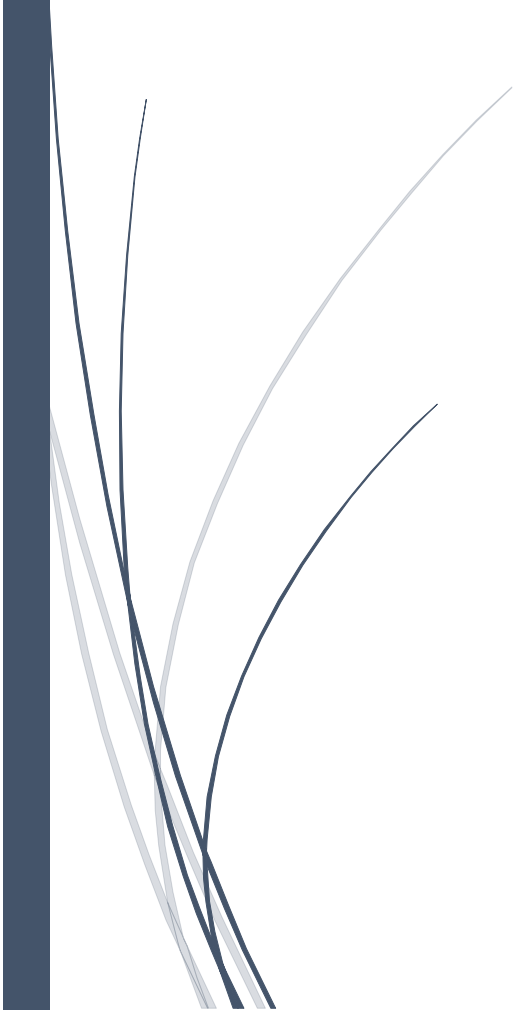
29/01/2023

Guide utilisateur Git et GitHub



Alexandre Cancel et Ganesh Jhugroo

CANCEL Alexandre
JHUGROO Ganesh

Table des matières

ıntroduction

Lors de notre projet nous étions en binôme. Ainsi nous avons dû collaborer ensemble pour effectuer notre travail. Afin de pouvoir travailler efficacement, nous avons décidé d'utiliser Git et GitHub. Avec ces deux outils nous étions à disposition d'un dépôt local ainsi que d'un dépôt distant. Nous pouvions aussi faire du contrôle de versions, ce qui est très utile dans un projet comme le nôtre qui consistait à créer un SOKOBAN.

En effet dans un projet avec des collaborateurs, chacun travaille sur une première version que tout le monde utilise. Pour cela, chacun fait une copie du document original et travaille dessus sur son ordinateur. Ainsi chaque collaborateur peut travailler dessus séparément quand il le souhaite. Une fois les modifications faites, chacun renvois son document. Mais au lieu de remplacer le document original, les documents sont placés en attente. Ainsi chaque collaborateur peut vérifier le travail de chacun avant de l'intégrer dans le document original. Une fois le travail approuvé, les parties modifiées sont intégrées à la version de production. Etant donné que le système effectue un suivi automatique de qui a changé quoi et quand, il est beaucoup plus facile de gérer de grands projets. De plus grâce à ce système, il est possible en cas d'erreur de revenir en arrière juste avec quelques raccourcis clavier.

