



Présentation du logiciel GIT avec les commandes de base & les règles pratiques du projet EZ.

Système de branche



compilation

architecture

organisation

langage

diffusion

master

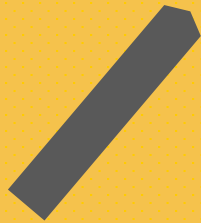
*

Qu'est ce qu'une branche ?

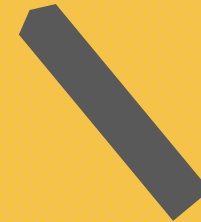
Une branche est une version parallèle d'un même projet qui permet de travailler sur le projet sans impacter la branche principale « master » qui représente une version stable du projet.

* Règle de nommage des branches

diff_frontPageContact



4 premiers caractères



nom de la tâche

* Règle de nommage des branches de bug

bug/diff_#2014



Type BUG

4 premiers caractères

Numéro du bug

* Qu'est qu'un PULL REQUEST ?

“

Une fois que l'on a travaillé sur notre branche on souhaite souvent pusher nos modifications avec la branche principale. On fera alors un **pull request** qui consiste tout simplement à demander à l'administrateur de **merger** nos modifications.

1

Pousser (push) vos modifications

2

Aller sur la page Github du projet

<https://github.com/ShihoWasTaken/ezlanguage/pulls>

3

Aller sur l'onglet "Pull request" et cliquez sur le bouton "New pull request"



Règles pour valider un pull request

Le code est commenté

Le code est accompagné des tests qui lui sont liés

Le code est propre

Il n'y a pas de conflit (entre les branches) dans le pull request

* Comment faire un PULL REQUEST ?

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: compilation ▾

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



Create pull request

Discuss and review the changes in this comparison with others.



🔑 2 commits

📄 1 file changed


💬 0 commit comments

👤 1 contributor



Commits on Sep 17, 2016





 **adrien3**

Update README.md

cb9a79a

* Comment faire un PULL REQUEST ?

 base: master ... compare: compilation ✓ Able to merge. These branches can be automatically merged.

 Compilation


Write

Preview

AA ▾ B i “ <> 🔗 ☰ ☷ ✓ ↶ @ 📌

Leave a comment

Attach files by dragging & dropping or [selecting them](#).

 Styling with Markdown is supported

Labels

None yet

Milestone

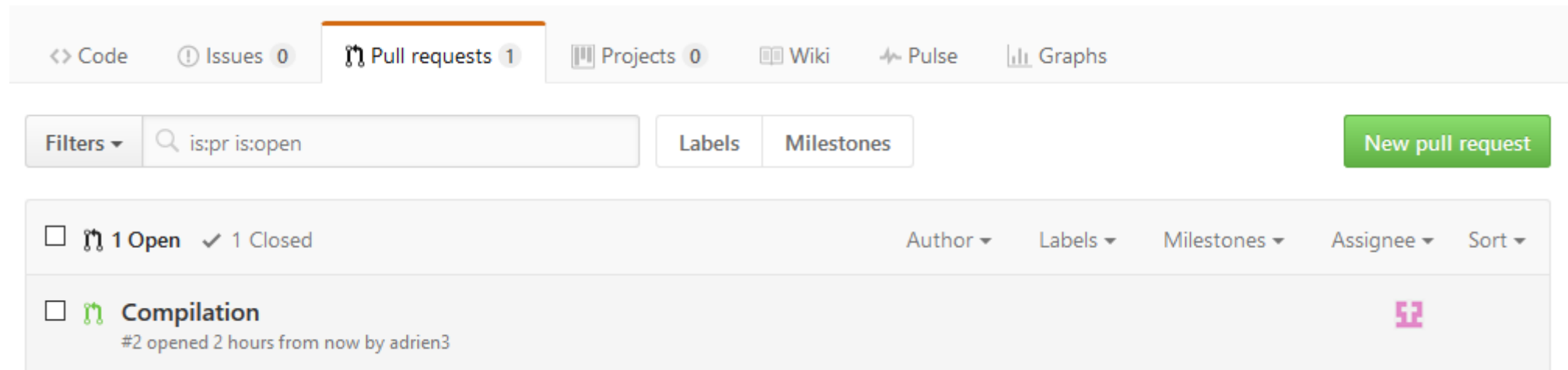
No milestone

Assignees






No one—assign yourself


Create pull request


* Comment faire un PULL REQUEST ?





