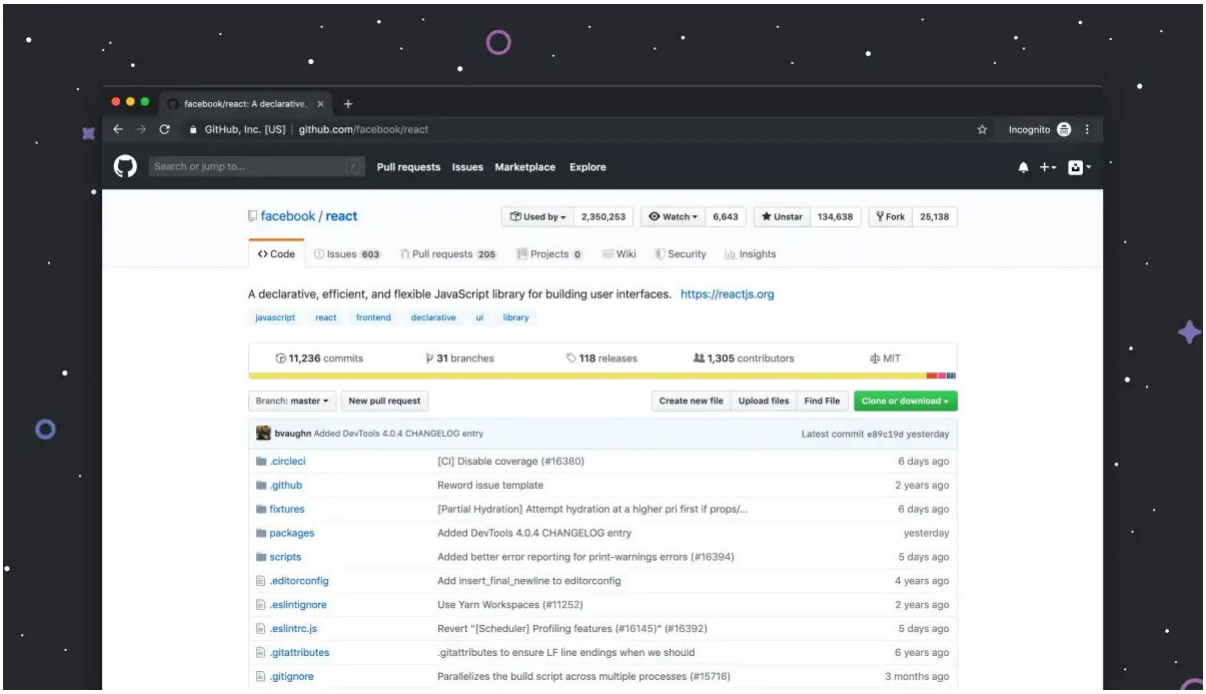
Qu'est-ce que Git & GitHub



Git peut être très difficile à comprendre si vous l'utilisez pour la première fois. Vous serez surement confronté à de nombreux problèmes lors de sa première utilisation. En travaillant régulièrement avec **Git et GitHub**, il vous sera beaucoup plus facile de travailler avec.

Git est un outil où les développeurs collaborent et travaillent ensemble, tandis que GitHub est un service d'hébergement de dépôt git. En termes simples, c'est un endroit où vit tout notre code. Nous pouvons créer un nombre illimité de dépôts (projets) sur GitHub.

Il y a deux façons d'utiliser git :

1. Ligne de commande (fortement recommandée)
2. Application graphique (par exemple : GitHub Desktop)

Avant d'aller plus loin, voyons ce qu'est réellement un git. Ce sera très amusant quand vous comprendrez le but réel de cet outil.

Supposons qu'une application web soit hébergée quelque part dans le monde. L'équipe qui a créé cette application web est composée de quatre membres principaux : **Bryan**, **Brice**, **Jordan** et **Marine**. Chacun d'entre eux possède sa propre copie du code source dans leur machine locale (ordinateur

portable/ordinateur de bureau). Imaginons que **Brice** ait apporté quelques modifications mineures à l'un des fichiers et qu'une pensée lui vienne soudainement à l'esprit.

« Hmm, il ne m'est pas possible d'aller informer mes coéquipiers chaque fois que j'ai apporté des modifications à ces fichiers. Alors, y a-t-il un outil qui peut faire cette charge de travail pour moi ? »

Eh bien, oui, il y en a un. Hourra ! C'est **Git → Système de contrôle de version**. En termes simples, Git garde juste la trace de notre code. Et GitHub est juste un endroit où vit notre code. Donc, maintenant que nous savons ce que sont Git et GitHub, voyons comment l'utiliser.

