



**ECOLE POLYVALENTE SUPERIEUR  
D'INFORMATIQUE ET D'ELECTRONIQUE**



# **FORMATION GIT-GITHUB**

**Présenté par :**

**THIOMBIANO MOHAMED**

**ANANI CAMILLE**

## Table des matières

I.	DEFINITION GIT ET GITHUB .....	3
II.	GITHUB CONFIGURATION ET PARAMETRAGE D'UN COMPTE .....	3
	2) Contribution à un projet.....	4
III.	LES BASES DE GIT .....	5
	1) Démarrer un dépôt GIT .....	6
	a) Initialisation d'un dépôt GIT dans un répertoire existant .....	6
	b) Cloner un dépôt existant .....	6
	2) Enregistrer les modifications dans le dépôt.....	7
	a) Vérifier l'état des fichiers .....	7
	b) Placer de nouveaux fichiers sous suivi de version.....	8
	c) Indexé des fichiers modifiés .....	9
	d) Inspecter les modifications.....	9
	e) Valider vos modifications .....	9
	f) Visualiser l'historique des validations .....	10
	g) Gestion de branche .....	10
	3) Référencer un dépôt distant .....	11
	a) Récupérer l'historique d'un dépôt distant .....	11
	b) Fusionner un dépôt distant et un dépôt local .....	11
	c) Récupérer les modifications du dépôt .....	11
	d) Envoyer des modifications sur le dépôt distant .....	11

## I. DEFINITION GIT ET GITHUB

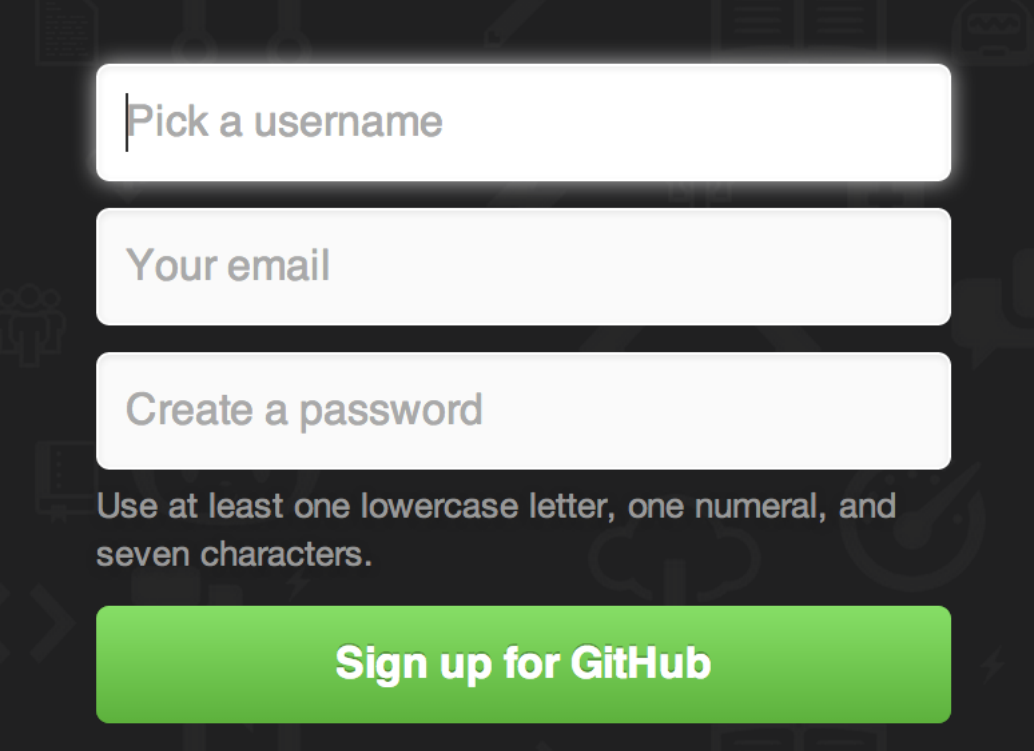
**Git** en tant que gestionnaire de version est un système qui enregistre l'évolution d'un fichier ou d'un ensemble de fichiers au cours du temps de manière à ce qu'on puisse rappeler une version antérieure d'un fichier à tout moment.

