Introduction à Git

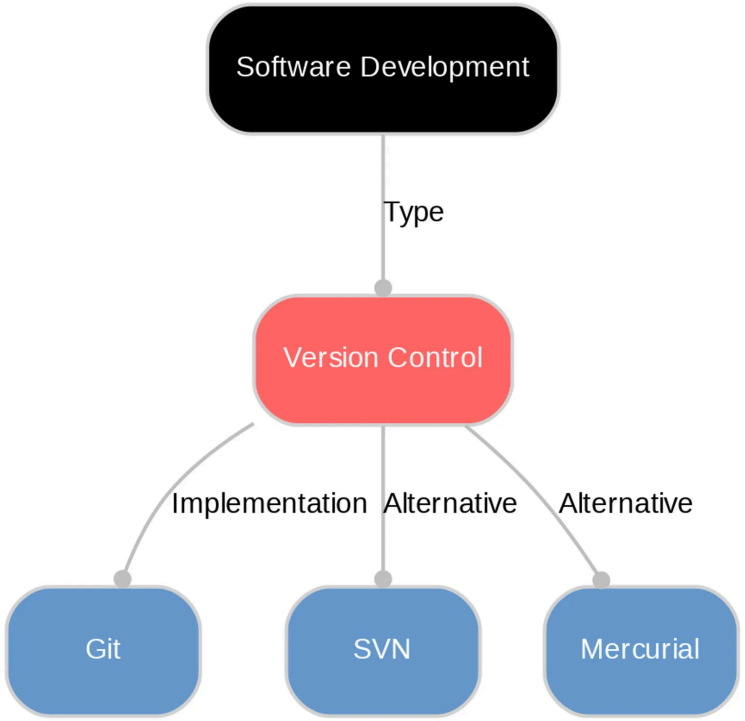
Le Contrôle de Version Simplifié

Pour les étudiants de 1ère année Supinfo

Au Programme

1. Qu'est-ce qu'un VCS ?
2. Git : Le VCS Distribué
3. Les 3 États Clés d'un Fichier
4. Commandes de Base
5. Branches et Fusion (Merge)
6. GitHub : La Plateforme

Qu'est-ce qu'un VCS ?

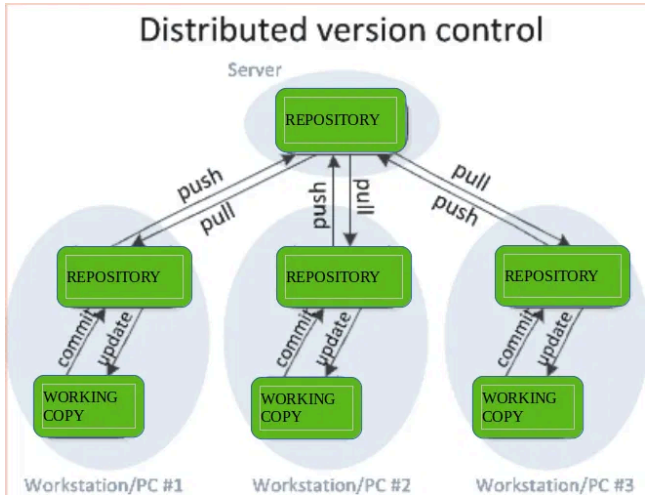


Un **Système de Contrôle de Version (VCS)** est un logiciel qui enregistre les modifications apportées à un ensemble de fichiers au fil du temps.

- ✓ **Historique**
  Revenir à une ancienne version du projet à tout moment.
- ✓ **Collaboration**
  Permettre à plusieurs personnes de travailler sur le même projet.
- ✓ **Sécurité**
  Sauvegarder le travail et éviter la perte de données.

Les Types de VCS

VCS Local	VCS Centralisé	VCS Distribué
Historique sur votre machine	Serveur central stocke l'historique	Chaque développeur a une copie complète
Ex: RCS	Ex: SVN	Ex: Git
Risque: Perte totale si disque dur lâche	**Risque:** Point de défaillance unique	**Avantage:** Pas de point de défaillance



