GIT

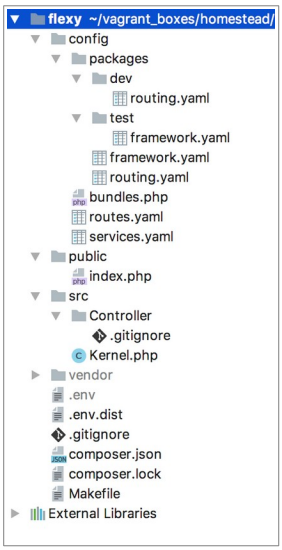
Commandes de base

Introduction

Problématique 1 : Comment travailler en FTP à plusieurs ? Risque de modifier les mêmes fichiers. Git détecte les différences avant de mettre à jour un fichier.

Problématique 2 : J'ai modifié une dizaine de fichiers dans un gros projet, comment mettre en ligne toutes les modifications sans rien oublier. Git détecte les changements et n'oublie aucun fichiers.

Problématique 3 : J'ai un projet stable et je veux le faire évoluer ; mais je risque de tout casser. Git enregistre l'historique des modifications dans chaque fichier et permet de revenir en arrière.



Programme pédagogique

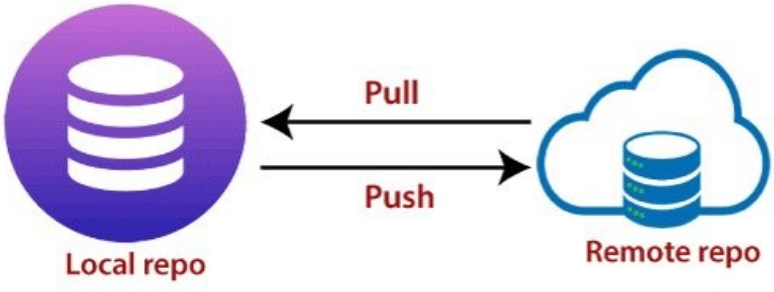
Travailler en local

Apprentissage des commandes de bases

À chaque fois que vous validez ou enregistrez l'état du projet dans Git, il prend un instantané du contenu de votre espace de travail.

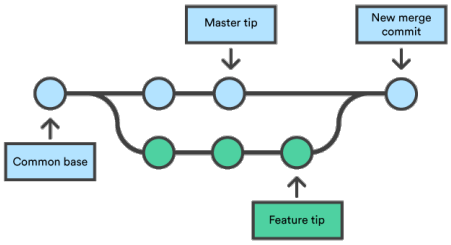
Travailler avec des dépots distants

Récupérer du code source et pousser des modifications



Travailler sur différentes branches

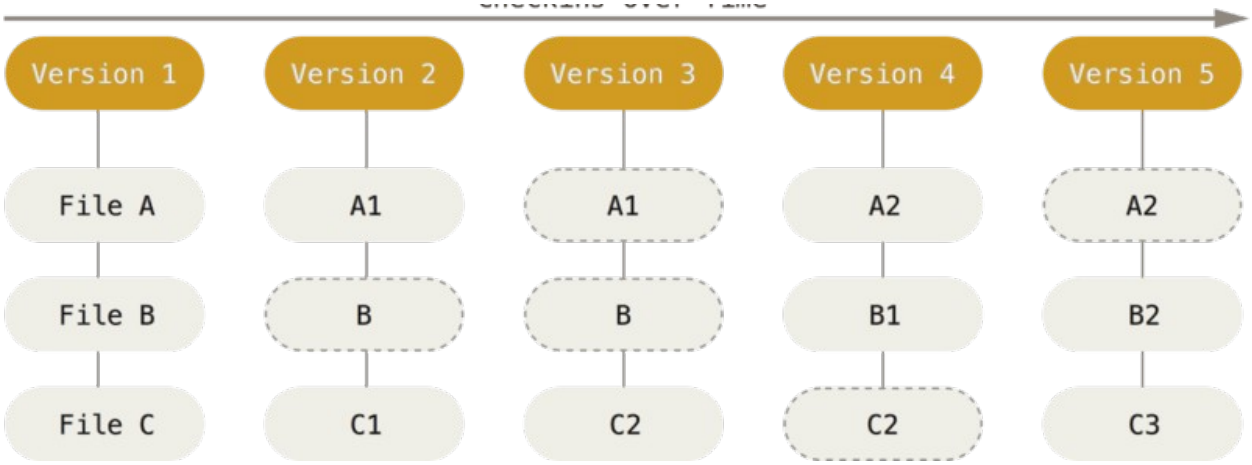
Git permet également de travailler sur des branches parallèles. Pratique pour faire évoluer son code.



Concepts

Historique des modifications

Avec git, vous pouvez enregistrer les modifications que vous faîtes de votre travail.



Comment sont enregistrées les modifications ?

