



Workflow P2



Rappels et précisions sur les rôles dans l' équipe



Les rôles

- L'équipe sera composée
 - d'un **Scrum Master**
 - d'un **Product Owner**
 - de **développeurs**
 - de **code reviewers**

(on peut porter plusieurs casquettes)

- À chaque sprint (1 ou 2 semaines), les SM, PO peuvent tourner, de manière à ce que tout le monde occupe chaque rôle au moins une fois.
- Les rôles ont tous des responsabilités particulières. En fin de semaine, il pourra vous être demandé "**Qu'avez vous mis en oeuvre cette semaine pour accomplir vos missions ?**")

⇒ Renseignez vos rôles dans [ce fichier](#)



Que signifie être PO ?

- C'est le garant de la **qualité fonctionnelle** de l'application
- Il est l'interface avec le client/les utilisateurs, **il communique** avec eux **fréquemment** pour leur demander leur avis ou faire des propositions.
- Il s'assure que **toute l'équipe partage une vision non-ambigüe** de ce qui est attendu
- Il revoit, précise, découpe et priorise les US pour bien "**préparer le terrain**".





Que signifie être CR pour une feature ?

Avant de signaler aux formateurs qu'une fonctionnalité est "prête", les développeurs en charge des revues de code ont la responsabilité de valider certains points (en plus du fait qu'il n'y ait pas de bugs...), à savoir :

- **Pas de code commenté (appelé "code mort")**
- **Variables clairement nommées**
- **Code facilement compréhensible car l'intention est claire**
- **Pas de fichiers n'ayant aucun lien avec la story**
- **Respect des normes de codage (ESLint, W3C validators ...)**
- **Le PO a vu et approuvé la partie fonctionnelle de la story**

Le CR est presque "le dernier rempart" avant la mise en commun du code. Il a l'oeil aiguisé afin de veiller à la bonne santé technique du projet. S'il prend son rôle au sérieux, il fera gagner énormément de temps (et de compétences) à l'équipe sur le long terme.

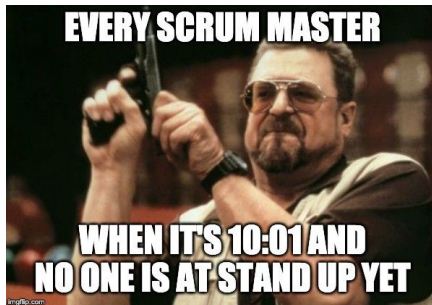
⇒ **Guide :** [comment faire une bonne review](#)





Que signifie être SM ?

- N'est **pas un chef**
- **anime les différents rituels** de Scrum :
 - Mêlées quotidiennes
 - Revue de Sprint
 - Rétrospectives...
 - Planning pokers
- Aide l'équipe à avancer de manière autonome dans une démarche d'**amélioration continue**
- S'assurer de l'**harmonie au sein de l'équipe**





Que signifie être dev ?

- Développeur != programmeur
- Produit du code dans un but précis
- Produit du code respectant les standards et bonnes pratiques
- Fiable \Rightarrow Sait tester son travail, imaginer les cas limites
- A l'écoute, rôle de conseil
- Veille techno





