Formation
Git Pour Tous
2020

http://www.math.u-bordeaux.fr/~lfacq/GitPourTous2020.pdf

Novembre 2020 v0.4

Institut de Mathématiques de Bordeaux

Logistique

2 pauses de 5 minutes !

- pause à H+1, reprise à H+1:05
- pause à H+2, reprise à H+2:05

à vous de revenir à l'heure !

à quoi sert git ?

- git permet de conserver tout l'historique des modifications codes sources et documents et aussi de ...
- ... travailler en local (mode déconnecté) : continuer à enregistrer des versions successives même sans réseau
- ... travailler depuis plusieurs ordinateurs
- ... travailler avec plusieurs personnes
- ... se resynchroniser à tout moment de façon maîtrisée et cohérente
- ... donc travailler depuis n'importe ou
- naviguer de façon claire et déterministe dans l'ensemble des versions
- système orienté fichiers textes : code source, latex, thèse, article, groupe de travail...
  - éviter les binaires, /!/ jamais de gros fichiers binaires/!/

au final, à quoi sert git ... ?

... à rester zen !

plus aucune inquiétude de perdre quoique ce soit car tout est conservé, cohérent et réversible !

objectifs de cette session

- que vous soyez capable de travailler seuls depuis un ou plusieurs ordinateurs avec un dépôt distant.
- nous allons volontairement présenter des éléments partiels pour que cela reste simple pour une première approche.
  - /!/ il existe plusieurs façons de faire les même choses, nous n'en verrons qu'une
  - /!/ les commandes git ont beaucoup d'options, nous ne verrons que quelques formes
  - /!/ nous ne parlerons pas de la notion de branches /!/

Préliminaires : installer la "ligne de commande" git

- si possible : installer le package git de votre distribution
  - linux (ubuntu) : sudo aptitude install git
  - linux (centos) : sudo yum install git
  - mac : brew install git
- sinon, téléchargez git sur `https://git-scm.com/downloads`
- windows : utilisez l'application "Git Bash"

Préliminaires : avoir une clef SSH et connaître son mot de passe (passphrase)

- si besoin :
  - `ssh-keygen` : pour générer une clef
  - `eval $(ssh-agent -s)` : pour démarrer un agent SSH qui conserve les clefs disponible pendant la session courante
  - `ssh-add` : pour nourrir le SSH Agent avec une clef déverrouillée

Préliminaires : accéder à un gitlab et y copier sa clef SSH

- vérifiez que vous avez les droits pour accéder à votre compte personnel sur un des gitlab :
  - `https://plmlab.math.cnrs.fr`
  - `https://gitlab.inria.fr`
  - `https://gitub.u-bordeaux.fr`
  - ...
- ajouter votre clef SSH dans votre profile gitlab pour faciliter les accès aux dépôts (à faire une seule fois par serveur gitlab) :
  - aller dans votre profile (en haut à droite)
  - cliquer sur Settings
  - dans la barre de gauche, cliquer sur SSH Keys
  - copier-coller le contenu du fichier `$HOME/.ssh/id_rsa.pub` (ou équivalent) dans le cadre
  - cliquer ok

TP Préliminaire : créer un dépôt distant de test "MonTest"

- créer un dépôt distant (interface web gitlab) en allant sur votre gitlab préféré
  - sur `https://plmlab.math.cnrs.fr`
  - sur `https://gitlab.inria.fr`
  - ...
- créer un dépôt en cliquant sur le + (en haut au milieu)
- choisissez un nom. par exemple : MonTest
- cliquer sur le bouton clone en haut a droite
- cette ligne référence de manière unique votre projet sur le serveur. c'est l'URL de votre projet (Uniform Ressource Locator). Nous en aurons besoin dans chaque TP.
- note: la page d'accueil d'un projet vide documente toutes les manières de remplir ce dépôt

Institut de Mathématiques de Bordeaux

TP Préliminaire : cloner un dépôt distant de test "**MonTest**"

- cloner le dépôt sur votre ordinateur avec ssh :
  - `git clone git@plmlab.math.cnrs.fr:VotreLogin/MonTest`
  - `git clone git@gitlab.inria.fr:VotreLogin/MonTest`
- cela va créer un répertoire *MonTest* avec une copie complète du dépôt git c.a.d :
  - la version courante ainsi que **tout l'historique de toutes les versions**.
  - ici, le dépôt étant vide, le répertoire sera également vide.

TP Préliminaire : définir son Nom et son Email

- Définissez votre identité pour l'enregistrement des versions :
  - `git config --global user.email p.nom@math.u-bordeaux.fr`
  - `git config --global user.name "Prénom Nom"`
- Pré-configurer un comportement de git : (non détaillé pour l'instant)
  - `git config --global push.default simple`

- ces informations de configuration sont stockées à la racine de votre répertoire utilisateur dans le fichier texte (lisible et éditable) : `$HOME/.gitconfig`
- elles sont uniquement stockées sur votre ordinateur actuel ...
- ... donc à faire sur tous vos postes de travail

cloner un dépôt distant - quelques précisions

- Cloner un projet :
  - avec un URL ssh - le plus pratique - mais pas de mode anonyme
    - `git clone git@plmlab.math.cnrs.fr:VotreLogin/MonProjet.git`
  - on peut aussi cloner avec un URL https => saisie du mot de passe à chaque fois... sauf en mode anonyme
    - `git clone https://plmlab.math.cnrs.fr/VotreLogin/MonProjet.git`
- Rappel: cela va créer un répertoire *MonProjet* avec une copie **complète** du dépôt git

syntaxe générale: `git clone URL [nom_du_repertoire_a_creer]`

par défaut la fin de l'URL (sans .git) donne le nom du répertoire crée
Exemples :

- `git clone git@mygit.fr:lf/Project` : clone dans `Projet`
- `git clone git@mygit.fr:lf/Project.git` : clone dans `Projet`
- `git clone git@mygit.fr:lf/Project.git Pro` : clone dans `Pro`

Institut de Mathématiques de Bordeaux

Objectif 1 : Travailler seul, en local
Session type

- Créer un dépôt local (ex: ici, en clonant un dépôt distant)
- Éditer des fichiers
- Enregistrer les changements dans le dépôt local
- Éditer des fichiers
- Enregistrer les changements dans le dépôt local
- ... etc ...

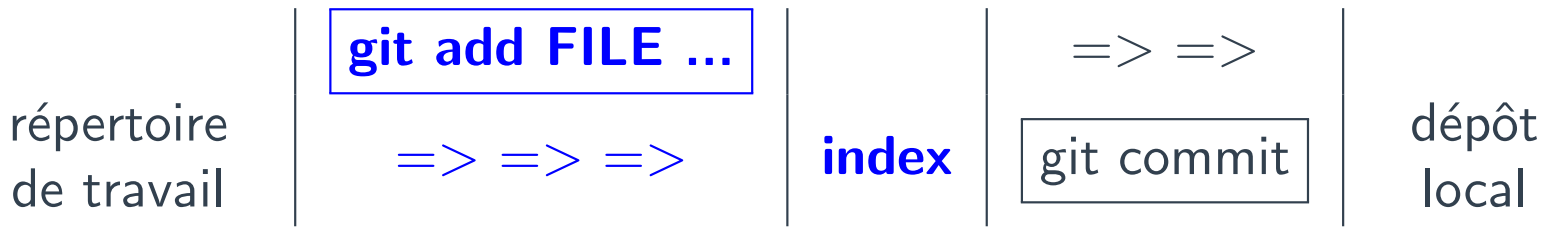
git : premier survol en un slide !

1. créer un projet (dépôt distant) sur gitlab
2. cloner ce projet dans le répertoire *MonProjet* :
   - cloner le dépôt (projet) distant : **git clone** *URL.../MonProjet*
   - aller dans ce projet : **cd MonProjet**
3. cela rajoute 3 zones locales contenant les fichiers (+1 distante) :
   - **répertoire de travail** :
     contient les fichiers courants de la **version courante**
   - **l'index** : contient la **nouvelle version** en préparation
   - **le dépôt local** : archive les anciennes **versions**
   - \+ **le dépôt distant** : permet sauvegardes et collaborations

les 3 zones locales : le répertoire de travail

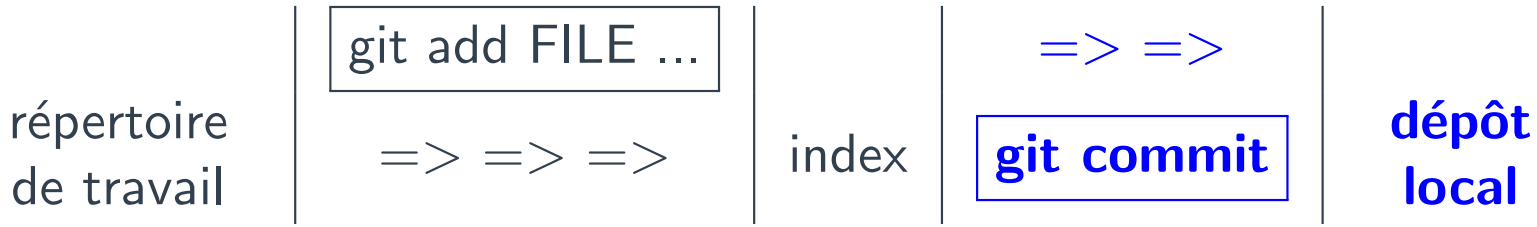
- **répertoire de travail** (*working directory*) :
  - c'est la seule zone directement visible.
  - elle contient une version des fichiers de votre projet, généralement la "dernière" version.
  - c'est toujours dans cette zone que vous travaillez : éditer les fichiers, lancer des tests, ...
  - elle peut contenir des fichiers qui ne sont pas du tout suivis par git (résultats de compilation par exemple : .o, exécutables, .dvi, pdf générés, ...)

les 3 zones locales : l'index



- **l'index** (*staging area*)
  - **zone de préparation de la prochaine version**
  - à tout moment, de manière incrémentale, quand un ou plusieurs fichiers du répertoire de travail vous conviennent, vous pouvez les recopier du répertoire de travail vers l'index, en prévision de la prochaine version que vous allez vouloir sauvegarder
  - **git add FILE ...** : **recopie** des fichiers du répertoire de travail vers l'index

les 3 zones locales : dépôt local



- **dépôt local** (*local repo[sitory]*) :
  - **stockage des versions**
  - une fois que la version (ensemble des fichiers) qui se trouve dans l'index vous convient :
  - **git commit -m "... commentaire ..."** sauvegarde, dans le dépôt local, **le contenu intégrale** de l'index sous la forme d'une nouvelle "version", comme une photo complète et instantanée de l'ensemble des fichiers
  - le dépôt contient, dans l'ordre, l'ensemble des versions précédemment sauvegardées

Terminologie: un **commit** git = une version complète du projet

- un **commit** est une version complète (comme une photo instantanée) de l'ensemble des fichiers et répertoires du projet
- pour bien comprendre git, il faut vraiment considérer que git sauvegarde à chaque fois TOUS les fichiers dans TOUTES les versions, même si bien sûr, dans la pratique, les choses sont optimisées.
- **chaînage des commits** dans le dépôt : pour pouvoir suivre l'évolution des versions , git conserve un lien entre les différents **commits** : ils sont chaînés entre eux : $A \rightarrow B \rightarrow C$
- **identifiant unique** : les commits sont référencés de manière unique par un **hash**[1] élaboré avec "l'ensemble des informations contenu dans la version associée"

[1] signature ou empreinte numérique unique

fonctionnement des 3 zones locales : création fichier$_1$ $\rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail	Index (staging area)	Dépôt Local
Etat Initial Vide	(vide)	(vide)	(vide)
$\Downarrow$ Création de Fichier$_1$	Fichier$_1$	(vide)	(vide)
$\Downarrow$ git add Fichier$_1$	Fichier$_1$	copie $\Rightarrow$ Fichier$_1$	(vide)
$\Downarrow$ git commit	Fichier$_1$	Fichier$_1$	copie tout $\Rightarrow$ Fichier$_1$

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création fichier$_1$ → git add → git commit

Actions	Répertoire de Travail	Index (staging area)	Dépôt Local
Etat Initial Vide	(vide)	(vide)	(vide)
⇓ Création de Fichier$_1$	**Fichier$_1$**	(vide)	(vide)
⇓ git add Fichier$_1$	Fichier$_1$	copie ⇒ Fichier$_1$	(vide)
⇓ git commit	Fichier$_1$	Fichier$_1$	copie tout ⇒ Fichier$_1$

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création fichier$_1$ → git add → git commit

Actions	Répertoire de Travail	Index (staging area)	Dépôt Local
Etat Initial Vide	(vide)	(vide)	(vide)
⇓ Création de Fichier$_1$	**Fichier$_1$**	(vide)	(vide)
⇓ git add Fichier$_1$	**Fichier$_1$**	copie ⇒ Fichier$_1$	(vide)
⇓ git commit	Fichier$_1$	Fichier$_1$	copie tout ⇒ Fichier$_1$

fonctionnement des 3 zones locales : création fichier$_1$ $\rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial Vide	(vide)		(vide)		(vide)
$\Downarrow$ Création de Fichier$_1$	**Fichier$_1$**		(vide)		(vide)
$\Downarrow$ git add Fichier$_1$	**Fichier$_1$**	copie $\Rightarrow$	**Fichier$_1$**		(vide)
$\Downarrow$ git commit	Fichier$_1$		Fichier$_1$	copie tout $\Rightarrow$	Fichier$_1$

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création fichier$_1$ $\rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)	Dépôt Local
Etat Initial Vide	(vide)		(vide)	(vide)
$\Downarrow$ Création de Fichier$_1$	**Fichier$_1$**		(vide)	(vide)
$\Downarrow$ git add Fichier$_1$	**Fichier$_1$**	copie $\Rightarrow$	**Fichier$_1$**	(vide)
$\Downarrow$ git commit	**Fichier$_1$**		**Fichier$_1$**	copie tout $\Rightarrow$ **Fichier$_1$**

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création fichier$_1$ $\rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial Vide	(vide)		(vide)		(vide)
$\Downarrow$ Création de Fichier$_1$	**Fichier$_1$**		(vide)		(vide)
$\Downarrow$ git add Fichier$_1$	**Fichier$_1$**	copie $\Rightarrow$	**Fichier$_1$**		(vide)
$\Downarrow$ git commit	**Fichier$_1$**		**Fichier$_1$**	copie tout $\Rightarrow$	**Fichier$_1$**

