

ADRAR

1

SEPTEMBRE 2020

**Qu’est-ce que GIT ?**

Git est un outil de gestion de projets et de versions collaboratif.

**Git** est un [logiciel de gestion de versions](https://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions) [décentralisé](https://fr.wikipedia.org/wiki/Gestion_de_versions#Gestion_de_versions_d.C3.A9centralis.C3.A9e). C'est un [logiciel libre](https://fr.wikipedia.org/wiki/Logiciel_libre) créé par [Linus Torvalds](https://fr.wikipedia.org/wiki/Linus_Torvalds), auteur du [noyau Linux](https://fr.wikipedia.org/wiki/Noyau_Linux), et distribué selon les termes de la [licence publique générale GNU](https://fr.wikipedia.org/wiki/Licence_publique_g%C3%A9n%C3%A9rale_GNU) version 2. En 2016, il s’agit du [logiciel de gestion de versions](https://fr.wikipedia.org/wiki/Logiciel_de_gestion_de_versions) le plus populaire qui est utilisé par plus de douze millions de personnes.

ADRAR

2

SEPTEMBRE 2020

**GIT download** :

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

**Installation GIT** :

Après l’installation

1 Lancer Git Bash

2 Enregistrer le login :

git config --global user.name "***nom\_utilisateur***"

3 Enregistrer son adresse mail :

git config --global user.email ***mail\_utilisateur***

4 Pour vérifier que sa configuration est correcte :

git config -–list

5 Activer la coloration :

$ git config --global color.diff auto

$ git config --global color.status auto

$ git config --global color.branch auto

6 Configurer son éditeur de texte par défaut et Vimdiff comme outil de merge :

pour Notepad++

pour Visual studio code outils de merge

7 Accéder à votre répertoire (dépôt) :

Taper cd puis l’url de votre répertoire par ex :

$ cd ***C:/Users/nom\_utilisateur/Desktop/git***

8 Initialiser le répertoire :

$ git init

ADRAR

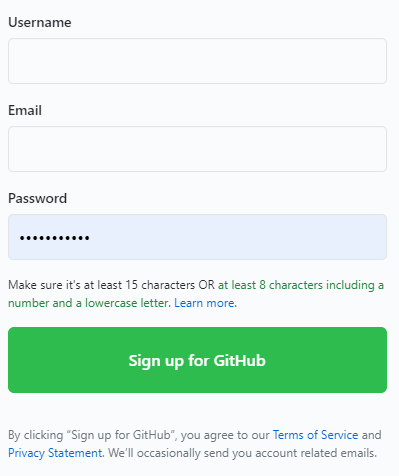
3

SEPTEMBRE 2020

$ git config --global core.editor ***notepad++***

$ git config --global core.editor ***code***

$ git config --global merge.tool ***vimdiff***



**Créer un compte GitHub** :

1 Se connecter sur github.com :

[https://github.com](https://github.com/)

2 Cliquer sur Sign up en haut à droite :

Renseigner un Username *(nom utilisateur*)

