



GIT & GITHUB

malik.h@webdevpro.net

LA GÉNÉRATION TRAVAILLEUR DU SAVOIR

Créer, éditer et partager des fichiers d'ordinateurs :

- textes
- images
- vidéos
- etc

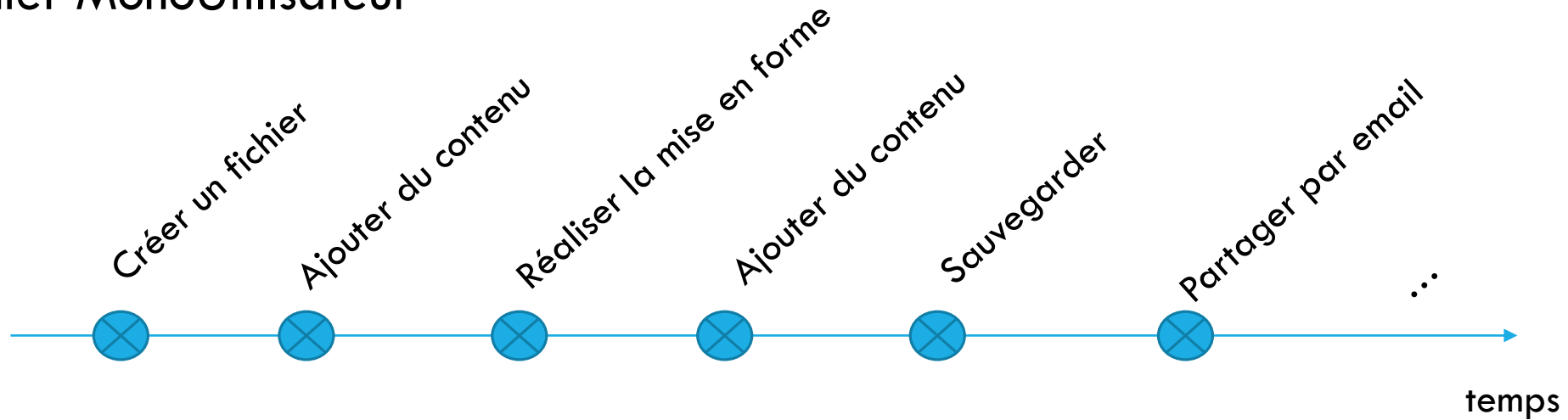


La grande partie de notre travail consiste à :

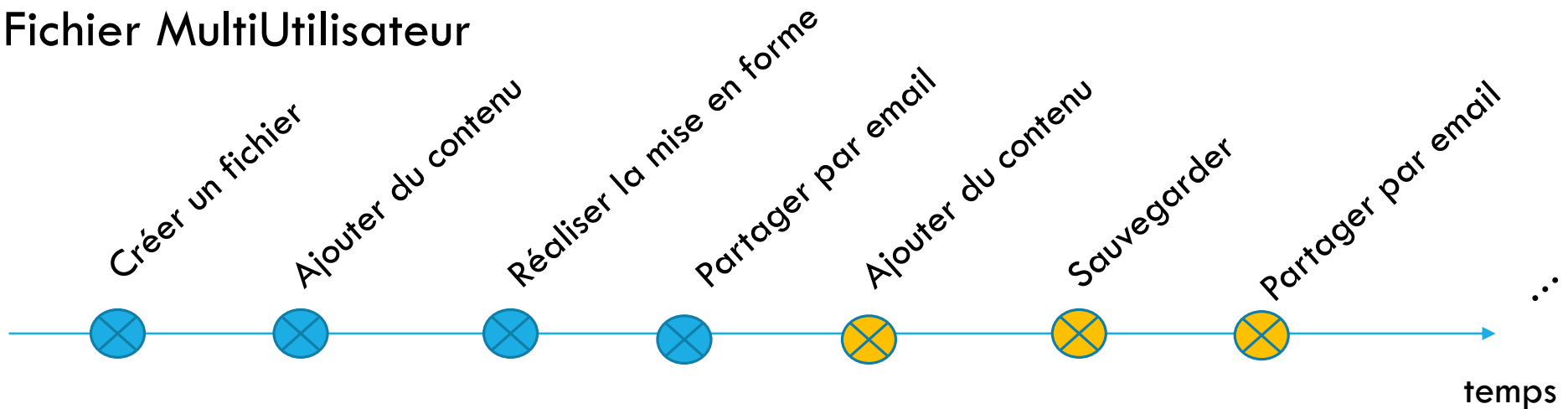
- Créer des fichiers
- Sauvegarder des fichiers
- Editer des fichiers
- Sauvegarder à nouveau ces fichiers
- Partager ces fichiers par email / Cloud
- Relire ces fichiers
- etc

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>



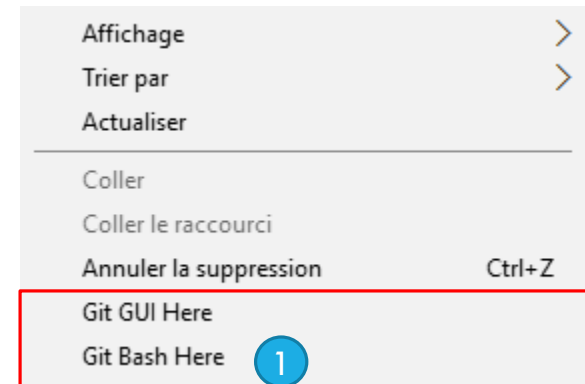
VÉRIFIER QUE GIT EST BIEN INSTALLÉ

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



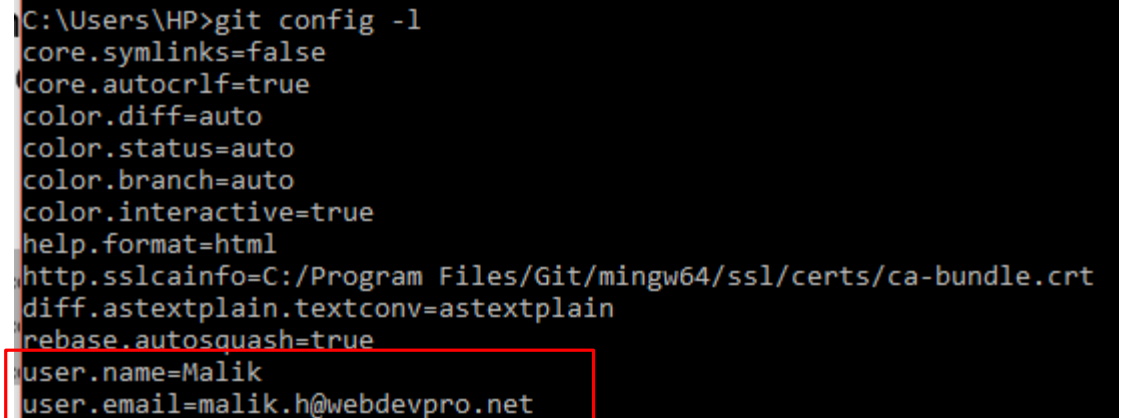
DÉFINIR VOTRE IDENTITÉ SUR GIT

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



```
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/
```

2

hooks	25/01/2017 20:37
info	25/01/2017 20:37
logs	25/01/2017 20:45
objects	25/01/2017 20:45
refs	25/01/2017 20:37
COMMIT_EDITMSG	25/01/2017 20:45
config	25/01/2017 20:37
description	25/01/2017 20:37
HEAD	25/01/2017 20:37
index	25/01/2017 20:45

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

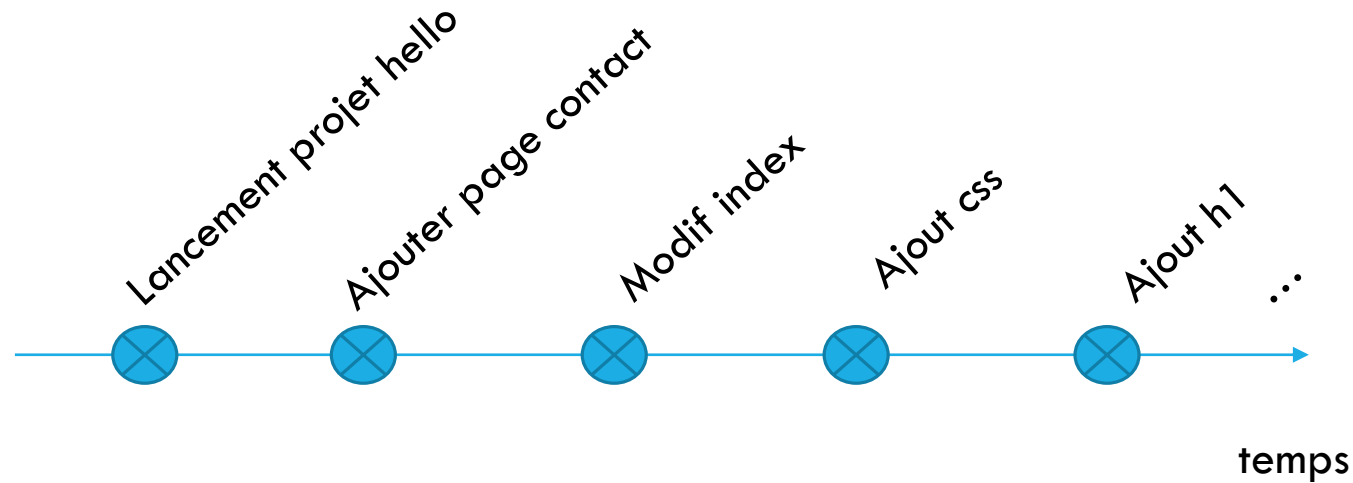
commit 5f2f20dfd2e30129ee5ac34cead6a54a0beda05a
Author: Malik <malik.h@webdevpro.net>
Date:   Mon Jan 23 13:16:55 2017 +0100

    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

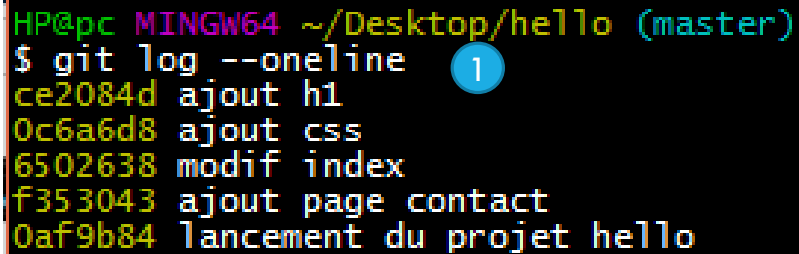
Un commit contient un ou plusieurs fichiers qui ont été add au commit

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' has been entered. The output shows six commit hashes followed by their descriptions: 'ce2084d ajout h1', '0c6a6d8 ajout css', '6502638 modif index', 'f353043 ajout page contact', and '0af9b84 lancement du projet hello'. A blue circle with the number '1' is positioned next to the command.

