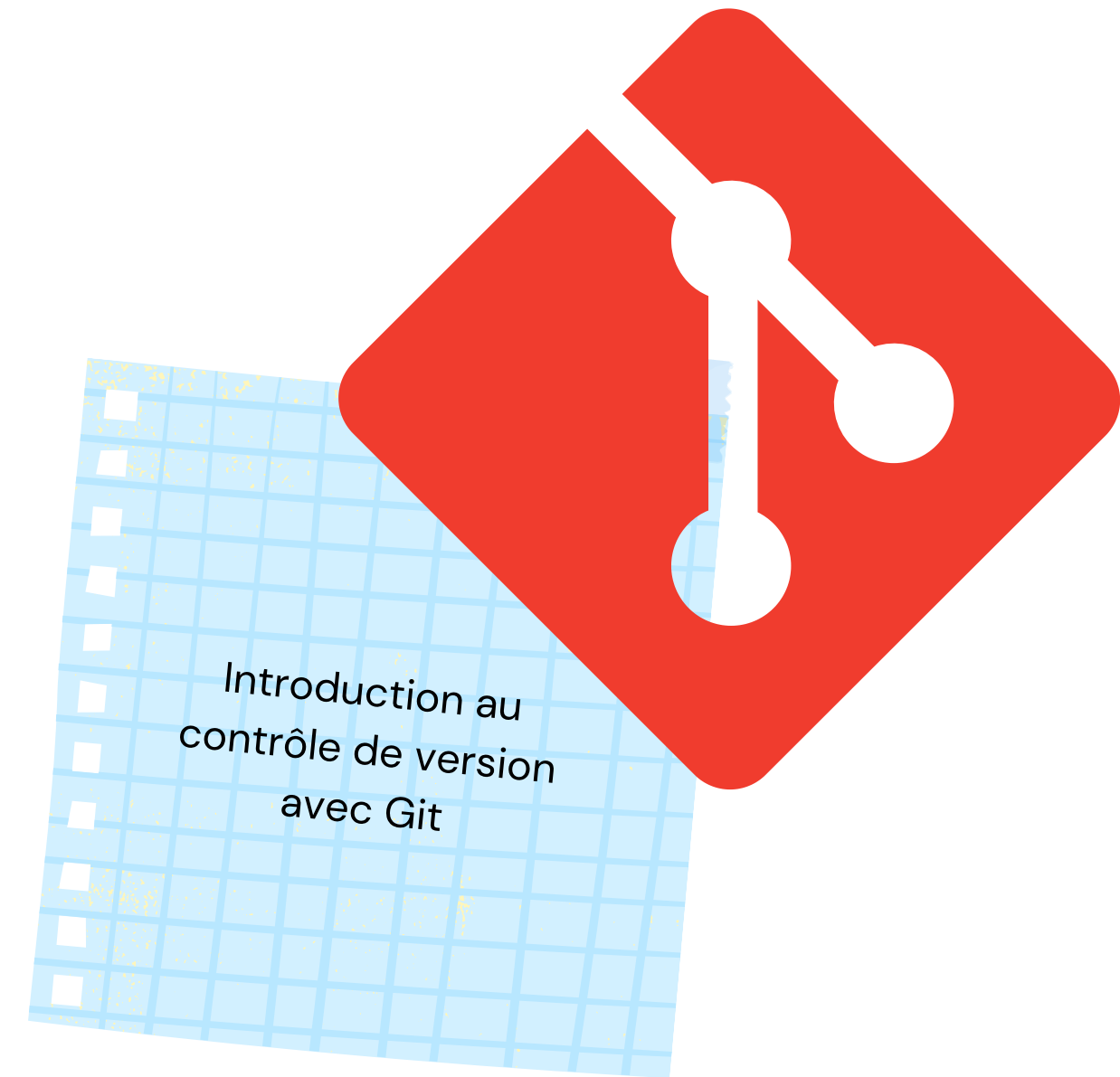


Les bases de Git, la gestion des branches, les fusions et les pull requests



Programme d'Aujourd'hui

1

Introduction à Git

2

Les bases de Git

3

Gestion des branches

4

Fusions

3

Pull Requests

4

simulations

Introduction

Qu'est-ce que Git ?

Système de contrôle de version distribué créé par Linus Torvalds en 2005.

C'est un outil qui permet à une équipe de bien collaborer à plusieurs sur son code.

