



01

Développeur Web & Web Mobile

GIT

DEVOLIE
agence de communication digitale

Qu'est-ce que GIT

02

git est Un logiciel de gestion de versions **décentralisé** il permet de créer et de gérer les différentes version de notre projet en permettant a tout moment de revenir sur des versions antérieure du projet, il facilite également le travail de groupe grâce à son système de branches.

Historiquement

03

Git un est logiciel libre et gratuit, créé en 2005 par Linus Torvalds.

Avant git plusieurs système de versioning ont été utilisé :

- Gestion manuelle : Renommage, dossiers séparés.
- SCCS (1972) : Local, fichier individuel.
- RCS (1982) : Diff, verrouillage.
- CVS (1990) : Centralisé, collaboration, fusion.
- Subversion (SVN, 2000) : Centralisé, gestion de dossiers, branches.
- BitKeeper (2000) : Distribué, décentralisation, influence Git.

Installation

04

l'installation de git se fait grâce au lien suivant :
<https://git-scm.com/>



05

Configuration de git

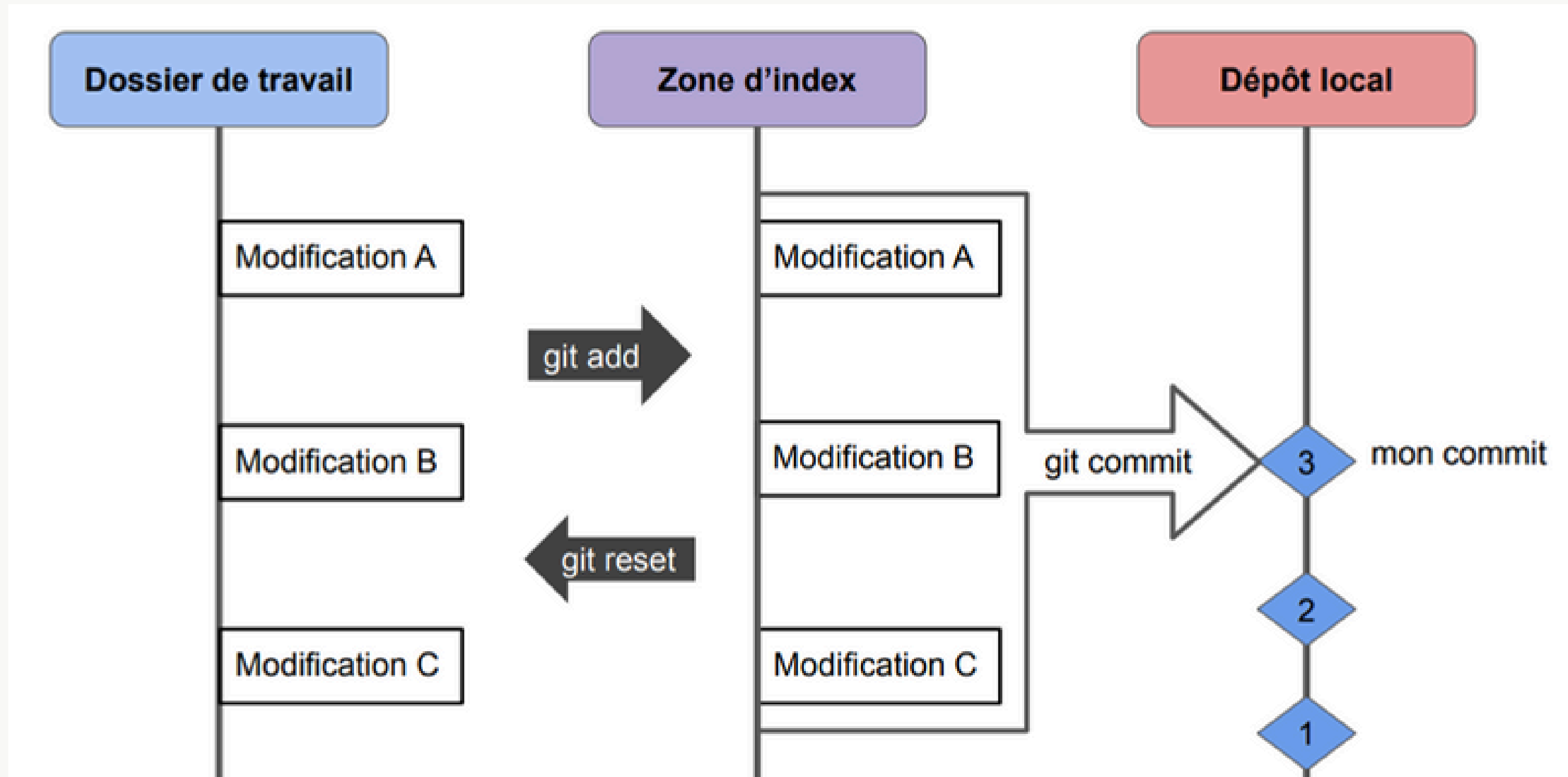
afin de pouvoir utiliser git pour ces projets il faut le configurer sur son poste de travail :

```
git config --global user.name "Ludovic Legros"  
git config --global user.email ludovic@devolie.fr  
git config --global --list
```

```
//configuration de votre nom  
//configuration de votre mail  
//voir la configuration en cours
```

