Formation GIT

Moussa Camara

Git

- GIT est un outil de Versionning, c'est ce qu'on appel un VCS (Version Control System).Il permet d'avoir plusieurs versions des fichiers qui composent une application (site web ou autre), et ainsi travailler en équipe tout en gardant des historiques (versions) des fichiers sur lesquels on travaille.

- GIT doit être installé et utilisé sur le pc serveur et le pc client. L'échange de données GIT peut se faire de serveur à serveur, de client à serveur, de serveur à client, de client à client.Il existe plusieurs concurrents de GIT (SVN etc..), mais GIT est le plus utilisé.

Github

- GITHUB est ce qu'on appel un hebergeur de repositories GIT. C'est à dire, qu'il se charge d'heberger les versions de notre application et ainsi on peut y accéder en ligne. C'est un peu comme Dropbox. Il existe plusieurs hebergeurs (bitbucket etc ..), mais github est le plus connu et le plus utilisé.

- Nous avons aussi la possibilité de ne pas passer par GITHUB et d'heberger son repository GIT dans un serveur perso. Un VPS ou un serveur dédié sera necessaire pour pouvoir utiliser les lignes de commandes.

- L'échange entre le repository en ligne et le repository local se fait en SSH ou HTTPS.GITHUB est une interface web qui permet de commander GIT en ligne sans passer par des lignes de commande.Sur son serveur perso, nous pouvons installer GITLAB qui fait la même chose que GITHUB.

GIT – Les bases

- gérer l'évolution d'un code
- partager efficacement son travail en équipe
- garder un historique de l'évolution d'un projet
- travailler en parallèle

GIT – Les bases

Autre systèmes existants

- Subversion (svn)
- Mercurial (Hg)
- Bazaar (bzr)
- Autres…

GIT – Son histoire

- créer pour le noyaux Linux
- créé par Linux Torvald
- maintenue par Junio Hamano
- adoption rapide et massive depuis 2005
- VCS numéros 1 dans le développement web

GIT – Avantage ?

- suivi précis de l'avancement d'un projet
- souplesse dans l'évolution
- facilité de mise en commun
- backup

GIT – Installation

Windows:

- git for windows

Linux:

- apt-get install git

Mac os:

- brew install git

GIT – 1er dépôt

Configuration global de git:

- git config --global user.name “Votre nom”
- git config --global user.email “root@email.fr”
- git config --global color.ui auto

GIT : envoyer mon travail sur mon propre depot

- #Comment envoyer un dossier ou projet sur GitHub .com depuis mon ordinateur
  - 1- Initialiser mon dossier c'est-à-dire ajour une référence .git
    - a. Git init (A faire une seule fois)
  - 2- Se connecter et créer un dépôt/ Repositories sur github.com
    - a. https://github.com/new
  - 3- Ajout de mes dossiers et/ou fichiers depuis mon ordinateur
    - a. Git add -A (L'option -A (All) : ajout de tous les dossiers/fichiers)
    - b. Git add exempledefichier.html (Envoie que le fichier : exempledeficher.html)
  - 4- Préparer un message
    - a. Git commit -m 'message correspond aux modifications effectuées'
  - 5- git remote add origin https://github.com/votrepseudo/le-nom-de-votreprojet.
    - a. Ce lien explique à quel endroit je veux envoyer mes sources qui sont sur ma machine en local (chez moi)
  - 6- Envoyer sur github.com
    - a. Git push origin master (ou le nom de ma branche)

GIT: Exemple

- - créer un compte sur https://github.com

- $ cd /C:/Desktop/workspace/mon_projet // je me positionne sur mon projet (à partir de mon PC local)
- $ git init // initialisation du repo dans le projet
- $ git config --global user.name 'Prenom NOM' // Configuration de mon git PS:à faire une seule fois
- $ git config --global user.email 'adresse@mail.com'-

- Aller sur github.com, créer un nouveau repository et executer ces commandes sur votre terminal
- $ git remote add origin https://github.com/utilisateur/votre_projet.git
- $ git add –all
- $ git commit -m 'Première version de mon projet'
- $ git push origin master ¤ va demander le login et mdp

GIT – Cloner un dépôt existant

Récupérer l'adresse du dépôt de bootstrap

git clone [*adresse*]

Ta Da !

