# Mémo des commandes git utiles

ITS

4 novembre 2020

## 1 Ajouter une clé SSH à Github

### 1.1 Générer une clé SSH

1. Crée une clée ssh :
**ssh-keygen -t rsa -b 4096 -C "your_email@example.com"**

1. Lorsqu'on vous demande "Enter a file in which to save the key", entrer **~/.ssh/id_rsa_github**

2. Lorsque la demande de passphrase apparait, vous pouvez appuyez sur la touche "entrée".

4. Ouvrer le fichier de config avec vi **vi ~/.ssh/config**. 5. Pour pouvoir modifier le fichier tapper **i** et entrer les configuration suivantes :

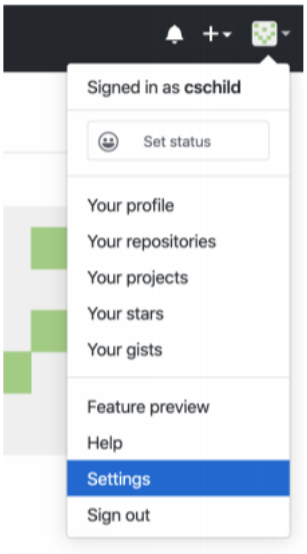
```
Host github.com
        HostName github.com
        User git
        IdentityFile ~/.ssh/id_rsa_github
        IdentitiesOnly yes
```

6. Appuyer sur la touche **esc** pour quitter le mode d'édition. 7. Pour enregistrer les modifications et quitter **vi** tapper **:wq** (**w** enregistre les modifications **q** quitte **vi**)

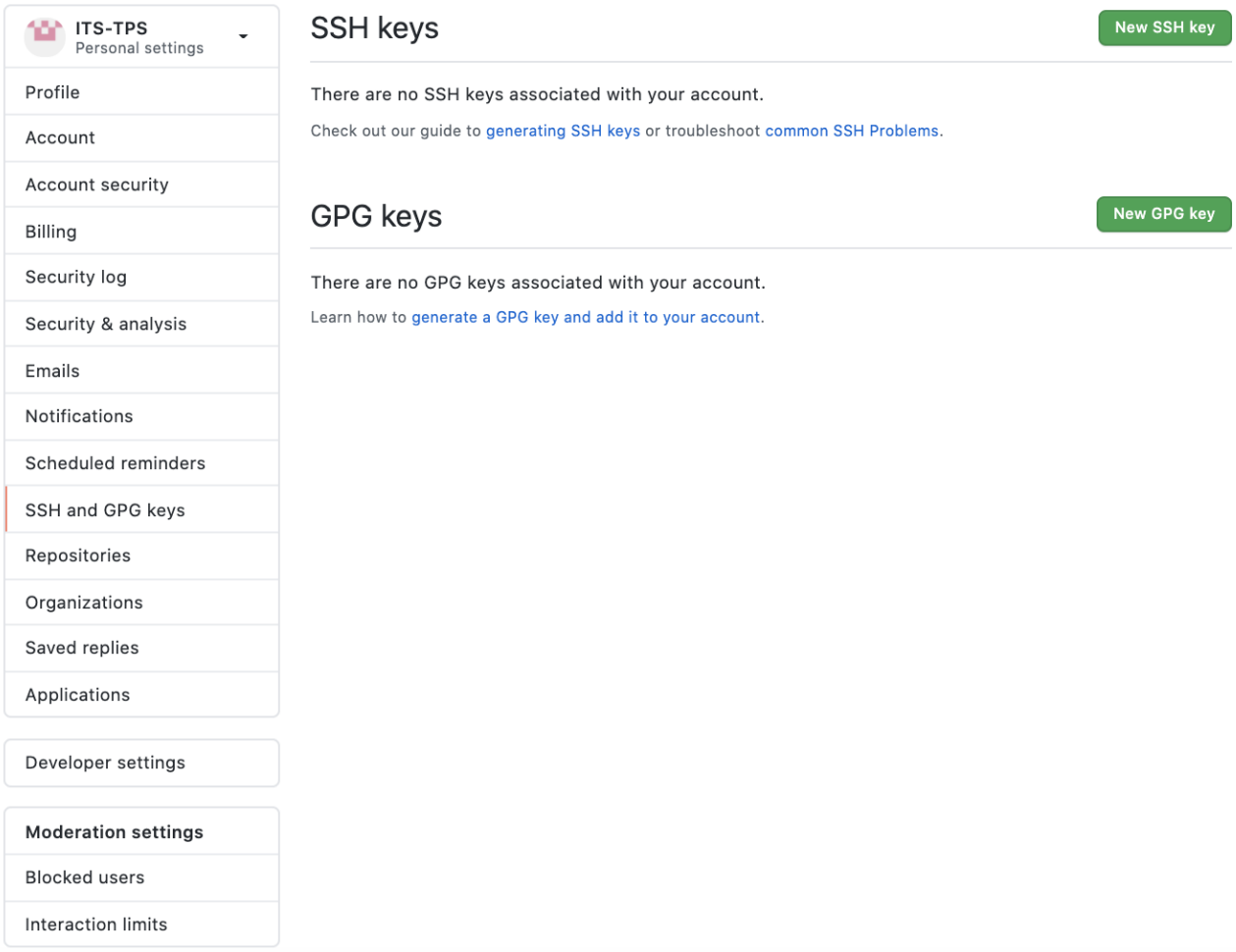
### 1.2 Ajouter une clé SSH à votre compte GitHub