Gestion de projet : outils



Le backlog

Backlog

Fichier

Édition

Afficher

Insertion

Format

Données

Outils

Modules complémentaires

Aide

100%

Lecture seule

N° US

	A	B	C	D	E	F	G	H	I	J	K	
1	N° US	SPRINT	TITRE	HISTOIRE UTILISATEUR	IMPORTANCE	POIDS	CRITERES DE VALIDATION	TESTS DE VALIDATION	DEVELOPPEUR			
2	1	1	FICHE PDL	En tant qu'utilisateur Je peux accéder à une fiche PDL afin de visualiser les infos principales	0	0	- Je peux accéder à une fiche PDL par son url - Je peux lire les infos directes du PDL	- L'entité PDL est créée - Le controller pour les pdl existe - La méthode du controller est créée - La vue Twig affiche bien les infos	Erwann			
3	2	1		En tant qu'utilisateur Je peux accéder aux informations du client sur une fiche PDL	0	0	- Je peux accéder aux infos du client sur une fiche pdl	- l'entité client existe - l'entité a une relation avec l'entité pdl - Les infos s'affichent dans la vue Pdl	Matthieu			
4	3	1		En tant qu'utilisateur Je peux visualiser sur une carte le PDL afin de savoir où il se trouve	0	0			Erwann			
5	4	2		En tant qu'utilisateur Je peux accéder au poste du PDL depuis sa fiche	0	0	- Je peux accéder au poste du PDL en cliquant sur un lien depuis la fiche PDL		Erwann			
6	25	3		En tant qu'utilisateur Je peux voir les infos du poste sur la fiche pdl	0	20	je vois les infos du poste sur la fiche pdl		Louise			
7												
8	5	2		En tant qu'utilisateur Je peux voir le poste du PDL sur la carte	0	0			Erwann			
9	6	2		En tant qu'utilisateur Je peux visualiser l'historique des interventions sur la fiche PDL	0	0	- Je vois la liste des interventions sur la fiche PDL concernant ce PDL		Matthieu			
10												
11	10	1	INTERVENTIONS	En tant qu'utilisateur, Je peux visualiser toutes les interventions	0	0	- j'accède à un tableau recensant toutes les interventions	- L'entité Intervention est créée - Le controller et la méthode existent - La vue affiche un tableau	Thomas			
12	7	3		En tant qu'utilisateur, Je peux parcourir les interventions par page	0	90	Je peux naviguer entre les différentes page de la liste		Nicolas			
13	8	3		En tant qu'utilisateur, Je peux trier toutes les interventions	0	90	Je peux cliquer sur une colonne et trier les interventions		Matthieu			
14	26	4		En tant qu'utilisateur, Je peux voir toutes les interventions sous forme de marqueur sur la carte	0	0			Erwann			
15	9	6		En tant qu'utilisateur, Je peux filtrer toutes les interventions	0	0			Nicolas			

Backlog

Stats

Explorer



Issues

- Les issues représentent les prochains développements à effectuer (ajout de fonctionnalité, correction d'un bug, ...)

The screenshot shows the GitHub interface for the 'open-it / pendujava' repository. The top navigation bar includes links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. Below this, the repository name is displayed along with 'Watch', 'Star', and 'Fork' buttons. A secondary navigation bar highlights 'Issues' with a count of 1, alongside 'Code', 'Pull requests', 'Projects', 'Wiki', 'Insights', and 'Settings'. A search bar with the filter 'is:issue is:open' and buttons for 'Labels' and 'Milestones' are present. A green 'New issue' button is on the right. The main content area shows a summary of '1 Open' and '0 Closed' issues, followed by a table of open issues. The first issue is titled 'un bug sur le pendu' and was opened on July 6 by 'open-it'.

Author	Labels	Projects	Milestones	Assignee	Sort
open-it					

💡 **ProTip!** Type `g` `p` on any issue or pull request to go back to the pull request listing page.



Définir vos tâches au moment du Sprint Planning

Faites un petit poker :

- <https://www.nutcache.com/fr/blog/quest-ce-que-le-planning-poker-scrum/>
- <https://www.planitpoker.com/>
- <https://www.mountaingoatsoftware.com/blog/the-best-way-to-establish-a-baseline-when-playing-planning-poker>

Quelques conseils sur le "bon niveau de détails" :

- <https://manifesto.co.uk/how-much-detail-should-a-user-story-have/>
- <https://stackoverflow.com/questions/2896249/how-do-i-manage-specs-in-scrum>
- <https://www.nutcache.com/blog/agile-functional-specifications>

Evitez les "technical stories" :

- <https://seilevel.com/requirements/user-stories-technical-stories-agile-development-productmanagement>
- <https://www.extremeuncertainty.com/why-technical-user-stories-are-bad/>

DoD :

- <https://manifesto.co.uk/the-definition-of-ready/>



Estimer vos tâches au moment du Sprint Planning

P2G1 > WildCodeSchool/lyon-js-0320-p2-g1#2 > US2 : Titre us2

Write Preview

AA B i “ < > 🔗 ☰ ☷ ☹ @ 📌 ↶

Story

En tant que ..., je souhaite..., afin de ...

(Critères et considérations spécifiques)

...

