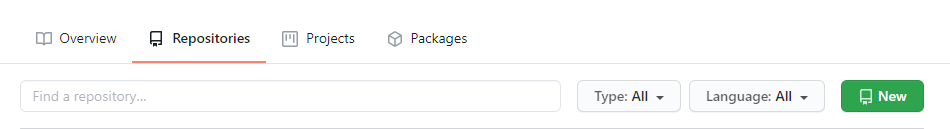
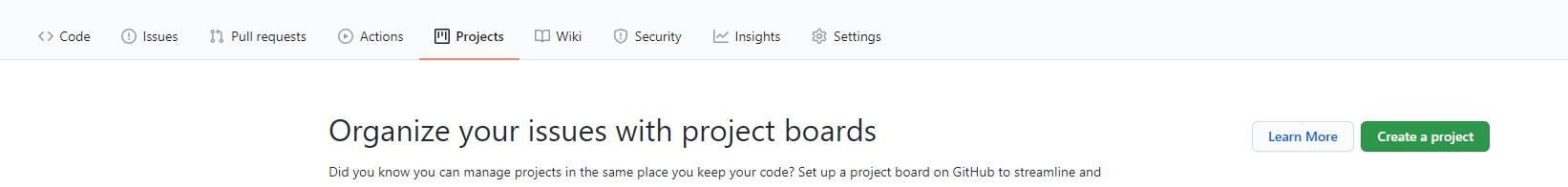


|  |
| --- |
| Gestion de projets sur GitHub |
|  |
| 5 juillet  Concepteur Développeur d’Application  Créé par : Jonathan JEANNIARD Contributeurs : Aurélien BOUDIER |

1. Création d’un repository sur GitHub

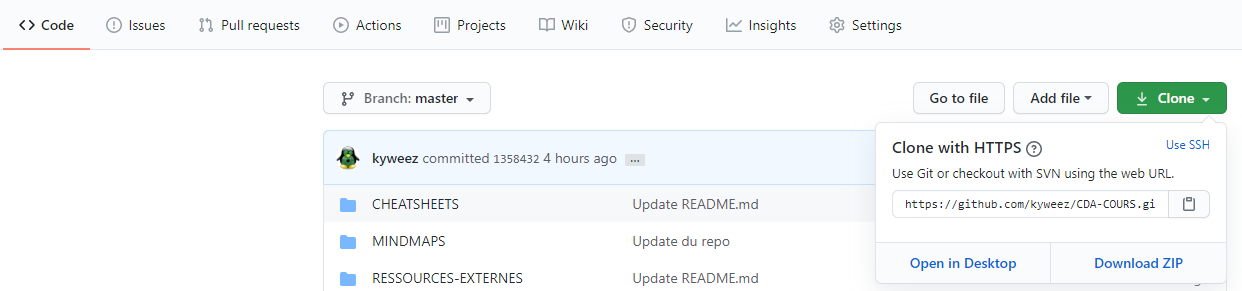


1. Création d’un projet



1. Récupération du repository distant en local

Dans un premier temps, il faut s’assurer que tous les collaborateurs du projet ont initialisé le dépôt distant en local (git clone).



1. Exemples de branches

Les branches de référence sont les suivantes :

