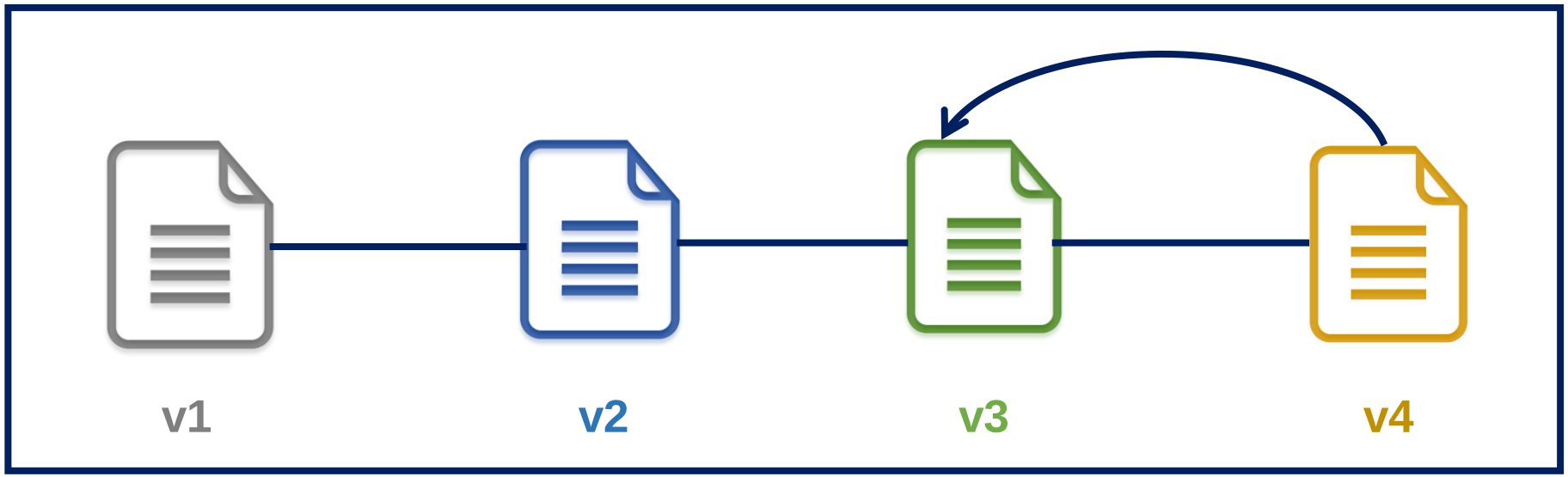
Gestion de versions avec Git

Gestion de versions

- Un contrôleur de versions (**V**ersion **C**ontrol **S**ystem ; VCS) est un programme qui permet aux développeurs de **conserver un historique** des modifications et des versions de tous les fichiers.
- La gestion de versions permet de :
  - travailler en équipe sans risquer de supprimer les modifications des autres collaborateurs,
  - revenir à une version précédente en cas de problème,
  - fournir une gestion de l'historique du logiciel.



Principales fonctionnalités

- Un logiciel de gestion des versions est un **dépôt de code** (**repository** ou **repo**) qui héberge le code source du projet.

