-> github.com

Créer un compte github
Créer un repository sur ce compte.
repository: répertoire pour sauvegarder une application en remote

git clone https://github.com/monCompte/le_repository
clone un repo distant

git status
état

1ere fois
git add .
Crée une sauvegarde du répertoire en entier

git add -p
ajout des modifs avec vérification

git checkout -p
rejet des modifs (genre console.log()…etc…)

git commit -m "message explicite du commit"
création d'une version dans la branche

git branch
liste les branches locales

git branch *new_branch*
crée une branche *new_branch*

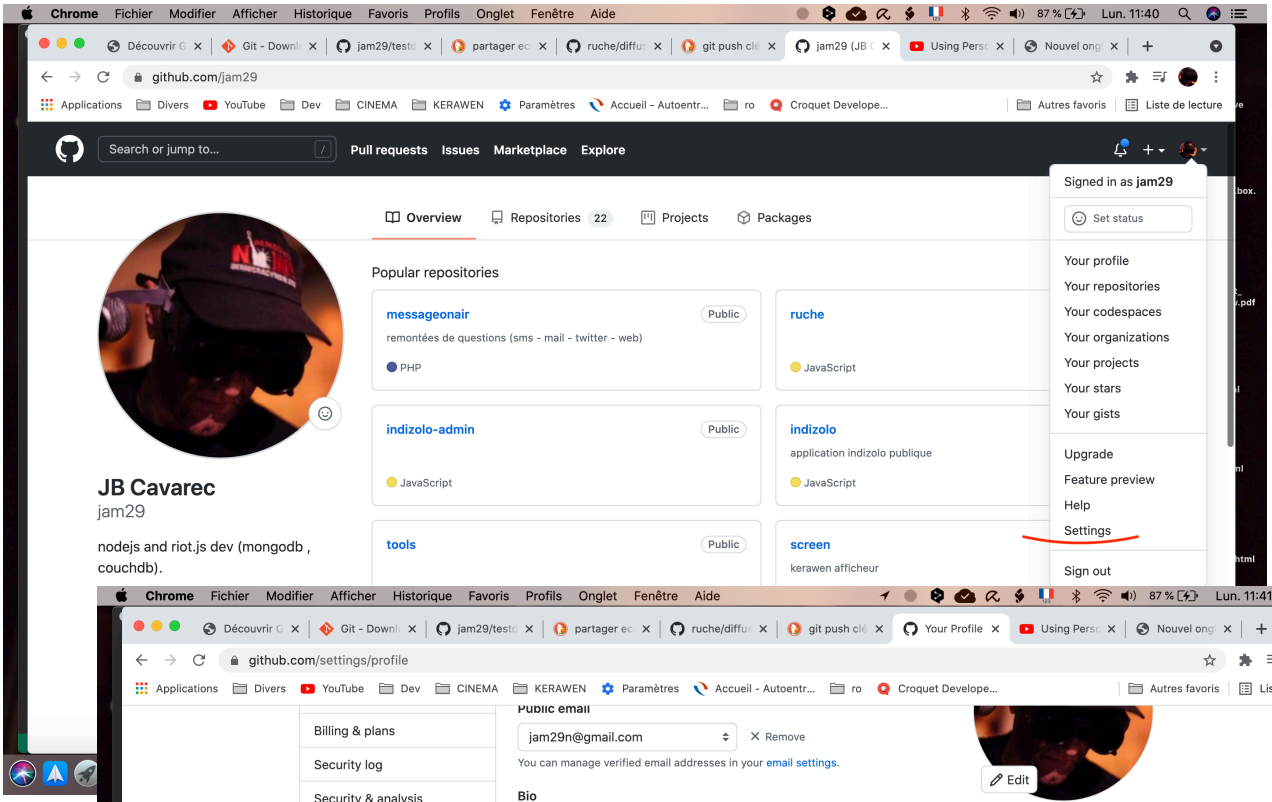
git checkout *branch_name*
se place dans la branche *branch_name*