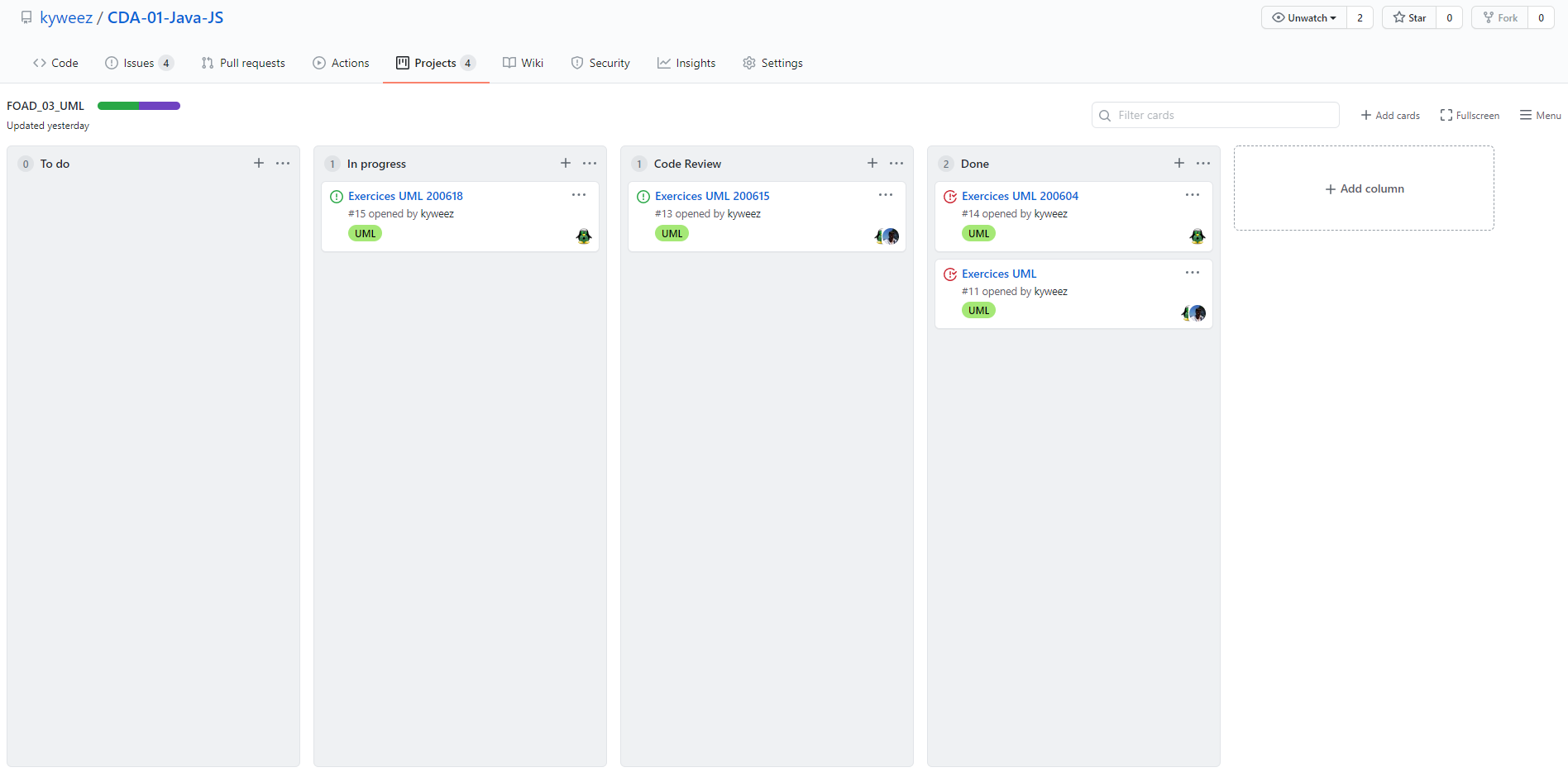
* Master : contient le rendu final et fonctionnel du projet
* Develop : point de rassemblement de tous les merges des différentes features passés en review
* Features : branche de développement des features. Ces branches sont push sur la branche develop après un code review de la team.

1. Le Kanban du projet

Les projets permettent une disposition de différentes colonnes en mode Kanban. Dans nos projets nous utiliserons 4 colonnes :

* To Do
* In progress
* Code Review
* Done

Ces 4 colonnes permettent d’avoir un suivi en temps réel du projet.



* 1. To Do

La colonne "Todo" est paramétrée de telle sorte à récupérer les tickets de l'onglet "issues" et les transformer en tache à faire.

* 1. In Progress

Chaque ticket est attribué à un (ou plusieurs) développeur(s), qui glissera ce ticket dans la colonne "In progress" pour montrer qu’il travaille dessus.

* 1. Code review

Après avoir développé la feature et rempli le ticket dans son ensemble, le développeur glisse le ticket dans "Code Review" et l’associe à un Pull Request afin d’en discuter avec les collègues du projet.

