

* **git fetch** is the command that says "bring my local copy of the remote repository up to date."
* **git pull** says "bring the changes in the remote repository where I keep my own code."

Normally "**git pull**" does this by doing a "**git fetch**" to bring the local copy of the remote repository up to date, and then merging the changes into your own code repository and possibly your working copy.

In the simplest terms, git pull does a git fetch followed by a git merge.

|  |  |
| --- | --- |
|  |  |
| The take away is to keep in mind that there are often at least **three copies** of a project on your workstation. One copy is your own repository with your own commit history. The second copy is your working copy where you are editing and building. The third copy is your local "cached" copy of a remote repository. |  |

$ git config --global user.name "AmineLouati"

$ git config --global user.email louati.amiine@gmail.com

Il faut vérifier 2 choses :

1. Le nombre de commit est le même
2. L’identifiant du dernier commit est le même

Mettre à jour le repo local d’un remote fork par rapport au remote origin

0. se positionner sous le dossier du projet

1. git remote add upstream git://github.com/ow2-proactive/scheduling.git

2.1 git fetch upstream // Now you can collect the latest changes of the upstream repository with fetch

*2.2 git checkout master*

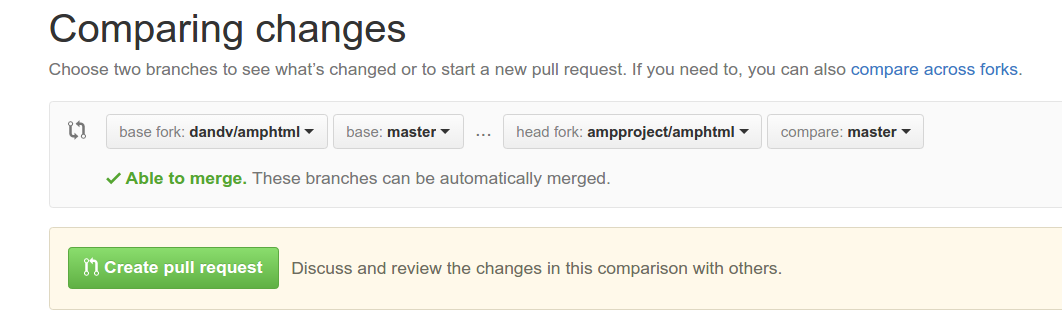
*2.3 git merge upstream/master*

3. git pull upstream master

4. git push origin master

5. ./gradlew build -x test  // faire le build sans les test après avoir récupéré une version de l’origine

Another method

1. Open your fork on GitHub.
2. Click on Pull Requests.
3. Click on New Pull Request. By default, GitHub will compare the original with your fork, and there shouldn't be anything to compare if you didn't make any changes.
4. Click switching the base if you see that link. Otherwise, manually set the base fork drop down to your fork, and the head fork to the upstream. Now GitHub will compare your fork with the original, and you should see all the latest changes. [](https://i.stack.imgur.com/FoNQu.png)
5. Create pull request and assign a predictable name to your pull request (e.g., Update from original).
6. Scroll down to Merge pull request, but don't click anything yet.

Créer un fichier gradle.properties dans le dossier caché .gradle du home (cd home) et ajouter la ligne suivante :

org.gradle.daemon=true 🡪 le build devient plus rapide

gradle gère le cycle de vie de tout le projet y compris la compilation, les tests …

La creation d’un nouveau sous module 🡪 l’ajouter dans le fichier settings.gradle

les builds se font toujours par ligne de commande.

Nous pouvons faire un build d’un sous module

faire un test sur une méthode

Junit tutorial fogella

Gradle tutorial

 Pour créer une branche et y basculer tout de suite, vous pouvez lancer la commande git checkout avec l'option -b :

$ git checkout -b prob53

Switched to a new branch "prob53"

C'est un raccourci pour :

$ git branch prob53

$ git checkout prob53

Vous pouvez l'effacer avec l'option -d de la commande git branch :

$ git branch -d correctif

Deleted branch correctif (3a0874c).

Vous pouvez lancer vos tests dans la branche correctif, vous assurer que la correction est efficace et la fusionner dans la branche master pour la déployer en production. Vous réalisez ceci au moyen de la commande git merge :

$ git checkout master

$ git merge correctif

$ git rebase -i HEAD~5