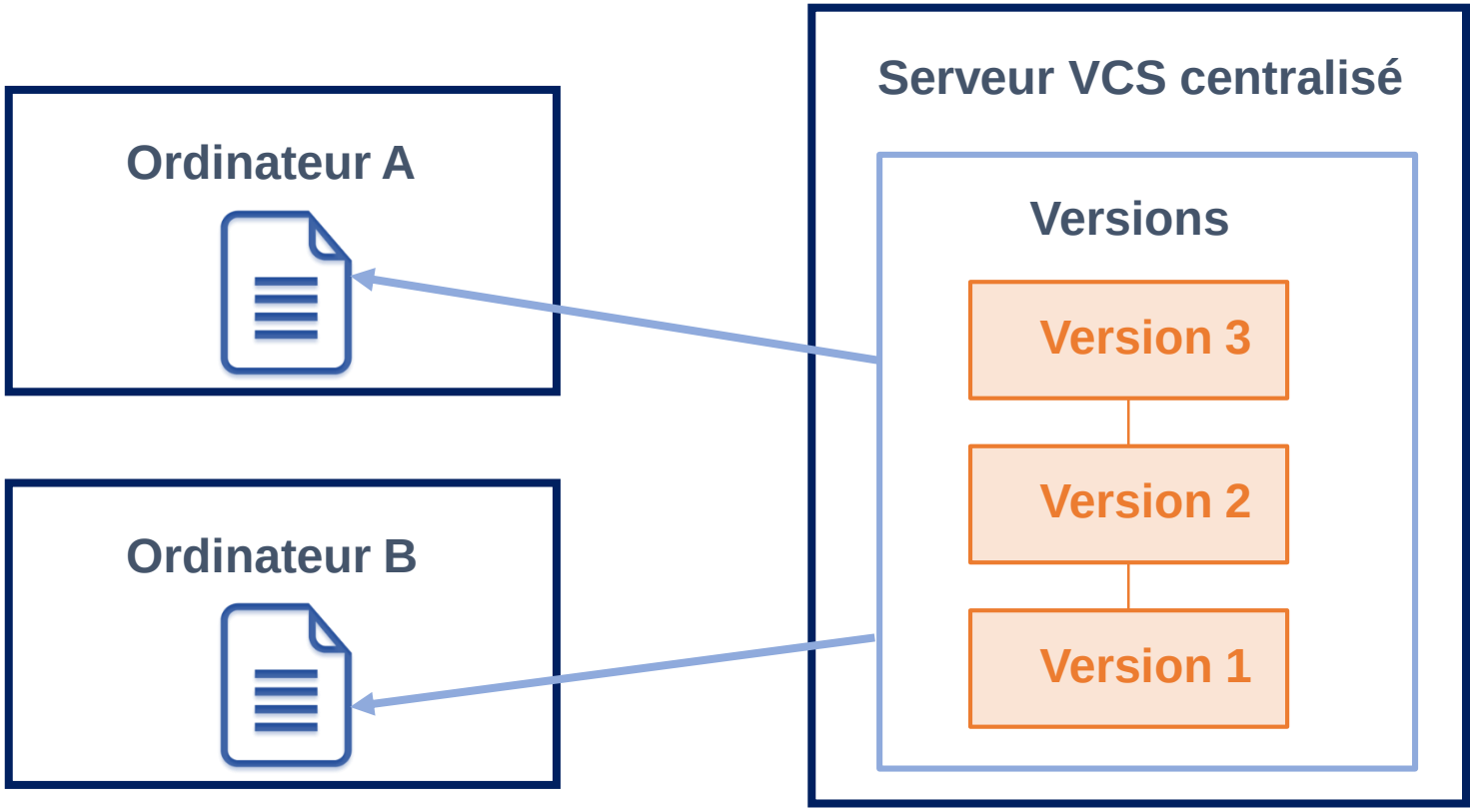
- Le logiciel garde la trace des modifications successives d'un fichier. Il permet d'en visualiser l'historique et de revenir à une version antérieure.

- Il permet de travailler **en parallèle** sur plusieurs problématiques en créant des **branches**. Les modifications réalisées sur une branche peuvent ensuite être intégrées à une autre.

- En cas d'un **conflit** (modifications simultanées du même fichier par plusieurs développeurs), le logiciel permet de comparer les versions du fichier et de choisir les modifications à conserver ou à rejeter pour créer le fichier fusionné final.

