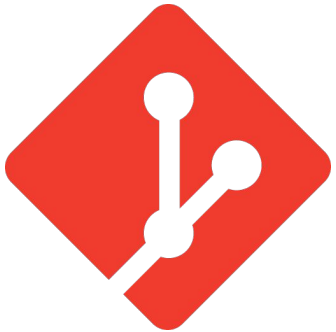



Aa

Objectifs

- **Créer une branche et la fusionner** avec la branche main via l'interface graphique GitHub
- Créer une branche via **Git CLI**
- Suivre le flow **GitHub** en utilisant Git

Un flow ? Kesako ?

Un **flow** (parfois aussi appelé *workflow*) est une **série d'étapes** que les membres de l'équipe doivent suivre pour **compléter une tâche.**

Dans notre cas, la **tâche consiste à apporter des modifications au code** d'un projet web.

S'entendre sur les étapes à suivre permet à plusieurs personnes de **s'organiser.** En effet, une telle approche :

- **Minimise** le risque de **conflits** car elle empêche plusieurs développeurs de travailler sur le même code en même temps (cela se fait grâce à l'utilisation de *branches* que nous découvrirons plus tard)
- Facilite le **contrôle** du code produit par chaque développeur, ce qui assure la **qualité du projet** (ceci est fait par l'utilisation de *pull requests* que nous verrons plus tard également).

Un flow ? Kesako ?

Nous allons découvrir progressivement le **flow GitHub** qui nous servira dans nos futurs projets.

- Dans un premier temps via l'interface graphique de Github
- Dans un second temps sur notre machine en lignes de commandes Git

Flow Github

GitHub fonctionne grâce à un système de **branches**.

Pour avoir une première idée de ce dont il s'agit, lire **cet article** (uniquement la partie "Yes, Master" et le premier paragraphe de la partie "Branching Out").

Découvrir comment les branches sont utilisées dans le flow GitHub en suivant **ce tutoriel**.

Une fois terminé, dans le repo GitHub créé en suivant le tutoriel, cliquer sur le bouton **Insights** puis sur **Network** pour accéder à un **historique visuel du projet.** Cela permet de voir la branche créée à partir de la branche **main**, le commit réalisé sur cette branche et la fusion avec la branche **main**.

Le **network graph** est un moyen simple de **visualiser les branches** créée dans chaque repo GitHub - il est très utile !

Git : la création de branches locales

Nous venons d'apprendre à utiliser l'interface graphique de GitHub.

Cette nouvelle branche nous permet de **modifier** un fichier directement dans le navigateur, puis de le **fusionner** avec le code de base pour le partager avec le reste de notre équipe.

Mais modifier du code en utilisant uniquement l'interface GitHub, c'est-à-dire faire tout dans le navigateur, n'est pas très pratique : il est plus pratique de **travailler localement** (sur notre ordinateur), dans un éditeur de texte, pour enregistrer nos modifications en utilisant Git (**git add** et **git commit**), et seulement ensuite de **les pousser vers notre repo GitHub.**

✅Il est justement possible de **créer une branche locale** pour effectuer des commits sur celle-ci, puis de pousser cette branche vers un repo GitHub (l'interface graphique de GitHub prend alors le relais pour une pull request).

First things first, nous allons apprendre à **créer une branche via Git** en regardant cette vidéo (de 0min à 5min30).

Git : la création de branches locales

Nous venons d'apprendre à utiliser l'interface graphique de GitHub.

Cette nouvelle branche nous permet de **modifier** un fichier directement dans le navigateur, puis de le **fusionner** avec le code de base pour le partager avec le reste de notre équipe.

Mais modifier du code en utilisant uniquement l'interface GitHub, c'est-à-dire faire tout dans le navigateur, n'est pas très pratique : il est plus pratique de **travailler localement** (sur notre ordinateur), dans un éditeur de texte, pour enregistrer nos modifications en utilisant Git (**git add** et **git commit**), et seulement ensuite de **les pousser vers notre repo GitHub.**

✅Il est justement possible de **créer une branche locale** pour effectuer des commits sur celle-ci, puis de pousser cette branche vers un repo GitHub (l'interface graphique de GitHub prend alors le relais pour une pull request).

First things first, nous allons apprendre à **créer une branche via Git** en regardant cette vidéo (de 0min à 5min30).

Git et Github : le flow final

La branche a été créée localement sur notre machine.

À nous de l’envoyer sur Github et de lier cette nouvelle branche à la branche principale.

Cette vidéo montre les étapes unes à unes.

Git : la création de branches locales

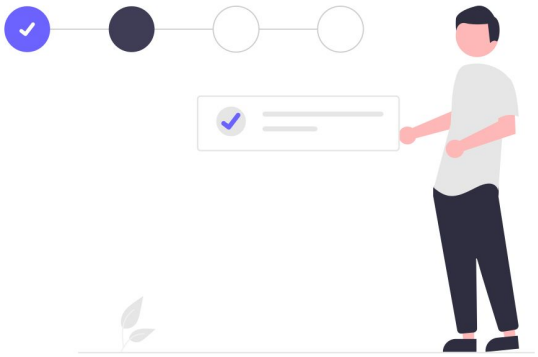
Petite exercice

- Créer un dépôt Git **distant** (repo sur Github)
- Initialiser un dépôt Git en **local** (ordinateur)
- Relier le dépôt distant (Github) au dépôt local (ordinateur)
- En local, créer une nouvelle branche
- Effectuer des modifications
- Commit ces modifications
- Push cette nouvelle branche (et ses modifications) sur le dépôt distant Github

En résumé

- Suivre un flow minimise les risques de conflits.
- GitHub fonctionne via un système de **branches**.
- Les branches permettent de travailler sur une version séparée
- Nous pouvons créer des branches via l'interface graphique ou via la CLI.

Pratiquons !



- Créer un nouveau dépôt nommé **website-flow** sur GitHub en cochant «Initialize this repository with a README»
- Cloner le dépôt sur l'ordinateur
- Localement (via Git), créer une nouvelle branche appelée **cheese** et se placer dessus.
- Modifier le fichier readme (par exemple, en écrivant une liste de fromages à pizza)
- En étant sûr d'être bien sur la branche cheese, valider les modifications et les transférer dans le dépôt GitHub.
- Sur GitHub, fusionner la branche cheese avec la branche **main** via une pull request, puis supprimer la branche cheese.