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création de Fichier$_2$ $\rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fichier$_1$		Fichier$_1$		Fichier$_1$
$\Downarrow$					
Création de Fichier$_2$	Fichier$_1$ **Fichier$_2$**		Fichier$_1$		Fichier$_1$
$\Downarrow$					
git add Fichier$_2$	Fichier$_1$ **Fichier$_2$**	copie $\Rightarrow$	Fichier$_1$ **Fichier$_2$**		Fichier$_1$
$\Downarrow$					
git commit	Fichier$_1$ **Fichier$_2$**		Fichier$_1$ **Fichier$_2$**	copie $\Rightarrow$ tout	Fichier$_1$ **Fichier$_2$** $\rightarrow$ Fichier$_1$

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création de Fichier$_2$ → git add → git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fichier$_1$		Fichier$_1$		Fichier$_1$
⇓					
Création de Fichier$_2$	Fichier$_1$ **Fichier$_2$**		Fichier$_1$		Fichier$_1$
⇓					
git add Fichier$_2$	Fichier$_1$ **Fichier$_2$**	copie ⇒	Fichier$_1$ **Fichier$_2$**		Fichier$_1$
⇓					
git commit	Fichier$_1$ **Fichier$_2$**		Fichier$_1$ **Fichier$_2$**	copie ⇒ tout	Fichier$_1$ **Fichier$_2$** → Fichier$_1$

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création de Fichier$_2$ $\rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail	Index (staging area)	Dépôt Local
Etat Initial	Fichier$_1$	Fichier$_1$	Fichier$_1$
$\Downarrow$			
Création de Fichier$_2$	Fichier$_1$ **Fichier$_2$**	Fichier$_1$	Fichier$_1$
$\Downarrow$			
git add Fichier$_2$	Fichier$_1$ **Fichier$_2$**	copie $\Rightarrow$ Fichier$_1$ **Fichier$_2$**	Fichier$_1$
$\Downarrow$			
git commit	Fichier$_1$ **Fichier$_2$**	Fichier$_1$ **Fichier$_2$**	copie $\Rightarrow$ tout Fichier$_1$ **Fichier$_2$** $\rightarrow$ Fichier$_1$

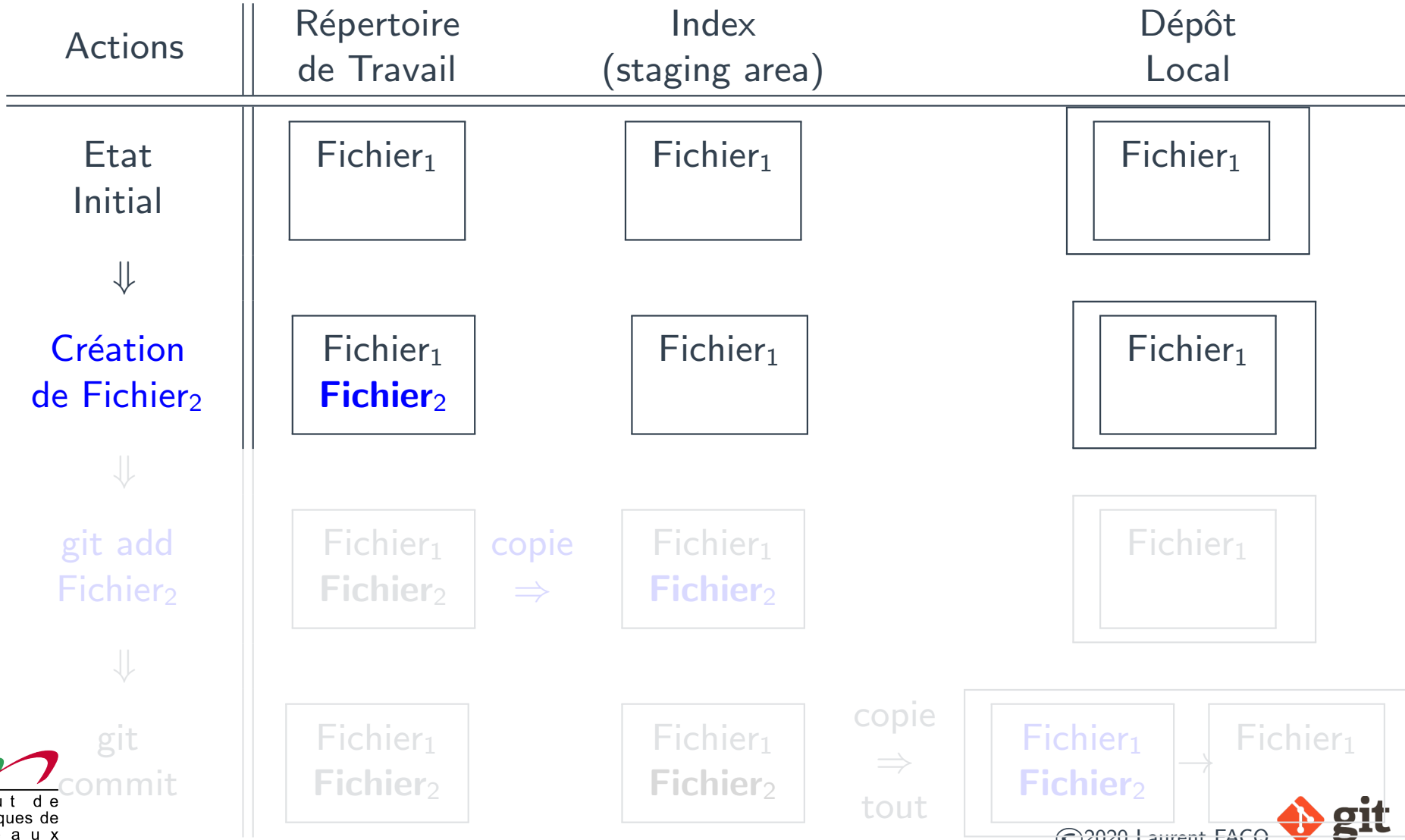
Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création de Fichier$_2$ → git add → git commit

Actions	Répertoire de Travail	Index (staging area)	Dépôt Local
Etat Initial	Fichier$_1$	Fichier$_1$	Fichier$_1$
⇓			
Création de Fichier$_2$	Fichier$_1$ **Fichier$_2$**	Fichier$_1$	Fichier$_1$
⇓			
git add Fichier$_2$	Fichier$_1$ **Fichier$_2$**	copie ⇒ Fichier$_1$ **Fichier$_2$**	Fichier$_1$
⇓			
git commit	Fichier$_1$ **Fichier$_2$**	Fichier$_1$ **Fichier$_2$**	copie ⇒ tout Fichier$_1$ **Fichier$_2$** → Fichier$_1$

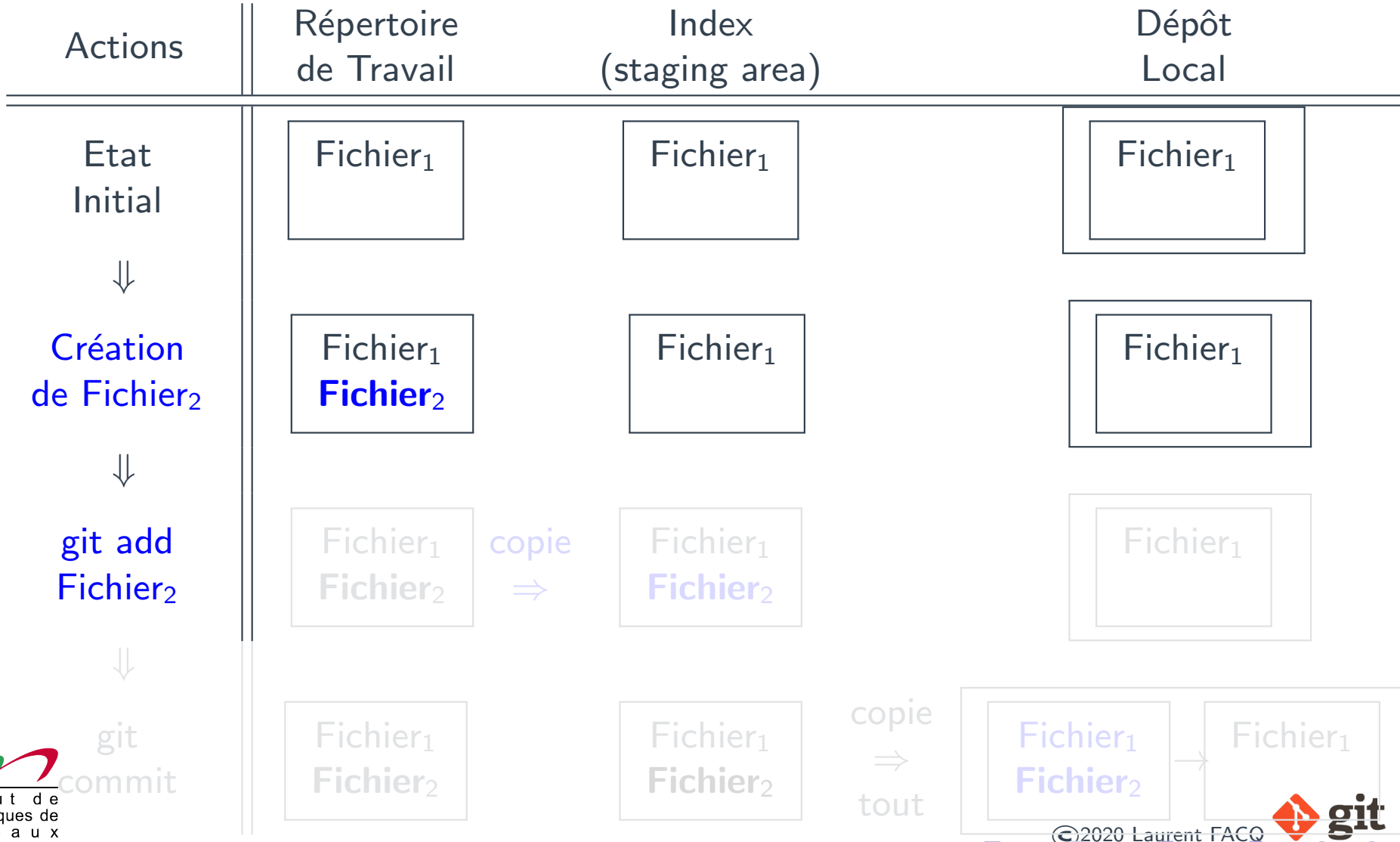
Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création de Fichier$_2$ $\rightarrow$ git add $\rightarrow$ git commit

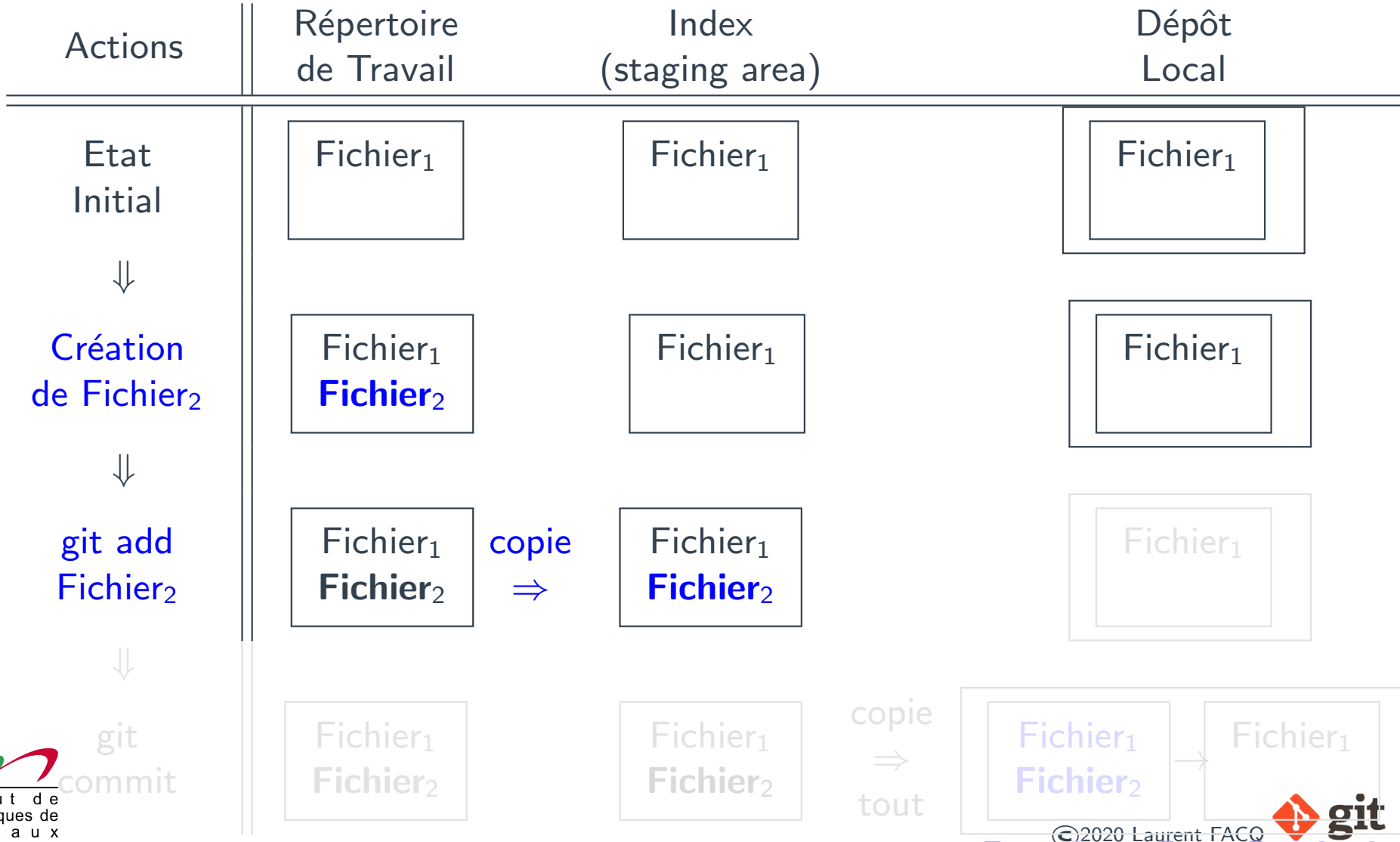


Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création de Fichier$_2$ $\rightarrow$ git add $\rightarrow$ git commit

