Formation
Git Pour Tous
2025

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

©2025 Laurent FACQ - CNRS - IMB UMR 5251
Institut de Mathématiques de Bordeaux

Janvier 2025 v0.15

Ce document est copiable et distribuable librement et gratuitement à la condition expresse que son contenu ne soit modifié en aucune façon, et en particulier que le nom de l'auteur et de son institution d'origine continuent à y figurer, de même que le présent texte.

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 !

©2025 Laurent FACQ

à quoi sert git ?

- git permet de conserver tout l'historique des modifications de codes sources, de 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 (des milliers!)
- ... 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
- super efficace pour les fichiers textes : code source, latex, thèse, article, données textuelles gérées à la main, groupe de travail... (fonctionne par différences entre fichiers)
  - éviter les (gros) binaires, fichiers compressés, documents office
  - /!/ jamais de très gros fichiers binaires (cela peut "bloquer" un dépôt pour des problèmes de performance)/!/
    - solution annexe : `git lfs`

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 seul 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 essentielles
  - /!/ nous ne parlerons pas de la notion de branche /!/ (cf formation "Collaborative" suivante)

Prérequis : installer la "ligne de commande" git

La documentation la plus complète est à jour : `https://git-scm.com/book/en/v2/Getting-Started-Installing-Git`

Quelques cas simples :

- 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?? $<$ 10 : utilisez l'application "Git Bash"
- windows?? récent : utilise le WSL ?? (sous système Linux/Ubuntu) ou installation via `https://git-scm.com/downloads`

Prérequis : "en mode CLI" - avoir une clef SSH et connaître son mot de passe (passphrase) (1/2)

- si besoin :
  - `ls -al $HOME/.ssh` : liste les clefs déjà crées
  - `ssh-keygen` : pour générer une clef. Par défaut dans $HOME/.ssh/id_xxx (privée) et $HOME/.ssh/id_xxx.pub (publique)
    - attention à ne pas écraser l'existant si toujours utile
    - xxx: le type de clef, ex: rsa, ed52259
  - `eval $(ssh-agent -s)` : pour démarrer un agent SSH qui conserve les clefs disponibles pendant la session courante
  - `ssh-add` : pour confier au SSH Agent votre clef par défaut (sinon il faut préciser le fichier de la clef) sous forme déverrouillée
  - `ssh-add -l` : liste les clefs connues de l'agent, pour vérifier que l'agent a bien au moins une clef

ATTENTION: sous Windows - ssh-add "à vide" (i.e. sans clef existante) est silencieux (pas de message d'erreur)

Prérequis : (avec Rstudio) avoir une clef SSH et connaître son mot de passe (passphrase) (2/2)

- si besoin :
  - `ls -al $HOME/.ssh` : liste les clefs déjà crées
  - RStudio - Menu : Tools > Global Options... > Git/SVN

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

- vérifiez que vous avez les droits pour accéder à votre compte personnel sur votre "gitlab de proximité" :
  - `https://plmlab.math.cnrs.fr`
  - `https://gitlab.humanum.fr`
  - `https://gitlab.inria.fr`
  - `https://gitub.u-bordeaux.fr`
  - `https://src.koda.cnrs.fr`
  - ...
- ajouter votre clef SSH dans votre profile gitlab pour faciliter les accès sécurisés 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

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

TP Préliminaire : créer et cloner un dépôt distant de test "**MonTest**" 1/2

- créer un dépôt distant (interface web gitlab) en allant sur l'interface Web de votre serveur GIT 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 vide

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

TP Préliminaire : créer et cloner un dépôt distant de test "**MonTest**" 2/2

- 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 créé 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)
  - `git config --global push.default simple`
  - `git config --global pull.rebase false`

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

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/Login/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) sert à nommer le 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

©2025 Laurent FACQ

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 ...

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 (+ 1 distante) contenant les fichiers :
   - **répertoire de travail** :
     contient les fichiers courants de la **version courante**
   - **l'index** : contient la **future nouvelle version** en préparation
   - **le dépôt local** : archive toutes les **versions**
   - + **le dépôt distant** : permet sauvegardes et collaborations

répertoire de travail

git add FILE

=> =>

index

git commit

=> =>

dépôt local index

git push =>

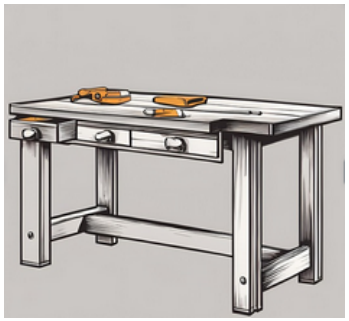
<= **git pull**

dépôt distant

©2025 Laurent FACQ

les 3 zones locales : le répertoire de travail

répertoire de travail



git add FILE ...

=> => =>

index

git commit

=> =>

dépôt local

- **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

	git add FILE ...		git commit	
		index		
répertoire de travail	=> => =>		=> =>	dépôt local

- **l'index** (*staging area*, zone de transit)
  - **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
  - en première approximation : l'index est généralement initialisé avec la dernière version sauvegardée des fichiers de votre projet

les 3 zones locales : dépôt local

	git add FILE ...		**git commit**	**dépôt local**
répertoire de travail	=> => =>	index	=> =>	

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

git commit avec ou sans **-m "commentaire ..."** ?

(rappel: **git commit** enregistre une nouvelle version du projet dans le dépôt local)

git commit sans **-m "bla bla"** ouvre un éditeur de texte !
Comment sauvegarder et sortir ?

- **vi** : `"ESC : w q ENTRÉE"` (Write Quit)
- **emacs** : `Ctrl-x Ctrl-s Ctrl-x Ctrl-c` (Save Quit)
- **Autre** : `Ctrl-s Ctrl-q` ?

git commit -m "bla bla" permet d'aller plus vite si le commentaire tient sur une ligne

Note: dans le message du commit, les lignes commençant par # sont ignorées

Terminologie: "un **commit** git" = une version complète du projet (1/2)

- 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.

