Présentation Git/Github

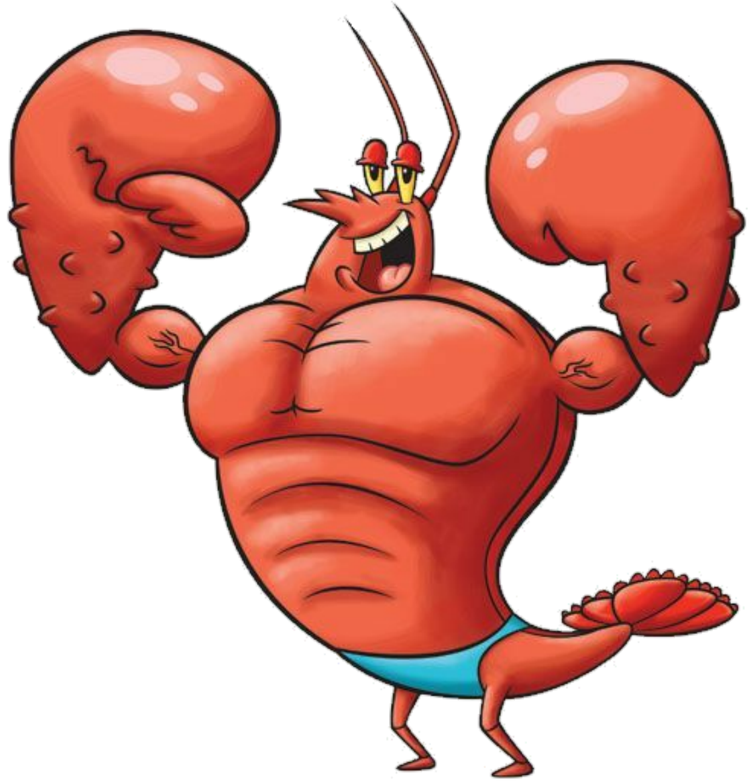
Nuit de l’info 2023

Equipe Bab les Ponges

On oublie pas:
L'équipe de BAB ne jetera pas Les Ponges !

7/12/2023 - 8/12/2023

Présenté Par



MOI

Plan

I -/ Brève présentation de Git et Github

II -/ Configurer sa machine et init d’un dépot

III -/ Les commandes de Bases qui vont servir

IV -/ Les Branches

V -/ Pull request

Désolé mais certaines diapo seront fat

I -/ Brève présentation de Git et Github

. Git est un logiciel qui permet notamment de versionner son code

. Créer par Linus Torvalds, créateur du noyau linux

. Versionner permet de garder des traces de son code/documentation etc..

. Grace à Git en cas de problèmes on peut restaurer d’anciennes versions de son code

. On peut savoir qui a fait quoi, quand, comment

. Outil de travail en collaboratif !!!

. Github -> service d’hébergement en ligne (compte pro avec adresse mailde l’iut)

II -/ Configurer sa machine et init d’un dépot

dans un git bash:

- on commence par s’authentifier avec les commandes

  git config --global user.name “MaxMontouro”

  git config --global user.email “max.montouro@gmail.com”

  git config --global core.editor “code --wait” commit avec VSCODE
- pour init un dépôt

  git init

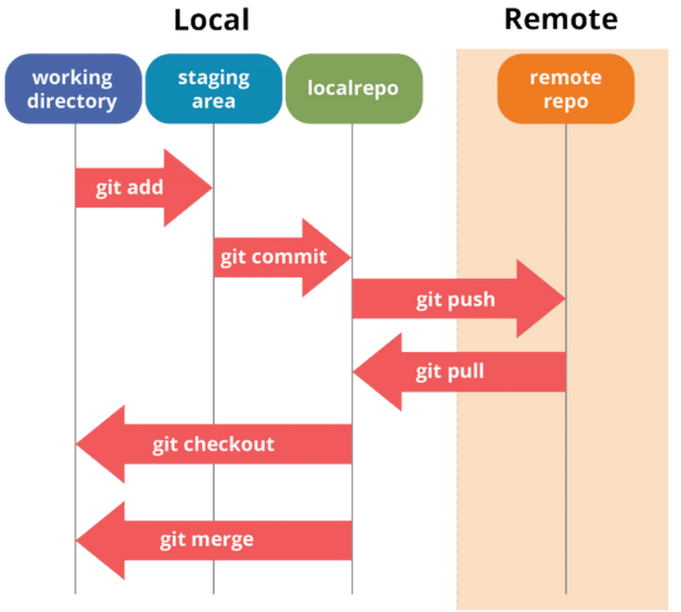
Ici on ne va pas se concentrer sur l’initialisation d’un dépôt puisque nous travaillerons ensemble sur mon dépôt (ou vous êtes censé être collaborateur)

