	28/08/2015 22:35	Dossier de fichiers	
150520 déploiement KDS.docx	20/05/2015 16:19	Document Micros...	25 Ko
150520 mail présentation.docx	20/05/2015 15:51	Document Micros...	22 Ko
150706compteRendu.docx	06/07/2015 18:31	Document Micros...	30 Ko
ASSISTANTES login et mdp.xlsx	16/06/2015 17:35	Feuille de calcul ...	24 Ko
ASSISTANTES.xlsx	16/06/2015 17:04	Feuille de calcul ...	22 Ko
Leaflet 1 1ère connexion_29-07-13.ppt	20/05/2015 16:00	Présentation Micr...	657 Ko
Manuel ADMIN Suite FBF.pdf	16/06/2015 15:52	Fichier PDF	927 Ko
Manuel utilisateur KDS Corporate Reserv...	05/06/2015 16:37	Présentation Micr...	3 135 Ko
mode d'emploi KDS v1.docx	08/06/2015 17:59	Document Micros...	1 454 Ko
mode d'emploi KDS v2.docx	09/06/2015 13:18	Document Micros...	1 483 Ko
mode d'emploi KDS v4.docx	18/06/2015 15:45	Document Micros...	1 481 Ko
mode d'emploi KDS v5.docx	18/06/2015 15:51	Document Micros...	1 453 Ko
point déploiement appli via la DSL.xps	16/06/2015 16:25	Document XPS	339 Ko

Uniquement la dernière version  
disponible

versionning réalisé à la main

# AVOIR UN SUIVI DES FICHIERS POUR OPTIMISER CE PROCESSUS

Pour chaque document, nous voulons un outil qui va permettre de savoir :

- **Quand** le fichier a été modifié ?
- **Qu'est ce qui** a été changé ?
- **Pourquoi** il a été modifié ?
- **Qui** l'a modifié (dans le cas d'un fichier MultiUtilisateur) ?

# GIT & GITHUB

Des outils permettant de :

- Suivre les versions des documents
- Garder un historique des changements
- Favoriser et gérer le travail en équipe

Pour les documents de type « code source »





# INSTALLATION ET CONFIGURATION GIT



# INSTALLATION GIT

Télécharger et installer Git sur votre ordinateur :

<https://git-scm.com/downloads>



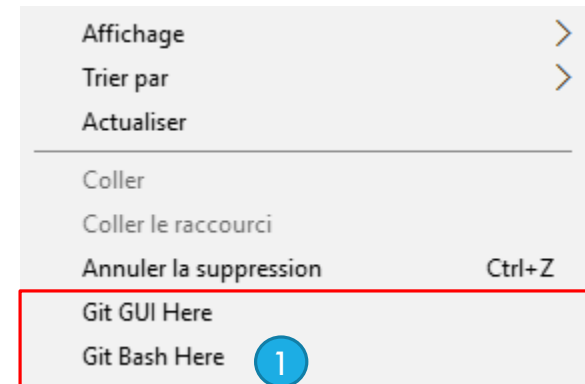
# VÉRIFIER QUE GIT EST BIEN INSTALLÉ

Faire un clique droit avec votre souris :  
Le menu contextuel du clique droit doit afficher désormais deux nouveaux choix :

- git GUI Here
- git Bash Here

Cliquer sur « Git Bash Here »