(Scénarios de tests)

...

(Tâches)

- [] Faire ci
- [] Faire ça

Attach files by dragging & dropping, selecting or pasting them.

Cancel Update

Pipelines

P2G1

Sprint Backlog

Assignees

ComicScrip

Labels

None yet

Projects

None yet

Milestone

Sprint 1

Estimate

How complex is this issue?

Filter Estimates (or type to create one)

1

2

3

5

US2 : Titre us2 has no dependencies

+ add dep



Vous pouvez utiliser [ZenHub](#) (avec [son extension](#)) ou [Trello](#)

WildCodeSchool / [lyon-js-0320-p2-g1](#) / P2G1

Watch 12 Star 0 Fork 0

Code Issues 2 Pull requests 0 **ZenHub** Actions Projects 0 Wiki Security Insights Settings

P2G1

Repos (1/1) Labels Milestones Assignees Epics Releases Estimates Authors

Find Issues (+i) [New Issue](#)

Board Reports Roadmap **New**

Create... Edit Workspace Invite

View tutorials Shortcuts Open in web app Support and training Changelog

1 Issue - 0 Story Points **Product Backlog**

1 Issue - 0 Story Points **Sprint Backlog**

0 Issues - 0 Story Points **In Progress**

0 Issues - 0 Story Points **Review/QA**

0 Issues - 0 Story Points **Done**

0+ Issues - 0 Story Points **Closed**

lyon-js-0320-p2-g1 #1
US1 : Titre de l'histoire utilisateur avec l'identifiant 1

lyon-js-0320-p2-g1 #2
US2 : Titre us2
Sprint 1

In Progress
What the team is currently working on, ordered by priority.

Review/QA
Issues open to the team for review and testing. Code is ready to be deployed pending feedback.

Done
Issues that are tested and ready to be deployed to production.

[Load more issues...](#)



Créer des Milestones pour les différents sprints

WildCodeSchool / lyon-js-0320-p2-g1

Watch 12 Star 0 Fork 0

Code Issues 2 Pull requests 0 ZenHub Actions Projects 0 Wiki Security Insights Settings

Labels Milestones New milestone

1 Open 0 Closed Sort

Sprint 1

Due by April 17, 2020 Last updated less than a minute ago

0% complete 0 open 0 closed

Edit Close Delete



Pull request

- Afin de réintégrer ta User Story (qui a été faite sur une branche à part) sur la branche **dev**. Tu **DOIS** passer par une **PR** car les branches **dev** et **master** sont protégées.
- Pour cela tu dois aller dans l'onglet "Pull request" sur Github et cliquer sur le bouton vert "New pull request"

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)

Label issues and pull requests for new contributors

Dismiss

Now, GitHub will help potential first-time contributors discover issues labeled with **help wanted** or **good first issue**

Filters

Labels 8

Milestones 0

New pull request

0 Open ✓ 6 Closed

Author Projects Labels Milestones Reviews Assignee Sort







Pull Request (PR)


- Tu dois ensuite choisir quelle branche merger dans quelle autre, ici **basic-3** dans **master** (**attention au sens de la flèche**)
- Puis créer la PR grâce au bouton "Create pull request".

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)


Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).


 base: **master** 


 compare: **basic-3**


✓ **Able to merge.** These branches can be automatically merged.


 **Create pull request**


Discuss and review the changes in this comparison with others.






 **1** commit

 **1** file changed

 **0** commit comments

 **1** contributor

 Commits on Mar 26, 2019

  **open-it**

[Add readme](#)

0e8cfe7




Pull Request (PR)

[Code](#) [Issues 0](#) [Pull requests 0](#) [Projects 0](#) [Wiki](#) [Insights](#) [Settings](#)

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#).

base: **master** compare: **basic-3** ✓ Able to merge. These branches can be automatically merged.



