




Projet dynamique

Git - Github

Le versioning



Introduction
Fonctionnement
Commandes de base

Principes de base



Application de gestion des versions

VCS en anglais pour Version Control System

Un 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.

Il crée des instantanés de l'évolution de votre projet que vous pouvez récupérer si vous constatez des dysfonctionnements ou des erreurs majeures.

Ne confondez pas **Git** et **GitHub**:

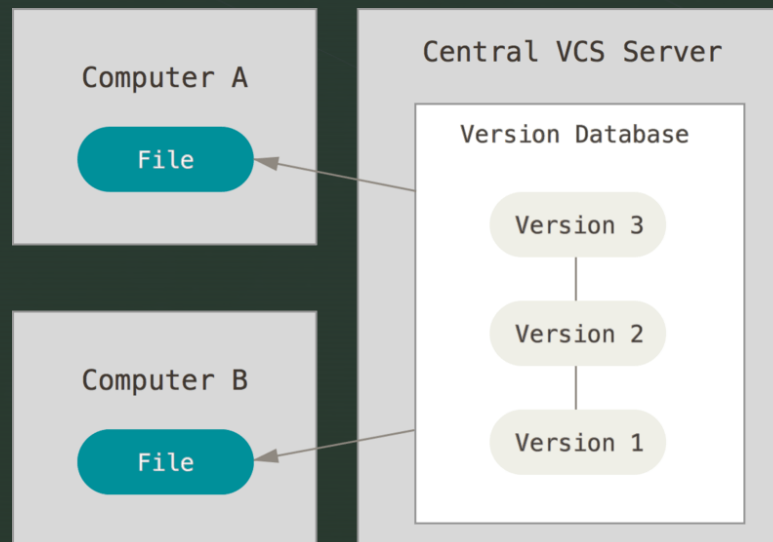
Git est l'application de versioning et GitHub qui un des services de dépôt en ligne.

Avantages

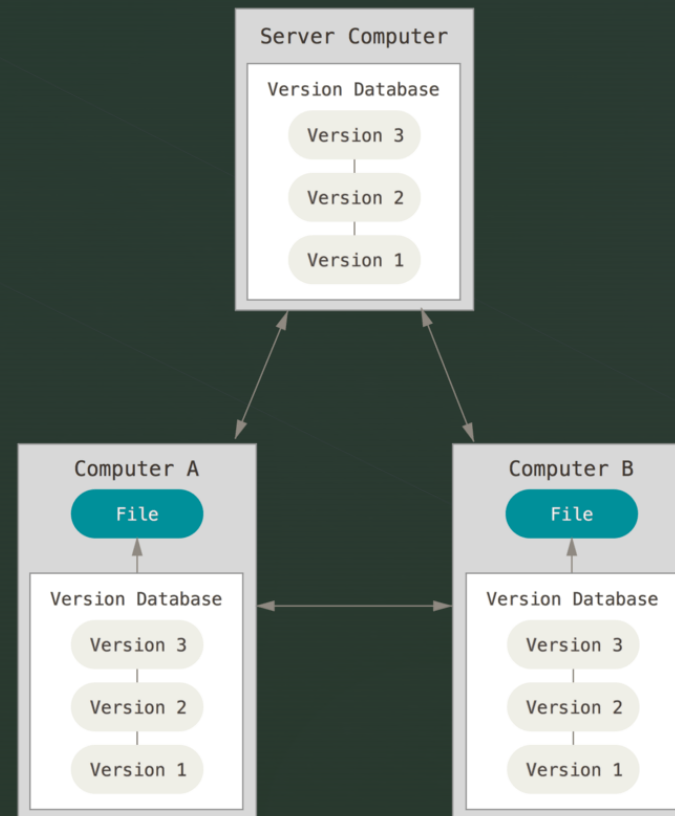
- De ce type d'application, on peut retirer trois grandes fonctionnalités
 - travailler à plusieurs sans risquer de supprimer les modifications des autres collaborateurs;
 - revenir en arrière en cas de problème;
 - suivre l'évolution étape par étape d'un code source pour retenir les modifications effectuées sur chaque fichier.

