Apprendre Git

Git est un logiciel de versioning qui permet de gérer l'évolution de sa code base. Git tient compte des changements effectué dans le code et vous permet d'enregistrer ses changements. Une fois les changements enregistrés il est possible de revenir en arrière en cas de soucis.

On appelle l'enregistrement d'un changement un *commit*.

Git prends tout son sens lors de projet de groupe car chaque contributeurs du projet va pouvoir commit les changements qu'il a effectué sur la code base et ainsi les rendres disponibles aux autres contributeurs. Un projet git s'appelle un *repository* ou *répo*

Lorsque plusieurs développeurs travails sur le même projet on dit qu'ils *commits sur le même répo*.

Par défaut git vous permet de versionner votre code en local sur votre ordinateur mais pour rendre le répo disponible aux autres contributeurs il est d'usage d'heberger la version principale du répo sur des plateformes en ligne comme GitHub (Microsoft) ou GitLab (License MIT open source).

Fondamentaux de git

Le workflow git est toujours plus ou moins le même :

1. Coder un changement dans votre code.
2. Dire à git quels fichiers prendre en compte pour le prochain commit. (*git add*)
3. Commit le changement. (*git commit*)
4. Et ansi de suite ...

Une fois satisfait de ces changements vous pouvez pousser (*push*) vos commits sur GitHub ou GitLab pour les rendres accessibles au autres devs du projet, nous verrons cette étape plus tard une fois les fondamentaux du versionning assimilés.

Pré-requis et mise en place du dossier de travail

Pour apprendre git partont d'un dossier vide sur notre ordinateur contenant un simple script javascript.

Créez un dossier de travail nommé `learn_git` et ouvrez VSCode à partir de ce dossier, puis créez un fichier appelé `app.js` contenant quelques lignes de code.

```
mkdir learn_git     # Création du dossier
cd learn_git        # On se déplace dans le dossier
code .              # On ouvre VSCode dans le dossier courant
```

~/learn_git/app.js

```
const word = "Hello";
console.log(word);
```

Installer git

```
apt update && apt upgrade
apt install git
```

> Sous Windows telechargez le depuis le site officiel https://git-scm.com/download/win

Initialiser le projet git

Avant d'utiliser git dans un dossier il faut initialiser git avec la commande : *git init*.
~/learn_git/

```
git init
```

```
Initialized empty Git repository in /root/learn_git/.git/
```

Effectuer un commit

Un commit se passe en deux temps :

- Définir les fichiers à commit (git add)
- Commit (git commit)

Définir les fichiers a commit : `git add`

Si vous êtes dans à la racine du repo :

```
git add .
```

> `.` signifie "la où je suis", c'est à dire à la racine du dossier.
> `git add .` permet donc d'ajouter tout le fichiers du dossier pour le prochain commit.

Lancer le Commit : `git commit -m`

Un commit doit toujours être accompagné d'un message. Le message s'écrit avec le paramètre `-m` de la commande git commit

```
git commit -m "Mon premier commit"
```

Voilà un commit à été effectué et une première version de notre logiciel existe à present.

Push son répo sur GitHub

Pour mettre sur GitHub notre répo il y a deux façon :

- Cloner un repo existant avec `git clone url` (le plus courant)
- Définir l'url du répo disstant à utiliser pour push, on appelle ça définir la `remote origin` .

Cloner un repo

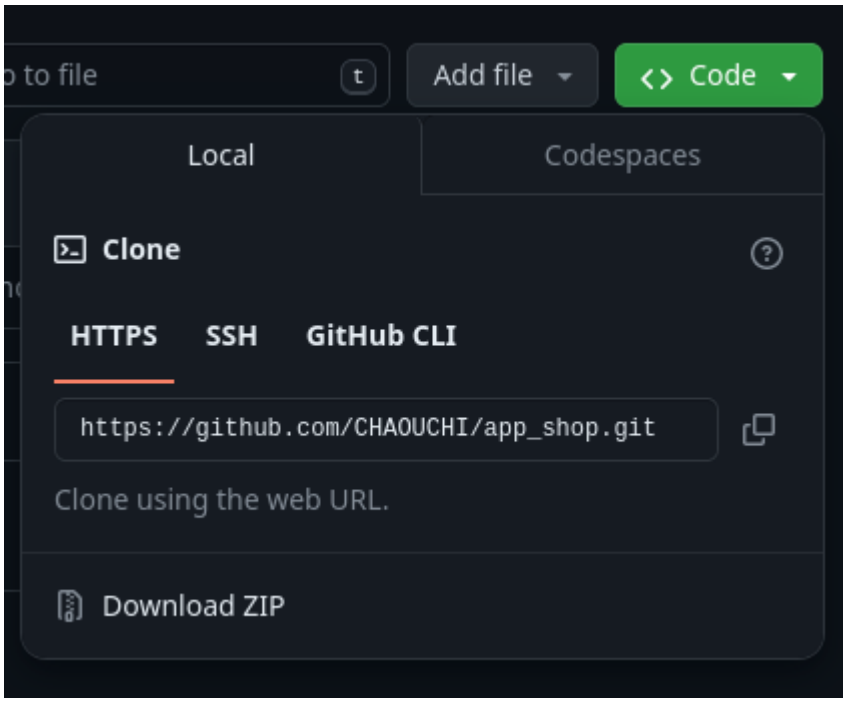
Le plus souvent vous travaillerez sur un répo déjà existant, et en rêgle général je vous recommande de créer votre répo directement sur GitHub plutôt que de faire un `git init` en local.

