**1. Démarrer avec Git**

Nous pouvons imaginer que le design web est comme un bâtiment et que Git est l'un des nombreux piliers importants sur lesquels le design web est construit. Nos principaux objectifs sont les suivants :

* Comprendre ce qu'est Git
* Apprendre à utiliser les commandes de base de Git
* Découvrir comment configurer Git
* Comprendre ce qu'est Github et comment travailler avec des dépôts distants

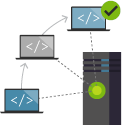
.

**2. Comprendre ce qu'est un système de contrôle de version**

Un système de contrôle de version est un outil logiciel qui aide les équipes de développeurs à suivre et à gérer les modifications du code source au fil du temps.

Lorsque vous modifiez et ajoutez des éléments à votre code, vous demandez au système de contrôle de version de prendre un instantané de vos fichiers ou d'enregistrer un point de contrôle de votre progression. Le système de contrôle de version enregistre cet instantané de façon permanente afin que vous puissiez le retrouver plus tard si vous en avez besoin.

Sans système de contrôle de version, vous êtes tenté de conserver plusieurs copies du code sur votre ordinateur. Les systèmes de contrôle des versions résolvent ce problème en vous présentant toutes les versions de votre code avec un historique clair des modifications apportées. Cela vous permet de revenir à des versions plus anciennes du code et d'essayer une approche différente.



Les systèmes de contrôle des versions résolvent ce problème en vous présentant toutes les versions de votre code avec un historique clair des modifications apportées.

**3. Explorer Git**

Git est un VCS libre et gratuit conçu pour gérer tous les projets, des plus petits aux plus grands, avec rapidité et efficacité.

En clair, Git est un outil qui permet aux développeurs de suivre les versions de leur code au fil du temps. Pour ce faire, il crée des "instantanés" de l'état actuel de la base de code chaque fois que vous le lui demandez.

Git est essentiel lorsque vous collaborez avec d'autres développeurs pour vous assurer qu'il n'y a pas de conflits de code entre eux et que les "instantanés" précédents du code peuvent être revus si nécessaire.

Par exemple, si vous êtes en train de coder et que vous cassez ou plantez accidentellement l'application, vous venez de perdre tous vos progrès et vous êtes contraint de repartir de zéro. Cependant, il est plus facile et plus sûr d'utiliser Git et de revenir à une version antérieure du code.

**Ces ressources peuvent vous aider**

**Les bases de Git**

[https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/](https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/" \t "_blank)

**1. Installer Git**

Git n'est généralement pas configuré par défaut sur votre ordinateur, vous devez donc l'installer et le configurer avant de pouvoir commencer à l'utiliser pour gérer votre code. Il est important de maintenir Git à jour, comme tous les autres logiciels de votre machine. Les mises à jour vous protègent contre les failles de sécurité, corrigent les bogues et vous donnent accès à de nouvelles fonctionnalités.

La méthode recommandée pour l'installation et la maintenance de Git est indiquée ci-dessous pour les trois principales plateformes:

* **Windows** Téléchargez et installez Git pour [Windows](https://git-scm.com/download/win). Une fois installé, vous pourrez utiliser Git à partir de l'invite de commande ou de PowerShell. Nous vous recommandons de conserver les paramètres par défaut sélectionnés lors de l'installation, sauf si vous avez une bonne raison de les modifier.
* **Linux** Dans le terminal, exécutez simplement sudo apt install git-all.
* **Mac** La meilleure chose à faire est d'installer [Homebrew](https://brew.sh/) puis, à partir du terminal, de lancer la commande brew install git.

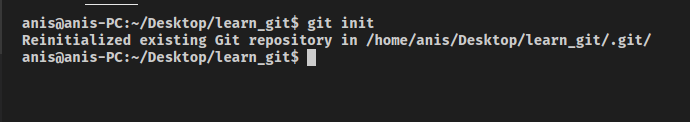
**2. Commencer à utiliser Git**

Une fois Git installé, il faut "initialiser" un dépôt avant de pouvoir commencer à l'utiliser. C'est très facile à faire, il suffit de suivre mon exemple :

