

# Git - exercice: Code Ping Pong

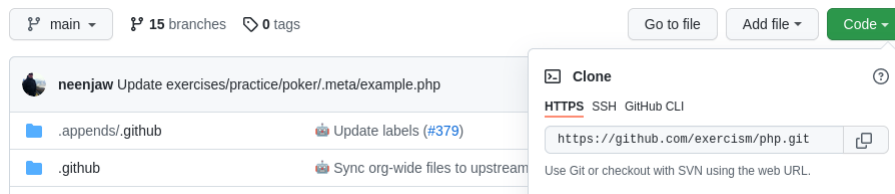
En binôme, vous allez mettre en place un système d'intégration continue et collaborer sur un dépôt Git partagé (sur GitHub).

Le code sur lequel vous allez travailler s'appuie sur les exercices proposés par le site [exercism.org](https://exercism.org)

## Mise en place du dépôt local

Nous allons nous servir de Git pour copier un sous-dossier d'un dépôt distant. Ce ne sera pas ce projet que nous versionnerons.

1. Créer un dossier sur l'un des ordinateurs de votre binôme
2. Initialiser un dépôt Git vide avec la commande `git init`
3. Choisir le langage dans lequel réaliser les exercices:
  - JavaScript (nécessite une version de Node.js): ouvrir dans un navigateur la page [github.com/exercism/javascript](https://github.com/exercism/javascript)
  - PHP (nécessite composer installé): ouvrir dans le navigateur la page [github.com/exercism/php](https://github.com/exercism/php)
4. Ajouter le dossier choisi dans la liste des dépôts distants (par exemple: `git remote add origin https://github.com/exercism/php.git`). On trouve l'adresse URL du dépôt en cliquant sur *Code* sur la page GitHub.

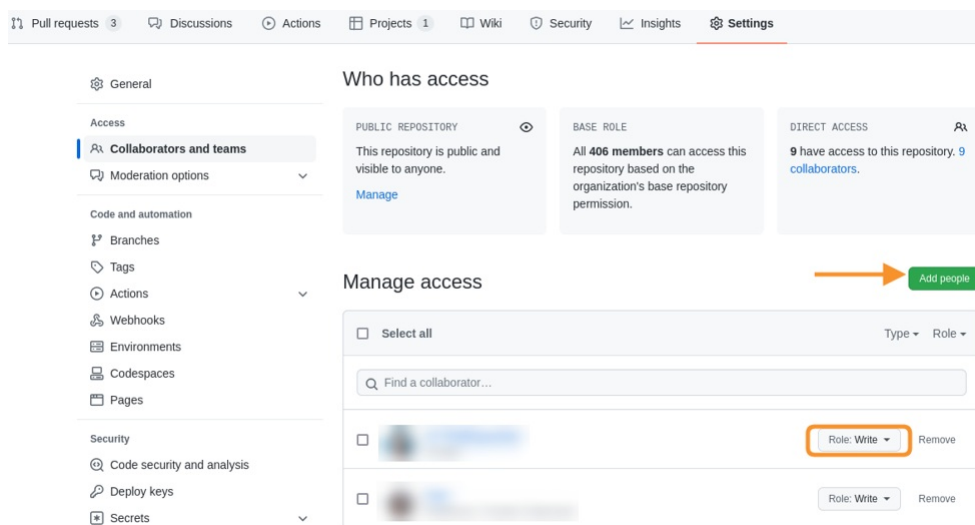


5. Choisir un des exercices dans le sous-dossier `exercises/practice` (par exemple le premier, "accumulate").
6. Cloner uniquement le dossier choisi, et uniquement le dernier commit:
  1. Ajouter le chemin du sous-dossier à cloner avec la commande: `git sparse-checkout exercises/practice/accumulate`
  2. Vérifier que le dossier est bien configuré avec la commande `git sparse-checkout list` (vous devez voir apparaître le chemin donné à la commande précédente)
  3. Récupérer le dernier commit du dépôt distant avec la commande `git pull origin main --depth=1` ("--depth=X" signifie "seulement les X derniers" commits).