Le fonctionnement

06



07

Initialisation de git

Une fois situé dans le projet en cours on peut initialiser git afin de créer un dossier caché qui contiendra l'historique de nos commits

```
git init
```

il est possible a tout moment de vérifier l'état des fichiers en cours de modifications

```
git status
```

Faire un commit

08

Afin d'effectuer un commit il faut avant tout indexer les fichiers, ne pas hésiter à effectuer des **git status** afin d'effectuer des vérifications.

```
git add index.html
```

```
git add .
```

```
git reset index.html
```

```
git diff
```

```
git commit -m"Mon premier commit"
```

//indexation du fichier index.html

//indexation de tout les fichiers modifié

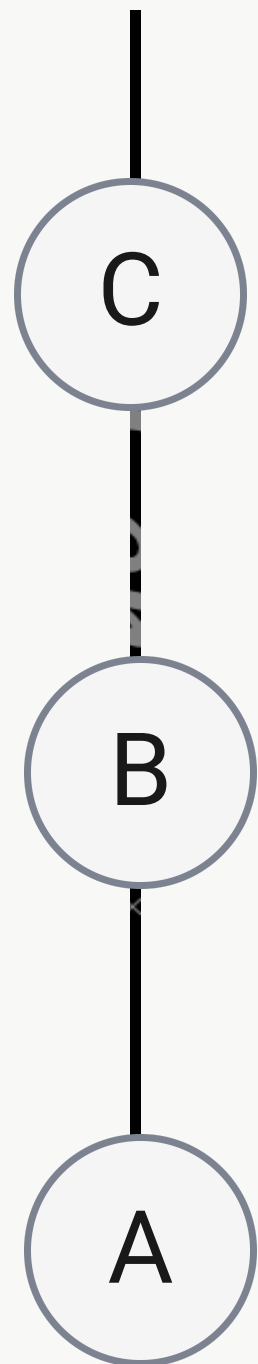
//retirer index.html de la zone d'indexation

//voir les modifications effectués dans les fichiers

//effectuer un commit avec un message

il faut **toujours** mettre un message lors d'un commit

Utilisation de l'historique



Git fonctionne avec un système de branche, nous sommes ici sur la branche principale appelé "master". Chaque commit de l'historique contient plusieurs informations :

- Un identifiant unique (SHA-1) (généré automatiquement)
- Les modifications effectuées
- Les commentaires
- Les infos sur l'auteur
- La date de création
- Liste (Sha-1) de ses parents

Utilisation de l'historique

Notion de tag

Le tag est un pointeur, il existe différent type de tag :

TAG de branche, le TAG de la branche principale est MASTER

TAG personnel qui reste sur les commits et par défaut un identifiant SHA-1

TAG Head qui permet de se déplacer dans l'historique

09

Utilisation de l'historique affichage de l'historique

Pour afficher la liste des commit qui ont été effectué sur la branche en cours :

```
git log  
git log -n 2  
git show le_sha_1_du_commit_voulu
```

```
//afficher la liste des commits  
//afficher les deux derniers commits  
//afficher le commit spécifié
```


09

Utilisation de l'historique navigation dans l'historique

la commande git checkout permet de se déplacer dans les différents commits de la branche.

```
git checkout le_sha_1_du_commit_voulu  
git tag patate  
git tag --delete patate
```

```
//se déplacer dans l'historique au commit voulu  
//changer le tag par défaut du commit  
//supprimer le tag du commit (pas le commit)
```

donner un tag personnalisé a un commit permet de revenir à ces versions plus facilement.

GitHub et Gitlab

10

Github et **Gitlab** sont des plateformes d'hébergement en ligne. Ils possèdent des fonctionnalités accès sur la gestion de dépôts et de projets

Certaines fonctionnalités de GitHub :

- les pull request, permettes de proposer des modifications a un projet
- Les gist permettent de partager des extraits de code ou de notes..
- Le fork permet de dupliquer le projet de quelqu'un dans votre gihub
- Faire une equipe pour l'organisation de projets

GitHub et Gitlab

Fichier .gitignore

10

il est possible de créer un fichier .gitignore dans notre projet qui va permettre d'ignorer certains fichiers lors de l'envoi sur notre dépôt distant.