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

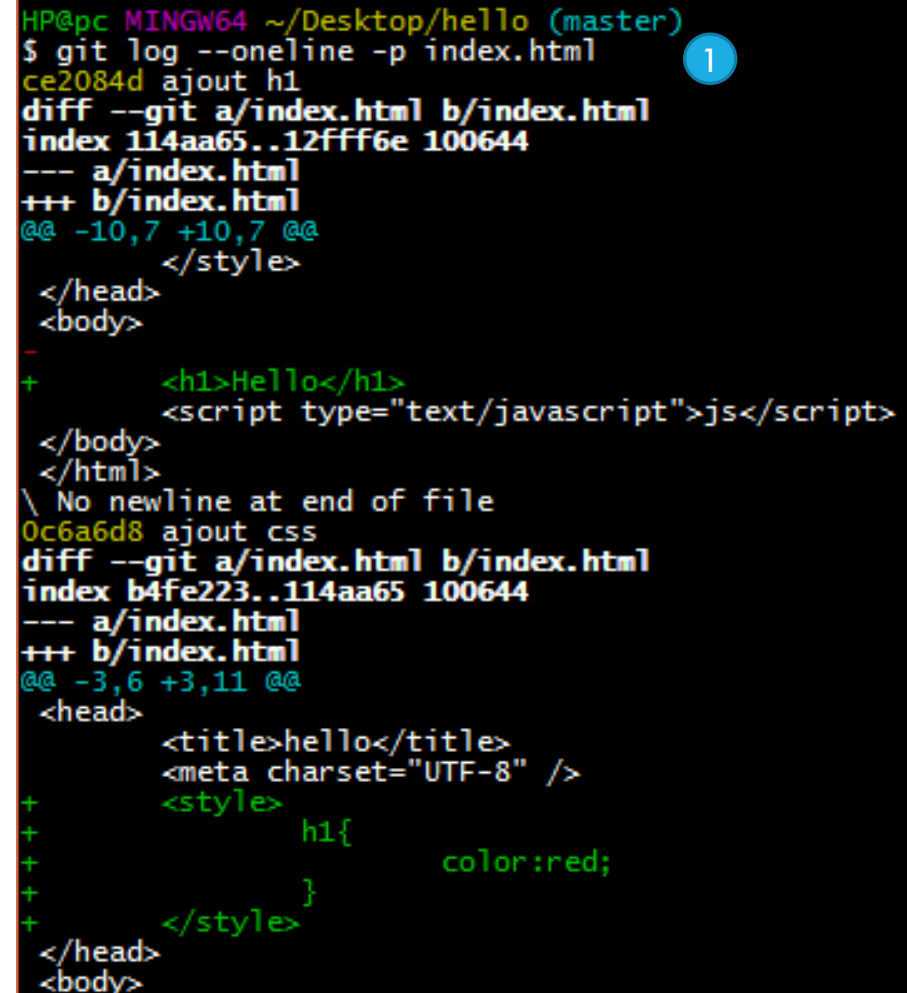
HISTOIRE D'UN FICHIER PAR COMMIT

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`

A terminal window with a black background and white text. The prompt is 'HP@pc MINGW64 ~/Desktop/hello (master)'. The command entered is '\$ git log --oneline -p index.html'. The output shows two commits. The first commit, 'ce2084d ajout h1', shows a diff between 'a/index.html' and 'b/index.html' with a line added: '<h1>Hello</h1>'. The second commit, '0c6a6d8 ajout css', shows a diff between 'a/index.html' and 'b/index.html' with a style block added to the head. A blue circle with the number '1' is next to the command line.

```
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

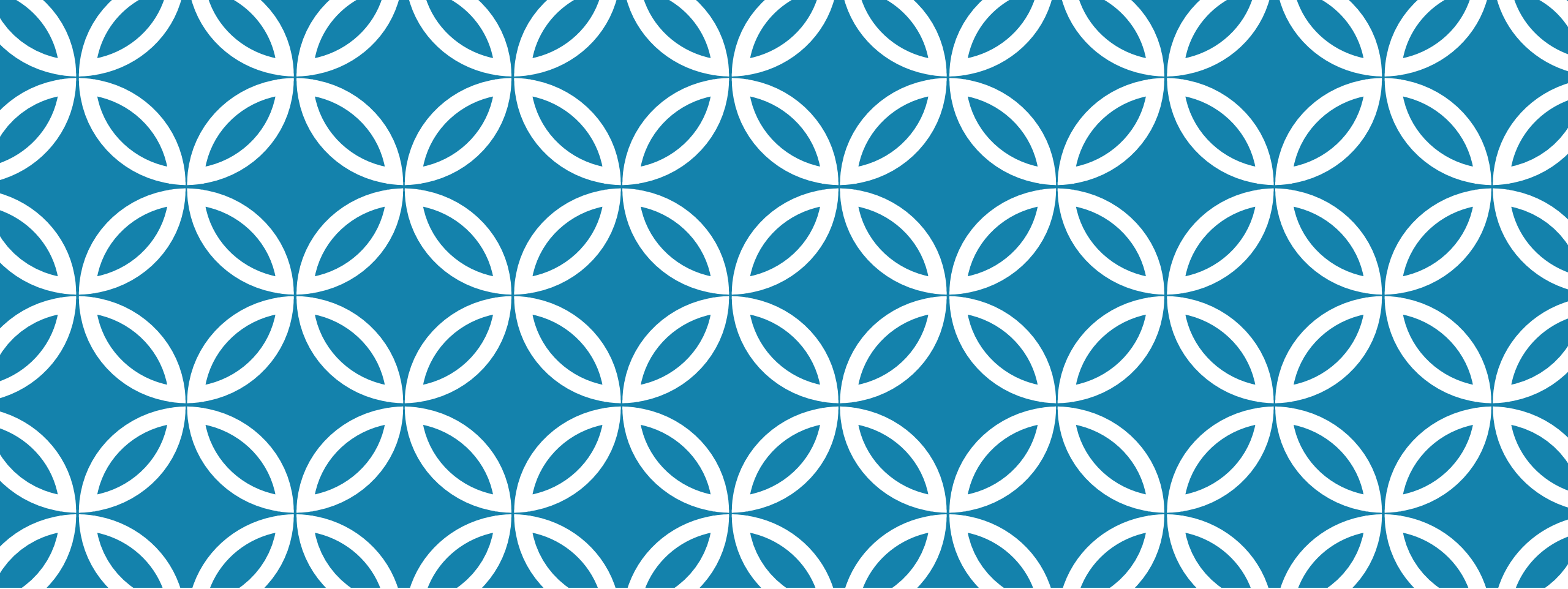


GIT ADD ET LES WILDCARDS



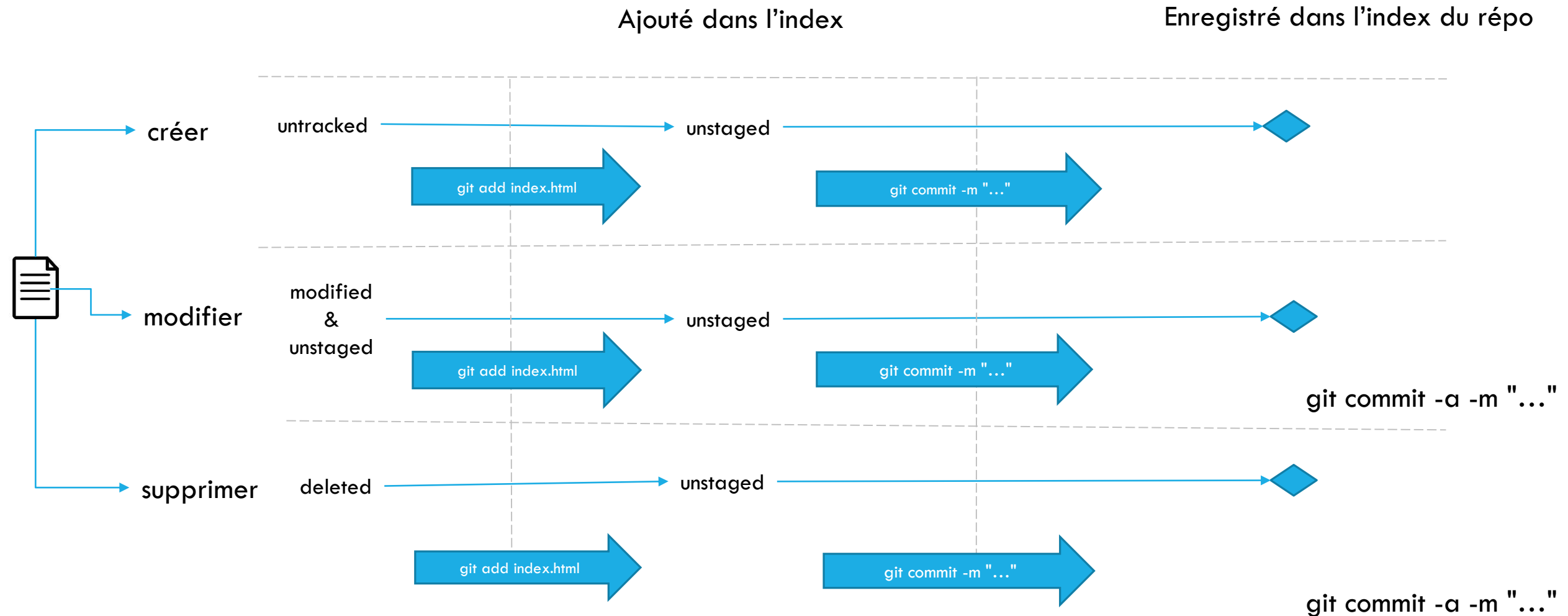
EXEMPLE D'UTILISATION DE GIT ADD AVEC WILDCARDS

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





REVENIR EN ARRIÈRE



REVENIR EN ARRIÈRE POUR UN FICHIER NON ADD

Dans le fichier « index.html » dupliquer la balise <h1> et son contenu

Dans le terminal saisir les commandes suivantes :

1. `git status`
2. `git checkout index.html`
3. `git status`

Regarder le contenu du fichier : la ligne dupliquée a disparu

Remarque : la commande **git checkout** peut être utilisée sans nom de fichier ce qui va remettre tous les fichiers dans leur état au moment du dernier commit

```
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 checkout index.html 2

C:\Users\HP\Desktop\hello>git status 3
On branch master
nothing to commit, working directory clean
```

VOIR LE DÉPÔT LORS D'UN PRÉCÉDENT COMMIT

Dans le terminal saisir les commandes suivantes:

- `git log --oneline`
- `git checkout 0c6a6d8` ← A

La première commande permet d'avoir la liste des commit avec leur id

La deuxième commande permet de revenir à un ancien commit.

Attention,

- veuillez saisir un id conforme à votre `git log --oneline` A
- si vous éditer un fichier et que vous faites un `git add` et `git commit`, ces commandes ne seront pas prise en compte

Pour revenir au présent :

- `git checkout master`

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

HP@pc MINGW64 ~/Desktop/hello (master) 2
$ git checkout 0c6a6d8
Note: checking out '0c6a6d8'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by performing another checkout.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -b with the checkout command again. Example:

    git checkout -b <new-branch-name>

HEAD is now at 0c6a6d8... ajout css
```

RÉCUPÉRER UN FICHIER À UN PRÉCÉDENT COMMIT

Dans le terminal saisir les commandes suivantes :


- `git checkout 0c6a6d8 index.html`
- `git status`
- `git commit -m "retour a l'ancien index.html"`
- La première commande permet de récupérer le fichier `index.html` à un précédent commit et il est en attente de commit

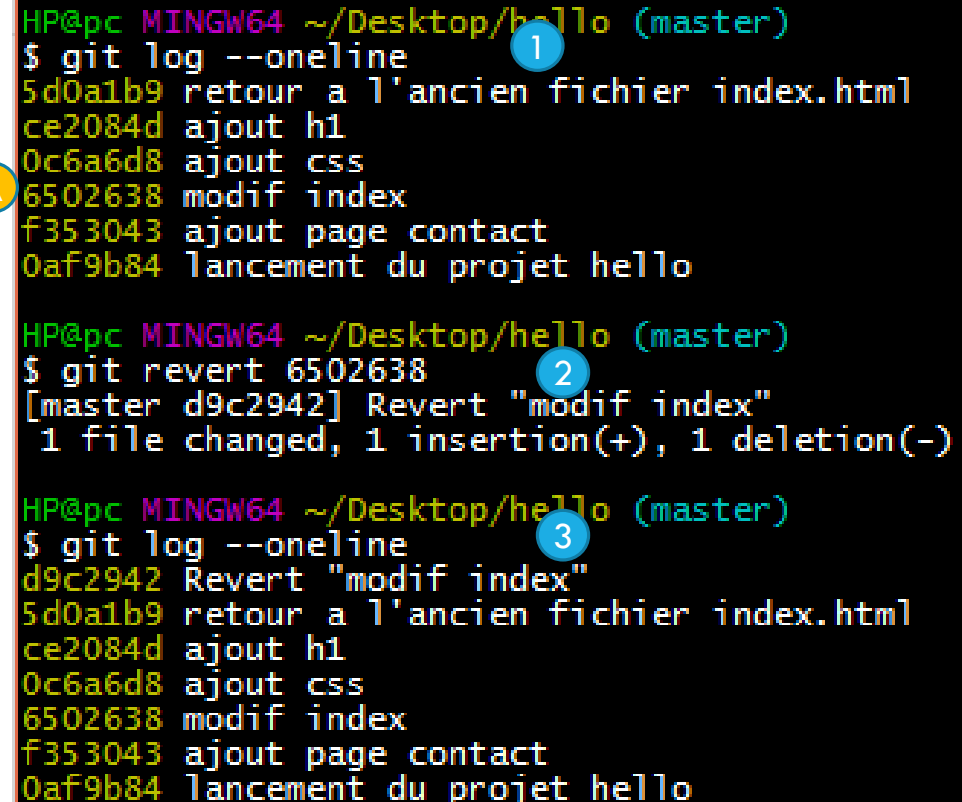
```
HP@pc MINGW64 ~/Desktop/hello (master)
$ git checkout 0c6a6d8 index.html 1
HP@pc MINGW64 ~/Desktop/hello (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html 2
HP@pc MINGW64 ~/Desktop/hello (master) 3
$ git commit -m "retour a l'ancien fichier index.html"
[master 5d0a1b9] retour a l'ancien fichier index.html
1 file changed, 1 insertion(+), 1 deletion(-)
```

COPIER UN ANCIEN COMMIT ET LE RENDRE ACTUEL

Dans le terminal saisir les commandes suivantes :

- git log --oneline 
- git revert 6502638 -m "retour à un ancien commit"
- git log --oneline



```
HP@pc MINGW64 ~/Desktop/hello (master)
$ git log --oneline
5d0a1b9 retour a l'ancien fichier index.html
ce2084d ajout h1
0c6a6d8 ajout css
6502638 modif index
f353043 ajout page contact
0af9b84 lancement du projet hello

HP@pc MINGW64 ~/Desktop/hello (master)
$ git revert 6502638
[master d9c2942] Revert "modif index"
1 file changed, 1 insertion(+), 1 deletion(-)

HP@pc MINGW64 ~/Desktop/hello (master)
$ git log --oneline
d9c2942 Revert "modif index"
5d0a1b9 retour a l'ancien fichier index.html
ce2084d ajout h1
0c6a6d8 ajout css
6502638 modif index
f353043 ajout page contact
0af9b84 lancement du projet hello
```

REVENIR À UN ANCIEN COMMIT ET SUPPRIMER TOUS CEUX ENTRE

Dans le terminal saisir les commandes suivantes :

- git log --oneline
- git reset 5d0a1b9 A
- git log --oneline

Comme vous pouvez le voir dans le résultat de la dernière commande le commit d9c2942 a disparu

```
$ git log --oneline 1  
d9c2942 Revert "modif index"  
5d0a1b9 retour a l'ancien fichier index.html  
ce2084d ajout h1  
0c6a6d8 ajout css  
6502638 modif index  
f353043 ajout page contact  
0af9b84 lancement du projet hello  
  
HP@pc MINGW64 ~/Desktop/hello (master)  
$ git reset 5d0a1b9 2  
Unstaged changes after reset:  
M      index.html  
  
HP@pc MINGW64 ~/Desktop/hello (master)  
$ git log --oneline 3  
5d0a1b9 retour a l'ancien fichier index.html  
ce2084d ajout h1  
0c6a6d8 ajout css  
6502638 modif index  
f353043 ajout page contact  
0af9b84 lancement du projet hello
```

LES 3 MODES DE GIT RESET

Il existe trois modes de git reset

- `git reset <id commit> --soft`
- `git reset <id commit> --mixte`
- `git reset <id commit> --hard`
- Le premier revient à l'ancien commit en conservant les fichiers à l'état unstaged
- Le deuxième revient à l'ancien commit en mettant les fichiers à l'état modified / untracked / deleted (mode par défaut)
- Le dernier revient à l'ancien commit sans rien conserver

```
HP@pc MINGW64 ~/Desktop/hello (master)
$ git log --oneline
5d0a1b9 retour a l'ancien fichier index.html
ce2084d ajout h1
0c6a6d8 ajout css
6502638 modif index
f353043 ajout page contact
0af9b84 lancement du projet hello

HP@pc MINGW64 ~/Desktop/hello (master)
$ git reset 6502638 --soft

HP@pc MINGW64 ~/Desktop/hello (master)
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   index.html

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
```

AUTRE MANIÈRE D'UTILISER GIT RESET

Dans le terminal saisir les commandes suivantes :

- git log --oneline
- git reset HEAD^^
- git log --oneline

La deuxième va permettre de revenir deux commit précédent (et supprimer tous ceux entre)

En + : A lire pour comprendre en détail toutes les possibilités de git reset : <https://delicious-insights.com/fr/articles/git-reset/>

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

HP@pc MINGW64 ~/Desktop/hello (master)
$ git reset HEAD^^ 2
Unstaged changes after reset:
M      index.html

HP@pc MINGW64 ~/Desktop/hello (master)
$ git log --oneline 3
0af9b84 lancement du projet hello
```


LISTE DES COMMANDES VUES

1. `git checkout <nom fichier> #` revenir au dernier commit en supprimant les modification non add
2. `git checkout <id commit> #` voir le dépôt lors d'un précédent commit (attention permet uniquement de visualiser) pour en sortir `git checkout master`
3. `git checkout <id commit> <nom fichier> #` récupérer le contenu du fichier lors d'un précédent commit en status unstaged
4. `git revert <id commit> :` copier le commit et le rendre actuel
5. `git reset <id commit> :` revenir à un ancien commit et supprimer tout ceux entre
6. `git resert <id commit> -- solf / --mixed / --hard :` différent niveau de retour en arrière en conservant peu ou pas les modifications antérieur commit et leur état
7. `git resert HEAD^ :` revenir au précédent commit en supprimant le commit actuel

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

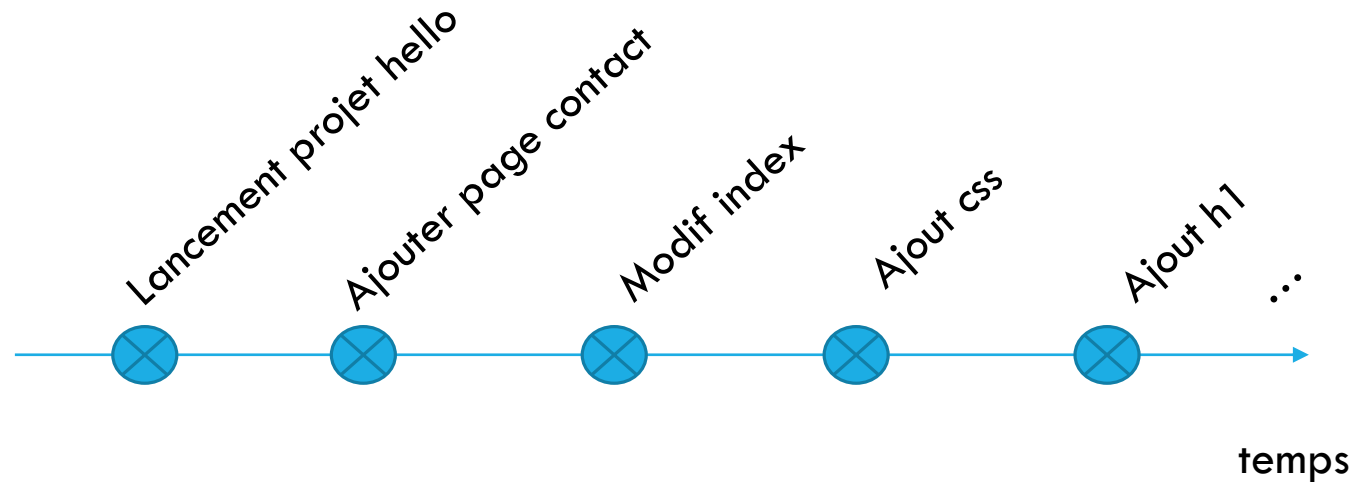


LES BRANCHES



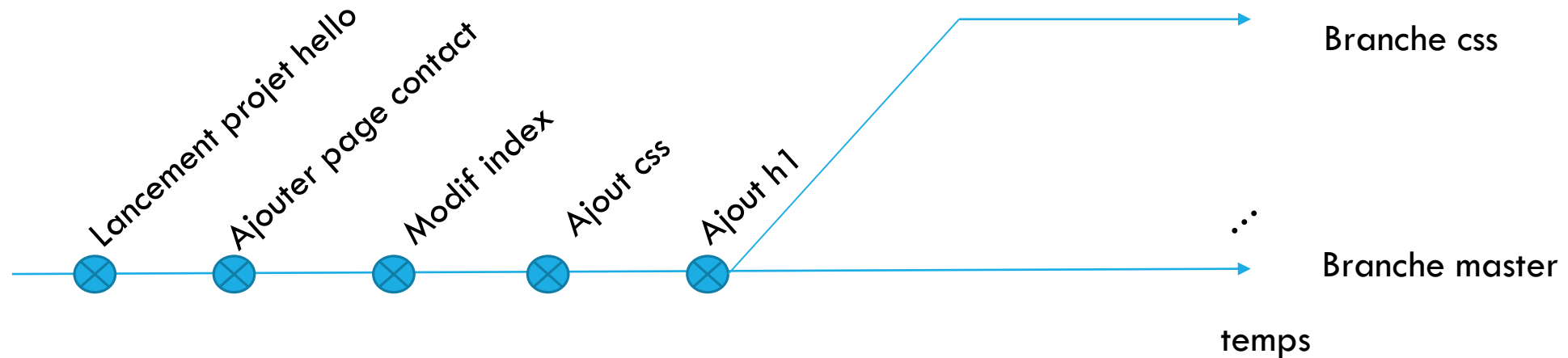
LA BRANCHE MASTER

Pour l'instant, nous avons travaillé sur la branche **Master**



Remarque la branche master contient tous les fichiers du dossier qui ont été add et commit

LE CONCEPT DE BRANCHE



Le concept de branche est très puissant :

- git va créer une nouvelle branche à partir du dernier commit
- Si on se place sur cette nouvelle branche, modifications et commit sont indépendants des commit de la branche master
- Pour vous, vous ne voyez aucune différence pour l'instant

CRÉER UNE NOUVELLE BRANCHE ET SE PLACER DESSUS

Dans le terminal saisir les commandes suivantes :

1. **git branch css**
2. **git checkout css**
3. **git status**

La première commande permet de créer la nouvelle branche intitulée css

La deuxième commande permet de se placer sur la branche css

La dernière commande facultative permet de confirmer que l'on est bien sur la branche css