## Mise en place du dépôt partagé

Nous allons créer le dépôt Git que nous partagerons avec notre binôme.

7. Se déplacer à l'intérieur du nouveau dossier (exemple avec la commande: `cd exercises/practice/accumulate`).
8. Créer un nouveau dépôt Git à ce niveau avec la commande `git init`, c'est uniquement ce dossier que nous partagerons pour cet exercice.
9. Installer les dépendances du projet (par exemple, avec la commande `npm install` pour JavaScript), un nouveau dossier va être créé (par exemple `node_modules` pour JavaScript). \*
10. Interdire à Git de versionner ce dossier:
  1. Créer un fichier nommé **exactement** `.gitignore` (par exemple, avec la commande `touch .gitignore`)
  2. Dans ce fichier, écrire le nom du dossier contenant les dépendances du projet. Par exemple avec la commande `echo node_modules/ >> .gitignore`
  3. Vérifier que Git ignore effectivement le dossier contenant les dépendances avec la commande `git status`, le nom du dossier ne devrait pas apparaître/
11. Ajouter les fichiers à l'index (`git add .`) puis enregistrer les modifications dans l'historique (`git commit -m "Mise en place du dépôt partagé"`).
12. Sur [github.com](https://github.com), créer un nouveau dépôt (bouton "+" > *New repository*), choisir un nom facile à se rappeler, par exemple `javascript-accumulate`, si vous faites l'exercice "accumulate" en JavaScript.
13. Copier l'adresse URL (ou SSH si configuré) et l'ajouter comme dépôt distant, par exemple: `git remote add origin https://github.com/johnny-b-goode/javascript-accumulate.git`
14. Pousser la branche locale sur le dépôt distant, par exemple avec la commande `git push origin master`.
15. Sur la page du dépôt GitHub, ajouter le binôme en tant que collaborateur du projet: *Settings* > *Access* > *Collaborators* > *Add people* et chercher son pseudonyme.



16. Dans la liste des collaborateurs, donner le rôle `Write` ou `Admin` au binôme ajouté.

17. Sur l'ordinateur du binôme, cloner le dépôt partagé, par exemple avec la commande `git clone https://github.com/johnny-b-goode/javascript-accumulate.git`, dans le dossier créé, installer les dépendances qui n'ont pas été versionnées (par exemple: `npm install` pour Javascript).

\*: Pour PHP, vous devrez créer le fichier `composer.json` avant d'installer les dépendances avec `composer install`. Vous trouverez un exemple de fichier `composer.json` à [l'adresse suivante](#).

## Mise en place de l'intégration continue

Maintenant que les deux collaborateurs ont accès au dépôt partagé, nous allons créer, sur celui-ci, un "workflow" d'intégration continue grâce au service GitHub Actions.

18. Sur la page du dépôt partagé, aller dans l'onglet *Actions*, défiler jusqu'à trouver la rubrique *Continuous Integration* et cliquer sur *View all*.

19. Trouver une configuration qui correspond le mieux au projet (par exemple *Node.js* pour Javascript) et cliquer sur *Configure*

20. Nous nous retrouvons devant un fichier Yaml éditable, vérifier qu'il contient bien:

```
on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]
```

Ce sont les conditions pour déclencher le workflow d'intégration continue: il sera déclenché à chaque fois qu'un collaborateur fera un `push` sur la branche `master` ou une Pull Request sur la branche `master`.

21. Dans `strategy` > `matrix` > `node-version` (ou `php-version`), ne garder qu'une seule version, exemple: `[ 14.x ]`, cela n'exécutera la suite de test qu'une seule fois par push/pull request (au lieu d'une fois par version de Node / PHP).

22. Dans `steps`, vérifier qu'il y a bien une étape: `- run: npm test *`

23. Si tout est valide, cliquer sur *Start commit*, donner un titre explicite (exemple: "mise en place de l'intégration continue") et cliquer sur *Commit new file*.