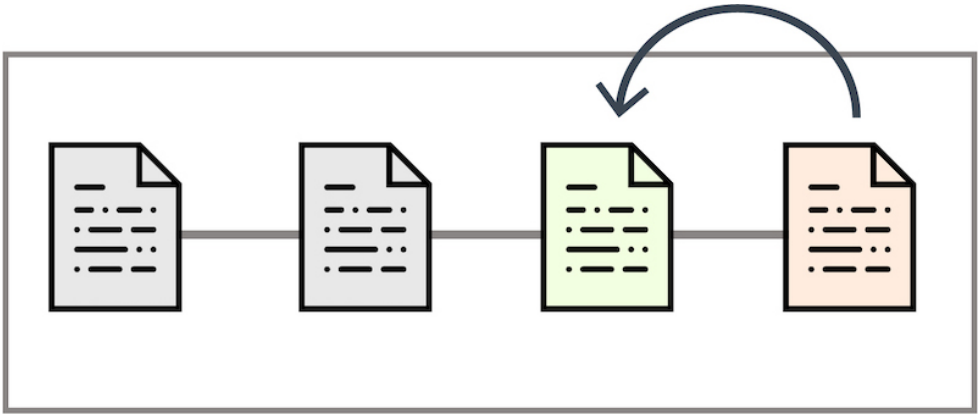


Figure 1 : Gestion de version

Présentation GitHub et Git

Git :

Git est un logiciel de gestion de version qui permet aux utilisateurs de suivre les modifications apportées à des fichiers et dossiers sur un projet. Il permet également à plusieurs personnes de travailler sur un projet en même temps, réconciliant les modifications apportées par chacun. C'est grâce à Git que nous allons pour créer notre dépôt local et ainsi travailler chacun de son côté.

Figure 2 : Logo Git

GitHub :

Concernant GitHub c'est un service en ligne basé sur Git qui permet aux utilisateurs de stocker et de partager des projets en utilisant Git. Il offre également des fonctionnalités supplémentaires telles que les issues, les pull requests et les wikis pour faciliter la collaboration sur des projets. C'est sur ce site que notre dépôt distant sera hébergé.

Figure 3 : Logo GitHub

En résume, Git est un outil de gestion de version qui permet de suivre les modification d'un projet, tandis que GitHub est un service en ligne basé sur Git qui facilite la collaboration et le partage de projets en utilisant des fonctionnalités supplémentaires.

Travaillez depuis votre dépôt local Git

Au début, nous avons initialisé notre dépôt avec la commande **git clone** suivi du lien de notre dépôt distant. C'est notre dossier de travail (Working directory) dans lequel nous allons créer nos codes. Après avoir écrit nos codes dans des fichiers, nous allons les indexer. C'est-à-dire que nous allons les conserver dans une zone que l'on appelle index ou stage (via la commande **Git Add**). A cette étape, nos modifications sont en attente de validation. Lorsque l'on est prêt à les valider, on va les envoyer dans le repository (dépôt local) l'endroit où seront stockées toutes les versions de notre code (via la commande **Git Commit**). C'est ce que l'on appelle commiter un fichier. Les modifications sont désormais enregistrées. Maintenant que nous avons une nouvelle version de notre code, il ne nous reste plus qu'à l'envoyer vers notre dépôt distant GitHub pour en conserver une copie sur le cloud (via la commande **Git Push**). C'est ce que l'on appelle pusher son code.