Terminologie: "un **commit** git" = une version complète du projet (2/2)

- **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.
  Ex. 3 commits A,B,C successifs :

$$A \leftarrow B \leftarrow C$$

- **identifiant unique** : les commits sont référencés de manière unique par un **hash**[1] - sorte de "condensé" - élaboré avec "l'ensemble des informations contenu dans la version associée"
  - 40 caractères hexadécimaux :
    9bba17be73178a221d015c60b7ae151e71b4e059
    "abréviable" au 4 premiers si univoque

[1] signature ou empreinte numérique unique

Légende
représentation des commits et dépôts dans la suite

A- $\boxed{\text{Fichier}_1}$: un **commit** "A" constitué d'un fichier

(des fichiers dans un cadre)

B- $\boxed{\begin{array}{c}\text{Fichier}_1\\ \text{Fichier}_2\end{array}}$: un **commit** "B" constitué de deux fichiers

$\boxed{\text{B-}\ \boxed{\begin{array}{c}\text{Fichier}_1\\ \text{Fichier}_2\end{array}} \rightarrow A - \boxed{\text{Fichier}_1}}$: un **dépôt** contenant les deux commits A et B (*des cadres dans un cadre*)

Note: On représentera également le **répertoire de travail** et **l'index** comme des "pseudos" commits

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 complète $\Rightarrow$	Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 complète ⇒	Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 complète ⇒	Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 complète ⇒	Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 complète $\Rightarrow$	**Fichier$_1$**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 complète $\Rightarrow$	**Fichier$_1$**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : création de $\text{Fichier}_2 \rightarrow$ 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 $\Rightarrow$ de tout l'index	Fichier_1 **Fichier_2** $\rightarrow$ Fichier_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 ⇒ de tout l'index	Fichier$_1$ **Fichier$_2$** → Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : création de $\text{Fichier}_2 \rightarrow$ 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 $\Rightarrow$ de tout l'index	Fichier_1 **Fichier_2** $\rightarrow$ Fichier_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : création de $\text{Fichier}_2 \rightarrow$ 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 $\Rightarrow$ de tout l'index	Fichier_1 **Fichier_2** $\rightarrow$ Fichier_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 ⇒ de tout l'index	$Fichier_1$ **$Fichier_2$** → $Fichier_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : création de $\text{Fichier}_2 \rightarrow$ 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 $\Rightarrow$ de tout l'index	Fichier_1 **Fichier_2** $\rightarrow$ Fichier_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 ⇒ de tout l'index	Fichier$_1$ **Fichier$_2$** → Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 ⇒ de tout l'index	Fichier$_1$ **Fichier$_2$** → Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 ⇒ de tout l'index	Fichier$_1$ **Fichier$_2$** → Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 ⇒ de tout l'index	Fichier$_1$ **Fichier$_2$** → Fichier$_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ → $Fich_1$
⇓ Modification de $Fich_2^{v2}$	$Fich_1$ $\mathbf{Fich_2^{v2}}$		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ → $Fich_1$
⇓ git add $Fich_2^{v2}$	$Fich_1$ $\mathbf{Fich_2^{v2}}$	copie ⇒	$Fich_1$ $\mathbf{Fich_2^{v2}}$		$Fich_1$ $Fich_2$ → $Fich_1$
⇓ git commit	$Fich_1$ $\mathbf{Fich_2^{v2}}$		$Fich_1$ $\mathbf{Fich_2^{v2}}$	copie ⇒ de tout l'index	$Fich_1$ $\mathbf{Fich_2^{v2}}$ → $Fich_1$ $Fich_2$ → $Fich_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : modification de $\text{Fich}_2 \rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fich_1 Fich_2		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ Modification de Fich_2^{v2}	Fich_1 Fich_2^{v2}		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git add Fich_2^{v2}	Fich_1 Fich_2^{v2}	copie $\Rightarrow$	Fich_1 Fich_2^{v2}		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git commit	Fich_1 Fich_2^{v2}		Fich_1 Fich_2^{v2}	copie $\Rightarrow$ de tout l'index	Fich_1 Fich_2^{v2} $\rightarrow$ Fich_1 Fich_2 $\rightarrow$ Fich_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fich_1 Fich_2		Fich_1 Fich_2		Fich_1 Fich_2 → Fich_1
⇓ Modification de Fich_2^{v2}	Fich_1 **Fich_2^{v2}**		Fich_1 Fich_2		Fich_1 Fich_2 → Fich_1
⇓ git add Fich_2^{v2}	Fich_1 **Fich_2^{v2}**	copie ⇒	Fich_1 **Fich_2^{v2}**		Fich_1 Fich_2 → Fich_1
⇓ git commit	Fich_1 **Fich_2^{v2}**		Fich_1 **Fich_2^{v2}**	copie ⇒ de tout l'index	Fich_1 **Fich_2^{v2}** → Fich_1 Fich_2 → Fich_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : modification de $\text{Fich}_2 \rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fich_1 Fich_2		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ Modification de Fich_2^{v2}	Fich_1 $\textbf{Fich}_2^{v2}$		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git add Fich_2^{v2}	Fich_1 $\textbf{Fich}_2^{v2}$	copie $\Rightarrow$	Fich_1 $\textbf{Fich}_2^{v2}$		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git commit	Fich_1 $\textbf{Fich}_2^{v2}$		Fich_1 $\textbf{Fich}_2^{v2}$	copie $\Rightarrow$ de tout l'index	Fich_1 $\textbf{Fich}_2^{v2}$ $\rightarrow$ Fich_1 Fich_2 $\rightarrow$ Fich_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : modification de $\text{Fich}_2 \rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fich_1 Fich_2		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ Modification de Fich_2^{v2}	Fich_1 $\textbf{Fich}_2^{v2}$		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git add Fich_2^{v2}	Fich_1 $\textbf{Fich}_2^{v2}$	copie $\Rightarrow$	Fich_1 $\textbf{Fich}_2^{v2}$		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git commit	Fich_1 $\textbf{Fich}_2^{v2}$		Fich_1 $\textbf{Fich}_2^{v2}$	copie $\Rightarrow$ de tout l'index	Fich_1 $\textbf{Fich}_2^{v2}$ $\rightarrow$ Fich_1 Fich_2 $\rightarrow$ Fich_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ → $Fich_1$
⇓ Modification de $Fich_2^{v2}$	$Fich_1$ **$Fich_2^{v2}$**		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ → $Fich_1$
⇓ git add $Fich_2^{v2}$	$Fich_1$ **$Fich_2^{v2}$**	copie ⇒	$Fich_1$ **$Fich_2^{v2}$**		$Fich_1$ $Fich_2$ → $Fich_1$
⇓ git commit	$Fich_1$ **$Fich_2^{v2}$**		$Fich_1$ **$Fich_2^{v2}$**	copie ⇒ de tout l'index	$Fich_1$ **$Fich_2^{v2}$** → $Fich_1$ $Fich_2$ → $Fich_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : modification de $\text{Fich}_2 \rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fich_1 Fich_2		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ Modification de Fich_2^{v2}	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git add Fich_2^{v2}	Fich_1 $\mathbf{Fich}_2^{v2}$	copie $\Rightarrow$	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git commit	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 $\mathbf{Fich}_2^{v2}$	copie $\Rightarrow$ de tout l'index	Fich_1 $\mathbf{Fich}_2^{v2}$ $\rightarrow$ Fich_1 Fich_2 $\rightarrow$ Fich_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : modification de $\text{Fich}_2 \rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fich_1 Fich_2		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ Modification de Fich_2^{v2}	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git add Fich_2^{v2}	Fich_1 $\mathbf{Fich}_2^{v2}$	copie $\Rightarrow$	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git commit	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 $\mathbf{Fich}_2^{v2}$	copie $\Rightarrow$ de tout l'index	Fich_1 $\mathbf{Fich}_2^{v2}$ $\rightarrow$ Fich_1 Fich_2 $\rightarrow$ Fich_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : modification de $\text{Fich}_2 \rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fich_1 Fich_2		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ Modification de Fich_2^{v2}	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git add Fich_2^{v2}	Fich_1 $\mathbf{Fich}_2^{v2}$	copie $\Rightarrow$	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git commit	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 $\mathbf{Fich}_2^{v2}$	copie $\Rightarrow$ de tout l'index	Fich_1 $\mathbf{Fich}_2^{v2}$ $\rightarrow$ Fich_1 Fich_2 $\rightarrow$ Fich_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : modification de $\text{Fich}_2 \rightarrow$ git add $\rightarrow$ git commit