Pour récuéprer un répo GitHub,

```
git clone https://github.com/VotreIdGitHub/le_nom_du_repo.git
```

Pour récupérer l'url du répo rendez-vous sur GitHub.com et copiez-collez l'url du répo.

Ici je vois l'url d'un repo GitHub nommé app_shop sur le GitHub de CHAOUCHI (moi)



> La commande pour cloner le répo précedent sera donc
>
> ```
> git clone https://github.com/CHAOUCHI/app_shop.git
> ```

Push après avoir cloner

Pour push des commit il faut toujours :

- Pull(tirer) le répo pour s'assurer d'avoir la dernière version
- add et commit les changements
- puis push

```
git pull
git add .
git commit -m "Nouveau changement dans le code"
git push
```

Définir une origine à la main

Si vous avez déjà un répo local que vous souhaitez placer sur GitHub il vous faut :

1. Créer un répo sur GitHub.com
2. Copiez-collez son url

3. Définir cette url comme `remote origin` du répo local, c'est à dire le définir comme le répo distant de référence du répo local.

```
git remote add origin https://github.com/VotreIdGitHub/le_nom_du_repo.git
```

La commande `git remote add` permet de donner un nom au répo distant, ici nous l'appelons `origin` .

Une fois ceci fait on peut push notre code sur le répo

Push après avoir add une remote origin

```
git push origin main # Je push sur l'orgin la branch main
```

Travailler à plusieurs grâce aux branches et les Pull request

L'avantage de git est sa capacitée à créer des branches alternatives du projet.

Par défaut il n'y à qu'une seul branche sur un repo git, elle se nomme main et contient la version principale de votre logiciel.

Vous pouvez la voir grace à la commande `git branch`

```
git branch
```

```
massinissa@massinissa-ThinkPad-T440p:~/Git/app_shop$ git branch
* main
```

Pour travailler à plusieurs il vous faudra créer un nouvelle branche que vous utiliserez le temps de coder votre fonctionnalité.

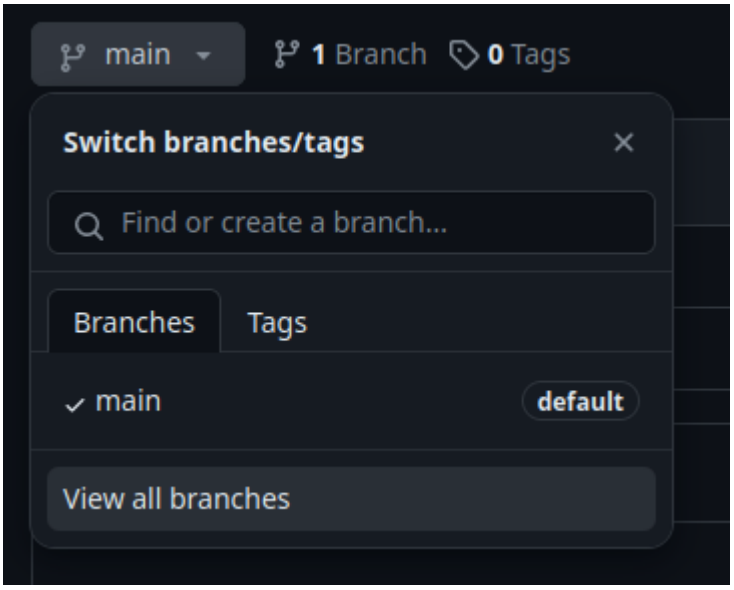
Une fois la fonctionnalité terminé vous pourrez fusionner (merge) la nouvelle branche avec la branche `main` pour integrer la nouvelle fonction au logiciel.

