Utilisation GitLab

Sommaire

I-Définition....1
II-Commandes Principales....2
III-Principe de GitLab....3
IV- WorkFlow du Projet....5
A- Les bonnes pratiques / règles....5
B- 4 types de branchs....5
C- Merge....6
D- Issues....9
1-Composition du Board....9
2-Création d'issue....10
3-Description des Labels....11
4-Assignation d'une issue....12
5-Nommer votre branch....14
V- Sources et conseils....14

I-Définition

Branch : dossier virtuel qui contient une version du programme.

Repository : Repertoire qui contient toutes les branch

Commit : sauvegarde de votre programme en local.

GitLab : C'est un logiciel de gestion de versions décentralisé. Développé par GitLab Inc et créé par Dmitriy Zaporozhets et par Valery Sizov.

Merge : C'est l'action de fusionner deux branch.

Issues : C'est un ticket qui décrit une bug ou un problème et qui doit être résolu.

II-Commandes Principales

Tout d’abord toutes les commandes doivent être entrées dans le terminal.

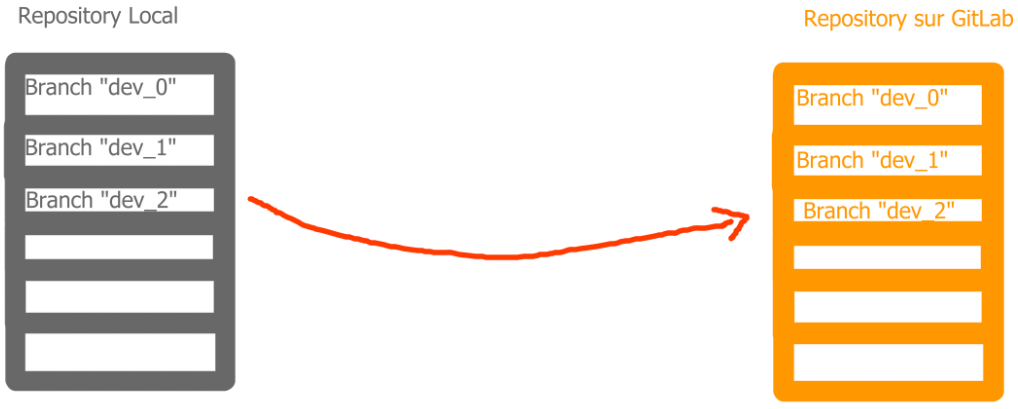
git pull origin nom_de_la_branch	Actualise votre programme avec celui qui est stocké sur la branche
git checkout -b « nom_de_la_branch »	Créer en local une branch nommée comme indiquée
git checkout nom_de_la_branch	Bascule sur la branch locale indiquée
git add .	Ajoute TOUS les fichiers ajouté ou modifiés au prochain commit.
git commit -m « nom de la sauvegarde »	Créer une sauvegarde local de votre programme nommé comme indiqué.
git push origin nom_de_la_branch	Pousse vos commit sur la branch indiquée. Votre branch local et sur le repository doivent être nommé de la même façon.
git status	Affiche tous les fichiers ajouté ou modifié
git fetch	Permet de localiser toutes les branchs du répository.

III-Principe de GitLab

Pour un projet il y a 2 repository, un est local donc sur votre machine, l'autre est versionné donc en sur gitLab.

Lorsque vous faites la commande [git checkout -b « dev_2 »] vous demandez la création de la branch dev_2 , elle sera créée en local

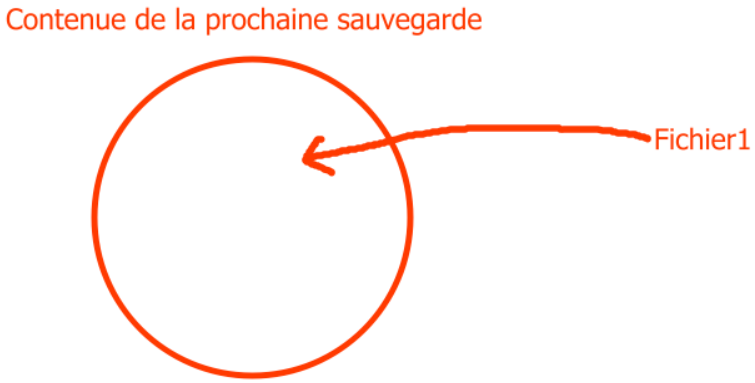
Pour créer la branch sur le Repository GitLab vous devrez exécuter la commande [git push origin dev_2]. Il est conseillé de le faire tout de suite après la création de la branch.



Après avoir modifier des éléments du projet vous pouvez faire la commande [git status] afin de connaître tous vos fichiers ajoutés ou modifiés.

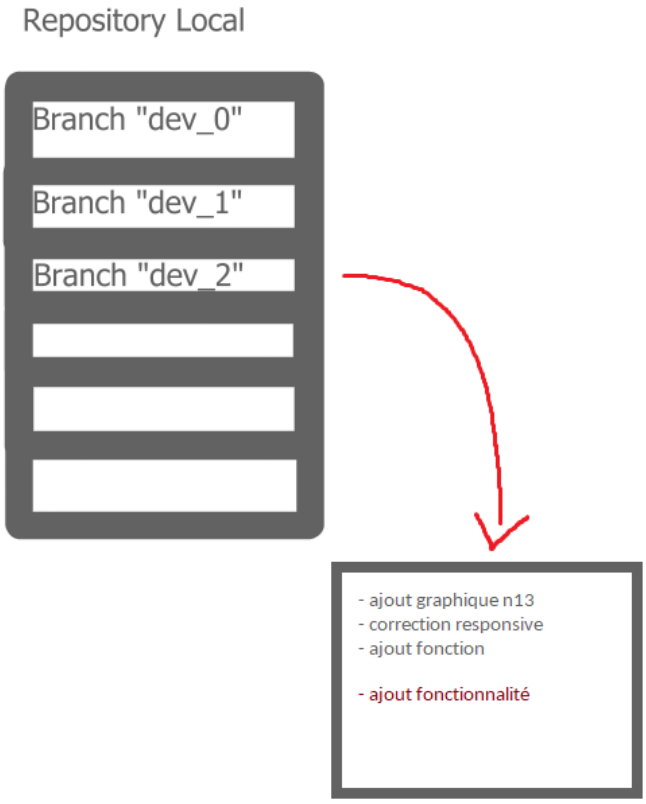
Pour ajouter des fichiers à votre prochaine sauvegarde vous devrez faire une des commandes :

[git add .] (ajoute TOUS les fichiers) [git add fichier1] (ajoute uniquement le fichier1)



Une fois des fichiers ajoutés, nous pouvons faire un commit,

[git commit -m « Ajout de fonctionnalité »]



Une fois que vous avez fini tous ce qu'il fallait faire sur votre branch vous pouvez la pousser sur le répository gitLab avec la commande [git push origin dev_2] . Ce sera la version de votre branch enregistrer dans votre dernier commit qui sera poussé sur la branch dev_2.

Pour importer des changements fait sur une branch du répository gitlab vous pouvez faire

[git pull origin nom_de_la_branch] . La branch sera importé sur votre branch local actuel. Donc si vous êtes sur la branch dev_0 et que vous faites [git pull origin dev_1] la branch dev_1 du gitLab sera importé sur votre branch local dev_0 .

