Git

système de gestion de version

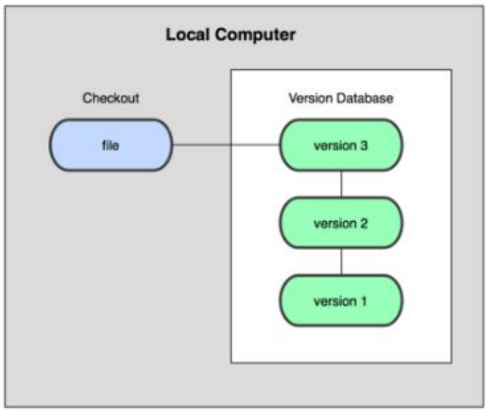
C’est quoi un système de gestion de version?

Système logiciel permettant de maintenir et gérer toutes les versions d’un ensemble de fichiers. Il permet de:

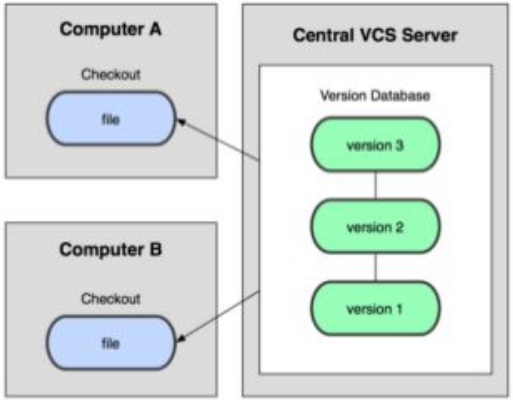
- Revenir aisément à une version précédente.
- Suivre l’évolution du projet au cours du temps.
- Travailler en parallèle sur des parties disjointes du projet et gérer les modifications concurrentes.
- Faciliter la détection et la correction d’erreurs.
- ....

Différents types de systèmes de gestion de version

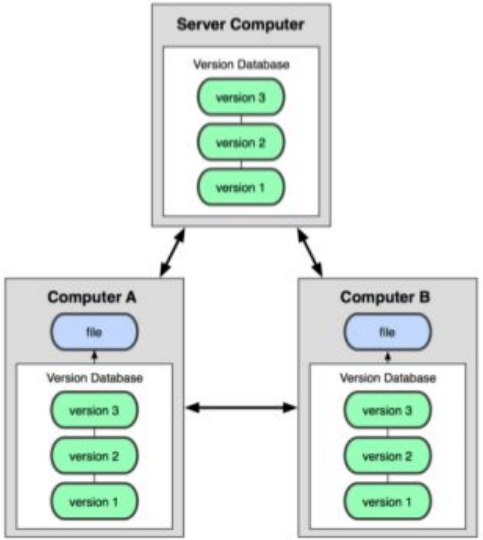
Local



Centralisé



Distribué



	Avantages	Inconvenients
Local	● Gestion et utilisation très simples.	● Est très sensible aux pannes. ● I Ne permet pas la collaboration.
centralisé	● Structurellement simple. ● Gestion et utilisation simples.	● Est très sensible aux pannes. ● Inadapté aux très grands projetset/ou avec une forte structure hiérarchique.
distribué	● Moins sensible aux pannes. ● Adapté aux très grands projets et/ou avec une forte structure hiérarchique	● Gestion et utilisation plus compliquées. ● Peut devenir très complexe structurellement.