24. Retourner sur l'onglet *Actions*, normalement il y devrait y avoir un "workflow run", car la configuration de l'intégration continue a ajouté un commit sur `master` et celui-ci devrait être un échec. Cliquer dessus, vous verrez un journal des opérations s'étant exécutées, les tests automatisés doivent actuellement échouer, car nous n'avons pas ajouter de code à l'exercice téléchargé! Nous allons remédier à cela dans l'étape suivante.

25. Sur chaque ordinateur du binôme, récupérer les modifications effectuées avec la commande `git fetch origin master:master` (ou `git pull`).

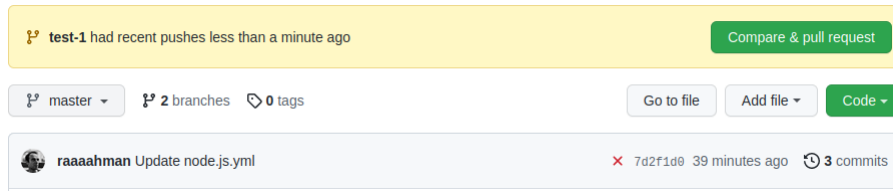
\*: Pour PHP il faudra décommenter les lignes `- name: Run test suite` et `run: composer run test` (les commentaires en Yaml sont les lignes commençant par un "#").

## Processus de travail

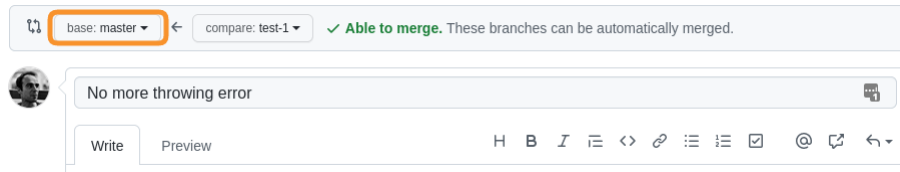
Maintenant que l'intégration continue est en place, nous allons pouvoir commencer à travailler et envoyer les modifications au fur et à mesure.

- Dans le dossier de l'exercice, il y a un sous-dossier `.doc` contenant un fichier `instructions.md`, ce sont les consignes de l'exercice. Elles sont en anglais, vous pouvez les traduire si cela vous est plus simple (par exemple avec [DeepL](#)).
- Sur l'un des deux ordinateurs du binôme, créer une nouvelle branche, par exemple `git branch test-1` et se positionner dessus (`git checkout test-1`).
- Dans le dossier de l'exercice, il y a un fichier qui contient les tests, par exemple `accumulate.spec.js` pour l'exercice "accumulate" en JavaScript. Chaque test commence par `test` s'il est actif, ou `xtest` s'il n'est pas actif.
  - Si le dernier test a échoué (l'intégration continue rapporte une erreur identifiée par une croix rouge), laisser tel quel.

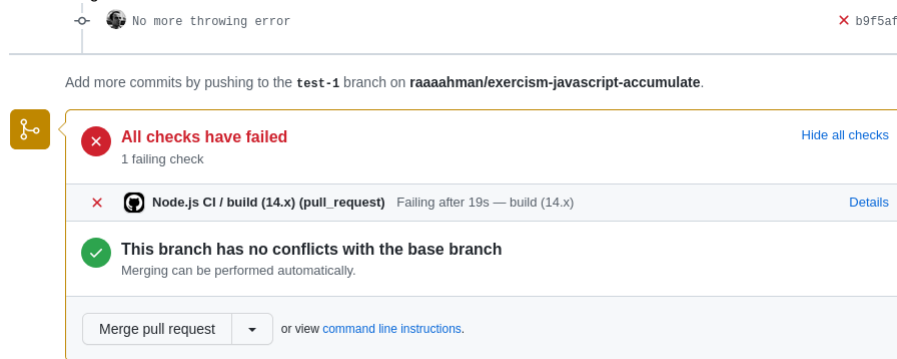
- Si tous les tests "passent" (l'intégration continue ne rapporte pas d'erreur, mentionné par une coche verte), activer le prochain test (en écrivant `test` à la place de `xtest`)
4. Ecrire le code qui permettrait de résoudre le test actuel dans le fichier `accumulate.js`, l'enregistrer dans un commit et envoyer votre modification sur le dépôt partagé (exemple `git push origin test-1`).
  5. Sur la page du dépôt partagé, basculer sur la branche `test-1` et cliquer sur *Compare and pull request*