GIT – Cloner un dépôt existant

- Pour cloner un projet c'est à dire avoir une copie d'un projet sur votre machine en local, on passe par git clone …
- PS: Pour cloner un projet, il faut ce dernier en public ou le propriétaire vous a invite sur son depot.

GIT – Cloner un dépôt existant

- Pour inviter un collaborateur sur votre projet,
  - Cliquer sur votre projet github.com > settings > Manges Access : ajouter les collaborateurs au projet

- - Les collaborateurs effectuent la commande suivante à partir de leur pc local à l'endroit où va être créé le dossier du projet

- $ cd /C:/Desktop/workspace
  - //positionnement dans son serveur

- $ git clone https://github.com/utilisateur/projet.git
  - //pour cloner directement le projet, **sans faire de git init** etc...

GIT – Cloner un dépôt existant

- - Le collaborateur effectue ses modifications après s'être positionné sur sa branche de travail
- $ git checkout -b nom_du_travail
  - il créé sa branche où il va travailler et se positionne dessus en une seule commande
- $ git add –A
- $ git commit -m 'Modification de la fonctionnalité'
  - le collaborateur fait ses commits réguliers au cours de son travail

- $ git push origin nom_du_travail
  - il pousse sa branche dans le repo en ligne

GIT – Cloner un dépôt existant

- - Il se rend sur github.com et clique sur "compare and pull request" qui est apparu en vert sur la page du projet
- - Il laisse un commentaire et clique sur "create pull request''

- Un autre collaborateur regarde son travail et clique sur "merge pull request" puis "confirm merge''

- - Il supprime la branche nom_du_travail sur github.com, rubrique "branches"

Quelques commandes

- $ git checkout master
  - Pour changer de branche
- $ git pull origin master
  - il recupère la derniére version
- $ git branch -D nom_du_travail
  - suppression de la branche en local
- $ git branch nouveau_travail
  - creation d'une branche pour nouveau travail

Quelques commandes

- - Tous les autres collaborateurs récupèrent le travail effectué par le collaborateur précedent
- $ git pull origin master
  - recuperation des fichiers modifiés

- En cas de modification:
  - création d'une branche, on se positionne sur la branche, on fait les modifs, on add, on commit, on push, on pull request, on merge, tout le monde pull .....

Quelques commandes

- $ git config --global user.email ‘mon.email@gmail.com’
  - pour s'identifier avec son email
- $ git config --global user.name ‘Moussa Camara’
  - pour s'identifier avec son nom (complémentaire de l'email)
- $ git init
  - permet de créer un repository dans le working directory
- $ git clone dossier_source_qui_contient_le_repo_a_cloner dossier_cible
  - permet de cloner un repository avec tous les fichiers sources du working directory
- $ git status
  - permet de voir quelles modifications ont été effectuées sur les fichiers de l'application

Quelques commandes

- $ git diff: permet de voir quels sont les modifications qui ont été effectuée
  - - après avoir regardé la différence , pour quitter l'editeur il faut taper ":q"
- $ git branch ¤ avoir la liste des branches
- $ git branch nom_de_la_branche
  - nous créons une branche parallèle, pour ne pas altérer le travail fait sur la branche master
- $ git checkout nom_de_la_branche
- $ git checkout -b nouvelle_branche
  - je crée et me positionne sur nouvelle_branche (racourci de branch nom_branche + checkout nom_branche)
- $ git checkout master
  - en revenant sur la branche master, tout le travail fait sur la branche "nom_de_la_branche" est annulé
- $ git checkout nom_de_la_branche
  - en revenant sur la branche nom_de_la_branche, tout le travail fait sur la branche "nom_de_la_branche" est rétabli

Quelques commandes

- $ git fetch . nom_de_la_branche:master
  - permet de faire en sorte que master devienne une copie de nom_de_la_branche Ainsi en faisant un checkout sur master on retrouve le travail fait sur nom_de_la_branche et on peut supprimer nom_de_la_branche
- $ git branch -d nom_de_la_branche
  - permet de supprimer une branche, -D permet de forcer la suppression d'une branche qui contient des fichiers untracked
- $ git merge nom_de_la_branche
  - en se positionnant sur master , on merge (fusionne) le travail de nom_de_la_branche, cela produit un commit automatique sur master avec les modifs de nom_de_la_branche
- $ git remote add origin https://github.com/moussa/moussa.git
  - lier un repository local à un repository en ligne (origin est le nom donné au repository en ligne, si nous en avons plusieurs, nous pouvons les nommer differemment (git remote add moussa etc...)).