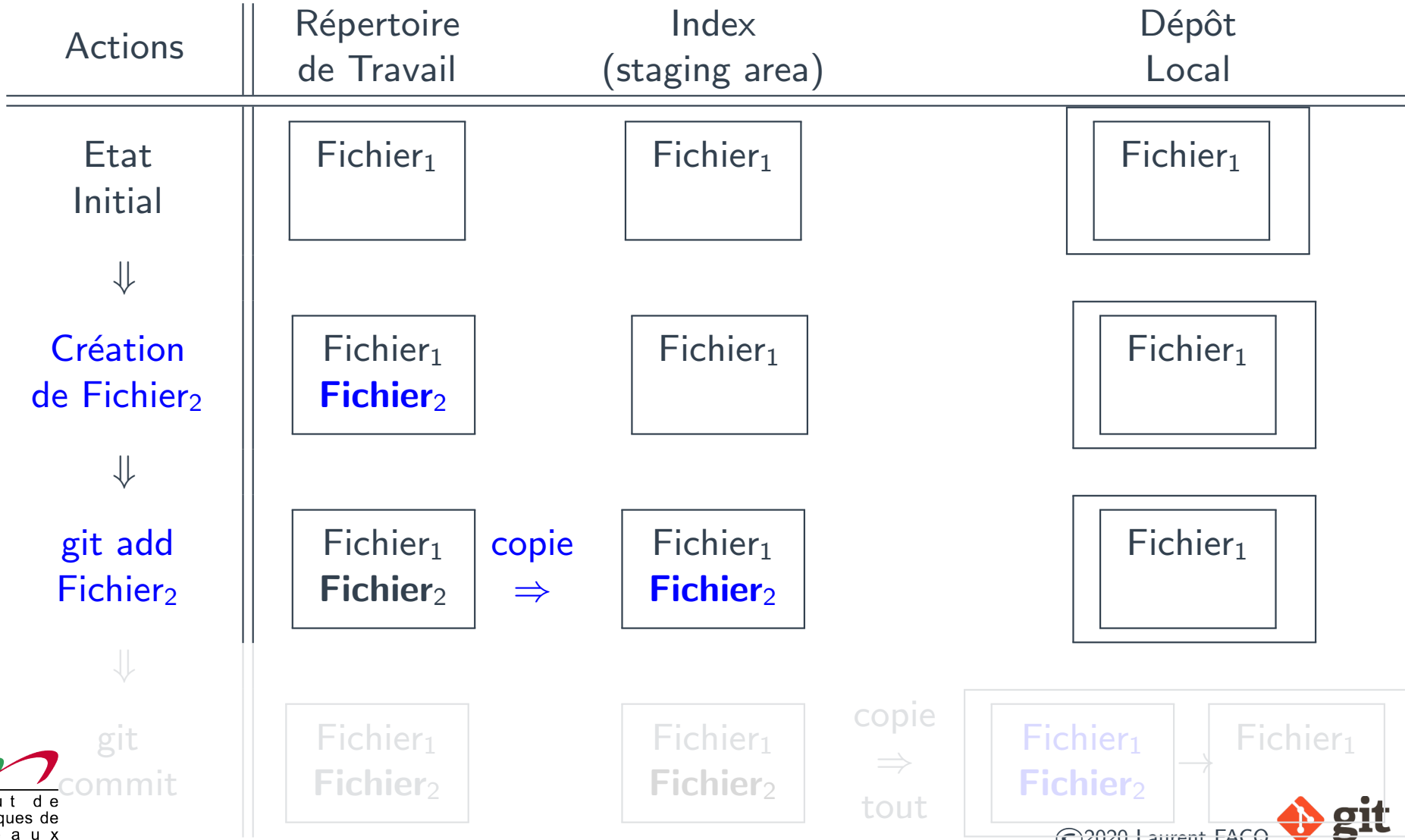


fonctionnement des 3 zones locales : création de Fichier$_2$ $\rightarrow$ git add $\rightarrow$ git commit

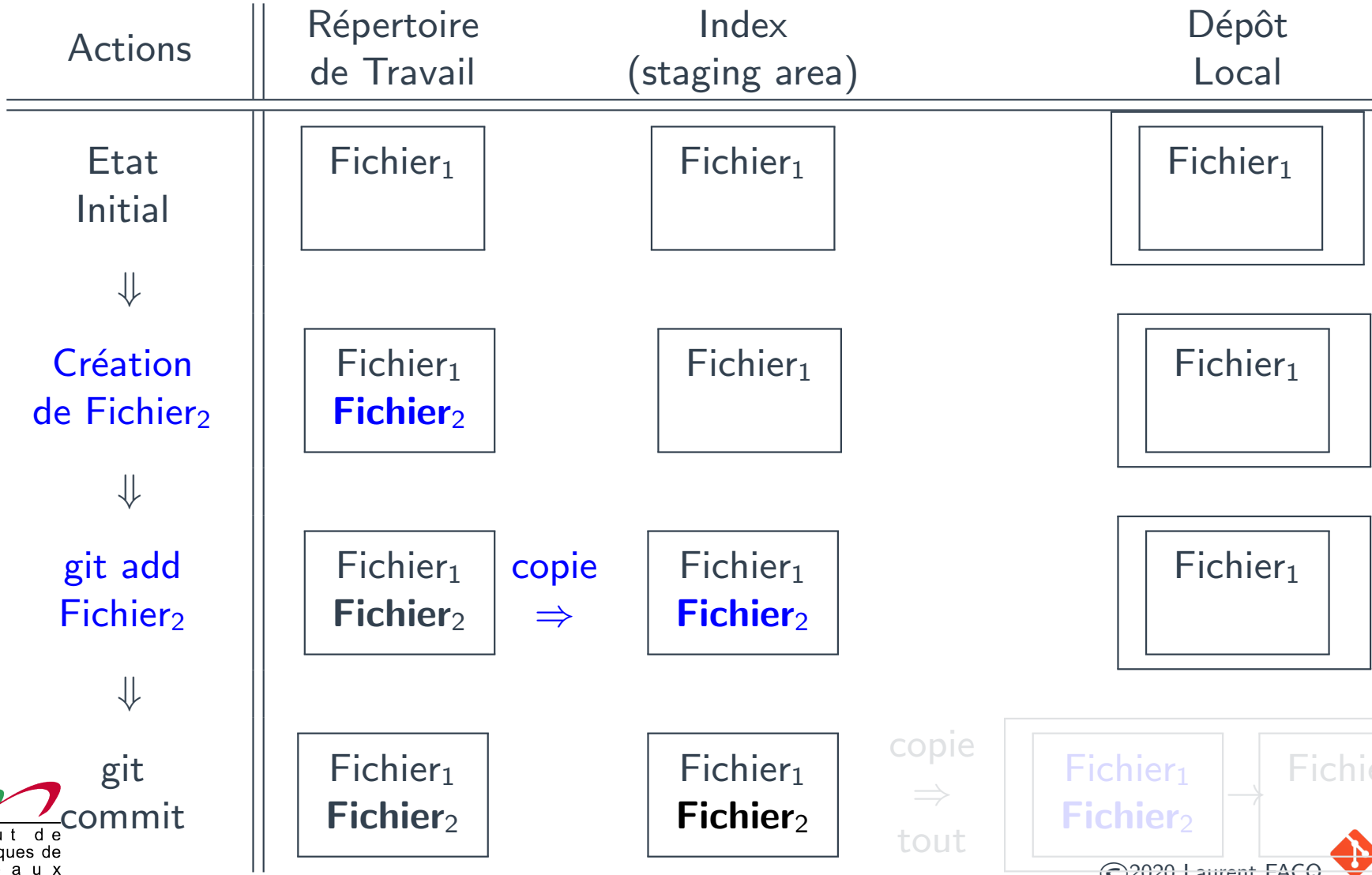


Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création de Fichier$_2$ $\rightarrow$ git add $\rightarrow$ git commit

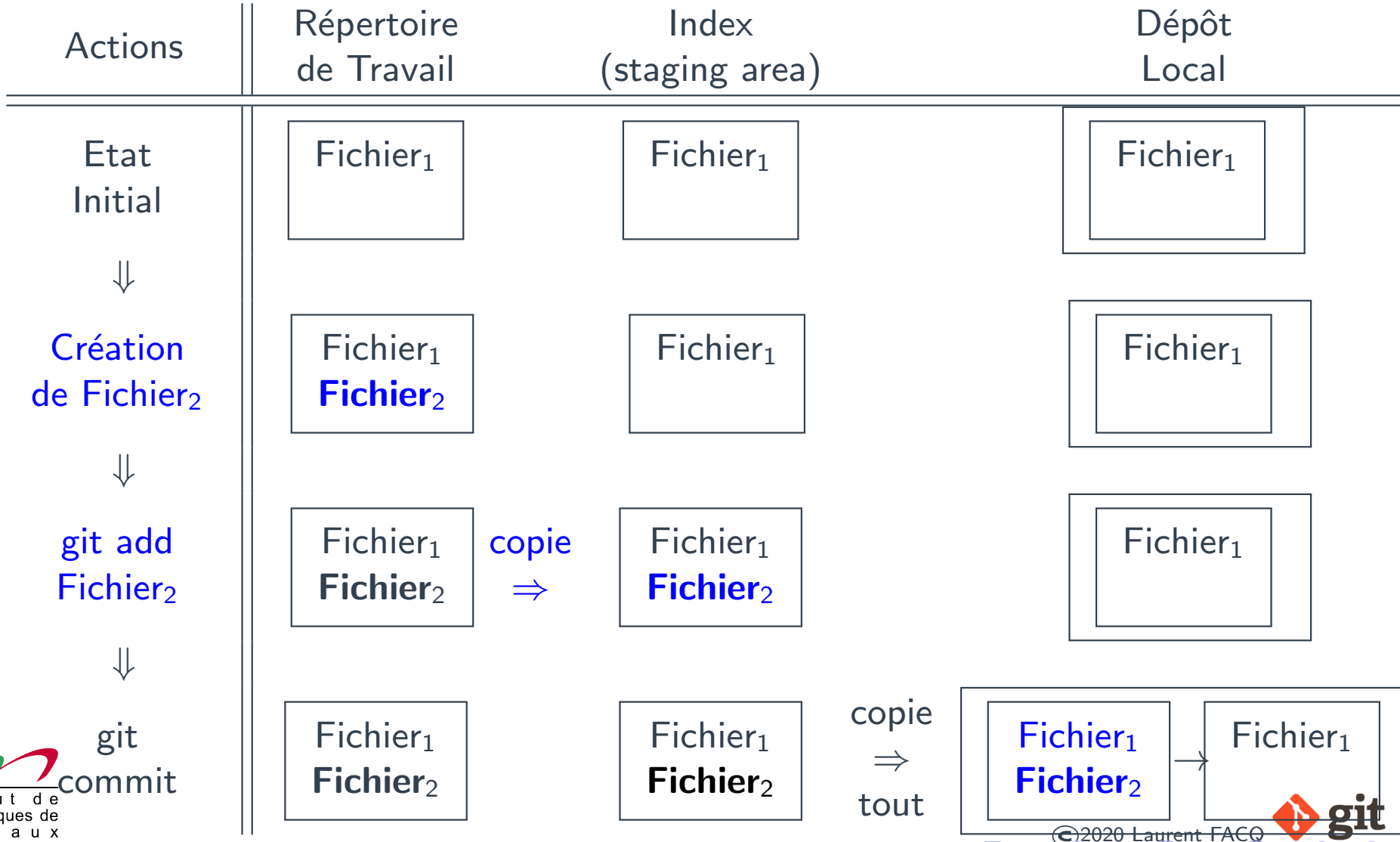


fonctionnement des 3 zones locales : création de Fichier$_2$ $\rightarrow$ git add $\rightarrow$ git commit



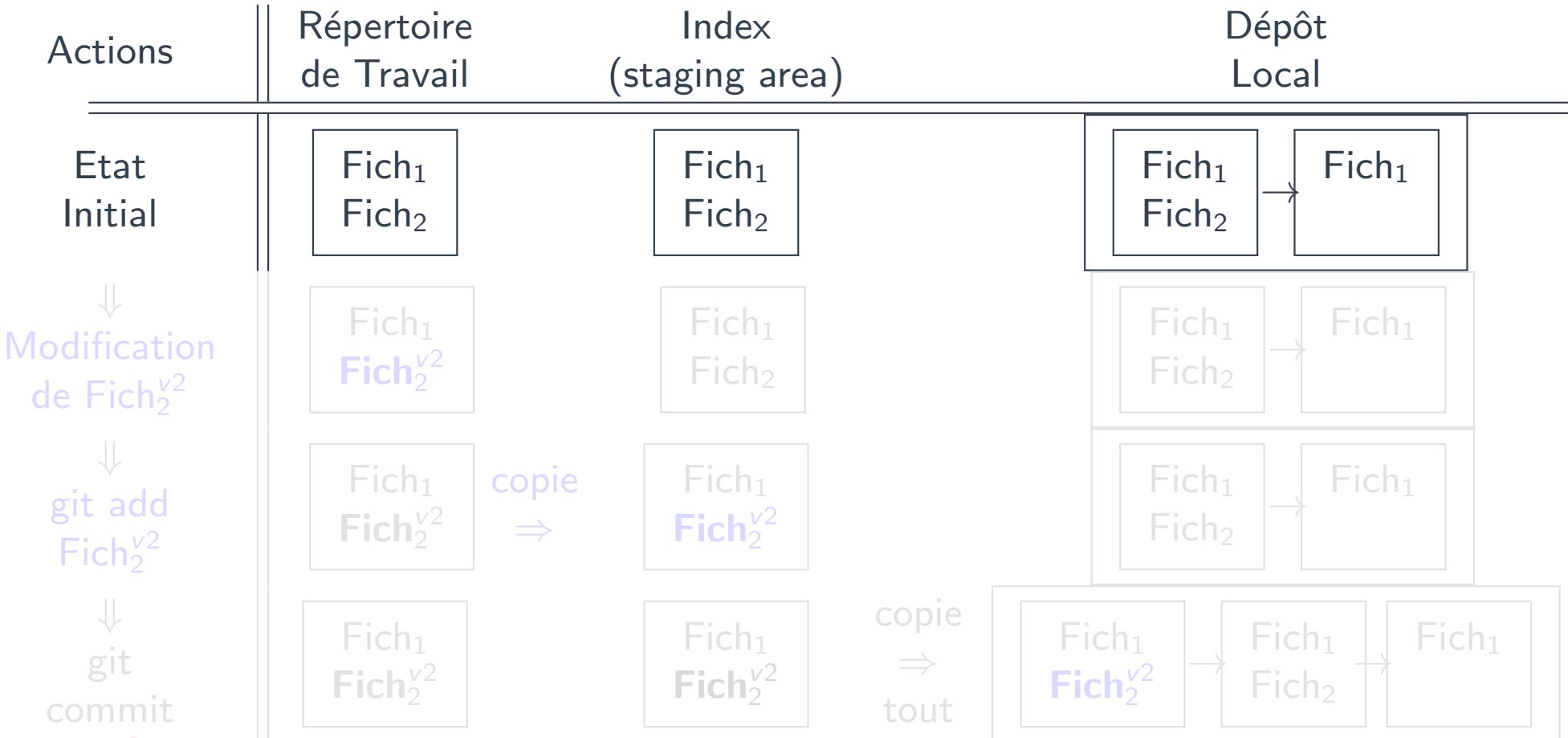
Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : création de Fichier$_2$ $\rightarrow$ git add $\rightarrow$ git commit