* 1. Done

Une fois la code review effectuée, le merge est fait avec la branche supérieure et le ticket est fermé puis glissé dans la colonne "Done".

1. Le travail sur des branches

Pour faciliter le travail en équipe, il est déconseillé de travailler tous ensemble sur la meme branche. Il est d’ailleurs frequent qu’on ne puisse pas travailler sur la branche master ou develop car ces dernières sont protégées pour des raisons évidentes de sécurité. La branche master contiendra une version finale fonctionnelle du projet, qui sera en production. La branche develop contiendra les nouvelles features. Cette branche sera souvent fonctionnelle, mais pas encore pushed sur la branche master car elle nécessitera plus de tests ou l’ajout supplémentaires de features avant de faire évoluer la version definitive.

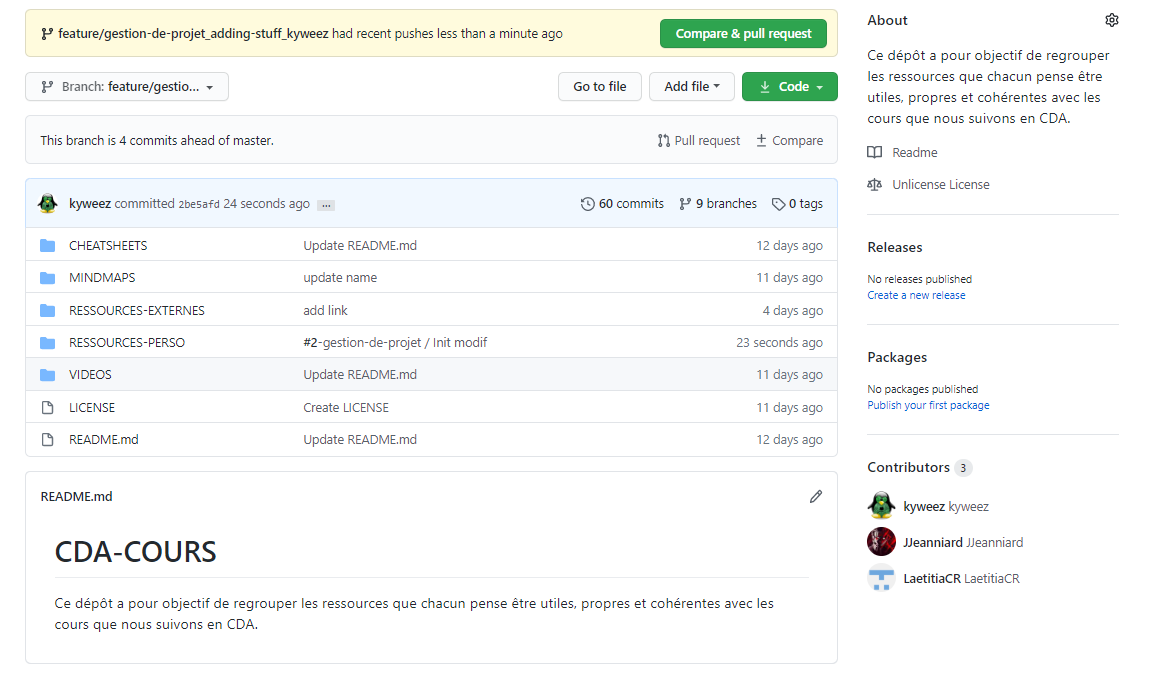
C’est pour cette raison qu’il est recommandé de créer une branche par nouvelle feature, que nous mergerons avec la branche develop quand cette feature sera fonctionnelle et que les effets de bords seront contrôlés. Il faudra souvent faire un pull request vers la branche develop.

Pour la gestion de branches, git flow est un excellent outil. [Un tutoriel est disponible ici](https://github.com/kyweez/CDA-RESSOURCES/blob/master/RESSOURCES-PERSO/gestion-de-branche-avec-flow.pdf).