IV- WorkFlow du Projet

A- Les bonnes pratiques / règles

Lorsqu'on nomme une branch son nom devra être en MINUSCULE, sans accent ou caractères spéciaux et avoir des _ à la place des espaces. Elles auront la logique suivante :

- Si c'est une fonctionnalité : nom_de_la_fonctionnalité (ex : creation_user)

-Si c'est une Issue : issue_numéro_de_l'issue (ex : issue_13)

On ne push que sur la branch qui a le même nom que notre branch locale.

Lors d'un Merge dans testing on ne supprime pas la branch d'origin et on ne fait pas de squash commit.

Avant d'être merge dans testing un minimum de test devront être fait pour vérifier s'il n'y a pas de bug évidant.

Une fois qu'un merge dans testing a été fait on le signale sur le discord pour que toute l'équipe fasse un commit puis pull testing afin d'éviter des conflits.

En cas de conflits garder en priorité les lignes importées quitte à refaire les vôtres.

Les merges dans master seront fait par le responsable gitLab.

B- 4 types de branchs

Le workflow sera divisé en 4 parties :

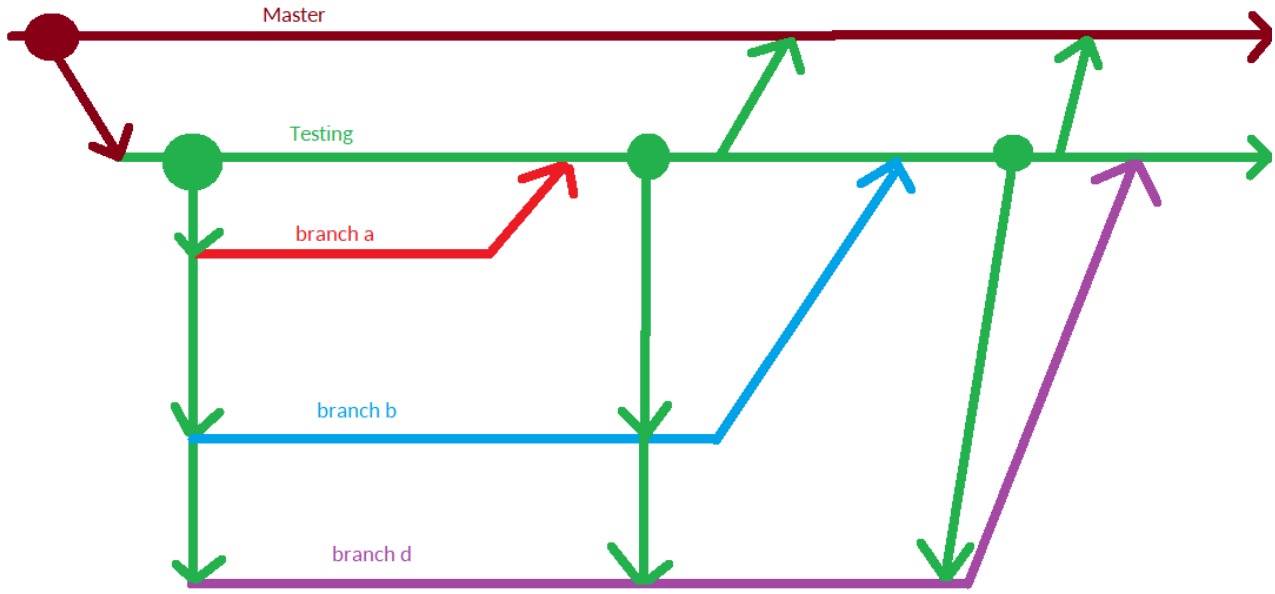
master : la branch principale qui sera la version la plus aboutie du projet.

testing : la branch sur laquelle les nouvelles fonctionnalités seront disponibles pour être testés. Une fois les fonctionnalités validées elle pourra être merge dans master.

nom_de_la_fonctionnalité : branch sur laquelle une fonctionnalité sera développé puis merge dans testing

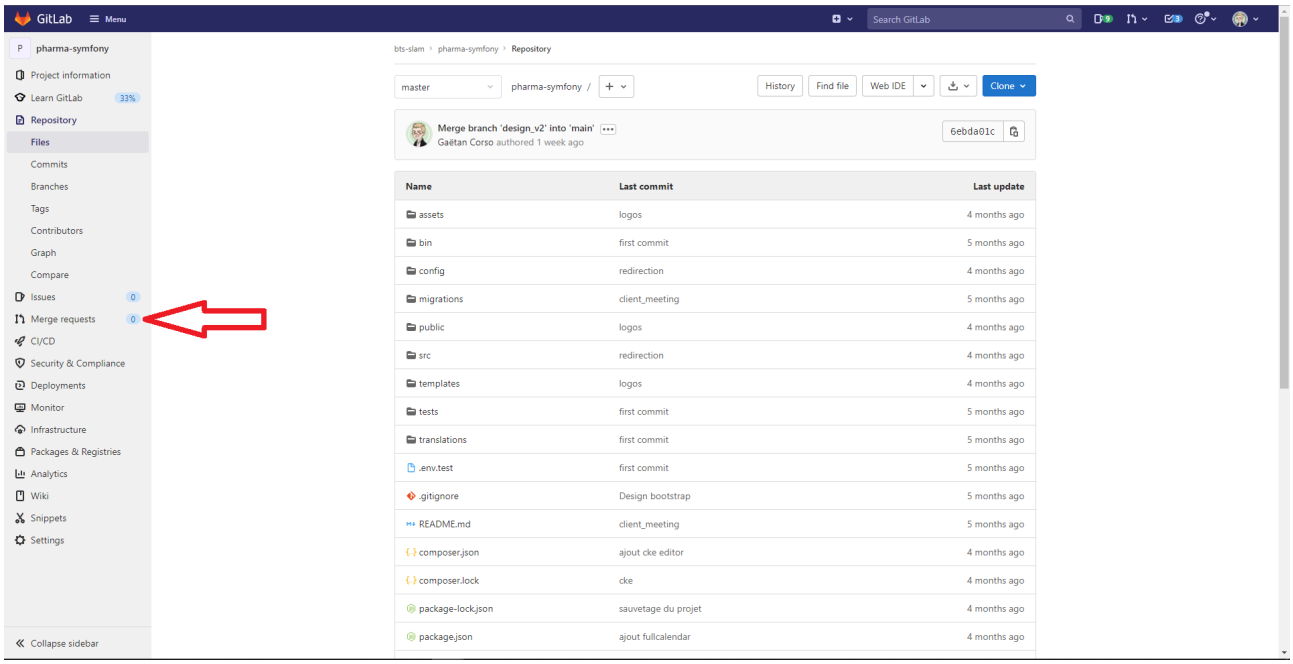
issue_XXX : issue suivie de son numéros, branch sur laquelle l'issue numéro XXX sera résolue puis merge dans testing.