Voici un exemple de fichier .gitignore :

.gitignore	//ignore le fichier .gitignore
tmp/*	//ignore tout les fichiers dans le dossier tmp
*.txt	//ignore tout les fichiers ayant l'extension txt
mon_fichier.jpg	//ignore le fichier mon_fichier.jpg

GitHub et Gitlab

Creation et configuration du dépôt GIT

10

La première étape consiste a créer un repository sur le GitHub

Une fois notre repository créé on peut choisir entre l'utilisation du **https** ou **ssh**, on utilisera ici le **https** et on copie la ligne fournis

Il est possible de rajouter un fichier gitignore et un fichier readme

GitHub et Gitlab

Envoie du projet sur le dépôt

10

La branche principale dans Github n'est plus master mais main, il convient donc de changer avant tout le tag de notre branche principale

```
git remote add origin lien\_du\_dépôt  
git branch -M main  
git push -u origin main
```

//permet de se connecter au repository
//force le changement du tag principal **master** en **main**
//envoie de son projet sur le dépôt distant

GitHub et Gitlab

Récupération d'un projet

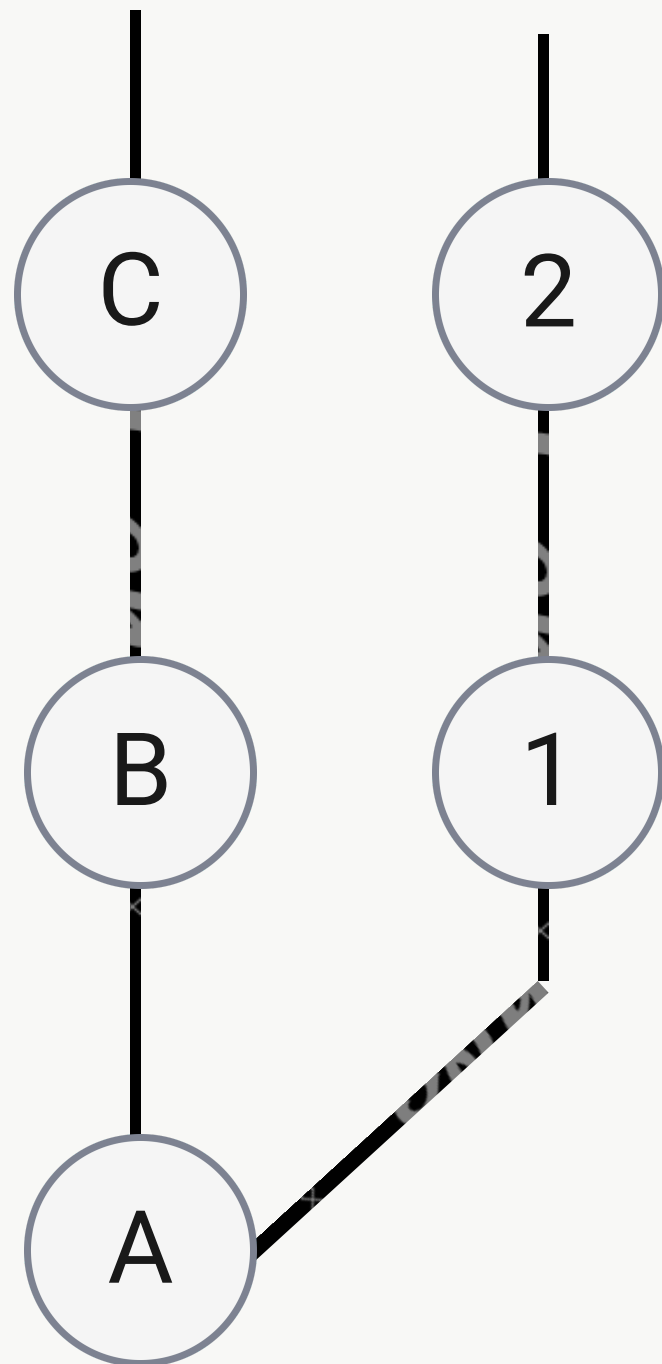
10

La récupération d'un projet nécessite de faire un clone du projet distant en local

```
git clone url_du_repository nom_du_dossier  
git pull origin main  
git branch -m master
```

```
//Clone le projet distant dans le dossier local choisi  
//récupère les changements distant dans son projet local  
//change le nom de la branche main (optionnel)
```


Création d'une nouvelle branche



L'utilisation de branche multiple permet de faciliter le travail de groupe , mais elle peut aussi permettre d'organiser notre code.

11

Création d'une nouvelle branche

```
git branch nom_de_branche  
git branch  
git checkout nom_de_branche
```

//créer une nouvelle branche
//voir les branches existantes
//se déplacer sur une autre branche

une fois des changements apporté a notre projet :

```
git add .  
git commit -m"message"  
git push --set-upstream origin nom_de_branche
```

//permet de push en créant la branche
sur le dépôt distant

11

Récupération d'une nouvelle branche

Récupération d'une branche sur le repository, qui n'est pas encore en local :

```
git pull  
git branch -a  
git checkout nom_de_branche
```

```
//récupérer le dépôt  
//voir les branches sur le dépôt  
//changement de branche
```


Merger deux branches

Afin de pouvoir merger deux branches, il faut en premier lieux se positionner sur la branche master

```
git merge nom_de_branche  
//choisir les modifications  
git add .  
git commit -m"merge de branches"  
git push
```

Bravo pour être resté jusqu'au bout