1. Le pull request

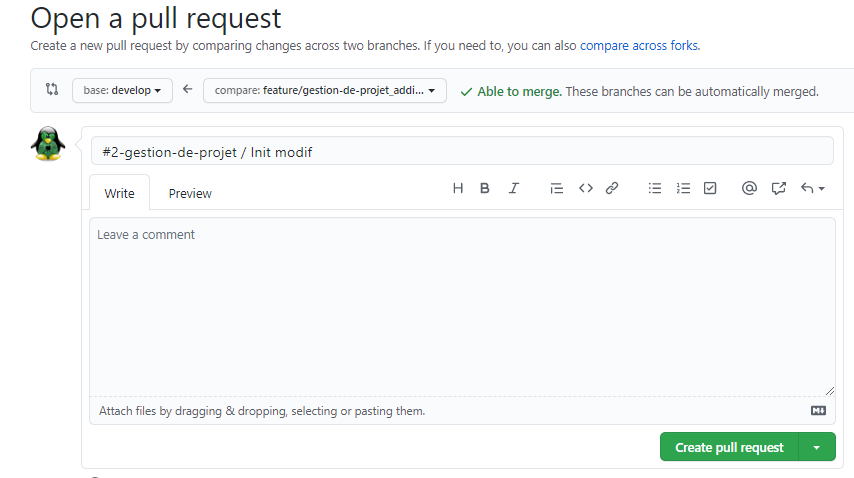
Lorsque l’on a fini notre travail sur notre branche, on souhaite faire une pull request vers la branche develop. Rien de plus simple : 2 façons de faire :

Cliquer ici



Se positionner sur notre branche et cliquer sur pull request

Ensuite s’assurer de faire la pull request de la bonne branche (1) vers la bonne branche (2) ! Marquer un petit commentaire pour les collègues et créer la pull request (3) :



**3**

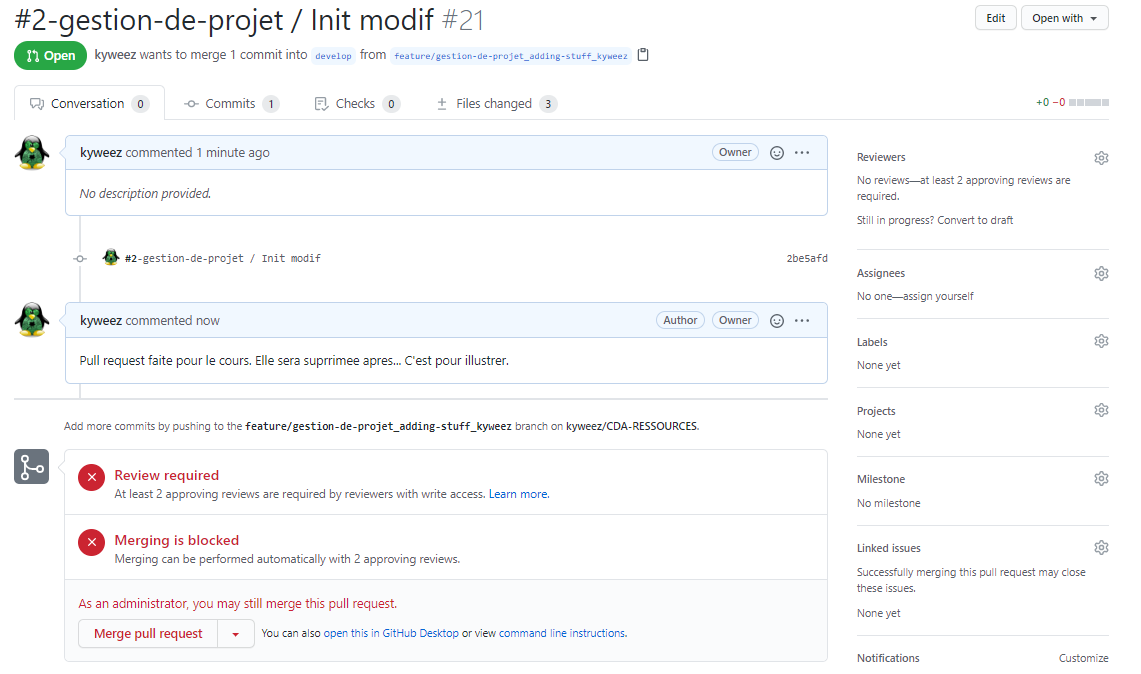
**2**

**1**

1. Le code review

Une fois la pull request faite si le repository est paramétré correctement, des reviewers seront nécessaires pour valider la pull request avant de faire un merge.