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	Fich_1 Fich_2		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ Modification de Fich_2^{v2}	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 Fich_2		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git add Fich_2^{v2}	Fich_1 $\mathbf{Fich}_2^{v2}$	copie $\Rightarrow$	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 Fich_2 $\rightarrow$ Fich_1
$\Downarrow$ git commit	Fich_1 $\mathbf{Fich}_2^{v2}$		Fich_1 $\mathbf{Fich}_2^{v2}$	copie $\Rightarrow$ de tout l'index	Fich_1 $\mathbf{Fich}_2^{v2}$ $\rightarrow$ Fich_1 Fich_2 $\rightarrow$ Fich_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Quelques précisions sur **git add** et **git commit**

- **git add** répertoire : ajoute tout le contenu (réccursivement)
  !! un répertoire vide ne sera pas pris en compte !!

- **git commit ...** : si pas de modification ajoutée au préalable avec **git add**, **git commit** ne fait rien ! - juste un petit message d'erreur.
  !! il est courant de ne pas y faire attention et de ne plus comprendre ensuite dans quel état on se trouve. C'est ce qui se produit quand on oubli de faire un **git add**

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...

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 y 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"
```


©2025 Laurent FACQ

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

©2025 Laurent FACQ

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

```
lfacq@scratchy2:/tmp/toto$ echo hello > file.txt
lfacq@scratchy2:/tmp/toto$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file.txt

nothing added to commit but untracked files present (use "git add" to track)
lfacq@scratchy2:/tmp/toto$
```

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

```
lfacq@scratchy2:/tmp/toto$ echo hello > file.txt
lfacq@scratchy2:/tmp/toto$ git add file.txt
lfacq@scratchy2:/tmp/toto$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file.txt

lfacq@scratchy2:/tmp/toto$
```

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

```
lfacq@scratchy2:/tmp/toto$ echo hello > file.txt
lfacq@scratchy2:/tmp/toto$ git add file.txt
lfacq@scratchy2:/tmp/toto$ echo salut > file.txt
lfacq@scratchy2:/tmp/toto$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file.txt

lfacq@scratchy2:/tmp/toto$
```

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ Modification $Fich_2$	$Fich_1$ $\mathbf{Fich_2^{v2}}$	modif non ← validée	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git add $Fich_2^{v2}$	$Fich_1$ $Fich_2^{v2}$	copie ⇒	$Fich_1$ $\mathbf{Fich_2^{v2}}$	modif qui sera ←validée	$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git commit	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$	copie ⇒	$\mathbf{Fich_1}$ $\mathbf{Fich_2^{v2}}$ - $Fich_1$ $Fich_2$ - $Fich_1$
⇓ Création $Fich_3$	$Fich_1$ $Fich_2^{v2}$ $\mathbf{Fich_3}$	non ←suivi	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$ - $Fich_1$ $Fich_2$ - $Fich_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓					
Modification $Fich_2$	$Fich_1$ $\mathbf{Fich_2^{v2}}$	modif non ← validée	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git add $Fich_2^{v2}$	$Fich_1$ $Fich_2^{v2}$	copie ⇒	$Fich_1$ $\mathbf{Fich_2^{v2}}$	modif qui sera ←validée	$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git commit	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$	copie ⇒	$\mathbf{Fich_1}$ $\mathbf{Fich_2^{v2}}$ - $Fich_1$ $Fich_2$ - $Fich_1$
⇓ Création $Fich_3$	$Fich_1$ $Fich_2^{v2}$ $\mathbf{Fich_3}$	non ←suivi	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$ - $Fich_1$ $Fich_2$ - $Fich_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ **Modification** $Fich_2$	$Fich_1$ $\mathbf{Fich_2^{v2}}$	modif non ← validée	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git add $Fich_2^{v2}$	$Fich_1$ $Fich_2^{v2}$	copie ⇒	$Fich_1$ $\mathbf{Fich_2^{v2}}$	modif qui sera ←validée	$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git commit	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$	copie ⇒	$\mathbf{Fich_1}$ $\mathbf{Fich_2^{v2}}$ - $Fich_1$ $Fich_2$ - $Fich_1$
⇓ Création $Fich_3$	$Fich_1$ $Fich_2^{v2}$ $\mathbf{Fich_3}$	non ←suivi	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$ - $Fich_1$ $Fich_2$ - $Fich_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ Modification $Fich_2$	$Fich_1$ **$Fich_2^{v2}$**	modif non ← validée	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git add $Fich_2^{v2}$	$Fich_1$ $Fich_2^{v2}$	copie ⇒	$Fich_1$ **$Fich_2^{v2}$**	modif qui sera ←validée	$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git commit	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$	copie ⇒	**$Fich_1$** **$Fich_2^{v2}$** - $Fich_1$ $Fich_2$ - $Fich_1$
⇓ Création $Fich_3$	$Fich_1$ $Fich_2^{v2}$ **$Fich_3$**	non ←suivi	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$ - $Fich_1$ $Fich_2$ - $Fich_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 3 zones locales : git status ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ Modification $Fich_2$	$Fich_1$ **$Fich_2^{v2}$**	modif non ← validée	$Fich_1$ $Fich_2$		$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git add $Fich_2^{v2}$	$Fich_1$ $Fich_2^{v2}$	copie ⇒	$Fich_1$ **$Fich_2^{v2}$**	modif qui sera ←validée	$Fich_1$ $Fich_2$ - $Fich_1$
⇓ git commit	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$	copie ⇒	**$Fich_1$** **$Fich_2^{v2}$** - $Fich_1$ $Fich_2$ - $Fich_1$
⇓ Création **$Fich_3$**	$Fich_1$ $Fich_2^{v2}$ **$Fich_3$**	non ←suivi	$Fich_1$ $Fich_2^{v2}$		$Fich_1$ $Fich_2^{v2}$ - $Fich_1$ $Fich_2$ - $Fich_1$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

TP 2.1 : Prédire **git status**

- reprendre les étapes du corrigé du TP 1
- prédire le résultat d'un "git status" executé après chaque commande echo, git add ou git commit (ModifiedOK, Modified, Untracked)

