*Ce document est une documentation visant à permettre à un nouveau développeur de configurer son environnement de développement afin de rejoindre une équipe travaillant sur un* ***projet symfony*** *entreposé sur un repo* ***github****. Les instructions suivantes prévoient que le développeur en question n’ait encore jamais travaillé avec les outils cités et reprend donc l’intégralité des étapes de configuration.*

**Téléchargement et configuration des outils nécessaires :**

1. **Git - Github**

• Installer **Git Bash** [ici](https://git-scm.com/downloads) afin d’avoir accès à l’ensemble des commandes de contrôle de versions proposées par ce dernier.

• Créer et configurer son compte **Github** [ici](https://github.com/signup?ref_cta=Sign+up&ref_loc=header+logged+out&ref_page=%2F&source=header-home).

• Fournissez ensuite à **l’administrateur** du repo Gihub votre **Identifiant de compte Github** afin que ce dernier vous donne un accès au repo si ce dernier est privé.

1. **Symfony - Composer**

• Installer la dernière **version de PHP** [ici](https://www.php.net/downloads) (symfony est un framework PHP)

• Installer le gestionnaire de paquet symfony (appelé **composer**) [ici](https://getcomposer.org/download/), permettant l’installation en CLI des différents outils qui seront utilisés en phase de développement.

• Il est également recommandé d’installer les paquets **profiler** qui offrent une interface de débogage via la commande ci-dessous en CLI:

**composer require –-dev symfony/profiler-pack**

1. **Récupération du repo en local**

• Afin de récupérer le repo et commencer à collaborer avec d’autres développeurs, **copiez** l’adresse web du repo distant du projet (dans notre cas, <https://github.com/Rakbaal/projet-Baranger-LCA-AFO>) .

• Lancez git bash, et créez le répertoire dans lequel vous entreposerez les fichiers du repo distant via la commande suivante

* **mkdir [nom du répertoire]**

• Toujours dans git bash, déplacez vous jusque dans le répertoire créé via

* **cd [nom du répertoire]**

• Initialisez le répertoire via la commande suivante

* **git init**

**•** Il nereste désormais plus qu’à descendre les fichiers du repo distant :

* **git pull [lien du repo github copié]**

Vous devriez désormais être en mesure de collaborer avec vos collègues sur le projet en cours de développement.

**EDIT :** Bien que les étapes précédentes soient effectivement valides, il est également possible de lier votre repo local à un repo distant via « remote » pour ne pas avoir à taper l’adresse du repo distant à chaque pull/push.  
Pour cela, il suffit de vous rendre dans le repo local que vous souhaitez lier et de taper :

**git remote add [nom à donner au remote] [adresse du repo distant]**Ainsi vos pulls et push pourront se faire via la commande **git pull/push [nom remote]** sans vous soucier de l’adresse du repo.

**Workflow en collaboration sur Github :**

Dans un premier temps, le travail collaboratif nécessite la création de **branches** afin de ne pas empiéter sur les avancées des collaborateurs.

1. **S’assurer de commencer avec une version à jour du projet**

Lorsque vous êtes sur le point de commencer à travailler sur une nouvelle fonctionnalité, il est important de récupérer l’état actuel de la branche principale via la commande **git pull [nom remote] master**. Ainsi, si une nouvelle fonctionnalité a été merge\* à la branche principale, vous l’aurez également en local ce qui vous permettra de commencer à travailler sur une version à jour du projet.

\*merge signifie « fusionner », le terme sera explicité plus tard.

1. **Création de branche et développement d’une fonctionnalité**

La création d’une branche s’effectue par le biais de la commande **git branch [nom de la branche créée].**

La branche créée est un **clone** de la branche master, mais les modifications faites à votre branche n’existeront qu’au sein d’elle-même, sans avoir de répercussions sur les autres.

Au sein de cette branche, il est important de commit dès qu’une modification a été apportée (ex : création d’un fichier, modification d’un code, correction d’un bug…).

Il ne s’agit pas de commit à la moindre ligne de code écrite, mais de s’assurer que vous serez capable de revenir en arrière à une version stable assez récente pour ne pas perdre trop de temps.  
Il est également important de **push** régulièrement votre branche sur le repo distant afin que vos collaborateurs voient l’avancement de votre travail et puissent pointer du doigt d’éventuelles modifications à effectuer.  
Pour push une branche particulière, la commande est **git push [nom remote] [nom branche à push].**

1. **Validation d’une fonctionnalité**

Une fois que votre fonctionnalité est finalisée sur votre branche, vous pouvez procéder à une **pull request** sur github. Lorsque cette dernière aura été validée par l’ensemble de l’équipe de développement, la fonctionnalité pourra alors être intégrée à la branche principale du projet.