1. Créer une nouvelle branche
2. Coder quelque chose
3. Commit les changement sur la branche
4. Merge la branche avec le main

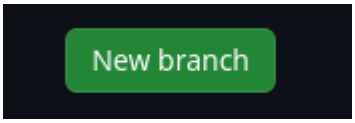
Créer une branche

Meme si il est possible de créer une branche avec la commande `git branch nom_de_la_branch` je vous recommande de la créer directement sur GitHub.com.

1. Cliquez sur "View all Branches" dans l'onglet des branches sur GitHub



2. Puis sur le bouton "New branch"



3. Nommez votre branche par le nom de la nouvelle fonctionnalité que vous voulez ajouter à votre logiciel, par exemple si vous codez un serveur web nommez la "web_server"

Pull la nouvelle branche et changer de branche

Une fois la branche créer vous devez passez de la `main` à la branche `web_server` avant de pouvoir coder vos changement et push vos commit, sinon vous risquez d'envoyer vos commit sur la branche main !

Changer de branche se fait avec la commande : `git checkout .`

```
# Je met à jour mon répo local pour prendre connaissance de la nouvelle branche
git pull
git checkout web_server  # Je change de branche pour la branche web_server
```

> Avec la commande `git branch -a` vous pouvez voir que la branch web_server est définie en remote (à distance).
> Une fois la commande `git checkout` faite, la branche apparait en local à l'appel de la commande `git branch`

```
massinissa@massinissa-ThinkPad-T440p:~/Git/app_shop$ git branch
  * main
massinissa@massinissa-ThinkPad-T440p:~/Git/app_shop$ git branch -a
  * main
    remotes/origin/HEAD -> origin/main
    remotes/origin/main
    remotes/origin/web_server
massinissa@massinissa-ThinkPad-T440p:~/Git/app_shop$ git checkout web_server
  Branch 'web_server' set up to track remote branch 'web_server' from 'origin'.
  Switched to a new branch 'web_server'
massinissa@massinissa-ThinkPad-T440p:~/Git/app_shop$ git branch
    main
  * web_server
massinissa@massinissa-ThinkPad-T440p:~/Git/app_shop$
```

Vous pouvez maintenant effectuer vos commits directement sur cette branche jusqu'à ce que votre fonctionnalité soit fini de codé.

Exemple de server web nodejs dans app.js

```
const express = require("express")
const cors = require("cors")

const app = express()

app.get("/",(req,res)=>{
    res.json("Hello World")
});

app.listen(3000);


# Après avoir codé votre fonctionnalité...
git pull # Au cas où je pull les changements de cette branche
git add . # J'ajoute mes changements pour le prochain commit
git commit -m "Server web express mit en place sur le port 3000"
git push # Je push mon commit sur la branche web_server
```

Maintenant que ma fonctionnalité est fini de coder il faut merge la branche `web_server` dans la branche main pour ajouter la fonctionnalité au projet principal.

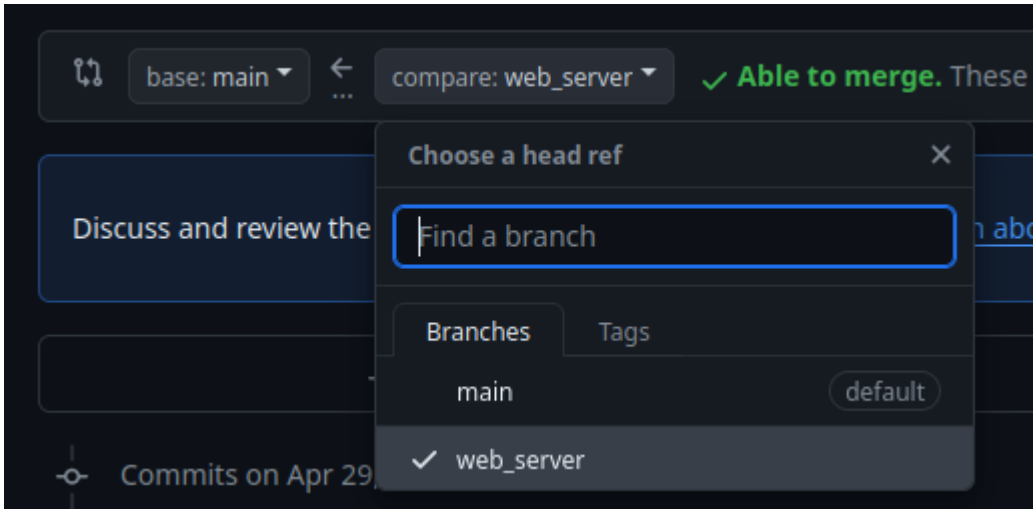
Fusionner les deux branches : merge

Pour fusionner les changements de la branche `web_server` vers la branche main la bonne pratique est de faire une demande de pull request sur le Projet GitHub.