Le terminal dédié à git s'affiche





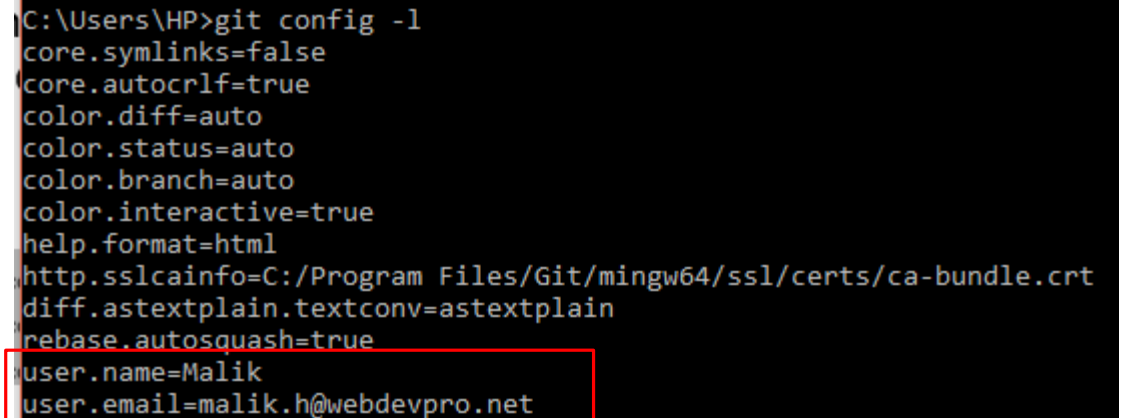
# DÉFINIR VOTRE IDENTITÉ SUR GIT

Dans le terminal, saisir les 3 lignes suivantes:

1. `git config --global user.name "prenom nom"`
2. `git config --global user.email "prenom.nom@email.com"`
3. `git config -l`

A la fin de chaque ligne utiliser la touche Enter pour valider la saisie

1. Cette première manipulation va permettre à git d'attacher votre nom
2. Si tout est bon, vous pouvez fermer le terminal
  - Soit en saisissant `exit` puis Enter
  - Soit en cliquant la petite croix en haut à droite via la souris

A screenshot of a Windows command prompt terminal window. The prompt is 'C:\Users\HP>'. The command 'git config -l' has been executed, and the output is displayed. The output lists various git configuration settings. The last two lines, 'user.name=Malik' and 'user.email=malik.h@webdevpro.net', are highlighted with a red rectangular box.

```
C:\Users\HP>git config -l
core.symlinks=false
core.autocrlf=true
color.diff=auto
color.status=auto
color.branch=auto
color.interactive=true
help.format=html
http.sslcainfo=C:/Program Files/Git/mingw64/ssl/certs/ca-bundle.crt
diff.astextplain.textconv=astextplain
rebase.autosquash=true
user.name=Malik
user.email=malik.h@webdevpro.net
```



# COMMANDE DE BASE GIT



# CRÉER UN NOUVEAU PROJET MONOUTILISATEUR

1. Pour l'instant, nous allons travailler avec Git (nous verrons par la suite GitHub)
2. Créer un nouveau dossier sur le Bureau de votre ordinateur intitulé « hello »
3. Se place dedans via l'Explorer
4. Faire un clique droit et choisir « Git bash here »



```
HP@pc MINGW64 ~/Desktop/hello  
$ |
```

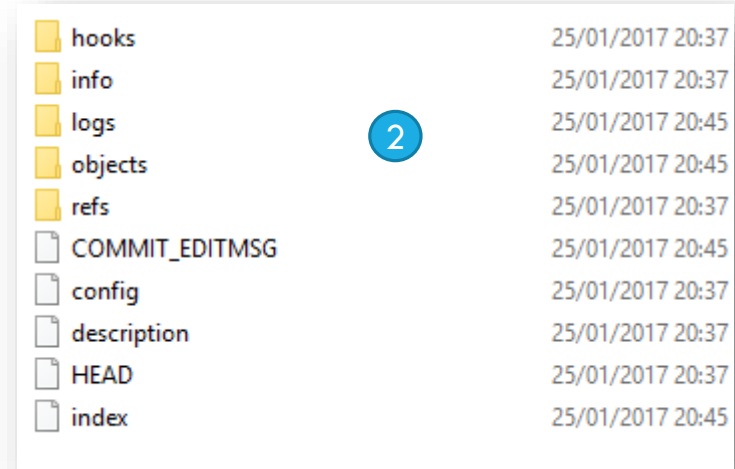
# INITIALISER LE PROJET AVEC GIT

Dans le terminal saisir la commande suivante :