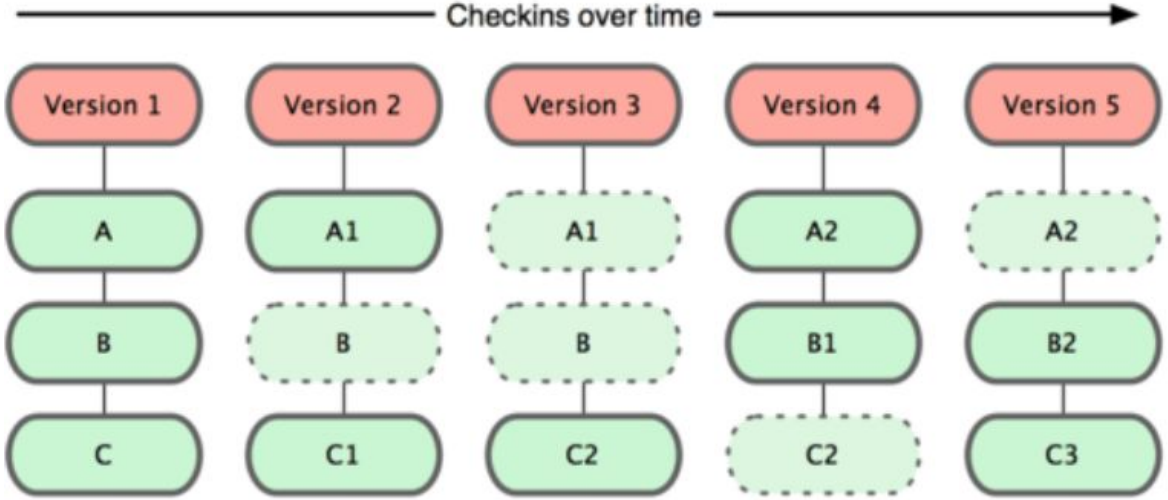
C’est quoi Git?

Système de gestion de version distribué (DVCS).

- De 1991 à 2002, le noyau Linux était développé sans utiliser de système de gestion de version.
- A partir de 2002, la communauté a commencé à utiliser BitKeeper, un DVCS propriétaire.
- En 2005, BitKeeper retire la possibilité d’utiliser gratuitement son produit. Linus Torvalds lance le développement de Git (Avril) et après seulement quelques mois de développement (juin), Git héberge le développement du noyau Linux.

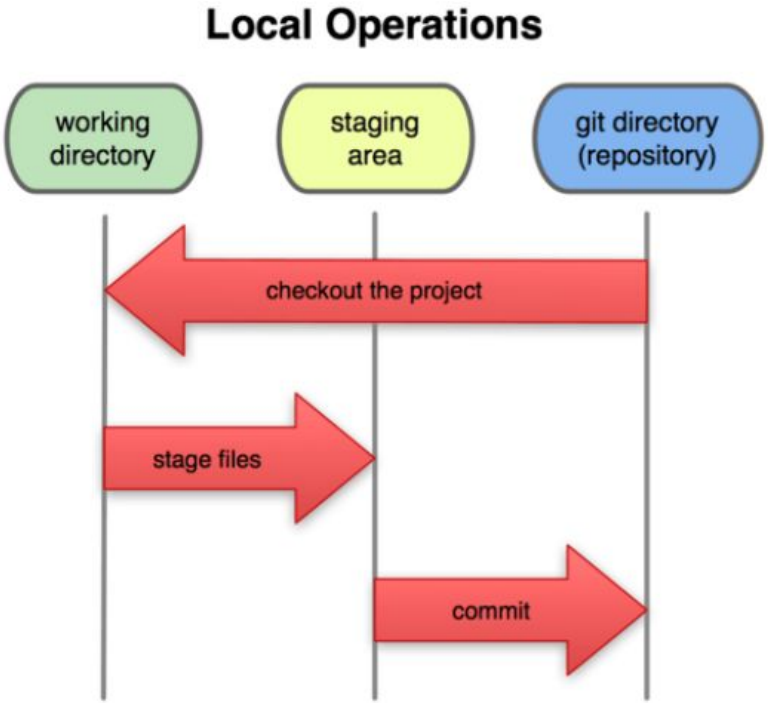
Git: principe de base

Un dépôt Git (git repository) est une sorte de système de fichiers (base de données), enregistrant les versions de fichiers d'un projet à des moments précis au cours du temps sous forme d'instantanés.