1. Accéder au dossier désiré en utilisant le terminal (prompt cmd).
2. Pour initialiser un dépôt, il suffit de lancer :

git init

Sortie



***P.S****: Cette commande vient de créer un dossier caché appelé .git, où toute la magie se produit.*

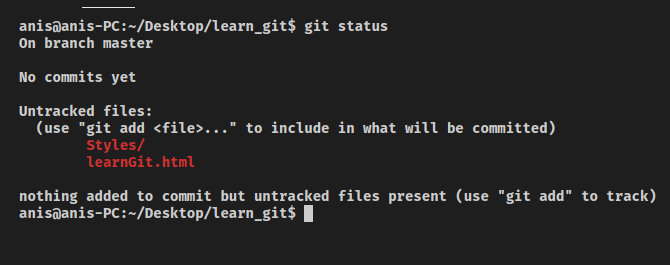
**3. ajouter et commiter des fichiers**

Vous vous souvenez du dossier que nous avons créé ? Il est temps de l'utiliser. Nous allons créer un fichier et le placer dans ce dossier. N'hésitez pas à ajouter ce que vous voulez !

Maintenant que nous avons quelques fichiers dans notre dépôt, voyons comment Git les traite. Pour vérifier l'état actuel de votre dépôt, nous utilisons la commande git status.

git status

Sortie



**4. Ajouter des fichiers à suivre par Git**

À ce stade, nous n'avons pas de fichiers à suivre par Git. Nous devons ajouter des fichiers spécifiquement à Git afin de lui demander de les suivre. Nous ajoutons des fichiers à l'aide de la commande "add". Après avoir lancé la commande git add, Git ajoutera tous les fichiers du dépôt dans une zone intermédiaire appelée **staging area**. Nous pouvons également ajouter ce que nous voulons simplement en exécutant git add monNomDeFichier

git add .

Sortie

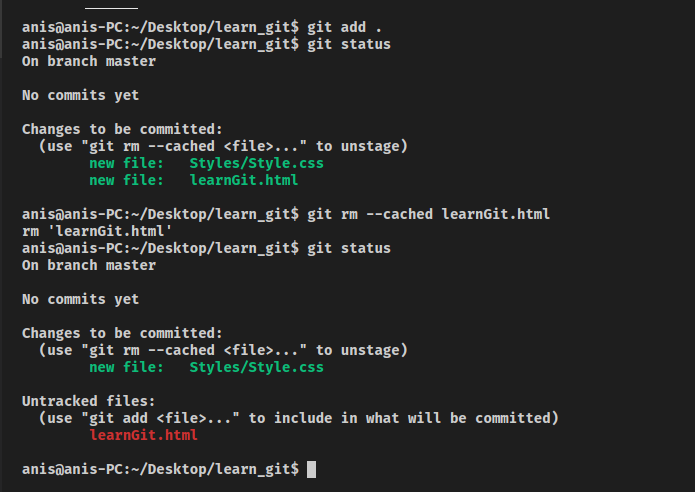


**5. Suppression de fichiers**

Disons que vous avez ajouté des fichiers à Git et que vous ne voulez pas qu'il les suive. Dans une telle situation, vous devez demander à Git d'arrêter de les suivre. Cependant, l'exécution d'une simple commande git rm ne l'enlèvera pas seulement de Git, mais aussi de votre système de fichiers local ! Pour demander à Git d'arrêter de suivre un fichier tout en le conservant dans votre système local, exécutez la commande suivante :

git rm --cached [nom\_du\_fichier]

Sortie



**6. Validation des modifications**

Une fois que vous avez mis en scène vos fichiers, vous pouvez les valider dans Git. Imaginez une commande de validation comme un instantané à un moment donné, auquel vous pouvez revenir pour accéder à votre référentiel à ce stade. Vous attribuez un message de validation à chaque validation, que vous pouvez transmettre avec le préfixe -m.

git commit -m 'first commit'

Sortie



**Ces ressources peuvent vous aider**

**Les commandes basiques git**

[https://dzone.com/articles/top-20-git-commands-with-examples](https://dzone.com/articles/top-20-git-commands-with-examples" \t "_blank)

**1. Statut du fichier**

Git voit chaque fichier dans votre copie de travail comme l'une des trois choses suivantes :

* **Tracked:** Un fichier qui a été précédemment mis à l'échelle ou livré.
* **Untracked:** Un fichier qui n'a pas été mis à l'échelle ou livré.

Les fichiers ignorés sont généralement des artefacts construits et des fichiers générés par la machine qui peuvent être dérivés de la source de votre dépôt ou qui ne devraient pas être livrés.

**PS** : Si nous avons un fichier que nous ne voulons pas suivre, nous le plaçons simplement à l'intérieur de la zone .gitignore.