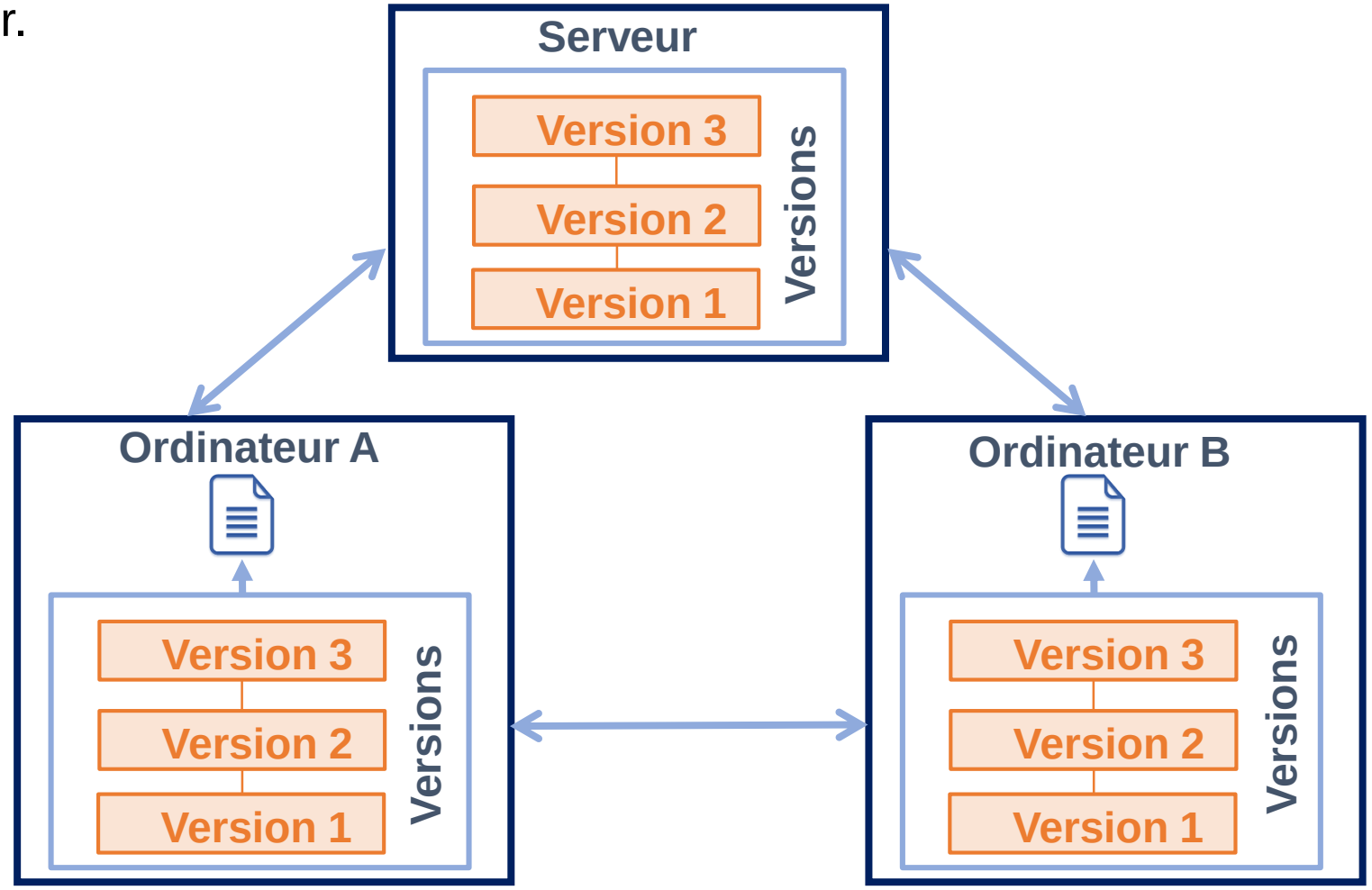
Gestion de versions centralisée

- Un seul dépôt qui fait référence.
- Chaque développeur se connecte au logiciel de VCS suivant le principe du client/serveur.
- ☹ Une connexion au logiciel de VCS est indispensable.



Gestion de versions décentralisée

- Un outil qui permet de travailler de manière décentralisé (hors ligne).
- Le logiciel fournit un service de synchronisation entre toutes ces bases de code. Cette solution fonctionne suivant le principe du pair-à-pair.



Principaux logiciels de gestion des versions

- **CVS** (**C**oncurrent **V**ersioning **S**ystem)
  - centralisé,
  - libre.
- **SVN** (Subversion)
  - centralisé,
  - libre.
- **Mercurial**
  - décentralisé,
  - libre.
- **Git**
  - décentralisé,
  - libre.
- **TFS** (**T**eam **F**oundation **S**erver)
  - décentralisé,
  - payant.