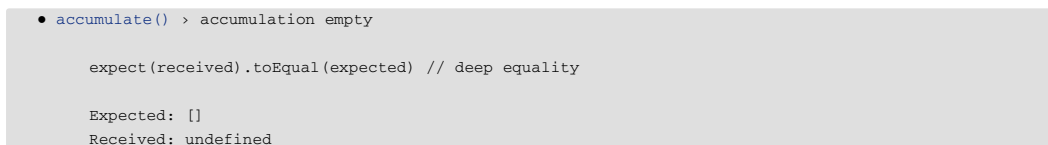
6. Dans la nouvelle fenêtre, vérifier que la pull request cible bien la branche `master` et cliquer sur *Create Pull Request*



7. Le workflow d'intégration continue devrait s'exécuter et afficher un résultat:

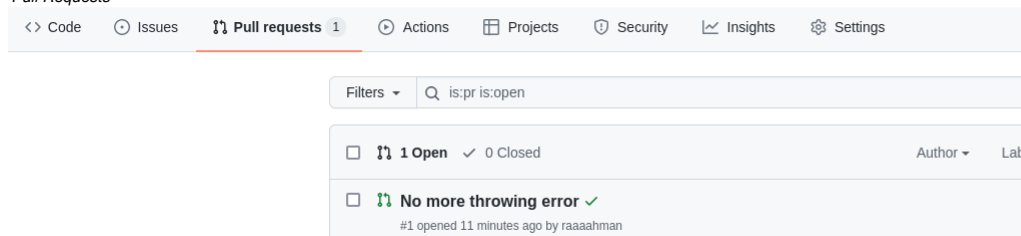


- Si les tests ont échoués (croix rouge), cliquer sur *details* pour comprendre ce que le code aurait du produire et ce qu'il s'est réellement produit, par exemple:




Dans cet exemple, la fonction aurait du retourner un tableau vide, mais elle n'a pas retourné de valeur du tout. Essayez de corriger cela dans votre dépôt local, et renvoyer les modifications sur le dépôt distant (étape 4). Pas besoin d'ouvrir une nouvelle Pull Request, les commits vont s'accumuler dans la Pull Request courante.

- Si les tests sont passés (coche verte), demander à votre binôme de valider en ouvrant la page du dépôt partagé et en se rendant dans l'onglet *Pull Requests*



8. Cliquer sur la Pull Request en cours, et cliquer sur *Merge pull request*

Add more commits by pushing to the **test-1** branch on **raaaahman/exercism-javascript-accumulate**.



✓ **All checks have passed**  
1 successful check [Show all checks](#)

✓ **This branch has no conflicts with the base branch**  
Merging can be performed automatically.


Merge pull request

 or view [command line instructions](#).

puis *Confirm*

*merge*

9. Une fois que le message de validation apparaît, vous pouvez cliquer sur *Delete branch*



**Pull request successfully merged and closed**

You're all set—the **test-1** branch can be safely deleted.

Delete branch

10. A votre binôme d'écrire le code, il doit mettre à jour sa branche **master** (`git fetch origin master:master`) et recommencer à l'étape 2 (en changeant le nom de la branche), puis passer à l'étape 3 en activant un nouveau test. Tant que l'exercice n'est pas résolu, continuez à alterner la personne qui code et la personne qui valide la Pull Request, en résolvant un test à la fois, et en changeant de rôle entre chaque nouveau test.

\*: Pour PHP il faudra à l'inverse désactiver tous les tests sauf le premier, en écrivant la ligne `$this->markTestSkipped()` pour chaque méthode commençant par `test...`.