Add readme

Write

Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Styling with Markdown is supported

Create pull request

Reviewers

No reviews

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

1 commit

1 file changed

0 commit comments

1 contributor

Commits on Mar 26, 2019

open-it

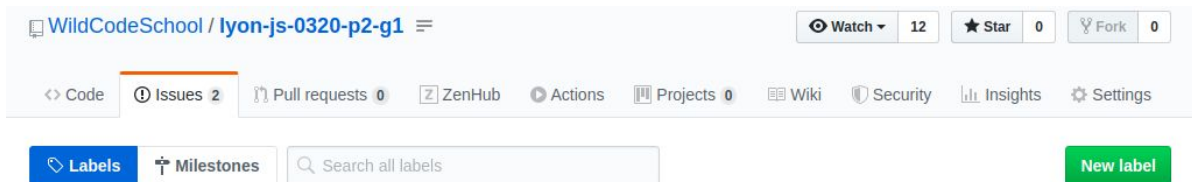
Add readme

0e8cfe7

Tu peux ajouter le **code reviewer**, ajouter des **labels**, mettre un **commentaire** explicite, etc.



Les labels

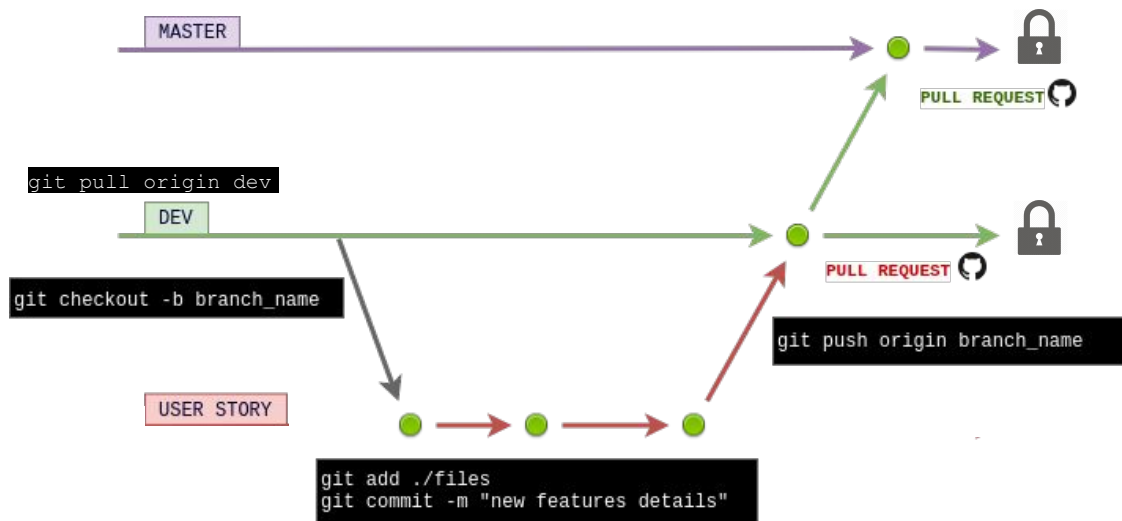


Tu utiliseras des “labels” (dans l’écran de la PR) pour communiquer avec ton équipe et les formateurs sur l’état d’avancement de la fonctionnalité.

- **WIP (#4FC2F7)** : Quand tu as commencé à travailler sur la fonctionnalité
- **Ready for validation (#01FFFF)** Quand la feature est prête pour être validée par le CR.
- **Ready for prod (#4987E8)** : Quand tout est prêt pour **sereinement** intégrer le travail sur dev.
- **Changes requested (#FF9600)** : La PR est refusée par le CR ou le formateur
- **Conflicts (#FF0000)** : La PR ne peut pas être *mergée*. A enlever quand vous avez mergé dev sur la branche feature
- **Fixed - ready for prod (#1F771A)** : Il y a eu un “Changes requested” et les corrections sont faites.
- **Feedback needed (#FF00FF)** : Vous voulez avoir des avis de toute l’équipe sur une question que vous vous posez au moment de l’implémentation
- **PO Feedback needed (#9F00FF)** : Vous souhaitez avoir l’avis du PO sur une question en particulier ou une suggestion de sa part