**2. Paramètres de configuration**

Le fichier .gitconfig contient une liste de configurations qui affectent le comportement des commandes git. Pour le manipuler, on utilise donc la commande git config <configuration>. Ainsi, pour changer le nom et l'email utilisés par Git pour identifier l'utilisateur, nous lançons la commande suivante : (PS. changez YOUR.NAME et YOUR.EMAIL avec vos propres valeurs.)

git config --global user.name "VOTRE NOM"

git config --global user.email "VOTRE EMAIL"

**3. Les alias Git**

Souvent, vous vous retrouverez à taper des commandes git encore et encore. Par exemple, git add, git init et git status sont des commandes que vous utiliserez de manière répétée.

La chose intelligente à faire dans ce cas est d'utiliser un raccourci, ou un alias. Les alias rendent votre expérience Git plus simple, plus facile et plus familière.

Pour créer un alias temporaire, qui durera aussi longtemps que votre session de terminal est ouverte, vous pouvez taper:

git config alias.KEYBOARD\_SHORTCUT COMMAND

Ainsi, si nous voulions taper git st et le faire fonctionner comme git status, nous devrions taper git config alias.st status. Nous pouvons maintenant taper git st et obtenir la même sortie que si nous avions tapé git status.

Si vous souhaitez que votre alias fasse partie de votre configuration globale, ajoutez la commande --global après git config. Par exemple, pour aliaser git i à git init globalement, vous devez taper git config --global alias.i init.

.

**Ces ressources peuvent vous aider**

**Installation Git**

[https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup](https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup" \t "_blank)

**1. Statut du fichier**

Git voit chaque fichier dans votre copie de travail comme l'une des trois choses suivantes :

* **Tracked:** Un fichier qui a été précédemment mis à l'échelle ou livré.
* **Untracked:** Un fichier qui n'a pas été mis à l'échelle ou livré.

Les fichiers ignorés sont généralement des artefacts construits et des fichiers générés par la machine qui peuvent être dérivés de la source de votre dépôt ou qui ne devraient pas être livrés.

**PS** : Si nous avons un fichier que nous ne voulons pas suivre, nous le plaçons simplement à l'intérieur de la zone .gitignore.

**2. Paramètres de configuration**

Le fichier .gitconfig contient une liste de configurations qui affectent le comportement des commandes git. Pour le manipuler, on utilise donc la commande git config <configuration>. Ainsi, pour changer le nom et l'email utilisés par Git pour identifier l'utilisateur, nous lançons la commande suivante : (PS. changez YOUR.NAME et YOUR.EMAIL avec vos propres valeurs.)

git config --global user.name "VOTRE NOM"

git config --global user.email "VOTRE EMAIL"

**3. Les alias Git**

Souvent, vous vous retrouverez à taper des commandes git encore et encore. Par exemple, git add, git init et git status sont des commandes que vous utiliserez de manière répétée.

La chose intelligente à faire dans ce cas est d'utiliser un raccourci, ou un alias. Les alias rendent votre expérience Git plus simple, plus facile et plus familière.

Pour créer un alias temporaire, qui durera aussi longtemps que votre session de terminal est ouverte, vous pouvez taper:

git config alias.KEYBOARD\_SHORTCUT COMMAND

Ainsi, si nous voulions taper git st et le faire fonctionner comme git status, nous devrions taper git config alias.st status. Nous pouvons maintenant taper git st et obtenir la même sortie que si nous avions tapé git status.

Si vous souhaitez que votre alias fasse partie de votre configuration globale, ajoutez la commande --global après git config. Par exemple, pour aliaser git i à git init globalement, vous devez taper git config --global alias.i init.

.

**Ces ressources peuvent vous aider**

**Installation Git**

[https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup](https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup" \t "_blank)

**1. Création d'un dépôt distant**

Une fois que vous avez un compte GitHub, rendez-vous sur https://github.com/new et créez un dépôt. Dans cet exemple, nous allons créer un dépôt appelé first repo. Ne vous préoccupez pas de la description ou de cocher la case pour initialiser le dépôt avec **README**.