```
echo "Laurent" > Invitations.txt ; git status ??
git add Invitations.txt ; git status ??
git commit -m "invitation lolo"  ; git status ??
echo "Montre" > IdeesCadeaux.txt ; git status ??
git add IdeesCadeaux.txt  ; git status ??
git commit -m "premieres idees"  ; git status ??
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 "compléments invitations + idees" ; git status ??
```

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

```
# 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 # Modified 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

©2025 Laurent FACQ

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

```
lfacq@scratchy2:/tmp/log$ echo aaa > aaa.txt
lfacq@scratchy2:/tmp/log$ git add aaa.txt ; git commit -m aaa
[master (root-commit) 41088e5] aaa
 1 file changed, 1 insertion(+)
 create mode 100644 aaa.txt
lfacq@scratchy2:/tmp/log$ echo bbb > bbb.txt
lfacq@scratchy2:/tmp/log$ git add bbb.txt ; git commit -m bbb
[master a840855] bbb
 1 file changed, 1 insertion(+)
 create mode 100644 bbb.txt
lfacq@scratchy2:/tmp/log$ git log
commit a84085521a584a9d4c817f32f699cb2a07c56d38 (HEAD -> master)
Author: Laurent Facq <laurent.facq@math.u-bordeaux.fr>
Date:   Mon May 30 11:59:35 2022 +0200

    bbb

commit 41088e564f8c5ca7665f7ac76a6f1ff06855f15b
Author: Laurent Facq <laurent.facq@math.u-bordeaux.fr>
Date:   Mon May 30 11:59:19 2022 +0200

    aaa
```

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-( )
$\Downarrow$				Versions	HASHY - HASHZ
Modification	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Fichier_2	$\textbf{Fichier}_2^{v2}$		Fichier_2		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
git add	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Fichier_2^{v2}	$\textbf{Fichier}_2^{v2}$	$\Rightarrow$	$\textbf{Fichier}_2^{v2}$		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
git	Fichier_1		Fichier_1	$\Rightarrow$	Fichier_1 -(Fichier_1)-(Fichier
commit	$\textbf{Fichier}_2^{v2}$		$\textbf{Fichier}_2^{v2}$	$\Rightarrow$	$\textbf{Fichier}_2^{v2}$-(Fichier_2)-(
				Versions	HASHX - HASHY - HASH

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Initial	Fichier_2		Fichier_2		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
Modification	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Fichier_2	$\textbf{Fichier}_2^{v2}$		Fichier_2		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
git add	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Fichier_2^{v2}	$\textbf{Fichier}_2^{v2}$	$\Rightarrow$	$\textbf{Fichier}_2^{v2}$		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
git	Fichier_1		Fichier_1	$\Rightarrow$	Fichier_1 -(Fichier_1)-(Fichier
commit	$\textbf{Fichier}_2^{v2}$		$\textbf{Fichier}_2^{v2}$	$\Rightarrow$	$\textbf{Fichier}_2^{v2}$-(Fichier_2)-(
				Versions	HASHX - HASHY - HASH

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Initial	Fichier_2		Fichier_2		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
Modification	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Fichier_2	$\textbf{Fichier}_2^{v2}$		Fichier_2		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
git add	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Fichier_2^{v2}	$\textbf{Fichier}_2^{v2}$	$\Rightarrow$	$\textbf{Fichier}_2^{v2}$		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
git	Fichier_1		Fichier_1	$\Rightarrow$	Fichier_1 -(Fichier_1)-(Fichier
commit	$\textbf{Fichier}_2^{v2}$		$\textbf{Fichier}_2^{v2}$	$\Rightarrow$	$\textbf{Fichier}_2^{v2}$-(Fichier_2)-(
				Versions	HASHX - HASHY - HASH

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Initial	Fichier_2		Fichier_2		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
Modification	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Fichier_2	$\textbf{Fichier}_2^{v2}$		Fichier_2		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
git add	Fichier_1		Fichier_1		Fichier_1-(Fichier_1)
Fichier_2^{v2}	$\textbf{Fichier}_2^{v2}$	$\Rightarrow$	$\textbf{Fichier}_2^{v2}$		Fichier_2-( )
$\Downarrow$				Versions	HASHY - HASHZ
git	Fichier_1		Fichier_1	$\Rightarrow$	Fichier_1 -(Fichier_1)-(Fichier
commit	$\textbf{Fichier}_2^{v2}$		$\textbf{Fichier}_2^{v2}$	$\Rightarrow$	$\textbf{Fichier}_2^{v2}$-(Fichier_2)-(
				Versions	HASHX - HASHY - HASH

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

TP 2.2 (optionnel) : 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

©2025 Laurent FACQ

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
```

git diff ... : afficher les différences entre versions

- **git diff** : affiche les différences entre **l'index** et **le répertoire de travail** c.a.d
  vos modifications en cours non encore ajoutées à l'index = *en partant de l'index, que faut il modifier pour aboutir au contenu du répertoire de travail*
- **git diff SRCHASH** : affiche les différences entre la version **SRCHASH** et celle du **répertoire de travail** = *en partant de la version SRCHASH, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH DSTHASH** : affiche les différences entre les versions **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

©2025 Laurent FACQ

git diff ... : afficher les différences entre versions

- **git diff** : affiche les différences entre **l'index** et **le répertoire de travail** c.a.d
  vos modifications en cours non encore ajoutées à l'index = *en partant de l'index, que faut il modifier pour aboutir au contenu du répertoire de travail*
- **git diff SRCHASH** : affiche les différences entre la version **SRCHASH** et celle du **répertoire de travail** = *en partant de la version SRCHASH, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH DSTHASH** : affiche les différences entre les versions **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

©2025 Laurent FACQ

git diff ... : afficher les différences entre versions

- **git diff** : affiche les différences entre **l'index** et **le répertoire de travail** c.a.d
  vos modifications en cours non encore ajoutées à l'index = *en partant de l'index, que faut il modifier pour aboutir au contenu du répertoire de travail*
- **git diff SRCHASH** : affiche les différences entre la version **SRCHASH** et celle du **répertoire de travail** = *en partant de la version SRCHASH, que faut il modifier pour aboutir à celle du répertoire de travail*
- **git diff SRCHASH DSTHASH** : affiche les différences entre les versions **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> ] ]
```

fonctionnement des 3 zones locales : git diff ?