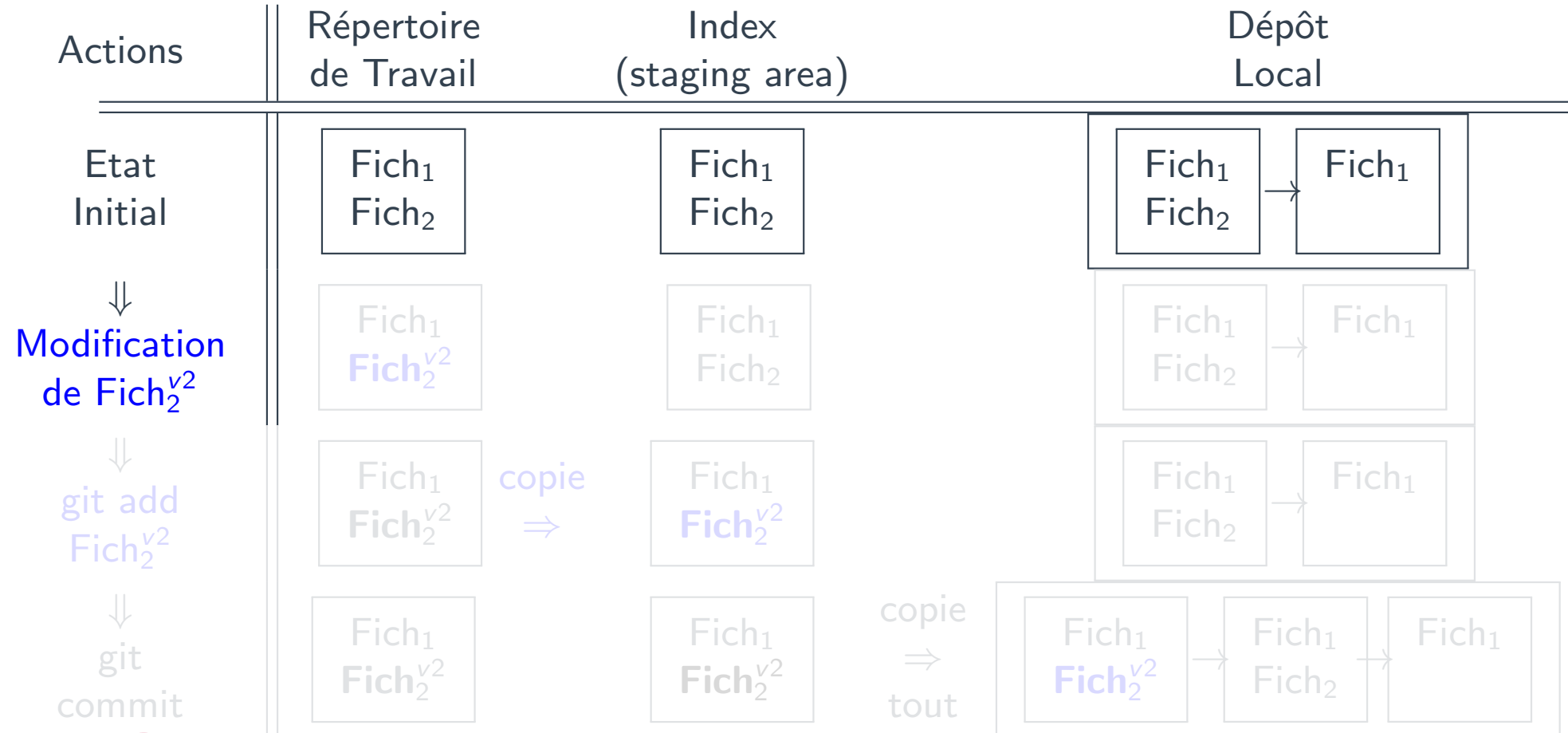


Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit

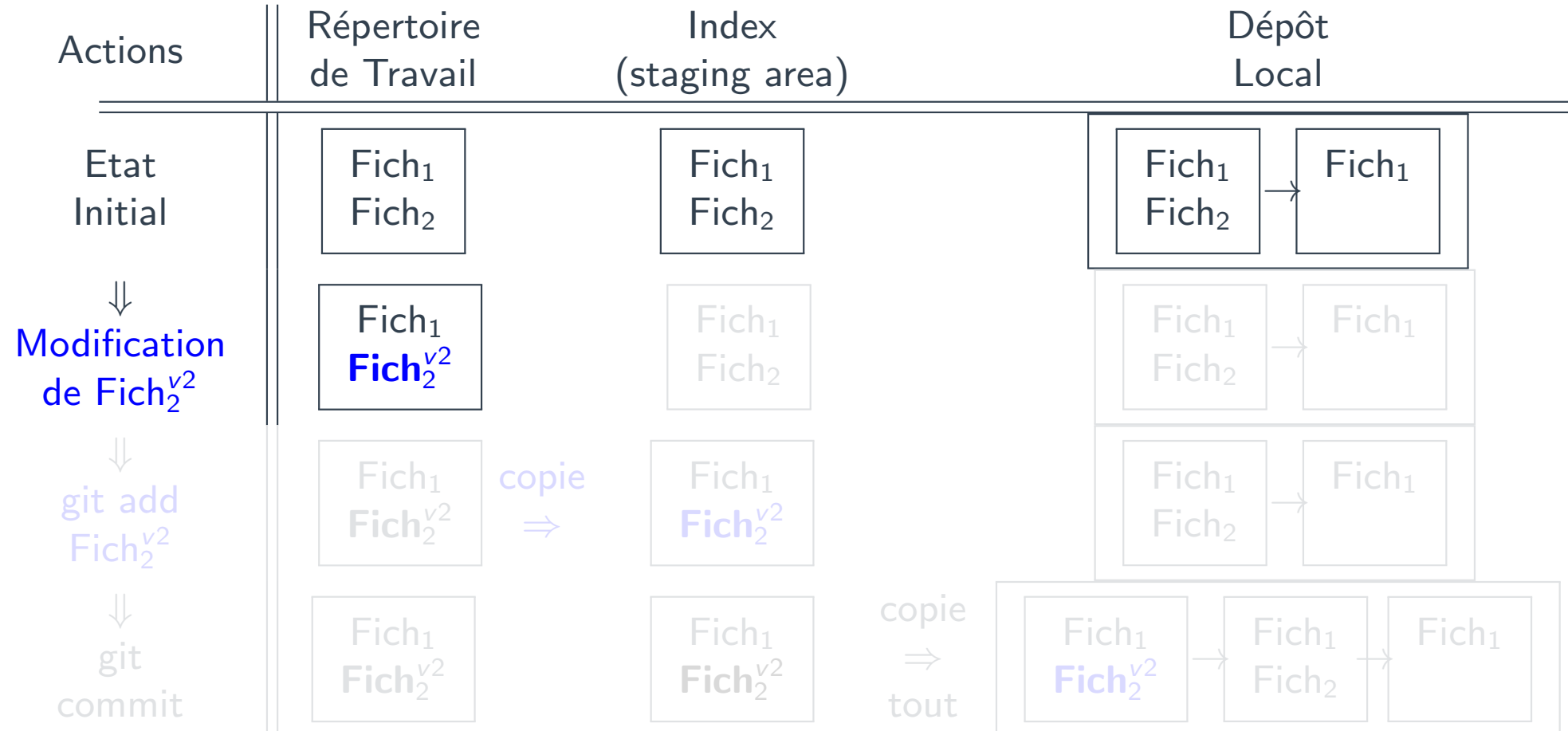


fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit



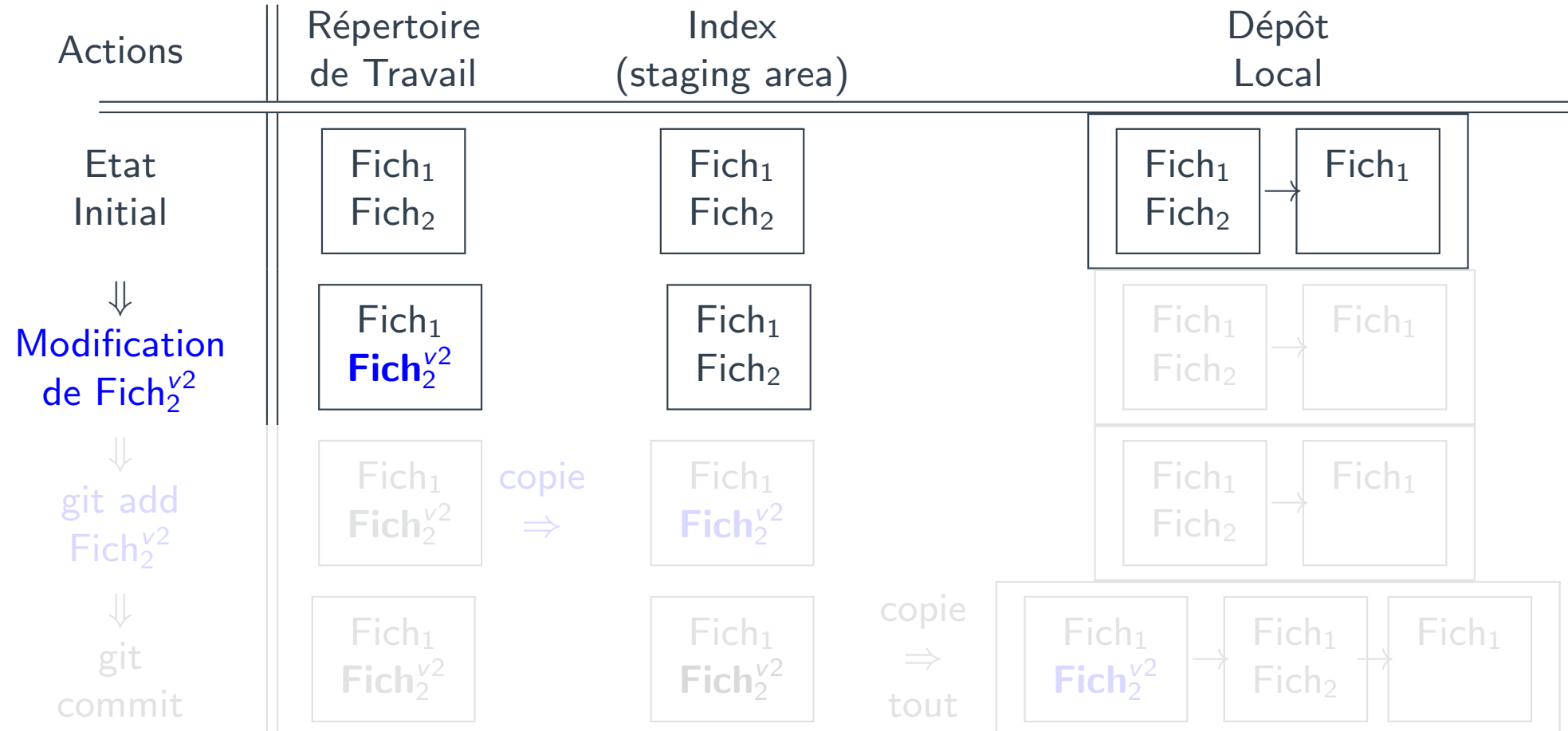
Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit



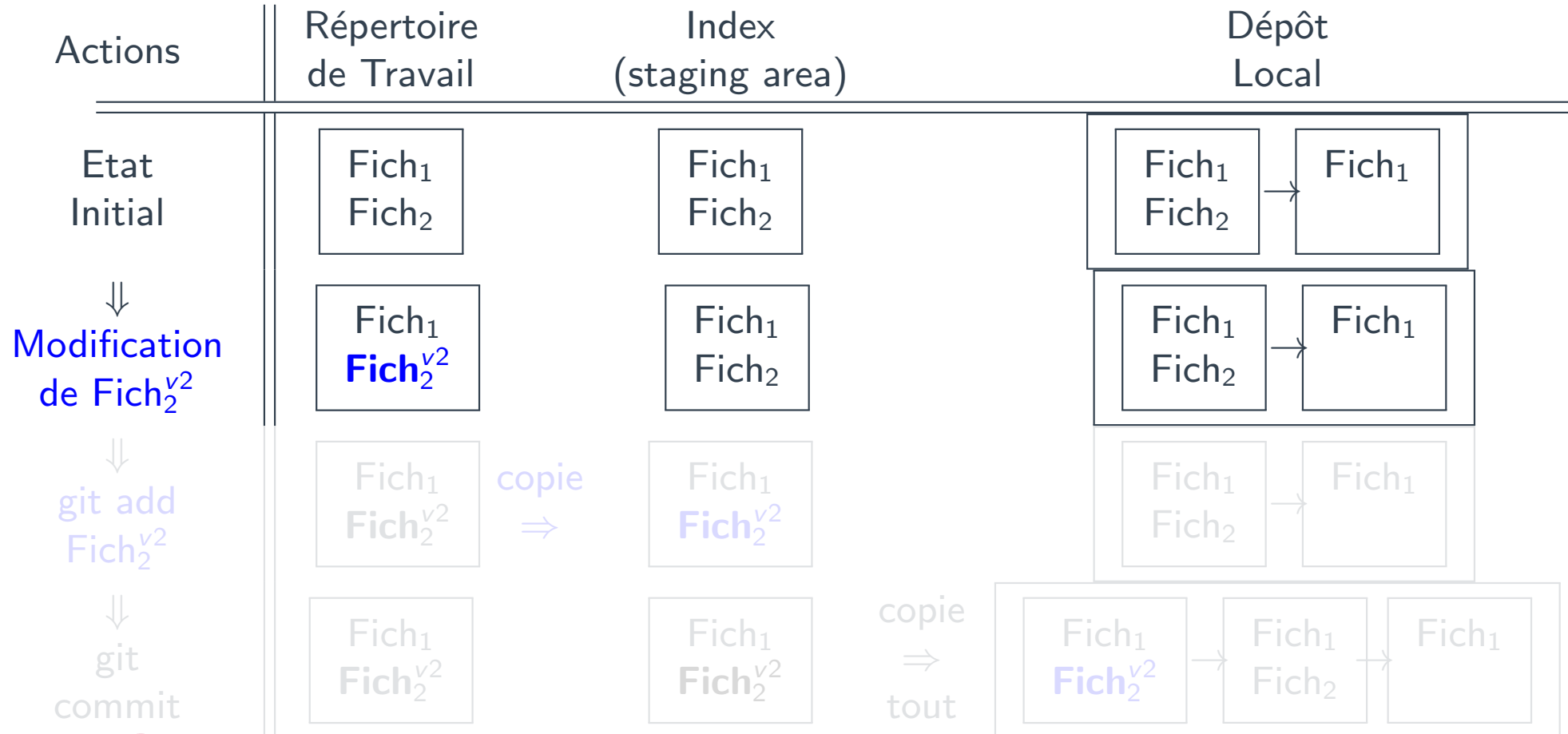
Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit



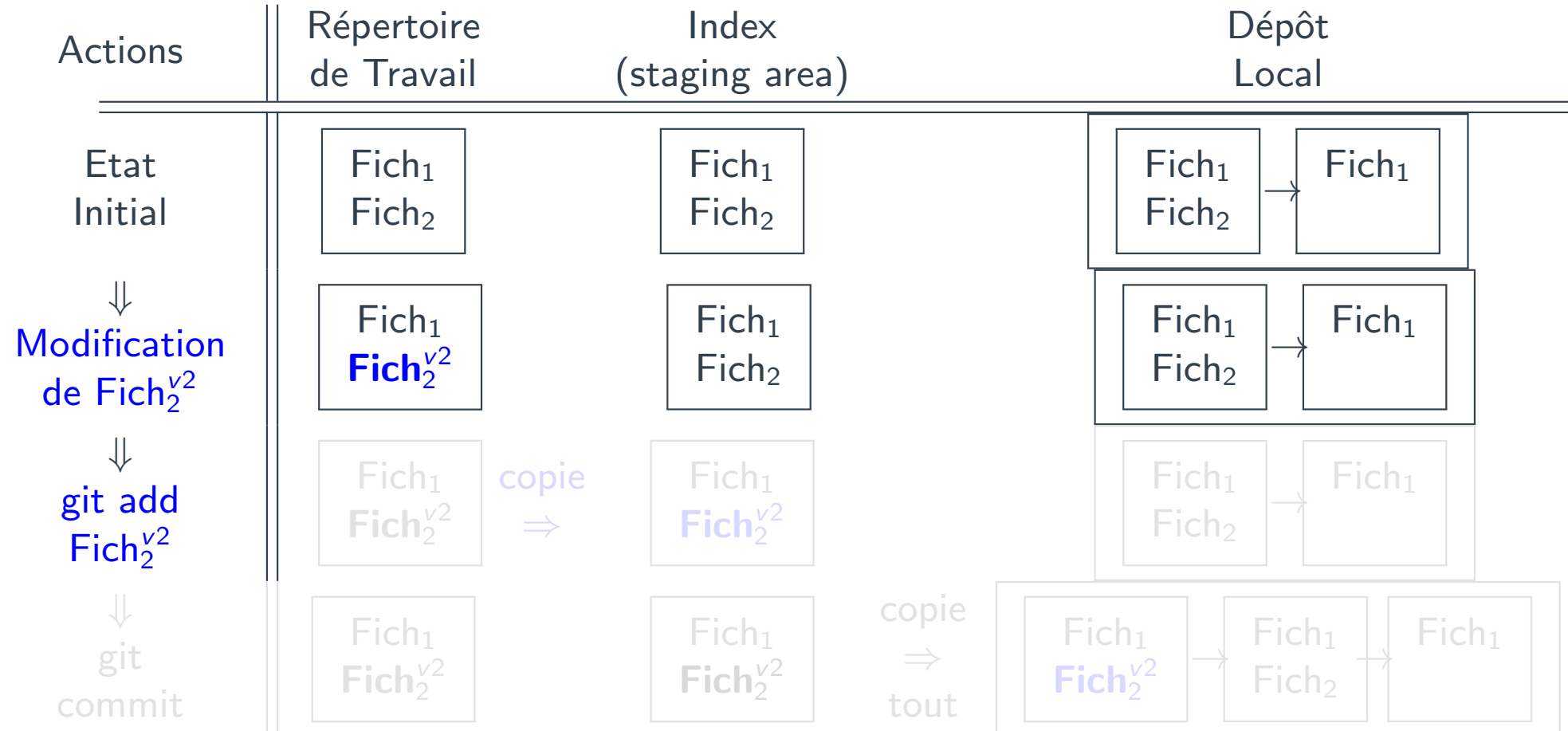
Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit

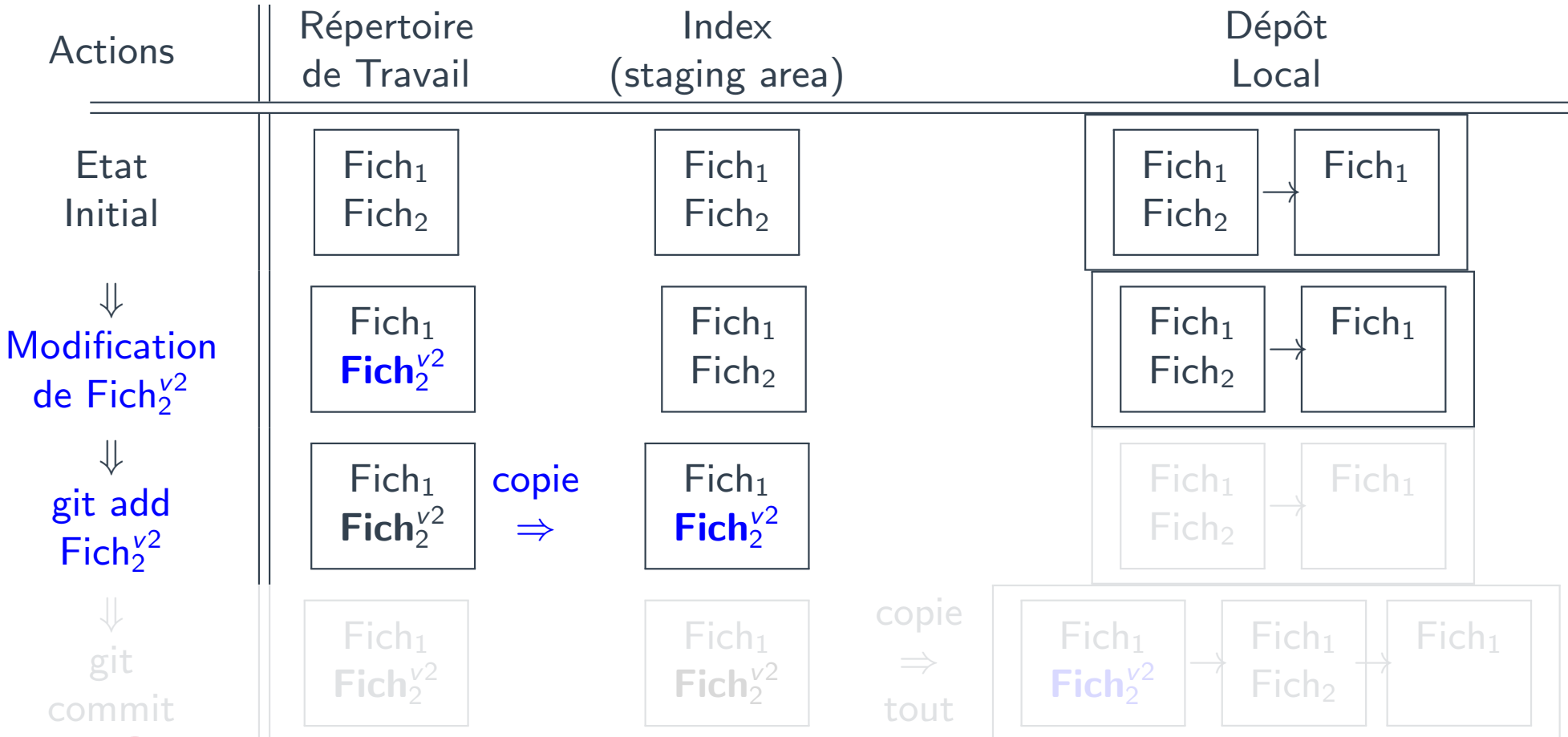


Institut de Mathématiques de Bordeaux

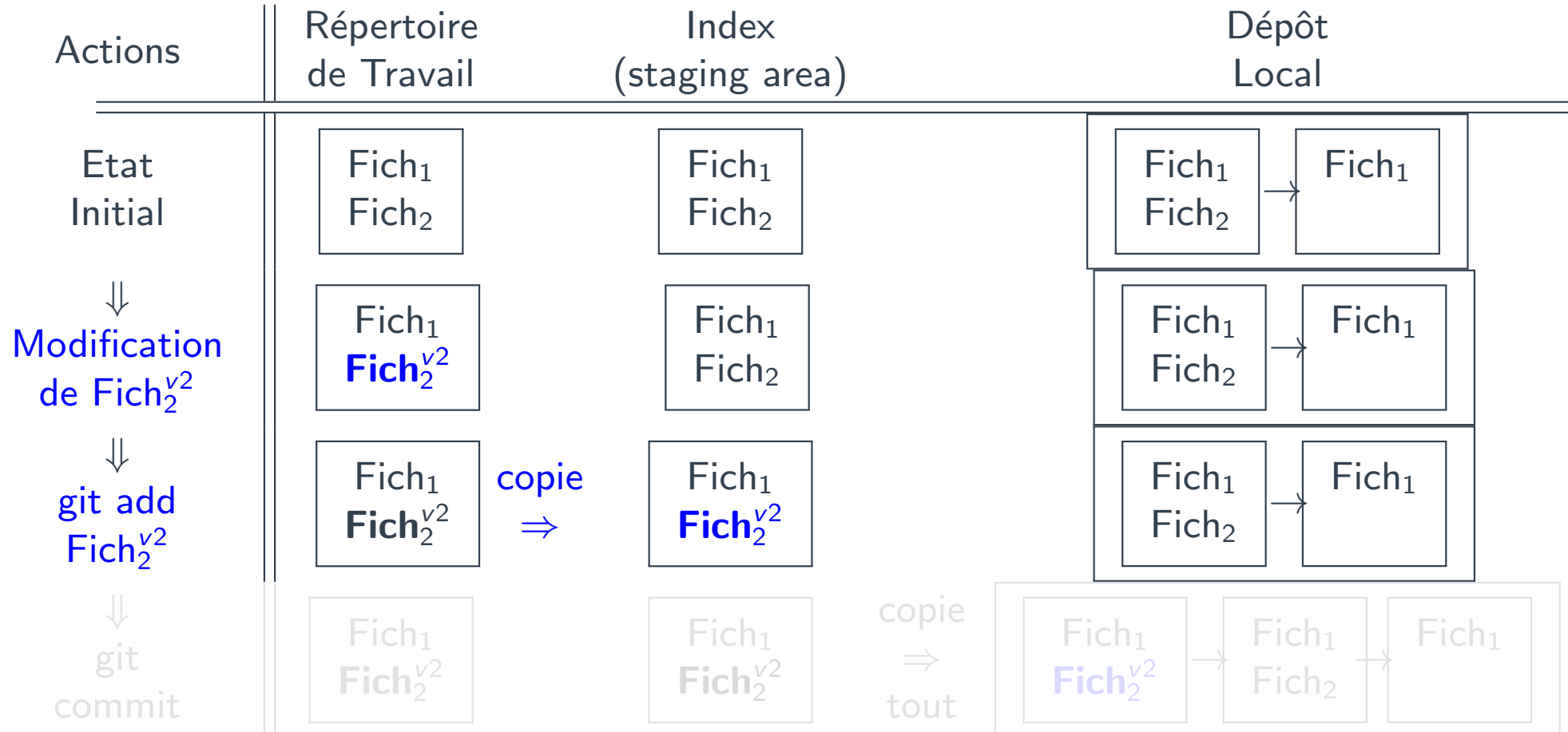
fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit



fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit

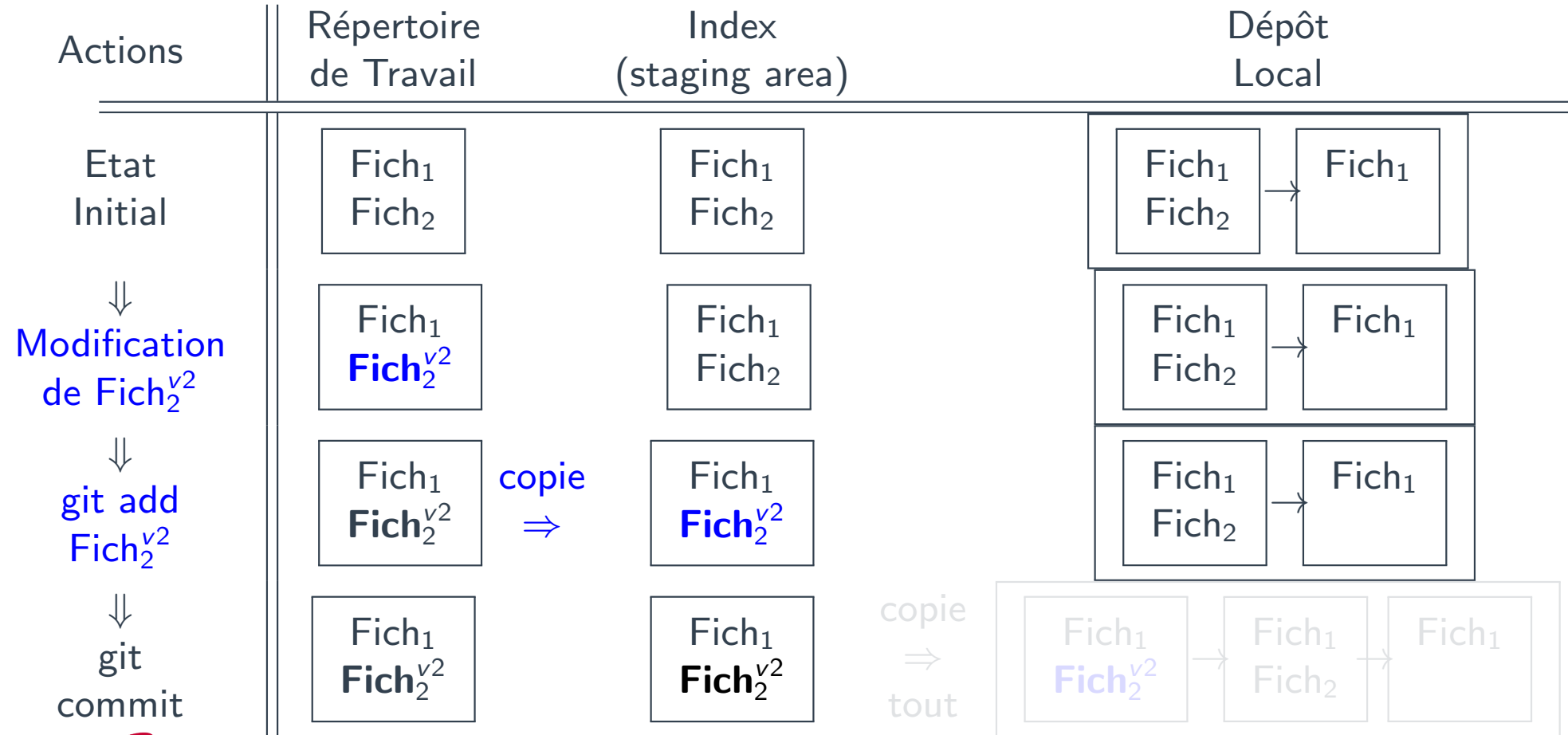


fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit

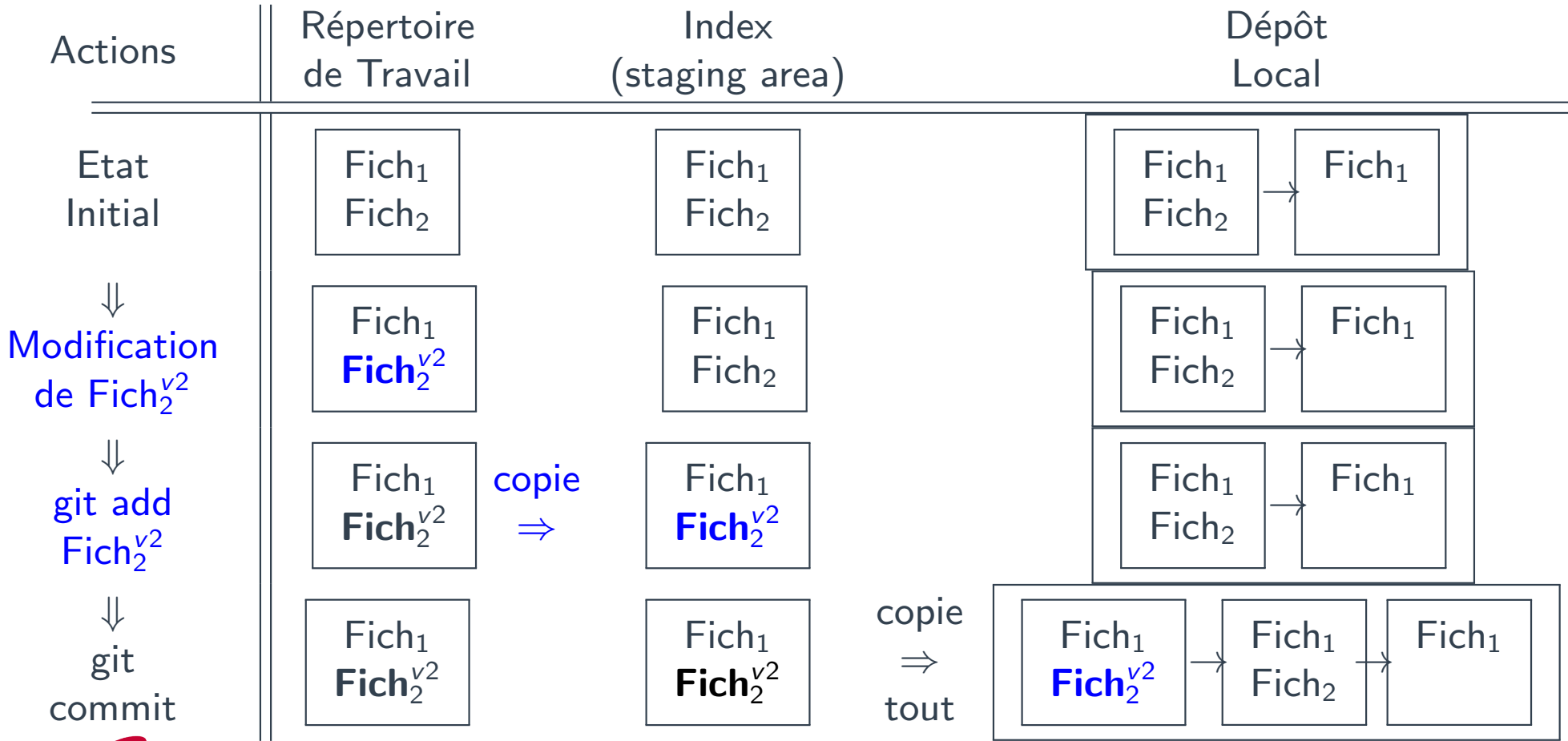


Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : modification de $Fich_2$ → git add → git commit



fonctionnement des 3 zones locales : modification de $Fich_2$ $\rightarrow$ git add $\rightarrow$ git commit



Institut de Mathématiques de Bordeaux

Première session classique (commandes)

- ||| créer un dépôt git dans le répertoire MonProjet :
- **git clone git@plmlab.math.cnrs.fr:VotreLogin/MonProjet**
- **cd MonProjet**
- ...création et modification de fichiers...
- **git add ...fichiers...**
- **git commit -m "modification 1"**
- ...création et modification de fichiers...
- **git add ...fichiers...**
- **git commit -m "modification 2"**
- etc...

Notes concernant les lignes de commandes des TP

Pour simuler les éditions de fichiers, dans les exemples qui suivent nous allons utiliser la commande "echo" avec ">" ou ">>" :

- **echo "zap-tout" > fichier**
- => cela **remplace** le contenu du fichier par la seule ligne 'zap-tout'

- **echo "add-line" >> fichier**
- => cela **ajoute** la ligne "add-line" à la fin du fichier

Cela permet d'avoir des exemples d'enchaînements de commandes git facilement re-jouables, de façon à pouvoir observer, étape par étape, ce qu'il se passe, tester des variantes, refaire, ... jusqu'à bien comprendre le fonctionnement des commandes.

TP 1 : j'organise mon anniversaire !

```
# créer un depot git "MonAnniv" et le cloner :
   git clone git@plmlab.math.cnrs.fr:VotreLogin/MonAnniv
   cd MonAnniv
# creer un fichier pour lister les invites (un par ligne)
   echo Laurent > Invitations.txt
   git add Invitations.txt
   git commit -m "invitation lolo"
# creer un fichier pour les idees de cadeaux (un par ligne)
   echo Montre > IdeesCadeaux.txt
   git add IdeesCadeaux.txt
   git commit -m "premieres idees"
# mise a jour des 2 fichiers dans un meme commit
   echo Philippe >> Invitations.txt
   echo Khodor >> Invitations.txt
   echo SmartPhone >> IdeesCadeaux.txt
   git add IdeesCadeaux.txt Invitations.txt
   git commit -m "invitation fifi koko et complement idees"
```

Objectif 2 : Travailler seul, en local, se repérer dans les versions

- Créer un dépôt

- Éditer des fichiers
- Enregistrer les changements

- **Afficher l'état courant**
- **Lister les versions enregistrées**
- **Afficher les différences entre 2 versions**

Deuxième session classique (commandes)

- ||| on se place dans un dépôt existant
- `cd MonProjet`
- **git status**
- ...modification de fichiers...
- `git add ...fichiers...`
- `git commit -m "modification 1"`
- ...modification de fichiers...
- **git status**
- **git diff**
- `git add ...fichiers...`
- **git diff**
- `git commit -m "modification 2"`
- **git log**
- **git diff REF**

Institut de Mathématiques de Bordeaux

git status : description de l'état des modifications en cours

git status affiche l'état courant de votre répertoire de travail et de l'index (modifications en cours) sur 3 paragraphes (présents que si nécessaire) :

1. "**Modifications qui seront validées**" = les fichiers modifiés qui sont déjà dans l'index = modifications actuellement prévues pour la prochaine version
   - *vous avez fait un git add sur ces fichiers*
   - un fichier de l'index est considéré comme modifié s'il est différent de la dernière version sauvegardée (commit) qui sert de référence
2. "**Modifications qui ne seront pas validées**" = les fichiers modifiés dans le répertoire de travail non encore placés dans l'index
   - *vous n'avez pas encore fait de git add, mais ces fichiers existent dans une précédente version*
   - *ou vous avez fait git add mais encore re-modifié le fichier ensuite*
   - un fichier de travail est considéré comme modifié s'il est différent de la version de l'index.
3. "**Fichiers non suivis**" = les fichiers qui ne sont pas (encore) connus de git mais néanmoins présents dans le répertoire de travail

git status : description de l'état des modifications en cours

Démo !

fonctionnement des 3 zones locales : **git status** ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
Modification $Fichier_2$	$Fichier_1$ **$Fichier_2^{v2}$**	←modif non validée	$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git add $Fichier_2^{v2}$	$Fichier_1$ **$Fichier_2^{v2}$**	copie ⇒	$Fichier_1$ **$Fichier_2^{v2}$**	modif sera ←validée copie	$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git commit	$Fichier_1$ **$Fichier_2^{v2}$**		$Fichier_1$ **$Fichier_2^{v2}$**	⇒ ⇒	$Fichier_1$ -(Fichier **$Fichier_2^{v2}$**-(Fichier
⇓					
Création $Fichier_3$	$Fichier_1$ $Fichier_2^{v2}$ $Fichier_3$	non ←suivi	$Fichier_1$ $Fichier_2^{v2}$	⇒ ⇒	$Fichier_1$ -(Fichier $Fichier_2^{v2}$-(Fichier

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : **git status** ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
Modification $Fichier_2$	$Fichier_1$ **$Fichier_2^{v2}$**	←modif non validée	$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git add $Fichier_2^{v2}$	$Fichier_1$ **$Fichier_2^{v2}$**	copie ⇒	$Fichier_1$ **$Fichier_2^{v2}$**	modif sera ←validée copie	$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git commit	$Fichier_1$ **$Fichier_2^{v2}$**		$Fichier_1$ **$Fichier_2^{v2}$**	⇒ ⇒	$Fichier_1$ -(Fichier **$Fichier_2^{v2}$**-(Fichier
⇓					
Création $Fichier_3$	$Fichier_1$ $Fichier_2^{v2}$ $Fichier_3$	non ←suivi	$Fichier_1$ $Fichier_2^{v2}$	⇒ ⇒	$Fichier_1$ -(Fichier $Fichier_2^{v2}$-(Fichier$_2$

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : **git status** ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
Modification $Fichier_2$	$Fichier_1$ **$Fichier_2^{v2}$**	←modif non validée	$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git add $Fichier_2^{v2}$	$Fichier_1$ **$Fichier_2^{v2}$**	copie ⇒	$Fichier_1$ **$Fichier_2^{v2}$**	modif sera ←validée	$Fichier_1$-(Fi $Fichier_2$-(
⇓				copie	
git commit	$Fichier_1$ **$Fichier_2^{v2}$**		$Fichier_1$ **$Fichier_2^{v2}$**	⇒ ⇒	$Fichier_1$ -(Fichier **$Fichier_2^{v2}$**-(Fichier
⇓					
Création $Fichier_3$	$Fichier_1$ $Fichier_2^{v2}$ $Fichier_3$	non ←suivi	$Fichier_1$ $Fichier_2^{v2}$	⇒ ⇒	$Fichier_1$ -(Fichier $Fichier_2^{v2}$-(Fichier$_2$

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : **git status** ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
Modification $Fichier_2$	$Fichier_1$ **$Fichier_2^{v2}$**	←modif non validée	$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git add $Fichier_2^{v2}$	$Fichier_1$ **$Fichier_2^{v2}$**	copie ⇒	$Fichier_1$ **$Fichier_2^{v2}$**	modif sera ←validée copie	$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git commit	$Fichier_1$ **$Fichier_2^{v2}$**		$Fichier_1$ **$Fichier_2^{v2}$**	⇒ ⇒	$Fichier_1$ -(Fichier **$Fichier_2^{v2}$**-(Fichier
⇓					
Création $Fichier_3$	$Fichier_1$ $Fichier_2^{v2}$ $Fichier_3$	non ←suivi	$Fichier_1$ $Fichier_2^{v2}$	⇒ ⇒	$Fichier_1$ -(Fichier $Fichier_2^{v2}$-(Fichier$_2$

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : **git status** ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
Modification $Fichier_2$	$Fichier_1$ **$Fichier_2^{v2}$**	←modif non validée	$Fichier_1$ $Fichier_2$		$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git add $Fichier_2^{v2}$	$Fichier_1$ **$Fichier_2^{v2}$**	copie ⇒	$Fichier_1$ **$Fichier_2^{v2}$**	modif sera ←validée copie	$Fichier_1$-(Fi $Fichier_2$-(
⇓					
git commit	$Fichier_1$ **$Fichier_2^{v2}$**		$Fichier_1$ **$Fichier_2^{v2}$**	⇒ ⇒	$Fichier_1$ -(Fichier **$Fichier_2^{v2}$**-(Fichier
⇓					
Création $Fichier_3$	$Fichier_1$ $Fichier_2^{v2}$ $Fichier_3$	non ←suivi	$Fichier_1$ $Fichier_2^{v2}$	⇒ ⇒	$Fichier_1$ -(Fichier $Fichier_2^{v2}$-(Fichier

Institut de Mathématiques de Bordeaux

TP 2.1 : tester **git status** (vue d'ensemble)

- reprendre le TP 1 en créant un nouveau répertoire MonAnniv2
- reprendre toutes les étapes mais en exécutant un "git status" après chaque commande echo, git add ou git commit
- objectif: bien observer les changements d'états des fichiers dans les 3 catégories (non suivi, modifié, pris en compte pour la prochaine modification)
- petit à petit, essayez de deviner, avant d'exécuter chaque "git status", quelle sera son résultat