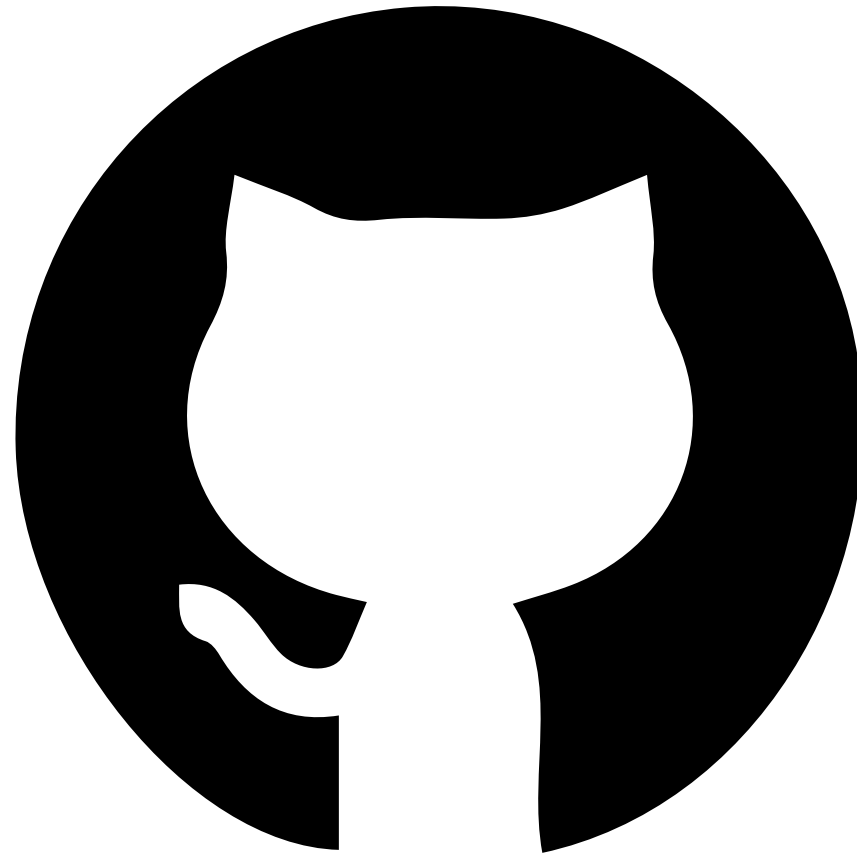
AVANTAGES

- Versionner son code
- Historique des modifications
- Travailler à plusieurs
- Sauvegarder son code à distance

INSTALLATION

- Sur Windows
<https://gitforwindows.org/>
- Sur Mac Déjà installé sinon
brew install git
- Sur Linux Déjà installé sinon
sudo apt update && sudo apt install git

Service hébergement répo git



GitHub



Gitlab

Ajouter une clé SSH

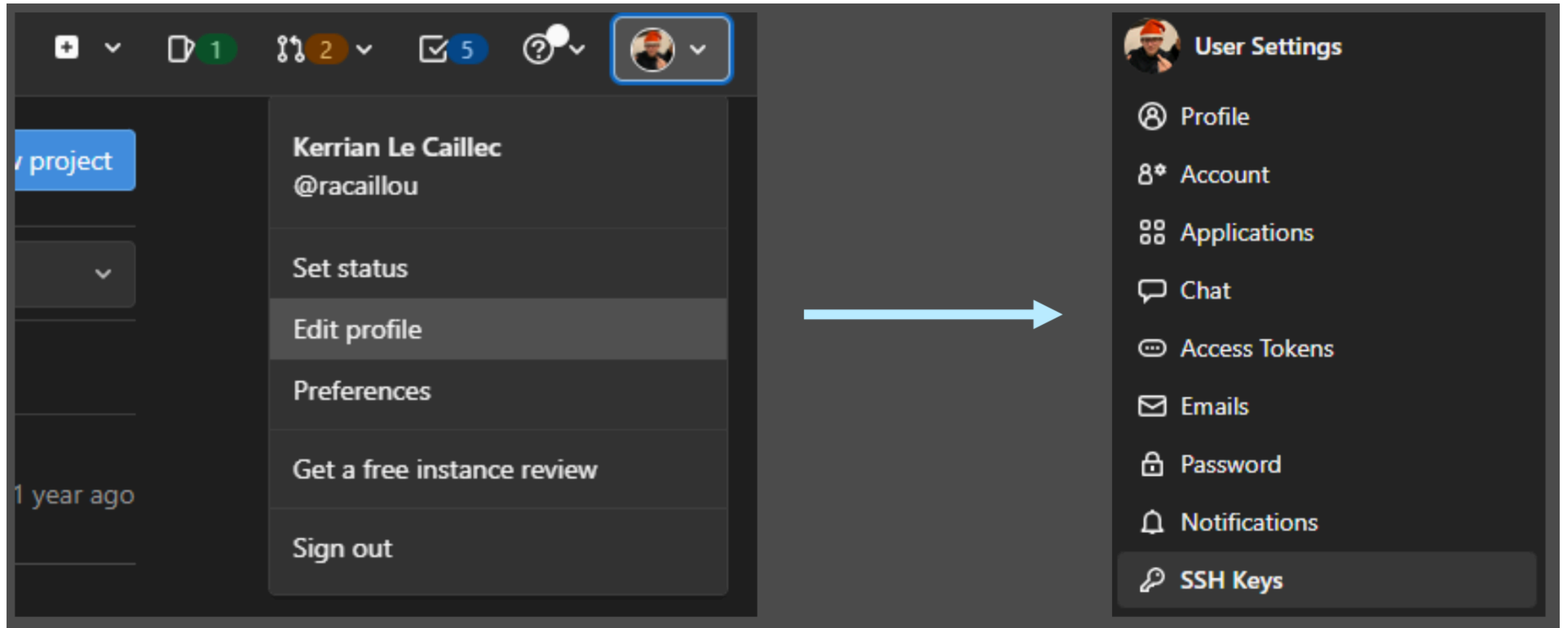
Les clés SSH permettent une authentification sans mot de passe en utilisant un couple de clés (publique et privée). La clé privée reste sur le client, tandis que la clé publique est placée sur le serveur.