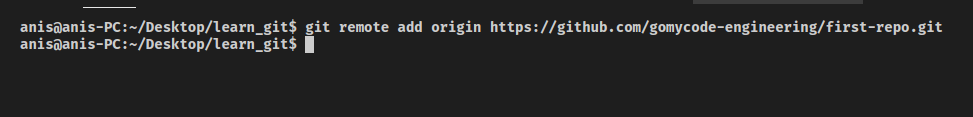
Après avoir créé le dépôt, GitHub vous donnera quelques instructions pour commencer. Ces instructions devraient ressembler à ceci (l'url vers la fin dépendra de votre nom d'utilisateur GitHub)



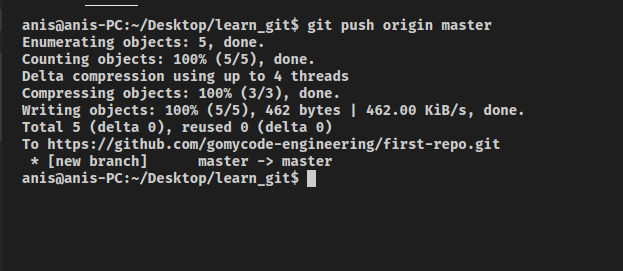
Les instructions devraient ressembler à ceci (l'url vers la fin dépendra de votre nom d'utilisateur GitHub).

**2. pousser le code vers GitHub**

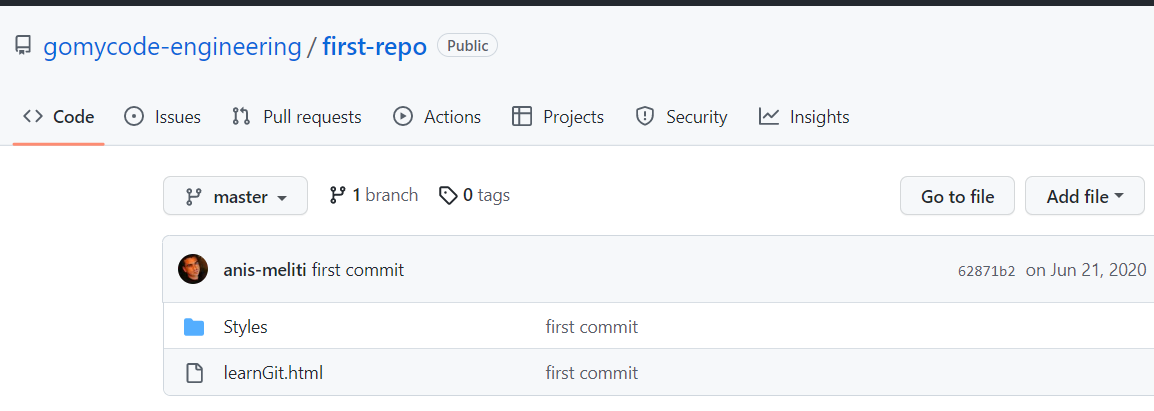
Après avoir créé un dépôt dans GitHub, nous devons ajouter notre code local (après les commits) au dépôt distant. C'est très simple. Tout d'abord, nous devons attacher l'adresse du dépôt GitHub à notre projet local, simplement en exécutant git remote add origin YOUR\_REPO\_URL.



Maintenant, après avoir fourni l'URL distante au dépôt local pour emmener les fichiers locaux vers l'hôte distant, nous devons exécuter cette commande git push origin master.



Et voilà ! Nos fichiers sont maintenant dans GitHub.



Les fichiers sont maintenant dans GitHub.

**Ces ressources peuvent vous aider**

**Travailler avec remote**

[https://help.github.com/en/desktop/contributing-to-projects/working-with-your-remote-repository-on-github-or-github-enterprise](https://help.github.com/en/desktop/contributing-to-projects/working-with-your-remote-repository-on-github-or-github-enterprise" \t "_blank)

**1. Fork**

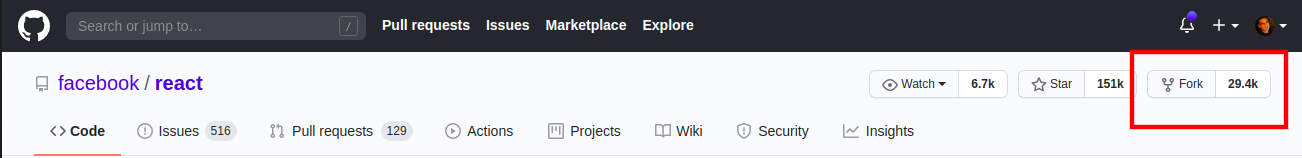
Maintenant que nous savons comment pousser du code sur GitHub, explorons l'une des fonctionnalités importantes de GitHub : **forking**.

Lorsque vous collaborez avec d'autres personnes, vous ne pouvez pas pousser directement vers le dépôt original (Imaginez que vous puissiez jouer avec l'un des plus grands projets et dépôts open source au monde... ce serait fou).