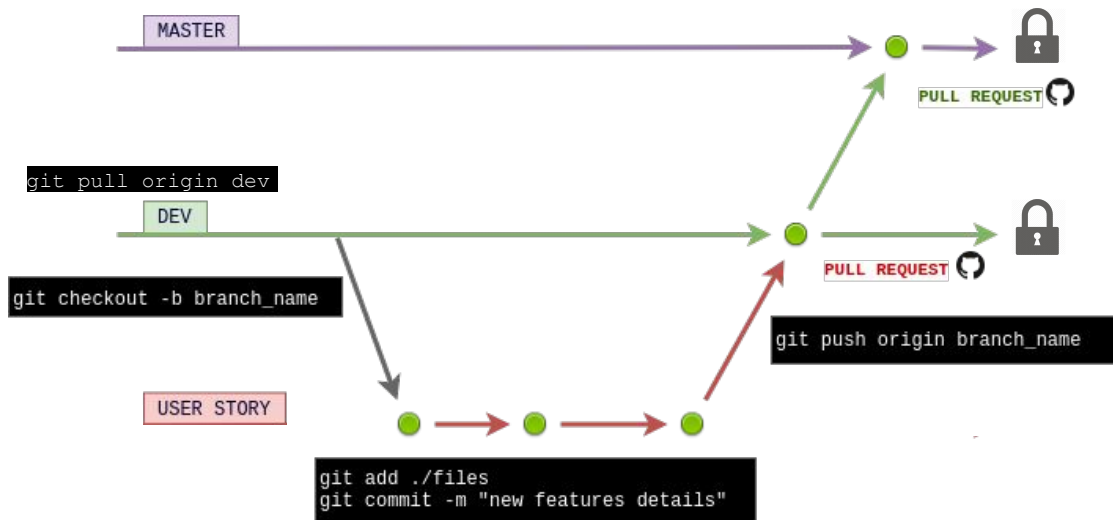
Le workflow



- Les branches **master/main** et **dev** sont protégées !
- Tire une branche à partir d'une référence (**dev**). Cette branche **DOIT** être le conteneur d'une **USER STORY** (et surtout pas d'une personne !)
- Ouvre directement une PR de <nom_branche> vers **dev** et ajoute le label **WIP**
- Sur cette branche, vous (la team projet) allez faire plusieurs *commits* (atomiques).
def : "Indivisible, en parlant de données ou d'opérations. Une opération atomique est garantie soit de s'effectuer intégralement, sans être interrompue, soit de ne pas s'effectuer du tout."