Commande ---

- `ssh-keygen -o -a 256 -t ed25519`
- `cd`
- `cat .ssh/id_ed25519.pub`

Ajouter votre clé ssh

Ajoutez votre clé ssh sur l'espace dédié de votre service d'hébergement git.



Cloner un repository

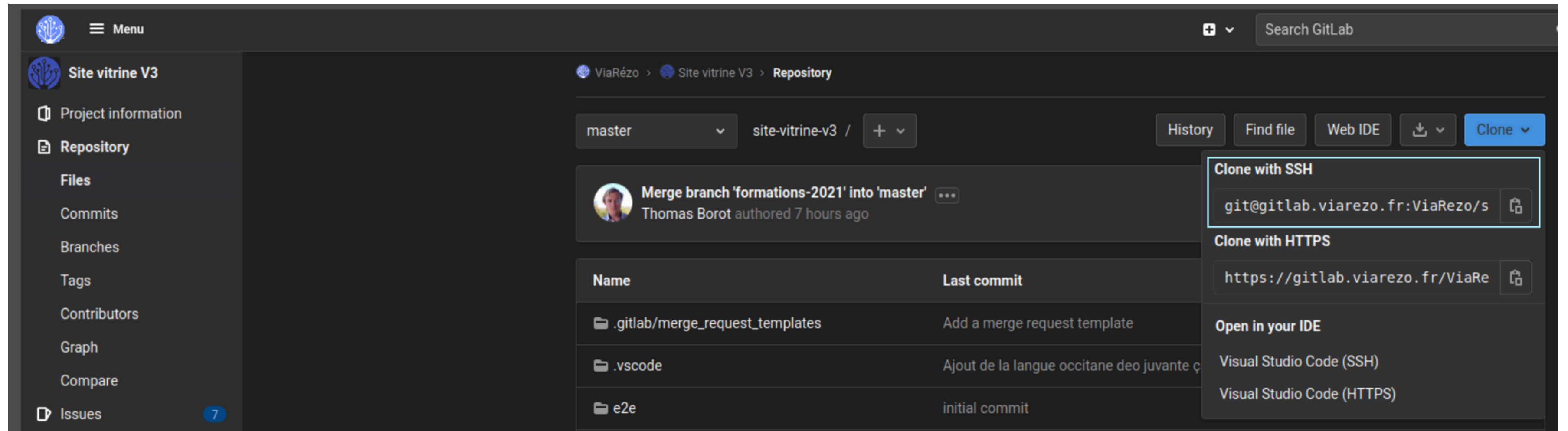
créer une copie locale d'un dépôt Git distant sur votre ordinateur. Cela vous permet de travailler sur les fichiers du projet en local et de synchroniser les changements avec le dépôt distant.

Commande

- `git clone <lien vers le répo>`

Lien vers le repos

Retrouvez lien vers le repository de votre projet dans votre fournisseur en cliquant sur l'onglet clone. (github)



The screenshot shows the GitLab interface for a repository named 'Site vitrine V3'. The left sidebar contains navigation links: Project information, Repository (selected), Files, Commits, Branches, Tags, Contributors, Graph, Compare, and Issues (with 7 new items). The main content area shows the repository path 'ViaRézo > Site vitrine V3 > Repository'. Below the path, there's a dropdown for the current branch 'master' and a button to add a new branch. To the right of these are buttons for 'History', 'Find file', 'Web IDE', a download icon, and a 'Clone' button. The 'Clone' button is open, showing three options: 'Clone with SSH' (git@gitlab.viarezo.fr:ViaRezo/s), 'Clone with HTTPS' (https://gitlab.viarezo.fr/ViaRe), and 'Open in your IDE' (Visual Studio Code (SSH) and Visual Studio Code (HTTPS)). Below the clone options, there's a table of files and their last commit.

Name	Last commit
.gitlab/merge_request_templates	Add a merge request template
.vscode	Ajout de la langue occitane deo juvante ç
e2e	initial commit

Commandes d'information

- `git status` // affiche un résumé de la situation
- `git log` //affiche un historique des "commit"
- `git log --graph` //affiche un historique sous forme de graphe
- `git log --stat` //affiche les détails des "commit"
- `git diff` //affiche les différences entre 2 "commit"
- `git fetch origin` // Récupérer des Informations sur le Dépôt Distant

Les branches (1)

Les branches dans Git sont une fonctionnalité essentielle pour gérer et organiser le développement de votre projet. Elles permettent de travailler sur différentes lignes de développement isolées, facilitant la gestion des nouvelles fonctionnalités, des corrections de bugs et des expérimentations.

Par défaut, Git crée une branche principale appelée main ou master.

Commande

- `git checkout <nom-branche>` change de branche
- `git branch <nom-branche>` crée une branche
- `git diff <nom-branche>` compare la branche courante avec une branche de référence
- `git checkout -b <nom-branche>` Créer et Changer de Branche Simultanément
- `git merge <nom-branche>` Fusionner une Branche dans la Branche Courante

Les branches(2)

Pourquoi Utiliser des Branches

- Isolation : Travailler sur des fonctionnalités ou des correctifs de manière isolée sans affecter la branche principale.
- Collaboration : Faciliter la collaboration entre plusieurs développeurs en permettant à chacun de travailler sur des branches distinctes.
- Expérimentation : Tester de nouvelles idées ou modifications sans risquer de compromettre la stabilité du code principal.

