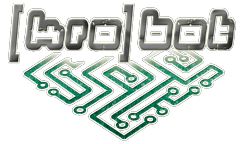
Git la vie

Git la vie

Courte initiation à git et gitlab

Émile Siboulet

2019



Git la vie

Introduction à git et au gitlab

Qu'es que git ?

Git est une application qui permet de versionner des fichiers et de les partager, pour cela il enregistre les différences.

Gitlab est une interface web qui permet d'afficher graphiquement l'état du dépôt.

Introduction à git et au gitlab

Les branches

Intérêt de ranger des différences et non des fichiers :

- alléger la taille du stockage
- avoir des modifications en parallèles possibles

On s'organise donc en branches qui partent toutes d'un même point et qui ajoutent leurs propres modifications sur le fichier original.

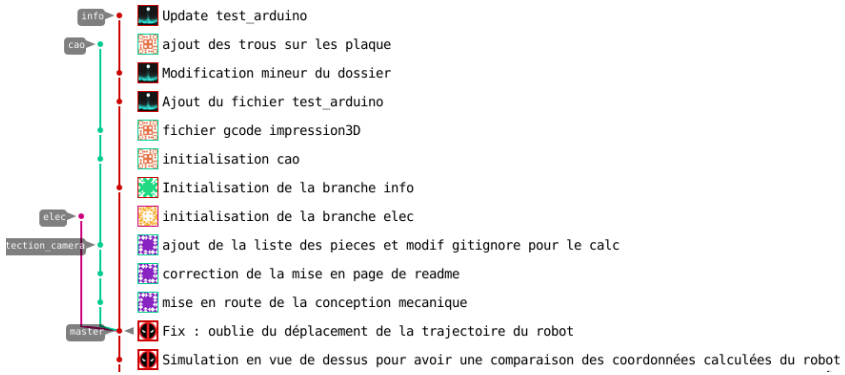


FIGURE – Représentation graphique des branches par gitlab

Les opérations sur les versions

La gestions des différences par git

Git est précis à la ligne prêt pour le stockage des différences. On peut donc éditer un même fichier sur deux branches différentes si on n'édite pas les mêmes lignes.
On peut voir les différences sur le gitlab :

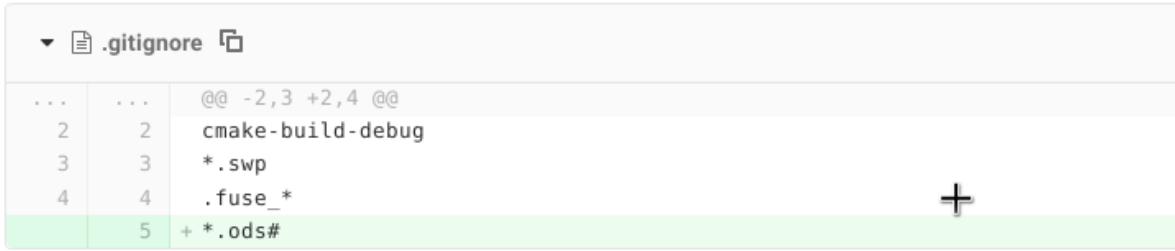


FIGURE – Différence entre deux branches

Sur la figure 2, il y a une ligne ajoutée dans le fichier .gitignore

Les opérations sur les versions

Ajouter ses modifications au projet final

Quand la nouvelle version convient, on peut la mettre dans la branche principale du projet ⇒ merge request. Ceci s'effectue sur le gitlab