C- Merge

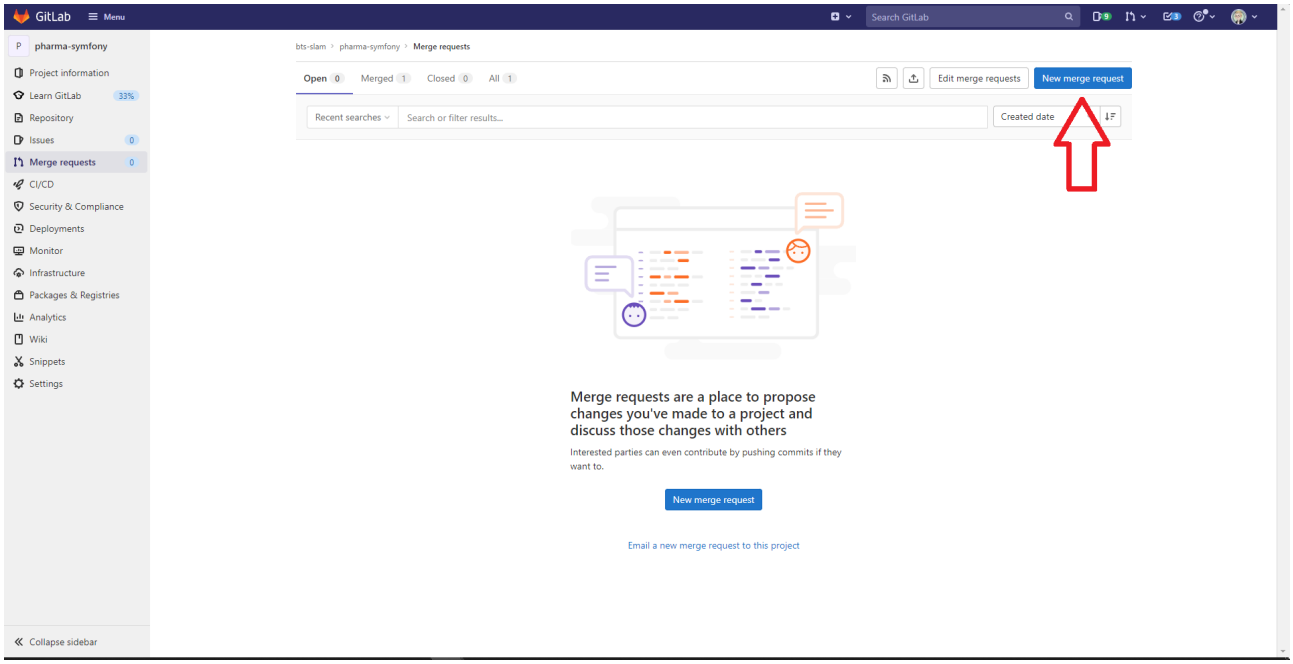


Une branch de développement devra être merge dans testing une fois terminé, puis une fois les tests sur testing fini, testing sera merge sur master.

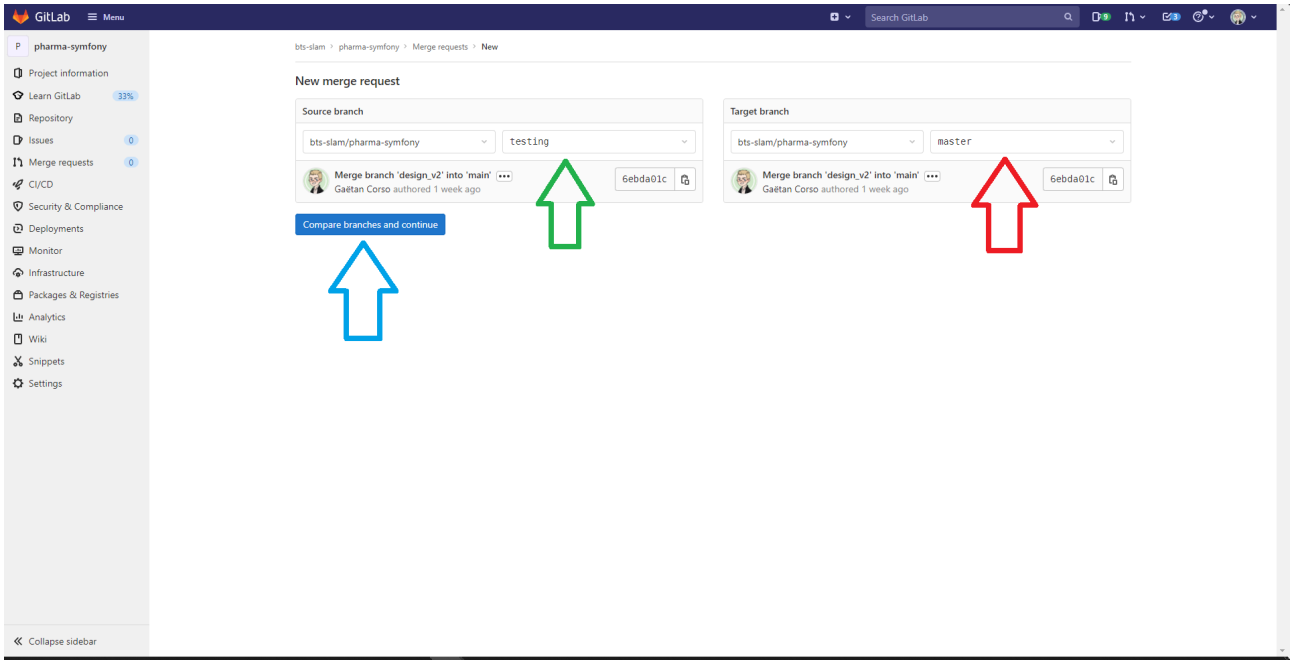
Une fois votre dernier commit et push effectué vous pourrez créer une merge request pour cela allez sur gitLab dans l'onglet merge request.