```
C:\Users\HP\Desktop\hello>git branch css 1
C:\Users\HP\Desktop\hello>git checkout css 2
Switched to branch 'css'
C:\Users\HP\Desktop\hello>git status 3
On branch css
nothing to commit, working directory clean
```

LA NOUVELLE BRANCHE FONCTIONNE COMME LA BRANCHE MASTER

Dans le dossier « Hello »

- créer un fichier style.css contenant les règles css contenu dans index.html
- dans index.html remplacer <style> par <link> vers le fichier style.css

Dans le terminal saisir les commandes suivantes :

1. **git status**
2. **git add -all**
3. **git commit -m "ajout style.css"**
4. **git status**

```
C:\Users\HP\Desktop\hello>git status 1
On branch css
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

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

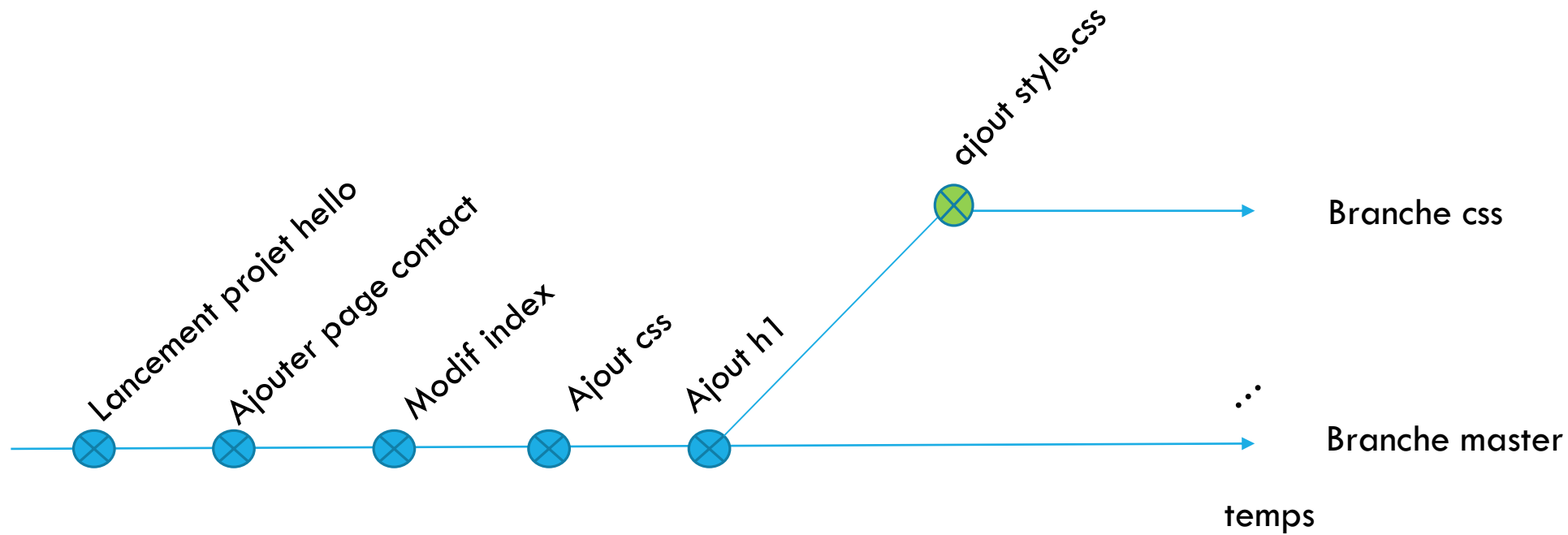
        style.css

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

C:\Users\HP\Desktop\hello>git add --all 2
C:\Users\HP\Desktop\hello>git commit -m "ajout style.css" 3
[css 951ab5e] ajout style.css
2 files changed, 6 insertions(+), 7 deletions(-)
create mode 100644 style.css

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

COMMIT SUR UNE BRANCHE



RETOUR VERS LE FUTUR

Dans le terminal, saisir les commandes suivantes :

1. `git checkout master`

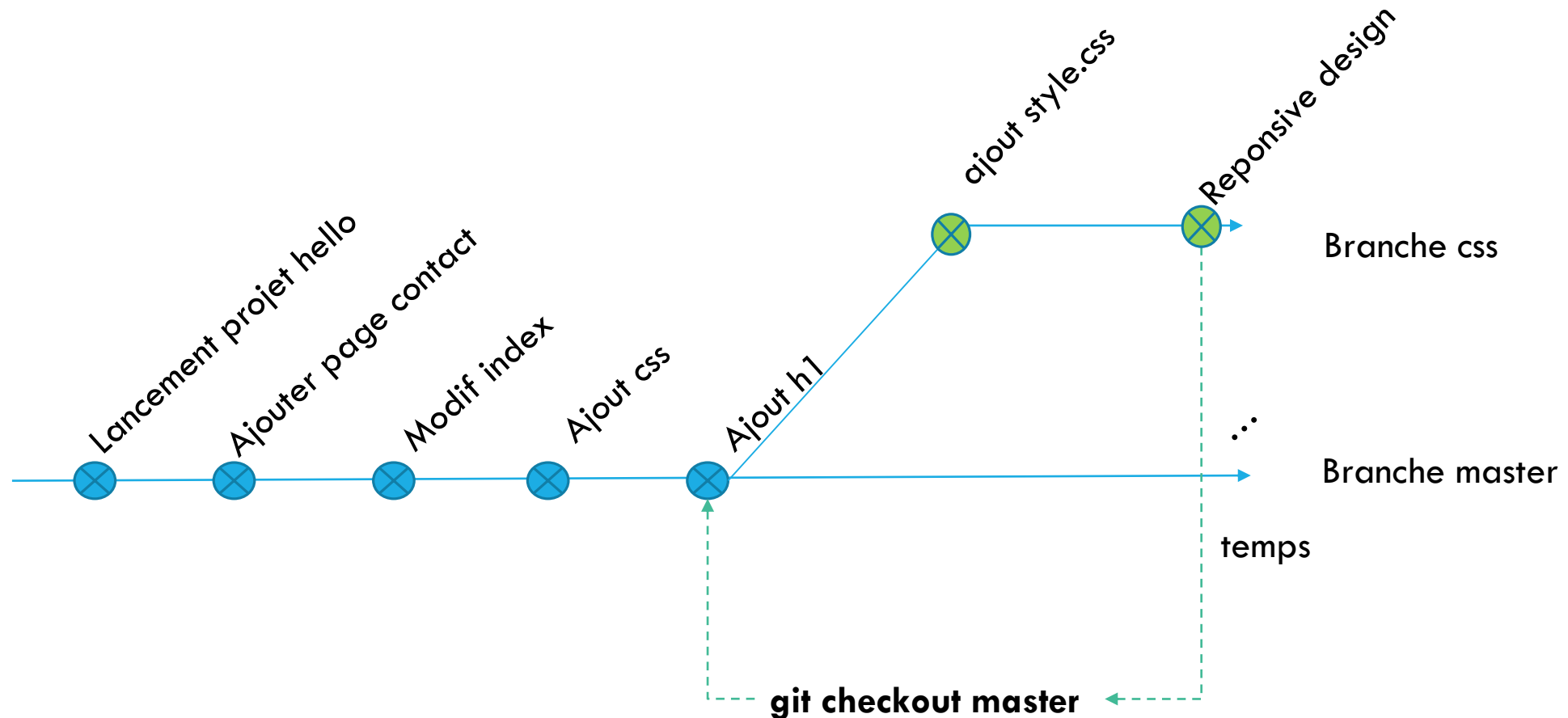
Cette commande nous a remis sur la branche master lors du dernier commit :

- `style.css` a disparu (car il n'existait pas)
- le fichier `index.html` a récupéré la balise `<style>` et son contenu

```
C:\Users\HP\Desktop\hello>git checkout master  
Switched to branch 'master'
```

```
C:\Users\HP\Desktop\hello>
```


Retour vers le dernier commit via `git checkout master`



RETOUR SUR LA BRANCHE

Dans le terminal, saisir les commandes suivantes :

1. `git checkout css`

Cette commande nous a remis sur le dernier commit de la branche css :

- style.css est réapparu
- le fichier index.html a récupéré la balise <link>

```
C:\Users\HP\Desktop\hello>git checkout css
Switched to branch 'css'
```

FUSIONNER LA BRANCHE AVEC MASTER

Dans le terminal, saisir les commandes suivantes :

1. `git checkout master`
2. `git merge css`

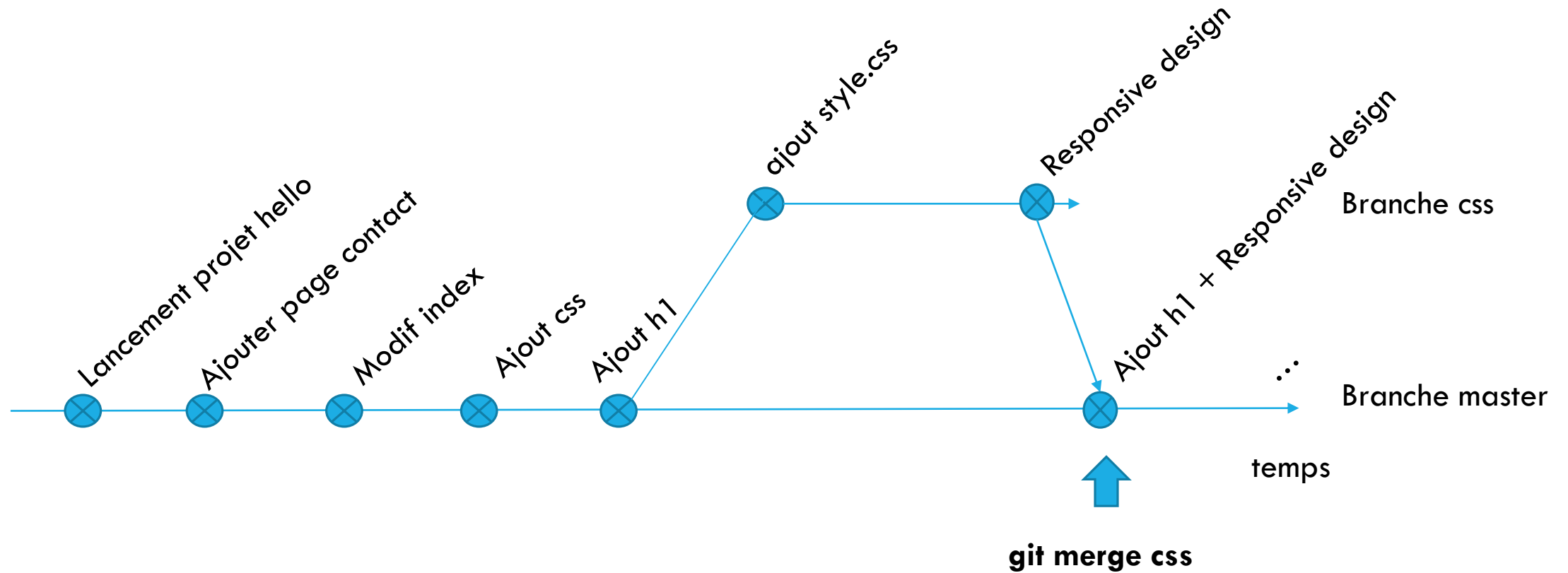
Quelques remarques :

1. Il faut au préalable se placer sur la branche dans laquelle on souhaite réaliser la fusion, car git merge va fusionner la branche dans la branche courante.
2. git merge crée un commit dans la branche
3. il est vivement conseillé de valider toutes les modifications réalisées dans la branche via add puis commit avant de merger

```
C:\Users\HP\Desktop\hello>git checkout master 1
Switched to branch 'master'

C:\Users\HP\Desktop\hello>git merge css 2
Updating 6f687b8..951ab5e
Fast-forward
 index.html | 8 +-----
 style.css  | 5 +++++
 2 files changed, 6 insertions(+), 7 deletions(-)
 create mode 100644 style.css
```

Fusion des branches



LISTER ET COUPER LA BRANCHE CSS

Dans le terminal, saisir les commandes suivantes :

1. `git branch`
2. `git branch -d css`

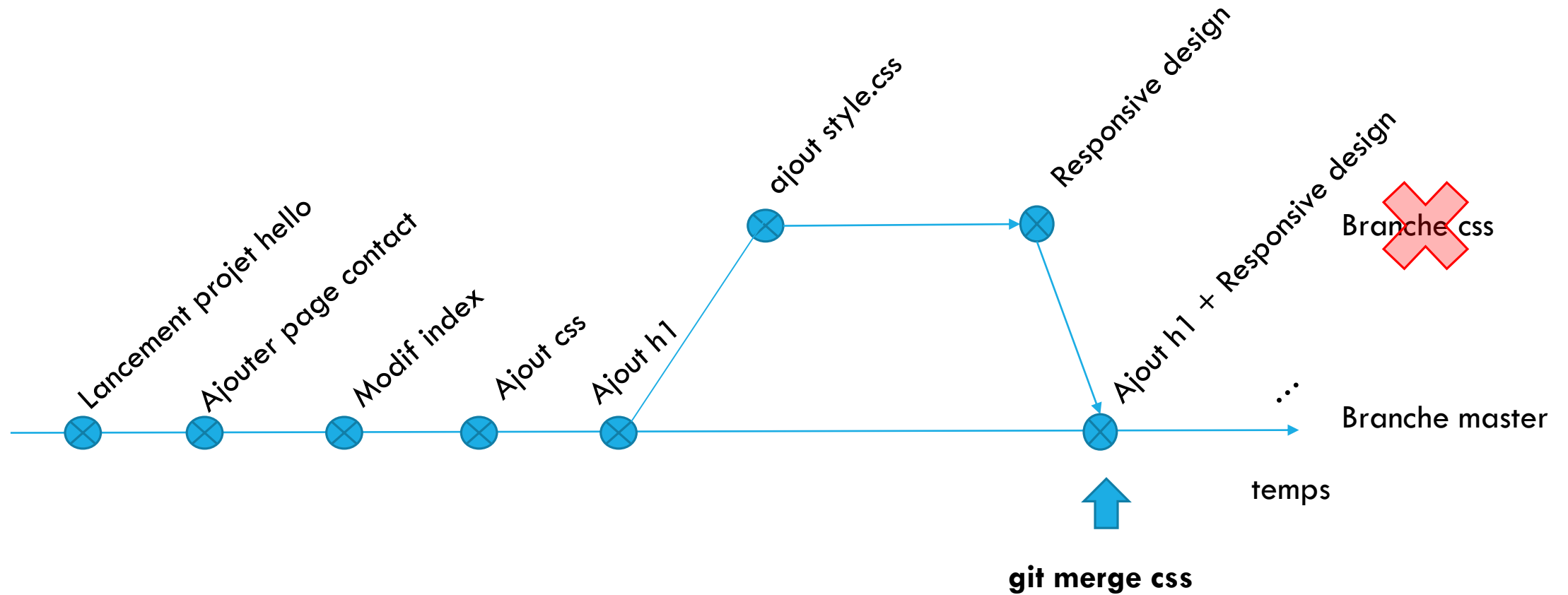
La première commande permet de lister toutes les branches disponibles et de savoir sur laquelle on travaille actuellement

La deuxième commande permet de supprimer la branche css

```
C:\Users\HP\Desktop\hello>git branch 1
css
* master

C:\Users\HP\Desktop\hello>git branch -d css 2
Deleted branch css (was 951ab5e).
```