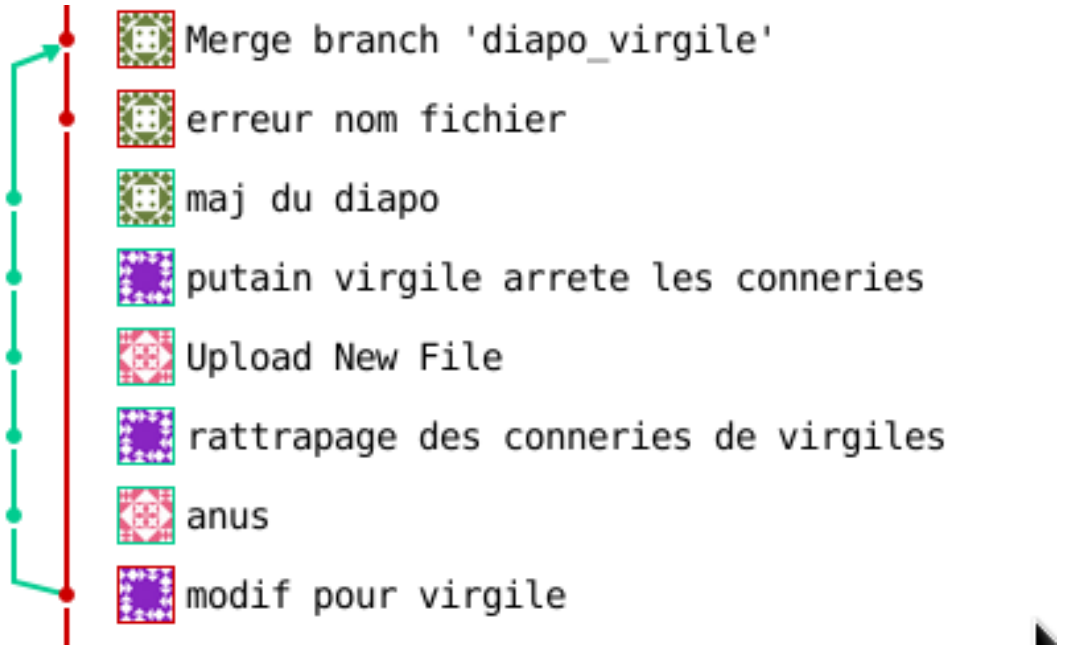
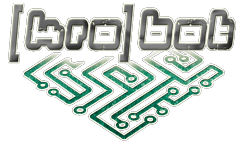


FIGURE – Exemple de branche mergée

Communication avec le serveur

La copie du dépôt sur le client

Git ne permet pas de partager des fichier en direct. Le serveur possède l'intégralité des branches et des fichier.
Lors de la copie sur la machine le serveur ne donne que la branche *master*. Pour accéder à d'autre branches il faut changer de branche et la demander au serveur.



Communication avec le serveur

La copie du dépôt sur le client

De même lors de la modification d'un fichier, il faut faire attention de partir de la version du serveur et de lui envoyer après. Cette méthode est peu pratique mais elle permet de travailler sans connexion direct au serveur et sans perte de vitesse (dossier partagé en réseau)

Communication avec le serveur

Organisation des messages au serveur

Chaque ligne modifiée dans un fichier et une modification indépendante. Pour plus de clarté, on range une liste de modifications dans un commit, ce qui permet de ranger l'avancement d'une branche.

Correction de la page de garde

- ajout : photo.png
- modification : presentation.tex
- modification : presentation.pdf
- suppression : signature.jpg

FIGURE – Exemple de commit

Communication avec le serveur

Représentation du serveur

On peut représenter le bout des branches (place de travail) par des flèches qui pointent sur des commits.