Actions	Répertoire de Travail		Index (staging area)		Dépôt Local
Etat Initial	$Fichier_1$		$Fichier_1$		$Fichier_1$-($Fichier_1$)
	$Fichier_2$		$Fichier_2$		$Fichier_2$-( )
$\Downarrow$					
Modification	$Fichier_1$	git	$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2$	$\mathbf{Fichier}_2^{v2}$	$\leftarrow$diff$\leftarrow$	$Fichier_2$		$Fichier_2$-( )
$\Downarrow$					
git add	$Fichier_1$		$Fichier_1$		$Fichier_1$-($Fichier_1$)
$Fichier_2^{v2}$	$\mathbf{Fichier}_2^{v2}$	$\Rightarrow$	$\mathbf{Fichier}_2^{v2}$		$Fichier_2$-( )
$\Downarrow$					
git	$Fichier_1$		$Fichier_1$	$\Rightarrow$	$Fichier_1$ -($Fichier_1$)-($Fichier_1$)
commit	$\mathbf{Fichier}_2^{v2}$		$\mathbf{Fichier}_2^{v2}$	$\Rightarrow$	$\mathbf{Fichier}_2^{v2}$-($Fichier_2$)-( )
					Versions HASHX - HASHY - HASHZ

git diff HASHZ HASHY ?

git diff ... : afficher les différences entre versions

```
lfacq@scratchy2:/tmp/diff$ echo bonjour > ok.txt
lfacq@scratchy2:/tmp/diff$ git add ok.txt
lfacq@scratchy2:/tmp/diff$ echo Monsieur >> ok.txt
lfacq@scratchy2:/tmp/diff$ git diff
diff --git a/ok.txt b/ok.txt
index 1cd909e..fc5c28d 100644
--- a/ok.txt
+++ b/ok.txt
@@ -1 +1,2 @@
 bonjour
+Monsieur
lfacq@scratchy2:/tmp/diff$
```

git diff ... : afficher les différences entre versions

```
lfacq@scratchy2:/tmp/diff$ echo bonjour > ok.txt
lfacq@scratchy2:/tmp/diff$ git add ok.txt
lfacq@scratchy2:/tmp/diff$ echo Monsieur >> ok.txt
lfacq@scratchy2:/tmp/diff$ git diff
diff --git a/ok.txt b/ok.txt
index 1cd909e..fc5c28d 100644
--- a/ok.txt
+++ b/ok.txt
@@ -1 +1,2 @@
 bonjour
+Monsieur
lfacq@scratchy2:/tmp/diff$ echo bye > ok.txt
lfacq@scratchy2:/tmp/diff$ git diff
diff --git a/ok.txt b/ok.txt
index 1cd909e..b023018 100644
--- a/ok.txt
+++ b/ok.txt
@@ -1 +1 @@
-bonjour
+bye
```

TP 2.3 (optionnel) : 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

TP 2.4 : Devinettes **git status/log/diff**

Prédire le résultat des commandes ";#??"

```
# Depuis un dépot "propre" (toutes modifications enregistrées)
   echo "Quiz" > Games.txt
   echo "Belotte" >> Games.txt
   git status ;#??
   git add Games.txt
   echo "OufBall" >> Games.txt
   echo "Ordures" > Trash.txt
   git status ;#??
   git diff ;#??
   git log ;#??
   echo "Banane" > Trash.txt
   git commit -m "game over"
   git diff ;#??
   echo "Bridge" > Games.txt  # (note: avec ">" on remplace Games.txt)
   git diff ;#??
   git log ;#??
```

TP 2.4 : Devinettes **git status/log/diff** (solution)

Prédire le résultat des commandes "??" (Untracked,Modified ModifiedOK, différences, évolution des commits)

```
# Depuis un dépot "propre" (toutes modifications enregistrées)
   echo "Quiz" > Games.txt
   echo "Belotte" >> Games.txt
   git status ;#?? Games.txt Untracked
   git add Games.txt
   echo "OufBall" >> Games.txt
   echo "Ordures" > Trash.txt
   git status ;#?? Games.txt: Modified et ModifiedOK Trash.txt: Untrack
   git diff ;#?? +OufBall à la fin de Games.txt
   git log ;#?? pas de changement
   echo "Banane" > Trash.txt
   git commit -m "game over"
   git diff ;#?? +OufBall à la fin de Games.txt
   echo "Bridge" > Games.txt  # (note: avec ">" on remplace Games.txt)
   git diff ;#?? Games.txt: -Quiz/-Belotte/+Bridge
   git log ;#?? un commit de plus !
```

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**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 4 zones : **git clone & git push** ?

Actions	Rép. Travail	Index	Dépôt Local	Dépôt Distant
Etat Initial	(vide)	(vide)	(vide)	F_1 F_2 → F_1
⇓ git clone ...	F_1 F_2 ⇐	F_1 F_2 ⇐	F_1 F_2 → F_1	⇐ F_1 F_2 → F_1
⇓ modif F_2 add F_2^{v2}	F_1 F_2^{v2} →	$F1$ F_2^{v2}	F_1 F_2 → F_1	F_1 F_2 → F_1
⇓ git commit	F_1 F_2^{v2}	F_1 F_2^{v2} →	F_1 F_2^{v2} → F_1 F_2 → F_1	F_1 F_2 → F_1
⇓ git push	F_1 F_2^{v2}	F_1 F_2^{v2}	F_1 F_2^{v2} → F_1 F_2 → F_1	→ F_1 F_2^{v2} → F_1 F_2 → F_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 4 zones : git clone & git push ?

Actions	Rép. Travail	Index	Dépôt Local	Dépôt Distant
Etat Initial	(vide)	(vide)	(vide)	F_1 F_2 → F_1
⇓ git clone ...	F_1 F_2 ⇐	F_1 F_2 ⇐	F_1 F_2 → F_1 ⇐	F_1 F_2 → F_1
⇓ modif F_2 add F_2^{v2}	F_1 F_2^{v2} →	$F1$ F_2^{v2}	F_1 F_2 → F_1	F_1 F_2 → F_1
⇓ git commit	F_1 F_2^{v2}	F_1 F_2^{v2} →	F_1 F_2^{v2} → F_1 F_2 → F_1	F_1 F_2 → F_1
⇓ git push	F_1 F_2^{v2}	F_1 F_2^{v2}	F_1 F_2^{v2} → F_1 F_2 → F_1 →	F_1 F_2^{v2} → F_1 F_2 → F_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 4 zones : git clone & git push ?