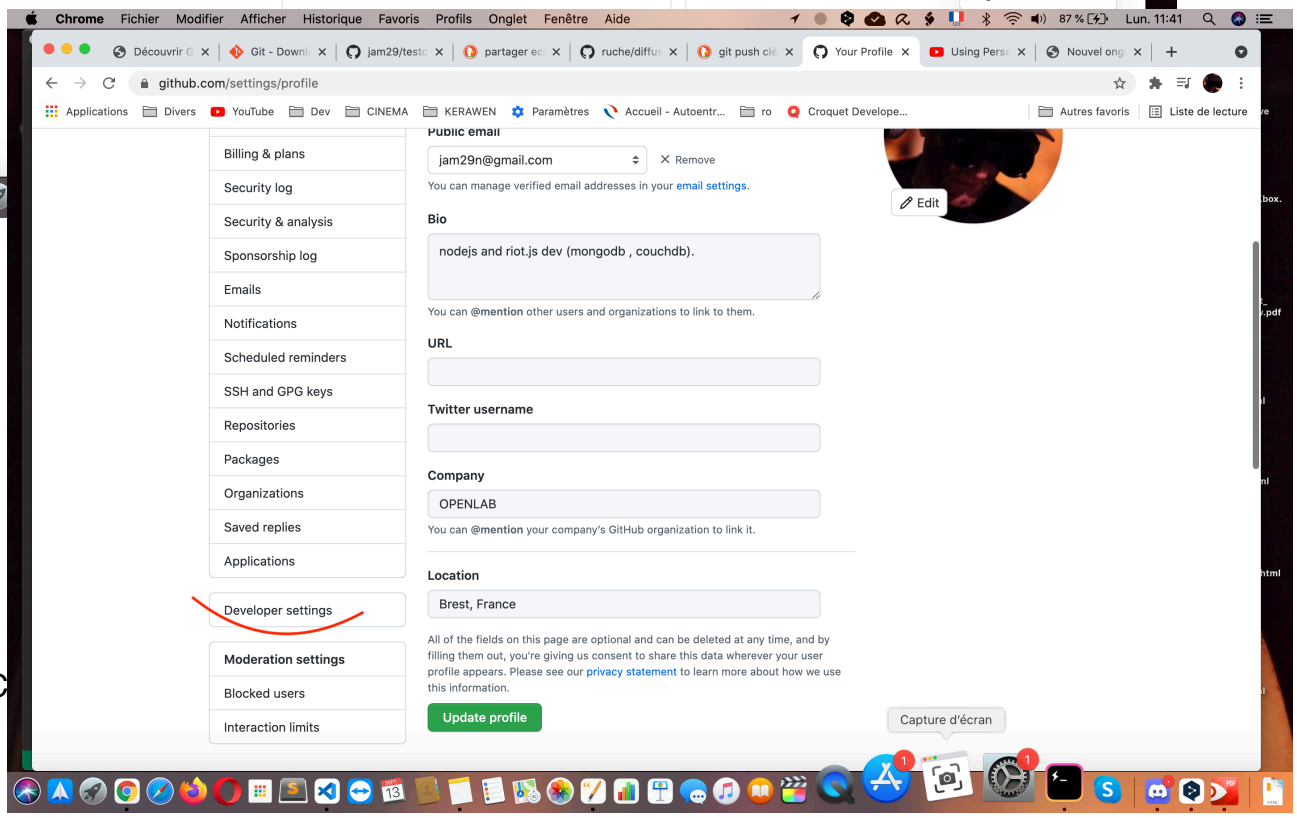
git branch -d *the_local_branch*

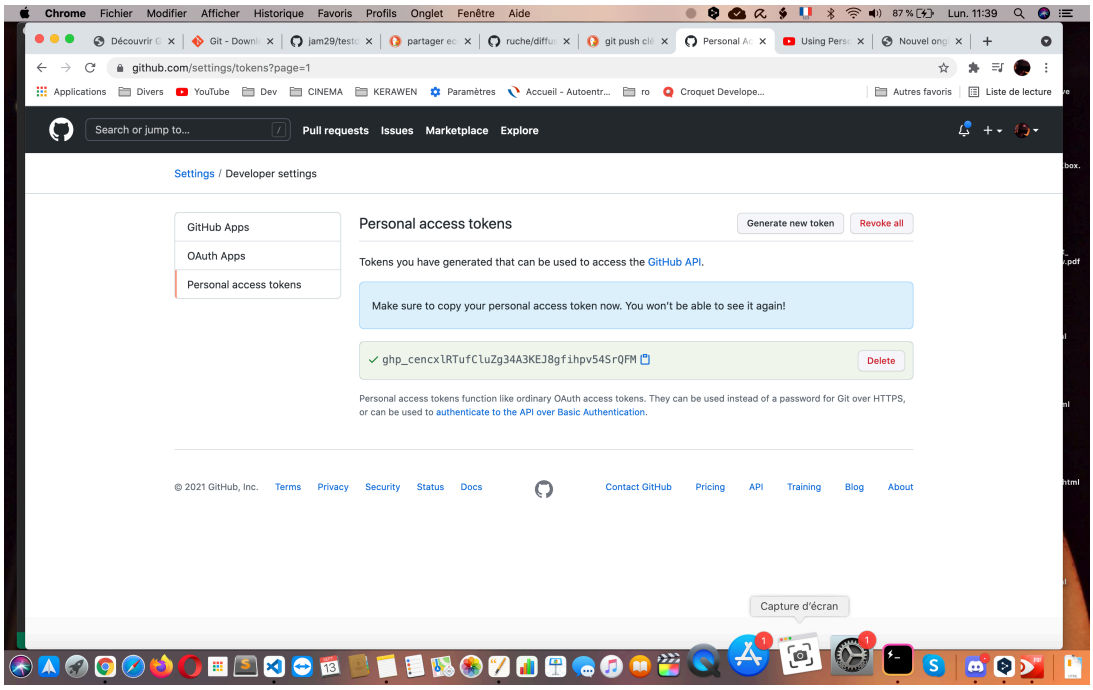
supprime une branche locale

Pour exporté le repository en local

1) icone user + settings
2) Developper Settings
3) Créer et sauvegarder le jeton (token)







git push

met à jour une branche distante à partir de la locale sur laquelle on est situé

demandera le nom du répertoire github ainsi que le token créé.

pour éviter de le saisir à chaque push

`git config --global credential.helper cache`
(à vérifier)

git pull

met à jour une branche locale à partir d'une branche distante

git stash

Sauvegarde provisoirement les modifications sans avoir besoin de faire un commit.