The screenshot shows the GitHub interface for pull requests. At the top, there is a navigation bar with tabs for Code, Issues (0), Pull requests (1), Projects (0), Wiki, Pulse, and Graphs. Below this is a search bar with the filter 'is:pr is:open' and buttons for Labels and Milestones. A green button labeled 'New pull request' is on the right. The main content area shows a list of pull requests. The first entry is 'Compilation' by 'adrien3', opened 2 hours ago. It has a green icon and a purple icon. The interface is in French.

<> Code ! Issues 0  Pull requests 1  Projects 0  Wiki  Pulse  Graphs

Filters ▾  is:pr is:open Labels Milestones [New pull request](#)

☐  1 Open ✓ 1 Closed Author ▾ Labels ▾ Milestones ▾ Assignee ▾ Sort ▾

☐  **Compilation** 
#2 opened 2 hours from now by adrien3



LES COMMANDES DE BASE DE GIT

* Initialisation



```
git clone https://github.com/ShihoWasTaken/ezlanguage.git
```

* Initialisation

Définition du nom de l'utilisateur

“

```
git config user.name 'nom'
```

Définition du mail de l'utilisateur

“

```
git config user.email 'mail'
```

* Commande GIT

Vous donne des informations sur l'état de votre branche en local

“

git status

Vous donne la branche sur laquelle vous vous trouvez

“

git branch

Changer de branche

“

git checkout nom_de_la_branche

* Commande pour les commits GIT

Ajoute un fichier dans votre futur commit

“

```
git add 'file_name'
```

Ajoute tous les fichiers modifiés dans votre futur commit

“

```
git add .
```

Création d'un commit

“

```
git commit -m 'message du commit'
```

* Commande PULL / PUSH

PULL => on récupère les sources depuis GIT

“

`git pull`

PUSH => on va pousser nos commits sur GIT

“

`git push origin ma_branche`

* Exemple

```
//On va se placer sur la branche
git checkout diff_developpementFront
//On va récupérer les sources
git pull origin diff_developpementFront
//On va travailler puis enregistrer notre travail
git add .
//On va faire un commit
git commit -m « mon message »
//On va pull pour voir s'il y a des conflits
git pull origin diff_developpementFront
//S'il y a des conflits on va les corriger sinon on peut push
git push origin diff_developpementFront
```


* Créer une branche

//On va se placer sur la branche où l'on souhaite récupérer les dernières sources

git checkout diffusion

//On va récupérer les sources

git pull origin diffusion

//On va ensuite créer une branche et se placer dessus

git checkout -b ma_nouvelle_branche

//On va faire un commit

git push origin ma_nouvelle_branche

* Correction des conflits

Si votre console vous indique qu'il y a un conflit lors d'un pull, il faut alors corriger les erreurs de conflit. Pour afficher les fichiers en conflit utiliser la commande ci-dessus

“

git status

Il suffit d'ouvrir ensuite le fichier en conflit et d'apporter les corrections. Les conflits sont identifiés par différentes balises :

```
<<<<<<<<< HEAD
// CODE LOCAL
=====
// CODE SUR LE REPO
>>>>>>>>>>> commit_nbr
```

* Correction des conflits

Après avoir appliqué les modifications, supprimer les Tags (<<<HEAD,====,>>>>>>) et vérifier que le programme compile toujours.

Si le programme compile toujours alors vous pouvez enregistrer la correction de vos fichiers.

Pour cela il suffit de rajouter tous les fichiers corrigés avec la commande

“

`git add “nom du fichier”`

Lorsque tous les conflits sont résolus, faire un commit (et/ou pousser vos commit)