Git : Le VCS Distribué

CRÉATEUR & CONTEXTE

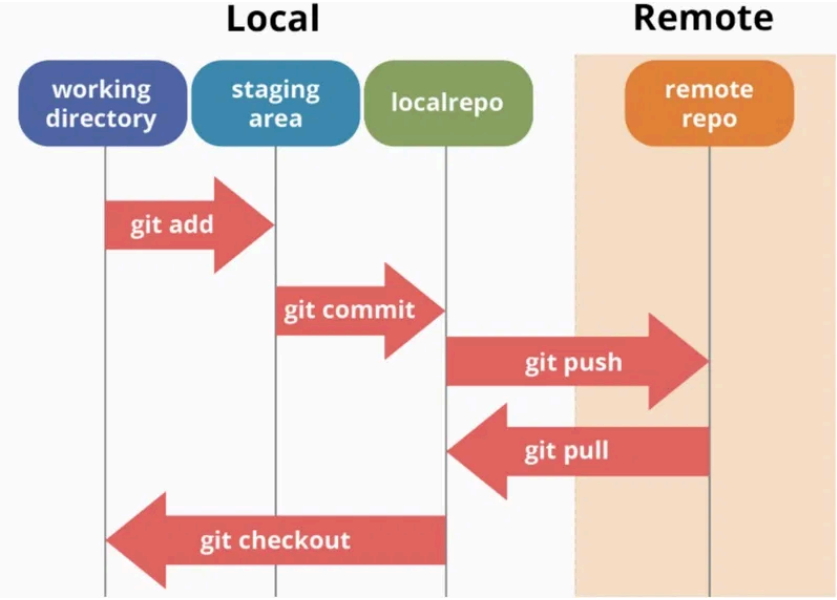
Créé par **Linus Torvalds** en **2005** pour le développement du noyau Linux.

> ### PRINCIPE CLÉ
>
> Git ne stocke pas les **différences**, mais des **instantanés (snapshots)** de l'ensemble du projet à chaque validation.

AVANTAGE PRINCIPAL

Rapidité et **robustesse**. L'historique complet est présent partout.

Les 3 États Clés d'un Fichier



1 Working Directory

Fichiers modifiés mais non suivis par Git.

2 Staging Area

Fichiers marqués pour être inclus dans le prochain commit (préparation).

3 Git Directory

Fichiers enregistrés de manière permanente après un commit.

> **Règle Simple :** Pour sauvegarder un changement, il doit passer de **Modifié** → **Staged** → **Committed**.

Commandes de Base (Démarrer)

`git init`

Initialise un nouveau dépôt Git dans le dossier courant. C'est la première étape pour commencer à suivre un projet.

EXEMPLE

```
$ git init
```

`git clone`

Télécharge un projet Git existant (un dépôt distant) sur votre machine locale.

EXEMPLE

```
$ git clone [url]
```

`git status`

Affiche l'état des fichiers (modifiés, staged, non suivis). Votre meilleur ami pour vérifier avant de commiter !

EXEMPLE

```
$ git status
```

Commandes de Base (Sauvegarder)

`git add [fichier] ou git add .`

Ajoute les modifications du fichier à la Zone de Staging (préparation pour le commit).

Utilité : Marquer les fichiers modifiés comme prêts à être enregistrés.

`git commit -m "Message"`

Enregistre l'instantané (snapshot) des fichiers staged dans l'historique Git.

Important : Le message doit être clair et explicite sur ce qui a changé.

`git log`

Affiche l'historique complet des commits effectués dans le projet.

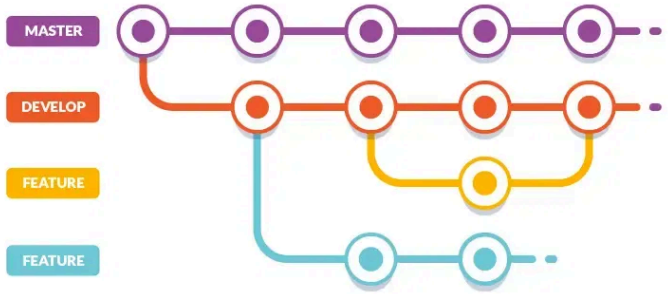
Utilité : Voir qui a fait quoi et quand, pour tracer l'évolution du projet.

Branches : Travailler en Parallèle

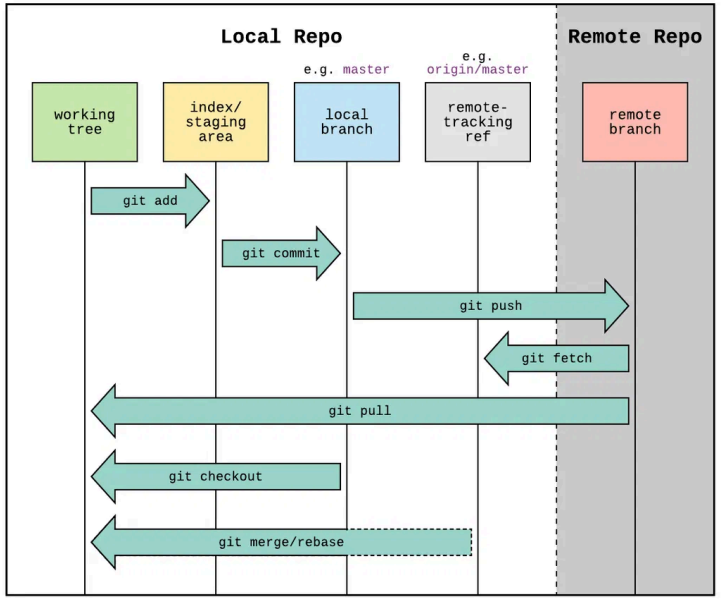
Une **branche** est un pointeur léger et mobile vers un commit. Elle permet de développer de **nouvelles fonctionnalités en parallèle** sans affecter la version principale.