Logiciel centralisé ou distribué ?

- Il existe deux types principaux de logiciels de gestion de versions.
- Les logiciels centralisés : un serveur conserve les anciennes versions des fichiers et les développeurs s'y connectent pour prendre connaissance des fichiers qui ont été modifiés par d'autres personnes et pour y envoyer leurs modifications.

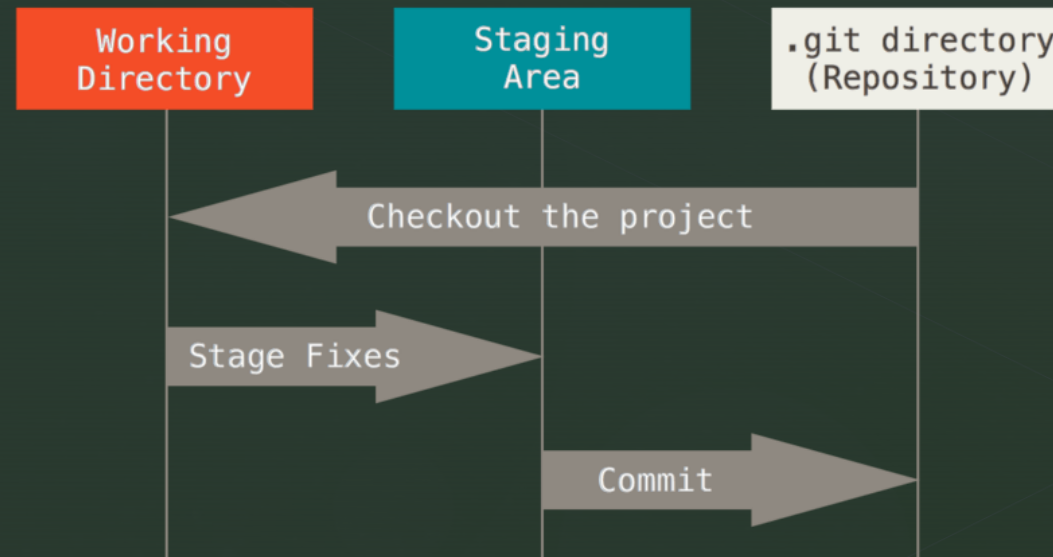


- Les systèmes de gestion de versions distribués
- les clients n'extraient plus seulement la dernière version d'un fichier, mais ils dupliquent complètement le dépôt. Ainsi, si le serveur disparaît et si les systèmes collaboraient via ce serveur, n'importe quel dépôt d'un des clients peut être copié sur le serveur pour le restaurer. Chaque extraction devient une sauvegarde complète de toutes les données.