Git

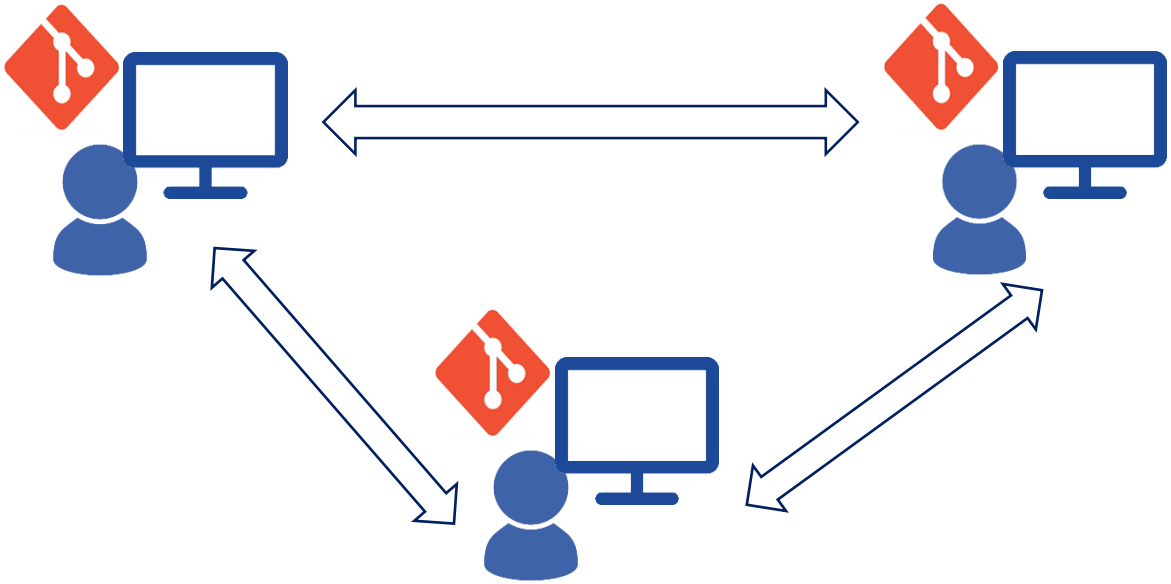
Git

- Logiciel libre de gestion de versions.
- Git a été créé par Linus Torval le créateur de Linux.
- Git permet de :
  - gérer le code source et ses différentes versions.
  - partager ce code source, permettant ainsi de travailler à plusieurs.
  - utiliser et de tester le code source des projets.
- Site officiel : https://git-scm.com
- Téléchargement : https://git-scm.com/download/win
- Ajout d'un compte sur GitHub : https://github.com

```
# Test de la version
git --version
```