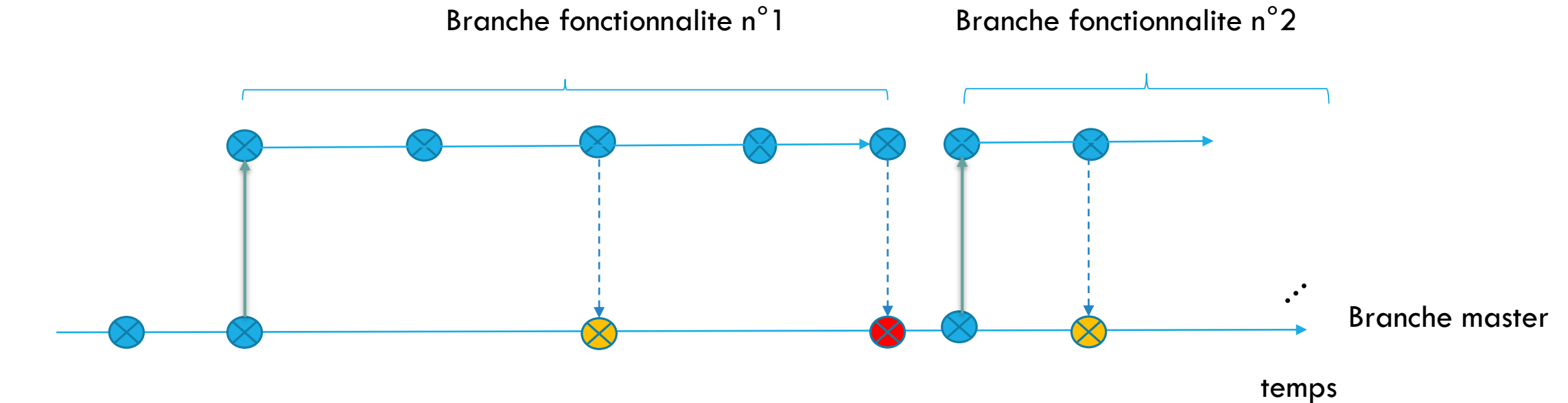
Après git branch -d css







CONSEILS POUR UTILISER LES BRANCHES

1. Créer une branche par fonctionnalité développée
2. Ne pas travailler simultanément sur la branche master et une branche
3. Eviter de créer trop de branches simultanément
4. Dès que la fonctionnalité est opérationnelle et stable
 - réalisez un merge avec la branche master et
 - supprimer la branche

Conseil utilisation Branche



- | | | | | | |
|---|---|--|--|---|---|
|  | <ul style="list-style-type: none">• git add <files>• git commit |  | <ul style="list-style-type: none">• git checkout master• git merge fonctionnalite |  | <ul style="list-style-type: none">• git checkout master• git merge fonctionnalite• git branch -d fonctionnalite |
|  | <ul style="list-style-type: none">• git branch fonctionnalite• git checkout fonctionnalite | | | | |

LISTE DES COMMANDES VUES

1. `git add --all` (ou `git add -A` ou `git add *`) : photo instantanée de tous les fichiers modifiés depuis le dernier
2. `git branch <nom branche>` : création d'une nouvelle branche et duplication du dernier commit de la branche en cours
3. `git checkout <nom branche>` : se positionner sur une autre branche au dernier commit
4. `git checkout master` : revenir sur la branche master au dernier commit
5. `git merge <nom branche>` : fusionner le dernier commit de la branche avec la branche en cours
6. `git branch` : lister toutes les branches disponibles et voir celle encours d'édition
7. `git branch -d <nom branche>` : supprimer une branche



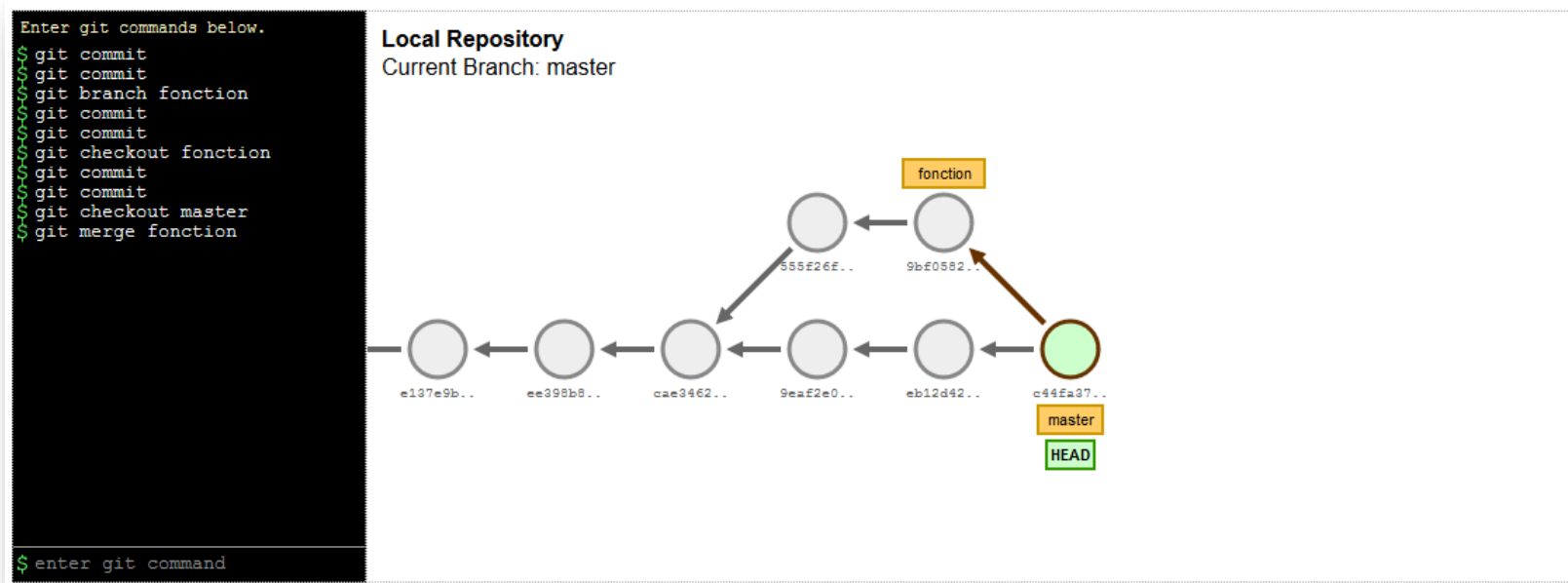
VISUALISER LES BRANCHES



OUTIL WEB

Outil graphique pour s'entraîner et comprendre le système de branche :

<https://onlywei.github.io/explain-git-with-d3/>



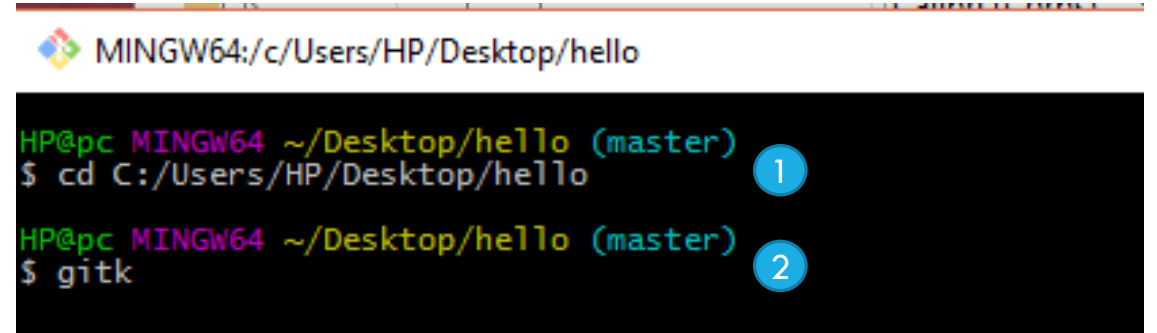
GITK

Lancer le terminal git-bash.exe

Dupliquer le path du dossier en cours

Saisir les commandes suivantes :

1. `cd C:/Users/HP/Desktop/hello`
2. `gitk`



```
MINGW64:/c/Users/HP/Desktop/hello  
HP@pc MINGW64 ~/Desktop/hello (master) 1  
$ cd C:/Users/HP/Desktop/hello  
HP@pc MINGW64 ~/Desktop/hello (master) 2  
$ gitk
```

VISUALISATION GRAPHIQUE DE L'ÉVOLUTION DU PROJET

The screenshot displays the gitk graphical user interface. At the top, the title bar reads "hello: All files - gitk". Below it is a menu bar with "File", "Edit", "View", and "Help".

The main window is divided into three panes. The left pane shows a commit history graph with nodes and branches. The central pane displays a list of commits, each with the author's name and email, and the commit date. The right pane shows the diff of the selected commit.

The commit history graph on the left shows a sequence of commits: "lancement du projet hello", "ajout page contact", "modif index", "ajout css", "ajout h1", "ajout style.css", "changement nom branche master", "changement titre", "resolution conflit", "texte final", and "Merge branch 'conflit'".

The commit list in the center shows 11 commits by Malik <malik.h@webdevpro.net> on 2017-01-23. The selected commit is the most recent one, with SHA1 ID 32f82952432a6429ab615ef9cd19c8406e8d00ce.

The diff view on the right shows the changes in the selected commit. It includes the author and committer information, the parent commit hashes, and the branch name. The diff itself shows the changes in the file "hello".

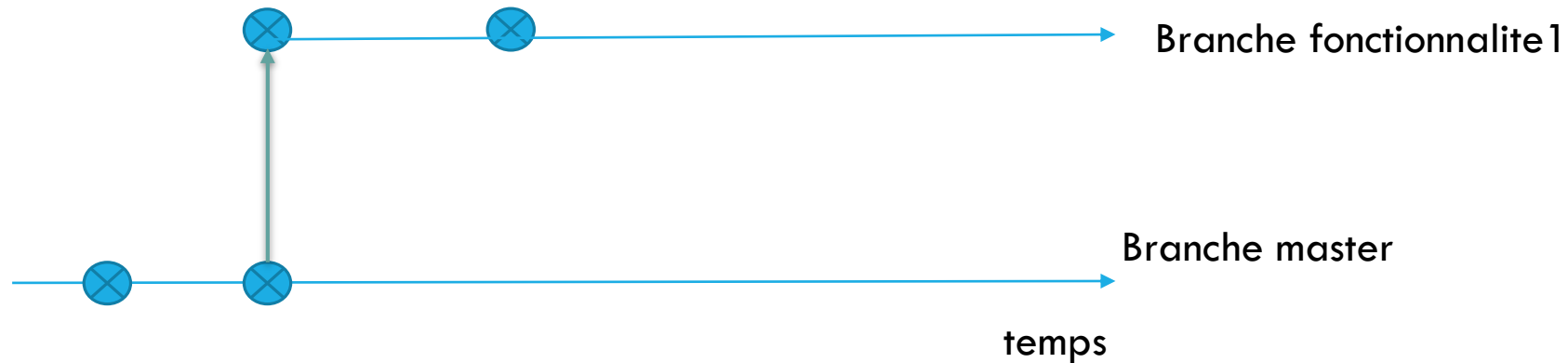
The bottom of the window features a search bar, a "Find" button, and a "Search" button. There are also radio buttons for "Diff", "Old version", and "New version", and a "Lines of context" dropdown set to 3. A checkbox for "Ignore space change" is also present.



CAS PARTICULIÈRE POUR LES BRANCHES



CAS PARTICULIER : SUPPRIMER UNE BRANCHE SANS MERGER DES COMMITS EN COURS



git vérifie en permanence l'état des différentes branches

git permet d'expérimenter en toute sécurité

PAR DÉFAUT GIT ÉVITER DE SUPPRIMER UNE BRANCHE DONT LE DERNIER COMMIT EST NON FUSIONNÉ

Dans le terminal, saisir les commandes suivantes

1. **git branch fonctionnalite1**
2. **git checkout fonctionnalite1**

Ajouter du contenu dans le fichier index.html

1. **git add --all**
2. **git commit -m "ajout texte"**
3. **git checkout master**
4. **git branch -d fonctionnalite1**

```
C:\Users\HP\Desktop\hello>git branch fonctionnalite1 1
C:\Users\HP\Desktop\hello>git checkout fonctionnalite1 2
Switched to branch 'fonctionnalite1'
C:\Users\HP\Desktop\hello>git status
On branch fonctionnalite1
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 --all 3
C:\Users\HP\Desktop\hello>git commit -m "nouvelle fonctionnalite" 4
[fonctionnalite1 a281c48] nouvelle fonctionnalite
1 file changed, 2 insertions(+)
C:\Users\HP\Desktop\hello>git checkout master 5
Switched to branch 'master'
C:\Users\HP\Desktop\hello>git branch -d fonctionnalite1 6
error: The branch 'fonctionnalite1' is not fully merged.
If you are sure you want to delete it, run 'git branch -D fonctionnalite1'.
```


LISTE DES COMMANDES VUES

1. `git branch -d <nom branche>` : supprimer une branche si tous ses commit ont bien été fusionnés (merge) avec la branche en cours
2. `git branch -D <nom branche>` : supprimer une branche sans prendre en compte les commit non fusionnés avec la branche en cours
Attention, cette commande est l'équivalent de « jeter son travail »

CONSEILS

1. Travailler le moins possible dans la branche Master
2. Travailler le plus possible dans des branches
3. Une fonctionnalité = une branche
4. Une branche ne doit pas exister plus de 2 jours



GESTION DES CONFLITS

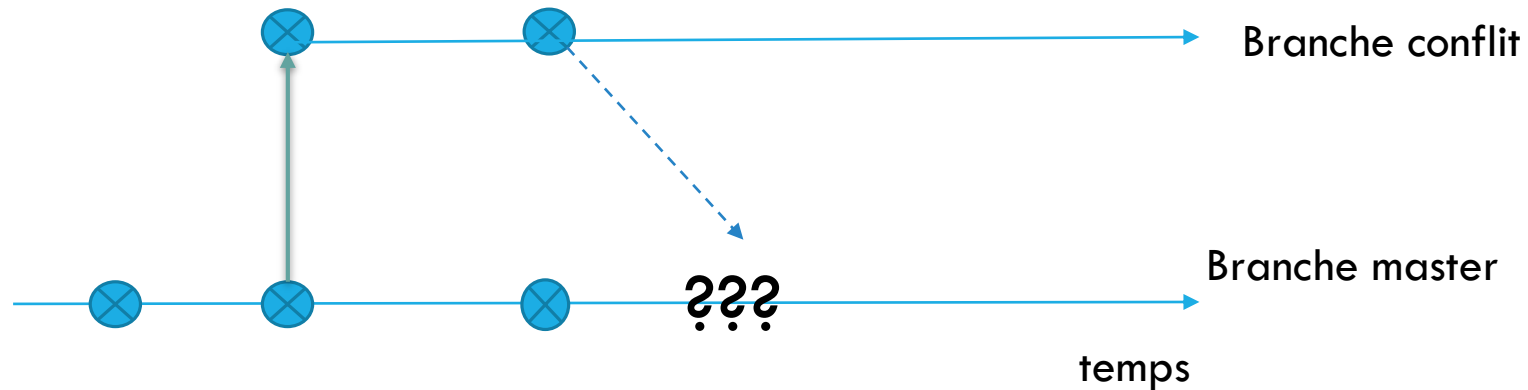


LES CONFLITS

La fusion de branches peut entrainer des conflits :

- Sur la même ligne du même fichier
- Deux branche demandent de faire des modifications différentes
- git n'arrive pas à choisir quelle est la bonne modification à prendre
- Il va noter dans le fichier où est le conflit
- C'est à l'utilisateur de choisir ce que doit contenir la fusion finale

EXEMPLE DE CONFLIT



Réaliser une fusion de deux branches où le texte du même fichier sur la même ligne dispose d'une information différente

SIMULER UN CONFLIT — ÉTAPE 1

Dans le terminal, saisir les commandes suivantes :

1. **git branch conflit**
2. **git checkout conflit**

Puis modifier le contenu de la balise <title> par Conflit dans le fichier index.html

1. **git add --all**
2. **git commit -m "changement titre"**

```
C:\Users\HP\Desktop\hello>git branch conflit 1
C:\Users\HP\Desktop\hello>git checkout conflit 2
M      index.html
Switched to branch 'conflit'
C:\Users\HP\Desktop\hello>git add --all 3
C:\Users\HP\Desktop\hello>git commit -m "changement titre" 4
[conflit 86b89a6] changement titre
1 file changed, 1 insertion(+), 1 deletion(-)
```

SIMULER UN CONFLIT — ÉTAPE 2

Dans le terminal, saisir les commandes suivantes :

1. `git checkout master`

Puis modifier le contenu de la balise `<title>` par no Conflit dans le fichier `index.html`

1. `git add --all`

2. `git commit -m "modif title master"`

```
C:\Users\HP\Desktop\hello>git checkout master 1
Switched to branch 'master'

C:\Users\HP\Desktop\hello>git add --all 2

C:\Users\HP\Desktop\hello>git commit -m "changement nom branche master" 3
[master 504d7a9] changement nom branche master
1 file changed, 1 insertion(+), 1 deletion(-)
```