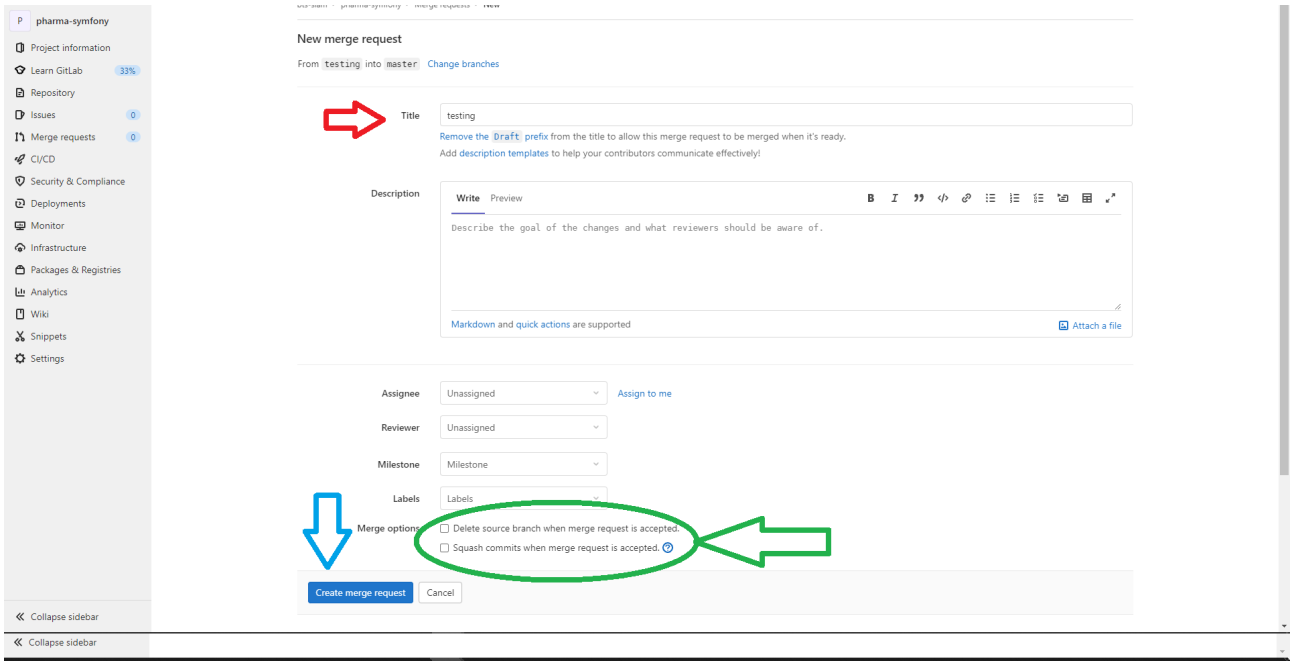
Puis cliquez sur New merge request



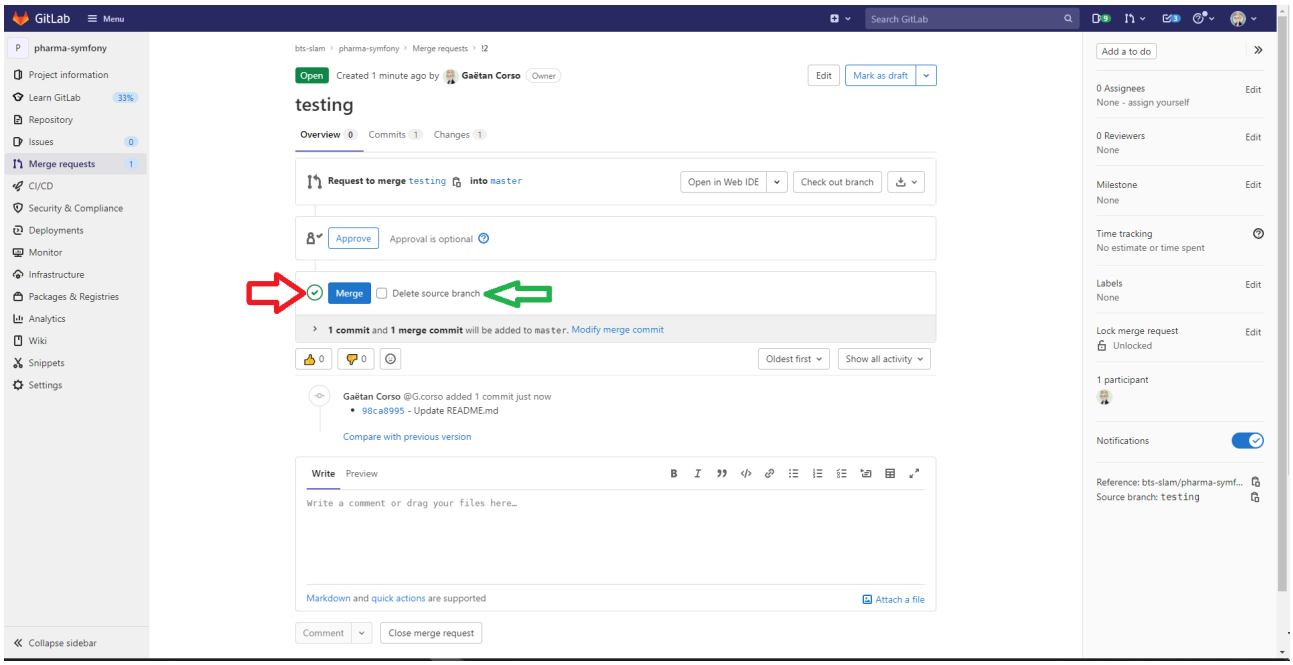
Puis sélectionnez votre branch et la branch dans laquelle vous voulez la merge et cliquez sur Compare branches and continue.



Ensuite mettez en titre le nom de votre branch, décochez les 2 merges options et validez en cliquant sur Create merge request.



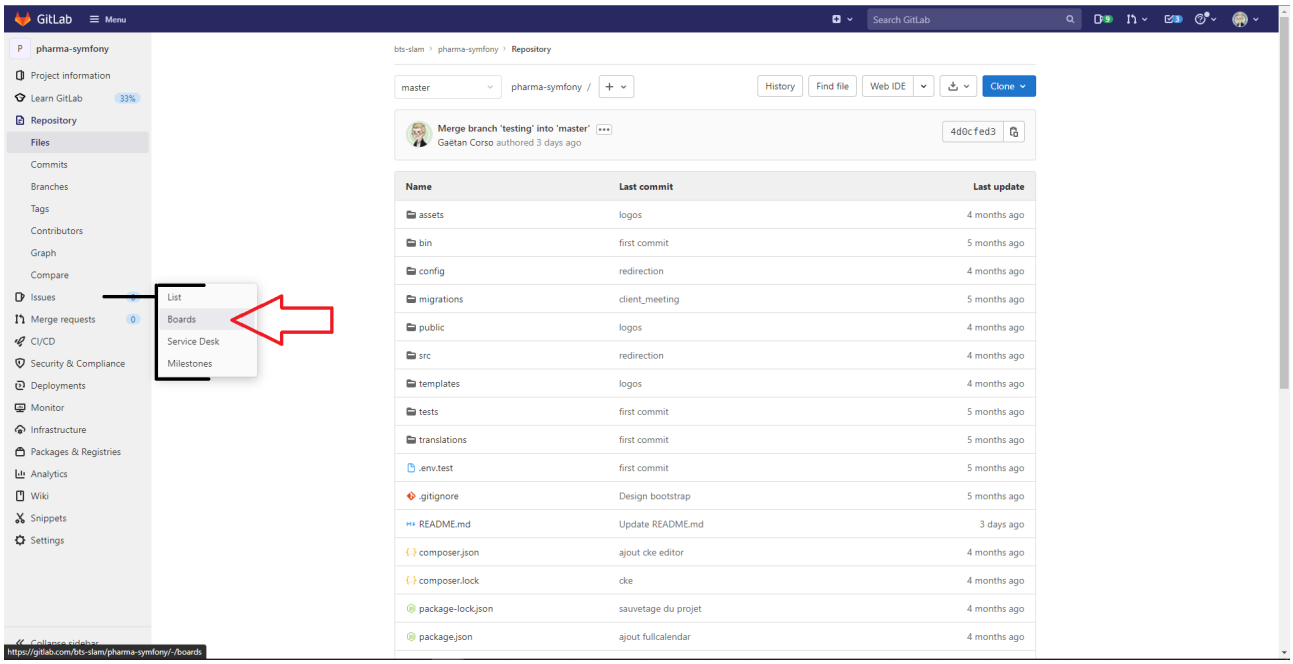
Si tout va bien vous n'avez plus qu'à vérifier que delete source branch est DECOCHE puis à cliquer sur Merge. Si on vous indique des conflits vous devrait les résoudre dans votre programme à la main puis commit et push puis revenir sur la merge request.



Ensuite prévenez l'équipe grâce à discord.

D- Issues

Pour accéder aux Issues du projet allez sur le répository du projet puis cliquez sur « board » dans l'onglet issues.



1-Composition du Board

Le Board est composé de 6 Tableaux différents,

Open : qui répertorie toutes les nouvelles issues.