Actions	Rép. Travail	Index	Dépôt Local	Dépôt Distant
Etat Initial	(vide)	(vide)	(vide)	F_1 F_2 → F_1
⇓ git clone ...	F_1 F_2 ⇐	F_1 F_2 ⇐	F_1 F_2 → F_1 ⇐	F_1 F_2 → F_1
⇓ modif F_2 add F_2^{v2}	F_1 F_2^{v2} →	$F1$ F_2^{v2}	F_1 F_2 → F_1	F_1 F_2 → F_1
⇓ git commit	F_1 F_2^{v2}	F_1 F_2^{v2} →	F_1 F_2^{v2} → F_1 F_2 → F_1	F_1 F_2 → F_1
⇓ git push	F_1 F_2^{v2}	F_1 F_2^{v2}	F_1 F_2^{v2} → F_1 F_2 → F_1 →	F_1 F_2^{v2} → F_1 F_2 → F_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

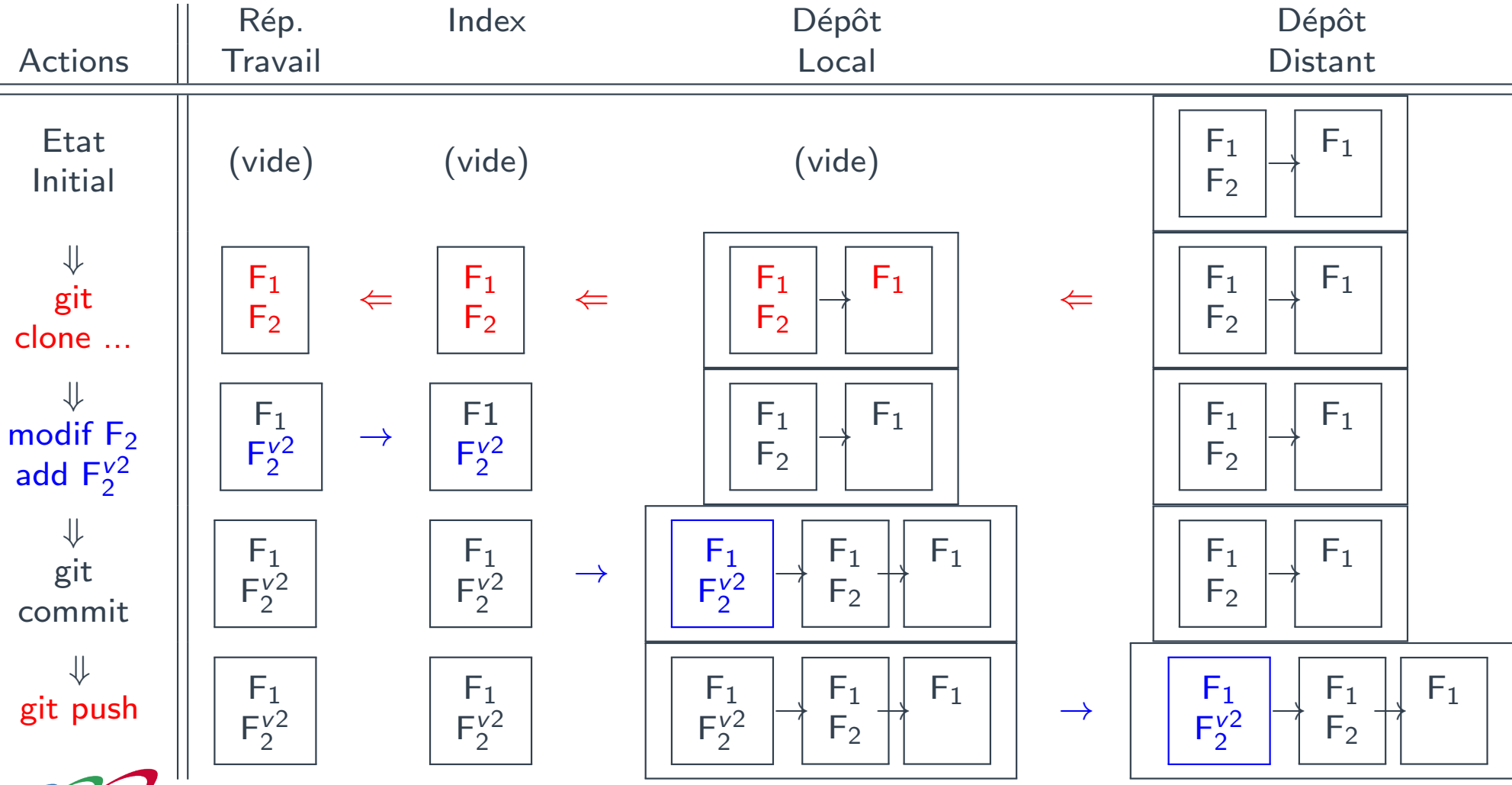
fonctionnement des 4 zones : git clone & git push ?

Actions	Rép. Travail	Index	Dépôt Local	Dépôt Distant
Etat Initial	(vide)	(vide)	(vide)	F_1 F_2 → F_1
⇓ git clone ...	F_1 F_2 ⇐	F_1 F_2 ⇐	F_1 F_2 → F_1 ⇐	F_1 F_2 → F_1
⇓ modif F_2 add F_2^{v2}	F_1 F_2^{v2} →	$F1$ F_2^{v2}	F_1 F_2 → F_1	F_1 F_2 → F_1
⇓ git commit	F_1 F_2^{v2}	F_1 F_2^{v2} →	F_1 F_2^{v2} → F_1 F_2 → F_1	F_1 F_2 → F_1
⇓ git push	F_1 F_2^{v2}	F_1 F_2^{v2}	F_1 F_2^{v2} → F_1 F_2 → F_1	→ F_1 F_2^{v2} → F_1 F_2 → F_1

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

fonctionnement des 4 zones : git clone & git push ?