Quelques commandes

- $ git remote –v
  - permet de voir si on a bien bindé notre remote repository
- $ git push -u origin master
  - cela va pousser la branche master local sur la branche master en ligne et mettre à jour les fichiers qui sont en lignes (demande du login et mot de passe de github.com)
- $ git fetch origin master
  - permet de rappatrier le remote repository SANS prendre les fichiers
- $ git pull origin master
  - permet de rappatrier le remote repository AVEC les fichiers (tous les fichiers la 1ere fois, ensuite ça prendra que les modifs des fichiers)
- $ git config user.name
  - permet de show l'username
- $ git config user.email ¤ permet de show l'user email

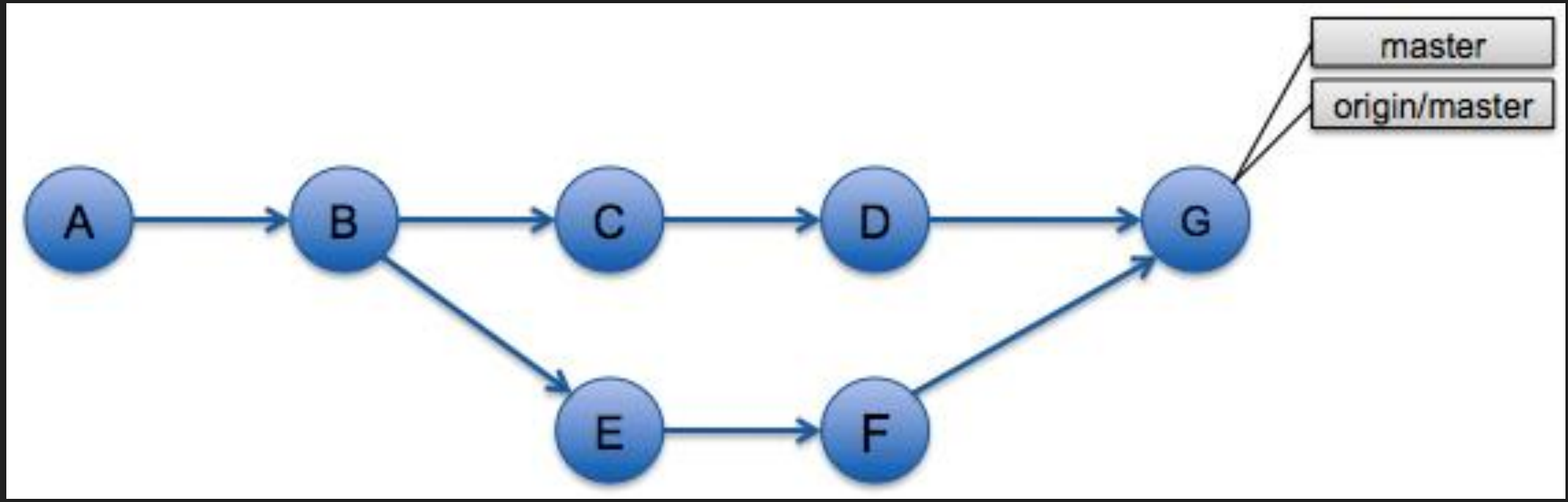
Quelques commandes

- $ git commit -a -m 'nouveau commit'
  - permet de faire un add + un commit en une seule commande A CONDITION qu'il n'y ai pas de nouveau fichier créé, mais que des fichiers modifiés
- $ git commit --amend -m 'modifier dernier message de commit'
  - permet de modifier le message du dernier commit
- $ git reset HEAD^
  - permet de supprimer le dernier commit et revenir à l'état avant celui-ci (sans restaurer le working directory Pour restaurer le working directory voir reset --hard)
- $ git reset HEAD^^
  - permet de supprimer le dernier et avant dernier commit et revenir à l'état avant celui-ci (sans restaurer le working directory Pour restaurer le working directory voir reset --hard)
- $ git reset HEAD^^^
  - permet de supprimer le dernier et avant dernier et avant avant dernier commit et revenir à l'état avant celui-ci. etc.. (un ^ = un commit)