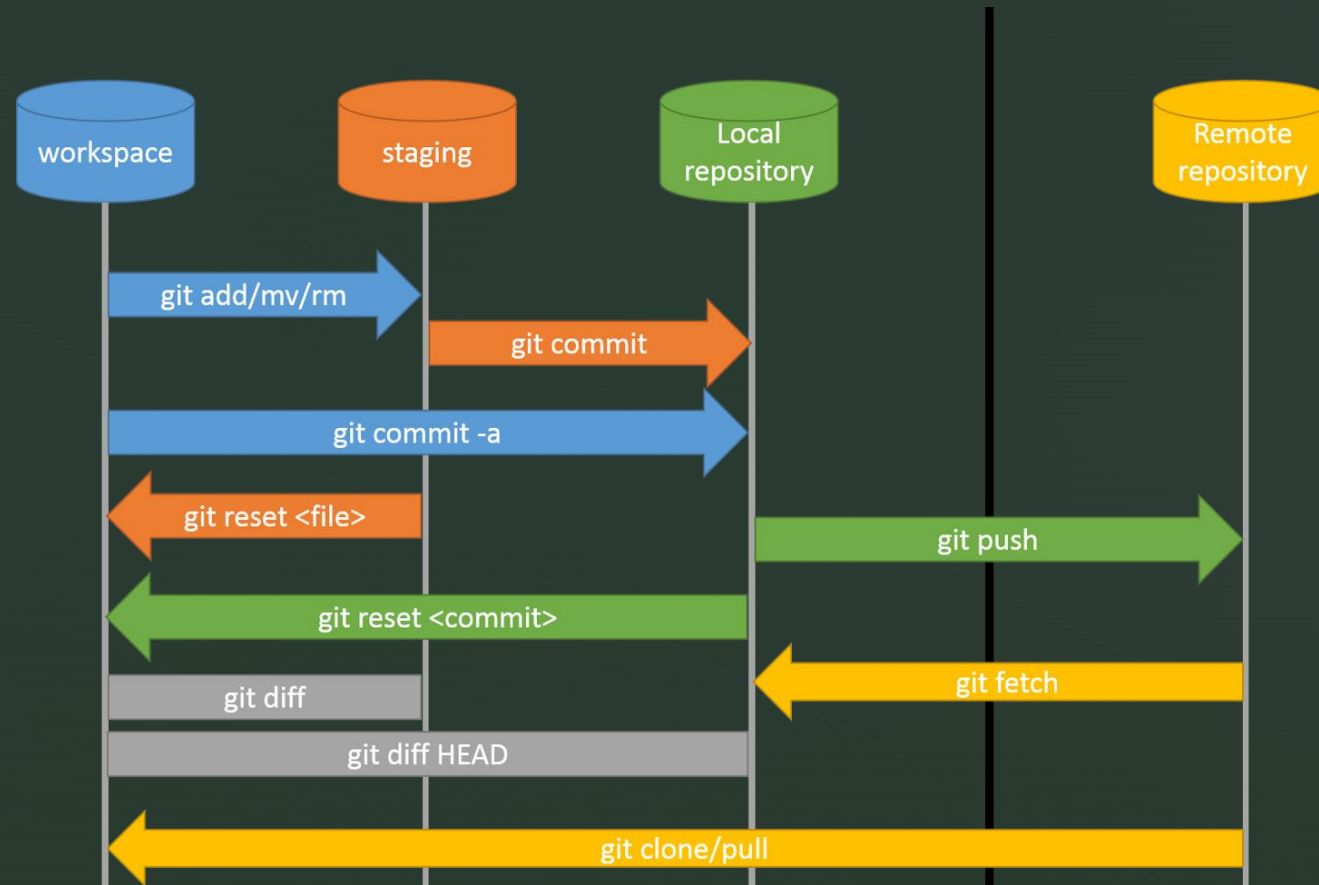


Fonctionnement général

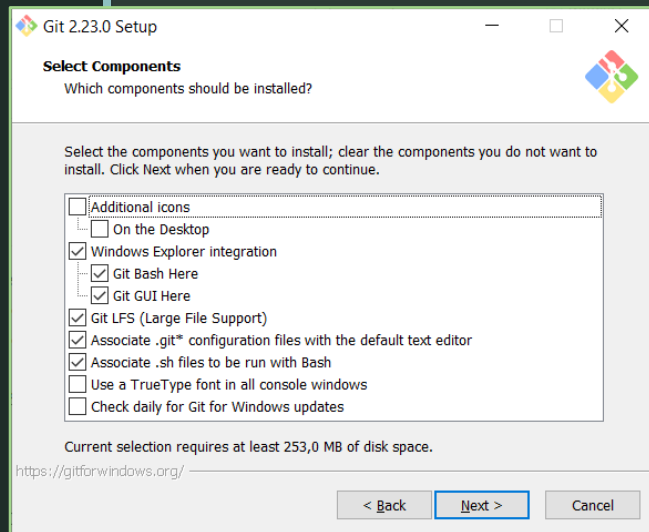
- Presque toutes les opérations sont locales et comme vous disposez de l'historique complet du projet sur votre disque dur, la plupart des opérations semblent instantanées.
- L'utilisation standard de Git se passe comme suit :
 - vous modifiez des fichiers dans votre répertoire de travail;
 - vous indexez les fichiers modifiés, ce qui ajoute des instantanés de ces fichiers dans la zone d'index;
 - vous validez, ce qui a pour effet de basculer les instantanés des fichiers de l'index dans la base de données du répertoire Git.