Conventionnellement, la branche principale s'appelle **« main »** ou **« master »**.

- `git branch` : Liste les branches du projet.
- `git switch -c [nom]` : Crée et passe à une nouvelle branche.
- `git switch [nom]` : Passe à une branche existante.



Fusion (Merge) : Rassembler le Travail



CONCEPT

Intégrer les changements d'une branche de travail dans la branche principale (ou une autre branche).

ÉTAPES SIMPLES

1. `git switch main`
2. `git merge [nom_branche]`

> **⚠ Conflits**
>
> Si deux personnes modifient la même ligne dans le même fichier, Git ne peut pas choisir. Il faut résoudre le conflit manuellement avant de finaliser le merge.

GitHub : La Plateforme de Collaboration

Une plateforme web qui héberge des dépôts **Git** (dépôts distants) et facilite la collaboration entre développeurs.

- ✓ **Hébergement**
  Stockage de votre projet en ligne, accessible de partout.
- ✓ **Collaboration**
  Gestion des accès, des équipes, et du contrôle des permissions.
- ✓ **Pull Requests**
  Mécanisme pour proposer des changements et les faire réviser avant fusion.

GitHub : Synchronisation

`git push`

Envoie vos commits locaux vers le dépôt distant (GitHub).

UTILISATION

- Vous avez créé des commits localement
- Vous voulez les partager avec l'équipe
- Exécutez : **$ git push**

`git pull`

Récupère les derniers commits du dépôt distant et les intègre localement.

UTILISATION

- Vos collègues ont poussé des commits
- Vous voulez avoir la dernière version
- Exécutez : **$ git pull**

💡 **Règle d'Or :** Poussez souvent, tirez souvent ! Cela évite les conflits et garde tout le monde synchronisé.

Bonnes Pratiques pour Débuter

1 Commits Fréquents

Sauvegardez souvent votre travail. Un commit = une petite étape logique et cohérente.

Exemple : Un commit par fonctionnalité ou par bug fix, pas un mégacommit à la fin.

2 Messages Clairs

Écrivez des messages de commit qui expliquent **pourquoi** vous avez fait le changement, pas seulement **quoi**.

❌ *Mauvais : "fix bug" | ✓ Bon : "Corriger le calcul du total dans le panier"*

3 Une Branche par Fonctionnalité

Ne travaillez jamais directement sur **main**. Créez une branche pour chaque nouvelle tâche.

Exemple : `git switch -c feature/login`

4 Vérifier l'État Avant

Utilisez **git status** régulièrement pour vérifier l'état avant d'ajouter ou de commiter.

Cela vous évite d'ajouter accidentellement des fichiers non désirés.

Conclusion : Git est Indispensable

1. **Git est Distribué**

   Chaque développeur a une copie complète de l'historique. Aucun point de défaillance unique.

2. **3 États Clés**

   Working Directory → Staging Area → Git Directory. Le cycle de vie d'un fichier.

3. **Commandes de Base**

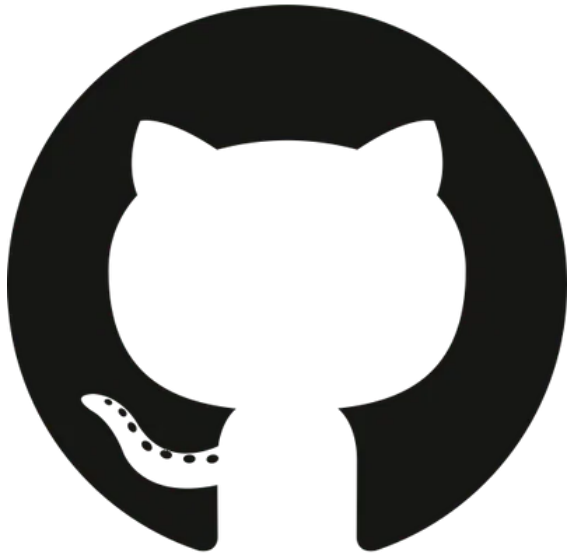
   init, clone, add, commit, push, pull. Les 6 commandes essentielles à maîtriser.

4. **Branches & Merge**

   Travailler en parallèle sans risque. Fusionner les changements de manière contrôlée.

5. **GitHub = Collaboration**

   La plateforme pour héberger, partager, et collaborer sur des projets Git.



Questions ?

N'hésitez pas à poser vos questions ! La compréhension de **Git** est essentielle pour votre carrière de développeur.

POUR APPROFONDIR

📚 **Pro Git Book** (gratuit en ligne)

🔗 **git-scm.com** - Documentation officielle

💻 **GitHub Learning Lab** - Tutoriels interactifs

🎓 **Supinfo Resources** - Matériel pédagogique

BESOIN D'AIDE ?

Consultez vos instructeurs ou les ressources Supinfo