Les branches(3)

Meilleures Pratiques pour Utiliser les Branches

- Nommez vos branches de manière descriptive pour indiquer clairement leur but (par exemple, feature/login-system ou bugfix/issue-42).
- Créez des branches pour chaque fonctionnalité ou correctif afin de maintenir l'isolement et faciliter les revues de code.
- Fusionnez régulièrement avec la branche principale pour éviter de gros conflits de fusion.
- Utilisez les **pull requests** pour discuter et revoir les modifications avant de les fusionner dans la branche principale.

Le commit

Représente un instantané ou un point de sauvegarde du projet à un moment donné. Chaque commit enregistre l'état actuel des fichiers dans le repository, permettant ainsi de suivre les modifications au fil du temps et de revenir à des versions antérieures si nécessaire.

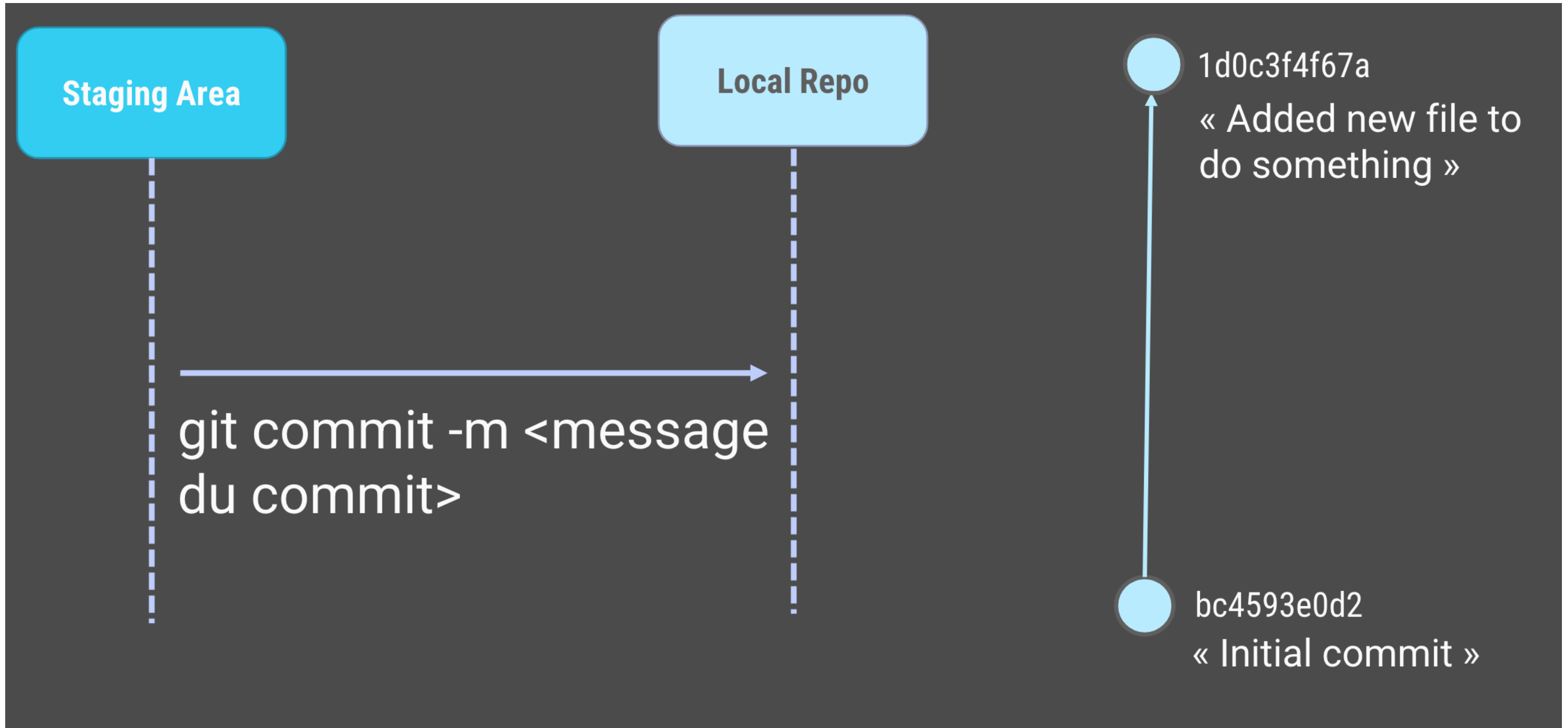
Un commit c'est :

- **L'unité de base de git**
- **Un ensemble de changement (diff)**
- **Une version du code**
- **1 commit = 1 fonctionnalité**