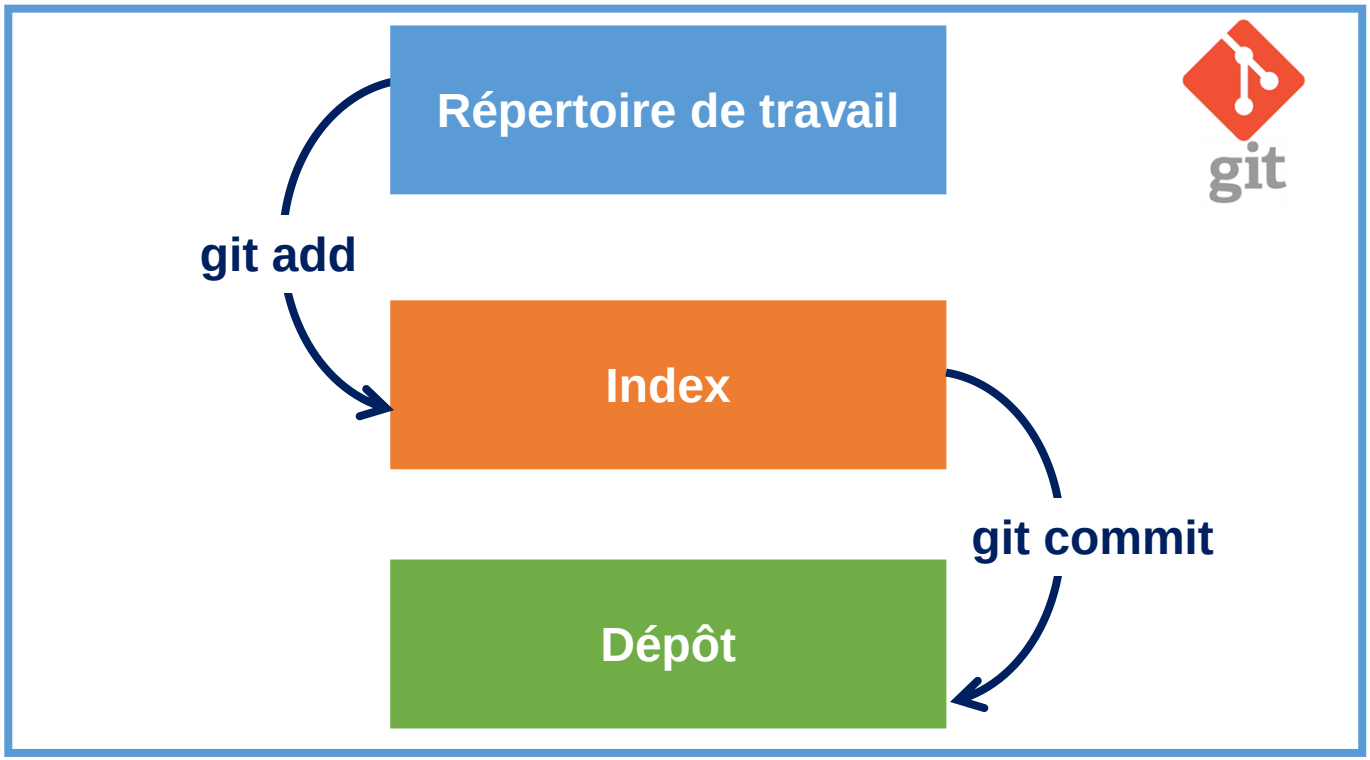
Fonctionnement

- Git rassemble dans un dépôt l'ensemble des données associées au projet (code source, historique, versions, etc).
- Chaque développeur travaille à son rythme sur son dépôt local.
- Git offre des mécanismes qui synchronisent les modifications entre tous les dépôts.