Pour la gestion de conflit

- Celle ci est constatée sur github.com quand l'utilisateur va tenter de faire un merge. Alors il faut :
- $ git pull origin master ¤ à partir de sa branche locale sur laquelle il travaillait : à ce moment là le conflit se retrouve en local.
- $ git diff --name-only --diff-filter=U
  - permet de dire quels sont les fichiers en conflit : là il solutionne son conflit.
- $ git add et git commit ¤ comme d'habitude
- $ git push origin sa_branche
  - il remet sa branche en ligne : il fait son merge en ligne normalement
- $ git checkout master ¤ se positionne sur master
- $ git pull origin master
  - il pull sur Master pour reprendre son travail
- $ git branch -D sa_branche
  - il supprime sa branche

GIT – Les branch



GIT – Les branch

- git checkout -b new_branch

ou

- git branch new_branch

git branch : listes toutes les branches en local

GIT – Les tag

- git tag : Listes les tags sur la machine.
- git tag -a v1 -m “first tag” : Création du tag “v1”
- git push origin v1 : Pousse le tag sur le dépôt distant.
- git show v1: Affiche les informations du tag

Recap

- On créé son projet en local,
- on initialise un repository GIT en local,
- on ajoute les fichiers de son son projet (git add --all),
- on créé un repository GIT en ligne (sur github.com),
- on lie le repository local au repository en ligne (git remote add origin https://github.com/utilisateur/projet.git),
- on push son projet local vers le repository en ligne (git push origin master).

Recap

- Ensuite, en local, on créé une branche (git branch nom_de_la_branche),
- on se positionne sur celle-ci (git checkout nom_de_la_branche),
- on fait ses modifs de fichier (travail sur sublime text..),
- on ajoute ses modifications dans le repo local et on commit (git add git commit).

- On push sa branche sur le repo en ligne (git push origin nom_de_sa_branche).

- On se rend sur github.com et on demande une pull request (compare and pull request), pour faire cette demande on clique sur pull request et on laisse un commentaire, eventuellement en donnant le nom de la personne dont on desire la vérification de notre code, puis on clique sur "create pull request".

Recap

- Ensuite, une autre personne (celle citée précedemment, ou pas...) vérifie notre code et clic sur "merge and confirm pull request".

- Enfin, on supprimme la branche en ligne (nom_de_la_branche), on supprime la branche en local (git branch -D nom_de_la_branche).

Recap

- Et les autres membres de l'équipe récupèrent notre travail : ils doivent se positionner sur leur branche master (git checkout master) et faire un git pull origin master.

- Y'a plus qu'à recommencer le process à partir de la création d'une nouvelle branche, pour un nouveau travail...

- NB: Les push se font généralement en fin de journée,
- et les pull / merge se font le matin en arrivant.

- Il ne faut jamais push master directement sauf si on sait ce qu'on fait.

Recap

- En cas de conflit lors d'un merge (travail sur le même fichier par 2 personnes différentes en même temps) :
- On revient en local, (à partir de la branche avec laquelle on a tenté le merge), on fai un git pull origin master, on corrige le conflit.
- On git push la_branche, et maintenan le merge devrait passer.

- Puis toute l'équipe peut faire un git pull origin master, et reprendre le processus normal.

- INFOS : Lorsqu'une nouvelle personne arrive dans le porjet et qu'elle n'a pas de repository initial, elle doit faire un git clone https://domain.com/le_projet.git

- Cette commande va créer un dossier avec tous les fichiers du projet.

Et enfin …

- Pour finir:
  - NE PAS CONFONDRE LES COMMANDES GIT ET LINUX

C’est juste une question de branche

Les arbres dans la vraie vie

Les arbres en informatique