La chose logique à faire est donc de faire une copie du dépôt distant de quelqu'un d'autre et de s'assurer qu'il est sous notre nom d'utilisateur afin que nous puissions y pousser du code. Cela garantit que nos expériences avec le dépôt n'affectent pas la version originale.

Pour pratiquer le forking, allez sur n'importe quel dépôt sur [github](https://github.com/) et en haut à droite, vous verrez un bouton avec le texte Fork. Cliquez sur ce bouton et vous aurez une copie du dépôt à votre nom!

N'oubliez pas que le "forking" est strictement une fonctionnalité de GitHub et qu'il n'est en aucun cas lié à Git. Il s'agit simplement d'un moyen de faire votre propre copie d'un dépôt sur votre compte où vous avez la permission de pousser votre code vers GitHub.

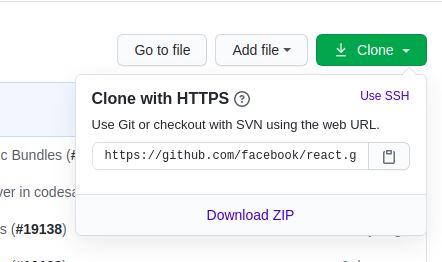


La fonction "forking" est strictement une fonctionnalité de GitHub et n'est en aucun cas liée à GitHub.

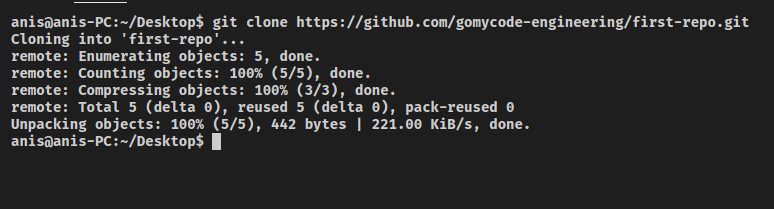
**2 Cloner**

Une fois que vous avez appliqué le "forking" sur le dépôt, vous devez le sélectionner (le dépôt distant que vous venez de créer) et télécharger le code sur votre ordinateur local (c'est-à-dire créer un dépôt local). Au lieu de créer un dossier et de passer par tout le processus git init et d'ajouter un dépôt distant, vous pouvez utiliser la commande git clone, qui accepte un lien vers le dépôt et le télécharge dans un dossier (avec tout ce qui est déjà configuré !).

Pour ce faire, cliquez sur le bouton clone et cela nous fournira l'URL du dépôt distant. Il suffit ensuite de copier l'URL. Dans le terminal, lancez git clone THE\_COPYED\_ADDRESS.



git clone https://github.com/facebook/react.git

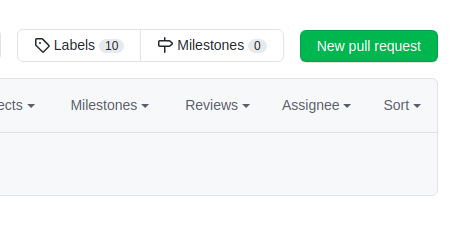


Il n'y a pas d'autre solution que d'utiliser l'adresse de l'utilisateur.

**3 Demande d'extraction**

Maintenant, disons que vous collaborez avec une organisation sur GitHub (d'où vous avez extrait le dépôt) et que vous souhaitez fusionner vos modifications avec le dépôt original que vous avez extrait (rappelez-vous que vous ne pouvez pas simplement pousser vers lui, parce que vous n'avez pas la permission de le faire). Vous pouvez émettre une demande d'extraction et la personne qui peut accorder la permission peut soit la fusionner soit la rejeter.

Pour ce faire, cliquez sur "New pull request" et ensuite sur "Create pull request". Vous devriez alors être en mesure d'aller dans le dépôt original et de voir votre demande d'extraction ou "PR".



La demande d'extraction est une demande d'autorisation de la personne qui peut accorder la permission de la fusionner ou de la rejeter.

**4 Immersion Git**

Il s'agit d'une visite guidée des principes fondamentaux de Git, inspirée par le principe selon lequel savoir une chose, c'est la faire.

**Exercice pratique :**

* Démarrez l'immersion Git [ici](https://gitimmersion.com/lab_01.html)
* Essayez de terminer les laboratoires autant que possible
* Bonne chance.

**Ces ressources peuvent vous aider**

**Introduction Github**

[https://guides.github.com/introduction/flow/](https://guides.github.com/introduction/flow/" \t "_blank)

**Github workflow**

[https://guides.github.com/introduction/flow/](https://guides.github.com/introduction/flow/" \t "_blank)

Mode Concentration