- **git init**

1. git va créer un dossier masqué .git dans le dossier en cours
2. C'est dans ce fichier caché que git l'activité du dossier aussi appelé dépôt (repository en anglais)
  - ajouter \.git à la fin de la barre d'adresse de votre Explorateur pour voir le contenu

```
HP@pc MINGW64 ~/Desktop/hello 1
$ git init
Initialized empty Git repository in C:/Users/HP/Desktop/hello/.git/
```



# VÉRIFIER LE STATUS DU PROJET

Dans le terminal saisir la commande suivante :

- `git status`

Cette commande donne des informations sur l'Etat de suivi du projet

1. Nous sommes sur la branche principale : la branche **master**
2. Nous sommes sur le commit initial
3. Information pour réaliser le suivi : ici il n'y a rien à faire

```
HP@pc MINGW64 ~/Desktop/hello (master)
$ git status
On branch master
Initial commit
nothing to commit (create/copy files and use "git add" to track)
```

# AJOUTER UN FICHIER DANS LE PROJET

Dans le dossier « Hello » créer un nouveau fichier « index.html »

Contenant le texte suivant :

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>hello</title>
5      <meta charset="UTF-8" />
6  </head>
7  <body>
8      Bonjour tout le monde
9  </body>
10 </html>
```

# VÉRIFIER À NOUVEAU LE STATUS

Dans le terminal saisir la commande suivante :

- **git status**

Par rapport à la saisie précédente de `git status`, on constate :

- git a bien repéré le nouveau fichier créé : `index.html`
- Ce nouveau fichier n'est pas suivi : **untracked**

```
HP@pc MINGW64 ~/Desktop/hello (master)
$ git status
On branch master

Initial commit

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

nothing added to commit but untracked files present (use "git add" to track)
```

# LANCER LE SUIVI DES FICHIERS

Dans le terminal saisir les commandes suivantes :

- **git add index.html**
- **git status**
- **git commit -m "lancement du projet hello"**

1. La première commande va permettre de faire une « photo instantanée » du contenu textuel des deux fichiers
2. La dernière commande va permettre d'enregistrer la « photo instantanée » dans la mémoire de git (dans le dossier masqué .git) avec un message du développeur pour expliquer ce qu'il a fait

```
HP@pc MINGW64 ~/Desktop/hello (master)
$ git add index.html 1

HP@pc MINGW64 ~/Desktop/hello (master)
$ git status 2
On branch master

Initial commit

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

    new file:   index.html

HP@pc MINGW64 ~/Desktop/hello (master)
$ git commit -m "lancement du projet hello" 3
[master (root-commit) 0af9b84] lancement du projet hello
1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.html
```



# LA DIFFÉRENCE ENTRE UN ADD ET UN COMMIT

Dans le dossier « Hello » créer un nouveau fichier « contact.html » vide

Dans le terminal saisir les commandes suivantes :

1. **git status**
2. **git add contact.html**
3. **git status**
4. **git commit -m "ajout page contact"**
5. **git status**

```
C:\Users\HP\Desktop\hello>git status 1
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

    contact.html

nothing added to commit but untracked files present (use "git add" to track)
C:\Users\HP\Desktop\hello>git add contact.html 2
C:\Users\HP\Desktop\hello>git status 3
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

    new file:   contact.html

C:\Users\HP\Desktop\hello>git commit -m "ajout page contact" 4
[master 5f2f20d] ajout page contact
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 contact.html
C:\Users\HP\Desktop\hello>git status 5
On branch master
nothing to commit, working directory clean
```

# MODIFIER UN FICHER EXISTANT ET L'ENREGISTRER

Dans le dossier « Hello » modifier le texte dans la balise <title> par Bonjour du fichier « index.html »

Dans le terminal saisir les commandes suivantes :

1. **git status**
2. **git add index.html**
3. **git commit -m "modif index"**
4. **git status**