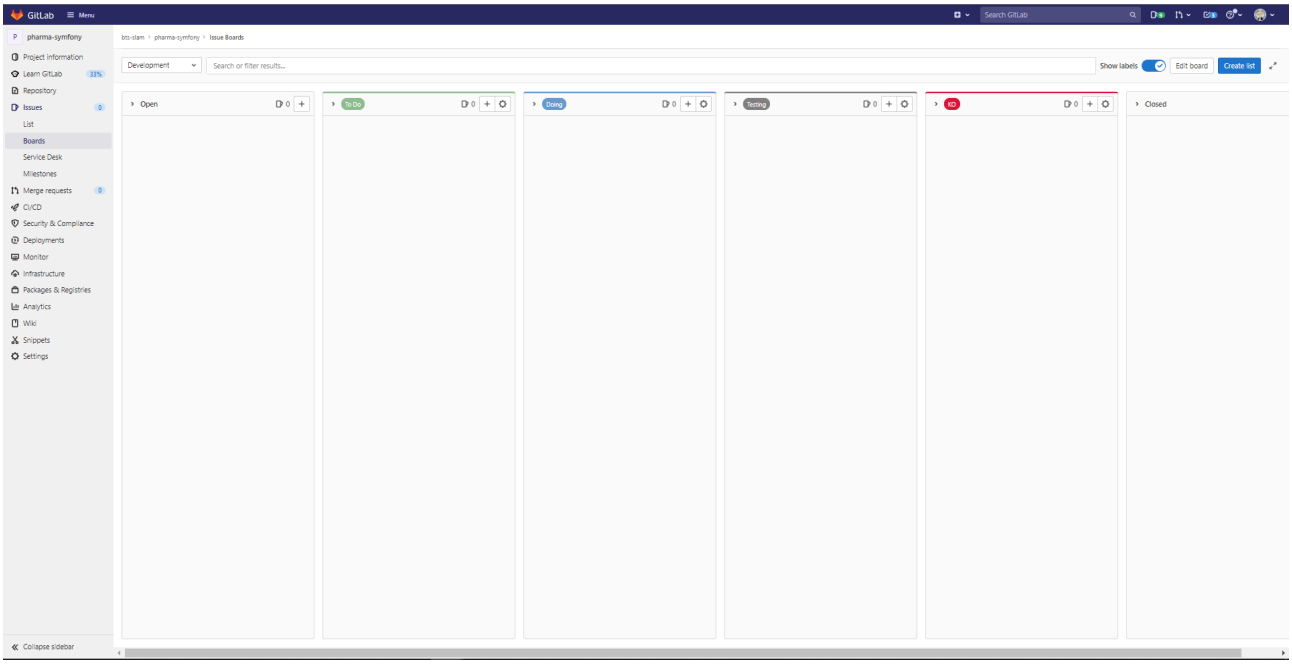
To Do : qui répertorie toutes les issues qui doivent être faites prochainement par les équipes.

Doing : qui répertorie toutes les issues qui sont en cours de traitement.

Testing : qui répertorie toutes les issues qui ont était corrigé et en attente d'être testés sur la branch testing.

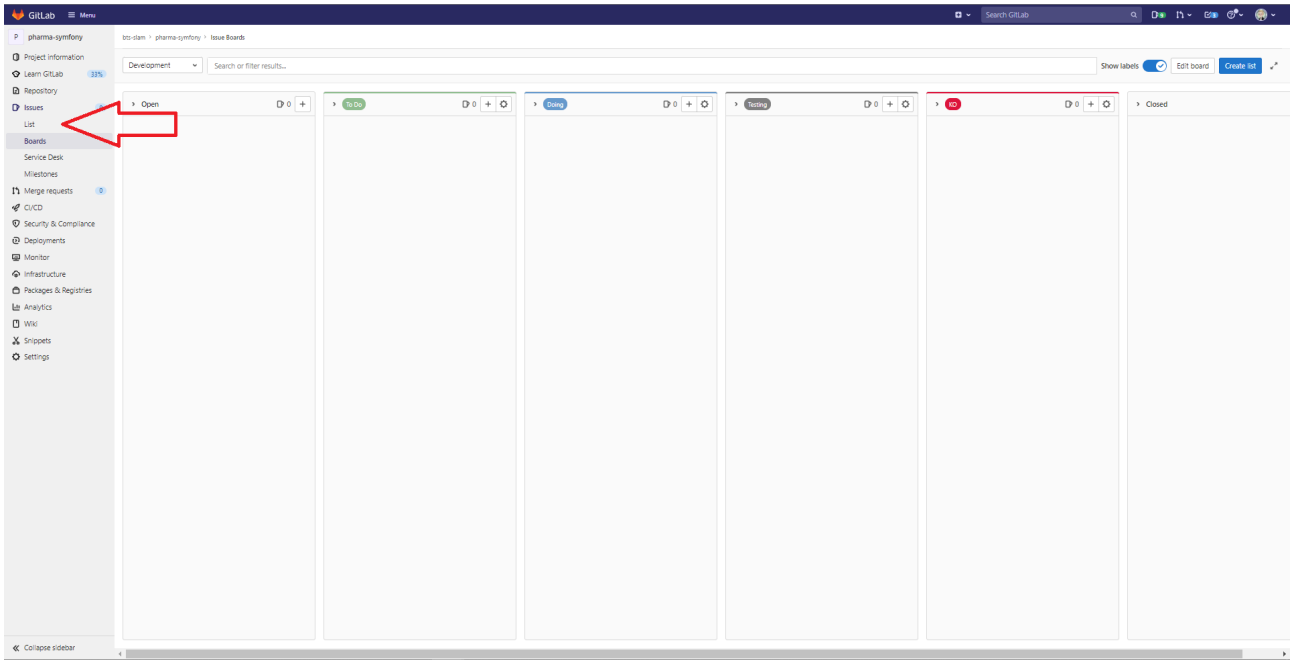
KO : qui répertorie toutes les issues qui présentent encore des bugs ou défaut sur la branch testing, elles devront donc être retravaillé. Quand une issue est mise en KO un commentaire dans l'issue devra être fait justifiant qu'elle soit mise en KO.

Close : qui répertorie toutes les issues qui ont été corrigé et approuvé.