“

`git commit -m “your_message”`
`git push origin ma_branche`

* Merge entre deux branches

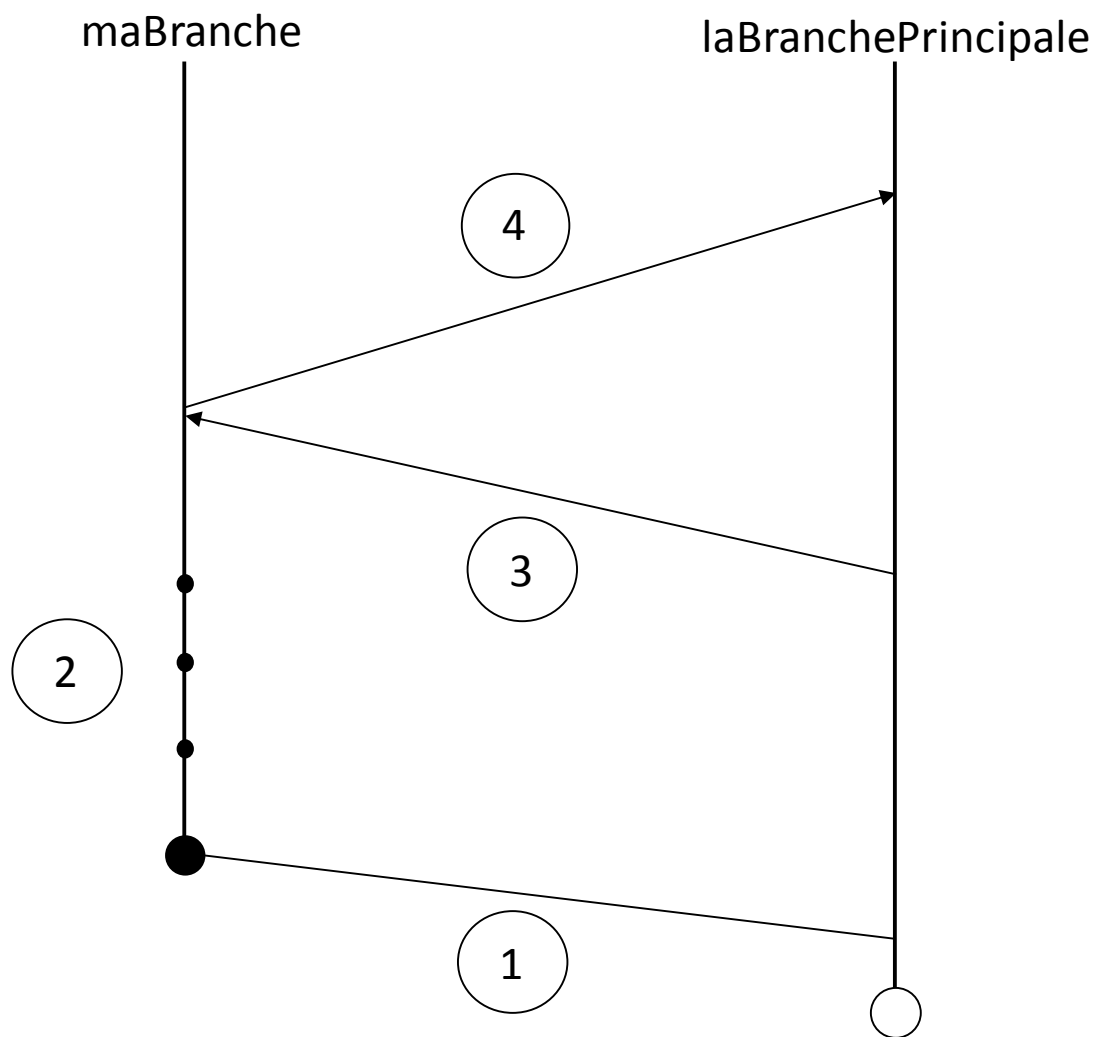
Un merge permet de réaliser une **fusion** entre deux branches. Cela répond au souhait de faire avancer la branche courante de sorte qu'elle incorpore le travail d'une autre branche.

“

```
git merge --no-ff <branche>
```

Fusionne la branche <branche> avec la branche courante en générant un commit de fusion

* Exemple de commande pour tout faire



* Etape 1 : création de la branche

```
git checkout -b mabranche
```

```
git push -u origin mabranche
```

* Etape 2 : création d'un commit

```
git add .
```

```
git commit -m « Message »
```

```
git push
```

* Etape 3 : Mise a jour de la branche local

```
git checkout branch_principale /git pull
```

```
git checkout mabranche
```

```
git rebase branche_principale /git push -f
```