SIMULER UN CONFLIT — ÉTAPE 3

Dans le terminal, saisir les commandes suivantes :

1. git merge conflit

git vous annonce que sa fonction de merge automatique n'a pas réussi à s'exécuter jusqu'au bout dans le fichier index.html

```
C:\Users\HP\Desktop\hello>git merge conflit 1
Auto-merging index.html
CONFLICT (content): Merge conflict in index.html
Automatic merge failed; fix conflicts and then commit the result.
```


SIMULER UN CONFLIT — ÉTAPE 4

Regardons le contenu du fichier index.html :

- git a ajouté des annotations sur la zone où il n'arrive pas à gérer le conflit
 1. HEAD : ce qu'il y a écrit dans la branche master
 2. conflit : ce qu'il y a écrit dans la branche conflit
- git a mis les deux choix
- Mais ne peut choisir

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <<<<<<< HEAD
5  |   <title>pas de conflit</title> 1
6  =====
7  |   <title>Conflit</title>        2
8  >>>>>>> conflit
9  |   <meta charset="UTF-8" />
10 |   <link rel="stylesheet" href="style.css">
11 </head>
12 <body>
13 |   <h1>Bonjour tout le monde</h1>
14 </body>
15 </html>
```

RÉSOUTRE LE CONFLIT — ÉTAPE 1

1. Dans le fichier index.html conserver ce que l'on veut garder au final :
 1. Head = Master ou
 2. Conflit

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <title>pas de conflit</title> 1
5     <meta charset="UTF-8" />
6     <link rel="stylesheet" href="style.css">
7 </head>
8 <body>
9     <h1>Bonjour tout le monde</h1>
10 </body>
11 </html>
```

RÉSOLUDRE LE CONFLIT — ÉTAPE 2

Dans le terminal, saisir les commandes suivantes :

1. **git status**
2. **git add index.html**
3. **git commit -m "resolution conflit"**

La première commande facultative montre qu'il y a un merge en conflit

La deuxième commande est un git add classique qui va prendre en compte le fichier que nous avons modifié

La dernière commande finalise le conflit

```
C:\Users\HP\Desktop\hello>git status 1
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")

Unmerged paths:
  (use "git add <file>..." to mark resolution)

        both 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 "resolution conflit" 3
[master 7d2ee8d] resolution conflit
```

REVENIR SUR LA BRANCHE CONFLIT

Dans le terminal, saisir les commandes suivantes :

1. **git checkout conflit**

Mettre *pas de conflit* dans la balise <title> de index (se mettre d'accord)

1. **git add --all**
2. **git commit -m "texte final"**
3. **git checkout master**
4. **git merge conflit**

```
C:\Users\HP\Desktop\hello>git checkout conflit 1
Switched to branch 'conflit'

C:\Users\HP\Desktop\hello>git add --all 2

C:\Users\HP\Desktop\hello>git commit -m "texte final" 3
[conflit c86873a] texte final
1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\HP\Desktop\hello>git checkout master 4
Switched to branch 'master'

C:\Users\HP\Desktop\hello>git merge conflit 5
Merge made by the 'recursive' strategy.
```



LES FICHIERS .GITIGNORE



FILTRER LES FICHIERS / DOSSIERS QUE L'ON VEUT SUIVRE ÉTAPE 1

Dans le dossier « Hello » créer un dossier image

Dans ce dossier, télécharger plusieurs images grâce au site <http://loempixel.com/>

Dans le terminal saisir :

1. `echo > .gitignore`
2. `git status`

Dans le dossier « Hello », un nouveau fichier a été créé intitulé

```
C:\Users\HP\Desktop\hello>echo > .gitignore

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

        .gitignore
        images/

nothing added to commit but untracked files present (use "git add" to
C:\Users\HP\Desktop\hello>git status
On branch master
nothing to commit, working directory clean
```

FILTRER LES FICHIERS / DOSSIERS QUE L'ON VEUT SUIVRE ÉTAPE 2

Editer le fichier .gitignore et saisir

```
1 .gitignore  
2 images/  
3
```

Enfin dans le terminal saisir :

1. git status

Le dossier images/ + son contenu ainsi que le fichier .gitignore ne sont plus repérés par git status

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

EXEMPLE D'INSTRUCTIONS À INSÉRER DANS .GITIGNORE

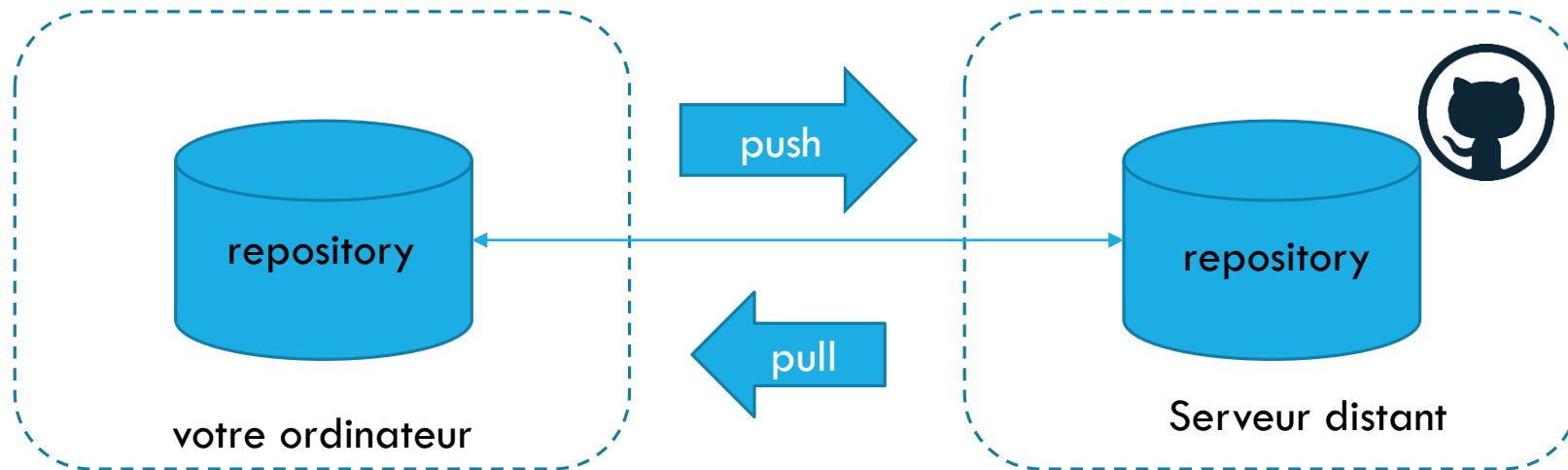
1. *.php # ignorer tous les fichiers finissant par .php
2. tmp/ #ignorer le dossier tmp et son contenu
3. *. (jpg|png|gif) #ignorer tous les fichiers images
4. httpdocs/img/ #ignorer tous ce qui est contenu et le dossier httpdocs/img/
5. httpdocs/*.txt # ignorer httpdocs/robots.txt, mais pas
httpdocs/log/connexion.txt
6. httpdocs/**/*.txt # ignorer tous les fichiers .txt sous le répertoire httpdocs/



TRAVAIL COLLABORATIF



PARTAGER SON CODE AVEC D'AUTRES DÉVELOPPEURS



Repository sur votre ordinateur correspond au dossier dans lequel nous avons exécuter git init

Repository : Dossier sur un serveur distant

ALTERNATIVES A GITHUB



GitLab


Voir article complet sur le sujet dans [exos/180617 alternatives github rachat microsoft.png](https://exos/180617-alternatives-github-rachat-microsoft.png) (developer.com)


CRÉER SON REPOSITORY SUR UN SERVEUR DISTANT ÉTAPE 1


Aller sur GitHub : <https://github.com/>



CRÉER SON REPOSITORY SUR UN SERVEUR DISTANT ÉTAPE 2

 Completed
Set up a personal account

 Step 2:
Choose your plan

 Step 3:
Tailor your experience

Choose your personal plan

☒ Unlimited public repositories for free.

☐ Unlimited private repositories for \$7/month. [\(view in EUR\)](#)

Don't worry, you can cancel or upgrade at any time.

☐ **Help me set up an organization next**
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
[Learn more about organizations.](#)

Continue

Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

CRÉER SON REPOSITORY SUR UN SERVEUR DISTANT ÉTAPE 3



Please verify your email address

Before you can contribute on GitHub, we need you to verify your email address.
An email containing verification instructions was sent to **malik1234.ifocop@gmail.com**.

Didn't get the email? [Resend verification email](#) or [change your email settings](#).

CRÉER SON REPOSITORY SUR UN SERVEUR DISTANT ÉTAPE 4

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch,
write comments, and open a pull request.


[Read the guide](#)

[Start a project](#)

CRÉER SON REPOSITORY SUR UN SERVEUR DISTANT ÉTAPE 5

Owner

Repository name

 webdevproformation ▼


 /

hello ✓


Great repository names are short and memorable. Need inspiration? How about **ideal-waffle**.

Description (optional)

une petite description

☒  **Public**

Anyone can see this repository. You choose who can commit.

☐  **Private**


You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

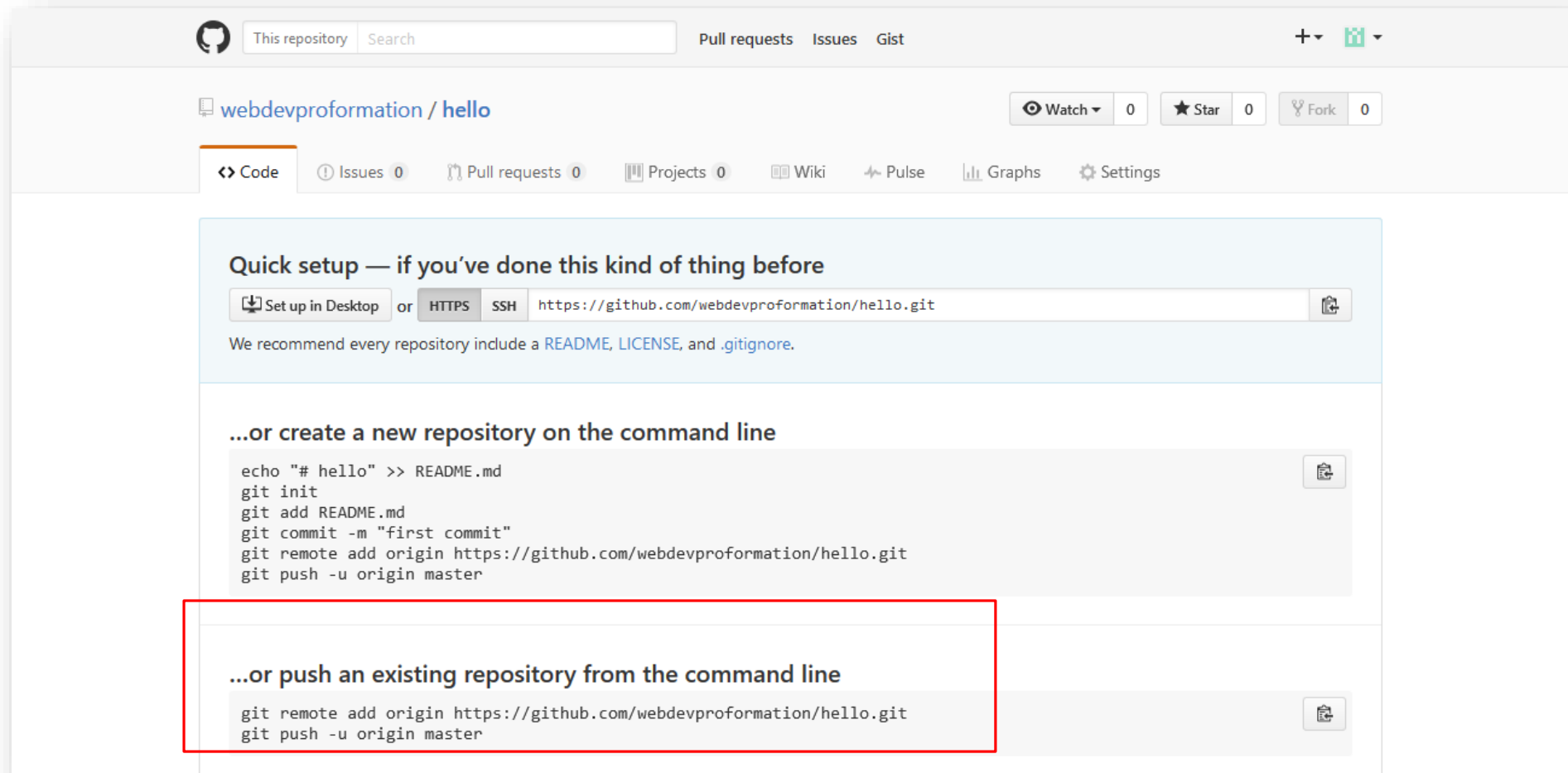
Add .gitignore: **None** ▼

 |

Add a license: **None** ▼ 

Create repository

CRÉER SON REPOSITORY SUR UN SERVEUR DISTANT ÉTAPE 6



POUSSER NOTRE PROJET HELLO LOCAL VERS LE RESPOSITORY DISTANT

Dans le terminal, saisir les commandes suivantes :

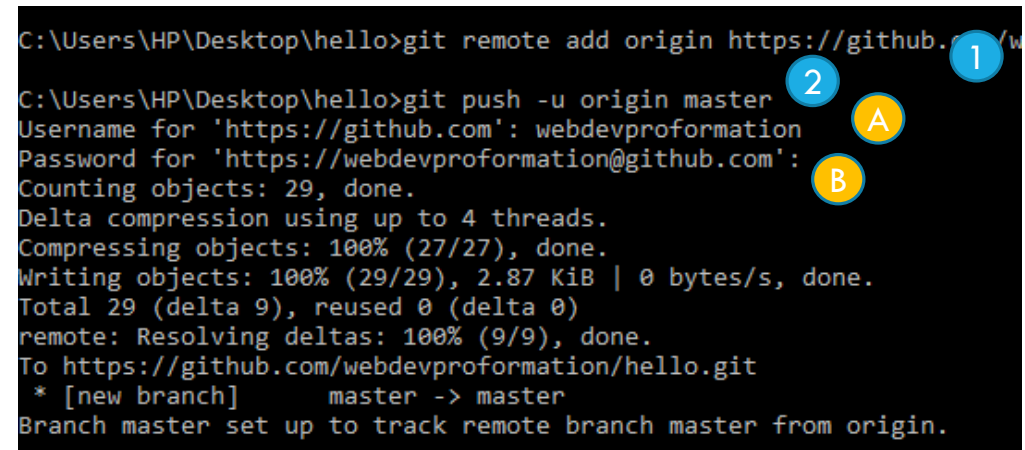
1. *** git remote add origin
 https://github.com/webdevproformation/hello.git
2. git push -u origin master
 - A. Un login est demandé
 - B. Un mot de passe est demandé

*** à récupérer à l'étape 6

La première commande ajoute à git l'adresse du serveur distant

La deuxième commande effectue le push des fichiers locaux vers le serveur distant :

- origin : repo github
- master : quelle branche



A terminal window showing the execution of git commands. The first command is 'git remote add origin https://github.com/webdevproformation/hello.git', annotated with a blue circle '1'. The second command is 'git push -u origin master', annotated with a blue circle '2'. The output shows the push process, including object counting, compression, and writing. It also shows the creation of a new branch 'master' and setting it up to track the remote branch 'master' from 'origin'. There are also yellow circles 'A' and 'B' next to the prompts for 'Username for 'https://github.com':' and 'Password for 'https://webdevproformation@github.com':' respectively.

```
C:\Users\HP\Desktop\hello>git remote add origin https://github.com/webdevproformation/hello.git
C:\Users\HP\Desktop\hello>git push -u origin master
Username for 'https://github.com': webdevproformation
Password for 'https://webdevproformation@github.com':
Counting objects: 29, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (27/27), done.
Writing objects: 100% (29/29), 2.87 KiB | 0 bytes/s, done.
Total 29 (delta 9), reused 0 (delta 0)
remote: Resolving deltas: 100% (9/9), done.
To https://github.com/webdevproformation/hello.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

VOTRE PROJET EST PRÊT À ÊTRE PARTAGÉ

The screenshot shows a GitHub repository page for 'webdevproformation / hello'. At the top, there are buttons for 'Watch' (0), 'Star' (0), and 'Fork' (0). Below these are tabs for 'Code', 'Issues' (0), 'Pull requests' (0), 'Projects' (0), 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The main content area has a section for 'une petite description' with an 'Edit' button. Below this is a summary bar showing '11 commits', '1 branch', '0 releases', and '0 contributors'. A progress bar is visible below the summary. Underneath the progress bar are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history section shows a list of commits: 'Malik Merge branch 'conflit'' (latest commit 32f8295, 7 hours ago), 'contact.html' (ajout page contact, 12 hours ago), 'index.html' (texte final, 7 hours ago), 'start.bat' (lancement du projet hello, 12 hours ago), and 'style.css' (ajout style.css, 10 hours ago). At the bottom, there is a blue box with the text 'Help people interested in this repository understand your project by adding a README.' and a green 'Add a README' button.

webdevproformation / hello

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

une petite description Edit

11 commits 1 branch 0 releases 0 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

Malik Merge branch 'conflit' Latest commit 32f8295 7 hours ago

contact.html	ajout page contact	12 hours ago
index.html	texte final	7 hours ago
start.bat	lancement du projet hello	12 hours ago
style.css	ajout style.css	10 hours ago

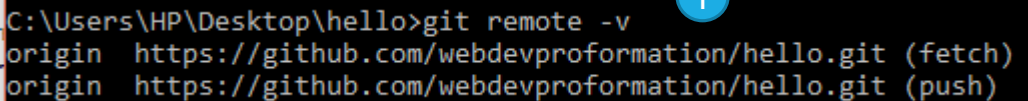
Help people interested in this repository understand your project by adding a README. Add a README

CONNAITRE LA LISTE DES REMOTES DISPONIBLES

Dans le terminal, saisir les commandes suivantes :

1. git remote -v

Liste l'ensemble des repository distants disponibles / leur nom abrégé (origin) / les actions que l'on peut effectuer (push / fetch)