* On peut assigner des reviewers. (1)
* On peut relier la pull request a un projet (dans un kanban de projet automatique, la tâche se déplacera toute seule). (2)
* On peut relier la pull request à un ticket (qui se clôturera si elle est acceptée). (3)
* Si les reviewers nécessaires (2 dans ce cas-ci) ont accepté les modifications, on sera en mesure de cliquer sur merge. (4)



**4**

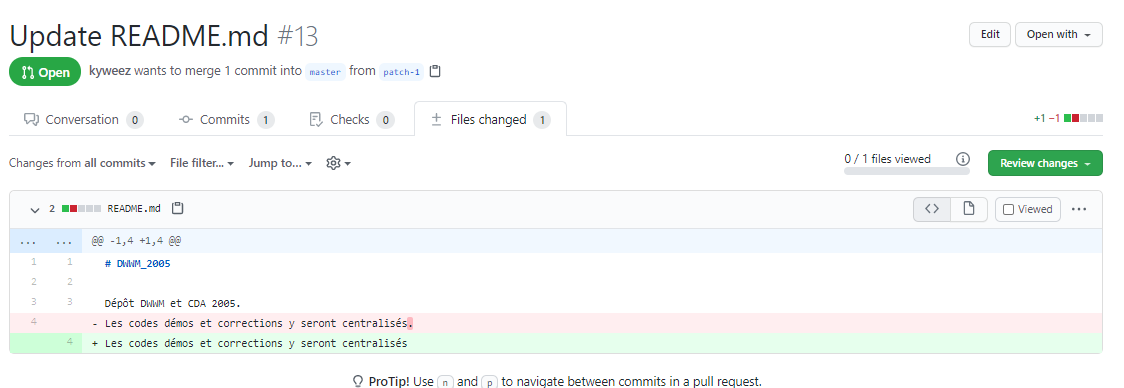
**3**

**2**

**1**

Coté reviewer :

Un reviewer se doit de vérifier les changements apportes. Il peut cliquer sur les onglets « Commits » (1) ou « Files changed » (2) pour observer les modifications. Une fois les vérifications faites, cliquer sur « Review changes » (3) :



Changements apportés

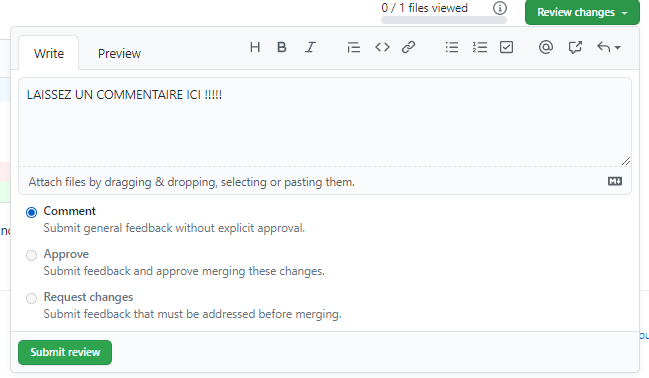
**3**

**2**

**1**

On peut simplement commenter, ou bien on peut approuver, ou encore demander des changements.

Quoi qu’il en soit, il faut laisser un commentaire pour valider la review, quel que soit le statut qu’on laisse :



Crédits :

|  |  |  |  |
| --- | --- | --- | --- |
| Action | Auteur | Descriptif | Date |
| Création du document | Jonathan JEANNIARD | Gestion de projet sur GitHub | 22/06/20 |
| Correction | Aurélien BOUDIER | Correction de l’orthographe et remise en page selon la trame | 23/06/20 |
| Ajout | Aurélien BOUDIER | Ajout de la partie sur le code review | 05/07/20 |