Zoom sur git push

cf transparent précédent

git push synchronise notre dépôt local **vers** le dépot distant : il recopie depuis le dépot local ce qui manque (ce qui n'existe pas encore) sur le dépot distant

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

- dans votre dépot de test
- **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)

```
# 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 cadeau 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

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs Synchronisation avec **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**
  - $\Rightarrow$ la dernière version est sur le dépôt distant
- **depuis un autre ordinateur B : cloner ce dépôt sur cet 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**
  - $\Rightarrow$ la dernière version est sur le dépôt distant
- **/!/ 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 - en cas de panique : effacer/re-cloner ce dépôt dans un autre répertoire... par très efficace mais utile au début si on est un peu perdu !*

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Objectif 4 : Travailler depuis 2 ordinateurs
Synchronisation avec **git pull**

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine**
git add ...	0'		0	version d'origine
git commit -m A	A		0	version d'origine
git push	A		0	**version A**
	A	**git pull**	A	version A
	A	git add ...	A'	version A
	A	git commit -m B	B	version A
	A	git push	B	**version B**
git pull	B		B	version B
git add ...	B'		B	version B
git commit -m A2	A2		B	version B
git push	A2		B	**version A2**

Zoom sur git pull : dépôt local <= *pull* <= dépôt distant

répertoire de travail	**git add FILE**	**index**	=> =>	**dépôt local**	**git push =>**	**dépôt distant**
	=> =>		**git commit**		**<= git pull**	

git pull fait deux choses:

- il synchronise le **dépôt local à partir** du dépôt distant : il récupère du dépôt distant tout ce qui n'est pas déjà sur le dépôt local et le recopie sur le dépôt local.

  (ce qui suit est approximatif, ce sera précisé dans le prochain cours / module git)

- et il met à jour **l'index** et le **répertoire de travail** avec la dernière version du projet récupérée sur le dépôt distant. De manière similaire au **git clone** initial

cela se fait de manière transparente si le projet n'a pas évolué localement entre temps, depuis le dernier push... ou nécessite une fusion (avec potentiellement une résolution de conflits) s'il y a eu de nouvelles modifications locales.

TP 4 : Travailler avec un dépôt distant depuis 2 ordinateurs

Sur l'ordinateur B

- astuce : pour simuler ce qui se passerait avec deux ordinateurs, nous allons cloner le dépôt "MonAnniv" dans un 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
- visualisez le contenu du fichier Invitations.txt

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

Les fusions et les conflits

- lors d'un **git pull**, si des modifications ont eu lieu sur le dépôt distant depuis notre dernier pull/clone, git est amené à effectuer une **fusion** des modifications réalisées en local et à distance
- la plus part des fusions sont gérées automatiquement par git
- fusion automatique impossible ? => conflit :
  - la modification "**simultanée**"
  - de la **même zone** d'un **même fichier**
  - engendre un conflit que git ne peut pas résoudre tout seul car il y a un/des choix à faire
  - cela se produit presque inévitablement :
    - quand on travail à plusieurs
    - quand on travail depuis plusieurs ordinateurs
  - il suffit alors de "dire à git" ce que l'on souhaite
    - éditer le fichier en conflit
    - puis faire un **git add** dessus pour qu'il soit pris en compte

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

- vous avez synchronisé vos deux ordinateurs (A et B) avec la dernière version de votre dépôt distant
- **depuis votre ordinateur A**, vous avez fait des modifications (modifications+commits) mais vous n'avez rien propagé sur le dépôt distant
- **depuis l'ordinateur B** vous faites maintenant des modifications (modifications+commits) 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** affiche 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 :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

sur ordi A :	v°	sur ordi B :	v°	dépôt distant
git clone	0	git clone	0	**version d'origine 0**
git add ...	0'		0	version d'origine 0
git commit -m A	A		0	version d'origine 0
((oubli git push))	A		0	**version d'origine 0**
	A	**git pull**	0	version d'origine 0
	A	git add ...	0''	version d'origine 0
	A	git commit -m B	B	version d'origine 0
	A	git push	B	**version B**
/!/ git push /!/ pb - refus	A		B	version B
git pull ((fusion))	A?B		B	version B
((résolution conflit))	A?B		B	version B
git add ...	A+B		B	version B
git commit -m A+B	A+B		B	version B
git push	A+B		B	**version A+B**

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

git pull - Comparaison des deux cas : sans et avec fusion

- Sans fusion / travail toujours sur la dernière version :

- Avec fusion/ deux modifications en partant de la version d'origine :

git pull - Comparaison des deux cas : sans et avec fusion

- Sans fusion / travail toujours sur la dernière version :

- Avec fusion/ deux modifications en partant de la version d'origine :

git pull - Comparaison des deux cas : sans et avec fusion

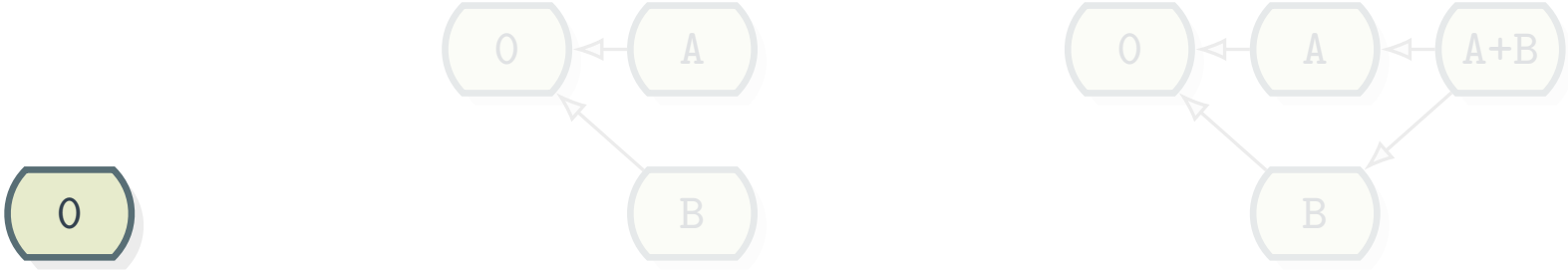
- Sans fusion / travail toujours sur la dernière version :

- Avec fusion/ deux modifications en partant de la version d'origine :

git pull - Comparaison des deux cas : sans et avec fusion

- Sans fusion / travail toujours sur la dernière version :

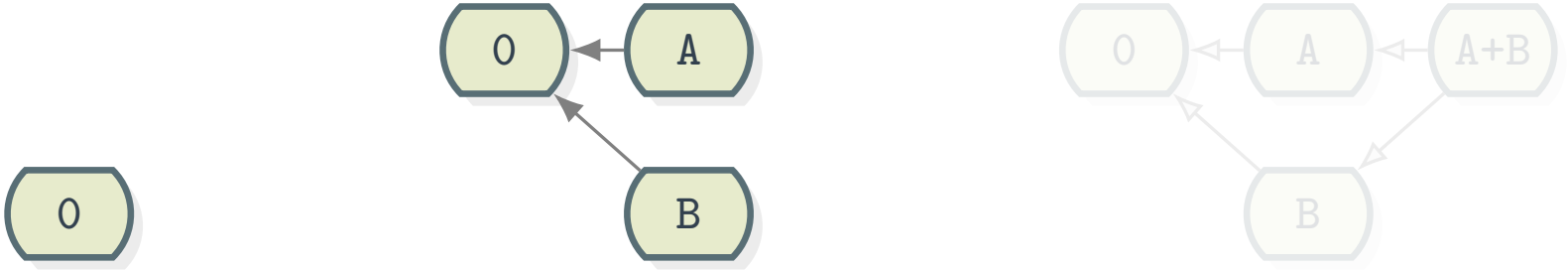
- Avec fusion/ deux modifications en partant de la version d'origine :



git pull - Comparaison des deux cas : sans et avec fusion

- Sans fusion / travail toujours sur la dernière version :

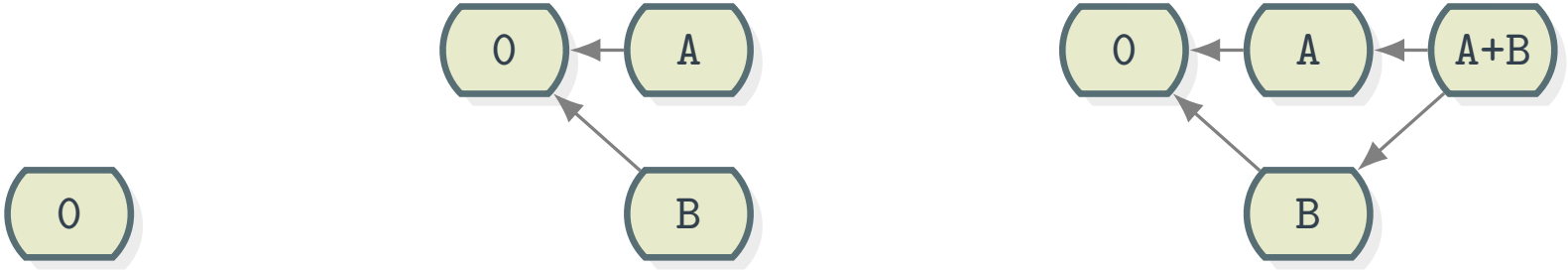
- Avec fusion/ deux modifications en partant de la version d'origine :



git pull - Comparaison des deux cas : sans et avec fusion

- Sans fusion / travail toujours sur la dernière version :

- Avec fusion/ deux modifications en partant de la version d'origine :



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 : (workdir)	fichier B : (distant)	conflit à résoudre : (fichier à éditer)	fichier final
blabla versionA blabla	blabla versionB blabla	blabla <<<<<<< HEAD versionA ======= versionB >>>>>> HASHBHASHB blabla	blabla versionAB 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 3 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 !

Résoudre les conflits de manière conviviale ?

A savoir ! (TODO)

`git mergetool fichiers ...`

- au lieu d'utiliser un simple éditeur
- permet de lancer un outil graphique dédié à la gestion de conflits

TODO proposer quelques outils !
exemple: **meld**

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** sans synchroniser
  - modifiez différemment la deuxième ligne du fichier
  - tentez de propager vos modifications sur le dépôt distant
  - synchronisez votre dépôt local 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.

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 222 ) > MonConflit.txt
git add MonConflit.txt ; git commit -m versionBBB ; git push
cd ../A
( echo 111 ; echo aaa ; echo 222 ) > MonConflit.txt
git add MonConflit.txt ; git commit -m versionaaa ; git push
git pull ; cat MonConflit.txt
( echo 111 ; echo aBa ; echo 222 ) > MonConflit.txt
git add MonConflit.txt
git commit -m fusion
git push
```

