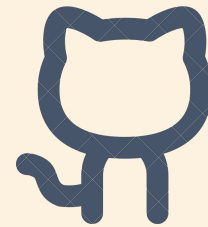


GIT ET GITHUB



FICHE TECHNIQUE

Qu'est-ce que Git :-



- Git est un système de contrôle de version.
- Git vous aide à suivre les modifications de code.
- Git est utilisé pour collaborer sur du code.
- Git et GitHub sont des choses différentes

Pourquoi Git :-

- Plus de 70 % des développeurs utilisent Git !
- Les développeurs peuvent travailler ensemble de n'importe où dans le monde.
- Les développeurs peuvent voir l'historique complet du projet.
- Les développeurs peuvent revenir à des versions antérieures d'un projet

Qu'est-ce que GitHub :-



- Git n'est pas la même chose que GitHub.
- GitHub crée des outils qui utilisent Git.
- GitHub est le plus grand hébergeur de code source au monde et appartient à Microsoft depuis 2018.

Configurer Git pour la première fois :-

```
$ git config --global user.name "<Entrez votre nom d'utilisateur ici>"
```

Fonctionnalités générales de Git :-

Initialisation de Git :-

```
$ git init
```

Git sait maintenant qu'il doit surveiller le dossier sur lequel vous l'avez lancé. Git crée un dossier caché pour garder une trace des modifications.

Fichiers intermédiaires/Ajout de fichiers au dépôt Git :-

Les fichiers intermédiaires sont des fichiers prêts à être validés dans le référentiel sur lequel vous travaillez. Lorsque vous ajoutez des fichiers à un référentiel vide pour la première fois, ils ne sont pas suivis. Pour que Git les suive, vous devez les préparer ou les ajouter à l'environnement intermédiaire.

```
$ git add <nom de fichier avec extension>
```

Transfert de tous les fichiers dans un dossier :-

```
$ git add --all
```

OU

```
$ git add -A
```

Faire un Commit :-

L'ajout de commits permet de suivre notre progression et nos modifications au fur et à mesure que nous travaillons. Git prend en compte chaque point de changement de commit ou « point de sauvegarde ». C'est un point du projet auquel vous pouvez revenir si vous trouvez un bogue ou si vous voulez apporter une modification. Lorsque nous nous engageons, nous devons toujours inclure un message.

```
$ git commit -m "Entrez votre message ici"
```

- **Git commit sans Stage (intermédiaire) :-**

Parfois, lorsque vous apportez de petits changements, l'utilisation de l'environnement de intermédiaire (staging) semble être une perte de temps. Il est possible de valider les modifications directement, en ignorant l'environnement intermédiaire.

```
$ git commit -a -m "Entrez votre message ici"
```

État des fichiers et du journal :-

```
$ git status
```

- État du fichier de manière plus compacte :-

```
$ git --short
```

- Journal d'un fichier :-

Log est utilisé pour afficher l'historique des validations d'un dépôt.

```
$ git log
```

```
$ git log --online
```

Git help :-

Si vous rencontrez des difficultés pour vous souvenir des commandes ou des options de commandes, vous pouvez utiliser Git help.

- Voir toutes les options disponibles pour la commande spécifique :

```
$ git <commande> -help
```

- Voir toutes les commandes possibles:

```
$ git help --all
```

Si vous vous retrouvez bloqué dans la vue de liste, MAJ + G pour sauter la fin de la liste, puis q pour quitter la vue.

Git branching :-

Dans Git, une branche est une nouvelle version du dépôt principal. Les branches vous permettent de travailler sur différentes parties d'un projet sans impact sur la branche principale. Une fois le travail terminé, une branche peut être fusionnée avec le projet principal. Nous pouvons même passer d'une branche à l'autre et travailler sur différents projets sans qu'ils n'interfèrent les uns avec les autres.