```
C:\Users\HP\Desktop\hello>git remote -v
origin  https://github.com/webdevproformation/hello.git (fetch)
origin  https://github.com/webdevproformation/hello.git (push)
```

LISTE DES COMMANDES VUES

1. `git remote add <name> <adresse repository distant> # ajouter un nouveau remote repository`
2. `git remote -v # liste des remotes disponibles`
3. `git push -u <name of remote repository> <branche> # transférer le contenu d'une branche sur son dernier commit vers un repository distant`



DÉCOUVERTE GITHUB



INTERFACE DE GITHUB DE VOTRE PROJET

The screenshot shows the GitHub interface for a repository named 'webdevproformation / hello'. At the top, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below this is a navigation bar with links for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The 'Code' tab is selected. The repository description is 'une petite description' with an 'Edit' button. Below the description, there is a summary bar showing '11 commits', '1 branch', '0 releases', and '0 contributors'. A progress bar is visible below this summary. The 'Branch: master' dropdown is set to 'master', and there is a 'New pull request' button. To the right, there are buttons for 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The commit history table shows the following entries:

























Commit	Message	Time
Malik Merge branch 'conflit'		Latest commit 32f8295 7 hours ago
contact.html	ajout page contact	12 hours ago
index.html	texte final	7 hours ago
start.bat	lancement du projet hello	12 hours ago
style.css	ajout style.css	10 hours ago

At the bottom, there is a prompt to 'Add a README' to help people understand the project.

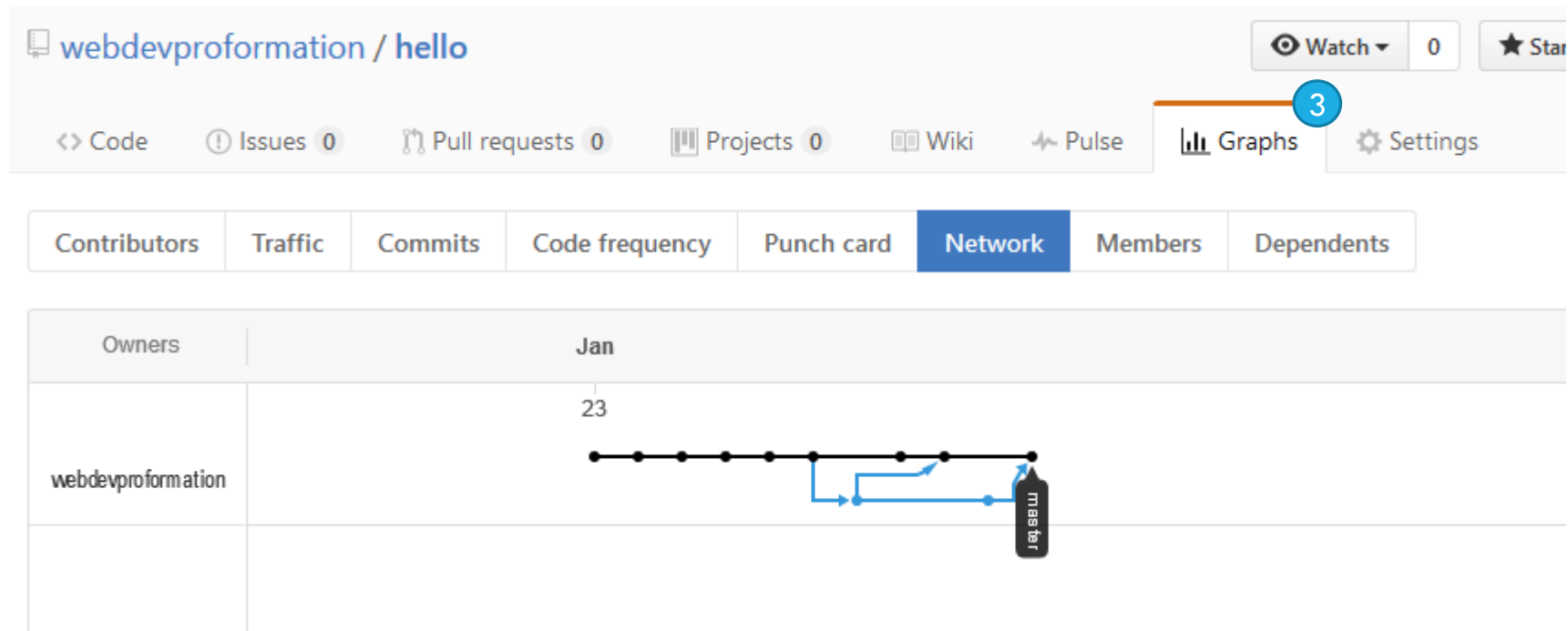
Numbered callouts in the image:

- 1: Points to the 'start.bat' file in the commit history table.
- 2: Points to the '11 commits' summary.
- 3: Points to the 'Graphs' link in the navigation bar.

GIT LOG & GIT DIFF VIA GITHUB

	Merge branch 'conflit' Malik committed 7 hours ago		32f8295	
	texte final Malik committed 7 hours ago		c86873a	
	resolution conflit Malik committed 7 hours ago		7d2ee8d	
	changement nom branche master Malik committed 8 hours ago		504d7a9	
	changement titre Malik committed 8 hours ago		86b89a6	 2
	ajout style.css Malik committed 10 hours ago		951ab5e	
	ajout h1 Malik committed 12 hours ago		6f687b8	
	ajout css Malik committed 12 hours ago		f3422a3	

GITK VIA GITHUB



MISE À JOUR DU REMOTE REPOSITORY

Modifier le fichier index.html

Dans le terminal, saisir les commandes suivantes :

1. git status
 2. git commit -a
 3. git push -u origin master
1. Login et mot de passe demandé

```
C:\Users\HP\Desktop\hello>git status 1
On branch master
Your branch is up-to-date with 'origin/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 commit -a 2
[master ba6cb2e] ajout de texte dans la page index
1 file changed, 1 insertion(+)

C:\Users\HP\Desktop\hello>git push -u origin master 3
Username for 'https://github.com': webdevproformation
Password for 'https://webdevproformation@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 552 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
To https://github.com/webdevproformation/hello.git
   32f8295..ba6cb2e  master -> master
Branch master set up to track remote branch master from origin.
```

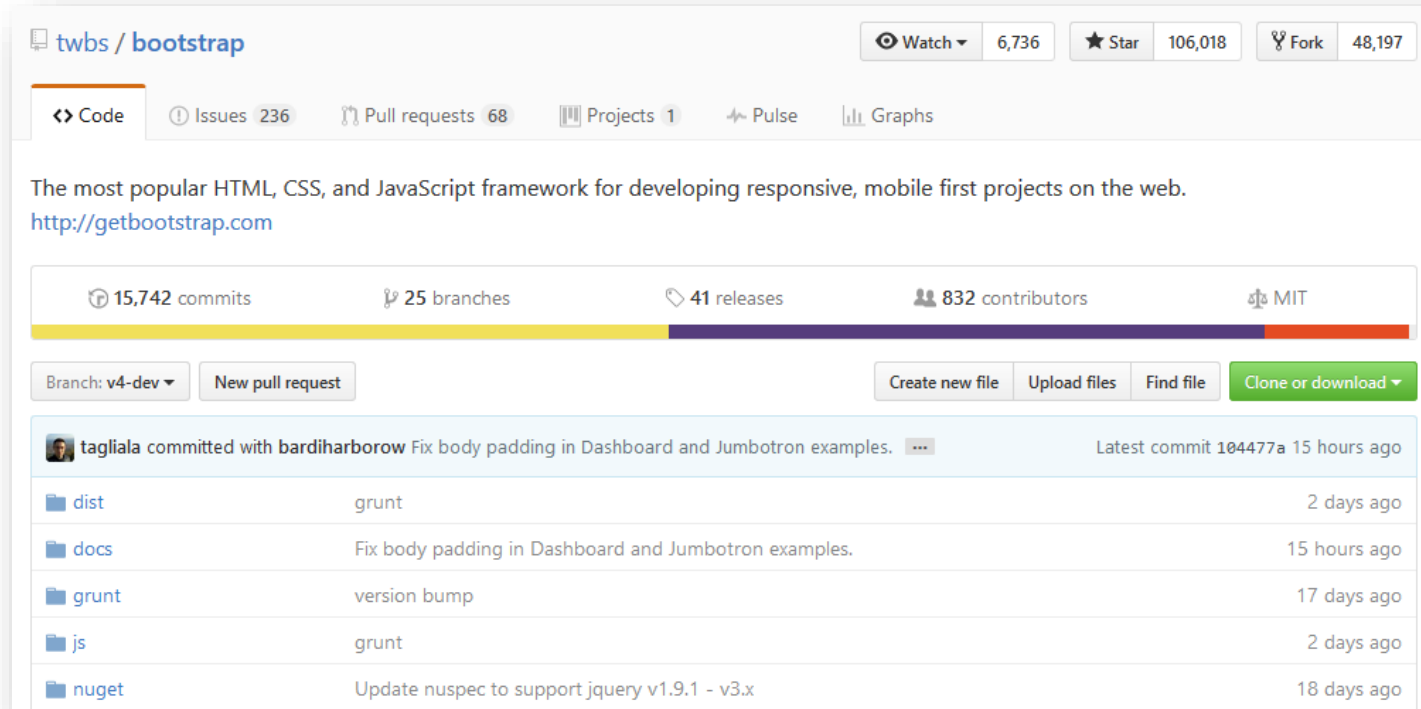


**GITHUB POUR DES PROJETS OPEN
SOURCE DE RÉFÉRENCE**



GITHUB DE TWITTER BOOTSTRAP

Aller sur <https://github.com/twbs/bootstrap>

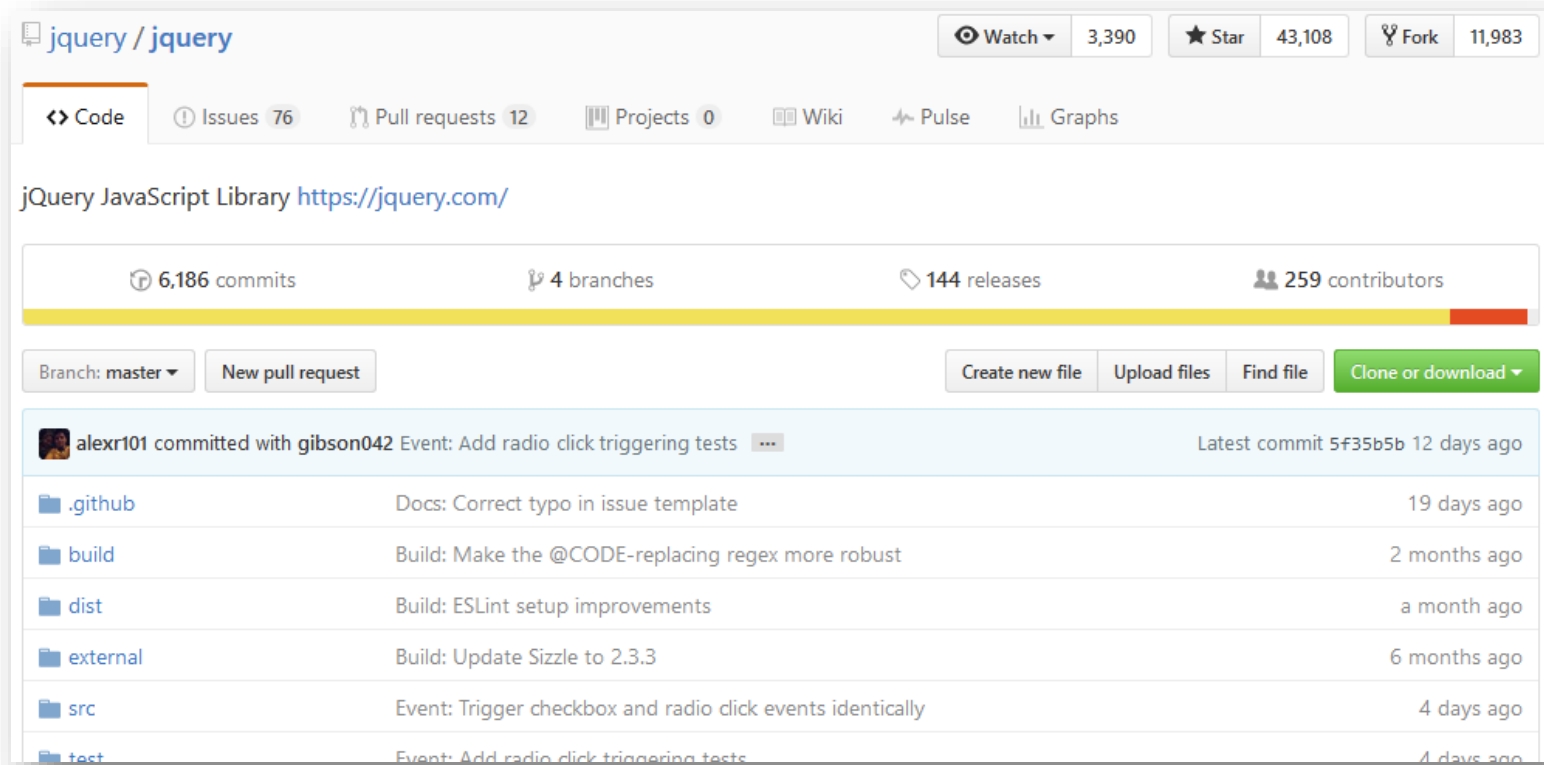


The screenshot shows the GitHub repository page for `twbs / bootstrap`. At the top, it displays the repository name and navigation links for Watch (6,736), Star (106,018), and Fork (48,197). Below this, there are tabs for Code, Issues (236), Pull requests (68), Projects (1), Pulse, and Graphs. The main description states: "The most popular HTML, CSS, and JavaScript framework for developing responsive, mobile first projects on the web." with a link to <http://getbootstrap.com>. A progress bar shows 15,742 commits, 25 branches, 41 releases, 832 contributors, and the MIT license. Below the progress bar, there are buttons for "Branch: v4-dev", "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download". The commit history table shows the latest commit by `tagliala` with `bardiharborow` fixing body padding in Dashboard and Jumbotron examples, committed 15 hours ago. The table lists several files and their commit messages:

File	Commit Message	Time Ago
<code>dist</code>	grunt	2 days ago
<code>docs</code>	Fix body padding in Dashboard and Jumbotron examples.	15 hours ago
<code>grunt</code>	version bump	17 days ago
<code>js</code>	grunt	2 days ago
<code>nuget</code>	Update nuspec to support jquery v1.9.1 - v3.x	18 days ago

GITHUB DE JQUERY

Aller sur <https://github.com/jquery/jquery>





GITHUB PULL REQUEST & GIT PULL



PUSH UNE BRANCHE PARTICULIÈRE

Dans un terminal saisir :

- git branch fonction
- Editer le fichier index.html et le modifier
- git commit -a
- git push -u origin fonction
- Saisir login et mot de passe