Note rapide :

- Dépôt signifie simplement dossier de projet où nos différents fichiers sont présents. C'est juste une convention de nommage que git utilise.
- Le dossier de projet qui vit sur les serveurs GitHub est appelé **dépôt distant**. Il est appelé distant parce que GitHub l'héberge quelque part dans le monde. Nous pouvons accéder à ce dépôt par une URL unique que GitHub nous fournit.

Par exemple, ceci

- Le dossier du projet qui vit sur nos machines locales est appelé **dépôt local**.

Voici toutes les commandes que nous allons apprendre :

- `git --version`
- `git init`
- `git remote add origin 'votre_url_de_dépôt_github'`
- `git remote`
- `git remote -v`
- `git status`
- `git diff`
- `git add 'nom de fichier'`
- `git add .`
- `git commit -m 'Commit initial'`
- `git push`
- `git config --global user.name 'Votre Nom'`
- `git config --global user.email 'Votre Email'`
- `git config --list`

Ne vous inquiétez pas. Nous verrons ce qu'est chaque commande dans la section suivante. La partie amusante commence... Alors, commençons :)

Comment utiliser Git & Github

Tout d'abord, téléchargez git à partir de ce lien officiel. Une fois le processus d'installation terminé, ouvrez **Git Bash** où nous allons écrire les commandes git. Si vous ne savez pas où trouver **Git Bash**, voici comment le trouver...

Allez sur le bureau →. Cliquez avec le bouton droit de la souris sur →. Sélectionnez l'option "Git Bash Here". Et c'est tout, cela va ouvrir un nouvel écran de terminal où nous allons écrire différentes commandes git.

Pour vérifier que git est correctement installé sur votre machine, exécutez la commande ci-dessous.

```
git --version
```

Avant de commencer, assurez-vous que vous avez bien suivi toutes les étapes ci-dessous :

- Visitez le site officiel de GitHub et inscrivez-vous (si vous ne l'avez pas déjà fait)
- Faites un nouveau dépôt (par exemple : Démo) en remplissant les détails.
- Créez un nouveau dossier (par exemple : Démo) sur votre ordinateur et créez un nouveau fichier "demo.txt" dans ce dossier.
- Ouvrez Git Bash et cd dans le dossier que vous avez créé à l'étape précédente.

Exemple

Supposons que vous ayez donné le même nom à votre dossier local et à votre dépôt GitHub, c'est-à-dire **Démo**.

Maintenant, pour utiliser les commandes git, nous devons nous assurer que le dossier que nous utilisons est un **dépôt Git**. Pour cela, utilisez la commande ci-dessous. Elle initialisera votre dossier local dans un dépôt git.

```
git init
```

Avertissement : Vous devez initialiser votre dossier local en tant que dépôt git, sinon les commandes git ne fonctionneront pas.

Vous avez peut-être remarqué une chose ici, c'est-à-dire que nous avons un fichier **demo.txt** vide dans notre dépôt local. Mais, dans le dépôt distant que vous avez créé, il n'y a pas de tel fichier. Hum, c'est étrange, non ? Eh bien, la réponse est simple. C'est parce que git ne sait pas où se trouve le dépôt distant. Pour l'instant, git va juste suivre les fichiers locaux parce qu'il n'a aucune connexion avec le dépôt distant. Alors, comment ajouter le dépôt distant ? C'est juste une question de commande.

```
git remote add origin
‘votre_url_de_dépôt_github’
```

Il ajoute simplement votre dépôt distant et donne le nom « **origin** ».

Maintenant, pour vérifier que votre dépôt distant a été ajouté correctement ou non, il suffit d'exécuter la commande ci-dessous.

```
git remote
```

Il donnera le nom du dépôt à distance. Par exemple, dans notre cas, il s'agirait de **origin**.

```
git remote -v
```

Il donnera le nom ainsi que l'URL du dépôt à distance. Par exemple, dans notre cas, il s'agirait de

```
origin ‘votre_url_de_dépôt_github’
```

C'est tout. Nous avons ajouté avec succès notre dépôt à distance. Passons à l'étape suivante.

Maintenant, disons que vous avez modifié le fichier **demo.txt** que vous avez créé plus tôt et que vous avez ajouté du texte (par exemple, "**Salut GitHub !!!**"). Git est assez intelligent pour suivre les modifications que vous avez apportées au fichier.