- **Création d'une nouvelle branche Git :-**

```
$ git branch <nom de la branche>
```

- **Vérification de toutes branches disponibles :-**

```
$ git branch
```

- **Passer à d'autres branches :-**

```
$ git checkout <nom de la branche>
```


- **Créer une nouvelle branche et y passer directement :-**

```
$ git checkout -b <nom de la branche>
```

- **Suppression d'une branche :-**

```
$ git branch -d <nom de la branche>
```

- **Fusion de deux branches :-**

Il est préférable de changer/passer en branche principale avant la fusion de toute autre branche avec elle.

```
$ git merge <nom de la branche>
```

Cela fusionnera la branche spécifiée avec notre branche principale.

Travailler avec GitHub :-

Créez un compte github pour créer vos dépôts distants. Maintenant, créez un nouveau dépôt dans lequel nous téléchargerons nos fichiers à partir du dépôt local.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

Limssly


Repository name *

kounama


✓ kounama is available.

Great repository names are short and memorable. Need inspiration? How about [refactored-sniffle](#) ?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

Remarque : Le référentiel local désigne le repository qui est sur notre système alors que le dépôt à distance. signifie celui qui est sur un autre système/serveur distant, par exemple. - GitHub, GitLab, Bitbucket, etc.

Envoyer un dépôt local à GitHub :-

Copiez l'url ou le lien du dépôt que nous venons de créer. Par exemple, cela devrait ressembler à ceci :

Collez l'url copiée dans la commande git ci-dessous.

```
$ git remote add origin <collez l'URL copiée ici>
```

'git remote add origin <URL>' spécifie que nous ajoutons un dépôt distant, avec l'URL spécifiée, en tant qu'origine à notre dépôt Git local.

Enfin, en poussant notre branche principale (master) vers l'URL d'origine (dépôt distant) et en la définissant comme branche distante par défaut.

```
$ git push --set-upstream origin master
```

Retournez dans GitHub et vérifiez que le dépôt a été mis à jour.

- **Pousser le dépôt local vers github après avoir effectué le processus ci-dessus au moins une fois :-**

Tout d'abord, validez toutes les modifications via un commit. Ensuite, poussez toutes les modifications vers notre origine distante, c'est-à-dire le dépôt distant sur GitHub

```
$ git push origin
```

Extraire le référentiel local de GitHub :-

Git pull est utilisé pour extraire toutes les modifications d'un dépôt distant dans la branche sur laquelle nous travaillons. C'est une combinaison de fetch et de merge. Utilisez-le pour mettre à jour votre Git local.

```
$ git pull origin
```

Extraire la branche de GitHub :-

Tout d'abord, vérifiez quelles branches nous avons et où travaillons nous en ce moment par la commande `'git branch'`. Comme nous n'avons pas la nouvelle branche sur notre Git local qui doit être extrait du GitHub. Ainsi, pour voir toutes les succursales locales et distantes, utilisez

Envoyer (push) une branche vers GitHub :-

Tout d'abord, créons une nouvelle branche locale que nous allons pousser vers GitHub. Entrez la commande sous la forme `'git checkout -b <nom de la branche>'`. Vous pouvez vérifier l'état des fichiers dans cette branche actuelle en utilisant `'git status'`. Validez toutes les modifications non validées pour tous les fichiers de cette branche en utilisant `'git commit -a -m <Message>'`. Maintenant, faire un push cette branche de notre dépôt local vers GitHub en utilisant `'git push origin <nom de la branche>'`.

Cloner un dépôt à partir de GitHub :-

Nous pouvons cloner un dépôt forké de GitHub sur notre dépôt local. Un clone est une copie complète d'un référentiel, y compris toutes les journalisations et versions des fichiers. Revenez au dépôt d'origine et cliquez sur le bouton vert « **Code** » pour obtenir l'URL à cloner. Copiez l'URL.

Maintenant, dans git bash, entrez la commande suivante pour cloner le dépôt copié sur votre machine locale

```
$ git clone <URL copiée>
```

Pour spécifier un dossier spécifique à cloner, ajoutez le nom du dossier après l'URL du référentiel, comme ceci :

```
$ git clone <URL copiée> <nom du dossier>
```

Merci d'avoir lu !!!

Partagez vos précieux commentaires ou suggestions dans la communauté si vous avez trouvé ce contenu utile.