Commande ---

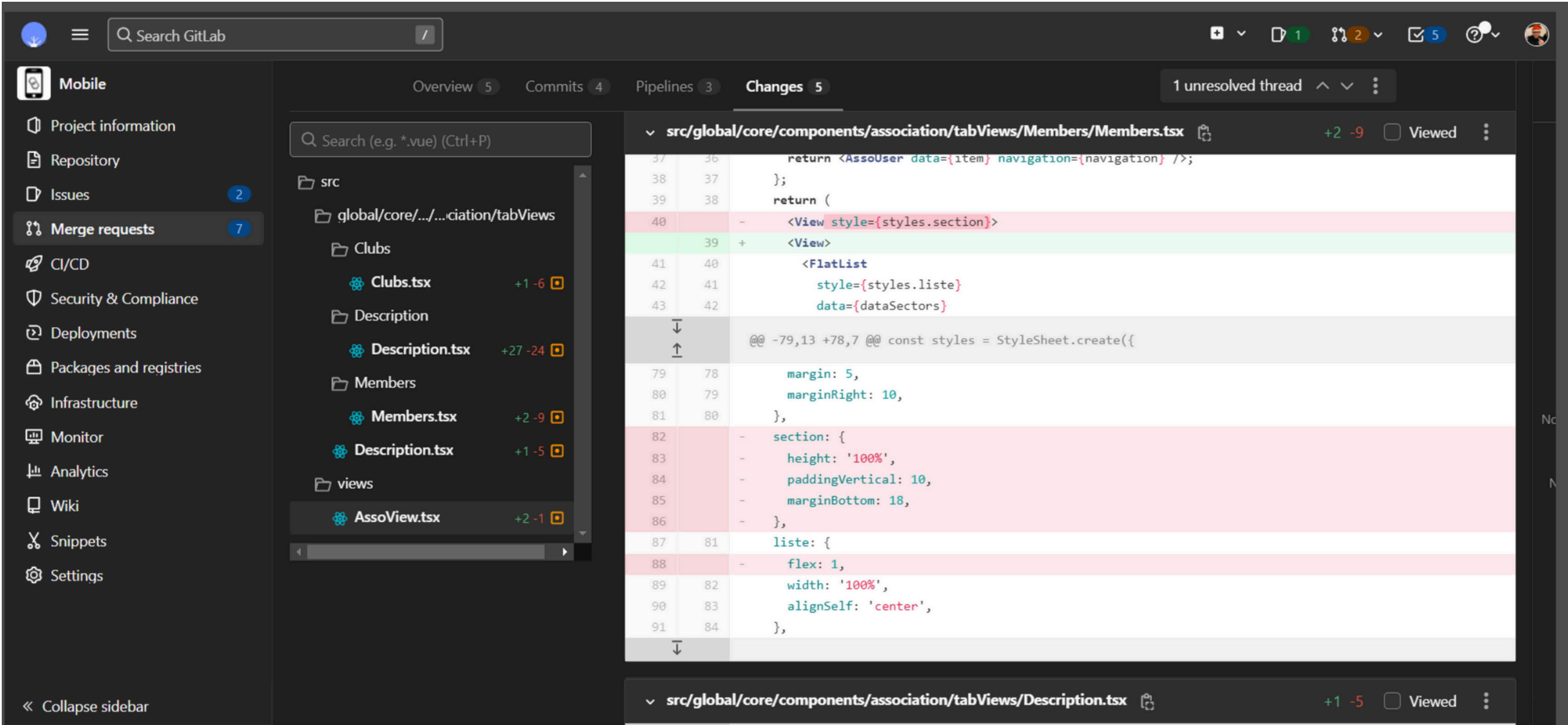
- `git commit -m <message du commit>`

Commande Commit



Simulation d'un diff

En rouge les éléments retirés et en vert les éléments rajoutés



Ajouter ou supprimer des fichiers

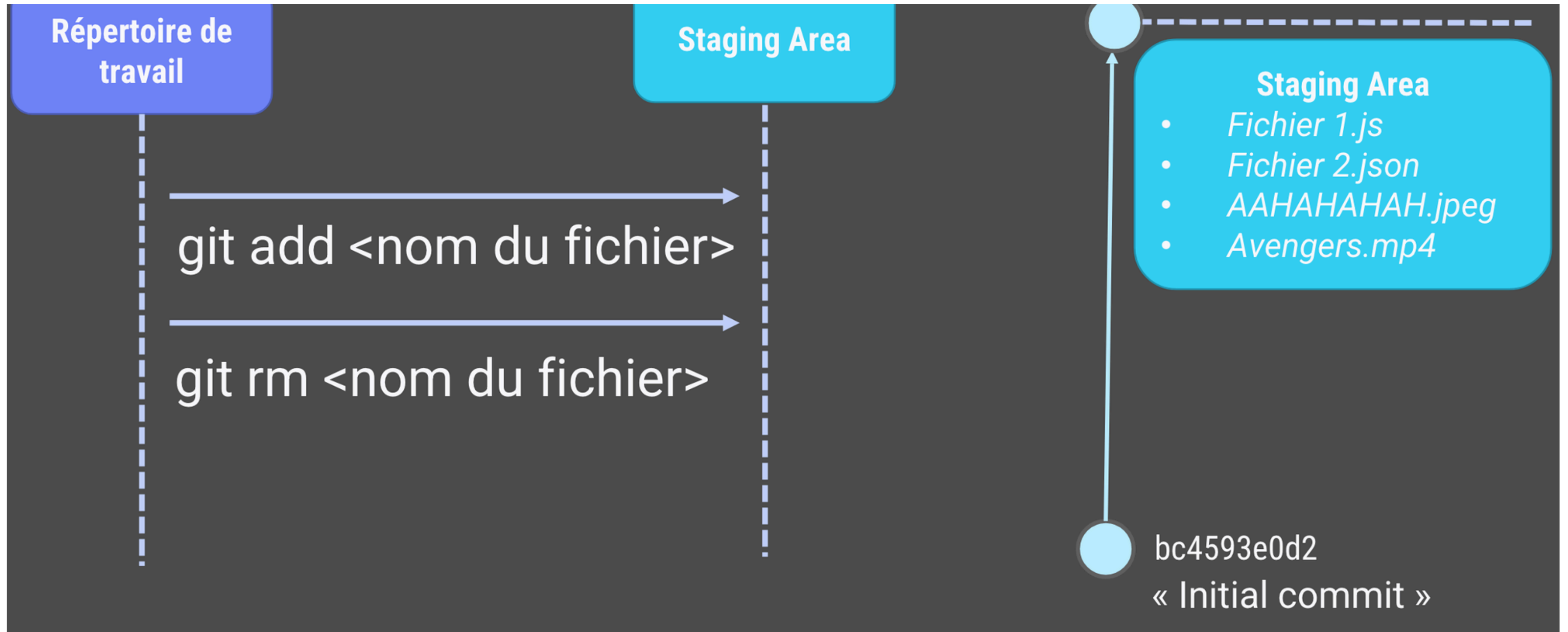
Les commandes de suppression et d'ajout de fichiers sont utilisées pour respectivement supprimer ou ajouter des modifications dans la zone de staging (index) de Git.