```
C:\Users\HP\Desktop\hello>git branch fonction
C:\Users\HP\Desktop\hello>git checkout fonction
Switched to branch 'fonction'
C:\Users\HP\Desktop\hello>git commit -a
[fonction 031675c] add js
1 file changed, 6 insertions(+)
C:\Users\HP\Desktop\hello>git push -u origin fonction
Username for 'https://github.com': webdevproformation
Password for 'https://webdevproformation@github.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 389 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/webdevproformation/hello.git
 * [new branch]      fonction -> fonction
Branch fonction set up to track remote branch fonction from origin.
```

QUI VA ÊTRE VALIDÉE SUR GITHUB ETAPE 1

The screenshot shows a GitHub repository page for 'webdevproformation'. The repository has 14 commits, 2 branches, 0 releases, and 0 contributors. The main branch is 'master'. The repository description is 'une petite description'. The commit history shows four recent commits: 'contact.html' (15 hours ago), 'index.html' (22 minutes ago), 'start.bat' (15 hours ago), and 'style.css' (12 hours ago). The commit messages are 'ajout page contact', 'add js', 'lancement du projet hello', and 'ajout style.css' respectively. The repository is currently in a state where a pull request is being merged. A green button 'Clone or download' is visible. At the bottom, there is a prompt to 'Add a README'.

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

une petite description Edit

14 commits 2 branches 0 releases 0 contributors

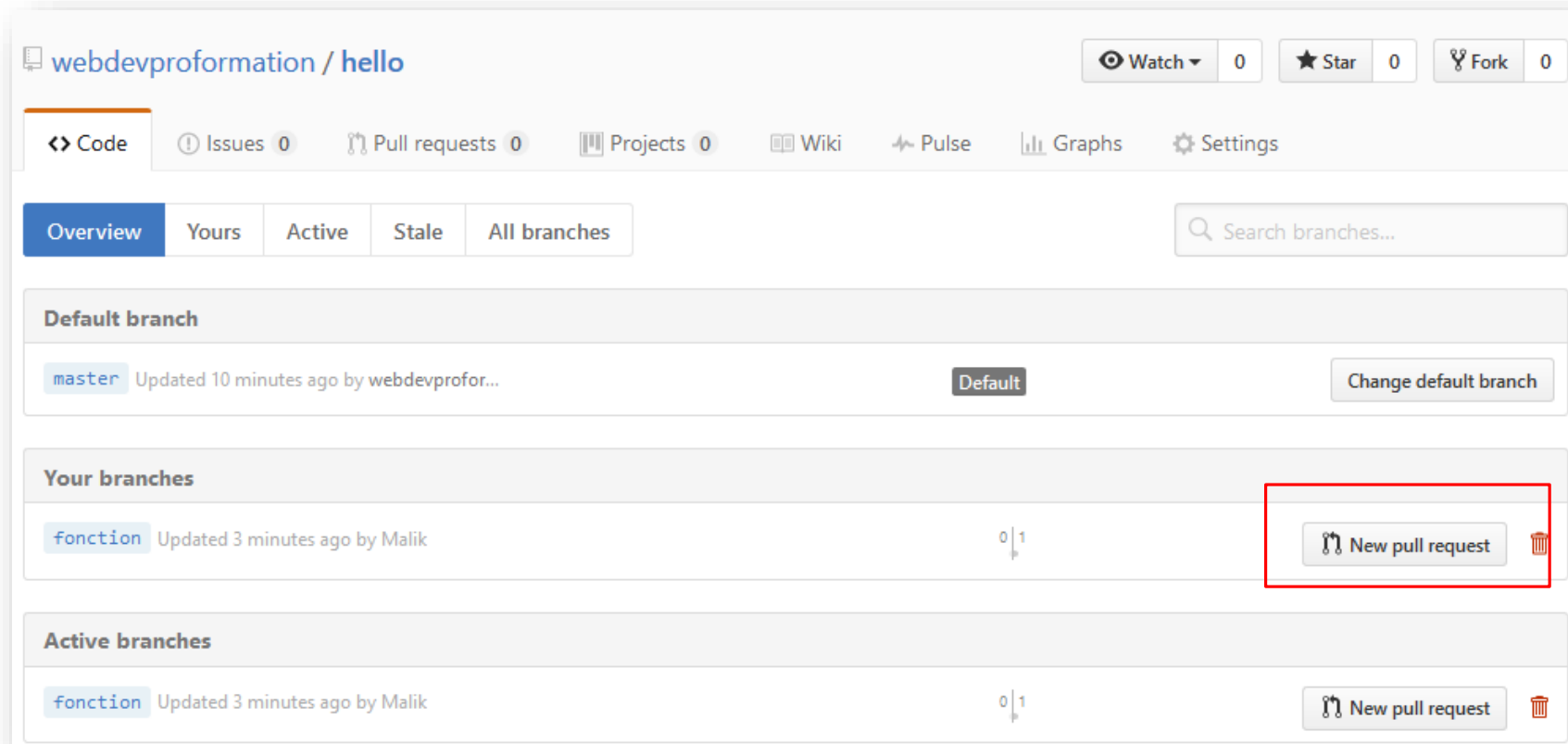
Branch: master New pull request Create new file Upload files Find file Clone or download

webdevproformation committed on GitHub Merge pull request #1 from webdevproformation/fonction Latest commit 6601d1a 9 minutes ago

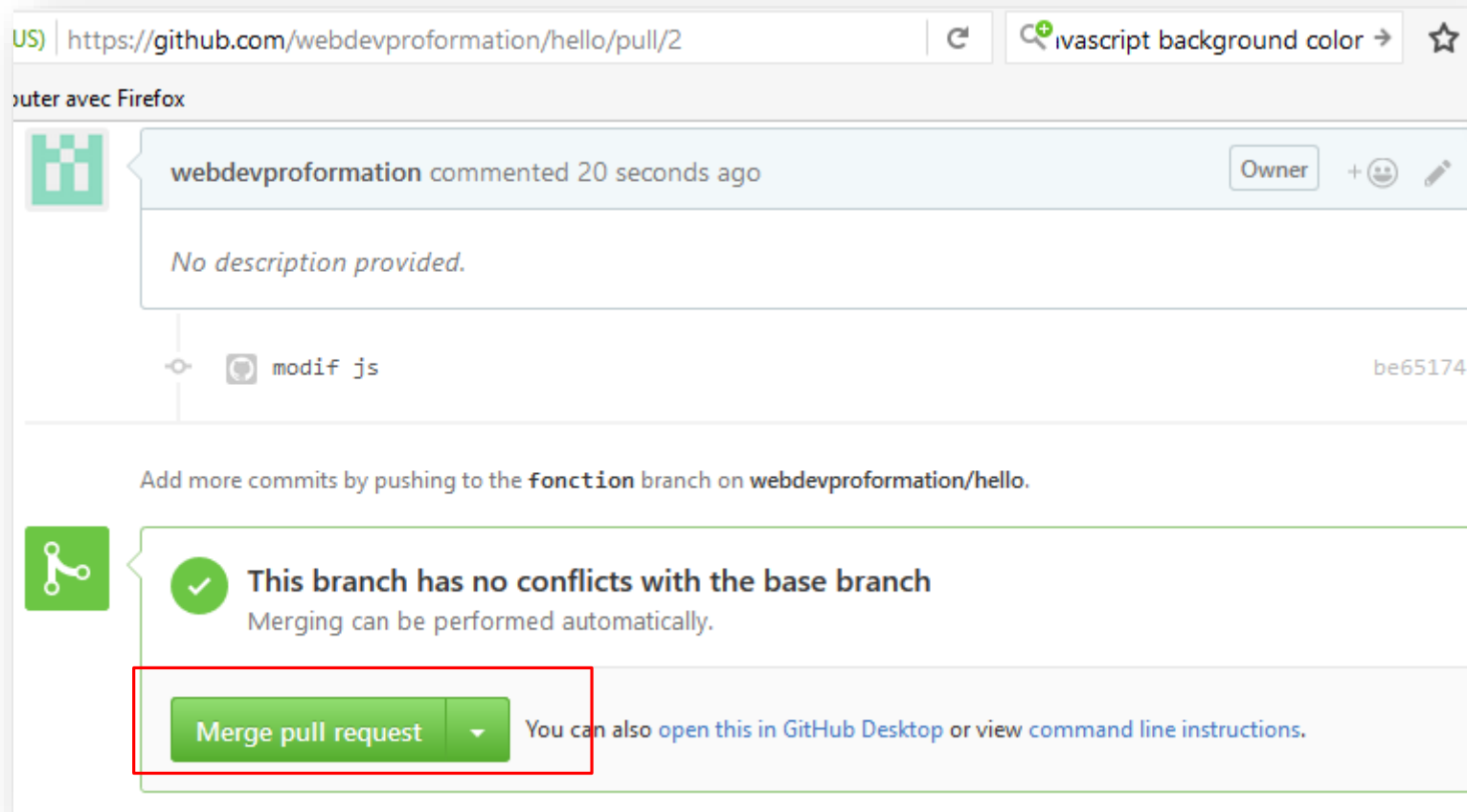
contact.html	ajout page contact	15 hours ago
index.html	add js	22 minutes ago
start.bat	lancement du projet hello	15 hours ago
style.css	ajout style.css	12 hours ago

Help people interested in this repository understand your project by adding a README. Add a README

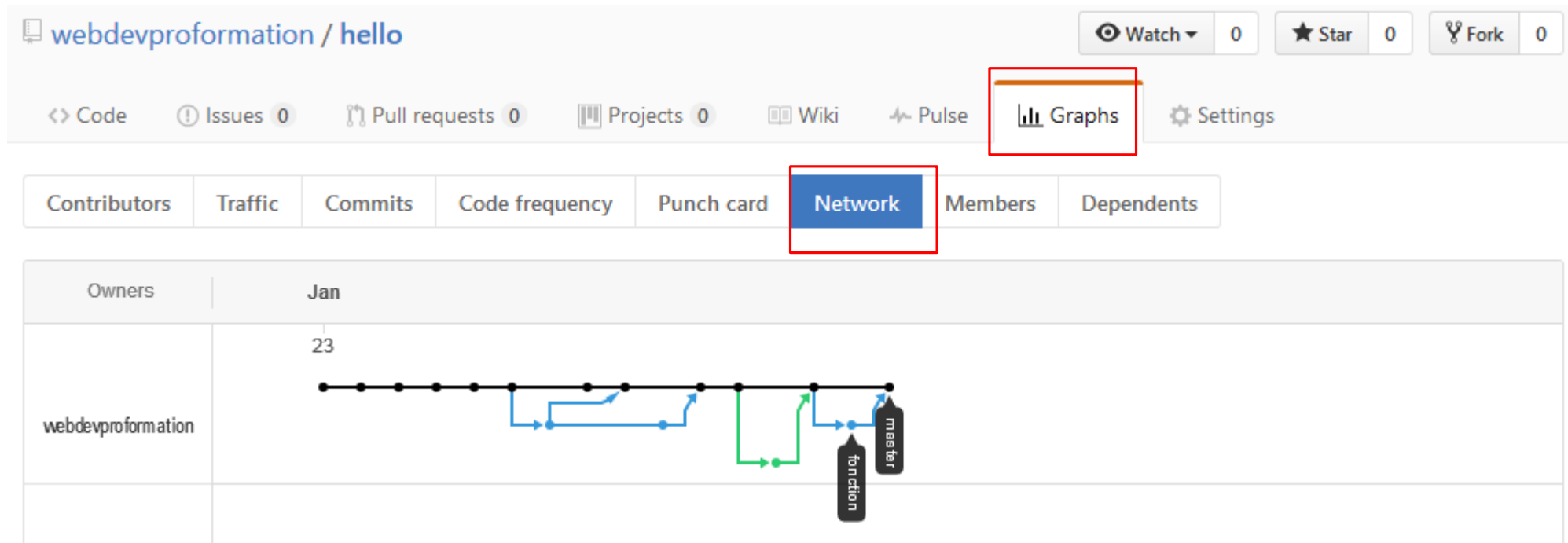
QUI VA ÊTRE VALIDÉE SUR GITHUB ETAPE 2



QUI VA ÊTRE VALIDÉE SUR GITHUB ETAPE 3



QUI VA ÊTRE VALIDÉE SUR GITHUB ETAPE 4



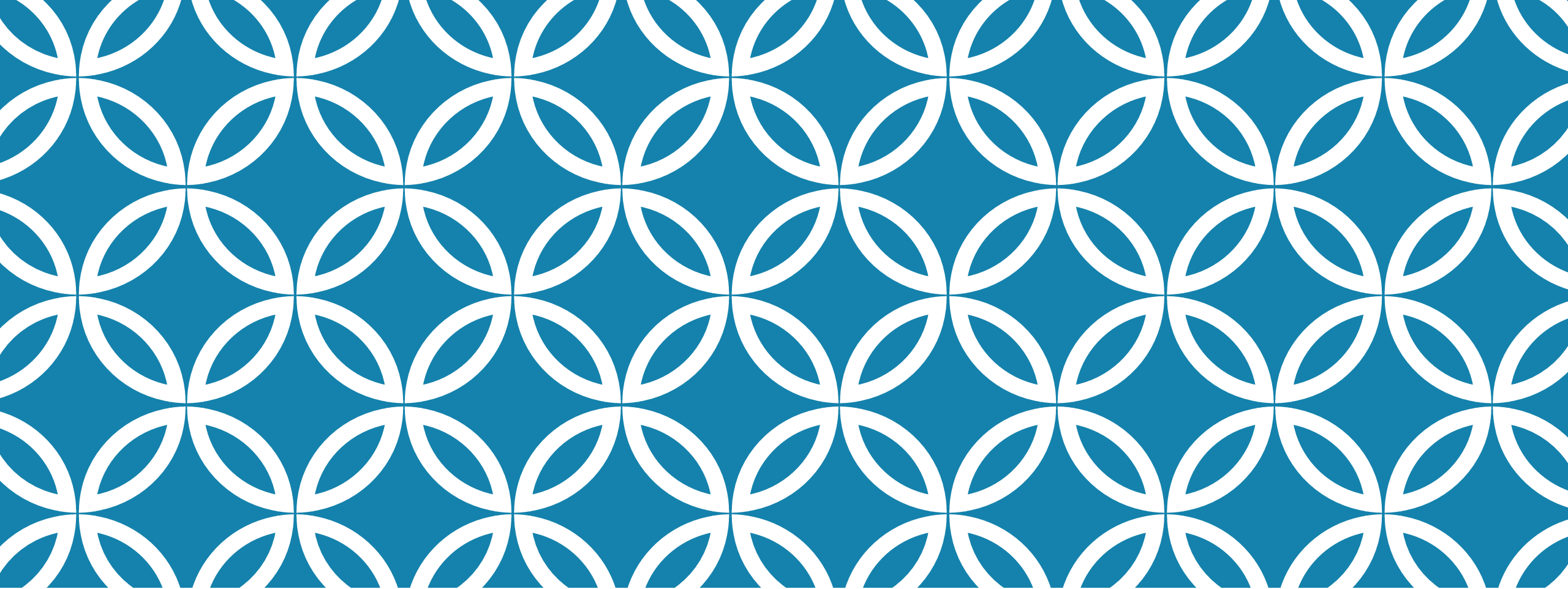
RAPATRIÉ EN LOCAL LES MODIFS VIA UN PULL

Dans un terminal saisir

1. git checkout master
2. git pull origin master

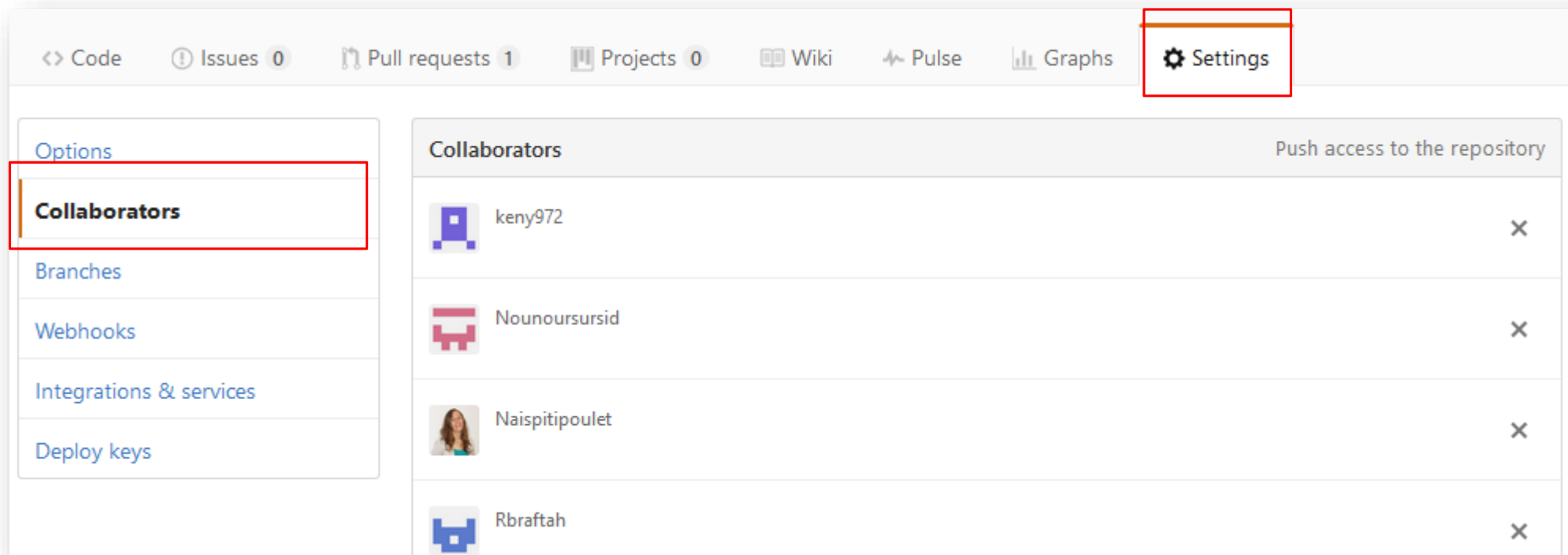
```
C:\Users\HP\Desktop\hello>git checkout master
Switched to branch 'master'
Your branch is up-to-date with 'origin/master'.