1. Copier la clé publique dans le presse papier avec **pbcopy < ~/.ssh/id_rsa_github.pub** (si cela ne marche pas afficher la clé avec **cat ~/.ssh/id_rsa_github.pub** et copier tout ce qui est affiché) 2. Dans le coin droit de la page GitHub, cliquer sur votre profil photo,

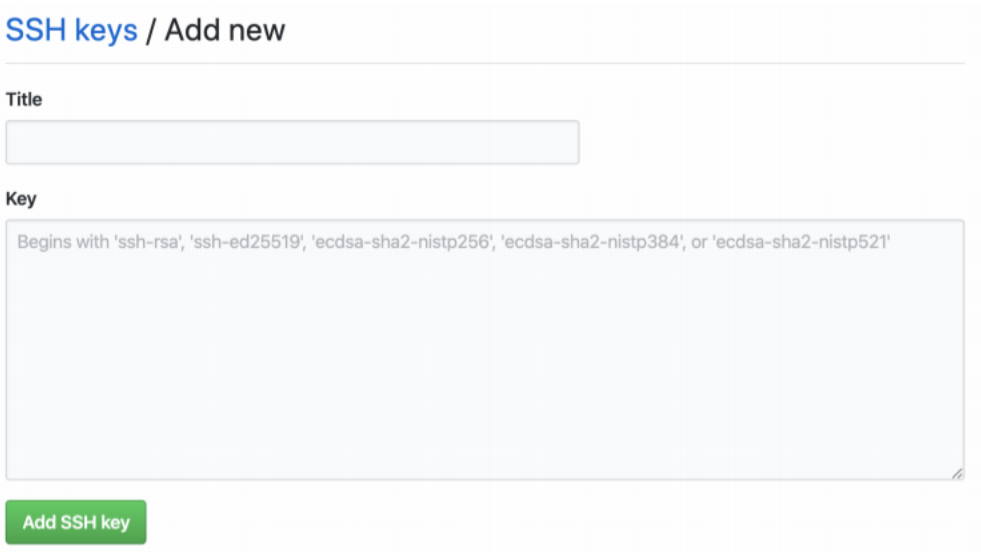
puis dans *Settings*.




3. Dans la barre latérale, cliquer sur *SSH et GPG keys*.




4. Dans le champ "Title", ajouter un label descriptive pour la nouvelle clé. Par exemple si vous utilisé un Mac, vous pouvez l'appeler MacBook.



5. Coller la clé copiée en 1) dans le champ "Key".

6. Confirmer le mot de passe GitHub (si demandé).

## 2 Voici une liste des commandes git qui pourront vous être utiles durant la réalisation de vos projets :

**git add [paths]** : permet d'ajouter des fichiers à la liste des fichiers "suivis"

**git commit -m [message]** : permet de sauvegarder les fichiers suivis localement (non accessible pour les autres). Le message doit décrire la sauvegarde (choisissez des noms pertinents).

**git push origin [branch]** : envoie les sauvegardes locales sur le repository distant (accessible aux autres).

**git branch [branche name]** : créer une nouvelle branche locale (elle sera disponible pour les autres au premier git push origin [nom de la branche]).

**git branch** : affiche les branches présentes en local.

**git checkout [branch]** : change votre branche de travail.

**git checkout -b [branch]** : créer une nouvelle branche et bascule immédiatement sur celle-ci.

**git fetch** : récupère les branches distantes qui n'existent pas en local (après cette commande, on peut basculer sur ces branches) ainsi que les modifications effectuées sans les fusionner.

**git status** : indique quels fichiers vont être sauvegardés localement (en vert) et ceux qui ne le seront pas (en rouge).

**git log −−abbrev-commit** : affiche la liste des derniers commits.

**git checkout [id du commit]** : retourne sur le commit donné en lecture.

**git reset −−hard [id du commit]** : retourne sur le commit donné. Comme si les commits

suivant avaient été effacés.

**git branch -D [nom de la branche]** : supprime localement la branche.

**git push origin −−delete [nom de la branche]** : supprime la branche distante.

**git merge [nom de la branche]** : fusionne 2 branches. Celle dans laquelle vous vous trouver est la branche qui sera modifiée.

### Astuces utiles

**git add .** : ajoute tous les fichiers modifiés du répertoire courant (ainsi que ceux les sous-répertoires) aux fichiers suivis.

**git add -A** : ajoute tous les fichiers modifiés depuis le dernier commit aux fichiers suivis.

**git reset HEAD∼[nb de commit] −−soft** : annule les "nb de commit" derniers commits sans perdre le code actuel.

**git reset HEAD∼[nb de commit] −−mixed** : annule les "nb de commit" derniers commits et annule le add sans perdre le code actuel.

**git reset −−hard** : revient au commit actuel.

**git reset HEAD∼[nb de commit] −−hard** : revient nb de commit commits en arrière.

Annuler des commits :
**git reset HEAD∼[nb de commit] −hard git push origin +[nom branche]** : le '+' n'est pas une erreur et est obligatoire.

### Exemple d'utilisation

```
git clone [clé ssh du projet sur Github]
cd [nom du dossier où se trouve le projet]
git checkout -b [nom de votre branche]
# Codez votre fonction !
git status
git add . / git add [liste fichier en rouge] / git add -A
git commit -m "Mon premier commit"
git push origin [nom de votre branche]
git pull origin master
git merge [nom de votre branche]
# si conflit: git merge --continue
git push origin --delete [nom de votre branche]
git branch -D [nom de votre branche]
```