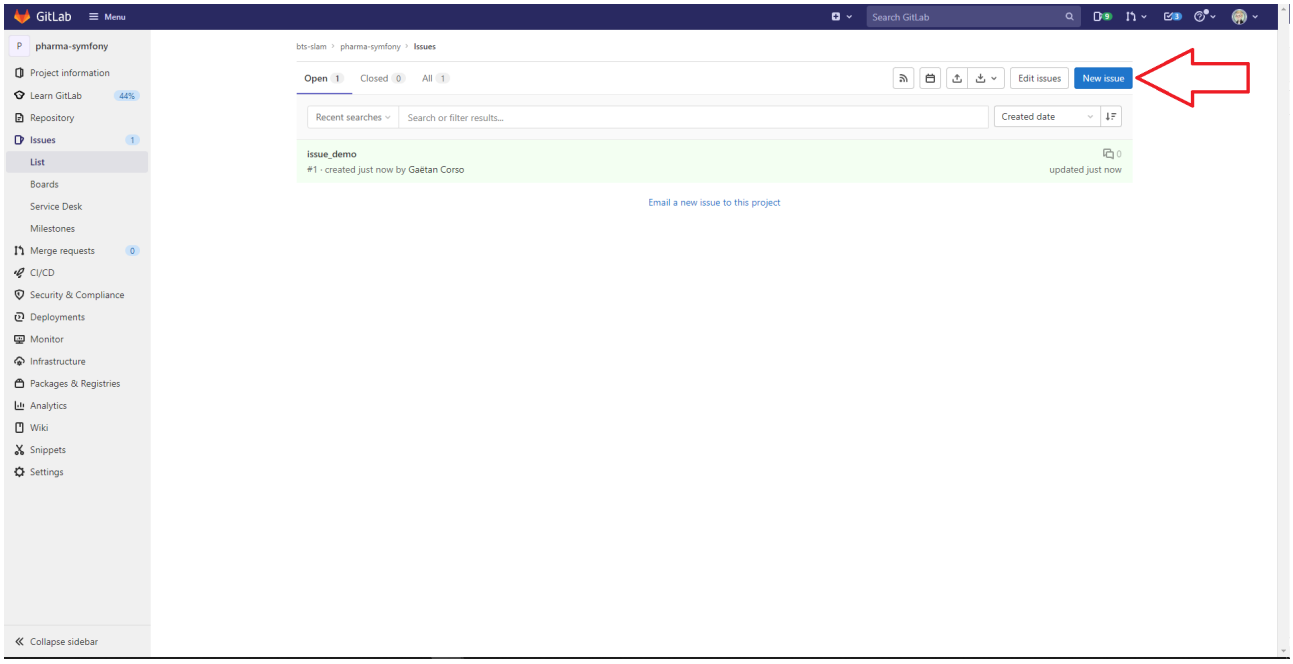


2-Création d’issue

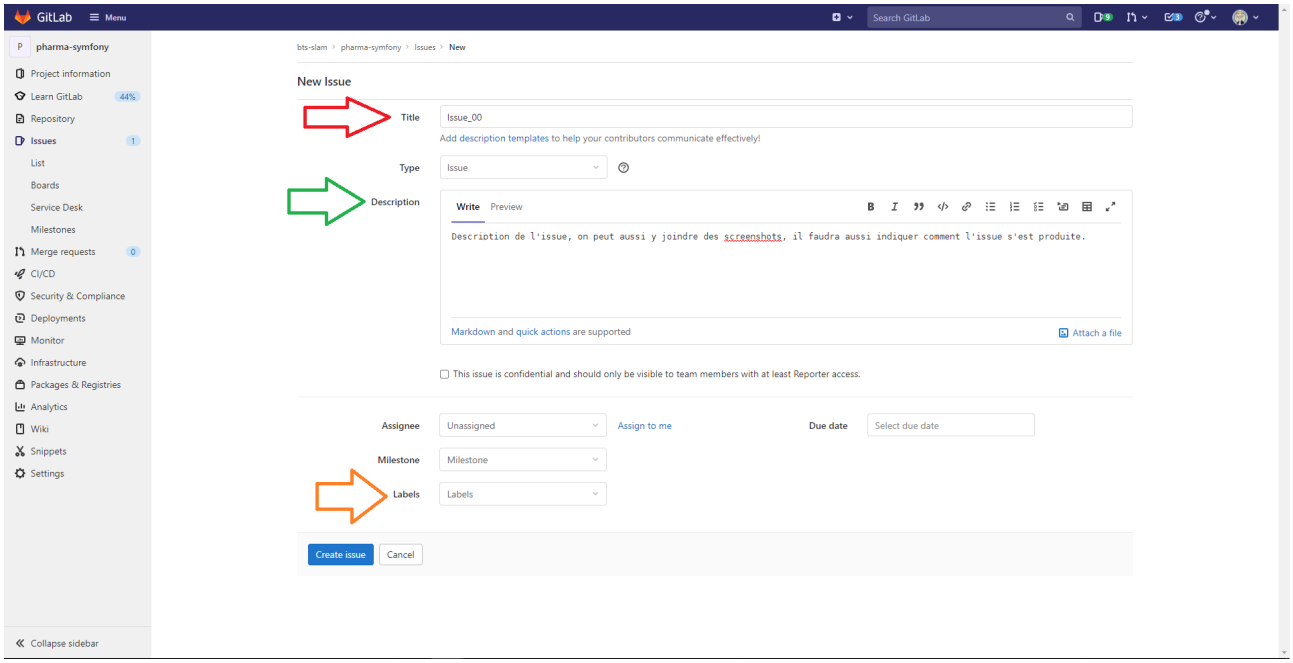
On peut créer une issue en cliquant list.



Puis sur le bouton New Issue.



Vous devrez renseigner, le titre, la description, ainsi que le label. Puis cliquer sur Create Issue



3-Description des Labels

Il y a 4 labels que vous pourrez mettre lors de la création d'une issue :

Discussion : qui indique que l'issue est encore en discussion elle n'est donc pas encore à faire.

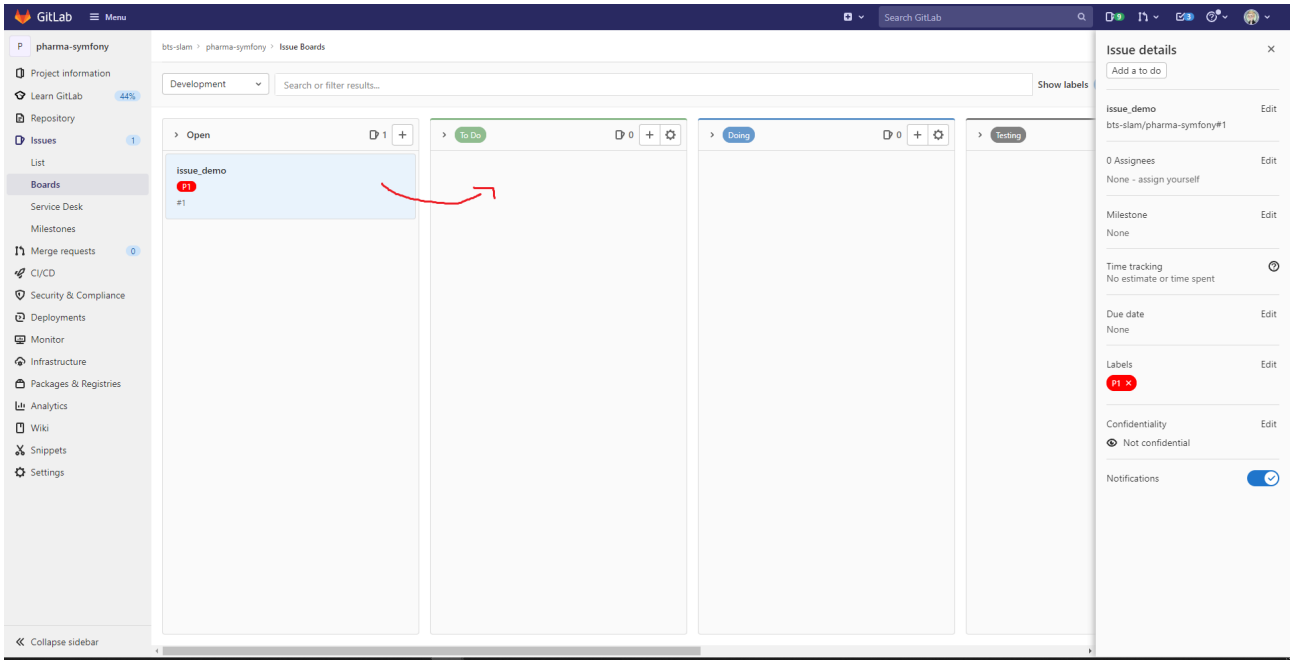
P1 : qui indique que l'issue est de TRÈS HAUTE priorité, elle devra donc être faite au plus tôt.

P2 : qui indique que l'issue est de Haute priorité, elle devra donc être faite après les P1.

P3 : qui indique que l'issue est de basse priorité, elle devra donc être faite après les P2.