Cela équivaut préparez les changements pour le prochain commit.

Commande ---

- `git add <nom du fichier>`
- `git add .` // Ajoute tous les fichiers ajoutés depuis le dernier commit
- `git rm <nom du fichier>`

Ajouter ou supprimer des fichiers



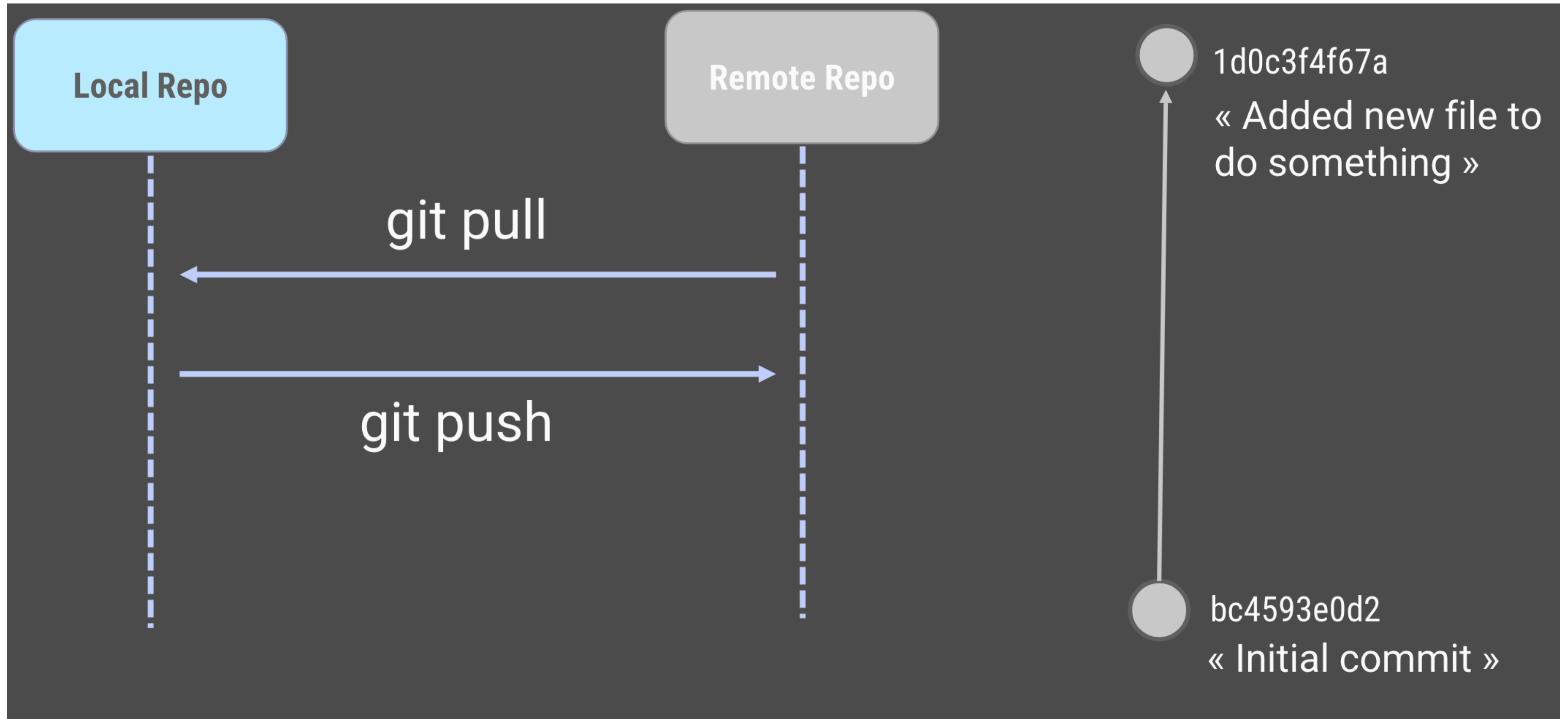
Recupérer / Envoyer son code dans le répo git

Pour récupérer ou envoyer son code vers un repository Git, vous avez besoin d'utiliser les commandes **git pull** et **git push**, respectivement.