C:\Users\HP\Desktop\hello>git pull origin master
remote: Counting objects: 1, done.
remote: Total 1 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (1/1), done.
From https://github.com/webdevproformation/hello
 * branch                master      -> FETCH_HEAD
    6601d1a..a5b0c00      master      -> origin/master
Updating 6601d1a..a5b0c00
Fast-forward
 index.html | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```







**CONTRIBUER SUR LE PROJET D'UN
COLLÈGE**

DONNER LES DROITS DE COLLABORER



The screenshot shows the GitHub repository settings page. The 'Settings' tab is selected in the top navigation bar. In the left sidebar, the 'Collaborators' option is highlighted. The main content area displays a list of collaborators with their avatars, usernames, and a button to remove them.

Collaborators		Push access to the repository
	keny972	×
	Nounoursursid	×
	Naispitipoulet	×
	Rbraftah	×

RÉCUPÉRER L'URL DU PROJET DU COLLÈGE

The screenshot shows the GitHub interface for the repository 'webdevproformation / bonjour'. At the top, there are buttons for 'Watch', 'Star', and 'Fork', each with a count of 0. Below these are tabs for 'Code', 'Issues', 'Pull requests', 'Projects', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. The repository name 'bonjour' is displayed with an 'Edit' button. A summary bar shows '25 commits', '2 branches', '0 releases', and '2 contributors'. Below this, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The 'Clone or download' button is highlighted, and a dropdown menu is open, showing 'Clone with HTTPS' (selected), 'Use SSH', and a crossed-out 'Use Git or checkout with SVN using the web URL'. The HTTPS URL 'https://github.com/webdevproformation/bc' is displayed in a text box with a copy icon. Below the URL are buttons for 'Open in Desktop' and 'Download ZIP'. The repository's commit history is visible in the background, showing a merge pull request and several commits with file changes like 'SQL', 'css', 'doc/images', and 'contact.html'.

webdevproformation / bonjour

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 1 Projects 0 Wiki Pulse Graphs Settings

bonjour Edit

25 commits 2 branches 0 releases 2 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

webdevproformation committed on GitHub Merge pull request #4 from webdevproformation/dev

Clone with HTTPS ? Use SSH

~~Use Git or checkout with SVN using the web URL.~~

https://github.com/webdevproformation/bc

Open in Desktop Download ZIP

2 days ago

SQL	nouvelle modification a mettre en ligne
css	ajout main.css et modif index.html
doc/images	tout est ok
contact.html	tout est ok

RÉCUPÉRER LE PROJET ET APPORTER SA CONTRIBUTION

Clique droit Git Bash Here (pas besoin de créer un dossier)

Dans le terminal saisir :

- git clone <url d'un collègue>
- cd <nom repository>
- git branch dev
- git checkout dev
- Editer un fichier et le modifier
- git commit -a -m "ma contribution"
- git push -u origin dev
- Saisir login et mot de passe

```
HP@pc MINGW64 ~/Desktop
$ git clone https://github.com/webdevproformation/bonjour.git
Cloning into 'bonjour'...
remote: Counting objects: 80, done.
remote: Compressing objects: 100% (43/43), done.
remote: Total 80 (delta 37), reused 68 (delta 29), pack-reused 0
Unpacking objects: 100% (80/80), done.
Checking connectivity... done.

HP@pc MINGW64 ~/Desktop
$ cd bonjour/

HP@pc MINGW64 ~/Desktop/bonjour (master)
$ git branch dev

HP@pc MINGW64 ~/Desktop/bonjour (master)
$ git checkout dev
Switched to branch 'dev'


HP@pc MINGW64 ~/Desktop/bonjour (dev)
$ git commit -a -m "une contribution"
[dev fdc00c4] une contribution
1 file changed, 1 insertion(+)

HP@pc MINGW64 ~/Desktop/bonjour (dev)
$ git push origin dev
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 315 bytes | 0 bytes/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/webdevproformation/bonjour.git
 * [new branch]      dev -> dev
```


FAIRE UN PULL REQUEST


Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).



base: master ▾ ... compare: dev ▾

✓ **Able to merge.** These branches can be automatically merged.



une contribution

Write

Preview

AA ▾ B i “ <> 🔗 ☰ ☷ ✓ ↶ @ 📌

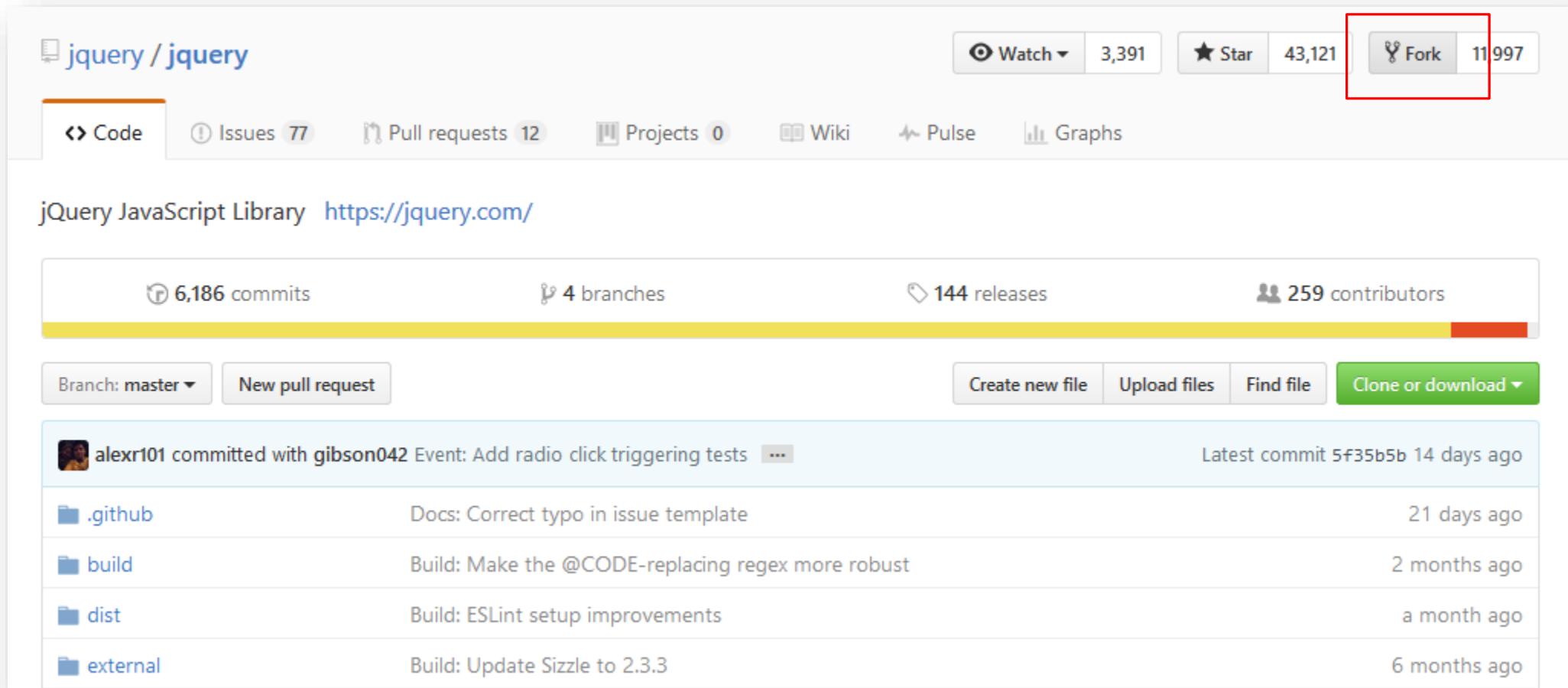
Leave a comment



GIT FORK



ALLER SUR UN PROJET SUR LEQUEL VOUS VOULEZ CONTRIBUER



The screenshot shows the GitHub repository page for jQuery. At the top, the repository name 'jquery / jquery' is displayed. To the right, there are buttons for 'Watch' (3,391), 'Star' (43,121), and 'Fork' (11,997). The 'Fork' button is highlighted with a red rectangle. Below these buttons, there are tabs for 'Code', 'Issues' (77), 'Pull requests' (12), 'Projects' (0), 'Wiki', 'Pulse', and 'Graphs'. The main content area shows the repository name 'jQuery JavaScript Library' and its URL 'https://jquery.com/'. Below this, there are statistics: '6,186 commits', '4 branches', '144 releases', and '259 contributors'. A yellow progress bar is shown below the statistics. At the bottom, there are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A commit history table is also visible, showing the latest commit by alexr101 and gibson042, and a list of recent commits with their descriptions and dates.

jquery / jquery

Watch 3,391 Star 43,121 Fork 11,997

Code Issues 77 Pull requests 12 Projects 0 Wiki Pulse Graphs

jQuery JavaScript Library <https://jquery.com/>


6,186 commits 4 branches 144 releases 259 contributors

Branch: master New pull request Create new file Upload files Find file Clone or download

alexr101 committed with gibson042 Event: Add radio click triggering tests Latest commit 5f35b5b 14 days ago

.github	Docs: Correct typo in issue template	21 days ago
build	Build: Make the @CODE-replacing regex more robust	2 months ago
dist	Build: ESLint setup improvements	a month ago
external	Build: Update Sizzle to 2.3.3	6 months ago

LE PROJET EST IMPORTÉ DANS VOTRE PROFIL

 **webdevproformation / jquery**
forked from jquery/jquery

Watch 0Star 0Fork 11,998


CodePull requests 0Projects 0WikiPulseGraphsSettings




jQuery JavaScript Library <https://jquery.com/> Edit

6,186 commits4 branches144 releases259 contributors

Branch: masterNew pull requestCreate new fileUpload filesFind fileClone or download

This branch is even with jquery:master. Pull requestCompare

 alexr101 committed with gibson042 Event: Add radio click triggering tests Latest commit 5f35b5b 14 days ago

 .github	Docs: Correct typo in issue template	21 days ago
 build	Build: Make the @CODE-replacing regex more robust	2 months ago
 dist	Build: ESLint setup improvements	a month ago

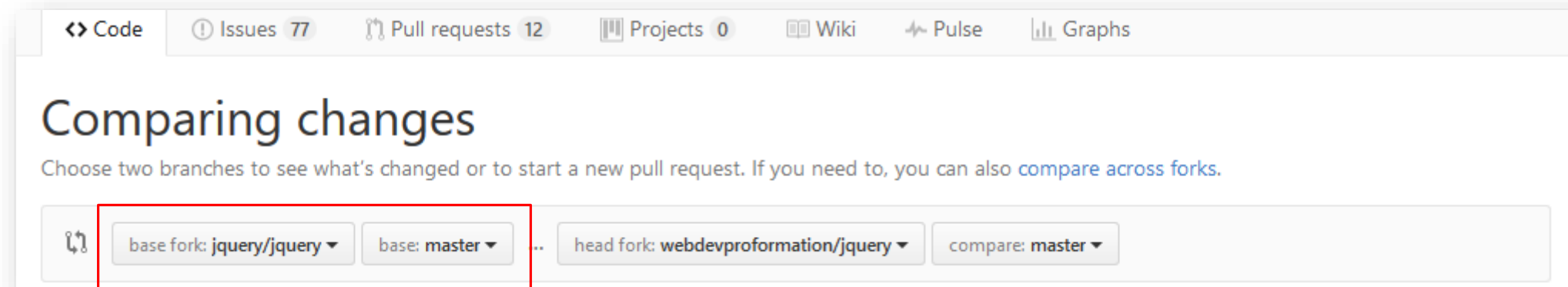
CONTRIBUER AU PROJET

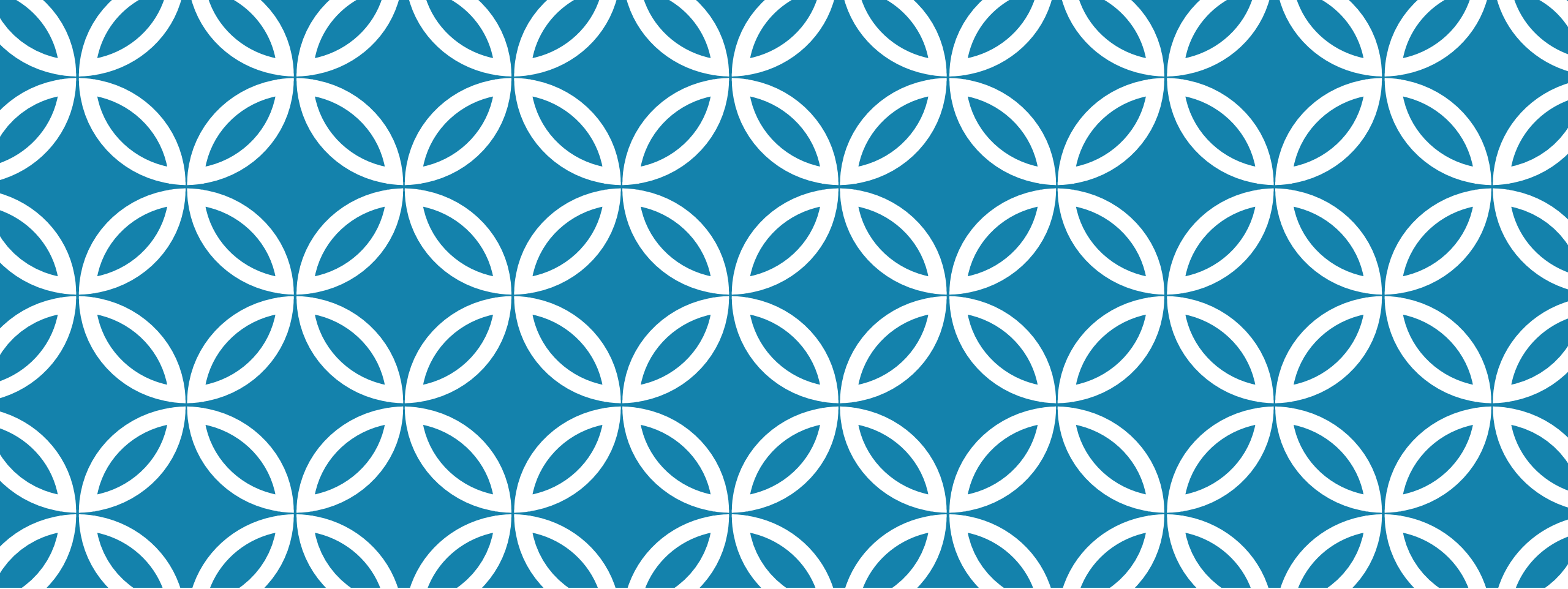
Clique droit Git Bash Here (pas besoin de créer un dossier)

Dans le terminal saisir :

- `git clone <votre url>`
- `cd <nom repository>`
- `git branch dev`
- `git checkout dev`
- Editer un fichier et le modifier
- `git commit -a -m "ma contribution"`
- `git push -u origin dev`
- Saisir login et mot de passe

PULL REQUEST AVEC LE PROJET





RÉFÉRENCE

SOURCE POUR SE FORMER

eBook (FR) : <https://git-scm.com/book/fr/v1/>

Grafikart (vidéos FR) : <https://www.grafikart.fr/formations/git>

Le Man en ligne (EN) : <https://www.kernel.org/pub/software/scm/git/docs/>

Site interactif avec toutes les commandes : <http://ndpsoftware.com/git-cheatsheet.html>

RESSOURCES INTÉRESSANTES

Générateur de couleurs harmonieuses : <http://flatuicolors.com/>

Générateur d'images fictives et libres de droits :

- <http://lorempixel.com/>
- <https://placeholder.com/>
- <https://unsplash.com/>