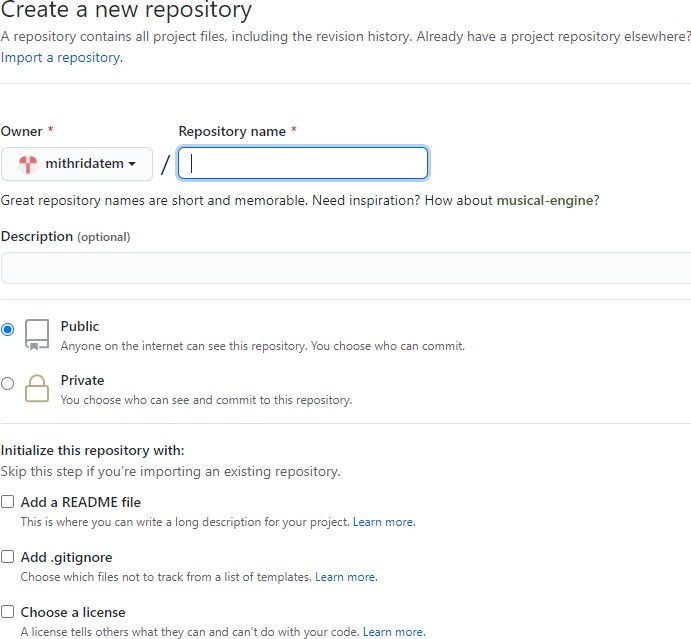
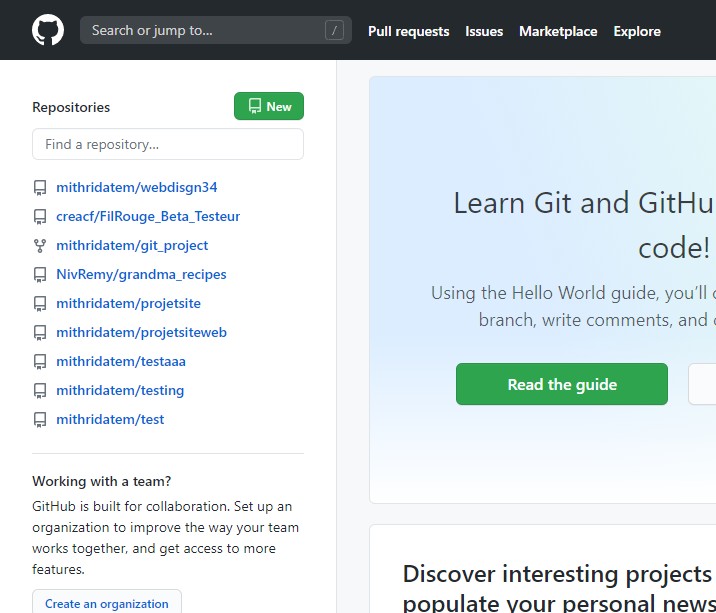
Renseigner un Email (*adresse mail*)

Renseigner un Password (*mot de passe*)

ADRAR

4

SEPTEMBRE 2020



**Créer un repositery sur GitHub** :

1 Cliquer sur new :

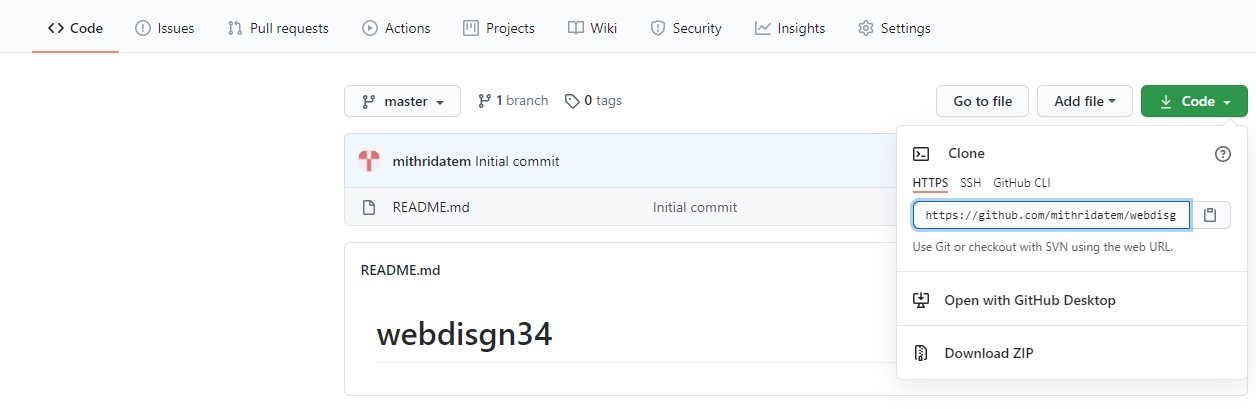
2 Choisir un nom pour votre dossier :

3 Cliquer sur create repositery en bas de page (bouton)

ADRAR

5

SEPTEMBRE 2020



**Cloner votre dépôt Github** :

1 Cliquer sur code

2 Récupérer l’url de votre dépôt (https)

3 Taper la ligne de commande :

git remote add clone ***https://github.com/votre\_url.git***

ADRAR

6

SEPTEMBRE 2020

**Système de branches :**

Le principal atout de Git est son système de branches

La branche principale est appelée la branche **master**. C’est celle-ci, où au final, vous aurez à la fin toutes vos

modifications. Le but est de ne surtout pas réaliser les modifications directement sur cette branche, mais de réaliser les modifications sur d’autres branches, et après tests, les intégrer sur la branche **master**.