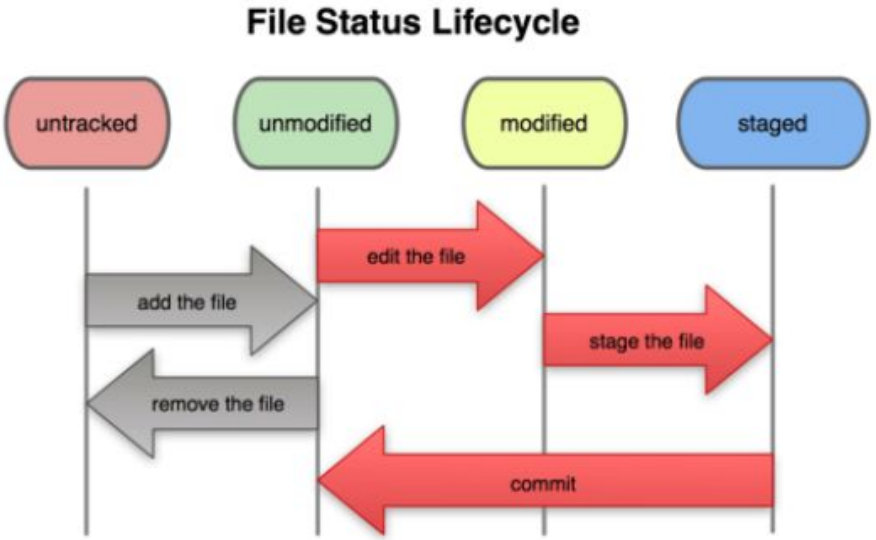
Git: principe de base

1. **Le répertoire Git/dépôt** :contient les méta-données et la base de données des objets du projet.
2. **Le répertoire de travail** : extraction unique d'une version du projet depuis la base de données du dépôt.
3. **La zone de transit/d'index** : simple fichier contenant des informations à propos de ce qui sera pris en compte lors de la prochaine soumission.

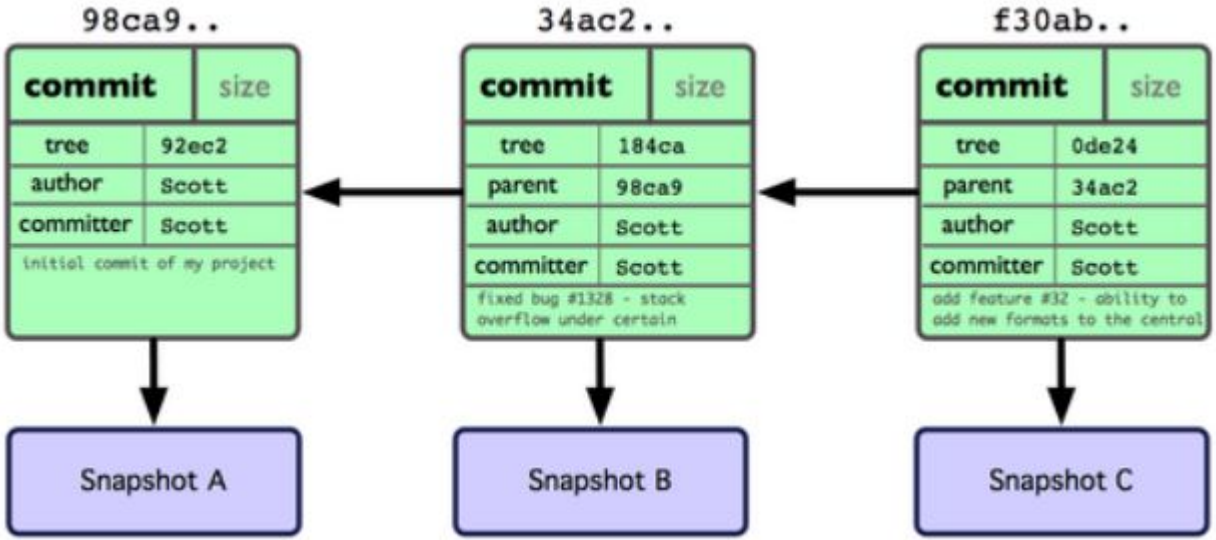


Git: principe de base

- **Non versionné** : fichier n'étant pas ou plus géré par Git ;
- **Non modifié** : fichier sauvegardé de manière sûre dans sa version courante dans la base de données du dépôt ;
- **Modifié** : fichier ayant subi des modifications depuis la dernière fois qu'il a été soumis ;
- **Indexé** (staged) : idem, sauf qu'il en sera pris un instantané dans sa version courante lors de la prochaine soumission (commit).