**GITHUB** est un service web d'hébergement et de gestion de développement de logiciels, utilisant le logiciel de gestion version GIT . En plus d'offrir l'hébergement de projets avec GIT, le site offre de nombreuses fonctionnalités habituellement retrouvées sur les réseaux sociaux comme les flux, la possibilité de suivre des personnes ou des projets ainsi que des graphes de réseaux pour les dépôts (repository) . GITHUB offre aussi la possibilité de créer un wiki et une page web pour chaque dépôt.

## II. GITHUB CONFIGURATION ET PARAMETRAGE D'UN COMPTE

### 1) Configuration et Paramétrage d'un compte

La première chose à faire consiste à créer un compte utilisateur gratuit. Allez tout simplement sur <https://github.com> , choisissez un nom d'utilisateur qui n'est pas déjà pris et saisissez une adresse électronique et un mot de passe, puis cliquez sur le gros bouton vert « **Sign up for GITHUB** » (S'inscrire sur GITHUB).

The image shows the GitHub sign-up form on a dark background. It consists of three white input fields stacked vertically. The first field is labeled 'Pick a username', the second 'Your email', and the third 'Create a password'. Below the third field, there is a line of text: 'Use at least one lowercase letter, one numeral, and seven characters.' At the bottom of the form is a large green button with the text 'Sign up for GitHub' in white. The background of the entire image is dark and features faint, repeating icons of people and code symbols.

Pick a username

Your email

Create a password

Use at least one lowercase letter, one numeral, and seven characters.

**Sign up for GitHub**

1-Formulaire d'inscription de GITHUB

La deuxième chose que vous verrez est la page des tarifs pour des projets améliorés mais il vaut mieux ignorer cela pour l'instant. GITHUB vous envoie un courriel pour vérifier l'adresse fournie. Suivez les instructions mentionnées, c'est très important (comme nous allons le voir plus tard).

## 2) Contribution à un projet

Après avoir configuré votre compte, examinons comment contribuer à un projet existant.

### Duplication des projets

Si vous souhaitez contribuer à un projet existant sur lequel vous n'avez pas le droit de pousser, vous pouvez dupliquer (fork) ce projet. Cela signifie que

GITHUB va faire pour vous une copie personnelle du projet. Elle se situe dans votre espace de nom et vous pouvez pousser dessus.

Ainsi, les gestionnaires de projets n'ont pas à se soucier de devoir ajouter des utilisateurs comme collaborateurs pour leur accorder un accès en poussée. Les personnes peuvent dupliquer un projet eux-mêmes, pousser sur leur copie personnelle et fournir leur contribution au dépôt originel en créant une requête de tirage (Pull Request), concept qui sera abordé par la suite. Ceci ouvre un fil de discussion avec possibilité de revue de code, pour que le propriétaire et le contributeur puissent discuter et modifier le code proposé jusqu'à ce que le propriétaire soit satisfait du résultat et le fusionne dans son dépôt.

Pour dupliquer un projet, visitez la page du projet et cliquez sur le bouton « **Fork** » en haut à droite de la page.



2-Bouton <<Fork>> de GITHUB

Quelques secondes plus tard, vous serez redirigé vers la page de votre nouveau projet, contenant votre copie modifiable du code.

### III. LES BASES DE GIT

Si vous ne deviez lire qu'un chapitre avant de commencer à utiliser Git, c'est celui-ci. Ce chapitre couvre les commandes de base nécessaires pour réaliser la vaste majorité des activités avec Git. À la fin de ce chapitre, vous devriez être capable de configurer et initialiser un dépôt, commencer et arrêter le suivi de version de fichiers, d'indexer et valider des modifications. Nous vous montrerons aussi comment paramétrer Git pour qu'il ignore certains fichiers ou

patrons de fichiers, comment revenir sur les erreurs rapidement et facilement, comment parcourir l'historique de votre projet et voir les modifications entre deux validations, et comment pousser et tirer les modifications avec des dépôts distants.