**Branche Master**

**Branche Secondaire**

1 afficher les branches :

git branch

Cela va s’afficher comme ci-dessous : (*git branch*)

git branch

\*master

2 Créer une branche :

git branch ***nom de la branche*** (secondaire) 3 pour basculer sur la branche secondaire :

git checkout ***nom de la branche***

Cette commande va afficher la liste des branches avec une astérisque sur la branche sélectionnée.

La branche va fonctionner comme un dossier virtuel. Avec la commande **Git checkout**, on va être téléporté dans le dossier virtuel ***nom de la branche*** « (secondaire) ».

4 sauvegarder les modifications :

Vous avez réalisé des évolutions sur la branche ***nom de la branche*** et il va falloir maintenant demander à Git de les enregistrer.

Pour cela on va utiliser la commande commit.

git commit -m ***“description de la modification”***

ADRAR

7

SEPTEMBRE 2020

**Ajouter un ou plusieurs fichiers à une branche et modifications :**

On utilise la commande git add pour ajouter un fichier. **(*Le fichier doit être déjà créé dans le répertoire*)**

git add ***PremierFichier.txt***

On utilise la commande git commit pour sauvegarder notre fichier.

git commit -m ***«nom du commit***»

git commit

On vous demande alors d'indiquer le message du commit puis de valider. Pour valider le message, une fois que vous l'avez écrit, appuyez sur Echap (votre curseur va basculer sur la dernière ligne) et tapez ***:x!*** .

Cette commande va sauvegarder et quitter l'éditeur des messages de commit.

**Ajouter plusieurs fichiers à une branche :**

On utilise la commande git commit pour sauvegarder tous nos fichiers.

git commit -am ***«nom du commit***»

**Création d’une remise :**

1 Voir l’état des fichiers :

git status

2 Créer une remise :

Une remise permet de mettre de coté des modifications pour pouvoir les copier sur une autre branche.

Modifier vos fichiers puis taper la commande.

git stash

3 Créer une nouvelle branche :

git branch brancheCommit (la nouvelle branche s’appelle brancheCommit)

4 Basculer sur la nouvelle branche (brancheCommit) :

git checkout brancheCommit

5 Application de la remise :

Cette commande va appliquer toutes les modifications précédentes (Créer une remise) sur la branche sélectionnée

(brancheCommit)

git stash apply

6 voir la liste des remises :

git stash list

Cela permet de voir la liste des remises avec un identifiant associé à chacune

7 Application d’une remise avec son identifiant :

git stash apply stash@{0}

Cette commande applique sur la branche sélectionné la remise « ***stash@{0}*** » (la première dans ce cas)

ADRAR

8

SEPTEMBRE 2020

8 Restaurer la branche master :

Si vous avez effectué un commit (update des fichiers) non désiré on peut restaurer nos fichiers à un état antérieur.

Pour se faire on va lancer la commande suivante pour récupérer l’id du commit :

-

Se positionner sur la branche master.

git checkout brancheCommit

Afficher le log des commits pour récupérer l’identifiant.

git log

-

-

$ git log

commit ***ca83a6dff817ec66f443420071545390a954664949*** *(identifiant*)

Author: ***nom\_utilisateur*** <***mail\_utilisateur***> Date: Mon Mar 19 21:52:11 2019 -0700

Récupérer l’identifiant ci- dessus

Assurez vous que vous êtes bien sur la branche master Taper la commande ci-dessous :

git reset --hard HEAD^

Basculer sur la branche ou vous voulez copier les modifications :

git checkout brancheCommit

Taper la commande ci-dessous : (on n’a besoin que des 8 premiers caractères de l’identifiant)

git reset --hard ***ca83a6df***

Cela va appliquer un commit sur la branche sélectionnée (brancheCommit)