Git: principe de base



Git: commandes de base

Initialiser un dépôt

$ **git init**

Afficher l'état des fichiers du répertoire courant

$ **git status**

- Untracked files : fichiers non versionnés.
- Changes to be committed : modifications (ajout, suppression,changements) chargées en zone de transit (staging area), ou indexées.
- Changes not staged for commit : modifications n'ayant pas été chargées en zone de transit (ou indexées).

Git: commandes de base

- Indexer l'ajout ou les changements d'un fichier

$ **git add <fichier>**

- Indexer la suppression d'un fichier

$ **git rm <fichier>**

- Annuler l'indexation des modifications d'un fichier

$ **git reset <fichier>**

- Annuler les modifications non encore indexées d'un fichier

$ **git checkout [--] <fichier>**

- Déversionner un fichier

$ **git rm --cached <fichier>**

- Afficher le détail des modifications non indexées

$ git diff

Git: commandes de base

- Afficher le détail des modifications indexées

$ git diff ––staged

- Soumettre les modifications indexées en zone de transit

$ git commit

- Voir l’historique des soumissions

$ git log

Git: branches

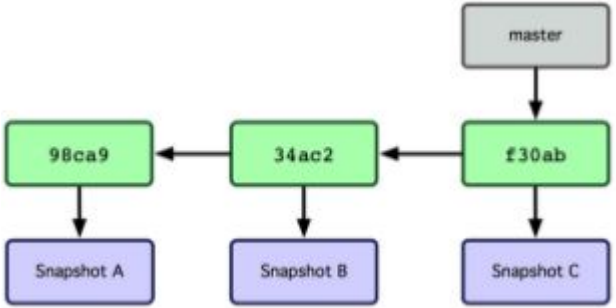
Une branche est une ligne d'évolution divergeant de la ligne d'évolution courante, celles-ci se poursuivant indépendamment l'une de l'autre. Une branche permet de:

- Pouvoir se lancer dans des évolutions ambitieuses en ayant toujours la capacité de revenir à une version stable que l'on peut continuer à maintenir indépendamment.
- Pouvoir tester différentes implémentations d'une même fonctionnalité de manière indépendante.

Git: branches

Git: branches

Une branche dans Git est tout simplement un pointeur vers un objet “commit”. Par défaut, il en existe une seule, nommée “master”

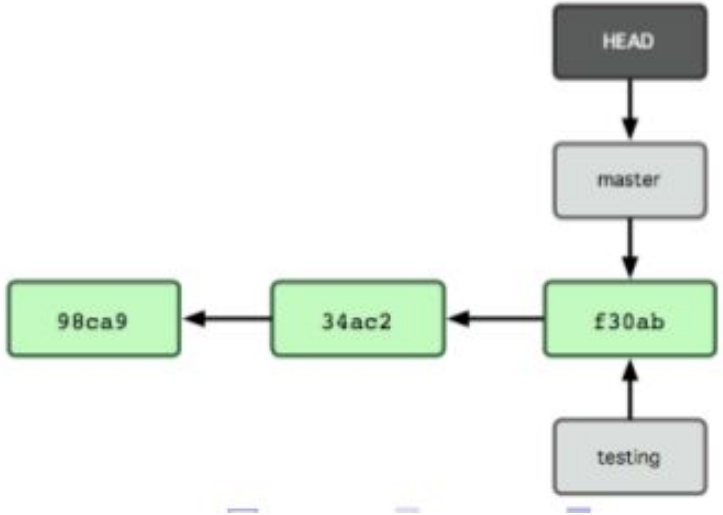


Créer une nouvelle branche

$ **git branch <branche>**

Git: branches

HEAD est un pointeur spécial vers la branche sur laquelle on travaille actuellement (extraite dans le répertoire de travail).



Git: branches

- Voir les branches du dépôt local

$ **git branch**

- Supprimer une branche

$ **git branch –d <branche>**

- Passer à une branche donnée (mise à jour de l'index et du répertoire de travail, ainsi que du pointeur HEAD)

$ **git checkout <branche>**