```
C:\Users\HP\Desktop\hello>git status 1
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\HP\Desktop\hello>git add index.html 2

C:\Users\HP\Desktop\hello>git commit -m "modif index" 3
[master 9f812cb] modif index
1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\HP\Desktop\hello>git status 4
On branch master
nothing to commit, working directory clean
```

# CAS PRATIQUE : AJOUTER DE LA MISE EN FORME CSS DANS LE FICHIER INDEX.HTML

Modifier le fichier index.html en lui ajoutant des règles css

Dans le terminal, lancer les commandes git qui vont enregistrer les modifications réalisées

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Bonjour</title>
5      <meta charset="UTF-8" />
6      <style>
7          body{
8              font-size :20px;
9              font-color:blue;
10             font-weight: bold;
11         }
12     </style>
13 </head>
14 <body>
15     Bonjour tout le monde
16 </body>
17 </html>
```

# CAS PRATIQUE : RÉPONSE

Dans le terminal saisir les commandes suivantes :

1. **git add index.html**
2. **git commit -m "ajout css"**
3. **git status**

Remarque : la dernière commande est facultative, elle permet d'avoir une confirmation que tout est enregistré

Par contre deux premières sont obligatoires :

- Faire une photo instantanée du contenu
- Puis l'enregistrer dans git qui va me permettre de suivre son évolution

```
C:\Users\HP\Desktop\hello>git add index.html 1
C:\Users\HP\Desktop\hello>git commit -m "ajout css" 2
[master f3422a3] ajout css
1 file changed, 7 insertions(+)
C:\Users\HP\Desktop\hello>git status 3
On branch master
nothing to commit, working directory clean
```

# VOIR LES EVOLUTIONS D'UN FICHER

Dans le dossier « Hello » ajouter la balise `<h1>` autour du texte dans du body

Dans le terminal saisir les commandes suivantes :

1. **git status**

2. **git diff**

Toutes les modifications de tous les fichiers suivis sont en vert avec un + devant

Remarque : la commande **git diff** peut aussi être utilisée suivi d'un **nom\_fichier** ce qui va permettre d'avoir les différences avec un fichier par particulier

1. `git diff <nom_fichier>`

```
C:\Users\HP\Desktop\hello>git status 1
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\HP\Desktop\hello>git diff 2
diff --git a/index.html b/index.html
index 72e9446..64cb9af 100644
--- a/index.html
+++ b/index.html
@@ -12,6 +12,6 @@
     </style>
</head>
<body>
-    Bonjour tout le monde
+    <h1>Bonjour tout le monde</h1>
</body>
</html>
\ No newline at end of file
```

# JE SUIS D'ACCORD AVEC LES MODIFICATIONS RÉALISÉES

Dans le terminal saisir les commandes suivantes :

1. **git add index.html**
2. **git diff**
3. **git commit -m "ajout h1"**

La deuxième commande est facultative mais il faut remarquer que suite à un add le diff ne retourne plus rien

```
C:\Users\HP\Desktop\hello>git add index.html 1
C:\Users\HP\Desktop\hello>git diff 2
C:\Users\HP\Desktop\hello>git commit -m "ajout h1" 3
[master 6f687b8] ajout h1
1 file changed, 1 insertion(+), 1 deletion(-)
```

# VOIR TOUTES LES MODIFICATIONS RÉALISÉES

Dans le terminal saisir la commande suivante:

## 1. git log

Cette nouvelle commande permet d'afficher tous les commits réalisés du plus récent au plus ancien

Puis chaque commit il y a :

- A. un id
- B. L'auteur qui a réalisé le commit
- C. La date où le commit a été réalisé
- D. Le message du commit