-

-

-

-

-

-

-

-

-

-

1. Modifier le message du dernier commit :

git commit --amend -m "***Votre nouveau message de commit***"

1. Ajouter un fichier oublié dans un nouveau commit :

Vous avez fait votre commit mais vous réalisiez que vous avez oublié un fichier. Ce n'est pas bien grave ! Nous allons

réutiliser la commande git --amend, mais d'une autre manière. La fonction git --amend, si vous avez bien compris, permet de modifier le dernier commit.

Nous allons donc réutiliser cette fonction, mais sans le -m qui permettait de modifier son message.

Nous allons dans un premier temps ajouter notre fichier, et dans un deuxième temps réaliser le git --amend.

git add ***nom du fichier***

git commit --amend --no-edit

Votre fichier a été ajouté à votre commit et grâce à la commande --no-edit que nous avons ajoutée, nous n'avons pas modifié le message du commit.

Pour résumer, git commit --amend vous permet de sélectionner le dernier commit afin d'y ajouter de nouveaux

changements en attente. Vous pouvez ajouter ou supprimer des changements afin de les appliquer avec

*commit --amend*.

Si aucun changement n'est en attente*, --amend* vous permet de git commit --amend -le dernier

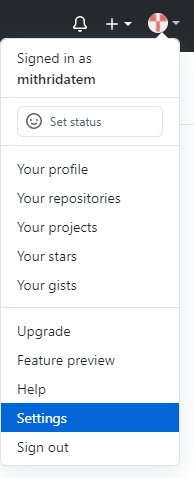
message de log du commit avec *-m*. exemple ci-dessous :

git commit --amend -m « ***mon message*** » (voir section 9 Modifier le message du dernier commit )

ADRAR

9

SEPTEMBRE 2020



**Sauvegarder vos fichiers sur un dépôt distant (ex Github) :**

Pour sauvegarder vers un dépôt distant :

1 Récupérer l’url du dépôt (voir cloner votre dépôt sur Github page 5) :

2 Saisir la commande suivante :

git push ***url de votre dépôt github***

Cela va ouvrir une fenêtre qui va vous demander de saisir vos identifiants login ou mail et mot de passe.

Si vous trompez de mot de passe ou d’identifiant relancer la commande.

NB : *vos identifiants et mot de passe seront sauvegardés*.

**Corriger un mauvais commit à distance :**

1 Corriger un mauvais commit (le dernier) envoyé avec un push sur votre Github saisir la commande ci-dessous :

git revert HEAD^

**Réparer l’accès à distance :**

Git base toute sa gestion d’authentification sur le mécanisme des clés SSH. Ce système est d’ailleurs immensément

utile de façon générale sous Linux, Unix et OSX, dès qu’il s’agit de s’authentifier sur une machine tierce.

1 Génération des clés SHH :

ssh-keygen -t rsa -b 4096 -C "adresse\_mail" saisir l’adresse mail créé au début.

Vous pouvez la retrouver à cette adresse :

**C:\Users\VotreNomD'Utilisateur\.ssh**, et d'afficher les dossiers masqués.

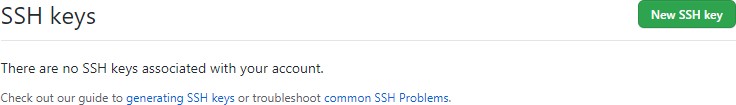
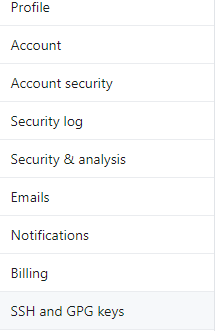
2 Ajouter vos clés SSH à Github :

-Se connecter à github.com puis allez sur votre icone de profil (en haut à droite) et cliquez sur settings :

ADRAR

10

SEPTEMBRE 2020



-Dans la fenêtre de profil cliquez sur SSH and GPG keys dans le menu à gauche :