(*git stash* effectue un *git stash push* par défaut)

git stash pop

retour à l'état initial

Il y a possibilité de créer plusieurs git stash en les nommants.

exemple:
git stash save "WIP ajout d'un menu"
git stash save "WIP remplacement des callbacks imbriqués par des promises"

git log

Affiche tous les commits.

```
commit fcd84ab4a88a0cf4d0d2f2947bec31c92c79e19d
Author: jam openlab <jb.cavarec@orange.fr>
Date:   Wed Apr 18 16:23:08 2018 +0200

    stocks ok

commit ddd1af95bbf91bb94d8160e409d6406a31ca9467
Author: jam openlab <jb.cavarec@orange.fr>
```

Date: Mon Apr 16 15:24:37 2018 +0200

modif label"

commit 5e881ecaa3de458adce22a4f08e2134f4cfba85c
Author: jam openlab <jb.cavarec@orange.fr>
Date: Mon Apr 9 15:51:32 2018 +0200

saisie qtes stocks avancés avec declinaison en positif et negatif

commit ef45fe3d095b730b88c843f94f652f96089a0516
Author: jam openlab <jb.cavarec@orange.fr>
Date: Thu Apr 5 14:21:48 2018 +0200

entrepots en colonne

commit 0db782fae5feba9b7cde1a10f77de3fdb613eb5a
Author: jam openlab <jb.cavarec@orange.fr>
Date: Wed Apr 4 16:58:36 2018 +0200

saisie qte par entrepot

commit f8b15d43b1abbb8a9f5ff0595693ffc23fd13b9e
Author: jam openlab <jb.cavarec@orange.fr>
Date: Tue Apr 3 10:40:03 2018 +0200

depot par entrepot en mode select

commit 13cbbd27f78f6df5a95c5e76b4623ef2693336df
Author: jam openlab <jb.cavarec@orange.fr>
Date: Wed Mar 28 17:14:22 2018 +0200

affichage des entrepots pour 1 article avancé

git checkout *id_commit*

Se détache du commit courant pour aller sur le commit *id_commit*

Pour revenir à l'entête

git checkout *branche*

LE MERGE:

1) je suis sur une branche master
2) **git branch evolution1**
   je crée une branche evolution1

3) **git branch**
liste les branches avec une étoile devant la branche courante.

4) **git ~~checkout~~ switch *evolution1***
je me déplace sur la branche evolution1

5) je modifie un (ou des) fichier(s) puis
   git add -p (ou git add file)
   git commit -m "changement dans évolution1"

6) **git ~~checkout~~ switch *master***
   je reviens sur la branche master.
   On ne voit plus les changements sur ***evolution1***

7) je modifie un (ou des) fichier(s) puis

git add -p
git commit -m "changements dans master"

8) **git merge evolution2**

J'intègre les modifications de evolution2 dans master
2 possibilités:

1) pas de conflits => ok
2) conflits (2 modifs effectuées dans la même fonction par exemple)
Un marquage est effectué sur les fichiers en conflit.

Pour résoudre les conflits modifier le(s) fichiers en conflit puis accepter le(s) fichier de la branche master en tapant la commande:

git checkout --ours <fichier_en_conflit>

Sinon accepter la version de la branche à "merger":

git checkout --theirs <fichier_en_conflit>

9) **git diff**
Affiche les conflits

10) une fois les conflits résolus faire un commit (tant que les conflits ne sont pas résolus le commit est impossible)

11) **git branch -d evolution1**
Suppression de la branche

12) les conflits sont marqués par une insertion de marqueurs dans les fichiers:

<<<<<<<< Head: fichier.js
function bcrypt(île) { …. }

========
function bcryp(cle, sel) { …. }
>>>>>>>>> 345354g3545f435345

13) Annuler un merge:

git reset —hard HEAD

14) pusher une autre branche que master sur un site distant (remote)

git push --set-upstream origin <branch>

15) Lister les branches d'un site distant

git branch -a

16) descendre une branche distante (et s'y déplacer)

~~**git checkout <branche>**~~ (old version)
git switch <branche>

En local:

git init

crée un répertoire .git pour gérer nos versions en local

cloner plusieurs branches

git clone https://github.com/jam29/cours.git

cd cours

aller dans le rep Cours

git branch -a

```
* main
  remotes/origin/HEAD -> origin/main
  remotes/origin/especes
  remotes/origin/main
  remotes/origin/version2
```

git switch version2

git branch

git switch espèces

git branch