Objectif 6 : Opérations complémentaires

- supprimer des fichiers,
- déplacer (renommer) des fichiers,
- annuler des ajouts dans l'index (git add ...)
- annuler des modifications de fichiers
- consulter une ancienne version
- afficher l'URL du dépôt distant
- raccourci "à la SVN" (commit -a)
- petite subtilité / re-modification après mise dans l'index (git add ... + re-modificaton)

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 ou renommer un fichier (ou un répertoire)

- **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 avec le nom/chemin d'origine
    - 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

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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$_2$
$\Downarrow$					
Modification de Fichier$_2$	Fichier$_1$ **Fichier**$_2^{v2}$		Fichier$_1$ Fichier$_2$		Fichier$_1$ Fichier$_2$
$\Downarrow$					
git add Fichier$_2$	Fichier$_1$ **Fichier**$_2^{v2}$	$\Rightarrow$	Fichier$_1$ **Fichier**$_2^{v2}$		Fichier$_1$ Fichier$_2$
$\Downarrow$					
git reset −− Fichier$_2$	Fichier$_1$ **Fichier**$_2^{v2}$		Fichier$_1$ **Fichier**$_2$	$\Leftarrow$	Fichier$_1$ Fichier$_2$

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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

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

- récupération depuis l'index : (expérimental: **git restore**)
- **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

©2025 Laurent FACQ

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** ou **git checkout main** : pour revenir à la dernière version (**git branch** pour connaître le nom exacte à utiliser)
- /!/ 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) (approximation à ce stade)
- /!/ 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 !

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

- affiche la liste des dépôts distants connectés à votre dépôt local
- Format: *nom-court URL (direction)*
- /!/ on peut en avoir plusieurs /!/
- /!/ et même selon la direction (fetch ou push) /!/
- par défaut : **origin** dans les deux directions

```
$ git remote  -v
origin git@gitlab.inria.fr:equipe/projet.git (fetch)
origin git@gitlab.inria.fr:equipe/projet.git (push)
```

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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$_1$ **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$_2$
⇓					
ReModification de Fichier$_2$	Fichier$_1$ $<$**Fichier$_2^{v3}$** $>$		Fichier$_1$ **Fichier$_2^{v2}$**		Fichier$_1$ Fichier$_2$
⇓					
git commit	Fichier$_1$ $<$**Fichier$_2^{v3}$** $>$		Fichier$_1$ **Fichier$_2^{v2}$**	⇒ ⇒	Fichier$_1$ - (Fichier$_1$) **Fichier$_2^{v2}$** - (Fichier$_2$)

Institut de Mathématiques de Bordeaux

©2025 Laurent FACQ

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 clairs ou mal expliqués ?
  - Des éléments trompeurs qui vous ont induit en erreur ?
- $\rightarrow$ merci de m'en faire part afin d'améliorer cette présentation :-)

- prochaine module : Git collaboratif (branches)
- en attendant : pratiquez ! pratiquez ! pratiquez !