6. git status

Il vous montrera les fichiers de votre arborescence de travail et les fichiers de votre zone de transit. En termes simples, il vous indiquera exactement quels fichiers/dossiers ont été modifiés. Dans notre cas, le fichier **demo.txt** a été modifié.

7. git add ‘nom de fichier’

Il ajoutera votre dossier à la zone de transit. Par exemple, **git add demo.txt**

8. git add .

Si nous ajoutons un "." au lieu du nom de fichier, tous vos fichiers seront ajoutés à la zone de transit. Par exemple, **git add .**

Maintenant, vous vous demandez peut-être ce qu'est un arbre de travail et une zone de transit ? Alors, découvrons-le.

En gros, il y a deux concepts importants ici.

Arbre de travail

L'arbre de travail est le domaine dans lequel vous travaillez actuellement. C'est l'endroit où se trouvent vos fichiers. Cette zone est également connue sous le nom de **zone non tracée** de git. Si vous modifiez des fichiers dans votre arbre de travail, git reconnaitra qu'ils ont été modifiés, mais tant que vous n'aurez pas dit à git de **faire attention à ces fichiers**, il ne sauvegardera rien de ce qui s'y passe.

Zone de transit

La zone de transit est le moment où git commence à suivre et à enregistrer les modifications qui se produisent dans les fichiers. Vous dites à git que vous voulez suivre ces fichiers spécifiques, puis git dit OK et les déplace de votre arbre de travail vers la zone de transit et dit "**Cool, je connais ce fichier dans son intégralité**".

OK, maintenant que vous connaissez l'arbre de travail et la zone de transit, passons à la commande suivante.

```
git diff
```

Il montrera la différence entre un fichier dans la zone de transit et un fichier présent dans l'arbre de travail (**fichier non suivi**).

Comprenons **git diff** avec un petit exemple. Supposons que votre fichier "demo.txt" soit vide. Maintenant, lancez la commande **git add demo.txt** pour ajouter votre fichier à la zone de transit. Si notre fichier est ajouté à la zone de transit, git suivra tous les changements que nous avons faits. Ajoutez du texte dans votre fichier "demo.txt" et avant d'ajouter votre fichier à la zone de transit, lancez simplement la commande **git diff**.

```
git commit -m ‘Votre message de commit’
```

Il enregistrera vos modifications dans votre dépôt local. Il est généralement bon d'inclure un message de validation (commit) pour que vous sachiez quelles modifications vous avez apportées. Le drapeau **-m** est utilisé pour écrire votre message de validation.

En voici un autre. Si vous lancez la commande **git commit** sans sauvegarder vos variables de configuration, alors git renverra une erreur parce que git ne sait pas qui essaie de faire un commit. Pour cette raison, nous devrons ajouter les variables de configuration, c'est-à-dire le **nom et le courriel**. Alors, comment les ajouter ? C'est très simple. Il suffit d'exécuter les commandes ci-dessous et c'est tout.

```
git config --global user.name ‘Votre Nom’
git config --global user.email ‘Votre Email’
```

Comment vérifier que vos variables de configuration sont ajoutées au fichier de configuration ?

```
git config --list
```

Il énumère toutes les variables de configuration. Cherchez votre nom et votre adresse électronique dans cette liste.

Jusqu'à présent, tous les changements que nous avons effectués vivent dans notre dépôt local. N'oubliez pas que nous avons également créé un dépôt sur GitHub. Dans notre dépôt local, nous avons un fichier nommé **demo.txt,** mais il n'y a pas de tel fichier dans notre dépôt distant. Donc, comment faire passer les modifications que nous avons apportées dans notre dépôt local au dépôt distant ? Voyons voir.

(**NOTE** : Lorsque vous exécutez cette commande pour la première fois, elle ouvre une fenêtre vous demandant vos identifiants de connexion à GitHub, il suffit donc de les fournir)

```
git push origin main
```

Il se contentera de pousser tous les changements à partir du dépôt (**local → distant**). Rappelez-vous comment nous avons ajouté notre dépôt distant par la commande `git remote add origin 'votre_url_de_dépôt_github'`. Ainsi, la commande push indique simplement que tout ce que nous avons dans notre dépôt local doit être poussé dans le dépôt distant. Et dans notre cas, la commande remote est **origin**.

Après avoir lancé la commande **git push origin main**, il suffit de visiter votre dépôt GitHub distant. Vous remarquerez que toutes les modifications que vous avez apportées à votre dépôt local apparaissent également dans le dépôt distant. C'est la beauté de l'utilisation de **Git & GitHub.**