Fonctionnement via les commandes



Installation sous Windows



- Rendez-vous sur le site officielle pour télécharger la dernière version: <https://git-scm.com/download/win>
- Sélectionnez « Git Bash » et « Git Gui »
- Laissez les autres options par défaut.
- Testez le bon fonctionnement dans **Git Bash** avec la commande: **git --help**

Paramétrage à la première utilisation

- Une fois Git installé sur votre système, vous devez personnaliser votre environnement Git. Vous ne devriez avoir à réaliser ces réglages qu'une seule fois ; ils persisteront lors des mises à jour. Vous pouvez aussi les changer à tout instant en relançant les mêmes commandes.
- Git contient un outil appelé `git config` pour vous permettre de voir et modifier les variables de configuration qui contrôlent tous les aspects de l'apparence et du comportement de Git.

Git config

- La première chose à faire après l'installation de Git est de renseigner votre nom et votre adresse email dans deux variables.
- `git config --global user.name "John Doe"`
- `git config --global user.email johndoe@example.com`
- Vérifiez les données avec: `git config --list`

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.

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 — un squelette de dépôt Git. Pour l'instant, aucun fichier n'est encore versionné.

Démarrer un dépôt Git

Cloner un dépôt existant

- Si vous souhaitez obtenir une copie d'un dépôt Git existant la commande dont vous avez besoin s'appelle `git clone [url]`.
- Ceci crée un répertoire portant le même nom que le dépôt, initialise un répertoire `.git` à l'intérieur, récupère toutes les données de ce dépôt, et extrait une copie de travail de la dernière version.

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 `git init`, vous devriez voir ce qui suit :

On branch master

No commits yet

Untracked files:

(use "git add <file>..." to include in what will be committed)


.idea/

nothing added to commit but untracked files present (use "git add" to track)

Placer de nouveaux fichiers sous suivi de version

- Commençons par créer un nouveau fichier `readme.md` dans la racine de notre projet. Si PhpStorm demande d'indexer le fichier refusez, nous procéderons manuellement pour une meilleure compréhension.
- Ecrivez pour l'exemple un simple texte et exécutez la commande `git status`
 - On branch master
 - No commits yet
 - Untracked files:
 - `.idea/`
 - `readme.md`

Readme.md apparaît dans la liste des fichiers non suivis

- 
- Pour commencer à suivre un nouveau fichier, vous utiliserez la commande `git add readme.md`
 - Avec `git status`, vous constaterez que le fichier n'est plus dans la zone non suivie "Untracked files" mais bien dans la zone "Modifications qui seront validées"

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
new file: readme.md

- La commande `git add --all` permet d'indexer l'ensemble des fichiers.

Désindexer un fichier

- Si vous souhaitez retirer un fichier ou un dossier de la « staging area » utilisez la commande `git reset <fichier>` ou `git reset --all`

Modifier un fichier indexé

- Si vous modifiez le contenu d'un fichier déjà indexé et que vous saisissez `git status`, vous constaterez que des modifications ont été effectuées sur le fichier:

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git restore <file>..." to discard changes in working directory)

modified: `readme.md`

- Vous devez alors le réindexer avec la commande `git add`
- Si vous souhaitez annuler la dernière modification depuis son indexation utilisez la commande `git restore <file>`

Ignorer des fichiers

- Il arrive souvent que certains fichiers présent dans la copie de travail ne doivent pas être ajoutés automatiquement ou même ne doivent pas apparaître comme fichiers potentiels pour le suivi de version. Ce sont par exemple des fichiers générés automatiquement tels que les fichiers de logs ou de sauvegardes produits par l'outil que vous utilisez. Dans ce cas, on peut indiquer leurs noms ou définir un modèle (patterns) dans le fichier `.gitignore`.