Commande ---

- `git pull origin <branche>`
- `git push origin <branche>`

RÉCUPÉRER / ENVOYER SON CODE DANS LE REPO GIT



RESET UN COMMIT

La commande git reset est utilisée pour annuler des commits précédents. Elle modifie l'état de la branche courante en supprimant les commits spécifiés et en rétablissant l'index (la zone de staging) et/ou le répertoire de travail à l'état souhaité.

Commande ---

- `git log --oneline` // Afficher l'historique des commits
- `git reset <commit-hash>`

RESET UN COMMIT



REVERT UN COMMIT

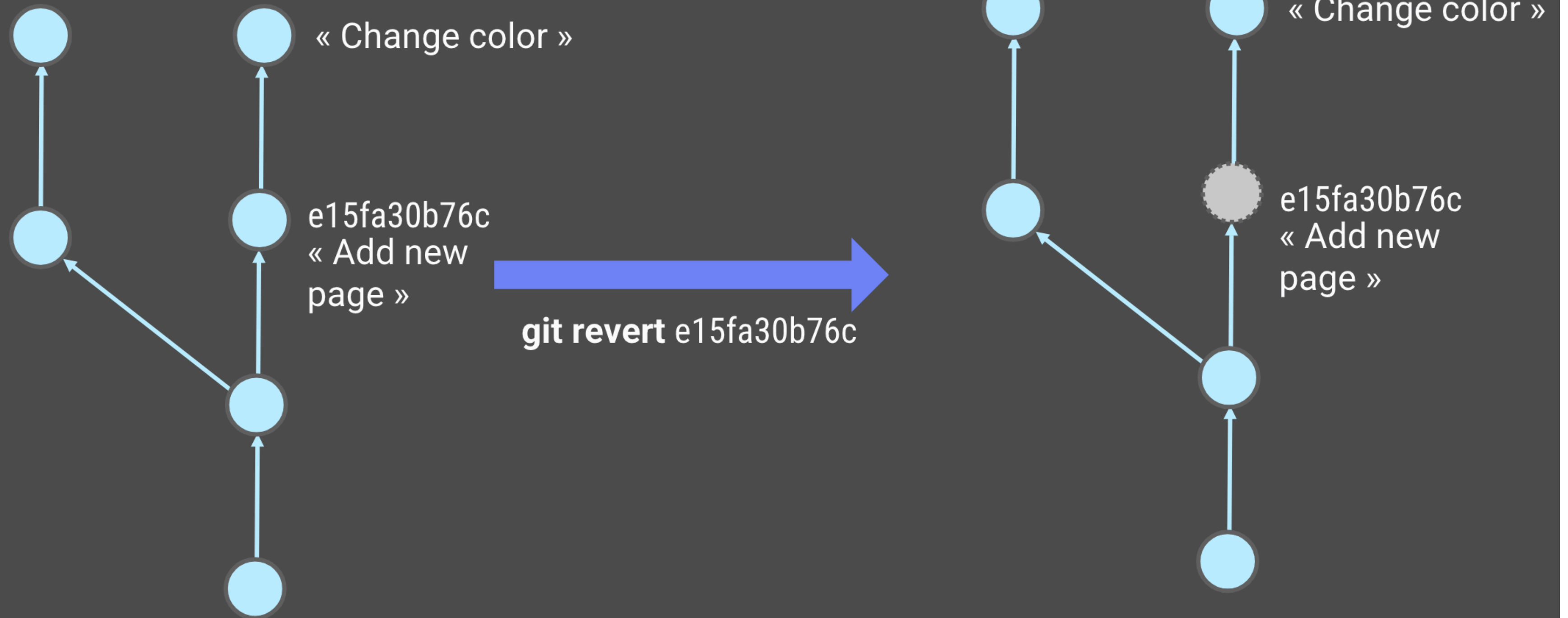
La commande **git revert** est utilisée pour créer un nouveau commit qui annule les modifications d'un commit précédent. Contrairement à `git reset`, qui modifie l'historique de commits, `git revert` ajoute un commit à l'historique, ce qui permet de conserver un historique propre et traçable.

C'est particulièrement utile dans des environnements de collaboration où l'historique des commits ne doit pas être réécrit.