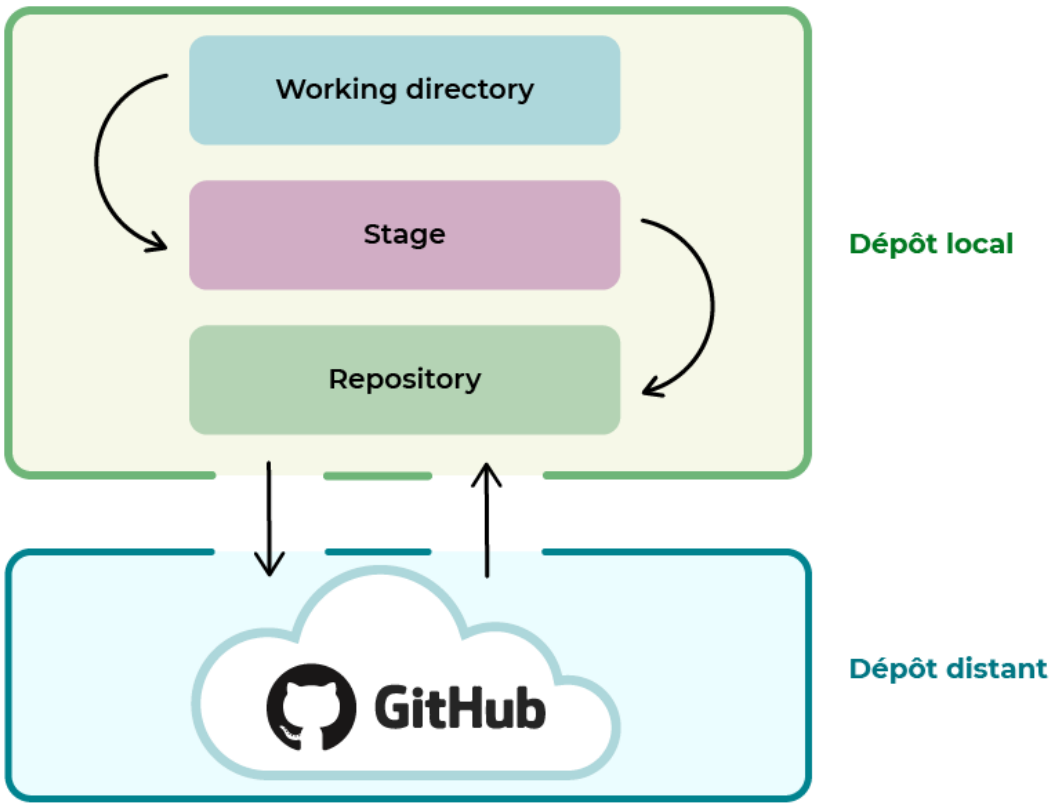


Figure 4 : Fonctionnement de Git

Appréhendez le système de branches

Le principal atout de Git est **son système de branches**.

Les différentes branches correspondent à des copies de votre code principal à un instant T, où vous pourrez tester toutes vos idées sans que cela impacte votre code principal.
Sous Git, la branche principale est appelée la ***branche main.***

Cette branche principale portera l'intégralité des modifications effectuées. Le but n'est donc pas de réaliser les modifications directement sur cette branche, mais de les réaliser sur d'autres branches, et après divers tests, de les intégrer sur la branche principale. Il faut voir les branches comme autant de dossiers différents.

Voici les différentes commandes pour switcher de branche en branche.

Pour connaître les branches présentes dans notre projet, il faut taper la ligne de commande :

```
git branch
```

Dans un premier temps, vous n'aurez que :

```
git branch
* main
```

L'étoile signifie que c'est la branche sur laquelle vous vous situez, et que c'est sur celle-ci qu'actuellement vous réalisez vos modifications.

Nous avons donc notre branche main, et nous souhaitons maintenant réaliser une autre branche pour un des collaborateurs. Pour cela, on tape :

```
git branch nom_de_la_branche
```

Cette commande va créer la branche nom_de_la_branche en local. Cette dernière ne sera pas dupliquée sur le dépôt distant.

Pour basculer de branche, nous allons utiliser :

```
git checkout nom_de_la_branche
```

La branche va fonctionner comme un dossier virtuel. Avec **git checkout**, on va être téléporté dans le dossier virtuel nom_de_la_branche.

Vous pouvez désormais réaliser votre évolution sans toucher à la branche main qui abrite votre code principal fonctionnel. Vous pouvez re-basculer si besoin à tout moment sur la branche main, sans impacter les modifications de la branche nom_de_la_branche.

Vous avez réalisé des évolutions sur la branche nom_de_la_branche, il faut maintenant demander à Git de les enregistrer, de créer une nouvelle version du projet comprenant les évolutions réalisées sur la branche nom_de_la_branche.

Vous devez créer un "commit" grâce à la commande :

```
git commit -m “message”
```

Vos modifications sont maintenant enregistrées avec la description “message”.

Avec la commande git commit, nous avons enregistré des modifications en local, uniquement. Il ne vous reste plus qu’à envoyer les modifications réalisées sur le dépôt distant grâce à la commande git push comme expliqué précédemment.