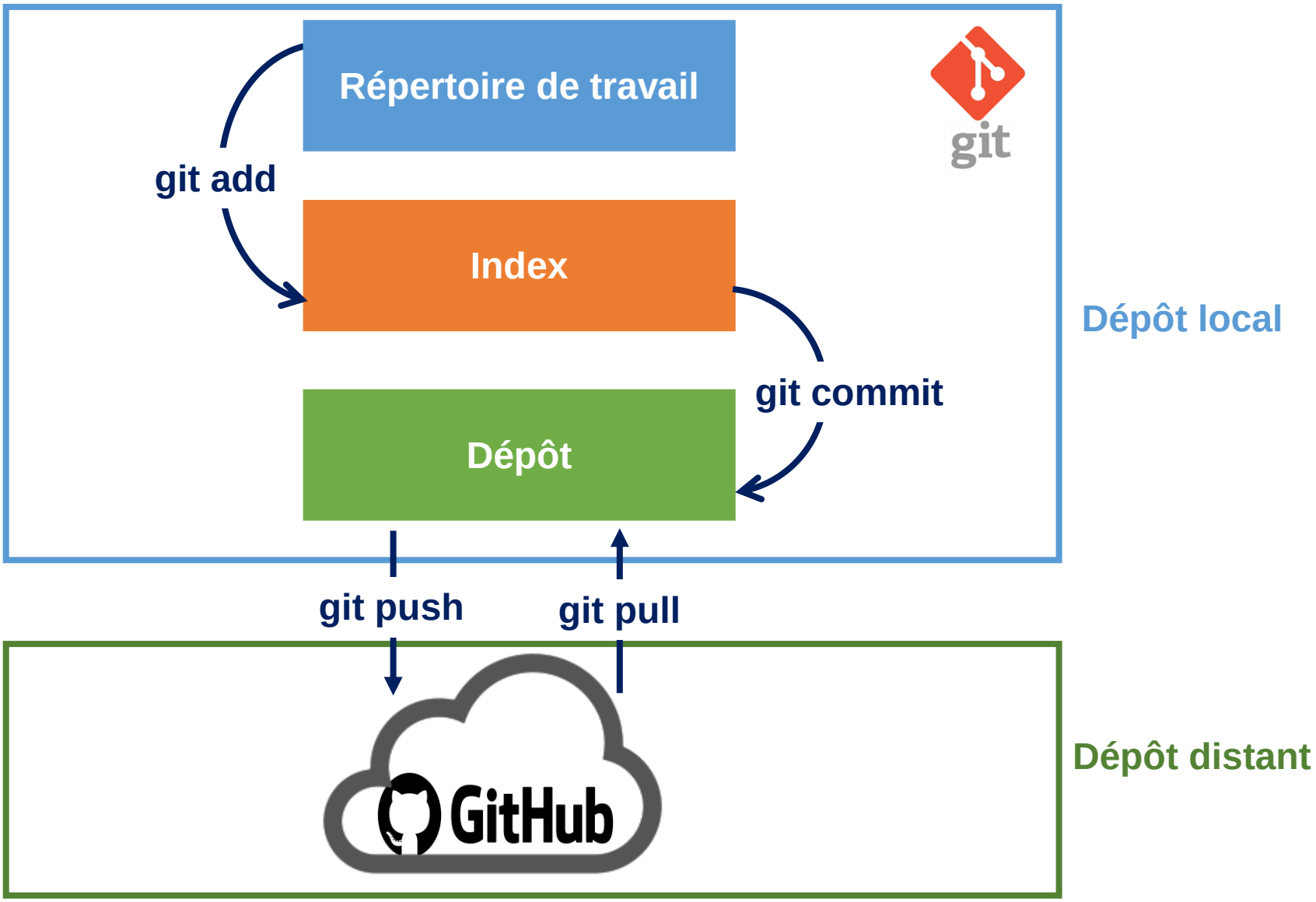


Espaces de Git

- Trois espaces de Git :
  - **Répertoire de travail** (working directory) : fichiers actuellement sauvegardés localement.
  - **Index** (staging area) : espace de transit.
  - **Dépôt** (repository) : fichiers dans l'état de la dernière validation effectuée



Git et GitHub

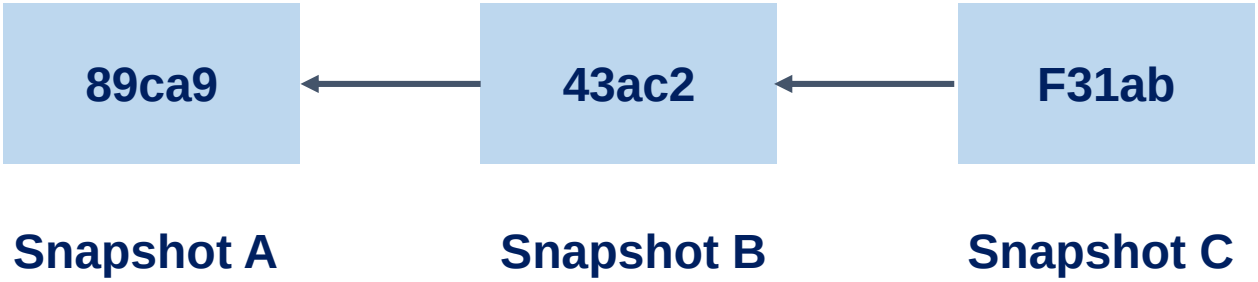


Commit

- Git stocke ses données comme une **série de fichiers d'instantanés** (snapshots), et non plus comme une série de modifications ou de différences successives.