Exemples de .gitignore

- Créez un fichier `.gitignore` dans votre projet
- Ajoutez y les fichiers à ignorer sous la forme de fichiers ou de patrons.
- Exemples:
 - `*.txt`
 - `readme.md`
 - `tmp/` (ignore tous les fichiers dans le répertoire)
 - `doc/**/*.txt` (ignorer tous les fichiers `.txt` sous le répertoire `doc/`)
 - `.idea`
 - `.gitignore`

Testez le .gitignore

- Avant la création du .gitignore, si vous lancez un git status vous obtiendrez dans la liste des fichiers non suivis le dossier système de PhpStorm: `.idea/`
- Vous devez absolument l'ajouter au `.gitignore` ça ne concerne pas votre projet.
- Après la création du gitignore ils n'apparaîtront plus même dans la zone des fichiers non suivis.



Commit

Valider vos modifications



Les commits

- Nous allons ajouter quelques fichiers dans notre projet:
 - `css/style.css`
 - `index.php`
- Ajoutez les fichiers pour indexation: `git add -all`
- 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.
- Saisissez `git commit -m « Votre message »`
- Si vous vérifiez avec `git status`, vous verrez le message suivant:

`On branch master`

`nothing to commit, working tree clean`

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.
- Par défaut, `git log` invoqué sans argument énumère en ordre chronologique inversé les commits réalisés. Cela signifie que les commits les plus récents apparaissent en premier. Comme vous le remarquerez, cette commande indique chaque commit les contrôles SHA-1, le nom et l'e-mail de l'auteur, la date et le message du commit.

git log

- commit 2984efa20c155bf567551c62714d8f9c6df01722 (HEAD -> master)
- Author: John Doe <johndoe@example.com>
- Date: Wed Oct 23 21:35:32 2019 +0200
- premier commit

git log

- Un commit correspond à une étape du travail. Evitez de faire de commits à tout bout de champs sinon l'historique devient ingérable.
- Souvenez-vous que la validation enregistre l'instantané que vous avez préparé dans la zone d'index. Tout ce que vous n'avez pas indexé est toujours en état modifié. vous pouvez réaliser une nouvelle validation pour l'ajouter à l'historique.
- Vous pourrez revenir facilement sur un instantané précédent si vous avez des problèmes.

Git log et les options

- git log dispose d'un très grand nombre d'options permettant de paramétrer exactement ce que l'on cherche à voir.
- git log -p (montre les différences introduites entre chaque validation)
- git log -2 (limite la sortie de la commande aux deux entrées les plus récentes)
- git log --stat (affiche un résumé des modifications fichier par fichier)

Dépôt distant GitLab

- Après la création du compte vous devez créer un dépôt distant avec la commande: `git remote add`
- Dans mon cas:
- `git remote add origin https://gitlab.com/Teacher01/learning-template.git`
- Origin représente par défaut le nom du dépôt distant. Vous pouvez le remplacer.
- Ensuite, vous devez transférer votre dossier de travail et votre historique avec: `git push -u origin master`

Cloner un dépôt existant

- Cloner un dépôt existant consiste à récupérer tout l'historique et tous les codes source d'un projet avec Git.
- Vous devez posséder l'url du dépôt et les accès s'il s'agit d'un dépôt privé.
- Pour cloner le projet: `git clone`
<https://gitlab.com/Teacher01/learning-template.git>
- Un dossier « learning-template » est créé et tous les fichiers source du projet ainsi que l'historique de chacune de leurs modifications sont ajoutés.