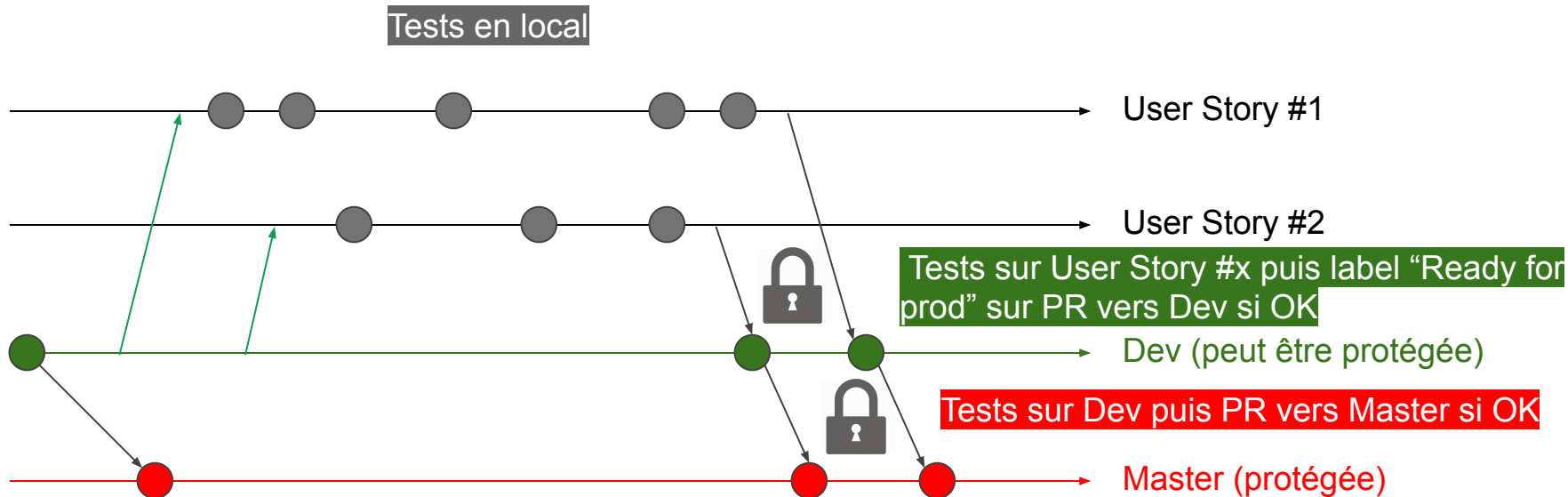
Le workflow



- Une fois ta **US** terminée et fonctionnelle, mets le label **Ready for validation** sur ta PR pour que ta branche soit vérifiée et ré-intègre la branche **dev**. Cette PR est examinée par tes collègues, qui mettront le label **Ready for prod** si tout est ok. Si les bonnes pratiques sont respectées et que tout semble correct, ton formateur accepte la PR et *merge* ta branche sur **dev**.
- En fin de semaine, l'équipe fait une PR de **dev** vers **master/main** et revoit avec le formateur l'ensemble du code de la semaine.
- Le formateur pourra éventuellement donner la liberté au groupe de merger directement sur **dev** sans passer par sa validation systématique.



Le workflow





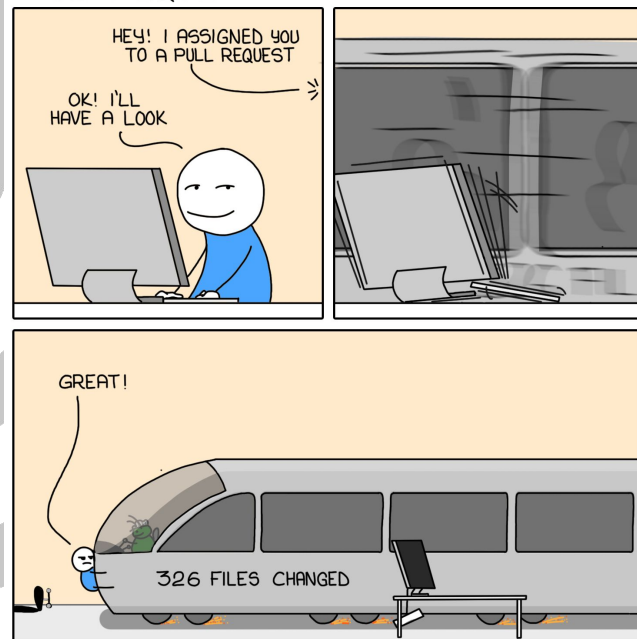
Pull Request (PR)

Une fois la PR créée **et les bons labels en place**, si elle n'est pas acceptée par le formateur ou tes collègues, tu dois **corriger** ce qui ne va pas dans ta branche et **refaire un commit** puis un **push** sur cette même branche, puis indiquer que la branche doit de nouveau être recettée

Pas la peine de refaire une nouvelle PR car **la précédente est toujours d'actualité**.

Une PR est associé **à une branche** et **non à un commit** précis.

PULL REQUEST





Issues - Astuces

- Toutes les *issues* sont référencées par un **identifiant** numérique unique.
- Lorsque tu "*commit*" une correction d'un bug, tu peux utiliser le numéro de l'issue dans ton **message de commit** de manière à lier le commit et l'issue.
- Il existe plusieurs mots clés qui te permettent de faire des actions automatique lors des commits (push) (Close, Resolve...)
- ex. : "*This closes #34, closes #23, and closes example_user/example_repo#42*" ferme les issues #34 et #23 du même repository, et l'issue #42 dans le repository "example_user/example_repo".

This repository Search Pull requests Issues Marketplace Explore

open-it / pendujava Watch 0 Star 0 Fork 0

<> Code Issues 1 Pull requests 1 Projects 0 Wiki Insights Settings

un bug sur le pendu #1

Open open-it opened this issue on Jul 6 · 0 comments

open-it commented on Jul 6

Owner + @

il se passe ceci ou cela.

open-it added a commit that referenced this issue on Jul 6

#1 correction bug 8b68525

open-it self-assigned this on Jul 6

Assignees open-it

Labels None yet

Projects None yet

Milestone No milestone

Notifications Unsubscribe You're receiving notifications because you were assigned.