-Cliquez sur new SSH Key :

Copier-coller le contenu du ficher situer à cette adresse **C:\Users\VotreNomD'Utilisateur\.ssh**

Et donner un titre :

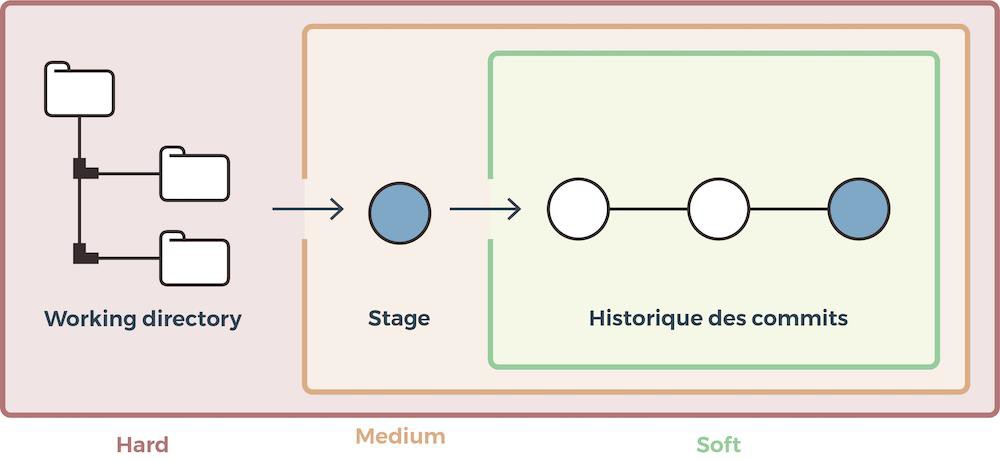
Cliquer sur Add SSH key.

Vous devrez ensuite confirmer votre mot de passe, et votre clé SSH sera alors ajoutée à votre compte GitHub.

ADRAR

11

SEPTEMBRE 2020



**Les 3 types de réinitialisations de GIT :**

La commande git reset est un outil complexe et polyvalent pour **annuler les changements**. Elle peut être appelée de trois façons différentes, qui correspondent aux arguments de ligne de commande **--soft, -- mixed et --hard**.

1 Commande Reset –hard :

Cette commande permet de revenir à n’importe quel commit mais il faut faire très attention car l’on **perd tous les**

**commits suivants !!!**

Pour se faire il nous faut l’identifiant du commit que l’on obtient avec la commande :

$ git log

Puis l’on saisit la commande suivante :

git reset ***identifiant\_commit*** --hard

Cette commande est à utiliser en dernier recours car l’on **perd tous les commits suivants** !!!!

2 Commande Reset –mixed :

Le git reset --mixed va permettre de revenir juste après votre dernier commit ou le commit spécifier, sans supprimer vos modifications en cours. Il va par contre créer un HEAD détaché. Il permet aussi, dans le cas de fichiers indexés mais pas encore commités, de désindexer les fichiers.

Si rien n'est spécifié après git reset, par défaut il exécutera un git reset --mixed HEAD~

3 Commande Reset –soft :

Le git reset --Soft permet juste de se placer sur un commit spécifique afin de voir le code à un instant donné ou créer une branche partant d'un ancien commit. Il ne supprime aucun fichier, aucun commit, et ne crée pas de HEAD détaché.

ADRAR

12

SEPTEMBRE 2020

**Sources :**

<https://fr.wikipedia.org/wiki/Git>

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

[https://openclassrooms.com/fr/courses/5641721-utilisez-git-et-github-pour-vos-projets-de-](https://openclassrooms.com/fr/courses/5641721-utilisez-git-et-github-pour-vos-projets-de-developpement/5641728-decouvrez-la-magie-du-controle-de-versions)

[developpement/5641728-decouvrez-la-magie-du-controle-de-versions](https://openclassrooms.com/fr/courses/5641721-utilisez-git-et-github-pour-vos-projets-de-developpement/5641728-decouvrez-la-magie-du-controle-de-versions)

ADRAR

13

SEPTEMBRE 2020