Formateur: Johnny Piette

Emai: johnny.piette@gmail.com

Tel: 0479/59.14.95

Bienvenue dans la formation Git/Github

Vous trouverez les différentes actions que nous avons faites lors de la formation.

Il est possible que ça ne soit pas exactement ce que nous avons vu car il est possible que le déroulement du cours ait changé les étapes suivantes.

De plus, voici un bon guide pour git/github: https://training.github.com/downloads/fr/github-git-cheat-sheet.pdf

# I. Installation de Git

Il faut au préalable avoir installé git sur votre machine.

a. Allez à l'adresse suivante : https://git-scm.com/downloads

b. Sélectionnez et installez en fonction de votre OS :

- MacOS

- Windows

- Linux/Unix

# II. Décompression de notre projet zippé

a. décompresser l'archive de notre projet

# III. Configurer les informations de l'utilisateur pour tous les dépôts locaux

Pour que l'on sache qui a fait une action sur un dépôt, il est de mise de configurer les variables globales suivantes :

a. git config --global user.name "[nom]"

- Exemple : git config --global user.name "Johnny Piette"

b. git config --global user.email "[adresse email]"

Exemple : git config --global user.email "johnny.piette@gmail.com"

# IV. Création de notre dépôt local

a. Aller dans le répertoire de notre projet. (commande cd)

b. git init (initialiser un dépôt depuis le dossier courant. Un dossier nommé .git a été créé ==> ne jamais l'effacer ou en modifier son contenu !!).

# V. Statut de notre dépôt

a. git status (donne le statut de notre dépôt).

git status nous dit qu'il y a des fichiers qui ne sont pas suivis. git status nous dit aussi que l'on peut utiliser la commande git add pour les indexer.

# VI. Indexer ses modifications

Nous pouvons indexer les fichiers non suivis par git avec la commande :

git add fichier1 [fichier2] [fichierX] (ici la notion de fichier prend aussi en compte les répertoires).

Ou

git add \* (copie tous les fichiers/répertoires récursivement sauf pour les fichiers et répertoires commençant par un .)

Ou

git add . (Copie tous les fichiers/répertoires récursivement même les fichiers et répertoires commençant par un .) ==> bien entendu, sauf le répertoire .git

a. git add . sera la méthode retenue dans notre cas.

b. git status (git nous dit qu'il y a 4 nouveaux fichiers qui sont indexés et qu'ils doivent maintenant faire partie de notre dépôt).

# VII. Initial Commit

Souvent quand c'est un projet existant qui n'a pas encore eu de dépôt, on fait ce qu'on appelle un Initial commit lorsqu'on a fait au préalable une indexation de celui-ci via un git add

a. git commit -m "Commit initial"

b. Et voilà, vous avez fait votre premier commit. :)

c. git status (pour voir le résultat)

# VIII. Première modification

a. Ouvrir le fichier index.html via un éditeur: VSCode

b. A la ligne 17, corriger "phormation" en "formation"

c. Enregistrer le fichier.

d. git status (git nous dit qu'il y a eu une modification dans le fichier index.html)

e. git add index.html

f. git commit -m "correction erreur Français"

# IX. Visualisation d'une modification non indexée

a. Ouvrir le fichier site.js dans le répertoire js

b. Ajouter en dessous de la ligne n°4 console.log("Vous avez cliqué sur l'image.");

c. Enregistrer le fichier

d. git status (on voit que le fichier site.js a été modifié)

e. 1: git diff js/site.js (git diff permet de voir les modifications/différences. Un + en début de ligne montre un ajout de code. Un - en début de ligne une suppression)

2: VSCode dans l'icone Git sur le côté gauche.

f. git add js/site.js

g. si on veut encore voir les modifications même si js/site.js est dans la zone d'index: git diff --cached js/site.js (et on retrouve le même résultat qu'avant l'indexation)

h. git commit -m "On affiche dans la console qu'on a cliqué sur l'image"

# X. QUIZ:

1. Pour créer un nouveau dépôt git, j’utilise :

[ ]a. git log

[ ]b. git init

[ ]c. git status

2. Pour afficher des informations sur mon dépôt, j'utilise la commande :

[ ]a. git init

[ ]b. git log

[ ]c. git info

[ ]d. git status

3. Pour indexer mon fichier index.html, j'utilise la commande:

[ ]a. git add index.html

[ ]b. git status index.html

[ ]c. git reset index.html

4. Dans le cas suivant, quel fichier va être enregistré dans l'historique de mon dépôt après la commande git commit -m"quiz" ?

$ git status

On branch master

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: index.html

no changes added to commit (use "git add" and/or "git commit -a")

[ ]a. index.html

[ ]b. Tous les fichiers du projet

[ ]c. Aucun fichier du projet

5. Pour voir les modifications apportées sur le fichier hello.html, on utilise la commande:

[ ]a. git diff index.html

[ ]b. git version index.html

[ ]c. git commit index.html

[ ]d. git add index.html

Réponses du QUIZ

1. b

2. d

3. a

4. c

5. a

# XI. Exercice récapitulatif

1. Modifier le fichier css/site.css.

2. définissez la couleur red à la classe pcolored.

3. Sauvegardez.

4. Affichez le statut de votre projet.

4. Regardez les modifications du fichier css/site.css

4. Ajoutez dans la zone d'index le fichier css/site.css.

5. Commitez les changements.

# XII. Voir l'historique d'un dépôt

Ici pour les exemples, SHA-1 doit être remplacé par ce que vous voyez à l'écran et doit contenir 40 caractères. Ex : fda909edc581dec5c4409980261bebc7a8cee51e

a. git log (Donne la liste de tous les commits effectués avec divers informations : SHA-1 (identifiant unique de 40 caractères), l'auteur, la date du commit et le message du commit)

b. git show SHA-1 (Pour un commit ayant ce SHA-1)

c. git show master (montre les dernières modifications du commit avec le tag master)

# XIII. Naviguer dans l'historique d'un dépôt

a. git log

b. git checkout SHA-1 (nous nous sommes déplacés sur un commit, nous avons remonté le temps)

c. git log (on voit que l'on a les commits précédents le commit en cours.)

d. Ouvrez index.html et regardez cette page qui est différente de celle de master.

e. git checkout master (on revient sur le dernier commit appelé master).

# XIV. Utilisez des tags

Les SHA-1 sont longs à recopier (vive le copier/coller) et à mémoriser. C'est pourquoi, il est possible de leurs donner un nom facile à retenir à taper.

Création

a. git checkout SHA-1 (nous sommes sur le commit désiré)

b. git tag montag (on donne un tag à ce SHA-1)3

c. git checkout master (on revient sur le commit master)

d. git checkout montag (on se déplace sur le nouveau tag que nous avons créé pour un commit)

Suppression

a. git checkout montag

b. git delete montag (le tag a été supprimé et on voit maintenant le numéro du SHA-1 au lieu du tag)