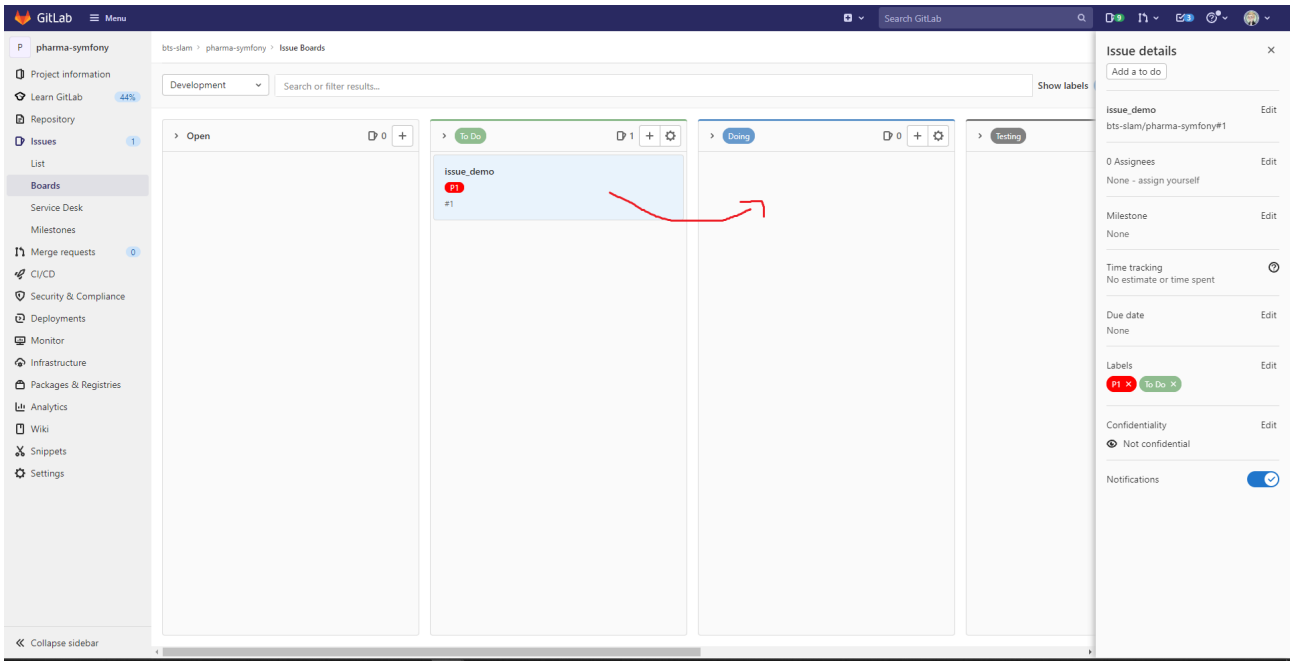
Les autres labels ne sont pas des labels que vous devrez attribuer.

Une fois de retour sur le board notre issue apparaît. Et nous pouvons la glisser dans le ToDO.

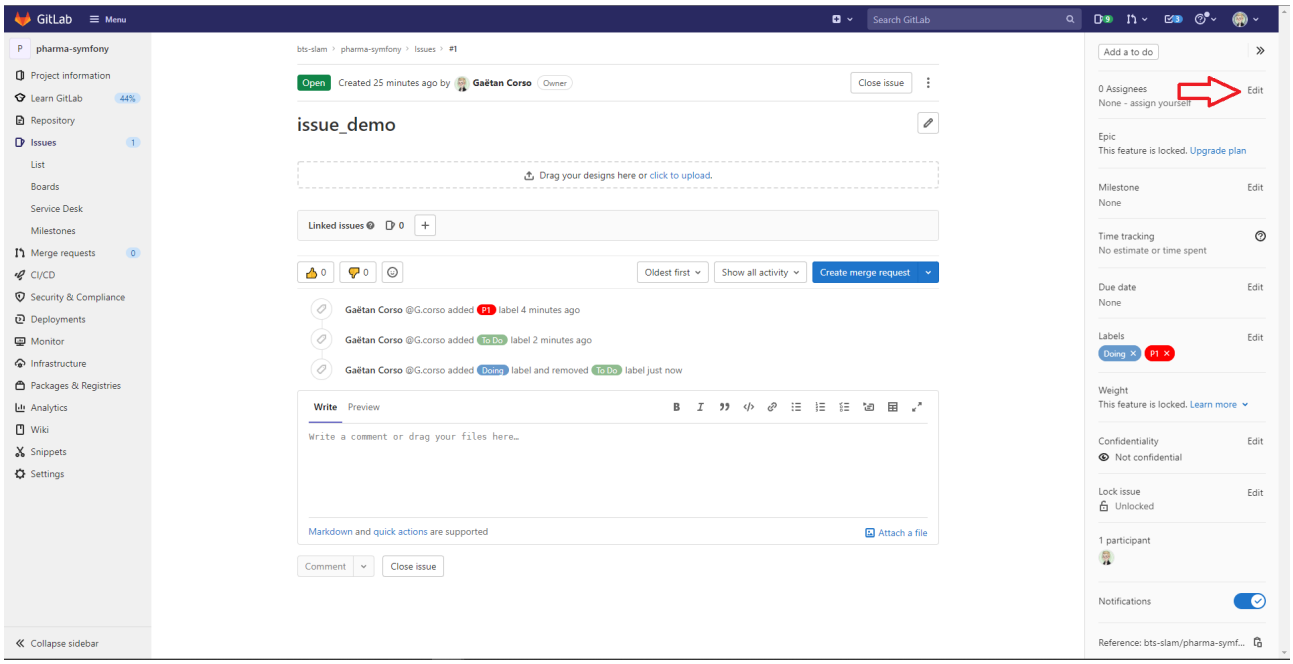


4-Assignation d’une issue

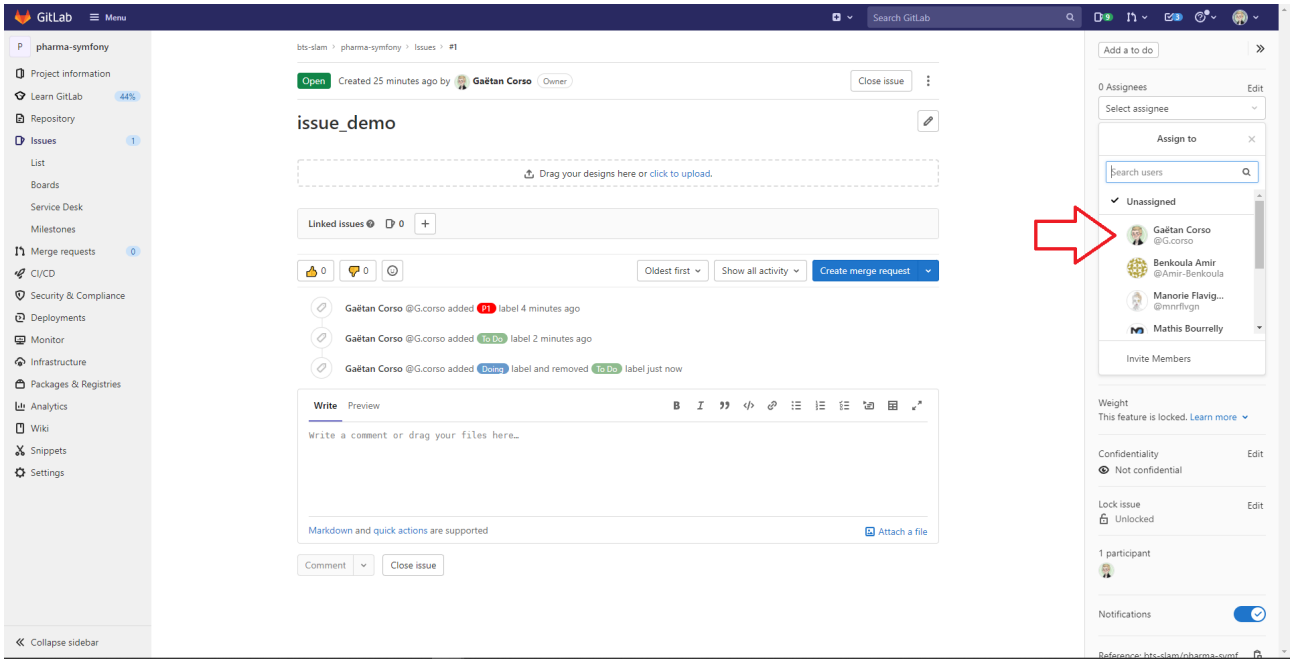
Lorsque vous voudrez vous occuper d’une issue vous devrait la faire glisser dans la colonne Doing et vous mettre en assignation. Pour cela vous devrait cliquer sur le titre de l’issue.