TP 2.1 : tester **git status**

```
# créer un depot git dans le repertoire MonAnniv2 :
   git clone git@plmlab.math.cnrs.fr:VotreLogin/MonAnniv MonAnniv2
   cd MonAnniv2
# creer un fichier pour lister les invites (un par ligne)
   echo "Laurent" > Invitations.txt ; git status
   git add Invitations.txt ; git status
   git commit -m "invitation lolo"  ; git status
# creer un fichier pour les idees de cadeaux (une par ligne)
   echo "Montre" > IdeesCadeaux.txt ; git status
   git add IdeesCadeaux.txt  ; git status
   git commit -m "premieres idees"  ; git status
# mise a jour des 2 fichiers dans un meme commit
   echo Philippe >> Invitations.txt ; git status
   echo Khodor >> Invitations.txt  ; git status
   echo "SmartPhone" >> IdeesCadeaux.txt ; git status
   git add IdeesCadeaux.txt Invitations.txt ; git status
   git commit -m "invitation fifi koko et complement idees"
   git status
```

Institut de Mathématiques de Bordeaux

TP 2.1 : tester **git status** (solution)

```
# créer un depot git dans le repertoire MonAnniv2 :
   git clone git@plmlab.math.cnrs.fr:VotreLogin/MonAnniv MonAnniv2
   cd MonAnniv2
# creer un fichier pour lister les invites (un par ligne)
   echo "Laurent" > Invitations.txt ; git status # Untracked
   git add Invitations.txt ; git status          # ModifiedOK
   git commit -m "invitation lolo"  ; git status # ---
# creer un fichier pour les idees de cadeaux (une par ligne)
   echo "Montre" > IdeesCadeaux.txt ; git status # Untracked
   git add IdeesCadeaux.txt  ; git status        # ModifiedOK
   git commit -m "premieres idees"  ; git status # ---
# mise a jour des 2 fichiers dans un meme commit
   echo Philippe >> Invitations.txt ; git status # Modified
   echo Khodor >> Invitations.txt  ; git status  # Modified
   echo "SmartPhone" >> IdeesCadeaux.txt ; git status # Mofidied x2
   git add IdeesCadeaux.txt Invitations.txt ; git status #ModifiedOK x2
   git commit -m "invitation fifi koko et complement idees" # ---
   git status                                               # ---
```

Institut de Mathématiques de Bordeaux

git log : afficher l'historique des versions

- affiche l'ensemble des versions en commençant par la plus récentes et en remontant le temps
- affiche pour chaque version (commit)
  - le numéro de référence du commit (hash)
  - le nom de l'auteur
  - la date
  - le commentaire associé

git log : afficher l'historique des versions

Démo !

fonctionnement des 3 zones locales : git log ?

Actions	Répertoire de Travail	Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$	$Fichier_1$		$Fichier_1$-($Fichier_1$)
	$Fichier_2$	$Fichier_2$		$Fichier_2$-( )
⇓			Versions	HASHY - HASHZ
Modification	$Fichier_1$	$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2$	**$Fichier_2^{v2}$**	$Fichier_2$		$Fichier_2$-( )
⇓			Versions	HASHY - HASHZ
git add	$Fichier_1$	$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2^{v2}$	**$Fichier_2^{v2}$** ⇒	**$Fichier_2^{v2}$**		$Fichier_2$-( )
⇓			Versions	HASHY - HASHZ
git	$Fichier_1$	$Fichier_1$	⇒	$Fichier_1$ -($Fichier_1$)-(Fichier
commit	**$Fichier_2^{v2}$**	**$Fichier_2^{v2}$**	⇒	**$Fichier_2^{v2}$**-($Fichier_2$)-(
			Versions	HASHX - HASHY - HASH

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : git log ?

Actions	Répertoire de Travail	Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$	$Fichier_1$		$Fichier_1$-($Fichier_1$)
	$Fichier_2$	$Fichier_2$		$Fichier_2$-( )
⇓			Versions	HASHY - HASHZ
Modification	$Fichier_1$	$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2$	**$Fichier_2^{v2}$**	$Fichier_2$		$Fichier_2$-( )
⇓			Versions	HASHY - HASHZ
git add	$Fichier_1$	$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2^{v2}$	**$Fichier_2^{v2}$** ⇒	**$Fichier_2^{v2}$**		$Fichier_2$-( )
⇓			Versions	HASHY - HASHZ
git	$Fichier_1$	$Fichier_1$	⇒	$Fichier_1$ -($Fichier_1$)-(Fichier
commit	**$Fichier_2^{v2}$**	**$Fichier_2^{v2}$**	⇒	**$Fichier_2^{v2}$**-($Fichier_2$)-(
			Versions	HASHX - HASHY - HASH

fonctionnement des 3 zones locales : git log ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$		$Fichier_1$		$Fichier_1$-($Fichier_1$)
	$Fichier_2$		$Fichier_2$		$Fichier_2$-( )
⇓				Versions	HASHY - HASHZ
Modification	$Fichier_1$		$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2$	**$Fichier_2^{v2}$**		$Fichier_2$		$Fichier_2$-( )
⇓				Versions	HASHY - HASHZ
git add	$Fichier_1$		$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2^{v2}$	**$Fichier_2^{v2}$**	⇒	**$Fichier_2^{v2}$**		$Fichier_2$-( )
⇓				Versions	HASHY - HASHZ
git	$Fichier_1$		$Fichier_1$	⇒	$Fichier_1$ -($Fichier_1$)-(Fichier
commit	**$Fichier_2^{v2}$**		**$Fichier_2^{v2}$**	⇒	**$Fichier_2^{v2}$**-($Fichier_2$)-(
				Versions	HASHX - HASHY - HASH

Institut de Mathématiques de Bordeaux

fonctionnement des 3 zones locales : git log ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$		$Fichier_1$		$Fichier_1$-($Fichier_1$)
	$Fichier_2$		$Fichier_2$		$Fichier_2$-( )
⇓				Versions	HASHY - HASHZ
Modification	$Fichier_1$		$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2$	**$Fichier_2^{v2}$**		$Fichier_2$		$Fichier_2$-( )
⇓				Versions	HASHY - HASHZ
git add	$Fichier_1$		$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2^{v2}$	**$Fichier_2^{v2}$**	⇒	**$Fichier_2^{v2}$**		$Fichier_2$-( )
⇓				Versions	HASHY - HASHZ
git	$Fichier_1$		$Fichier_1$	⇒	$Fichier_1$ -($Fichier_1$)-(Fichier
commit	**$Fichier_2^{v2}$**		**$Fichier_2^{v2}$**	⇒	**$Fichier_2^{v2}$**-($Fichier_2$)-(
				Versions	HASHX - HASHY - HASH

Institut de Mathématiques de Bordeaux

TP 2.2 : tester **git log** (vue d'ensemble)

- reprendre le TP 1 en créant le répertoire MonAnniv3
- étape par étape en exécutant un "git log" après chaque git commit

TP 2.2 : tester **git log**

```
# creer un depot git dans le repertoire MonAnniv3 :
   git clone git@plmlab.math.cnrs.fr:LoGiN/MonAnniv MonAnniv3
   cd MonAnniv3
# creer un fichier pour lister les invites (un par ligne)
   echo "Laurent" > Invitations.txt
   git add Invitations.txt
   git commit -m "invitation lolo"  ; git log
# creer un fichier pour les idees de cadeaux (un par ligne)
   echo "Montre" > IdeesCadeaux.txt
   git add IdeesCadeaux.txt
   git commit -m "premieres idees"  ; git log
# mise a jour des 2 fichiers dans un meme commit
   echo Philippe >> Invitations.txt
   echo Khodor >> Invitations.txt
   echo "SmartPhone" >> IdeesCadeaux.txt
   git add IdeesCadeaux.txt Invitations.txt
   git commit -m "invitation fifi koko et complement idees"
   git log
```

Institut de Mathématiques de Bordeaux

git diff : afficher les différences entre versions

- **git diff** : affiche les différences, entre l'index et le répertoire de travail, de vos modifications en cours non encore ajoutée à l'index = *en partant de l'index, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH** : affiche les différences entre cette version SRCHASH et celle du répertoire de travail = *en partant de cette version, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH DSTHASH** : affiche les différences entre les version SRCHASH et DSTHASH = *en partant de SRCHASH, que faut il modifier pour aboutir à la version DSTHASH*

\+ : ligne à ajouter (vert) - : ligne à supprimer (rouge)

mémo :

```
git diff [ <source=index> [ <destination=courant> ] ]
```

Institut de Mathématiques de Bordeaux

git diff : afficher les différences entre versions

- **git diff** : affiche les différences, entre l'index et le répertoire de travail, de vos modifications en cours non encore ajoutée à l'index = *en partant de l'index, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH** : affiche les différences entre cette version SRCHASH et celle du répertoire de travail = *en partant de cette version, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH DSTHASH** : affiche les différences entre les version SRCHASH et DSTHASH = *en partant de SRCHASH, que faut il modifier pour aboutir à la version DSTHASH*

\+ : ligne à ajouter (vert) - : ligne à supprimer (rouge)

mémo :

```
git diff [ <source=index> [ <destination=courant> ] ]
```

Institut de Mathématiques de Bordeaux

git diff : afficher les différences entre versions

- **git diff** : affiche les différences, entre l'index et le répertoire de travail, de vos modifications en cours non encore ajoutée à l'index = *en partant de l'index, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH** : affiche les différences entre cette version SRCHASH et celle du répertoire de travail = *en partant de cette version, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH DSTHASH** : affiche les différences entre les version SRCHASH et DSTHASH = *en partant de SRCHASH, que faut il modifier pour aboutir à la version DSTHASH*

\+ : ligne à ajouter (vert) - : ligne à supprimer (rouge)

mémo :

```
git diff [ <source=index> [ <destination=courant> ] ]
```

Institut de Mathématiques de Bordeaux

git diff : afficher les différences entre versions

Démo !

fonctionnement des 3 zones locales : git diff ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$		$Fichier_1$-($Fichier_1$) $Fichier_2$-( )
⇓					
Modification $Fichier_2$	$Fichier_1$ **$Fichier_2^{v2}$**	git ←diff←	$Fichier_1$ $Fichier_2$		$Fichier_1$-($Fichier_1$) $Fichier_2$-( )
⇓					
git add $Fichier_2^{v2}$	$Fichier_1$ **$Fichier_2^{v2}$**	⇒	$Fichier_1$ **$Fichier_2^{v2}$**		$Fichier_1$-($Fichier_1$) $Fichier_2$-( )
⇓					
git commit	$Fichier_1$ **$Fichier_2^{v2}$**		$Fichier_1$ **$Fichier_2^{v2}$**	⇒ ⇒	$Fichier_1$ -($Fichier_1$)-($Fichier_1$) **$Fichier_2^{v2}$**-($Fichier_2$)-( )
					Versions HASHX - HASHY - HASHZ

git diff HASHZ HASHY ?

TP 2.3 : tester **git diff**

- reprendre le TP 2.2 précédent (MonAnniv3) dans son état final
- lister les versions avec **git log**
- afficher les différences entre :
  - la version courante et la 1ère version enregistrée (la plus ancienne) avec **git diff HASHASHASHASH1**
  - la version courante et la 2ème version enregistrée avec **git diff HASHASHASHASH2**
  - entre la 1ere version et la 2ème version avec **git diff HASHASHASHASH1 HASHASHASHASH2**
- modifiez un fichier du répertoire de travail et afficher, avec **git diff** les différences introduites

Objectif 3 : Travailler avec un dépôt distant
Session type

- **créer un dépôt distant et le cloner localement** pour en avoir une copie intégrale
- Répéter autant que nécessaire :
  - modifier des fichiers
  - "commiter" localement les modifications
  - ...
  - modifier des fichiers
  - "commiter" localement les modifications
  - ...
  - **propager toutes ces modifications (commits) vers le dépôt distant : git push**

fonctionnement des 4 zones : git clone & git push ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local		Dépôt Distant
Etat Initial							F_1-(F_1) F_2-( )
⇓							
git clone ...	F_1 F_2	⇐ ⇐	F_1 F_2	⇐ ⇐	F_1-(F_1) F_2-( )	⇐ ⇐	F_1-(F_1) F_2-( )
⇓							
modif F_2+ git add F_2^{v2}	F_1 F_2^{v2}	 ⇒	F_1 F_2^{v2}		F_1-(F_1) F_2-( )		F_1-(F_1) F_2-( )
⇓							
git commit	F_1 F_2^{v2}		F_1 F_2^{v2}	⇒ ⇒	F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )		F_1-(F_1) F_2-( )
⇓							
git push	F_1 F_2^{v2}		F_1 F_2^{v2}		F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )	⇒ ⇒	F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )

Institut de Mathématiques de Bordeaux

fonctionnement des 4 zones : git clone & git push ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local		Dépôt Distant
Etat Initial							F_1-(F_1) F_2-( )
⇓							
git clone ...	F_1 F_2	⇐ ⇐	F_1 F_2	⇐ ⇐	F_1-(F_1) F_2-( )	⇐ ⇐	F_1-(F_1) F_2-( )
⇓							
modif F_2+ git add F_2^{v2}	F_1 $\mathbf{F_2^{v2}}$	⇒	F_1 $\mathbf{F_2^{v2}}$		F_1-(F_1) F_2-( )		F_1-(F_1) F_2-( )
⇓							
git commit	F_1 F_2^{v2}		F_1 F_2^{v2}	⇒ ⇒	F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )		F_1-(F_1) F_2-( )
⇓							
git push	F_1 F_2^{v2}		F_1 F_2^{v2}		F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )	⇒ ⇒	F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )

Institut de Mathématiques de Bordeaux

fonctionnement des 4 zones : git clone & git push ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local		Dépôt Distant
Etat Initial							F_1-(F_1) F_2-( )
⇓							
git clone ...	F_1 F_2	⇐ ⇐	F_1 F_2	⇐ ⇐	F_1-(F_1) F_2-( )	⇐ ⇐	F_1-(F_1) F_2-( )
⇓							
modif F_2+ git add F_2^{v2}	F_1 $\mathbf{F_2^{v2}}$	 ⇒	F_1 $\mathbf{F_2^{v2}}$		F_1-(F_1) F_2-( )		F_1-(F_1) F_2-( )
⇓							
git commit	F_1 $\mathbf{F_2^{v2}}$		F_1 $\mathbf{F_2^{v2}}$	⇒ ⇒	F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )		F_1-(F_1) F_2-( )
⇓							
git push	F_1 F_2^{v2}		F_1 F_2^{v2}		F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )	⇒ ⇒	F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )

Institut de Mathématiques de Bordeaux

fonctionnement des 4 zones : git clone & git push ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local		Dépôt Distant
Etat Initial							F_1-(F_1) F_2-( )
⇓							
git clone ...	F_1 F_2	⇐ ⇐	F_1 F_2	⇐ ⇐	F_1-(F_1) F_2-( )	⇐ ⇐	F_1-(F_1) F_2-( )
⇓							
modif F_2+ git add F_2^{v2}	F_1 $\mathbf{F_2^{v2}}$	 ⇒	F_1 $\mathbf{F_2^{v2}}$		F_1-(F_1) F_2-( )		F_1-(F_1) F_2-( )
⇓							
git commit	F_1 $\mathbf{F_2^{v2}}$		F_1 $\mathbf{F_2^{v2}}$	⇒ ⇒	F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )		F_1-(F_1) F_2-( )
⇓							
git push	F_1 F_2^{v2}		F_1 F_2^{v2}		F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )	⇒ ⇒	F_1 -(F_1)-(F_1) F_2^{v2}-(F_2)-( )

Institut de Mathématiques de Bordeaux

TP 3.0 : Travailler et sauvegarder sur un dépôt distant
git push

- créer un nouveau dépôt MonAnniv
- cloner ce dépôt
- créer les fichiers Invitations.txt et IdeesCadeaux.txt
- les ajouter à l'index
- commiter cette nouvelle (et première) version
- propagez les modifications sur le dépôt distant : **git push**
- créez et remplissez un nouveau fichier IdeesMenu.txt
- faites le nécessaire pour propager cette nouvelle version sur le dépôt distant

TP 3.0 : Travailler et sauvegarder sur un dépôt distant (solution)

```
# créer un depot git dans le repertoire MonAnniv :
   git clone git@plmlab.math.cnrs.fr:VotreLogin/MonAnniv
   cd MonAnniv
# creer les fichiers Invitations.txt et IdeesCadeaux.txt
   echo Laurent > Invitations.txt
   echo Montre > IdeesCadeaux.txt
   git add Invitations.txt IdeesCadeaux.txt
   git commit -m "invitation et cadeaux lolo"
# sauvegarder sur le serveur
   git push
# creer un fichier pour les idees de menus
   echo Couscous >> IdeesMenu.txt
   git add IdeesMenu.txt
   git commit -m "idee menu"
# sauvegarder sur le serveur
   git push
```

Institut de Mathématiques de Bordeaux

Objectif 4 : Travailler depuis 2 ordinateurs
git pull

- **créer un dépôt distant**
- **cloner ce dépôt sur votre ordinateur A**
  - modifier des fichiers
  - propager les modifications sur le dépôt distant avec **git push**
- **depuis un autre ordinateur B : cloner ce dépôt sur votre ordinateur** : on récupère donc la dernière version qui contient les modifications effectuées sur l'ordinateur A
  - modifier des fichiers
  - propager les modifications sur le dépôt distant : **git push**
- **/!/ de retour sur votre ordinateur A... le dépôt n'est plus à jour !! deux solutions :**
  1. la bonne : synchroniser votre dépôt avec le dépôt distant pour récupérer la dernière version : **git pull**
  2. *l'autre : effacer/re-cloner ce dépôt dans un autre répertoire... par très efficace mais utile au début si on est un peu perdu !*

TP 4 : Travailler avec un dépôt distant depuis 2 ordinateurs

- astuce : pour simuler ce qui se passerait avec deux ordinateurs, nous allons cloner le dépôt "MonAnniv" dans deuxième répertoire :
  - git clone git@mongitlab.fr:VotreLogin/MonAnniv MonAnnivB
  - nous avons un nouveau répertoire `MonAnnivB` qui contient la vue du dépôt coté ordinateur B
- ajouter un invité dans le fichier Invitations.txt et propager cette nouvelle version sur le dépôt distant
- revenez sur l'ordinateur A (cd ../MonAnniv)
- **git pull**
- votre répertoire de travail de l'ordinateur A est maintenant synchronisé avec la dernière version

git remote -v : liste les dépôts distants (remote)

- affiche la liste des dépôts distants connectés à votre dépôt local
- /!/ on peut en avoir plusieurs /!/

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

- vous avez synchronisé vos deux ordinateurs avec la dernière version de votre dépôt distant
- vous avez fait des modifications (modifications+commits) depuis votre ordinateur A mais vous n'avez rien propagé sur le dépôt distant
- vous faites maintenant des modifications (modifications+commits) depuis l'ordinateur B et vous les propagez sur le dépôt distant (push)
- quand vous revenez sur l'ordinateur A, vous vous rendez compte que vos modifications n'ont pas été propagées, **git push** fait un message d'erreur car le dépôt distant a évolué, et la version initiale sur laquelle vos modifications ont été élaborées n'est plus la dernière version.
- avant de pouvoir faire **git push** vous devez faire un **git pull** et résoudre les éventuels conflits si vous avez touché aux mêmes fichiers (mêmes zones) depuis les 2 ordinateurs pour des modifications différentes

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Objectif 5 : Résoudre les conflits avec un dépôt distant depuis 2 ordinateurs

sur ordi A :	sur ordi B :	dépôt distant
git clone	git clone	**version d'origine**
git add ...		version d'origine
git commit -m A		version d'origine
((oubli git push))		version d'origine
	git add ...	version d'origine
	git commit -m B	version d'origine
	git push	**version B**
/!/ git push /!/ pb - refus		version B
git pull ((fusion))		version B
((résolution conflit))		version B
git push ((fusion))		**version "A+B"**

Institut de Mathématiques de Bordeaux

Comment résoudre les conflits ?

- en cas de conflits sur un fichier, git combine les deux versions possibles de chaque modifications avec **3 marqueurs** : <<<<<<< ======= >>>>>>>

fichier A :	fichier B :	conflit à résoudre :	final
blabla	blabla	blabla	blabla
versionA	versionB	>>>>>>> HEAD	versionAB
blabla	blabla	versionA	blabla
		=======	
		versionB	
		<<<<<<< HASHHASHHASH	
		blabla	

- vous devez alors :
  - 1- éditer le fichier pour qu'il ressemble à la version que vous voulez (qui peut être une 3ème version très différentes des 2 précédentes),
  - 2- ... et supprimer les marqueurs
  - 3- ajouter dans l'index ce fichier rectifié avec **git add FICHIER**
  - 4- faire un nouveau commit de fusion avec **git commit**
  - ... et le pousser **git push**
- à tout moment, **git status** vous dit quoi faire !

TP 5 Créer et résoudre un conflit

- cloner un même dépôt git distant dans deux répertoires A et B
- dans le répertoire A, créer un fichier MonConflit.txt avec 3 lignes de texte
- propagez cette modifications dans le dépôt distant
- allez dans le répertoire B et synchronisez le dépôt
- modifiez la 2ieme ligne du fichier MonConflit.txt
- propagez cette modifications dans le dépôt distant
- revenez dans A et modifiez différemment la deuxième ligne du fichier
- tentez de propager vos modifications sur le dépôt distant
- synchronisez votre dépôt avec le dépôt distant
- résoudre le conflit en éditant le fichier MonConflit.txt
- ajouter ce fichier dans l'index pour valider la correction
- commiter le résultat pour terminer la fusion : une nouvelle version vient d'être créée comme fusion de votre version et de la version qui était sur le dépôt distant. Propagez cette version.

Institut de Mathématiques de Bordeaux

TP 5 Créer et résoudre un conflit (solution)

```
git clone git@plmlab.math.cnrs.fr:votrelogin/MonProjet A
git clone git@plmlab.math.cnrs.fr:votrelogin/MonProjet B
cd A
( echo 111 ; echo AAA ; echo 222 ) > MonConflit.txt
git add MonConflit.txt ; git commit -m versionAAA ; git push
cd ../B
git pull
( echo 111 ; echo BBB ; echo ccc ) > MonConflit.txt
git add MonConflit.txt ; git commit -m versionBBB ; git push
cd ../A
( echo 111 ; echo aaa ; echo ccc ) > MonConflit.txt
git add MonConflit.txt ; git commit -m versionaaa ; git push
git pull ; cat MonConflit.txt
( echo 111 ; echo aBa ; echo ccc ) > MonConflit.txt
git add MonConflit.txt
git commit -m fusion
git push
```

Institut de Mathématiques de Bordeaux

Objectif 6 : Opérations complémentaires

- supprimer des fichiers,
- déplacer des fichiers,
- annuler des ajouts dans l'index (git add ...)
- annuler des modifications de fichiers
- consulter une ancienne version

supprimer des fichiers

- **git rm** `fichier1 fichier2 ...` :
  - efface les fichiers spécifiés du répertoire de travail et de l'index.
  - la nouvelle version, si on la commit, ne contiendra plus ces fichiers.
  - ... mais ces fichiers seront toujours présents dans **toutes** les précédentes versions.
    - note: "rm fichier1 fichier2 ..." : effacerait les fichiers uniquement dans le répertoire de travail, mais la suppression ne serait pas propagée dans l'index : git ne serait pas au courant de la modification (modification vu mais non enregistrée)

déplacer un fichier

- **git mv** `fichiersource fichierdestination` :

comme pour rm !

  - renomme (ou déplace) le fichier spécifié dans répertoire de travail et dans l'index.
  - la nouvelle version, si on la commit, aura également ce fichier renommé ou déplacé.
  - ... **toutes** les anciennes versions continuent à avoir le fichier non déplacé
    - note: "mv fichiersource fichierdestination" : renommerait ou déplacerait le(s) fichier(s) uniquement dans le répertoire de travail, sans que cette modification ne soit propagée dans l'index : git ne serait pas au courant du déplacement mais verrait un fichier manquant (non validé)
    - note: appliquer "mv + git rm old + git add new" sur des fichiers a pour effet de casser l'historique des modifications sur ces fichiers. ils apparaissent comme de nouveaux fichiers déconnectés des anciens.

annuler des modifications enregistrées dans l'index : **annuler un git add**

- **git reset** `-- fichier1 fichier2 ...` : les fichiers spécifiés (de l'index) sont remis dans l'état correspondant à leur précédente version enregistrée dans le dépôt.
  Typiquement, cela annule un "git add fichier1 fichier2 ..."

- **git reset** `-- répertoire` : même chose, avec un répertoire et tout son contenu (récursif)

- **git reset** : même chose avec **tout le contenu de l'index** (annule tous les git add précédents)

- note: **git status** vous rappelle les commandes pour annuler les opérations en cours

annuler un **git add Fichier$_2$** : git reset −− Fichier$_2$

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fichier$_1$ Fichier$_2$		Fichier$_1$ Fichier$_2$		Fichier$_1$ Fichier$_2$
⇓					
Modification de Fichier$_2$	Fichier$_1$ **Fichier$_2^{v2}$**		Fichier$_1$ Fichier$_2$		Fichier$_1$ Fichier$_2$
⇓					
git add Fichier$_2$	Fichier$_1$ **Fichier$_2^{v2}$**	⇒	Fichier$_1$ **Fichier$_2^{v2}$**		Fichier$_1$ Fichier$_2$
⇓					
git reset −−Fichier$_2$	Fichier$_1$ **Fichier$_2^{v2}$**		Fichier$_1$ **Fichier$_2$**	⇐	Fichier$_1$ Fichier$_2$

Institut de Mathématiques de Bordeaux

annuler des modifications dans le répertoire courant : **annuler une modification de fichier**

- **/!/ risque de perte de données /!/**

- récupération depuis l'index :
- **git checkout** `-- fichier1 fichier2 ...` : écrase et remplace les fichiers spécifiés du répertoire de travail, par les version de ces mêmes fichiers tels qu'ils existent dans **l'index** (l'index contient initialement la dernière version sauvegardées (version parente). $wd(f1, f2) <= index(f1, f2)$

- récupération depuis la dernière version :
- **git reset** `--hard` : écrase et remplace tous les fichiers "en cours" du projet (répertoire courant **et** index) en recopiant les versions du commit précédent (dernière sauvegarde) contenue dans le dépôt.
- **/!/ risque de perte de données /!/**

Institut de Mathématiques de Bordeaux

consulter une ancienne version : **git checkout [HASH]**

- **git checkout HASH** : extraire la version HASH pour l'observer
  - **git log** pour avoir la liste
- **git checkout master** : pour revenir à la dernière version
- /!/ avant de changer de version courante, votre répertoire courant doit être "propre" = ne doit pas contenir de fichier modifié (toutes les modifications doivent avoir été enregistrées/commitées)
- /!/ à ce stade, on ne peut initier des modifications qu'à partir de la dernière version
- pour en savoir plus... il faudra suivre la formation suivante !

fonctionnement des 3 zones : petite subtilité !
modification → git add → **re-modification** →git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$
⇓					
Modification de $Fichier_2$	$Fichier_1$ $\mathbf{Fichier_2^{v2}}$		$Fichier_1$ $Fichier_2$		$Fichier_1$ $Fichier_2$
⇓					
git add $Fichier_2$	$Fichier_1$ $\mathbf{Fichier_2^{v2}}$	⇒	$Fichier_1$ $\mathbf{Fichier_2^{v2}}$		$Fichier_1$ $Fichier_2$
⇓					
ReModification de $Fichier_2$	$Fichier_1$ $<\mathbf{Fichier_2^{v3}}>$		$Fichier_1$ $\mathbf{Fichier_2^{v2}}$		$Fichier_1$ $Fichier_2$
⇓					
git commit	$Fichier_1$ $<\mathbf{Fichier_2^{v3}}>$		$Fichier_1$ $\mathbf{Fichier_2^{v2}}$	⇒ ⇒	$Fichier_1$ - $(Fichier_1)$ $\mathbf{Fichier_2^{v2}}$ - $(Fichier_2)$

Institut de Mathématiques de Bordeaux

la cuisine interne de git : le sous répertoire .git

- dans le répertoire du projet, on ne voit que le répertoire de travail courant (*working directory*)
- tout le reste est caché dans le sous répertoire **.git** :
  - l'index
  - les archives du dépôt local et des dépôts distants (gros !)
  - la configuration du dépôt : .git/config (fichier texte éditable)

Raccourci '-a' (pour les utilisateurs de SVN)

- faire un commit qui prend automatiquement en compte **tous les fichiers modifiés ou effacés déjà suivis** par git :
- `git commit -am "commentaire"`

Fin

- Merci pour votre attention !
- et surtout merci pour votre feedback à venir :
  - Des éléments essentiels manquants ?
  - Des éléments pas claires ou mal dites ?
  - Des éléments trompeurs qui vous ont induit en erreur ?
- $\rightarrow$ merci de m'en faire part afin d'améliorer cette présentation :-)