Il vous suffira de cloner sur un dépôt local mon dépôt avec la commande suivante:

git clone https://github.com/MaxMontouro/Nuit_Info_2023.git

Vous sera demander une authentification donc se connecter avec ses codes (flemme du token)

III -/ Les commandes de Bases qui vont servir



Les commandes qu'on va utilisé:

- git add . (pour ajouter des fichiers nouveaux ou modifiés)
- git commit (-m " ") (pour ajouter un message au commit)
- git push (permet de push sur le dépôt github après une première connexion établie avec le dépôt
- git pull (permet de mettre son dépôt local à jour par rapport à celui sur github)

utilisé git add . car prend tout les fichiers du repertoire courant alors que git add * prend tout les fichiers en global

On peut aller bcp plus loi mais ici on reste sur quelque chose de très simple.
Pour toutes infos et/ou problèmes je suis la

IV -/ Les Branches

Pour éviter tout problèmes, nous ne travaillerons jamais sur la branche master(main) nous utiliserons les branches (git branch nomBranche pour en créer une). La branche est une copie du projet à un instant T (sa création). Son espérance de vie doit être courte pour éviter de créer des conflits. 1 fonctionnalité = 1 branche

Les commandes importantes pour les branches :

- git branch nomBranche
- git switch nomBranche (pour naviguer entre les branches)
- git switch -d nomBranche pour supprimer une branche (depuis master seulement fait par les chefs de groupe ou moi si fusionner et -D sinon)
- git push origin nomBranche (pour push la branche)

Pour la syntaxe nous utiliserons le formalisme suivant : GROUPE-Fonctionnalite le groupe (FE, BE ou encore DS) cela me permettra de voir l'état des branches et qui fait quoi dans ces branches.

Attention les branches doivent être réaliser dans les différents dossiers et non à la racine

V -/ Pull request

Eviter la demande de pull request pour correction de bug

Pour push sur master/main il faudra donc effectuer une pull request (demande de validation auprès du chef de groupe/projet/responsable github)

IMPORTANT avoir un dépôt à jour donc utilisation du git pull pour merge dossier local/github
Pour éviter tout conflits chacun des dev travaillera si possible sur des fonctionnalités pour faire simple si pendant la nuit il y a un conflit appelez moi

Lorsqu'on va push une branche une fois qu'on estime fini (sans bug/ test validé etc..) on va créer en une pull request. Pour cela faire la commande:
git push origin nomBranche (origin est le type de connexion mais on détaille pas)
par le suite celui qui a push la branche doit aller sur github et lancé la rubrique compare & pull request
Le but va être de demander au chef de section de validé le code en le vérifiant (éviter les conflits etc..) en ajoutant par exemple des messages, des tags le but est d'être le plus claire possible pour éviter de perdre trop de temps à vérifier

V -/ Pull request 2

Le chef de groupe va donc pouvoir voir la pull request (sur son compte github et sur le dépôt). Pour chaque Pull Request, on peut voir son titre, son auteur, son étiquette (si elle en a une) et à qui elle est assignée (c'est pour ça qu'il faut être claire)

Le chef pourra voir le code changé et/ou créer ainsi que validé ou pas la pull request. Il doit s'engager à commenter le code si il y a des pb ou quoi. Le bouton Start a review indique que le commentaire demande un travail de ré-écriture du code (sinon on utilisera le bouton Add single comment). Selon le travail effectuer le chef validera ou pas la pull request avec les boutons Approuve ou Request changes et si validé merge la branche

SI pas validé le dev rapatriera son code avec git pull, se mettra dans sa branche et modifiera son code (si le code a bien été documenté sur github le dev aura directement la doc sur son vscode)

Petite précision, je checkerai toute la nuit l’état
du dépôt avec une application web graphique
en node.js

ungit

Le seul dossier auquel la branche peut ne pas être mis en place est celle pour la documentation
Les défis par contre si !!!

hesitez pas a merge main sur la branche avant de merge la branche sur main pour éviter de niquer main