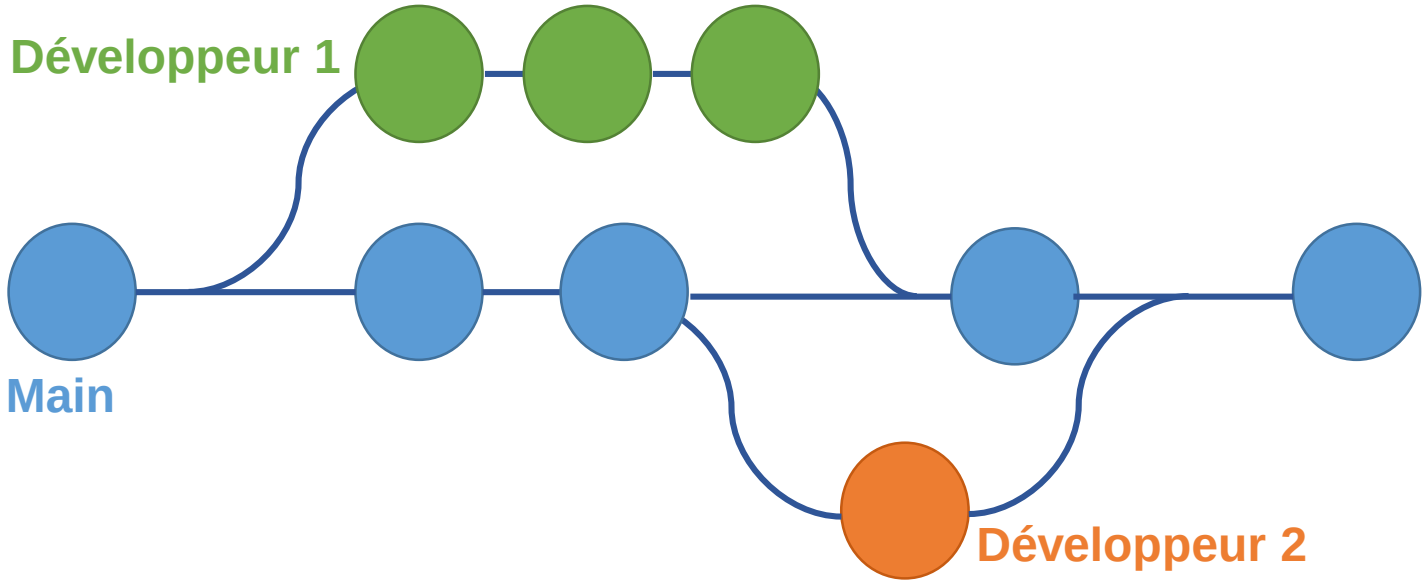
- Après des modifications et une validation à nouveau, le prochain snapshot validé avec un *commit* stocke un pointeur vers le snapshot qui le précédé immédiatement.

- **Exemple :**



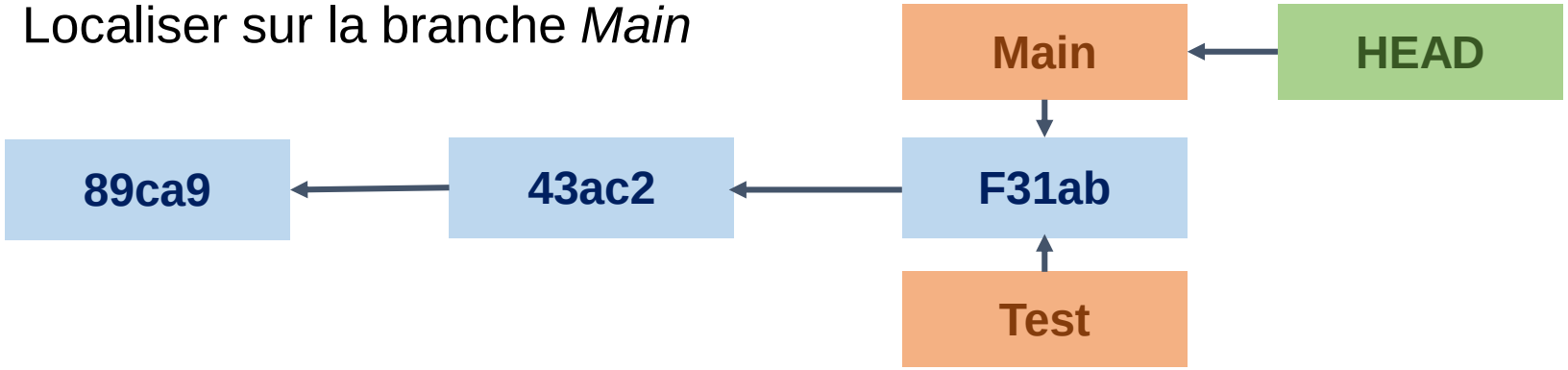
Système de branches

- Une branche correspond à un **pointeur** léger et déplaçable vers des snapshots.
- Sous Git, la branche principale est appelée la branche **main** (ou master pour les dépôts créés avant octobre 2020).
- La branche main porte l'intégralité des modifications effectuées.
  - Les modifications ne sont pas directement réalisées sur cette branche.
  - Les modifications sont réalisées sur l'autres branches, et après divers tests, elles sont intégrées sur la branche main.