## 1) Démarrer un dépôt GIT

Vous pouvez principalement démarrer un dépôt Git de deux manières. La première consiste à prendre un projet ou un répertoire existant et à l'importer dans Git. La seconde consiste à cloner un dépôt Git existant sur un autre serveur.

### a) Initialisation d'un dépôt GIT dans un répertoire existant

Si vous commencez à suivre un projet existant dans Git, vous n'avez qu'à vous positionner dans le répertoire du projet et saisir :

```
$ git init
```

Cela crée un nouveau sous-répertoire nommé `.git` qui contient tous les fichiers nécessaires au dépôt . Pour l'instant, aucun fichier n'est encore versionné.

### b) Cloner un dépôt existant

Si vous souhaitez obtenir une copie d'un dépôt Git existant par exemple, un projet auquel vous aimeriez contribuer la commande dont vous avez besoin s'appelle `git clone`.

Vous clonez un dépôt avec `git clone [url]`. Par exemple, si vous voulez cloner la bibliothèque logicielle Git appelée `libgit2`, vous pouvez le faire de la manière suivante :

```
git clone https://github.com/libgit2/libgit2
```

Ceci crée un répertoire nommé “libgit2”, initialise un répertoire .git à l’intérieur, récupère toutes les données de ce dépôt, et extrait une copie de travail de la dernière version. Si vous examinez le nouveau répertoire libgit2, vous y verrez les fichiers du projet, prêts à être modifiés ou utilisés. Si vous souhaitez cloner le dépôt dans un répertoire nommé différemment, vous pouvez spécifier le nom dans une option supplémentaire de la ligne de commande :

```
git clone https://github.com/libgit2/libgit2 monlibgit2
```

Cette commande réalise la même chose que la précédente, mais le répertoire cible s’appelle :

**monlibgit2**

## 2) Enregistrer les modifications dans le dépôt

Vous avez à présent un dépôt Git valide et une extraction ou copie de travail du projet. Vous devez faire quelques modifications et valider des instantanés de ces modifications dans votre dépôt chaque fois que votre projet atteint un état que vous souhaitez enregistrer.

### a) Vérifier l’état des fichiers

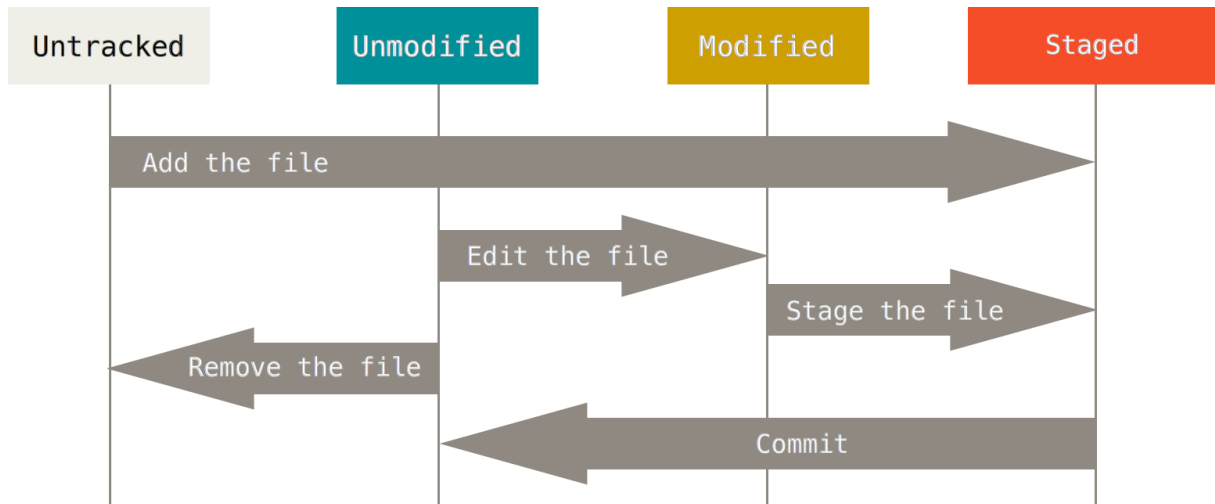
L’outil principal pour déterminer quels fichiers sont dans quel état est la commande **git status**. Si vous lancez cette commande juste après un clonage, vous devriez voir ce qui suit :

```
$ git status
```