Les modifications ne sont pas enregistrées au fur et à mesure que vous modifier vos fichiers. Les modifications sont faites lorsque vous le décidez. Lorsque vous souhaitez enregistrer vos modifications, il faut :

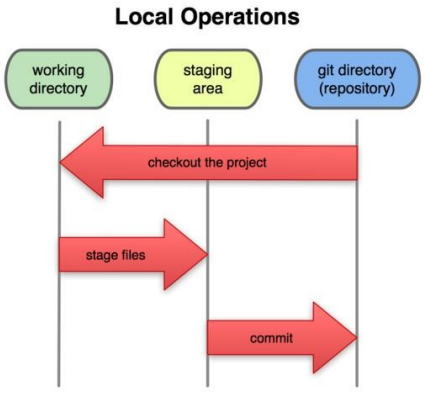
- d'abord **indexer (suivre)** les fichiers concernés
- ensuite, lorsque tous les fichiers ont été indexés, **valider**

Par analogie, l'enregistrement de l'état de vos fichiers est comme une photo de famille : chacun prend place, et lorsque tout le monde est prêt : la photo est prise

Les 5 Zones

Par conséquent,
les fichiers « existent » dans 3 « zones » différentes :

- Le répertoire de travail : **Working directory**
- La zone d'index : **Staging Area**
- Le répertoire (dépôt) git : **Git Directory**



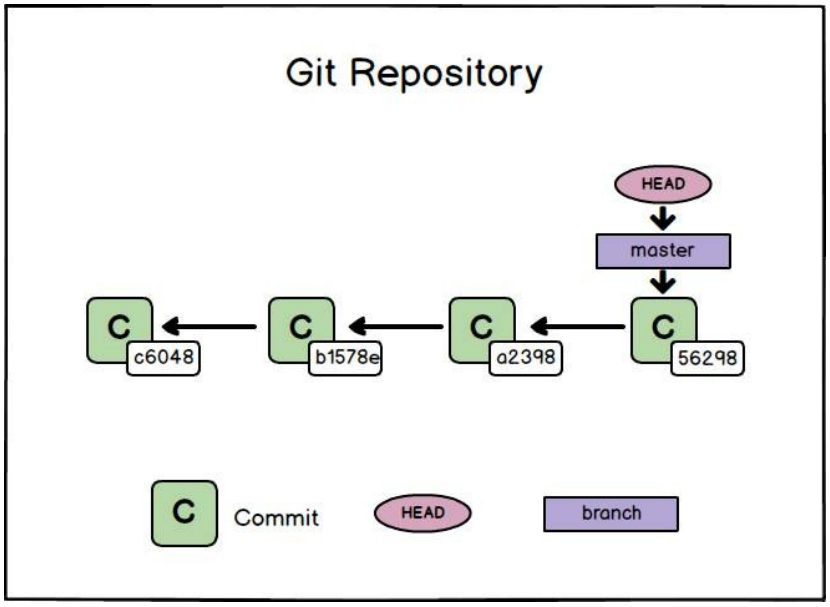
La première manipulation à connaître : placer les fichiers dans la zone d'index puis dans le répertoire git

A ces 3 zones, s'ajoute 2 autres :

- le(s) dépôts distants : **Remote repository**
- la remise : **Stash**

HEAD

La tête « HEAD » est un pointeur de la zone de commit. Il pointe en général sur le dernier commit de la branche principale du dépôt GIT.



Installation

Télécharger le soft : https://gitforwindows.org/ et l'exécuter.

Premières commandes

Les commandes qui vont suivre doivent être saisies dans un terminal (Gitbash, vscode, oh-my-zsh…). Il faudra d'abord vous placer dans le répertoire de travail.

Créer un nouveau dossier et ouvrir un terminal

git config

```
git config --global user.name "Mon nom"
git config --global user.email "mon@email.fr"
```

git init

Cette commande doit être exécutée dans votre dossier de travail. Elle initialise git. Concrètement : elle crée un dossier .git qui va enregistrer l'historique.

git status

Cette commande permet d'afficher l'état de votre application :

- Sur quelle branche vous êtes (master par défaut)
- Si les fichiers sont « suivis », et s'ils sont modifiés

git add

Lorsqu'on fait un git add … on demande à Git d'enregistrer dans le stage les fichiers qu'on souhaitera plus tard valider (intégration au commit). En d'autres termes, on ajoute, modifie, supprime des fichiers dans notre copie de travail et on signale à Git que tout ou partie de ces modifications est à mettre dans un statut intermédiaire (staged) avant d'être validé sous la forme d'un commit.

```
git add index.html
git add *.html
git add .
git add –all
```

alternative :

```
git stage
```

Ouvrir votre dossier dans VSCode et créer 2 fichiers : index.php et styles.css. Lancer les commandes git status, git add . Et de nouveau git status.

git commit

Cette commande permet de déplacer les fichiers depuis la zone « stage » vers la zone de «depot». Il faut mettre un message de commit.

```
git commit -m « message de commit »
```

```
[master (commit racine) 5b445ea] first commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 index.php
 create mode 100644 styles.css
```

En retour, on peut voir les 7 premiers caractères de l'identifiant du commit (Ici : 5b445ea), ainsi que d'autres informations

Comitter vos modifications, puis ajouter un fichier script.js et commiter les nouvelles modifications.

git log

Cette commande permet de lister tous les commits enregistrés.

```
git log --oneline
git log -p index.php
```

git tag

Cette commande permet d'ajouter une étiquette pour marquer un commit

Pour tagger le dernier commit

```
git tag v1.0
```

Pour tagger un commit particulier

```
git tag v1.0 <hash>
```

Pour obtenir la liste des tags

```
git tag --list
```

Softs

Git dans VSCode

Télécharger l’extension git history

GitKraken

https://www.gitkraken.com/

Git dans Sublime Text

https://www.sublimemerge.com/

Pour aller plus loin

S’entraîner

http://gitimmersion.fr/

Documentation

https://git-scm.com/book/fr/v2

https://ndpsoftware.com/git-cheatsheet.html#loc=index

Apprendre

https://delicious-insights.com/fr/articles/apprendre-git/

https://delicious-insights.com/fr/articles/git-zones/

https://openclassrooms.com/fr/courses/7162856-gerez-du-code-avec-git-et-github

https://grafikart.fr/formations/git

https://delicious-insights.com/fr/articles/git-stash/