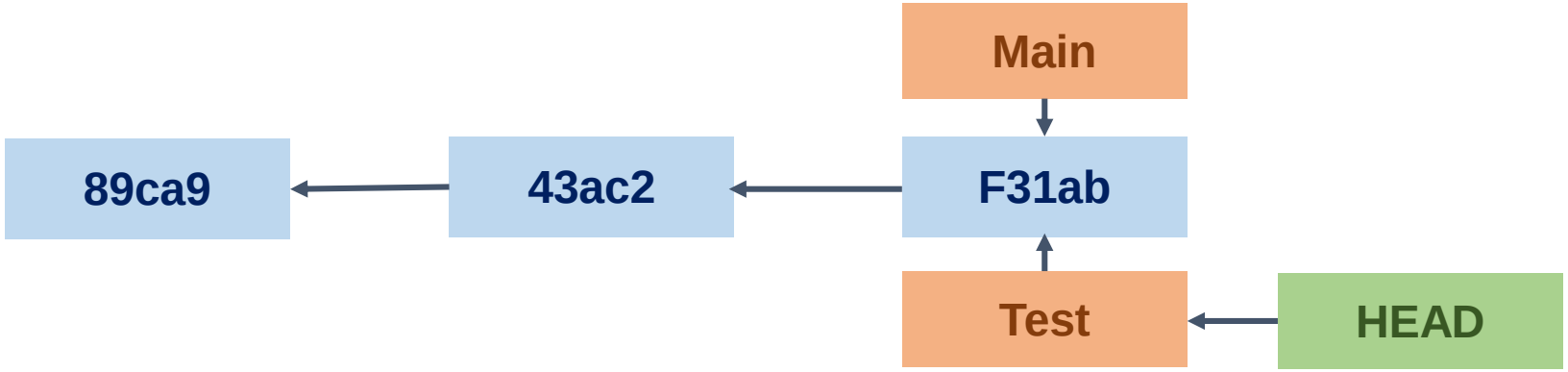


Système de branches

- Git conserve un pointeur spécial appelé **HEAD** afin de connaître la branche sur la quelle se trouve le développeur.
- **Exemple :** 2 branches *Main* et *Test* pointant vers la même série de snapshots
  - Localiser sur la branche *Main*



  - Basculer sur la branche *Test*



Initialisation de Git

- Configuration du nom d'utilisateur pour qu'il soit utilisé par Git

```
git config --global user.name "Nom"
```

- Configuration de l'email pour qu'il soit utilisé par Git

```
git config --global user.email <email>
```

- Vérification de la configuration globale Git

```
git config --list
```

- Activation des couleurs afin d'améliorer la lisibilité des différentes branches

```
$ git config --global color.diff auto
$ git config --global color.status auto
$ git config --global color.branch auto
```

Commandes Git de base

- Création d'un nouveau dépôt Git

```
git init
```

- Ajout de fichiers à l'index

```
git add .
git add "Nom fichier"
```

- Clonage d'un dépôt local ou distant

```
git clone /chemin/vers/dépôt
```

- Validation des modifications apportées au dépôt local

```
git commit –m "Description du commit"
```

- Affichage de liste des fichiers modifiés et les fichiers qui doivent encore être ajoutés ou validés

```
git status
```

Commandes Git de base

- Connexion du dépôt local à un serveur distant

```
git remote add origin /url/vers/dépôt/distant
```

- Envoie des modifications locales apportées à la branche principale

```
git push origin master
```

- Création d'une branche

```
command git checkout -b "nom-branche"
```

- Passage d'une branche à une autre

```
git checkout "nom-branche"
```

- Répertoriage de toutes les branches présentes dans le dépôt

```
git branch
```

- Suppression d'une branche

```
git branch –d "nom-branche"
```

Commandes Git de base

- Fusion de toutes les modifications présentes sur le dépôt distant dans le répertoire de travail local.

```
git pull
```

- Fusion d'une branche dans la branche active

```
git merge "nom-branche"
```

- Visualisation des conflits d'un fichier

```
git diff --base "nom-fichier"
```

- Affichage des conflits entre les branches à fusionner avant de les fusionner

```
git diff "branche-source" "branche-cible"
```

- Énumération de tous les conflits actuels

```
git diff
```

- Réinitialisation de l'index et le répertoire de travail à l'état du dernier commit

```
git reset --hard HEAD
```