À présent, il faut intégrer l'évolution réalisée dans la branche " nom_de_la_branche " à la branche principale "main". Pour cela, vous devez utiliser la commande "git merge".

Cette commande doit s'utiliser à partir de la branche dans laquelle nous voulons apporter les évolutions. Dans notre cas, la commande s’effectuera donc dans la branche main. Pour y retourner, utilisez la commande :

```
git checkout main
Switched to branch 'main'
```

Maintenant que vous êtes dans votre branche principale, vous pouvez fusionner votre branche " nom_de_la_branche " à celle-ci grâce à la commande suivante :

```
git merge nom_de_la_branche
Merge made by the 'recursive' strategy
```

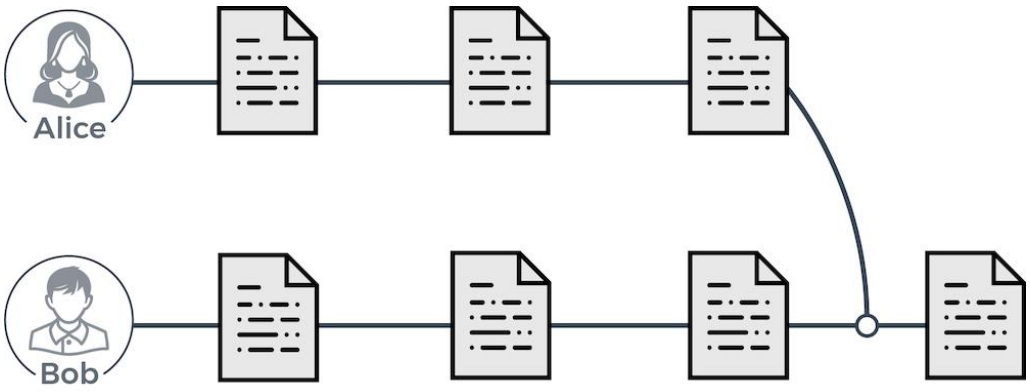


Figure 5 : Schéma représentant notre mode de fonctionnement

Procédure de réalisation de notre projet avec Git et GitHub:

Voici comment procéder si vous voulez travailler à deux en utilisant Git et GitHub.

Dans un premier temps, assurez-vous d'avoir un compte GitHub avant d'installer Git.

1. Créez un compte GitHub si vous n'en avez pas déjà un.
2. Créez un compte dépôt (repository) pour votre projet en cliquant sur le bouton « New » sur votre page d'accueil de GitHub. Donnez un nom à votre dépôt et une courte description.
3. Invitez l'autre collaborateur sur le projet en allant dans les paramètres de votre dépôt et en ajoutant son nom d'utilisateur dans la section « Collaborators »
4. Téléchargez Git
5. Suivre les différentes étapes de téléchargement
6. Utilisez Git pour cloner le dépôt sur votre ordinateur local. Ouvrez un terminal et tapez : « git clone https://github.com/votre_nomd'utilisateur/nom_du_dépot.git »
7. Travaillez sur les fichiers de votre projet localement sur cotre ordinateur. Utilisez les commandes Git pour suivre les modifications :
    a. git status : pour voir les fichiers modifiés
    b. git add pour ajouter les fichiers modifiés à l'index Git
    c. git commit : pour enregistrer les modifications dans l'historique local
8. Utilisez la commande git push pour envoyer les modifications sur le dépôt GitHub
9. Utilisez la fonctionnalité « Pull requests » de GitHub pour demander à l'autre collaborateur de vérifier et de fusionner vos modifications avec la version principale du projet.
10. Utilisez la commande git pull pour récupérer les modifications de l'autre collaborateur depuis le dépôt GitHub.
11. Utilisez git merge pour fusionner les modifications de l'autre collaborateur avec votre version locale.

Il est important de communiquer régulièrement avec le collaborateur et de s'assurer que les modifications apportées ne causent pas de conflits avant de les fusionner.