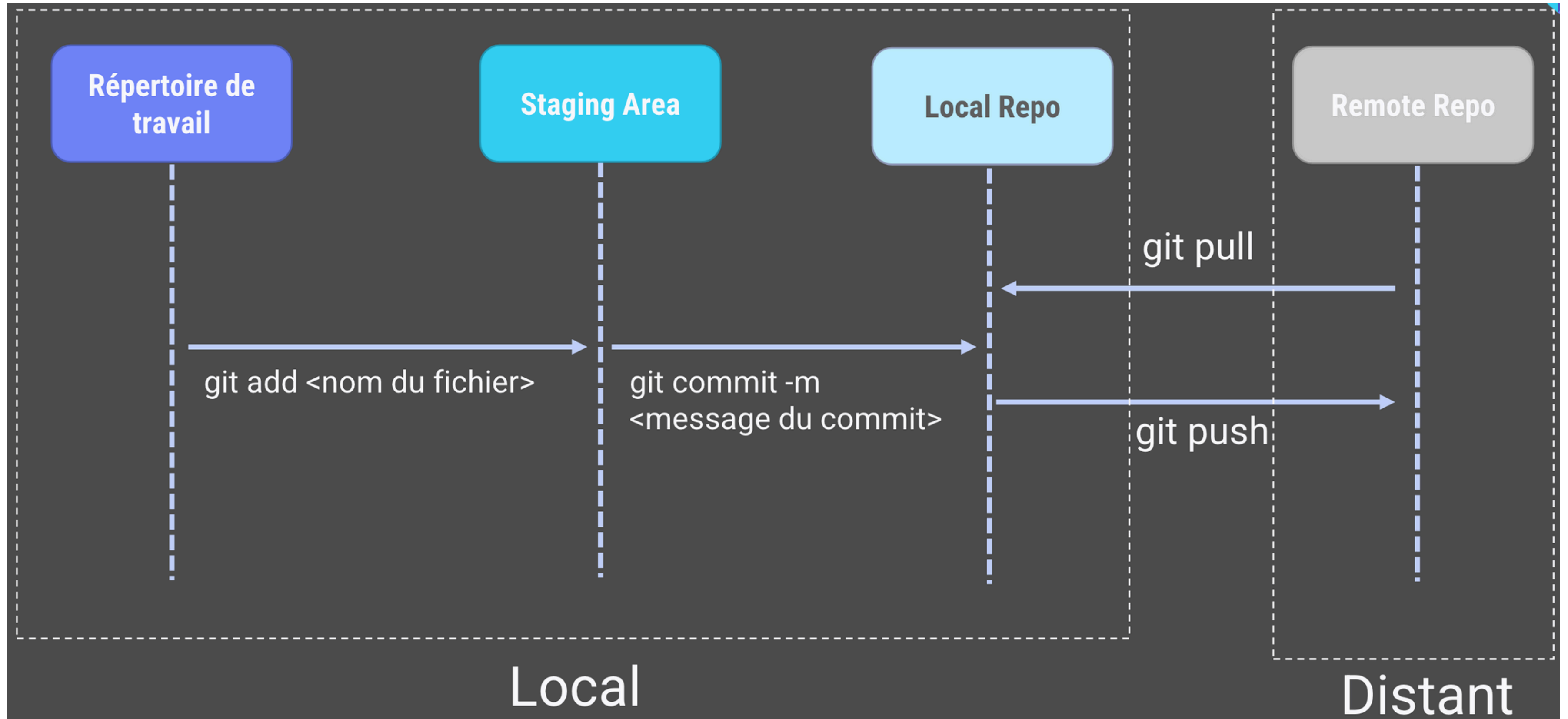
Commande

- `git log --oneline` // Afficher l'historique des commits
- `git revert <commit-hash>`

Revert un commit



BILAN DES ACTIONS



Merge Request

Une Merge Request (ou Pull Request sur GitHub) est un mécanisme permettant aux développeurs de notifier les autres membres de l'équipe qu'ils ont terminé une fonctionnalité ou une correction de bug et qu'ils souhaitent fusionner ces modifications dans la branche principale du projet.

Cela facilite la revue de code, les discussions et les tests avant que les modifications ne soient intégrées dans le code de production.

SIMULATION MERGE REQUEST

ViaRézo

>

Site vitrine V3

>

Merge requests

Open 2

Merged 175

Closed 10

All 187

Edit merge requests

New merge request

Recent searches ▾

Search or filter results...

Created date ▾

⌵

Feature: add latex slides

!189 · created 11 hours ago by Thomas Borot

Ready for review

✓

0

updated 11 hours ago

Update from demande to viarezo.demande (yes, even in the PDF.)

!188 · created 2 days ago by Téo Chaillou

Easy

Ready for review

✓

2

updated 11 hours ago

Conflits(1)

Les conflits se produisent dans Git lorsque deux branches modifient la même partie d'un fichier et que Git ne peut pas automatiquement fusionner ces modifications. Voici comment comprendre, résoudre et éviter les conflits dans Git.

Quand et Pourquoi les Conflits Se Produisent

- **Fusion (Merge) :**
 - Lorsqu'on essaie de fusionner deux branches qui ont modifié la même ligne ou des lignes adjacentes d'un fichier.
- **Rebasage (Rebase) :**
 - Lorsqu'on tente de réappliquer des commits sur une nouvelle base, en modifiant l'ordre des commits.

Conflicts(2)

Si des conflits sont détectés, vous verrez quelque chose comme :

Auto-merging fichier.txt

CONFLICT (content): Merge conflict in fichier.txt Automatic merge failed; fix conflicts and then commit the result.

Exemple de fichier en conflit (fichier.txt) :

<<<<<<< HEAD

Contenu de la branche principale.

=====

Contenu de la branche fusionnée.

>>>>>>> nouvelle-fonctionnalité

SIMULATION CONFLIT

```
Accept Current Change | Accept Incoming Change | Accept Both Changes | Compare Changes
79 <<<<<< HEAD (Current Change)
80   navigation.push(pathType, getPathIdByType(pathType, pathId));
81 =====
82   navigation.navigate(pathType, pathType === 'association' ? { code: pathId } : { id: pathId });
83 >>>>>> b09c6d2 (replace oush by navigate) (Incoming Change)
```

Bilan de comment travailler sur Git

- Je crée ma branche
- J'ajoute des commit dessus
- Je fusionne ma branche avec main

« L'esprit est juste comme un muscle ; plus vous l'exercez, plus il devient fort et plus il peut s'étendre. »

Idowu Koyenikan

Merci !