**Ces informations sont appelées les méta données du commit**



```
C:\Users\HP\Desktop\hello>git log
commit 6f687b840ec3b213a24b95ba165bb8f2f6868a97
Author: Malik <malik.h@webdevpro.net>
Date:   Mon Jan 23 13:58:55 2017 +0100

    ajout h1

commit f3422a3d0098b5c4b2a3f0b2928d61eecb845143
Author: Malik <malik.h@webdevpro.net>
Date:   Mon Jan 23 13:42:28 2017 +0100

    ajout css

commit 9f812cbc7ec622c85b997856d4496d369d155d81
Author: Malik <malik.h@webdevpro.net>
Date:   Mon Jan 23 13:31:23 2017 +0100

    modif index

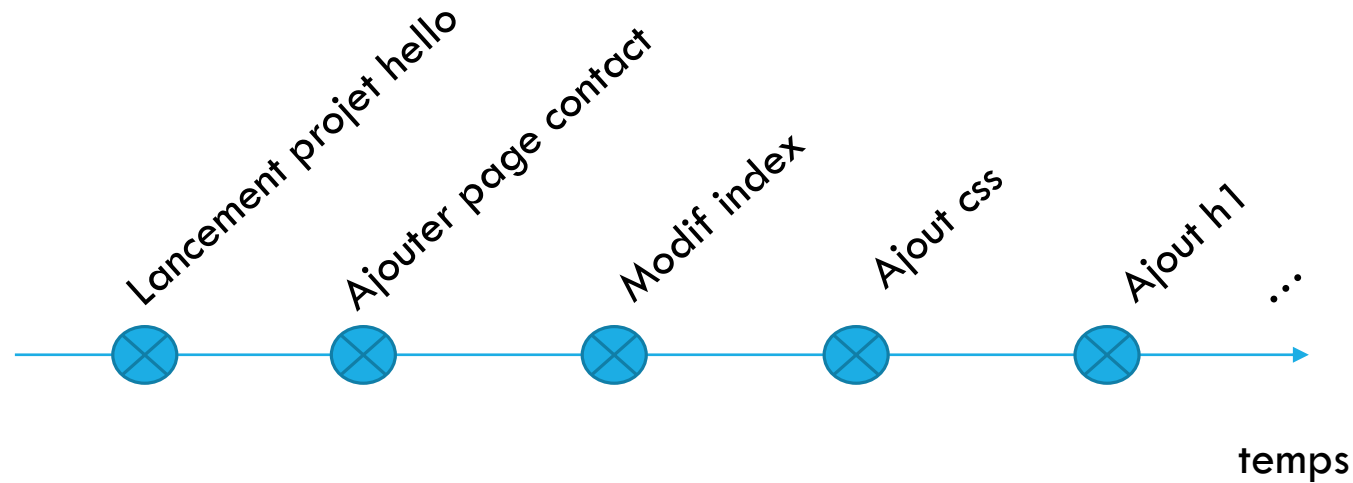
commit 5f2f20dfd2e30129ee5ac34cead6a54a0beda05a
Author: Malik <malik.h@webdevpro.net>
Date:   Mon Jan 23 13:16:55 2017 +0100

    ajout page contact

commit 6b81fb17bfc93fffe611fd3b888210d0ba32b70
Author: Malik <malik.h@webdevpro.net>
Date:   Mon Jan 23 13:05:02 2017 +0100

    lancement du projet hello
```

# GIT LOG SOUS FORME GRAPHIQUE



Chaque point correspond a un commit

Un commit contient un ou plusieurs fichiers qui ont été add au commit

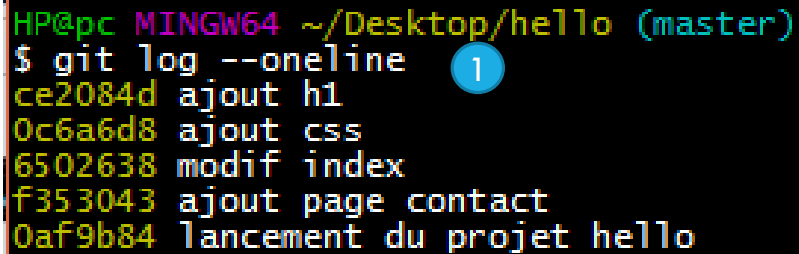


# GIT LOG SOUS FORME COMPACT

Dans le terminal saisir la commande suivante:

## 1. `git log --oneline`

Cette commande donne une version compacte de tous les commit réalisés ainsi que de leur id

A terminal window with a black background and colorful text. The prompt is 'HP@pc MINGW64 ~/Desktop/hello (master)'. The command '\$ git log --oneline' is entered, followed by a blue circle with the number '1'. The output shows five commit hashes and their messages: 'ce2084d ajout h1', '0c6a6d8 ajout css', '6502638 modif index', 'f353043 ajout page contact', and '0af9b84 lancement du projet hello'.

```
HP@pc MINGW64 ~/Desktop/hello (master)
$ git log --oneline 1
ce2084d ajout h1
0c6a6d8 ajout css
6502638 modif index
f353043 ajout page contact
0af9b84 lancement du projet hello
```

# HISTOIRE D'UN FICHIER PAR COMMIT

Dans le terminal saisir la commande suivante:

## 1. `git log --oneline -p index.html`

Cette commande donne l'historique du fichier `index.html` du commit le plus récent au plus ancien

Pour sortir sans avoir à faire défiler tout l'historique faire : `Ctrl + C`

```
HP@pc MINGW64 ~/Desktop/hello (master)
$ git log --oneline -p index.html
ce2084d ajout h1
diff --git a/index.html b/index.html
index 114aa65..12fff6e 100644
--- a/index.html
+++ b/index.html
@@ -10,7 +10,7 @@
     </style>
  </head>
  <body>
-
+     <h1>Hello</h1>
+     <script type="text/javascript">js</script>
  </body>
</html>
\ No newline at end of file
0c6a6d8 ajout css
diff --git a/index.html b/index.html
index b4fe223..114aa65 100644
--- a/index.html
+++ b/index.html
@@ -3,6 +3,11 @@
  <head>
+     <title>hello</title>
+     <meta charset="UTF-8" />
+     <style>
+         h1{
+             color:red;
+         }
+     </style>
  </head>
  <body>
```

# LISTE DES COMMANDES DE BASE

1. `git init` : lancer le suivi d'un dossier par git (à ne saisir que lors du démarrage d'un projet)
2. `git add <noms fichiers>` : faire une photo instantanée des fichiers
3. `git commit -m "message"` : enregistrer la photo instantanée
4. `git status` : voir l'état du dossier par rapport au dernier commit
5. `git diff <noms fichiers>` : voir les différences sur un fichier modifié et non « addé »
6. `git log` : liste des différents commits réalisés
7. `git log --oneline` : liste compacte des différents commit réalisés
8. `git log --oneline -p <nom fichier>` : historique de toutes les modifications liées à un fichier par commit

Remarque importante : toutes les opérations que nous avons réalisées sont faites en local sans connexion à internet



# **GIT ADD ET LES WILDCARDS**



# EXEMPLE D'UTILISATION DE GIT ADD AVEC WILDCARDS

1. `git add *` #tout ajouter dans le snapshot quelquesoit de manière récursive
2. `git add --all` # équivalent de `git add *`
3. `git add -A` # équivalent de `git add *`
4. `git add *.php` #tous les fichiers finissant par php de manière récursive
5. `git add httpdocs/` # tous les fichiers se trouvant dans le dossier httpdocs ainsi que les fichiers et dossiers créés ou modifiés dans les dossiers enfants de httpdocs
6. `git add *.html | *.php | *.css | *.js | *.xml` # tous les fichiers finissant avec ces 5 extensions récursivement
7. `git add index.html` # ajouter le fichier index.html situé à la racine du projet mais pas le fichier `httpdocs/index.html`



# LES DIFFÉRENTS ÉTATS D'UN FICHIER DANS GIT

# LES DIFFÉRENTS ÉTATS D'UN FICHIER INDEX.HTML