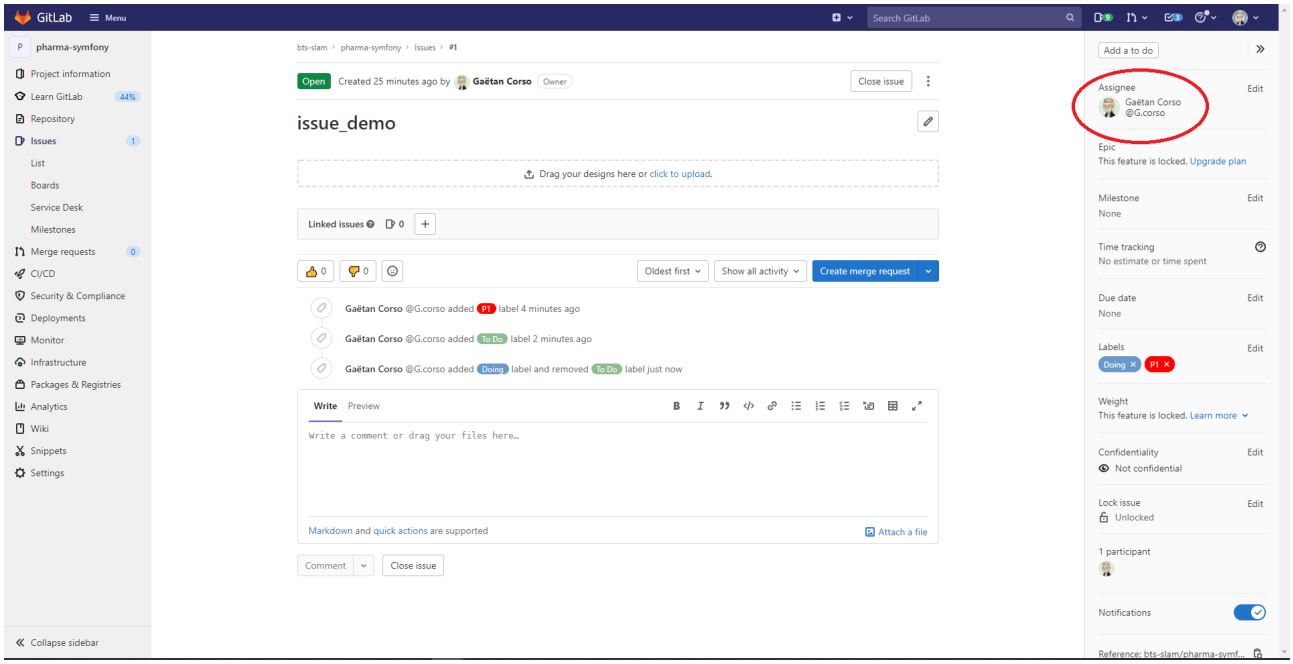
Une fois sur le descriptif de l’issue cliquez sur edit à droite.



Et cliquez sur votre profil.

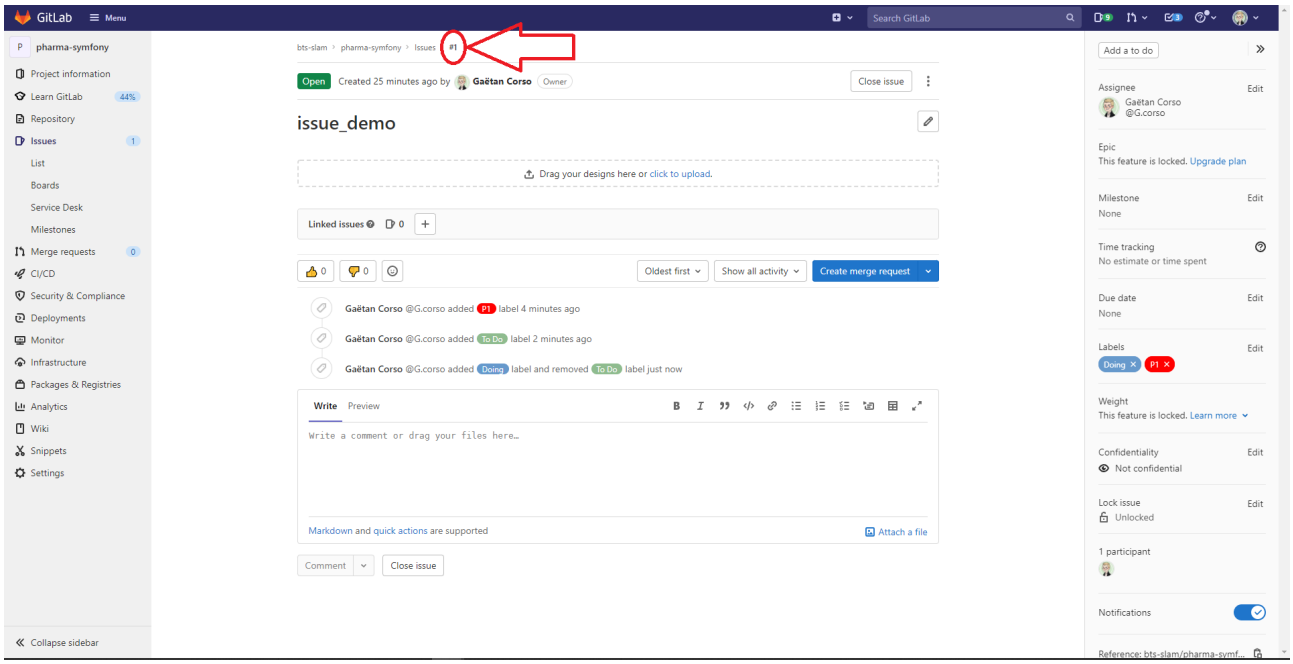


Maintenant vous êtes assigné.



5-Nommer votre branch

Pour nommer votre branch vous utiliserez le numéro de l'issue indiqué ici :



La branch de l'issue_demo sera : issue_1

(git checkout -b « issue_1 » pour créer la branch).

V- Sources et conseils

Certaines définitions : Wikipédia

Commande : documentation officielle de git

Système de workflow : Inspiré de celui utilisé chez Nablaware.

Screenshot : effectué sur le répository du projet par Gaëtan Corso.

Pour plus d'information sur le fonctionnement de git veuillez d'abord relire plusieurs fois la documentation si dessus, puis en cas d'incompréhension ou de problème persistant contacter Gaëtan Corso sur discord.