# **Atelier Git**

*Premier conflit git*

Travaillez en binôme. Il y aura le développeur A, qu’on va appeler Kirby. Et le développeur B, qu’on appellera Link.

|  |  |
| --- | --- |
| **Actions de Kirby** | **Actions de Link** |
| Créer un dépôt github “wild-project” | Regarder ce que fait Kirby |
| Faire les commandes “…or create a new repository on the command line” affichées sur Github depuis un dossier “wild-project” en local (mkdir wild-projet depuis ~/WildProjects par exemple) | Link explique à Kirby ce que chaque commande dans “…or create a new repository on the command line” fait |
| Regarder ce que fait Link | **Faire un fork du dépôt créé par Kirby** |
| git statsuAider Link à cloner le dépôt Fork et comprendre ensemble comment faire | Cloner le fork nouvellement créé en renommant le projet en local “wild-link” Ma commentaire (attention, bien utiliser git clone) |
| Ouvrir le projet dans l’éditeur de code | |
| Modifier le readme en y mettant :  # wild-project  Comment installer ce projet ?  Il suffit, pour le moment, de le cloner. | Modifier le readme en y mettant :  # wild-project  Comment installer ce projet ?git |
| Créer le fichier “same\_file.js”  Y mettre un commentaire dedans | Créer le fichier “same\_file.js”  Y mettre un commentaire, pas le même que Kirby |
| Créer le fichier “another\_file.js”  Y mettre un commentaire | Créer le fichier “not\_same\_file.js”  Y mettre un commentaire |
| Regarder ce que fait Link, comprendre avec chaque commande et leur résultat. | Créer le fichier “secret\_file.js”  Contenant : // Mustn't be shared on github |
| Exécuter : git status |
| Ajouter le fichier secret\_file.js à gitignore (créer le fichier .gitignore) |
| Exécuter : git status | |
| git add . | |
| git status | |
| git commit -m “ “ (mettre un message explicite de ce que vous avez fait, pas de limites, vous pouvez appuyer sur entrée tant que vous n’avez pas fermé les guillemets) | |
| Mettre en ligne le commit (utiliser git push) | |
| Vérifier la mise à jour sur github - le fichier secret.js, est-il là ? | |
| Regarder ce que fait Link, comprendre avec lui | Créer une pull request depuis github |
| Vérifier que “can’t automatically merge” s’affiche |
| Ajouter la description : Explain here what you modified and how to test it. |
| Accéder à la pull request sur le lien Github | Regarder ce que fait Kirby, comprendre avec lui |
| Dans “This branch has conflicts that must be resolved”, cliquer sur “command line” |
| Suivre les instructions (après avoir cliqué sur le bouton “Git”) du STEP 1. **Ne pas faire STEP 2 pour le moment.** | Expliquer les 2 commandes à Kirby |
| Comprendre la sortie de la dernière commande du STEP 1 dans le terminal | |
| git status |  |
| Comprendre ensemble le résultat de git status | |
| **Pause, attendre l’explication du formateur à l’ensemble de la classe.**  **Aider ceux qui n’y sont pas encore.** | |
| Résoudre les conflits depuis l’éditeur de texte | Regarder ce que fait Kirby, l’aider (le deuxième dév ne sera pas toujours là pour résoudre les conflits, mais s’il l’est, on en profite !) |
| Sauvegarder les fichiers | Continuer d’aider Kirby et de comprendre ensemble ce que vous faites et ce qu’il se passe. |
| git status |
| git add |
| git commit |
| **Suivre step 2 : )** |
| Ici, moins de détails volontairement, aider Link et comprendre avec lui | Ajouter le dépôt github de Kirby en tant qu’upsteam : <https://help.github.com/articles/configuring-a-remote-for-a-fork/> |
| Récupérer le travail de Kirby en local |
| Le mettre en ligne sur son propre github |