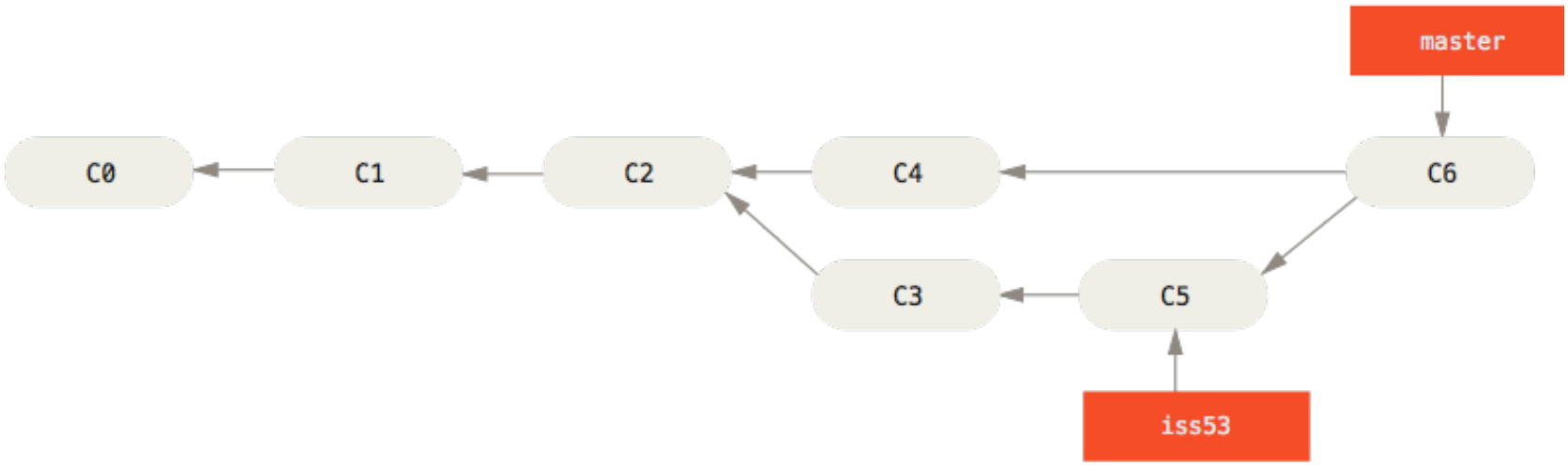


FIGURE – Exemple de branche mergée

Et dans la vrai vie ça donne quoi ?

Commande propre à Linux

Bash :
utilisateur@machine :localisation $ commande

Organisation des fichiers :

- . dossier actuelle
- .. dossier parent
- ~ dossier home (parent de document, bureau, image, musique..)

Et dans la vrai vie ça donne quoi ?

Commande propre à Linux

Toutes les commandes bash fonctionnent avec la même syntaxe :

$$\text{[commande] [option1] [option2] ...} \tag{1}$$

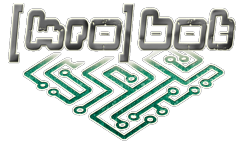
Généralement les raccourcis des option sont avec un seul tiret devant, les deux commandes suivantes sont équivalentes :

$$\text{ls -a} \iff \text{ls –all} \tag{2}$$

En cas de doute :

$$\text{man [commande] ou [commande] -h} \tag{3}$$

Ça fait peur mais ça explique tout !



Et dans la vrai vie ça donne quoi ?

Commande propre à Linux

Listes des commandes pour naviguer sur son ordinateur :

`ls` affiche les dossiers / fichiers

`cd` change la position dans les dossiers

`mkdir nom_du_dossier` crée un dossier avec le nom donné

`touch nom_du_fichier` crée un fichier avec le nom donné

`nano nom_du_fichier` édite un fichier text

Et dans la vrai vie ça donne quoi ?

Commandes propres à git

Liste des commandes utiles pour git :

git clone {url}	clone le dépôt qui avec l'url fourni
git status	affiche l'état du dépôt (à utiliser sans ménagements)
git pull	demande au serveur les nouvelles modifications
git add {fichier/dossier}	valide l'avancement du fichier pour le prochain commit
git commit -m "{message}"	range dans un commit les modifications validées
git push	envoie les modifications au serveur
git branch	donne la liste des branches locales
git branch {nom_branche}	crée la branche locale nom_branche
git checkout {branche/fichier/dossier}	revient à l'état connu par le dernier commit

Et dans la vrai vie ça donne quoi ?

L'utilisation du gitlab

L'url pour cloner le liens se trouve sur la page d'accueil, le bouton bleu en haut a droite. HTTPS est plus simple.

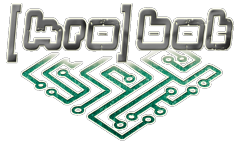
Utilisez des README.md avec la syntaxe appropriée (markdown). Il s'afficheront ensuite en dessous des fichiers dans le gitlab.

Onglets importants :

Repository/Graph permet de visualiser rapidement l'avancement des branches

Repository/Branches permet de vérifier les droits sur les branches

Merge Requests permet de proposer la mise en service de votre branche



Et dans la vrai vie ça donne quoi ?

Ce que je tape pour travailler

Ce que j'utilise dans la vraie vie dans cet ordre :

- git status
- git pull
- git branch
- git checkout emile (si je dois changer de branche)

Travail

- git status
- git add .
- git reset HEAD {fichier/dossier} (si j'ai pris un mauvais fichier)
- git status
- git commit -m "mon nouveau commit"
- git push

Je peux être fier de mon travail ! ! ! N'oubliez pas de parsemer votre travail de *git status* pour être sûr de ne pas faire n'importe quoi.
Git vous aide, lisez toutes les lignes qu'il vous dit, il propose souvent la bonne commande quand il y a un problème.