Sur la branche master

Votre branche est à jour avec 'origin/master'.

Rien à valider, la copie de travail est propre.



3-Cycle de vie des états des fichiers

## b) Placer de nouveaux fichiers sous suivi de version

Pour commencer à suivre un nouveau fichier, vous utilisez la commande **git add**.

Par exemple pour commencer à suivre le fichier LISEZMOI, vous pouvez entrer ceci :

**git add LISEZMOI**



### c) Indexé des fichiers modifiés

Maintenant, modifions un fichier qui est déjà sous suivi de version. Si vous modifiez le fichier sous suivi de version appelé LISEZMOI et que vous lancez à nouveau votre commande `git status`, vous verrez ceci :

`git status`

Sur la branche master

Votre branche est à jour avec 'origin/master'.

Modifications qui ne seront pas validées :

modifié : LISEZMOI

### d) Inspecter les modifications

Si le résultat de la commande `git status` est encore trop vague lorsqu'on désire savoir non seulement quels fichiers ont changé mais aussi ce qui a changé dans ces fichiers on peut utiliser la commande `git diff`.

### e) Valider vos modifications

Maintenant que votre zone d'index est dans l'état désiré, vous pouvez valider vos modifications. Souvenez-vous que tout ce qui est encore non indexé tous les fichiers qui ont été créés ou modifiés mais n'ont pas subi de `git add` depuis que vous les avez modifiés ne feront pas partie de la prochaine validation. Ils resteront en tant que fichiers modifiés sur votre disque.

Dans notre cas, la dernière fois que vous avez lancé `git status`, vous avez vérifié que tout était indexé, et vous êtes donc prêt à valider vos modifications. La manière la plus simple de valider est de taper `git commit`.

#### f) Visualiser l'historique des validations

Après avoir créé plusieurs commits ou si vous avez cloné un dépôt ayant un historique de commits, vous souhaitez probablement revoir le fil des événements. Pour ce faire, la commande `git log` est l'outil le plus basique et le plus puissant.

#### g) Gestion de branche

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
```

Cette commande est un raccourci pour :

```
$ git branch prob53
```

```
$ git checkout prob53
```

Vous pouvez lancer vos tests, 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 prob53
```

À présent que votre travail a été fusionné, vous n'avez plus besoin de la branche `prob53`. Vous pouvez fermer le ticket dans votre outil de suivi des tâches et supprimer la branche :

**git branch -d prob53**

Pour visualiser les branches qui contiennent des travaux qui n'ont pas encore été fusionnés, vous pouvez utiliser la commande **git branch --no-merged** .

### **3) Référencer un dépôt distant**

#### **a) Récupérer l'historique d'un dépôt distant**

Après les modifications du dépôt, vous pouvez récupérer l'historique du dépôt distant vers le dépôt local en exécutant la commande.

**\$ git fetch [nom-de-depot]**

#### **b) Fusionner un dépôt distant et un dépôt local**

Pour fusionner la branche du dépôt dans la branche locale courante, exécutez la commande :

**\$ git merge [nom-de-depot]/[branche]**

#### **c) Récupérer les modifications du dépôt**

Après modification vous pouvez récupérer tout l'historique du dépôt github et incorporer les modifications en exécutant la commande :

**\$ git pull**

#### **d) Envoyer des modifications sur le dépôt distant**

Pour envoyer les modifications effectuées en local sur le dépôt distant, exécutez la commande :

**\$ git push [alias] [branche du dépôt distant]**