1. Allez dans l'onglet Pull Request du repo

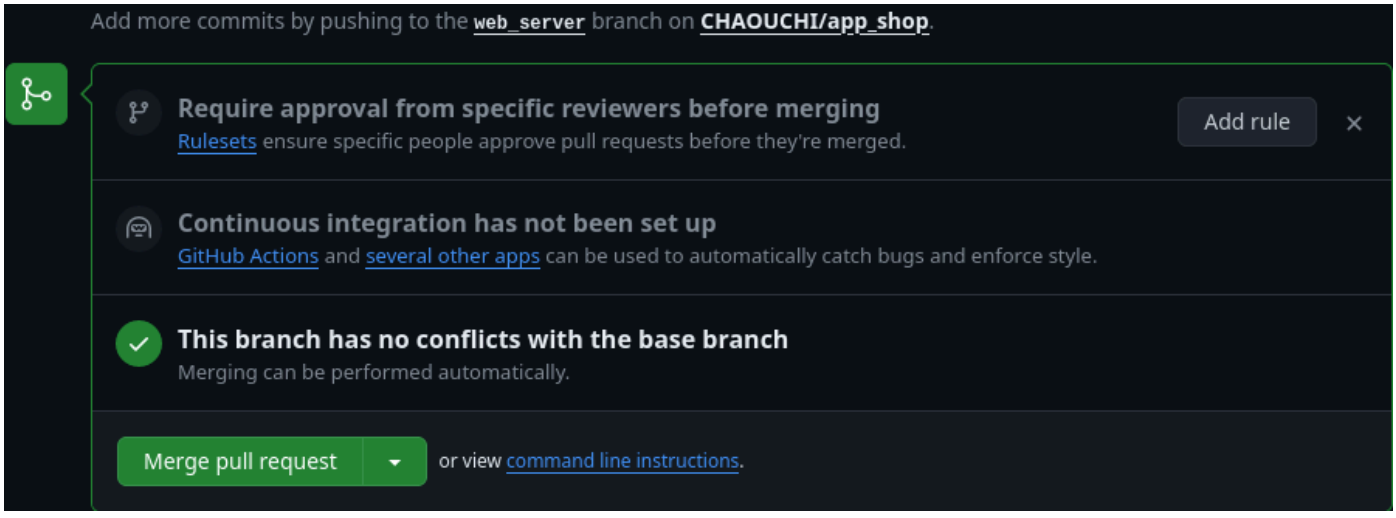
   Pull requests
2. Créez une nouvelle pull request avec le bouton "New pull request"

   New pull request
3. Dans le **deuxième menu déroulant** choisissez votre branche comme la branche à fusionner.

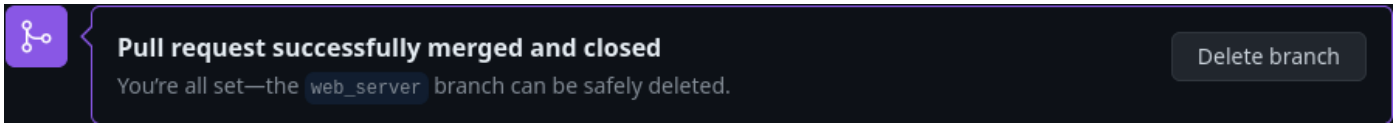


4. Créez la pull request

   Create pull request
5. Vos collègues peuvent commenter votre code avant de valider le merge avec le bouton "Merge pull request"



6. Une fois le merge validé vous pouvez supprimer la branch rémanante.



> Pour merge une branche je peux aussi :
>
> ```
> git checkout main  # Changer de branche pour la branche main
> git merge web_server # Merge ma branche web_server dans main
> git push origin main # Push les changements sur GitHub
> ```
>
> Cette méthode fonctionne bien lorsque l'on travail seul mais le système des `pull request` permet à l'équipe de dev de s'assurer de la qualité du code merge avant de valide la fusion.