1 participant

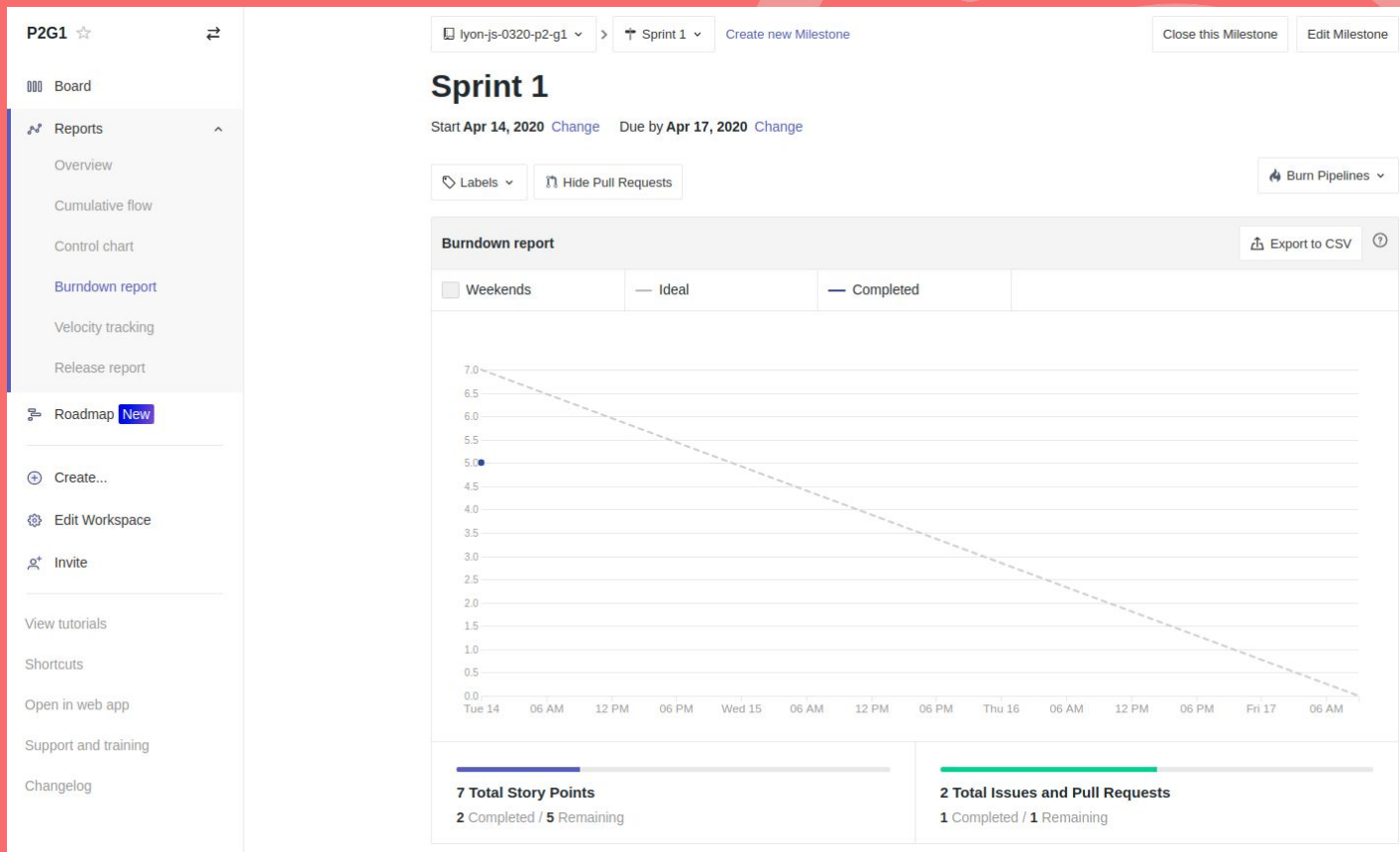
Lock conversation

```
$ git commit -m "#1 correction bug"
```

<https://help.github.com/articles/closing-issues-using-keywords/>



Suivre votre avancée



Des questions ?

**GIT PULL
MA POULE**

